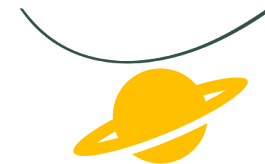


人工智能科普班(十七)





今日目標

- 1 : 認識線性迴歸
- 2 : 認識多項式迴歸
- 3 : 認識多重/多元迴歸



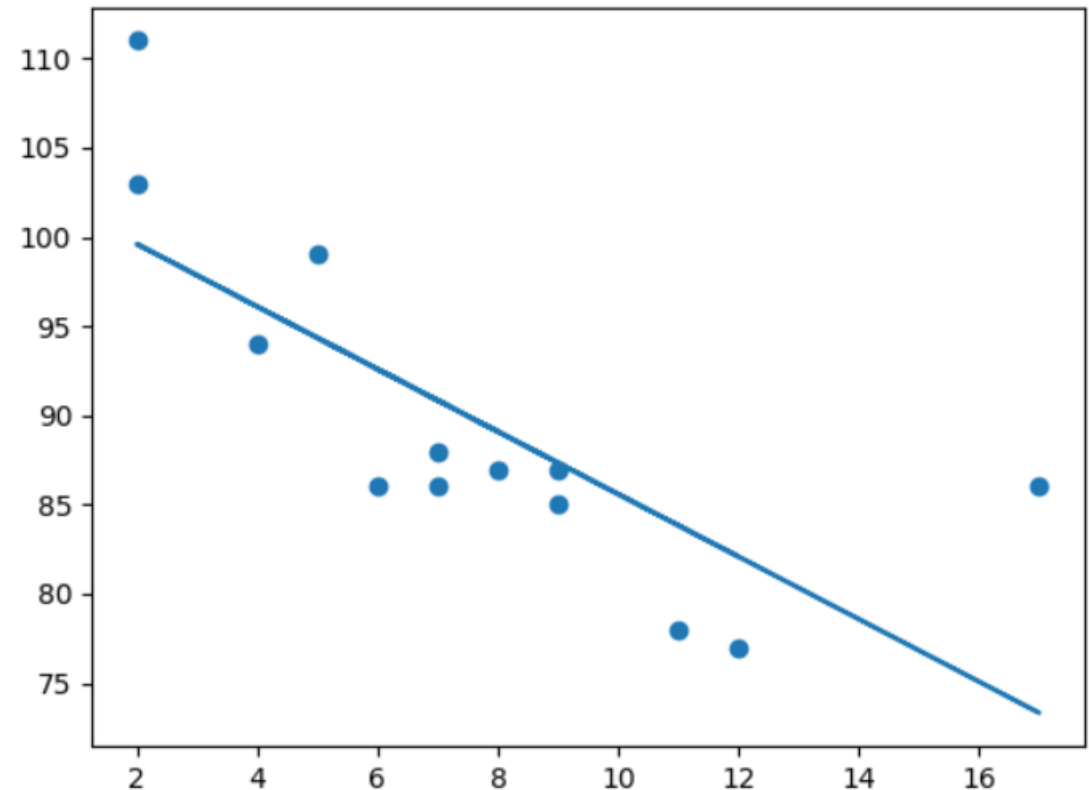
迴歸 Regression

迴歸

當您嘗試尋找變數之間的關係時，會使用術語迴歸。
在機器學習和統計建模中，這種關係用於預測未來事件的結果。

線性迴歸 (Linear Regression)

線性迴歸利用資料點之間的關係來繪製一條穿過所有資料點的直線。這條線可用於預測未來值。



什麼時候適合使用線性迴歸? (Linear Regression)

了解 x 軸值和 y 軸值之間的關係非常重要，如果沒有關係，則線性迴歸不能用於預測任何內容。

這種關係——相關係數——稱為 r 。

值 r 範圍從 -1 到 1 ，其中 0 表示沒有關係， 1 （和 -1 ）表示 100% 相關。

Python 和 **Scipy** 模組將為您計算這個值，您所要做的就是向它提供 x 和 y 值。

什麼時候適合使用線性迴歸? (Linear Regression)

```
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

k,b,r,p,std = stats.linregress(x, y)

print("r :",r)
```

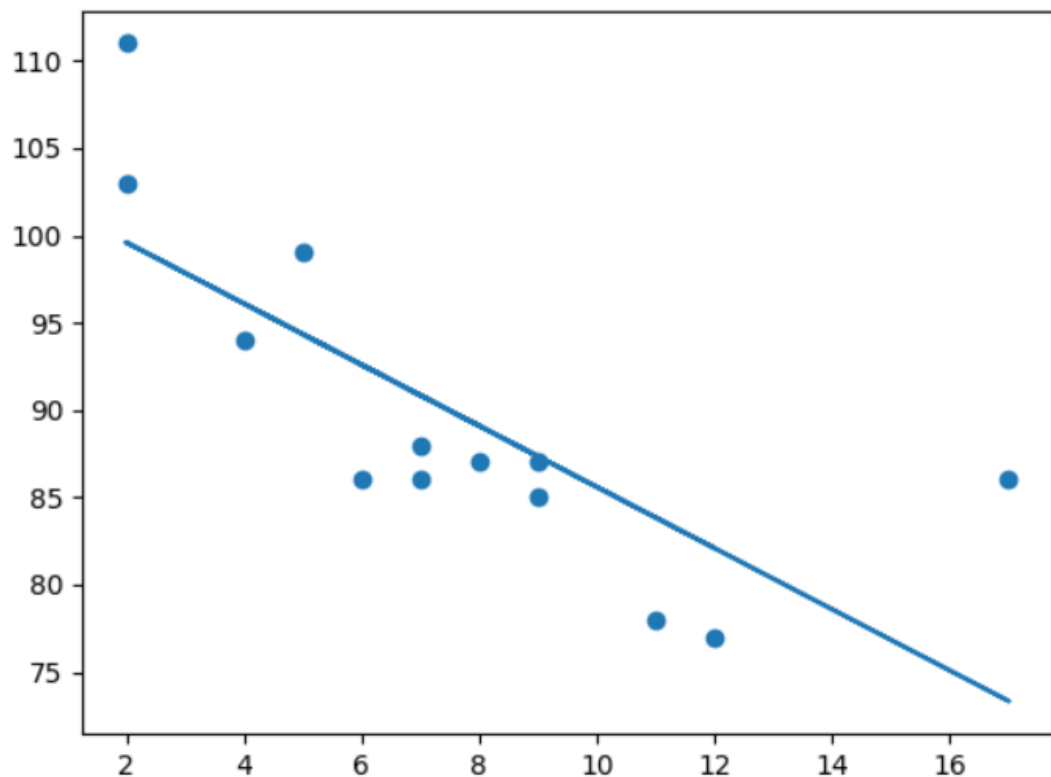
值 r 範圍從 -1 到 1 ，其中 0 表示沒有關係， 1 (和 -1) 表示 100% 相關。

結果 -0.76 表明存在關係，但並不完美，但它表明我們可以在未來的預測中使用線性迴歸。

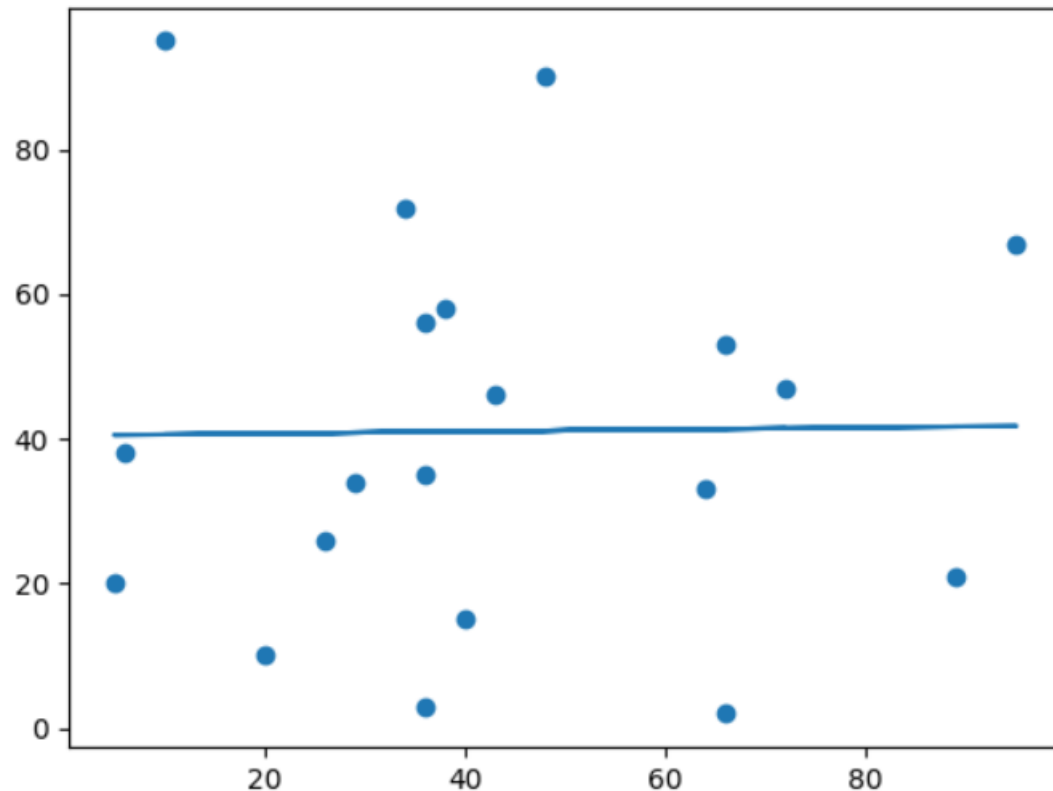
r : -0.758591524376155

什麼時候適合使用線性迴歸? (Linear Regression)

絕對值 $r \leq 0.7$ 時，表明我們不適合使用線性迴歸。



$r = -0.76$ ，表明我們可以在未來的預測中使用線性迴歸。



$r = 0.013$ ，表明我們不可以在未來的預測中使用線性迴歸。

Example : 線性迴歸(Linear Regression)

X : 樓的年齡

Y : 樓的價錢

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

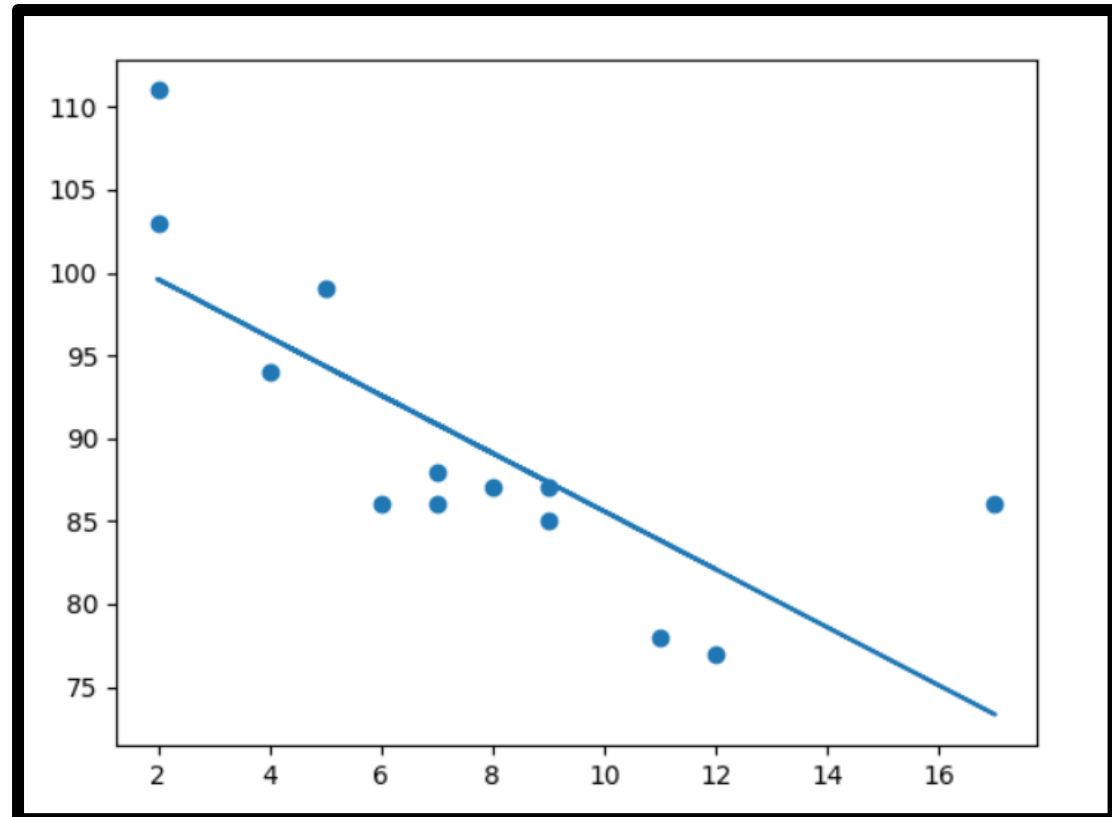
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

```
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

k,b,r,p,std = stats.linregress(x, y)

#假設關係為 :  $y = kx + b$ 
print("k :",k)
print("b :",b)
```



線性迴歸公式推導(Linear Regression)

假設方程為 $y=kx+b$

$$b = \bar{y} - a\bar{x}$$

$$a = \frac{\sum_{i=0}^n x_i y_i - n\bar{x} \bar{y}}{\sum_{i=1}^n x_i x_i - n\bar{x}^2}$$

線性迴歸公式推導(Linear Regression)

中國大學MOOC

线性回归：参数学习

回归模型参数求取: $y_i = ax_i + b$ ($1 \leq i \leq n$) $\min_{a,b} L(a,b) = \sum_{i=1}^n (y_i - a \times x_i - b)^2$

$$\frac{\partial L(a,b)}{\partial a} = \sum_{i=1}^n 2(y_i - ax_i - b)(-x_i) = 0$$

將 $b = \bar{y} - a\bar{x}$ ($\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$, $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$)

代入上式

$$\rightarrow \sum_{i=1}^n (y_i - ax_i - \bar{y} + a\bar{x})(x_i) = 0$$

$$\rightarrow \sum_{i=1}^n (y_i x_i - ax_i x_i - \bar{y} x_i + a\bar{x} x_i) = 0$$

$$\rightarrow \sum_{i=1}^n (y_i x_i - \bar{y} x_i) - a \sum_{i=1}^n (x_i x_i - \bar{x} x_i) = 0$$

$$\rightarrow \left(\sum_{i=1}^n x_i y_i - n\bar{x}\bar{y} \right) - a \left(\sum_{i=1}^n x_i x_i - n\bar{x}^2 \right) = 0$$

$$a = \frac{\sum_{i=1}^n x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^n x_i x_i - n\bar{x}^2}$$

Example : 線性迴歸(Linear Regression)

X : 樓的年齡

Y : 樓的價錢

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

問：當樓的年齡是10年時,大約能賣多少錢呢? 18年時,又是多少呢?

```
from scipy import stats
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

k,b,r,p,std = stats.linregress(x, y)
#假設關係為 : y = kx+b
print("k :",k)
print("b :",b)
print("10年樓的價錢為 : ",k * 10 + b)
print("18年樓的價錢為 : ",k * 18 + b)
```

```
k : -1.751287711552612
b : 103.10596026490066
10年樓的價錢為 : 85.59308314937454
18年樓的價錢為 : 71.58278145695364
```

練習：找規律填數

1:

(10,15,20,25,30,35,40,45)

(13,__,26,32,__,42,49,56)

問：圖中缺少的數字為多少呢？

能找出它們的關係式嗎？

練習：找規律填數

```
from scipy import stats
import sys
import matplotlib
import matplotlib.pyplot as plt

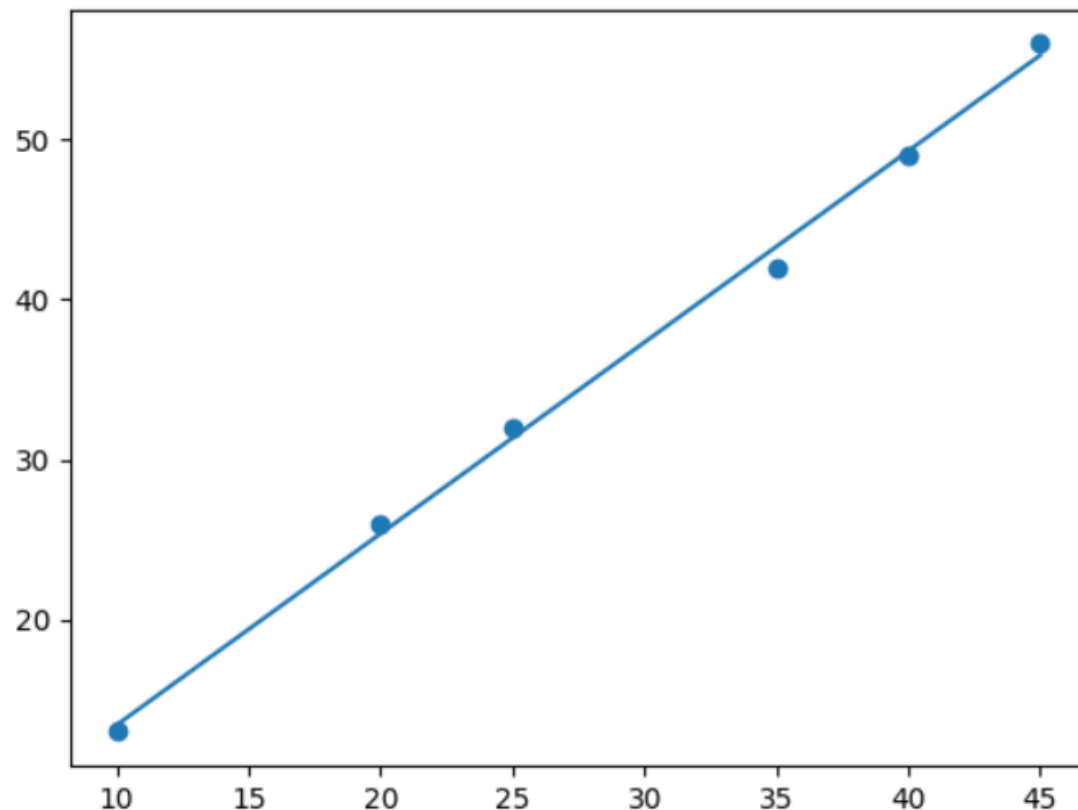
x = [10,20,25,35,40,45]
y = [13,26,32,42,49,56]

k,b,r,p,std = stats.linregress(x, y)

def myfunc(x):
    return k * x + b

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
plt.savefig(sys.stdout.buffer)
sys.stdout.flush()
#假設關係為 :  $y = kx + b$ 
print("k :",k)
print("b :",b)
print("15為 : ",k * 15 + b)
print("30為 : ",k * 30 + b)
```



```
k : 1.1961722488038278
b : 1.4449760765550224
15為 : 19.38755980861244
30為 : 37.33014354066986
```

練習2：找出17歲學生的綜合考試分數

X：學生的年齡

Y：學生的綜合考試分數

x = [6,7,8,9,10,12,13,14,15,16,18]

y = [98,95,93,90,80,76,70,76,78,79,90]

問：**17**歲學生的綜合考試分數為多少呢？

能找出它們的關係式嗎？

練習2：找出17歲學生的綜合考試分數

X：學生的年齡

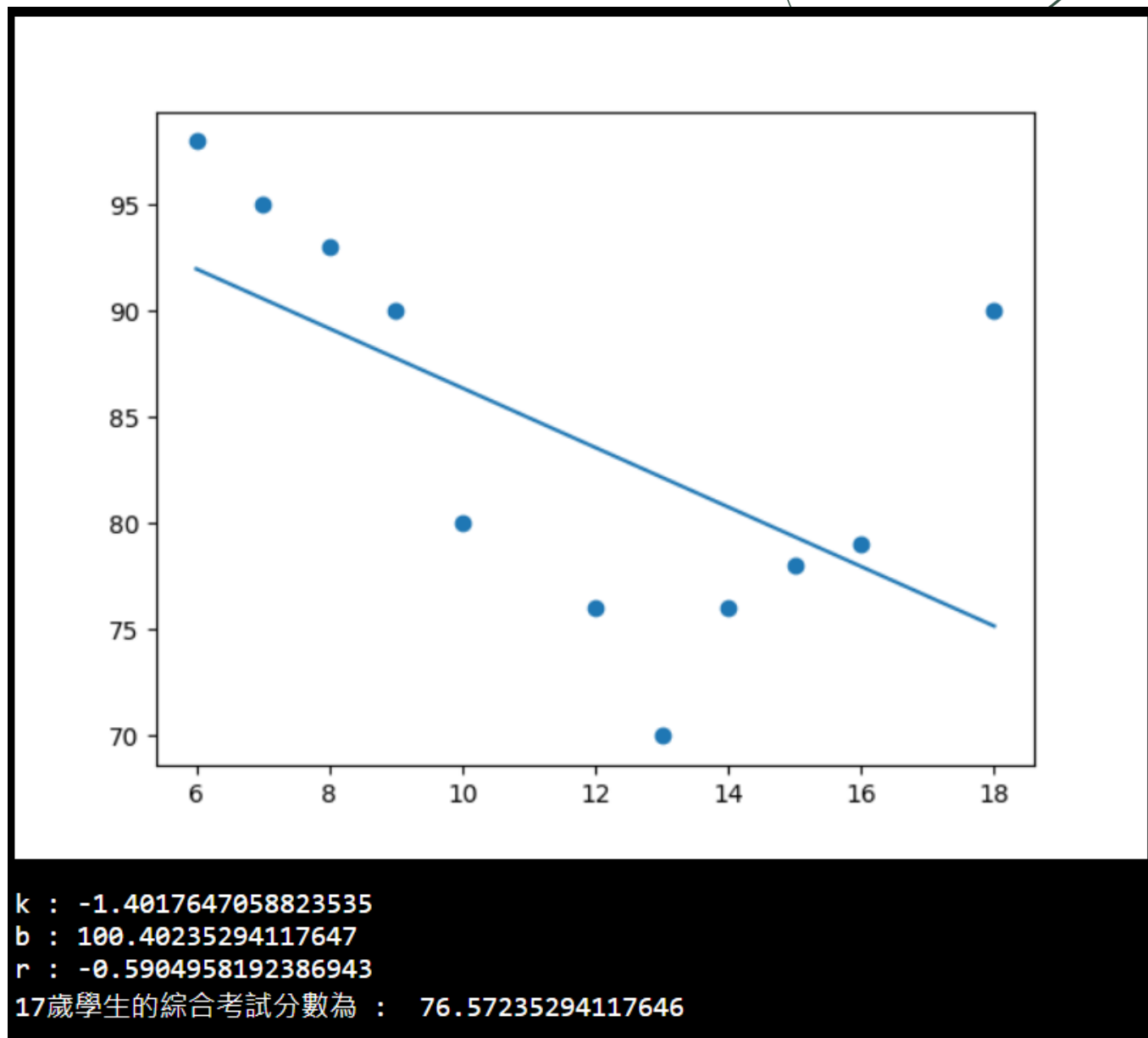
Y：學生的綜合考試分數

$x = [6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18]$

$y = [98, 95, 93, 90, 80, 76, 70, 76, 78, 79, 90]$

$R = -0.5904$

適合使用線性回歸嗎???



多項式迴歸(Polynomial Regression)

X : 學生的年齡

Y : 學生的綜合考試分數

x = [6,7,8,9,10,12,13,14,15,16,18]

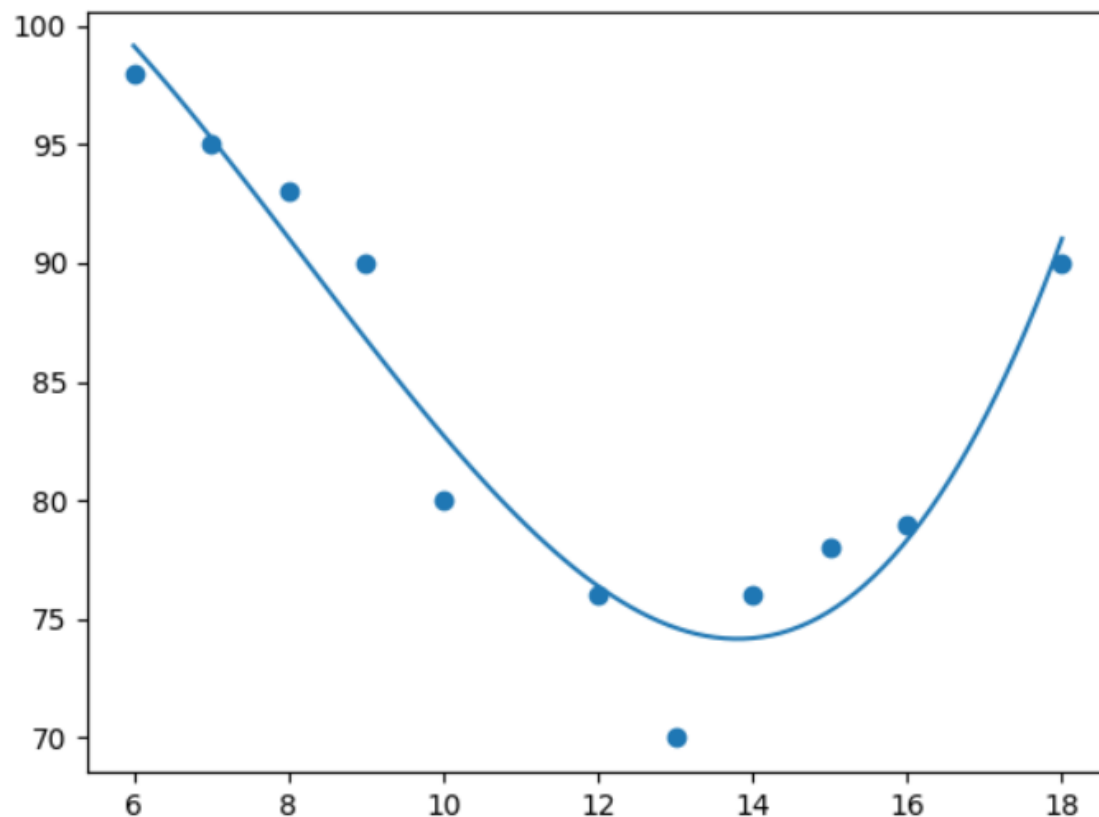
y = [98,95,93,90,80,76,70,76,78,79,90]

```
import numpy
from sklearn.metrics import r2_score

x = [6,7,8,9,10,12,13,14,15,16,18]
y = [98,95,93,90,80,76,70,76,78,79,90]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

seven = mymodel(17)
print("17歲學生的綜合考試分數為 : ",seven)
print("關係式為 : ")
print(mymodel)
```



17歲學生的綜合考試分數為 : 83.47904124021693

關係式為 :

$$0.04568 x^3 - 1.125 x^2 + 4.946 x + 100.1$$

什麼時候適合使用多項式迴歸? (Polynomial Regression)

```
import numpy
from sklearn.metrics import r2_score

x = [6,7,8,9,10,12,13,14,15,16,18]
y = [98,95,93,90,80,76,70,76,78,79,90]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))
```

```
print(r2_score(y, mymodel(x)))
```

```
seven = mymodel(17)
print("17歲學生的綜合考試分數為 :", seven)
print("關係式為 : ")
print(mymodel)
```

0.9351788788037994

17歲學生的綜合考試分數為 : 83.47904124021693

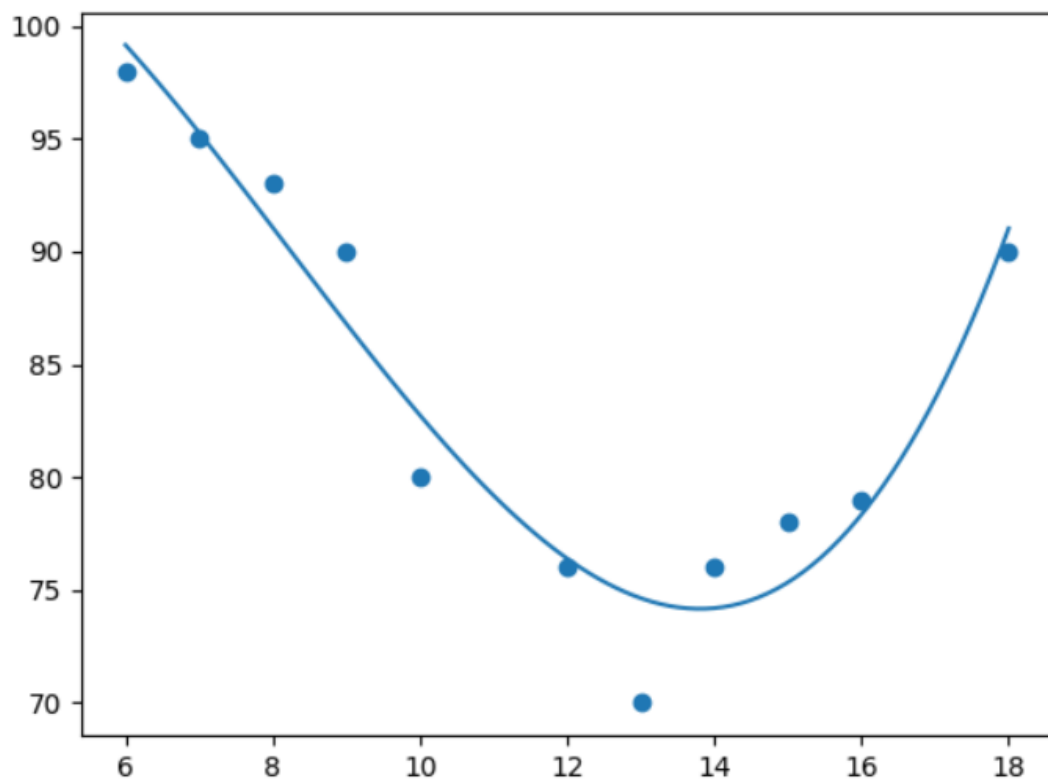
關係式為 :

$$0.04568 x^3 - 1.125 x^2 + 4.946 x + 100.1$$

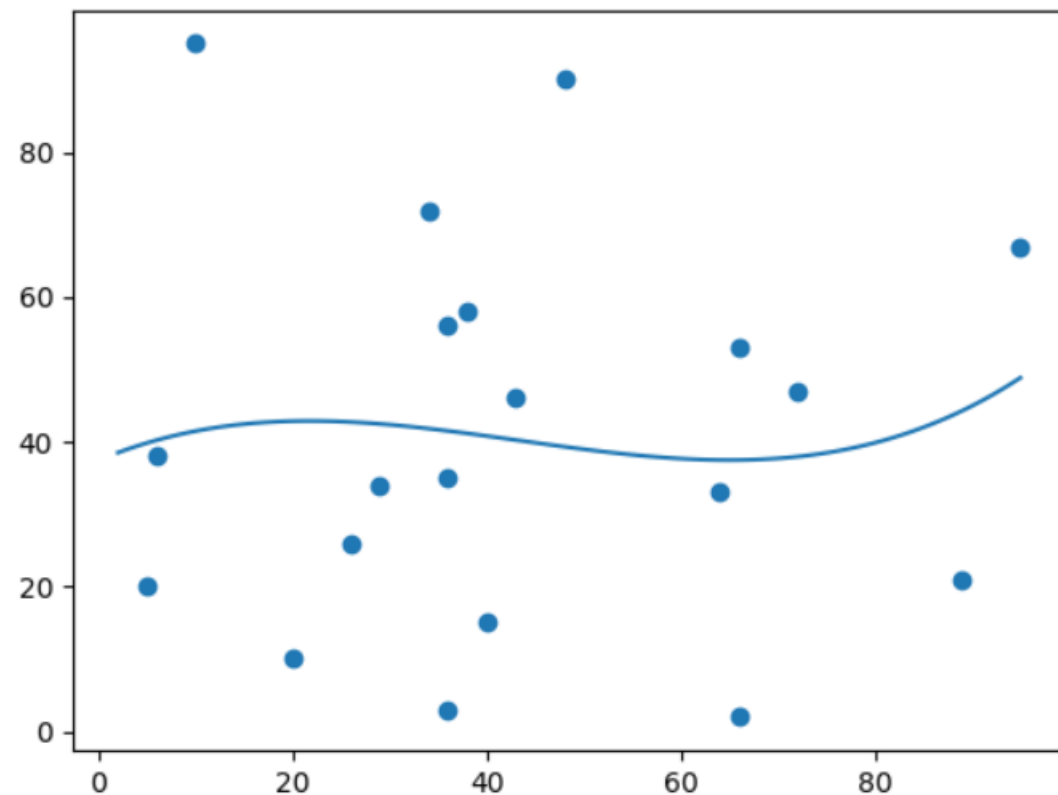
$r = 0.94$ ，表明我們可以在未來的預測中使用多項式迴歸。

什麼時候適合使用多項式迴歸? (Polynomial Regression)

絕對值 $r \leq 0.7$ 時，表明我們不適合使用多項式迴歸。



$r = 0.94$ ，表明我們可以在未來的預測中使用多項式迴歸。



$r = 0.01$ ，表明我們不可以在未來的預測中使用線性迴歸。

練習2：找出17歲學生的綜合考試分數

X：學生的年齡

Y：學生的綜合考試分數

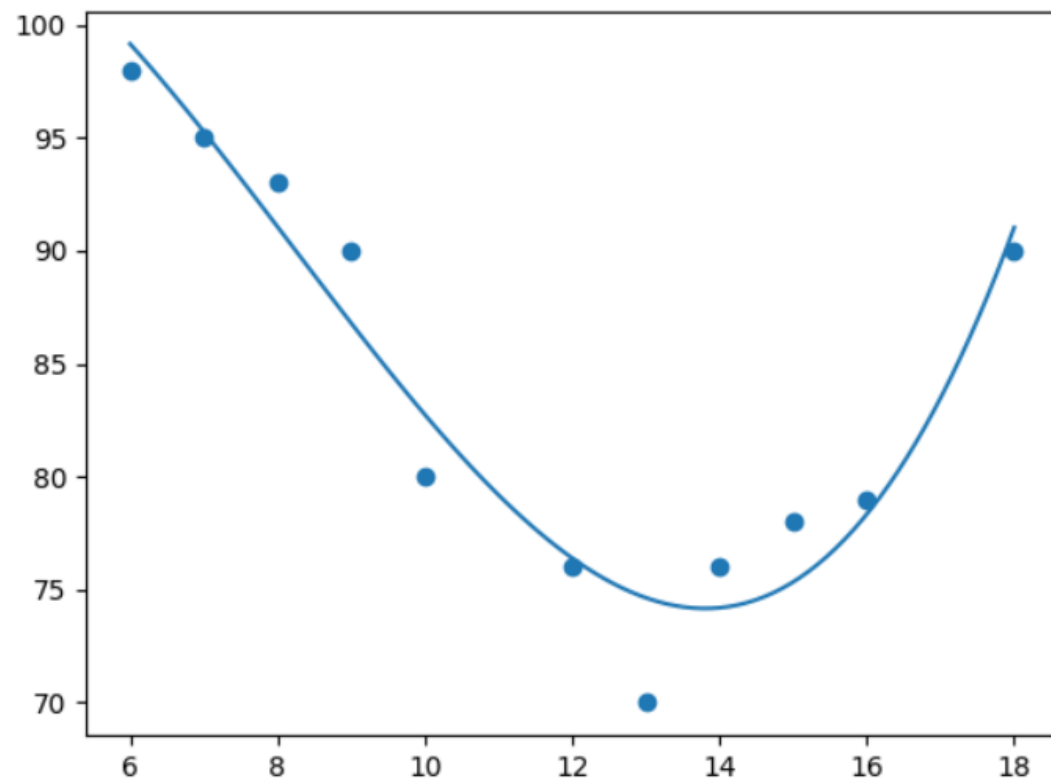
x = [6,7,8,9,10,12,13,14,15,16,18]

y = [98,95,93,90,80,76,70,76,78,79,90]

```
import sys
import numpy
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

x = [6,7,8,9,10,12,13,14,15,16,18]
y = [98,95,93,90,80,76,70,76,78,79,90]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))
myline = numpy.linspace(6, 18, 100)
plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
plt.savefig(sys.stdout.buffer)
sys.stdout.flush()
seven = mymodel(17)
print("17歲學生的綜合考試分數為：", seven)
print("關係式為：")
print(mymodel)
```



17歲學生的綜合考試分數為：83.47904124021693

關係式為：

$$0.04568 x^3 - 1.125 x^2 + 4.946 x + 100.1$$

練習2：找出17歲學生的綜合考試分數

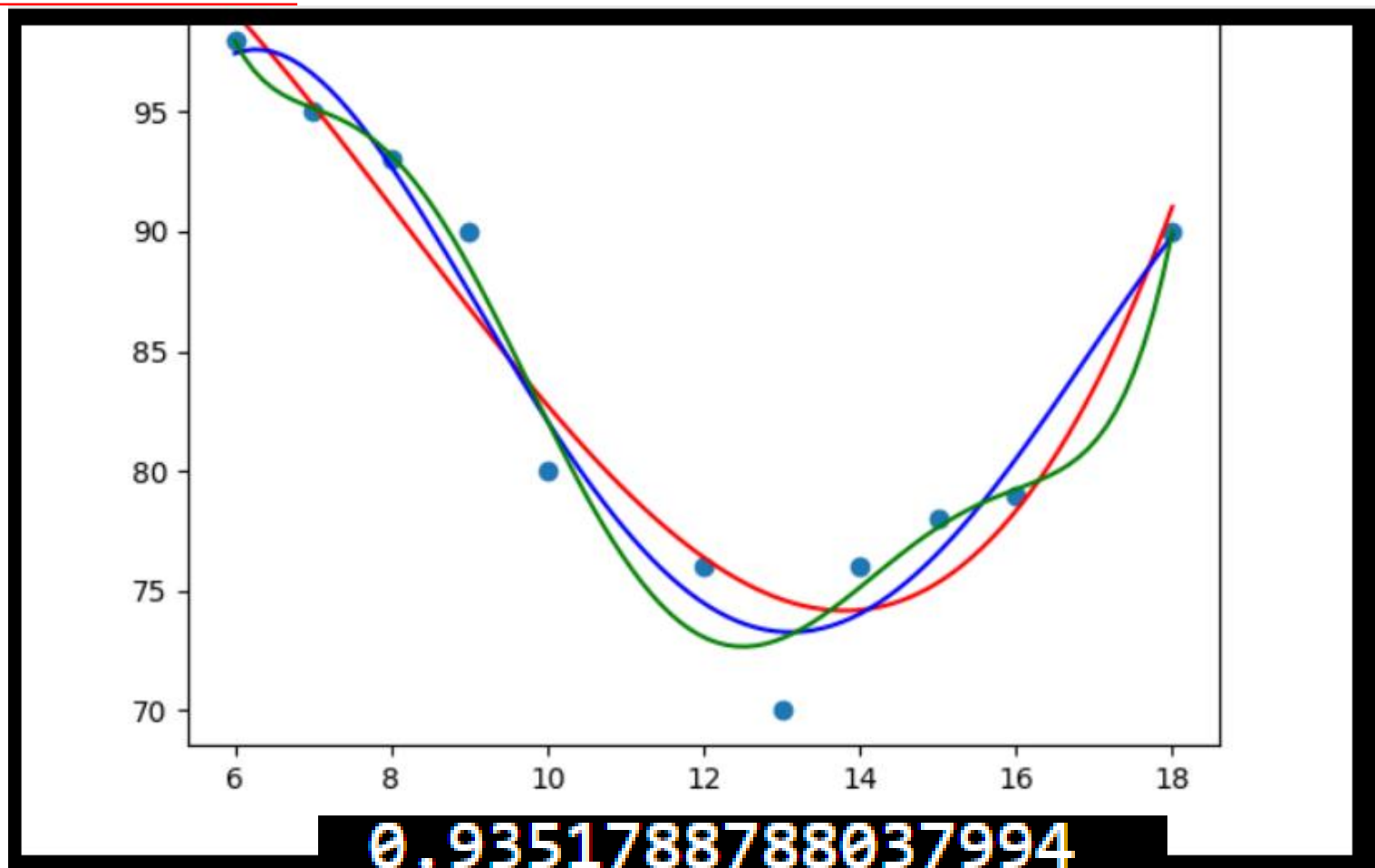
分析擬合程度： r^2_score 愈接近1愈好

```
import sys
import numpy
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

x = [6,7,8,9,10,12,13,14,15,16,18]
y = [98,95,93,90,80,76,70,76,79,90]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))
mymodel2 = numpy.poly1d(numpy.polyfit(x, y, 5))
mymodel3 = numpy.poly1d(numpy.polyfit(x, y, 7))
myline = numpy.linspace(6, 18, 100)
plt.scatter(x, y)
plt.plot(myline, mymodel(myline), color = 'r')
plt.plot(myline, mymodel2(myline), color = 'b')
plt.plot(myline, mymodel3(myline), color = 'g')
plt.show()
plt.savefig(sys.stdout.buffer)
sys.stdout.flush()
seven = mymodel(17)
print("17歲學生的綜合考試分數為：", seven)
print("關係式為：")
print(mymodel)

print(r2_score(y, mymodel(x)))
print(r2_score(y, mymodel2(x)))
print(r2_score(y, mymodel3(x)))
```



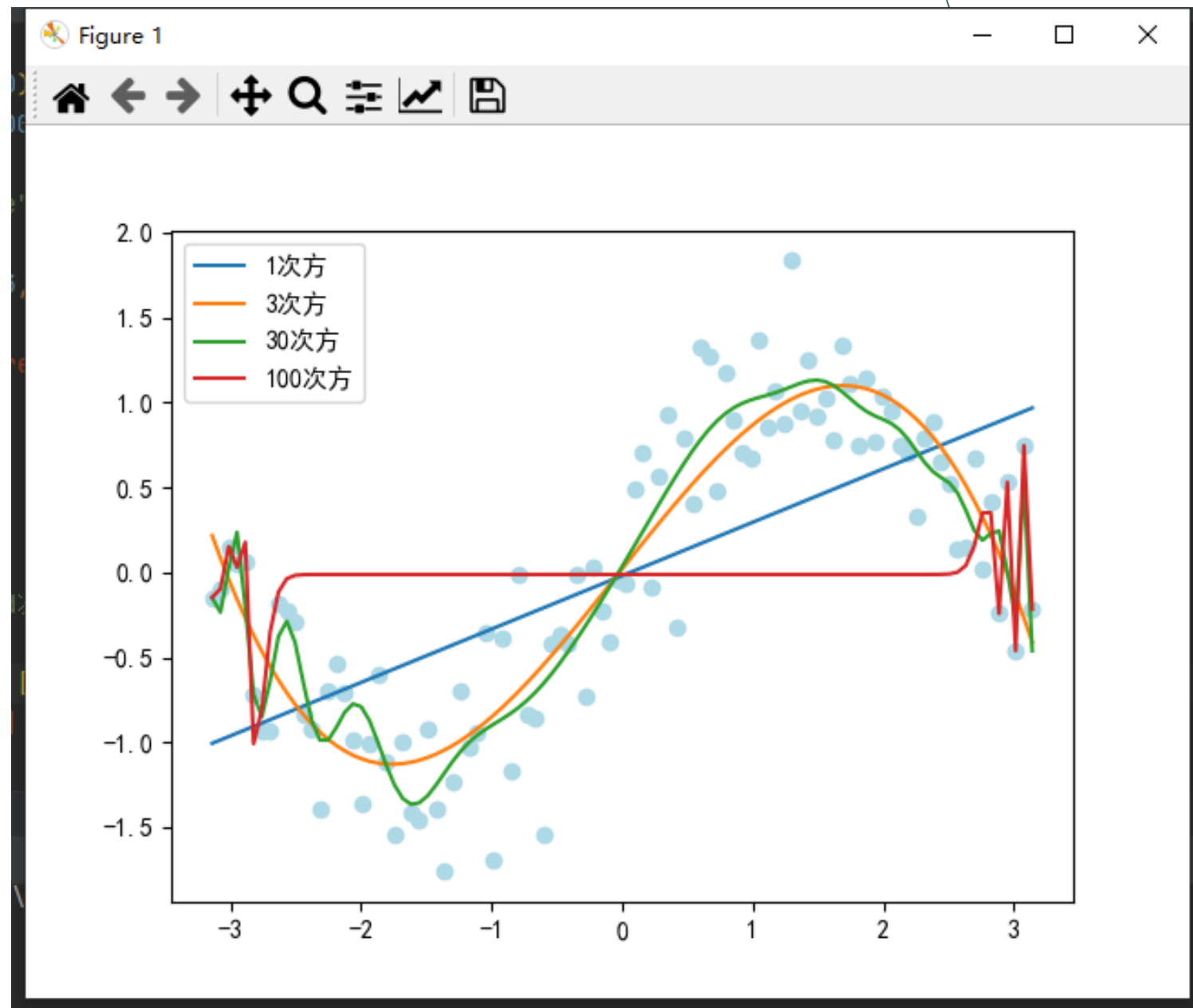
0.9351788788037994

0.9598732961245457

0.9710841092033379

多項式迴歸(Polynomial Regression)

次方愈多愈好嗎?



多項式迴歸(Polynomial Regression)

我們將 y 定義為二次函數，增加一個二次項，就能用它來表示這條曲線了。

$$f(x) = ax^2 + bx + c$$

或者我們用更高次數的表達式也可以。
這樣就能表示更複雜的曲線了。

$$y = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m$$

多項式迴歸推導(Polynomial Regression)

Polynomial Regression - Example

$m=2$

- Fit a second order polynomial to the data given in table.
- Procedure

$$(n)a_0 + (\sum x_i)a_1 + (\sum x_i^2)a_2 = \sum y_i$$

$$(\sum x_i)a_0 + (\sum x_i^2)a_1 + (\sum x_i^3)a_2 = \sum x_i \sum y_i$$

$$(\sum x_i^2)a_0 + (\sum x_i^3)a_1 + (\sum x_i^4)a_2 = \sum x_i^2 y_i$$

$$\rightarrow y = a_0 + a_1x + a_2x^2$$

x	y
0	2.1
1	7.7
2	13.6
3	27.2
4	40.9
5	61.1



今日目標

- 1 : 認識邏輯迴歸 (Logistic Regression)
- 2 : 認識多元邏輯迴歸 (Logistic Regression)



多元迴歸(Multiple Regression)

多元迴歸類似於線性迴歸，但具有多個獨立值，這意味著我們嘗試根據兩個或多個變數來預測一個值。

樓的年齡	樓的呎數(呎)	樓的價錢(萬)
5	910	99
7	1035	86
8	1100	87
7	1050	88
2	930	111
14	1380	86
2	900	103
4	800	87
11	1280	94
12	850	78
9	760	77
6	880	85

多元迴歸(Multiple Regression)

X : 樓的年齡,呎數

Y : 樓的價錢

X =

[[5,910],[7,1035],[8,1100],[7,1050],[2,930],[14,1380],[2,900],[4,800],[11,1280],
[12,850],[9,760],[6,880]]

y = [99,86,87,88,111,86,103,87,94,78,77,85]

問：一棟**1100**呎 **10**年的樓, 價格大概為多少呢?

能找出它們的關係式嗎?

多元迴歸(Multiple Regression)

```
from sklearn import linear_model

X = [[5,910],[7,1035],[8,1100],[7,1050],[2,930],[14,1380],[2,900],[4,800],[11,1280],
[12,850],[9,760],[6,880]]

y = [99,86,87,88,111,86,103,87,94,78,77,85]

regr = linear_model.LinearRegression()
regr.fit(X, y)

#假設關係式為  $z = ax+by+c$ 
print(regr.coef_)
print(regr.intercept_)
print("關係式為 :  $z =$ ",regr.coef_[0]," $x_1 +$ ",regr.coef_[1]," $x_2 +$ ",regr.intercept_)

predict = regr.predict([[10, 1100]])
print("10年1100呎的樓價格為 : ",predict)

print(regr.coef_[0]*10+regr.coef_[1]*1100+regr.intercept_)|
```

```
[-2.71057559  0.03468515]
```

```
75.41115626784405
```

```
關係式為 :  $z = -2.710575594468361 x_1 + 0.03468515381091526 x_2 + 75.41115626784405$ 
```

```
10年1100呎的樓價格為 : [86.45906952]
```

```
86.45906951516723
```

邏輯迴歸(Logistic Regression)

邏輯迴歸旨在解決分類問題。它透過預測分類結果來實現這一點，這與預測連續結果的線性回歸不同。

在最簡單的情況下，有兩個結果，稱為二項式，電腦會使表示為0或1。其他情況有兩個以上的結果需要分類，在這種情況下稱為多項式。多項邏輯迴歸的常見範例是預測 3 個不同物種之間的鳶尾花的類別。

邏輯迴歸（**Logistic Regression**）主要解決二分類問題，用來表示某件事情發生的可能性。

邏輯迴歸(Logistic Regression)



什么是逻辑回归？

表达某件事情发生的可能性

邏輯迴歸(Logistic Regression) 優缺點

優點：

- 實現簡單，廣泛的應用於工業問題；
- 分類時計算量非常小，速度很快，儲存資源低；
- 便利的觀測樣本機率分數；
- 對邏輯迴歸而言，多重共線性並不是問題，
- 計算代價不高，易於理解和實現；

缺點：

- 當特徵空間很大時，邏輯迴歸的表現不是很好；
- 容易欠擬合，一般準確度不太高不能很好地處理大量多類特徵或變數；
- 只能處理兩分類問題（在此基礎上衍生出來的softmax可以用於多分類），
- 必須線性可分；
- 對於非線性特徵，需要進行轉換；

邏輯迴歸(Logistic Regression)－應用

醫療保健

醫學研究人員透過預測患者疾病的可能性，來規劃預防性照護和治療。他們使用邏輯迴歸模型，來比較家族史或基因體對疾病的影響。

金融

金融公司必須分析金融交易是否存在詐騙，並評估貸款申請和保險申請的風險。這些問題適用於邏輯迴歸模型，因為它們具有離散結果，例如高風險或低風險，以及詐騙性或非詐騙性。

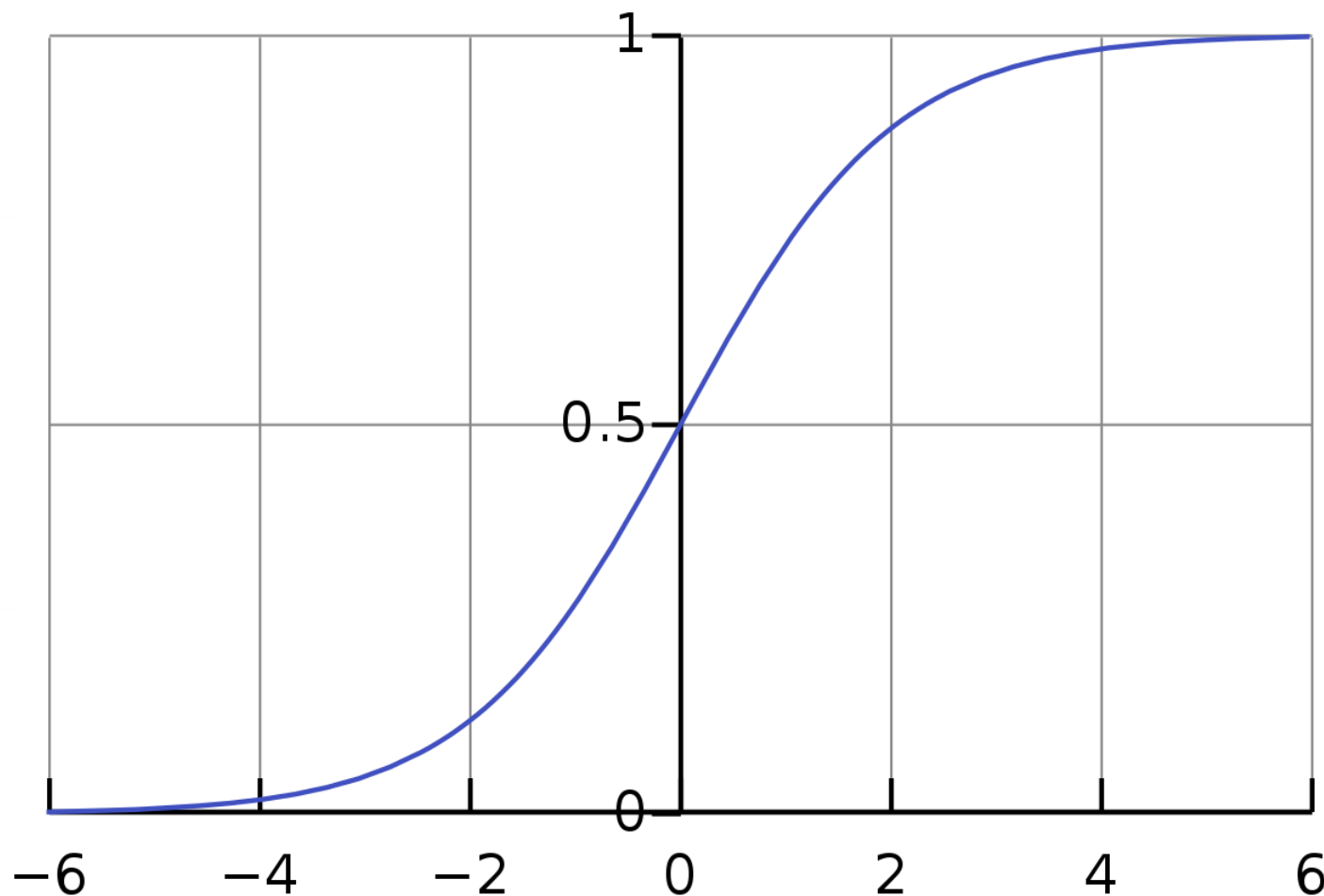
行銷

線上廣告工具使用邏輯迴歸模型，來預測使用者是否會點擊廣告。因此，行銷人員可以分析使用者對不同文字和影像的回應，並建立高效能廣告，讓客戶能夠與之互動。

邏輯迴歸(Logistic Regression)－數學原理

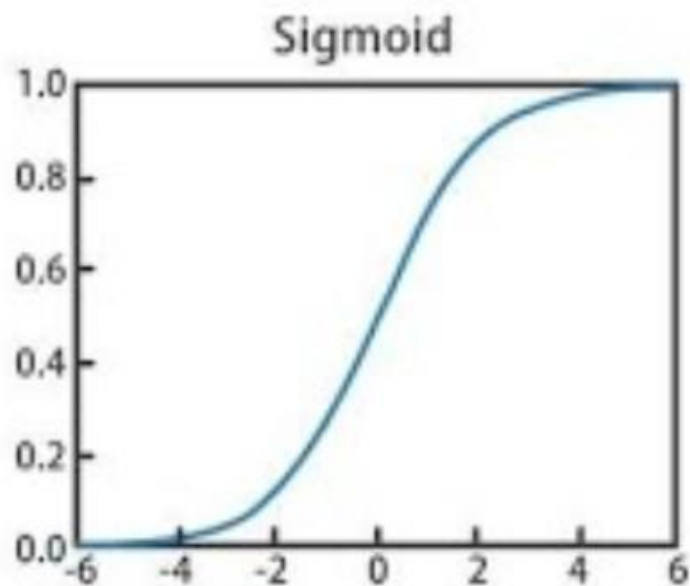
線性迴歸中的假設函數的形式不滿足分類問題，因為在分類問題中我們預期把得到的函數看作是一個機率，所以其值應該是在區間 $[0,1]$ 內的。為此，我們可以對原始函數進行“擠壓”，使其值域屬於 $[0,1]$ 。此時使用到了 S 型函數（Sigmoid function）：

$$A = \frac{1}{1+e^{-x}}$$



邏輯迴歸(Logistic Regression)－數學原理

逻辑回归的基本概念



$$h_{\theta}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n)}}$$

邏輯迴歸(Logistic Regression)--Example

X : 學生的成績平均分

Y : 學生是否成功保送xx大學

x = [78, 93, 83, 88, 92, 82, 98, 88, 95, 86]

y = [0, 1, 0, 0, 1, 0, 1, 1, 1, 0] 1:代表成功保送, 0:代表保送失敗

問：一位平均分為**90**分的同學, 能否成功保送呢?

能知道保送的成功率嗎?

邏輯迴歸(Logistic Regression)

在 sklearn 模組中，我們將使用 `LogisticRegression()` 方法建立邏輯迴歸。

該物件有一個名為的方法 `fit()`，該方法將獨立值和相關值作為參數，並用描述關係的資料填充迴歸物件：

```
logr = linear_model.LogisticRegression()  
logr.fit(X,y)
```

進行預測的函數

```
predicted = logr.predict(numpy.array([90]).reshape(-1,1))
```


完整Program如右圖

```
import numpy  
from sklearn import linear_model  
  
X = numpy.array([78,93,83,88,92,82,98,88,95,86]).reshape(-1,1)  
y = numpy.array([0, 1, 0, 0, 1, 0, 1, 1, 1, 0])  
  
logr = linear_model.LogisticRegression()  
logr.fit(X,y)  
  
predicted = logr.predict(numpy.array([90]).reshape(-1,1))  
print(predicted)
```

多元邏輯迴歸--Example

嘗試根據數據進行分析天氣是snow還是Clear

1	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
69	1/3/2012 19:00	-16.9	-24.8	50	24	25	101.7	Clear
116	1/5/2012 18:00	-7.1	-14.4	56	11	25	100.7	Clear
117	1/5/2012 19:00	-9.2	-15.4	61	7	25	100.8	Clear
118	1/5/2012 20:00	-9.8	-15.7	62	9	25	100.8	Clear
119	1/5/2012 21:00	-9	-14.8	63	13	25	100.8	Clear
243	1/11/2012 1:00	-10.7	-17.8	56	17	25	101.4	Clear
244	1/11/2012 2:00	-12	-18.9	56	19	25	101.5	Clear

	A	B	C	D	E	F	G	H
	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_kmh	Visibility_km	Press_kPa	Weather 
7	1/3/2012 7:00	-14	-19.5	63	19	25	100.95	Snow
5	1/4/2012 12:00	-13.7	-21.7	51	11	24.1	101.25	Snow
8	1/4/2012 14:00	-11.3	-19	53	7	19.3	100.97	Snow
9	1/4/2012 15:00	-10.2	-16.3	61	11	9.7	100.89	Snow
0	1/4/2012 16:00	-9.4	-15.5	61	13	19.3	100.79	Snow
1	1/4/2012 17:00	-8.9	-13.2	71	9	4.8	100.76	Snow
2	1/4/2012 18:00	-8.9	-12.6	75	11	9.7	100.69	Snow
3	1/4/2012 19:00	-8.4	-12.7	71	9	16.1	100.65	Snow

多元邏輯迴歸--Example

嘗試根據數據進行分析天氣是snow還是Clear

1	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
69	1/3/2012 19:00	-16.9	-24.8	50	24	25	101.7	Clear
116	1/5/2012 18:00	-7.1	-14.4	56	11	25	100.7	Clear
117	1/5/2012 19:00	-9.2	-15.4	61	7	25	100.8	Clear
118	1/5/2012 20:00	-9.8	-15.7	62	9	25	100.8	Clear
119	1/5/2012 21:00	-9	-14.8	63	13	25	100.8	Clear
243	1/11/2012 1:00	-10.7	-17.8	56	17	25	101.4	Clear
244	1/11/2012 2:00	-12	-18.9	56	19	25	101.5	Clear

輸入新數據,以驗證多元邏輯迴歸模型是否精準?

A	B	C	D	E	F	G	H
Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
1/3/2012 7:00	-14	-19.5	63	19	25	100.95	Snow
1/4/2012 12:00	-13.7	-21.7	51	11	24.1	101.25	Snow
1/4/2012 14:00	-11.3	-19	53	7	19.3	100.97	Snow
1/4/2012 15:00	-10.2	-16.3	61	11	9.7	100.89	Snow
1/4/2012 16:00	-9.4	-15.5	61	13	10.3	100.76	Snow
1/4/2012 17:00	-8.9	-13.2	71	9	4.8	100.76	Snow
1/4/2012 18:00	-8.9	-12.6	75	11	9.7	100.65	Snow
1/4/2012 19:00	-8.4	-12.7	71	9	16.1	100.65	Snow

多元邏輯迴歸--Example

嘗試根據數據進行分析天氣是snow還是Clear

```
import numpy
from sklearn import linear_model

X = numpy.array([[ -14, -19.5, 63, 19, 25], [-13.7, -21.7, 51, 11, 24.1], [-11.3, -19, 53, 7, 19.3], [-10.2, -16.3, 61, 11, 9.7],
[-9.4, -15.5, 61, 13, 19.3], [-16.9, -24.8, 50, 24, 25], [-7.1, -14.4, 56, 11, 25], [-9.2, -15.4, 61, 7, 25], [-9.8, -15.7, 62, 9, 25],
[-9, -14.8, 63, 13, 25]]).reshape(10,5)
y = numpy.array([1,1,1,1,1,0,0,0,0,0])
#1 : means snow
#0 : means clear

print(X)

logr = linear_model.LogisticRegression()
logr.fit(X,y)

log_odds = logr.coef_
odds = numpy.exp(log_odds)

predicted = logr.predict(numpy.array([[-12, -18.9, 56, 19, 25]]).reshape(1,5))
print("[ -12, -18.9, 56, 19, 25] is (1 : snow, 0 : clear) : ", predicted)

predicted2 = logr.predict(numpy.array([[-8.9, -13.2, 71, 9, 4.8]]).reshape(1,5))
print("[ -8.9, -13.2, 71, 9, 4.8] is (1 : snow, 0 : clear) : ", predicted2)
```

Run Result

```
[[-14.  -19.5  63.   19.   25. ]
 [-13.7 -21.7  51.   11.   24.1]
 [-11.3 -19.   53.    7.   19.3]
 [-10.2 -16.3  61.   11.    9.7]
 [ -9.4 -15.5  61.   13.   19.3]
 [-16.9 -24.8  50.   24.   25. ]
 [ -7.1 -14.4  56.   11.   25. ]
 [ -9.2 -15.4  61.    7.   25. ]
 [ -9.8 -15.7  62.    9.   25. ]
 [ -9.  -14.8  63.   13.   25. ]]
[-12, -18.9, 56, 19, 25] is (1 : snow, 0 : clear) :  [0]
[-8.9, -13.2, 71, 9, 4.8] is (1 : snow, 0 : clear) :  [1]
```

多元邏輯迴歸--Example

分析天氣是snow的概率

```
def logit2prob(logr, X):
    log_odds = logr.coef_[0][0] * X[0] + logr.coef_[0][1] * X[1] + logr.coef_[0][2] * X[2] + logr.coef_[0][3] *
X[3] + logr.coef_[0][4] * X[4] + logr.intercept_
    odds = numpy.exp(log_odds)
    probability = odds / (1 + odds)
    return(probability)
print("=====")
print("下雪的概率為 : ",logit2prob(logr, [-12,-18.9,56,19,25]))
print("下雪的概率為 : ",logit2prob(logr, [-8.9,-13.2,71,9,4.8]))
print(logr.coef_)
print(logr.intercept_)
```

Run Result

```
[ -9.   -14.8  63.   13.   25. ]
[-12,-18.9,56,19,25] is (1 : snow,0 : clear) : [0]
[-8.9,-13.2,71,9,4.8] is (1 : snow,0 : clear) : [1]
=====
→ 下雪的概率為 : [0.03901305]
→ 下雪的概率為 : [0.99999999]
[[-0.58713931 -0.52204897  0.34543707 -0.28728029 -0.93219004]]
[-10.69786146]
```

機械學習—數據分析項目

分析天氣是snow的概率

Code

```
Python code data.csv
import pandas
import numpy
from sklearn import linear_model

df = pandas.read_csv("data.csv")
print(df)

d = {'Snow': 1, 'Clear': 0}
features = ['Temp_C', 'Dew Point Temp_C', 'Rel Hum_%', 'Wind Speed_kmh', 'Visibility_km']
X = df[features]
y = df['Weather']

logr = linear_model.LogisticRegression()
logr.fit(X,y)

log_odds = logr.coef_
odds = numpy.exp(log_odds)

predicted = logr.predict(numpy.array([[-12,-18.9,56,19,25]]).reshape(1,5))
print("[-12,-18.9,56,19,25] is (1 : snow,0 : clear) : ",predicted)

predicted2 = logr.predict(numpy.array([[-8.9,-13.2,71,9,4.8]]).reshape(1,5))
print("[-8.9,-13.2,71,9,4.8] is (1 : snow,0 : clear) : ",predicted2)

def logit2prob(logr, X):
    log_odds = logr.coef_[0][0] * X[0] + logr.coef_[0][1] * X[1] + logr.coef_[0][2] * X[2]
    + logr.coef_[0][3] * X[3] + logr.coef_[0][4] * X[4] + logr.intercept_
    odds = numpy.exp(log_odds)
    probability = odds / (1 + odds)
    return(probability)

print("=====")
print("下雪的概率為 : ",logit2prob(logr, [-12,-18.9,56,19,25]))
print("下雪的概率為 : ",logit2prob(logr, [-8.9,-13.2,71,9,4.8]))
print(logr.coef_)
print(logr.intercept_)
```

Result

	Temp_C	Dew Point	Temp_C	Rel Hum_%	Wind Speed_kmh	Visibility_km	Weather
0	-14.0		-19.5	63	19	25.0	Snow
1	-13.7		-21.7	51	11	24.1	Snow
2	-11.3		-19.0	53	7	19.3	Snow
3	-10.2		-16.3	61	11	9.7	Snow
4	-9.4		-15.5	61	13	19.3	Snow
5	-16.9		-24.8	50	24	25.0	Clear
6	-7.1		-14.4	56	11	25.0	Clear
7	-9.2		-15.4	61	7	25.0	Clear
8	-9.8		-15.7	62	9	25.0	Clear
9	-9.0		-14.8	63	13	25.0	Clear

[-12,-18.9,56,19,25] is (1 : snow,0 : clear) : ['Clear']
[-8.9,-13.2,71,9,4.8] is (1 : snow,0 : clear) : ['Snow']
=====

下雪的概率為 : [0.03901305]
下雪的概率為 : [0.99999999]
[[-0.58713931 -0.52204897 0.34543707 -0.28728029 -0.93219004]]
[-10.69786146]

機械學習—數據分析項目

分析天氣是snow的概率

Data.csv

Result

Python code

data.csv

```
Temp_C,Dew Point Temp_C,Rel Hum_%,Wind Speed_kmh,Visibility_km,Weather
-14,-19.5,63,19,25,Snow
-13.7,-21.7,51,11,24.1,Snow
-11.3,-19.53,7,19.3,Snow
-10.2,-16.3,61,11,9.7,Snow
-9.4,-15.5,61,13,19.3,Snow
-16.9,-24.8,50,24,25,Clear
-7.1,-14.4,56,11,25,Clear
-9.2,-15.4,61,7,25,Clear
-9.8,-15.7,62,9,25,Clear
-9,-14.8,63,13,25,Clear
```

```
Temp_C Dew Point Temp_C Rel Hum_% Wind Speed_kmh Visibility_km Weather
0 -14.0 -19.5 63 19 25.0 Snow
1 -13.7 -21.7 51 11 24.1 Snow
2 -11.3 -19.0 53 7 19.3 Snow
3 -10.2 -16.3 61 11 9.7 Snow
4 -9.4 -15.5 61 13 19.3 Snow
5 -16.9 -24.8 50 24 25.0 Clear
6 -7.1 -14.4 56 11 25.0 Clear
7 -9.2 -15.4 61 7 25.0 Clear
8 -9.8 -15.7 62 9 25.0 Clear
9 -9.0 -14.8 63 13 25.0 Clear
[-12,-18.9,56,19,25] is (1 : snow,0 : clear) : ['Clear']
[-8.9,-13.2,71,9,4.8] is (1 : snow,0 : clear) : ['Snow']
=====
下雪的概率為 : [0.03901305]
下雪的概率為 : [0.99999999]
[[-0.58713931 -0.52204897 0.34543707 -0.28728029 -0.93219004]]
[-10.69786146]
```


機械學習—數據分析項目

繳交時間：2024年4月底

繳交方式：ZIP檔(把文檔及.PY壓縮成ZIP)

**分組方式：可1人1組,也可2人1組,也可3人1組
(最多3人1組)**

**項目要求：1人1組(至少要有20組以上的數據)
2人1組(至少要有35組以上的數據)
3人1組(至少要有50組以上的數據)**

注意事項：ZIP檔名稱為“項目名稱”+“各組員中文全名”