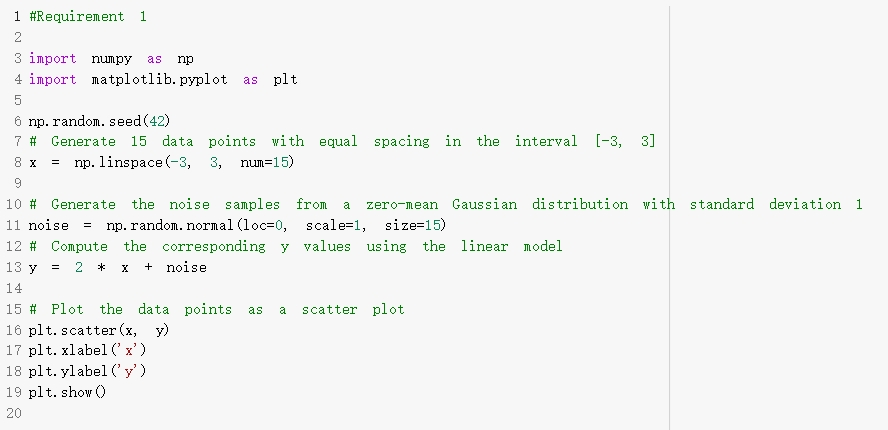
**機器學習< Assignment #2 – Regression>**

**姓名: 翁佳煌　　　學號: 409430030**

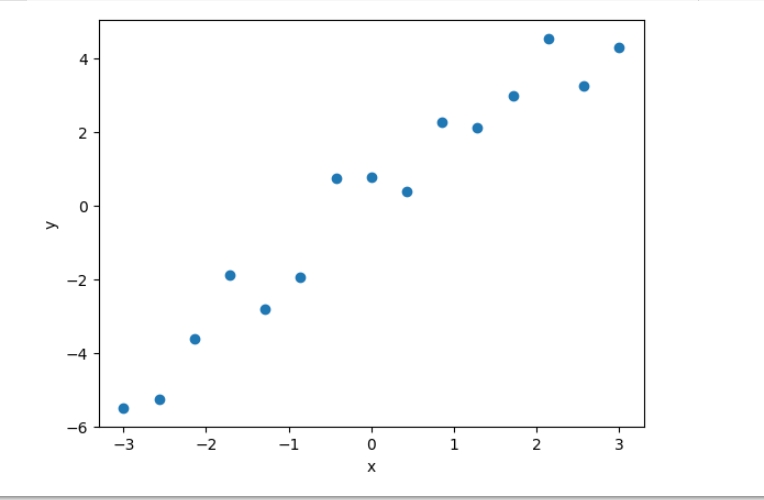
# 要求1:

下圖1.1為要求1的code。

1. 在第6行設定了隨機種子，確保在每次運行程式時生成的隨機數序列都是相同的，以方便此次實驗的觀察。
2. 接下來第8行，使用NumPy的linspace函數在區間[-3, 3]中均勻生成15個數字，並將結果儲存在變數x中。
3. 第11行使用NumPy的normal函數來生成一個包含15個隨機數的數組，其符合均值為0，標準差為1的高斯分佈，並將結果儲存在變數noise中。
4. 接著13行，將噪聲加到x中，並通過線性模型生成對應的y值。
5. 15~19行，我在這裡使用Matplotlib庫，將數據點以散點圖的形式呈現出來。

▲圖1.1

## 下圖1.2為要求1的執行結果:



▲圖1.2

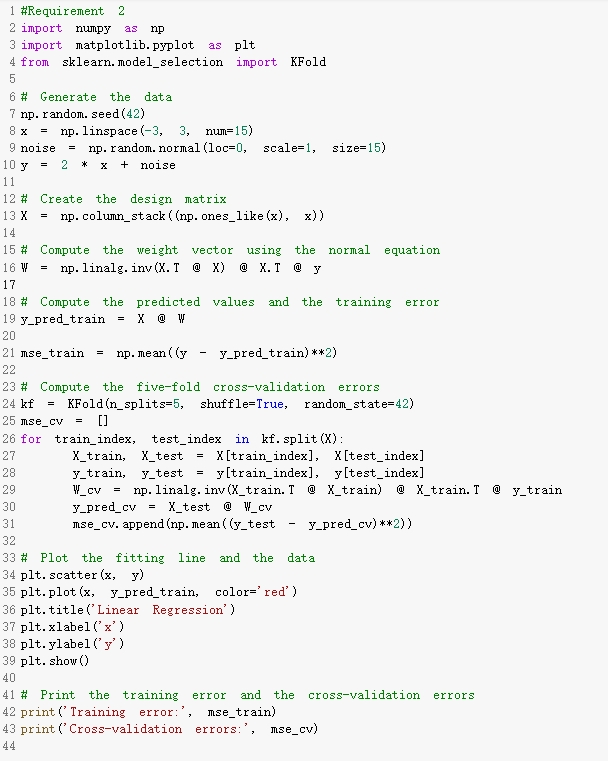
## 要求1問題討論:

一開始遇到要加上高斯noise，之前在機率學過，但幾乎要忘光了，於是又多花了點時間去搞懂這部分。另外，透過將高斯噪聲加入到數據集中，可以使生成的數據更加逼近現實情況，從而使得線性回歸等機器學習模型更能夠有效地處理這些數據。

# 要求2:

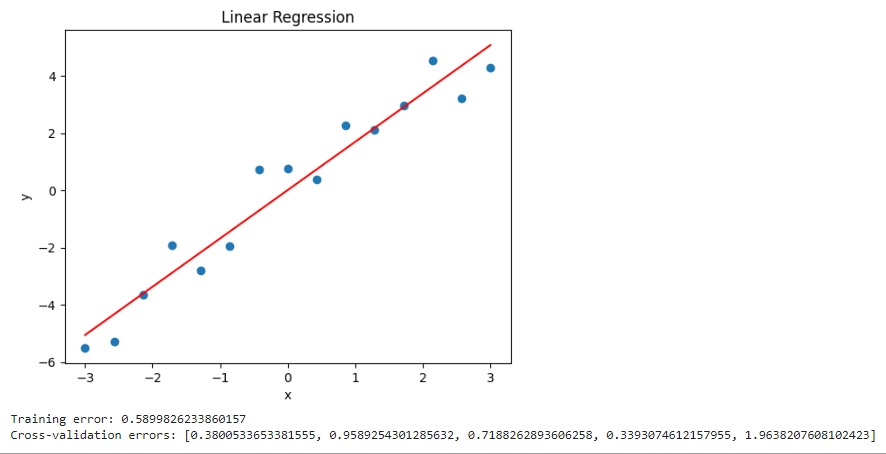
下圖2.1為要求2的code。

1. 1~10行直接採用要求1的code去做延伸，生成帶有高斯噪聲的數據。使用np.linspace生成-3到3之間的15個等距x值，然後生成對應的y值，並且通過給定的線性模型y = 2x + ε和高斯噪聲ε來計算y值。
2. 13行創建設計矩陣。將剛生成的x和一個全為1的向量，按列組成一個矩陣，即設計矩陣X。
3. 計算權重向量。使用課堂上提到的Linear Regression方程式求解權重向量W，即W = (X.T \* X)-1 \* X.T \* y。
4. 計算訓練誤差。使用設計矩陣X和權重向量W，計算預測值y\_pred\_train，並且通過平均平方誤差(MSE)來計算訓練誤差mse\_train。
5. 計算五倍交叉驗證誤差。使用sklearn中的KFold函數將數據集分成5份，然後對每個部分進行循環，每次選擇其中一份作為驗證集，其餘部分作為訓練集。對於每次分割，使用訓練集計算權重向量W\_cv，然後在驗證集上進行預測，計算預測值y\_pred\_cv和MSE，並且將MSE添加到mse\_cv列表中。
6. 繪製擬合線和數據。使用plt.scatter和plt.plot函數繪製數據點和擬合線。
7. 輸出訓練誤差和交叉驗證誤差。使用print函數輸出訓練誤差和交叉驗證誤差。



▲圖2.1

## 下圖2.2為要求2的執行結果:

▲圖2.2

## 要求2問題討論:

我發現Linear Regression其實可以直接from sklearn.linear\_model import LinearRegression來做使用，但我知道老師一定是要我們實作這部分，我現在才漸漸清楚很多模型其實是可以直接套模來使用的。

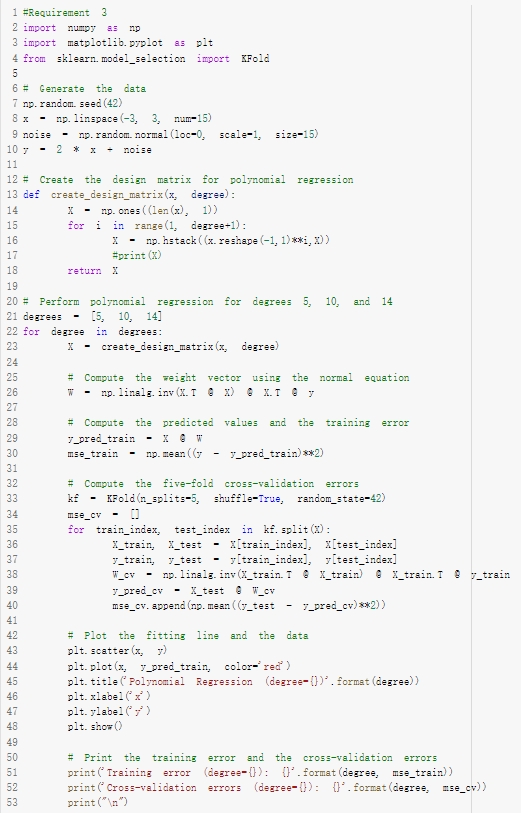
此外，根據執行結果，訓練誤差 (training error) 為 0.5899826233860157，感覺

在訓練集上的表現不是很好，存在一定程度的誤差，而且交叉驗證的平均誤差(cross-validation errors) 比訓練誤差 (training error) 要大，這表示模型可能過度擬合了訓練資料。

# 要求3:

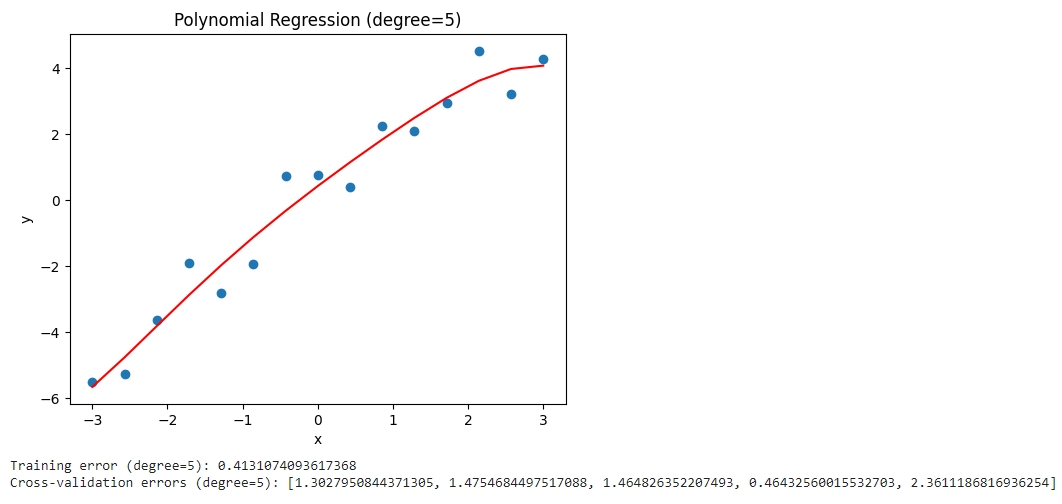
下圖3.1為要求3的code。

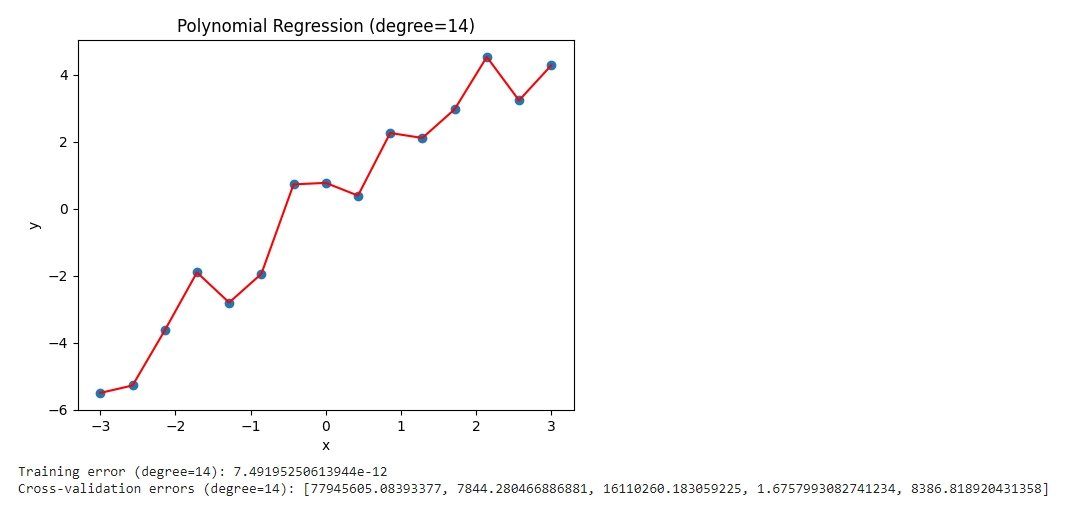
1. 前面1~10行都與要求1相同，不多加贅述。
2. 創建了一個函數create\_design\_matrix，該函數接受x軸數據和多項式的次數degree作為輸入。它通過將x的各次方加入到一個數組中來創建一個矩陣X，使其成為一個多項式，函式的第一行創建一個形狀為(len(x),1)的矩陣X，每個元素都初始化為1，這是因為任何自變量的零次冪都是1。接下來的for循環從1迭代到degree，每次將一個新的冪次向量x.reshape(-1,1)\*\*i添加到X的左側，以形成新的設計矩陣。最後，函式返回設計矩陣X。。
3. 使用5倍交叉驗證來計算模型的平均交叉驗證誤差。為此，我們將數據分為五個不同的集合，每個集合用於訓練模型的不同版本。然後，我們使用這些不同版本來計算交叉驗證誤差，這是指使用不同數據子集訓練模型並測試其在不同數據子集上的表現。
4. 使用matplotlib繪製數據和模型的擬合線，將模型的訓練誤差和交叉驗證誤差印出。

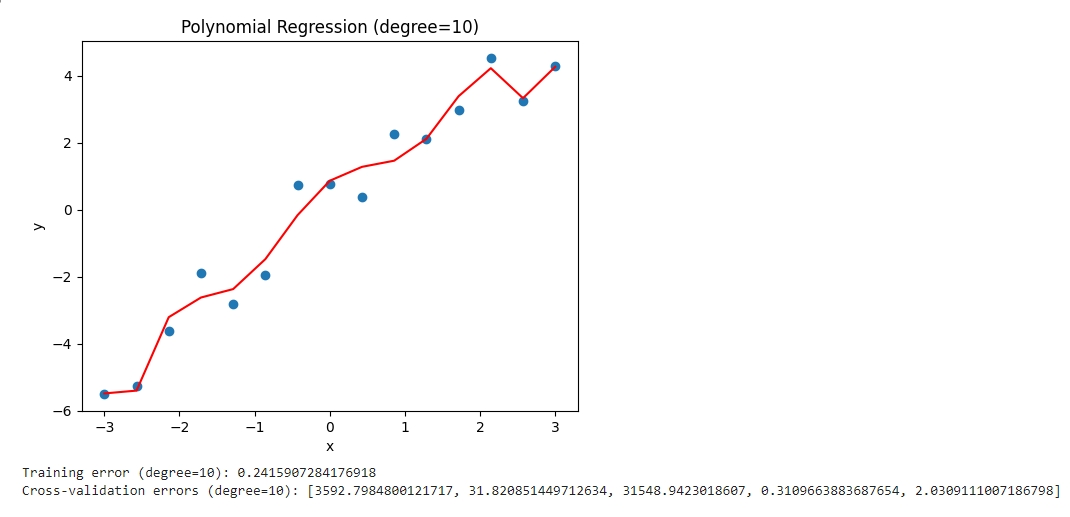


▲圖3.1

## 下圖3.2為要求3的執行結果:







▲圖3.2

## 要求3問題討論:

在一開始看到要實作Polynomial Regression的時候有腦袋卡一下，看到了pdf上的矩陣表示後才發現並不是很難實作，另外，他也可以直接拿Sklearn裡面的模型來直接使用。

另外，從上述輸出結果可以觀察到，隨著多項式回歸的次數增加，訓練誤差逐漸減小，但交叉驗證誤差卻不是一直減小，而是在某一個次數後開始增加，這就是過度擬合的現象。對於次數較低的多項式回歸（degree=5），訓練誤差和交叉驗證誤差都不算太高，這時模型可以較好地擬合數據，泛化能力較好。但是對於次數較高的多項式回歸（degree=10, 14），雖然訓練誤差很小，甚至趨近於0，但是交叉驗證誤差卻很高，這時模型過度擬合，對新數據的泛化能力很差。

我認為這種現象的原因在於，當模型的複雜度過高時，模型會過度擬合訓練數據，從而導致模型對新數據的預測能力下降。此外，當模型的複雜度過高時，模型中的參數會變得非常多，使得模型容易產生噪聲，進而影響模型的預測能力。因此，對於多項式回歸模型，需要在模型複雜度和預測能力之間進行權衡，避免過度擬合的現象發生。

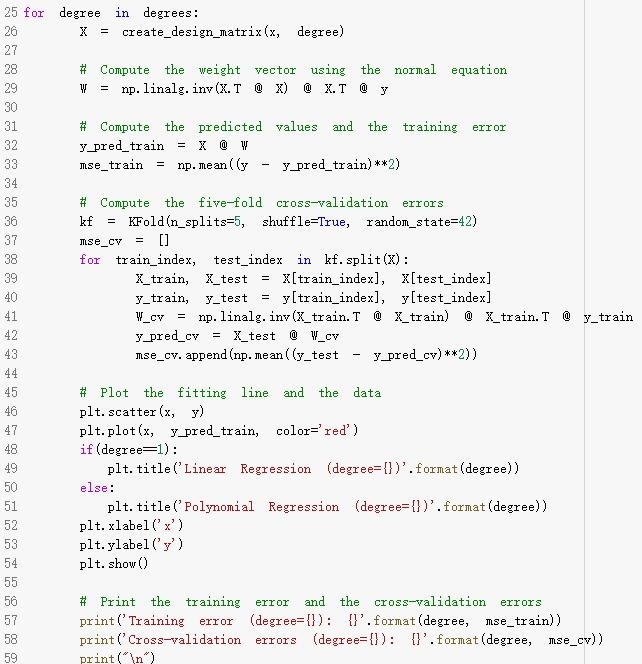
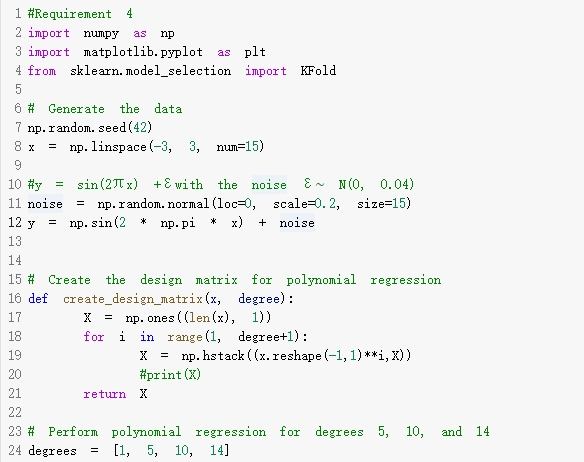
# 要求4:

下圖4.1為要求4的code。

1. 與要求1~3線性函數不同，這裡使用非線性的函數 y=sin(2πx)+ε作為樣本資料，其中 ε 是服從平均值為 0、標準差為 0.2 的常態分佈。

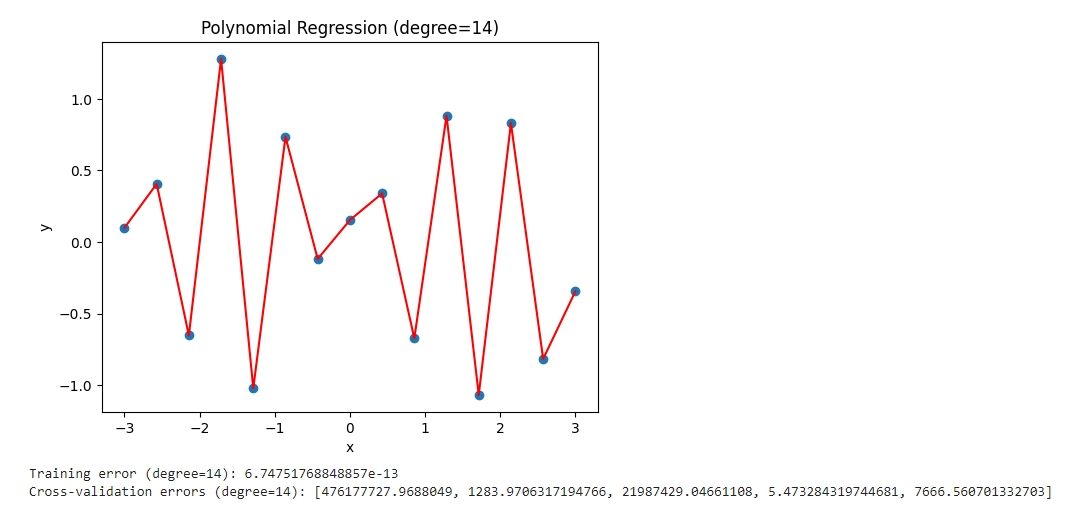
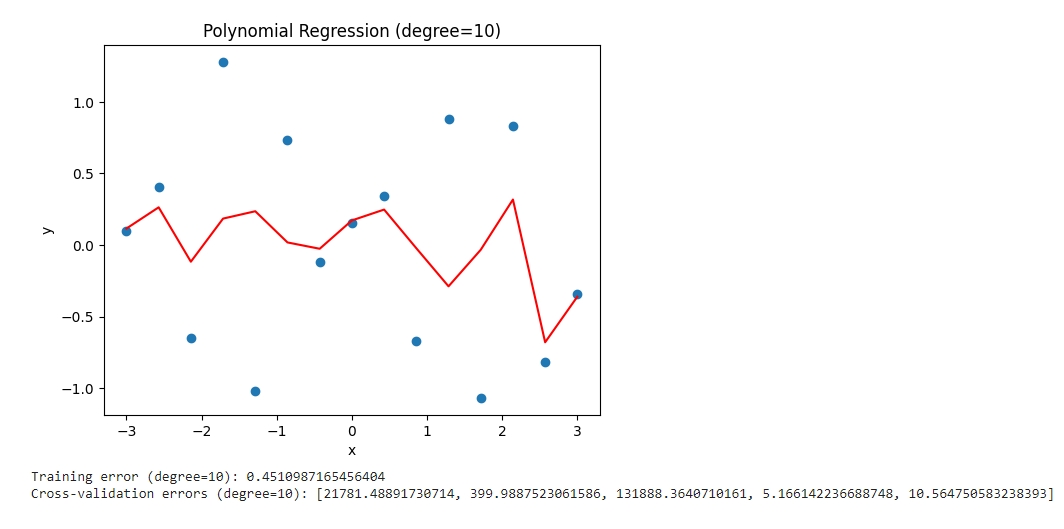
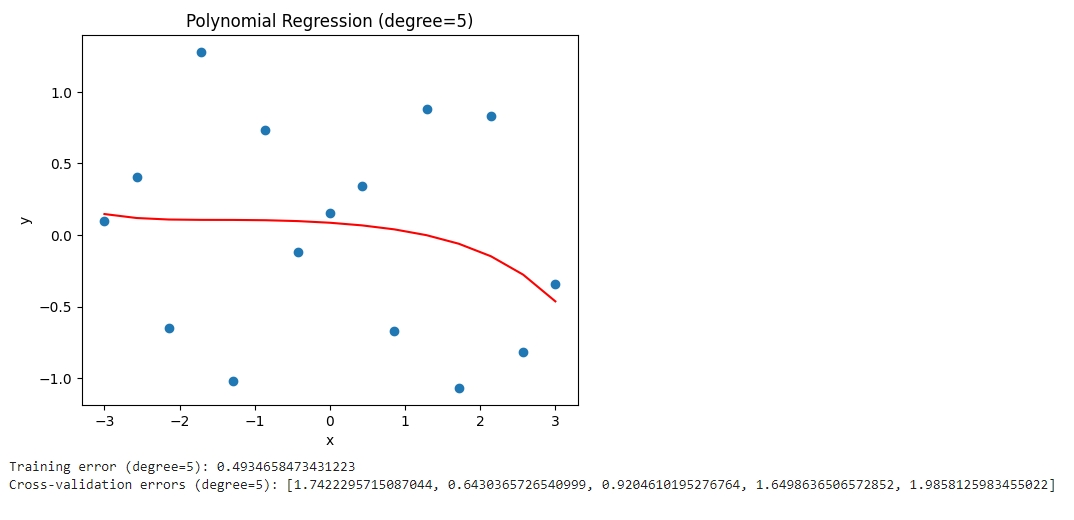
在第12行修改成y = np.sin(2 \* np.pi \* x) + noise

1. 其餘部分皆與要求2和3相同，唯一不同的是，題目要求比對linear和polynomial regression的結果，因此degrees[]的部分多添加了1進去做觀察和比較 。



▲圖4.1

## 下圖4.2為要求4的執行結果:



▲圖4.2

## 要求4問題討論:

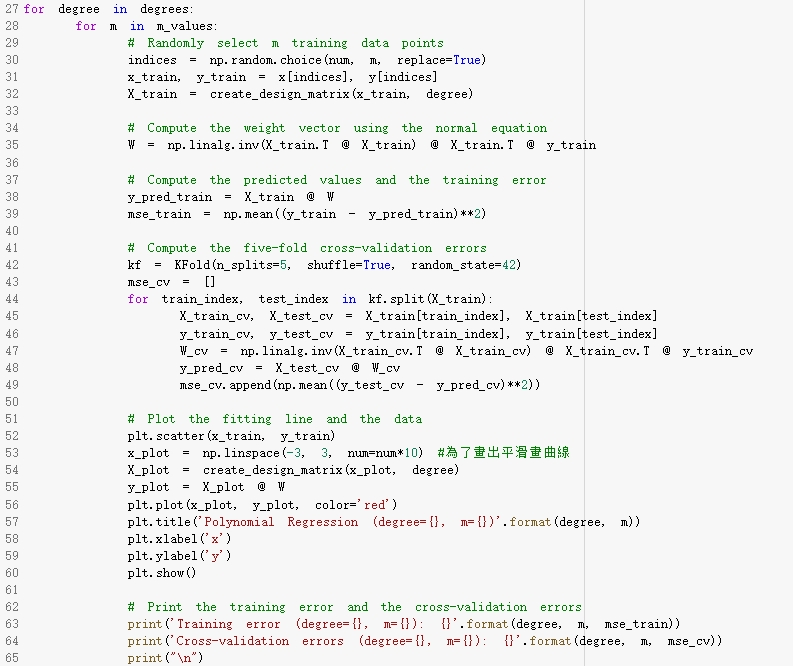
由以上圖4.2執行結果可知，當degree為1時，多項式回歸模型的訓練誤差約為0.50，交叉驗證誤差約為0.57~1.02。當degree為5時，多項式回歸模型的訓練誤差約為0.49，交叉驗證誤差約為0.64~1.99。當degree為10時，多項式回歸模型的訓練誤差約為0.45，交叉驗證誤差約為5.17~2.18x10^5。當degree為14時，多項式回歸模型的訓練誤差為6.75x10^-13，交叉驗證誤差約為5.47~4.76x10^8。可以發現，當degree=1時，線性回歸的表現比多項式回歸好，雖然training error略大於多項式回歸，但看到cross-validation errors相對多項式回歸就小非常多。

隨著degree的增加，訓練誤差逐漸降低，而交叉驗證誤差則先下降後上升。當degree為14時，雖然模型的訓練誤差最小，但交叉驗證誤差反而變得非常大，這表示模型出現了過度擬合現象，對新的未見過的資料的預測能力變得非常差。因此，該選擇degree的適當值非常的重要，以避免過度擬合或欠擬合的問題。

# 要求5:

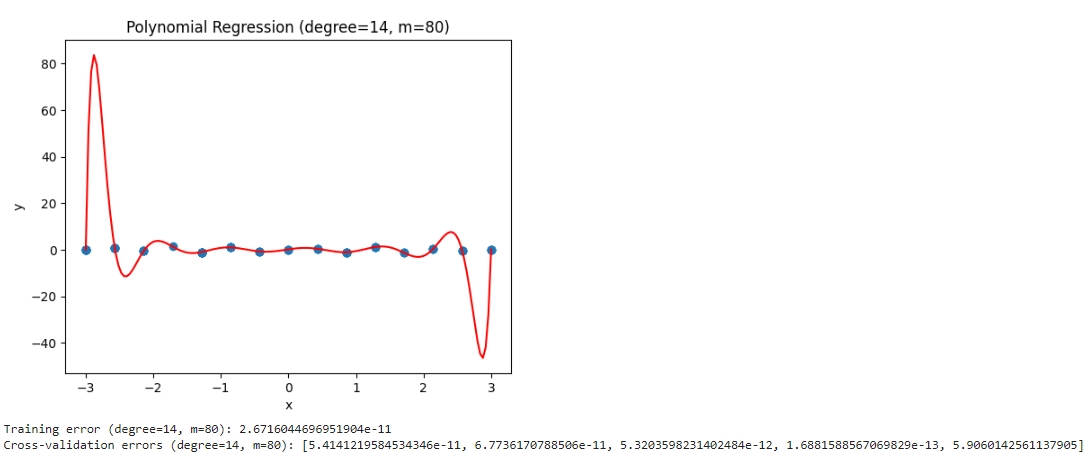
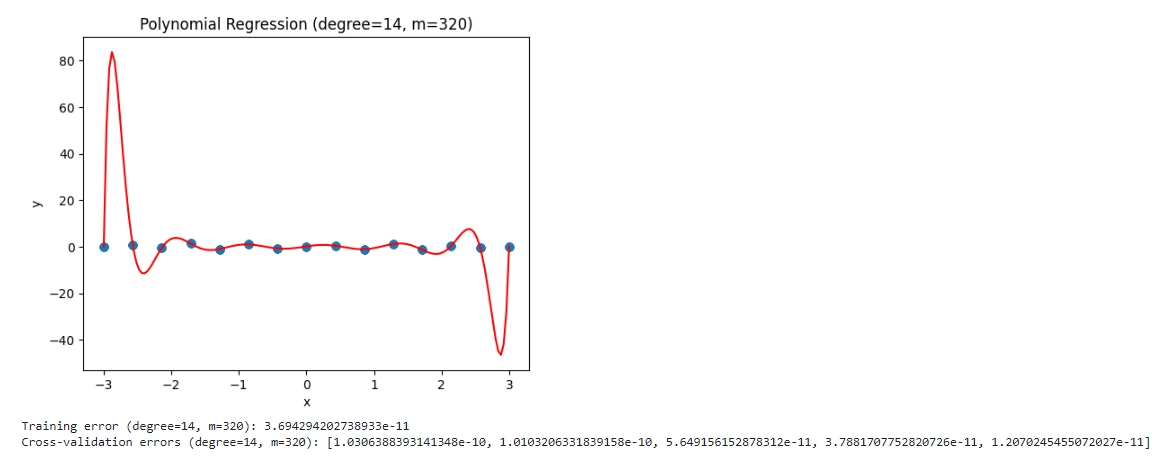
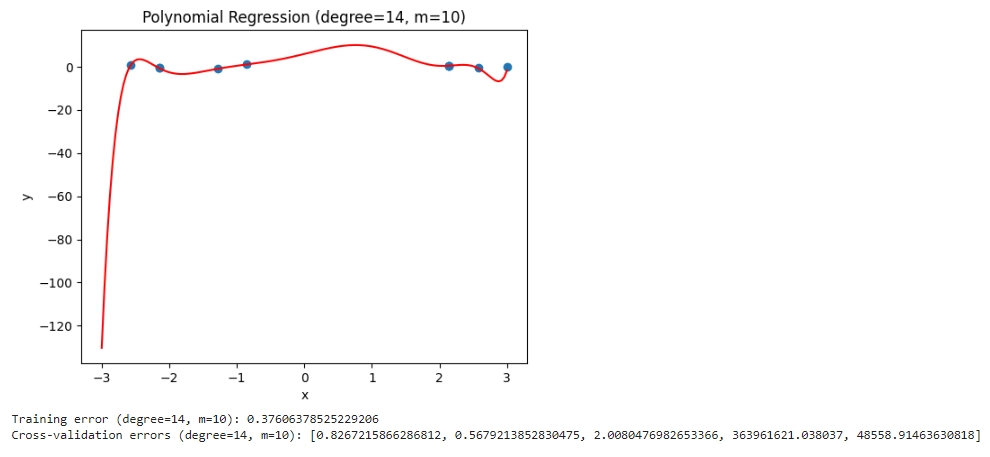
下圖5.1為要求5的code。

1. 在第8行定義m\_values = [10, 80, 320]是為了題目要求以不同的訓練數據點數量，並在26行這裡必須把degrees固定為14，然後在第27~28行使用迴圈來執行不同的次方次數(degree)和不同的訓練資料筆數(m)的多項式迴歸，接下來30行使用 np.random.choice() 函數從 num 個數據點中隨機選取 m 個數據點的索引，並使用這些索引從 x 和 y 中選取對應的 m 個數據點，作為本次多項式迴歸的訓練數據集，之後呼叫create\_design\_matrix() 函數將訓練數據集轉換為設計矩陣，接下來的部分基本上與要求4相同，在此不再贅述。



▲圖5.1

## 下圖5.2為要求5的執行結果:



▲圖5.2

## 要求5問題討論:

由上圖5.2結果可知，這個結果顯示了隨著 m 增加，訓練誤差和交叉驗證誤差都趨於穩定，這是因為隨著 m 的增加，我們使用的樣本數增加，使得我們可以更好地捕捉到數據的真實變化。

當m=10時，訓練誤差為0.37606378525229206，交叉驗證誤差則在第4和第5中非常高，分別為363961621.038037和48558.91463630818。這表示當數據量太少時，模型很容易過度擬合訓練集，因此在未見過的數據上表現較差。

當m=80和m=320時，訓練誤差非常小，分別為2.6716044696951904e-11和3.694294202738933e-11。交叉驗證誤差也顯著降低，並且比m=10時的結果要好得多。這表明當數據量增加時，模型可以更好地泛化到未見過的數據，並且不容易過度擬合訓練集。

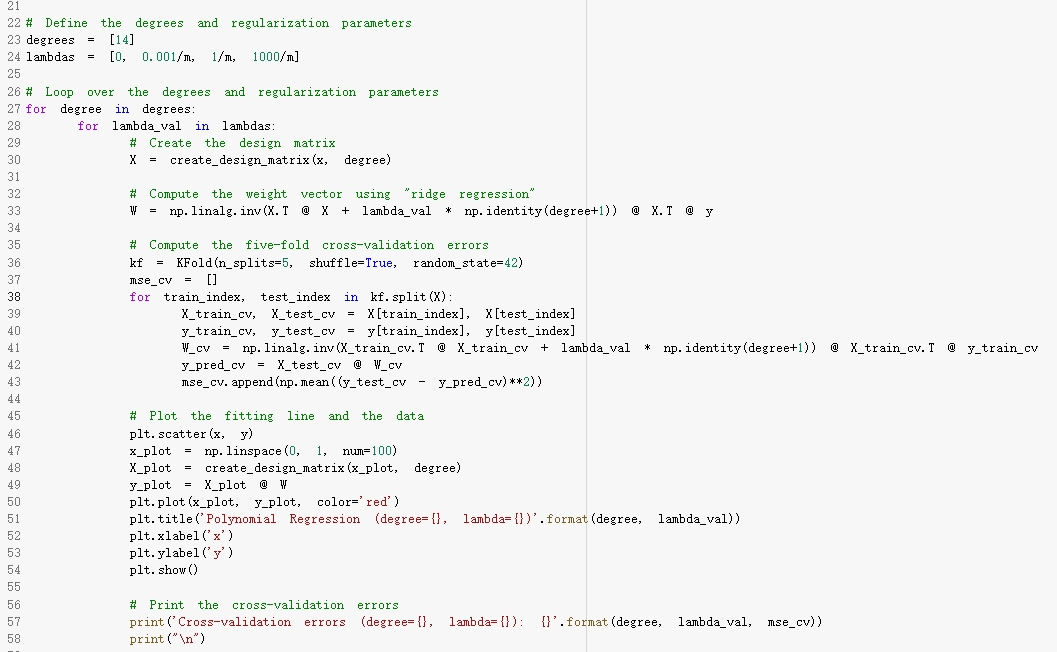
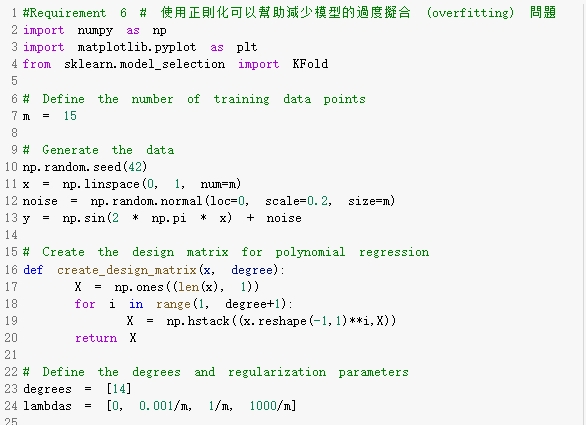
此外，再回到**要求4**去查看**，**當使用更高的多項式次數時，訓練誤差會優於交叉驗證誤差，這是因為使用更高的多項式次數會使模型變得更複雜，更容易對訓練數據過度擬合，而無法泛化到新的數據。因此，當多項式次數過高時，模型的泛化性能將會變得較差，這就是訓練誤差和交叉驗證誤差之間差異變大的原因。

# 要求6:

下圖6.1為要求6的code。

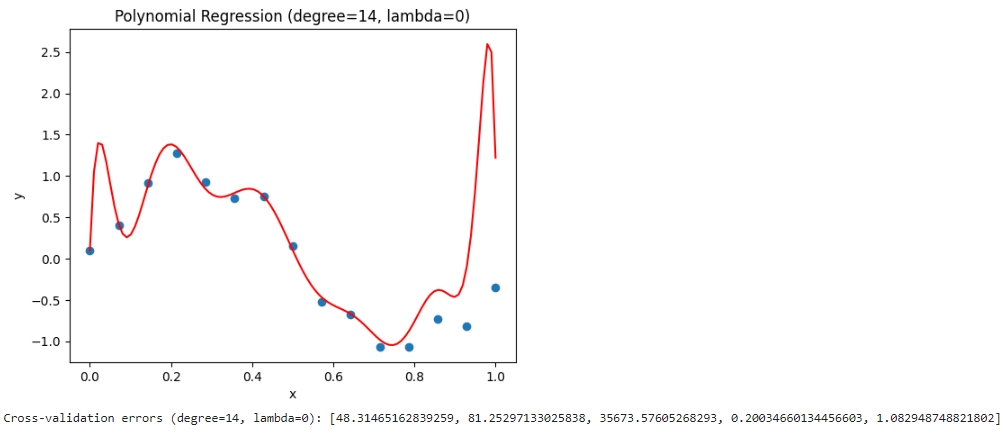
1. 這段程式碼是由**要求4**做延伸，使用正則化（regularization）的方法，對一個具有14次多項式回歸（polynomial regression）的模型進行訓練。透過調整正則化參數λ的大小（第24行，分別設為0、0.001/m、1/m、1000/m），來比較不同正則化強度對模型的影響。其中，m為樣本數，本例中為15個。

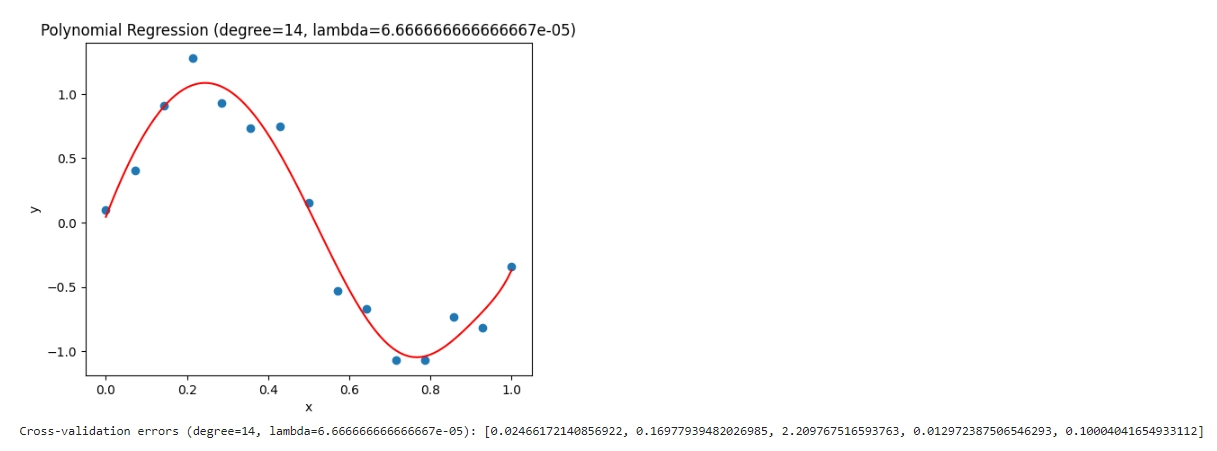
1. 再來與**要求**4不同的地方為第33行W的計算方式，這裡採用的是"ridge regression"，它是一種正規化線性回歸的方法，可以有效解決過度擬合(overfitting)的問題。它的公式為：W = (X.T @ X + λ \* I)^(-1) @ X.T @ y。
2. 接下來的步驟都相同，利用K-fold交叉驗證（K-Fold Cross Validation）來計算五次驗證集的均方誤差（MSE），從而評估模型的泛化能力。最後，根據模型的訓練結果，繪製相關的圖和多項式回歸的線。

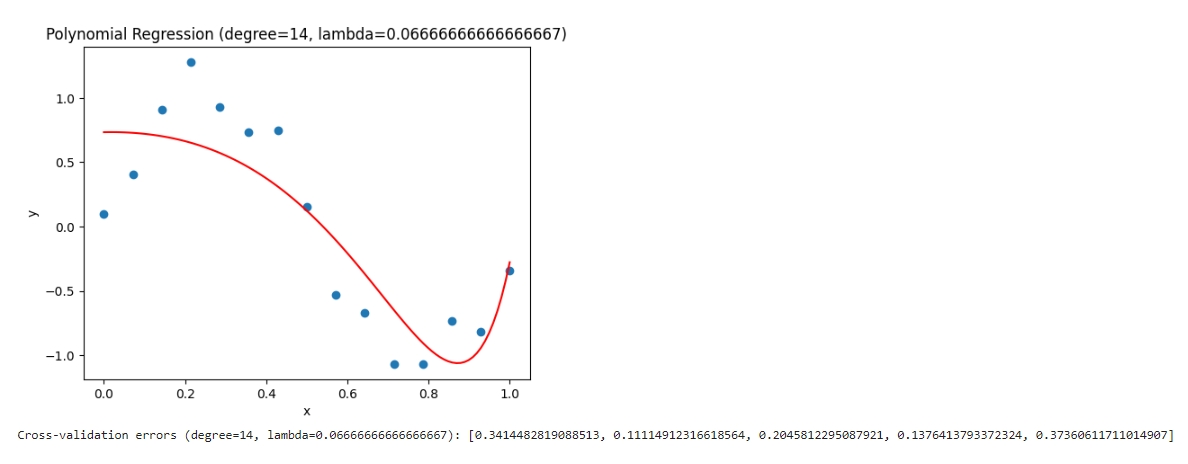
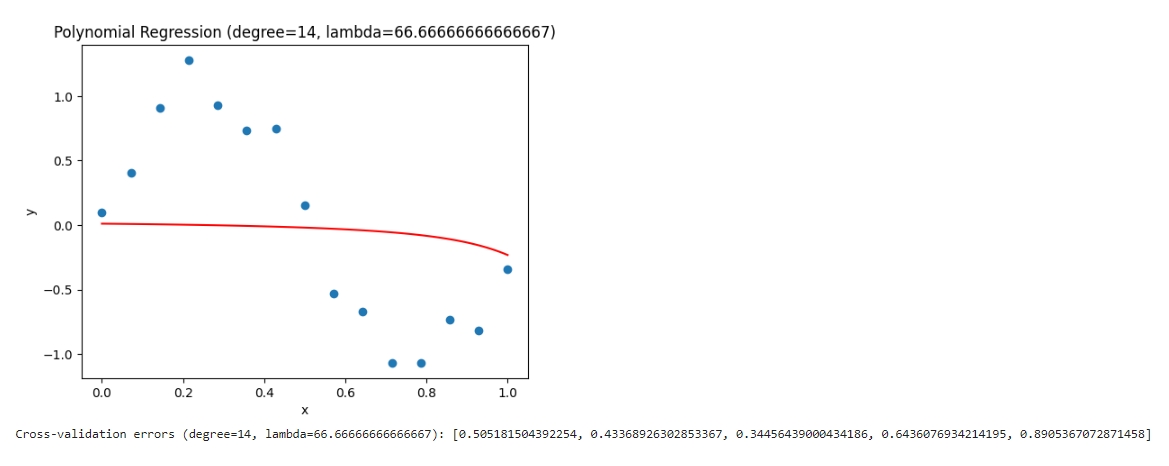


▲圖6.1

## 下圖6.2為要求6的執行結果:







▲圖6.2

## 要求6問題討論:

由上圖6.2觀察得知，當lambda為0時，模型沒有進行正則化，結果出現了明顯的過度擬合，因為交叉驗證誤差很大。當lambda增加到0.001/m時，交叉驗證誤差明顯減小，表示正則化有助於減少過擬合。當lambda增加到1/m時，交叉驗證誤差繼續減小，但是當lambda增加到1000/m時，交叉驗證誤差又有所增加，這可能是因為太強的正則化導致欠擬合的原因。因此得知，可以根據交叉驗證誤差選擇合適的lambda值來平衡擬合效果和泛化性能。