**人工智慧與數位學習 期中報告**

* **作業大綱**

|  |  |
| --- | --- |
| 09/13 | 請畫出各種degree的多項式回歸和訓練樣本數N的關係 |
| 09/20 | 依照投影片資料，以GD訓練出線性回歸 |
| 09/27 | 對登記的資料集 進行決策樹與隨機森林的分析 |
| 10/04 | 隨機森林Random Forest或Adaboost，對參數進行分析 |
| 10/11 | KNN, CART, LR 比較k值 |
| 10/25 | SVR, SVC |

* **各種degree的多項式回歸和訓練樣本數N的關係**

1. **分別定義兩種訓練樣本數的x和y**

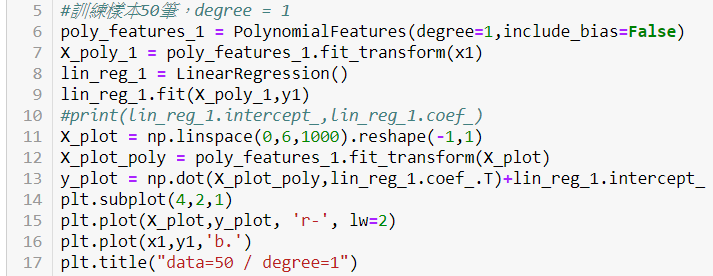
|  |  |
| --- | --- |
| 訓練樣本50筆 | 訓練樣本100筆 |
| x1 = np.linspace(0,2\*np.pi,50)  y1 = np.sin(x1)+np.random.randn(len(x1))/5.0 | x2 = np.linspace(0,2\*np.pi,100)  y2 = np.sin(x2)+np.random.randn(len(x2))/5.0 |

1. **分別計算迴歸係數和截距**

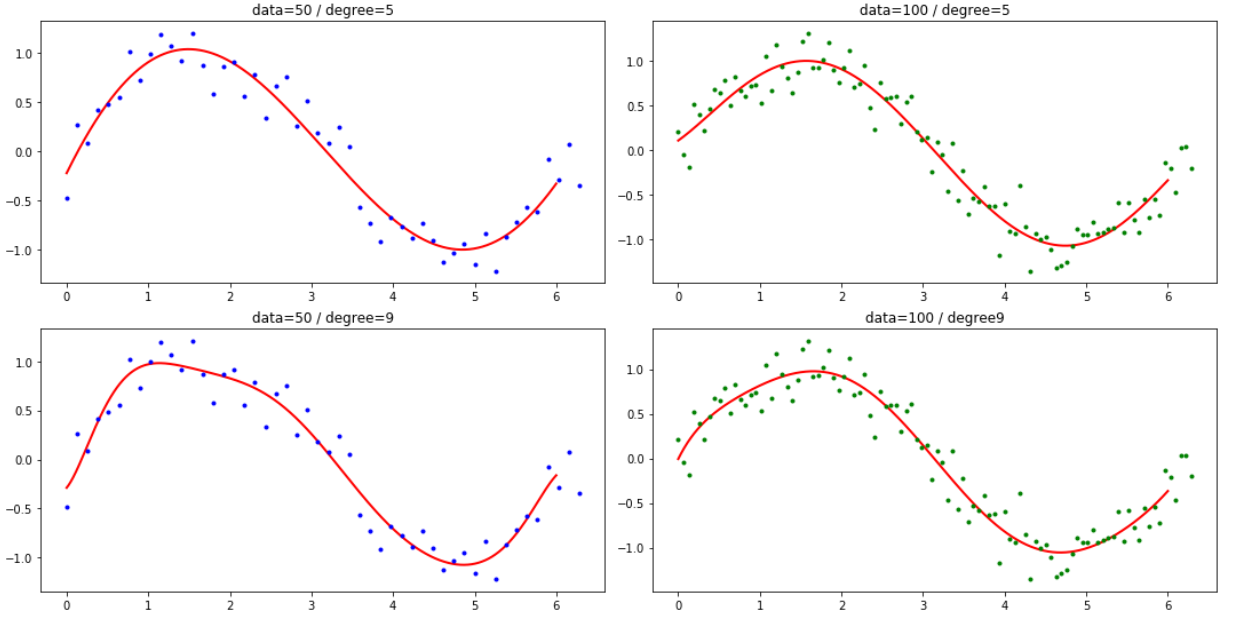
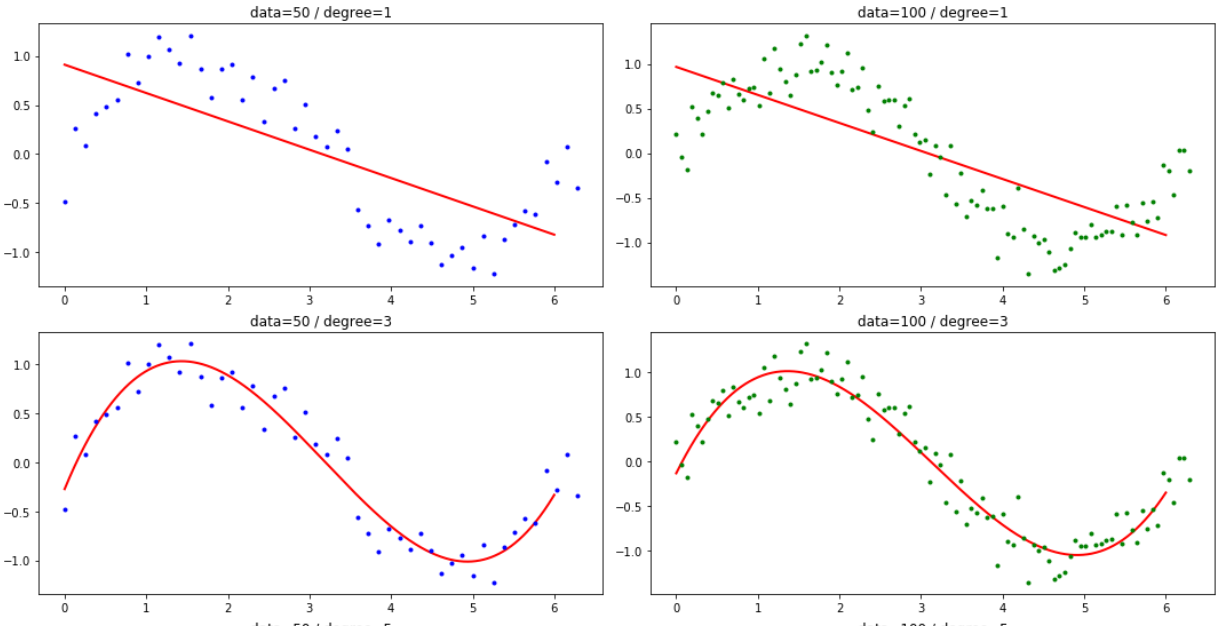
|  |  |
| --- | --- |
| 訓練樣本50筆 | 訓練樣本100筆 |
| slr = LinearRegression()  x1 = x1.reshape(-1,1)  slr.fit(x1,y1)  print("樣本數50的迴歸係數：",slr.coef\_)  print("樣本數50的截距：",slr.intercept\_) | slr = LinearRegression()  x2 = x2.reshape(-1,1)  slr.fit(x2,y2)  print("樣本數100的迴歸係數：",slr2.coef\_)  print("樣本數100的截距：",slr2.intercept\_) |
| 樣本數50的迴歸係數： [-0.28863508]  樣本數50的截距： 0.9112656660524244 | 樣本數100的迴歸係數： [-0.31414697]  樣本數100的截距： 0.9657530018885288 |

1. **使用subplots來視覺化呈現**

以訓練樣本50筆，degree = 1的作為例子，其餘的以此類推。



1. **畫出函數結果**



1. **結果分析**

由上圖發現，不管訓練樣本數為50或100，degree = 3時，訓練出來的多項式回歸線有最佳的擬合；degree = 9時，函數圖會穿過一部分的樣本點，幾乎所有的訓練樣本都落在了擬合的曲線上，訓練誤差趨近於0，可以説是近乎完美的模型了。但是，這樣的曲線與最一開始數據源差異非常大。訓練誤差非常小，但是新數據點的測試誤差非常大的情況，就叫形成過擬合(overfitting)，表示模型過於複雜，使得模型的泛化能力下降。

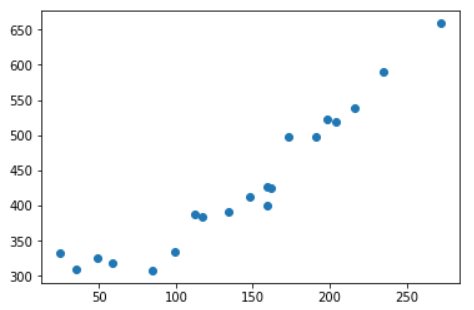
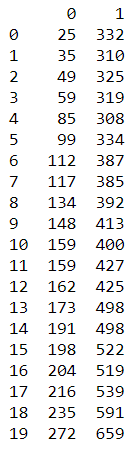
1. **解決方法**

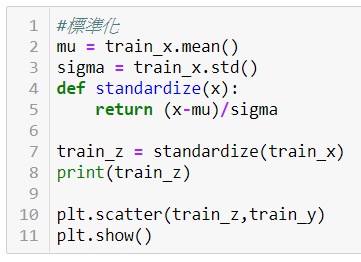
* 降低模型複雜度(縮小degree)
* 降維(減少特徵的數量)
* 增加訓練樣本
* 添加正則化項
* **梯度下降法(Gradient Descent，GD)、線性回歸(Linear Regression，LR)**

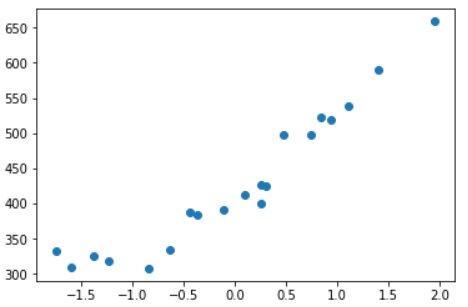
一種不斷去更新參數找解的方法，希望用此方法找到函數的局部最小值，因為梯度的方向是走向局部最大的方向，所以在梯度下降法中是往梯度的反方向走。

1. **載入資料集**

將左邊的資料用散佈圖表示



1. **標準化(standardize)**



把train\_x標準化後的定義為train\_z

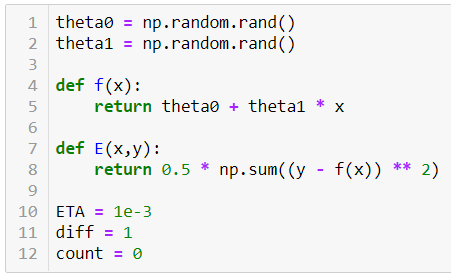
並用散佈圖表示

1. **定義參數**

隨機定義theta0和theta1

定義函數

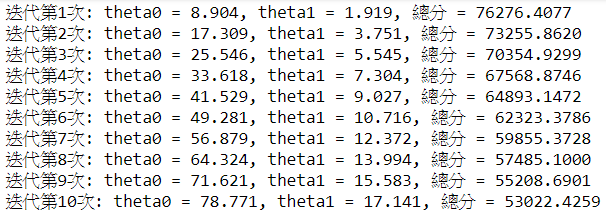
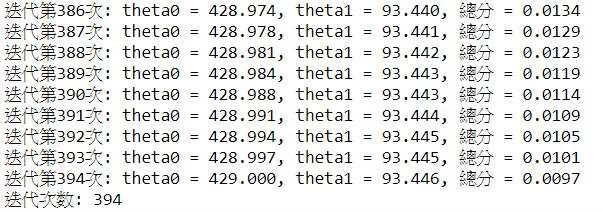
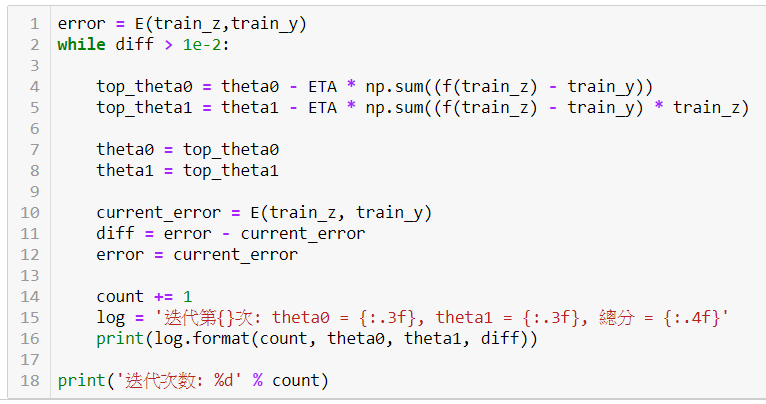
學習速率設為1e-3(0.0001)



1. **計算迭代的次數**

定義error

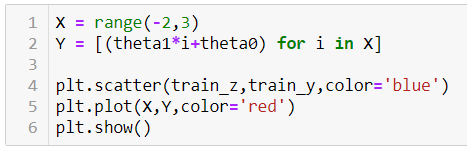
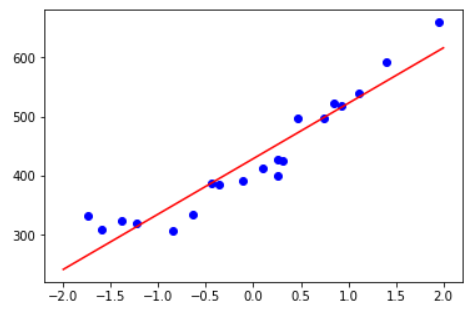
定義diff



…

1. **結果分析**

利用每次迭代的theta0和theta1來訓練迴歸，下圖為訓練後的迴歸線。

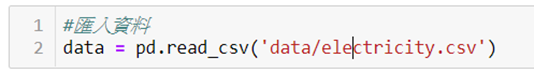


* **決策樹(Decision Tree)**

在學習的時候就需要考慮特徵節點的選取順序，常用的度量方式包括信息熵(Information Gain)和基尼不純性(Gini Impurity)，性能分析會使用準確率(Accuracy)、召回率(Recall)、精確率(Percision)以及F1指標。

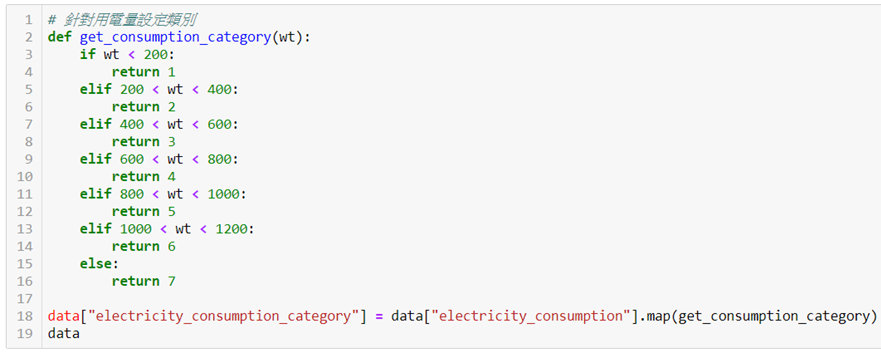
1. **選擇資料集**

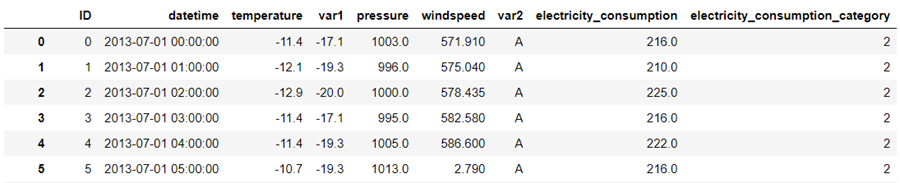
以electricity.csv這個dataset作為例子



1. **資料預處理**
2. 資料集欄位資料進行分類

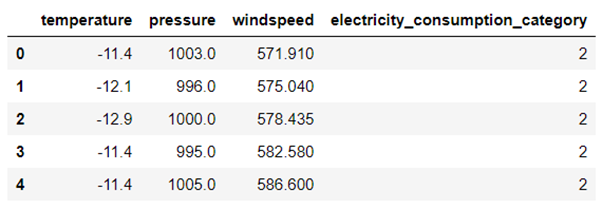
把原資料集electricity\_consumption依據使用的電量進行分類，分成7類，把分類的結果在資料集中新增一個欄位electricity\_consumption\_category，如下圖所示。





1. 只列出需要使用的欄位



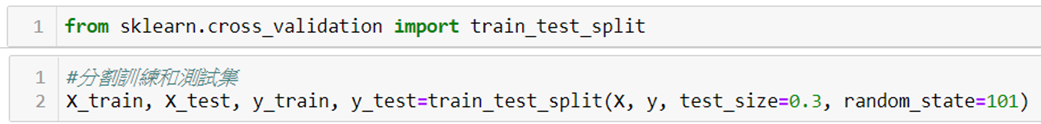


1. 定義X和y (X移除electricity\_consumption\_category)

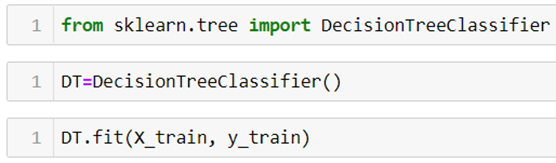


1. 分割訓練集和測試集

將資料的70%拿出來train，剩下的30％用來檢測train的好壞



1. **執行決策樹**
2. 載入決策樹函式DecisionTreeClassifier

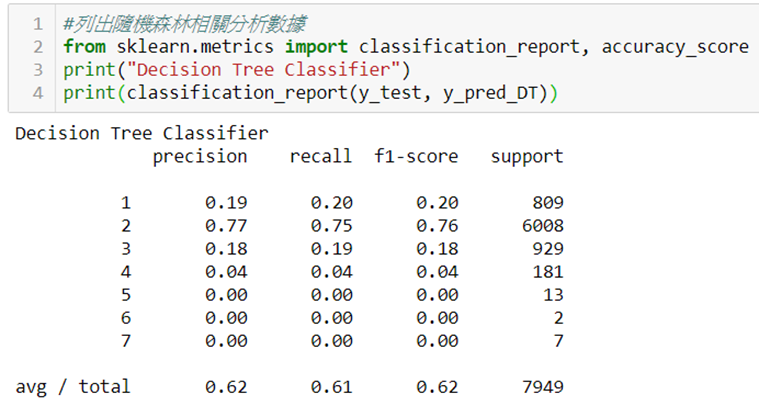


1. 使用y\_train來作預測

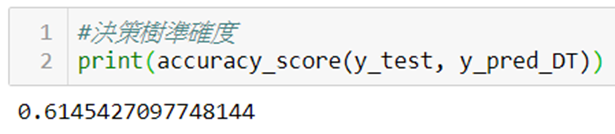


1. 列出隨機森林相關分析數據

分別有召回率(Recall)、精確率(Percision)以及F1指標



計算決策樹的準確度(Accuracy)



* **隨機森林(Random Forest，RF)**

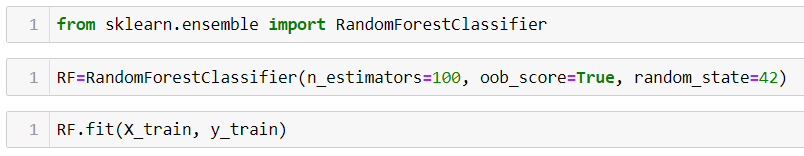
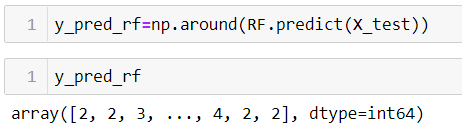
利用相同的訓練數據來建立多個分類模型，以少數服從多數的原則做出最終的分類決策，而隨機森林分類器就是這一類型的代表。在相同的訓練數據上同時建立多棵決策樹，一棵標準的決策樹根據每維特徵來對預測結果的影響程度進行排序，進而決定不同特徵從上至下構建分裂節點的順序，但如果都這樣，那麼每一棵樹都類似，那麼就丟失了多樣性，而隨機森林分類器在構建過程中放棄了這一固定的排序算法，轉而隨機選取特徵。

1. **選擇資料集**

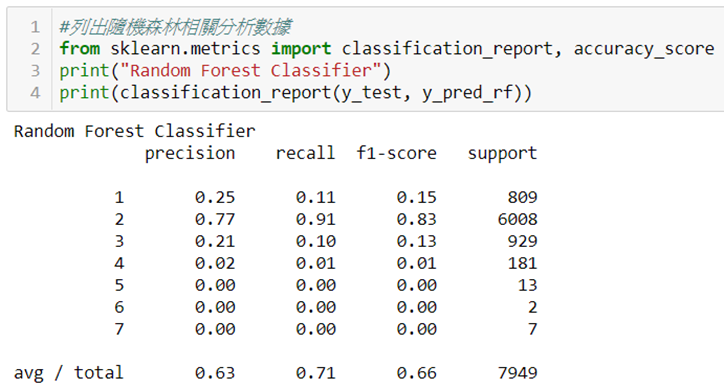
與前一節相同

1. **資料預處理**

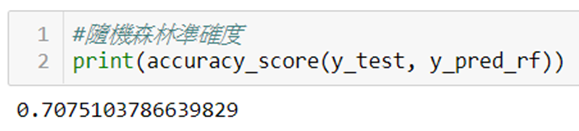
與前一節相同

1. **執行隨機森林**
2. 設定隨機種子
3. 載入隨機森林函式RandomForestClassifier
4. 使用X\_test來作預測
5. 列出隨機森林相關分析數據

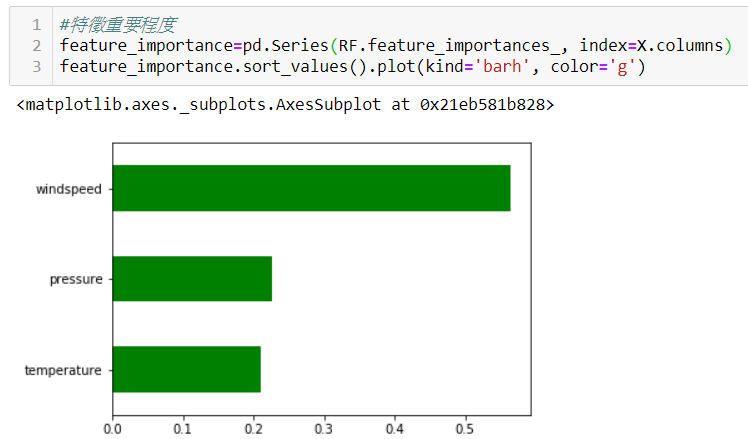
分別有召回率(Recall)、精確率(Percision)以及F1指標



計算決策樹的準確度(Accuracy)

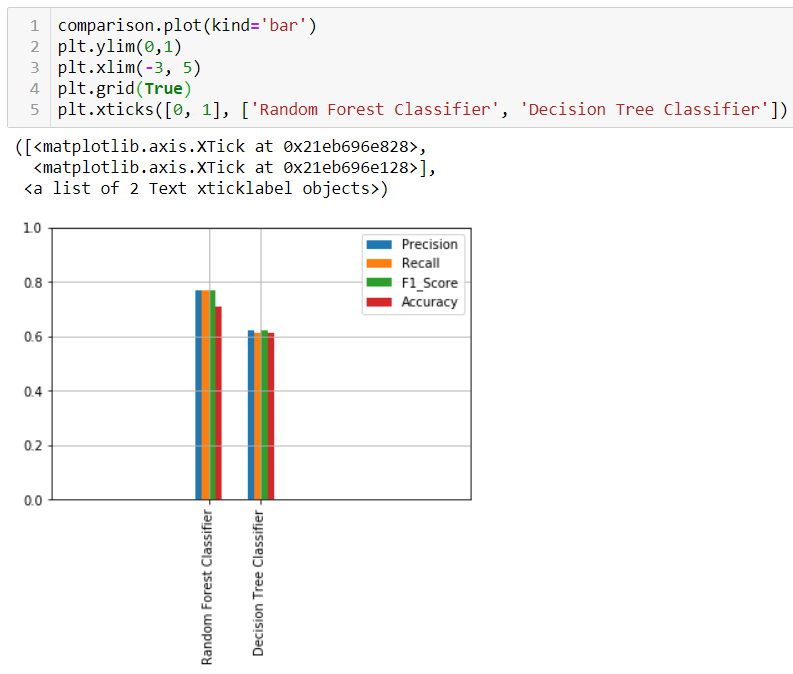
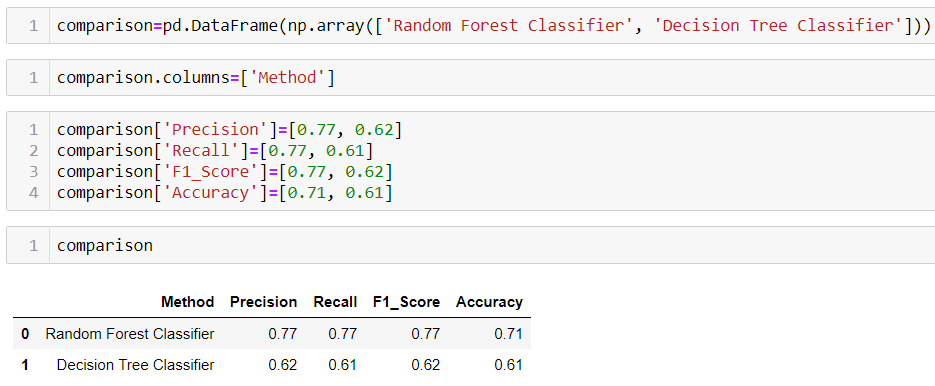


1. **分析特徵的重要程度**



資料集中3個特徵有windspeed、pressure、temperature，由上圖可以發現，windspeed對預測結果有較大的影響力。

* **比較隨機森林(Random Forest)和決策樹(Decision Tree)**



以electricity.csv這個資料集來說，

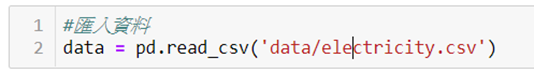
可以發現Random Forest的性能比較好。

* **AdaBoost**

Adaboost是一種反覆運算演算法，其核心思想是針對同一個訓練集訓練不同的分類器（弱分類器），然後把這些弱分類器集合起來，構成一個更強的最終分類器（強分類器）。Adaboost演算法本身是通過改變資料分佈來實現的，它根據每次訓練集之中每個樣本的分類是否正確，以及上次的總體分類的準確率，來確定每個樣本的權值。將修改過權值的新資料集送給下層分類器進行訓練，最後將每次得到的分類器最後融合起來，作為最後的決策分類器。

1. **選擇資料集**

以electricity.csv這個dataset作為例子

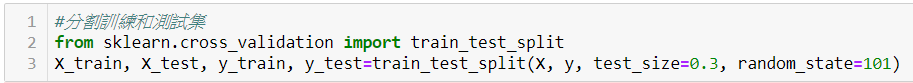


1. **資料預處理**

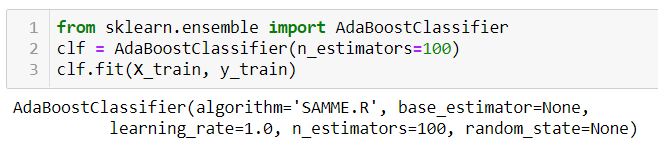
與前一節相同

1. **建立訓練與測試資料**

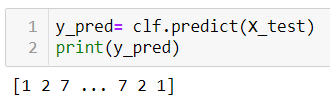
將資料的70%拿出來train，剩下的30％用來檢測train的好壞



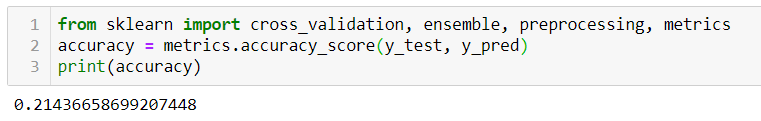
1. **建立boosting模型**
2. 載入隨機森林函式AdaBoostClassifier



1. 預測y\_pred



1. 計算準確率



* **K-近鄰演算法(K-Nearest Neighbor，KNN)**

對於一個待分類的樣本，選取待分類樣本在特徵空間中距離最近的K個已標記樣本作為參考，來幫助做出分類決策，因此可以得知K值不同，得到的分類器也可能不同，過程中沒有參數訓練過程，所以如果數據集非常的龐大，那麼會增加計算的複雜度。

* **CART(Classification and Regression Trees)**

CART是一種產生二元樹的技術，以吉尼係數(Gini index)做為選擇屬性的依據。CART與ID3、C4.5、C5.0演算法的最大相異之處是，其在每一個節點上都是採用二分法，也就是一次只能夠有兩個子節點，ID3、C4.5、C5.0則在每一個節點上可以產生不同數量的分枝。

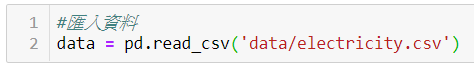
* **線性回歸(Logistic Regression，LR)**

在線性分類器的時候，為了把實數域的結果映射到(0,1)區間，引入了邏輯斯函數，而在線性回歸問題中，預測目標直接就是實數域上的數值，因此優化目標更為簡單，即最小化預測結果與真實值之間的差值。

* **比較KNN、CART、LR**

1. **選擇資料集**

以electricity.csv這個dataset作為例子

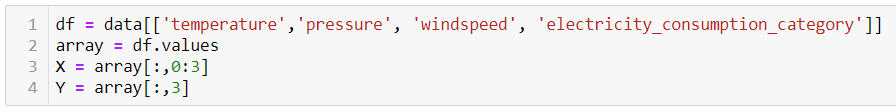


1. **資料預處理**

針對用電量設定類別



挑選4個欄位作為主要依據

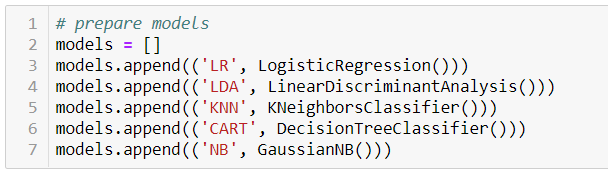


1. **設置隨機種子**



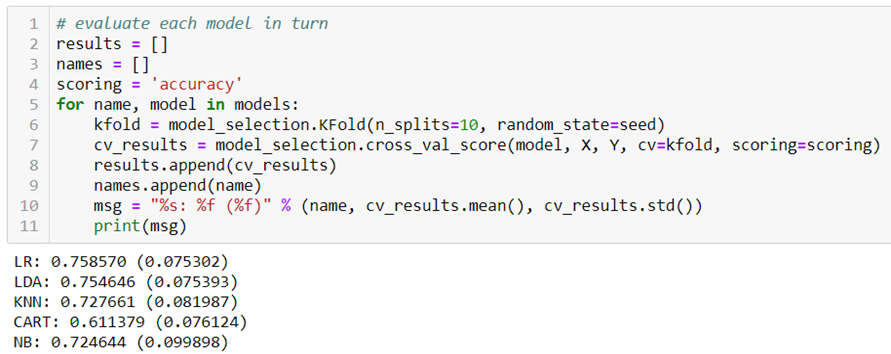
1. **各類模型比較**

定義一個models陣列來存放比較的結果



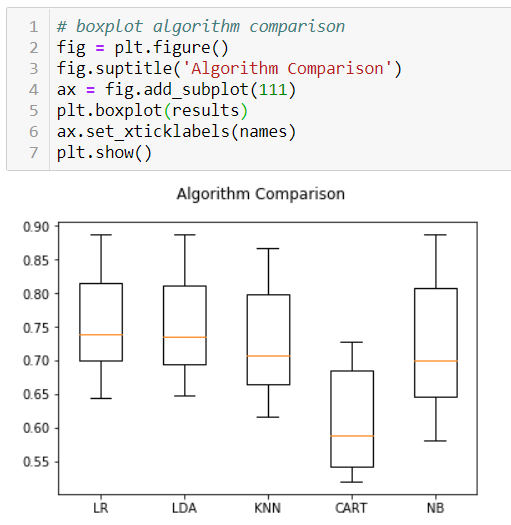
1. **各類模型比較結果**

使用的評估依據是accuracy準確率，並搭配10次交叉驗證，印出各類別的平均準確率和括號中的標準差。



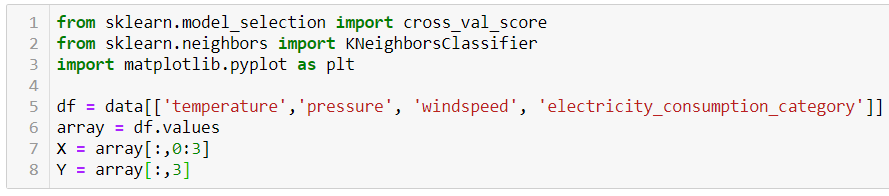
1. **視覺化表現**

利用箱體圖(boxplot)表現，從下到上五條線分別表示準確度的最小值、下四分位數、中位數、上四分位數和最大值。



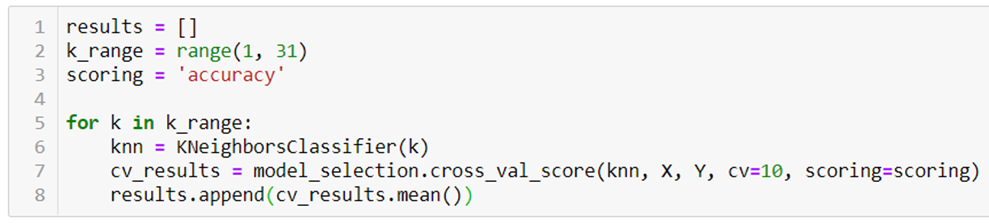
可以發現以LR的表現最好。

1. **KNN比較不同的K值**
2. 匯入模組，使用4個欄位並分別定義X和Y

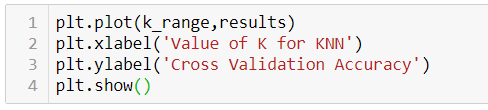


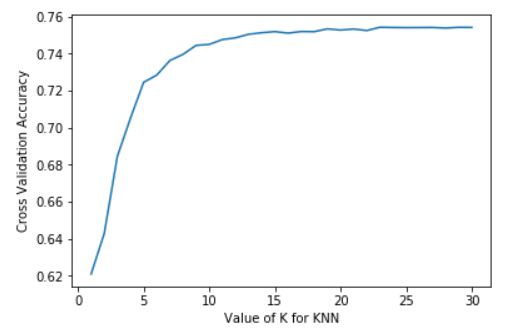
1. 設定k的範圍

這邊把range設在1~31這區間，cv=10代表10次交叉驗證法，scoring = 'accuracy'代表使用準確率來評估每一個K值。



1. 視覺化





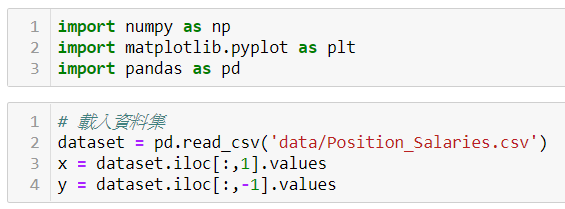
由上圖發現，隨著K值的增加，準確率也會提升，但到了一定的K值，準確率成長的趨勢比較緩慢。

* **支援向量迴歸(Support Vector Regression，SVR)**

它是使用SVM來擬合曲線，做迴歸分析。

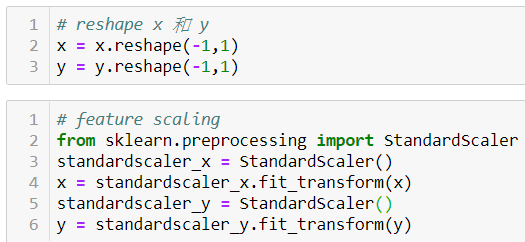
1. **匯入模組、載入數據集**

以Position\_Salaries.csv數據集為例

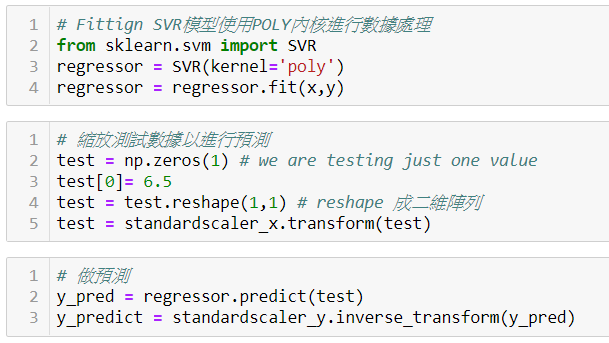


1. **資料預處理**

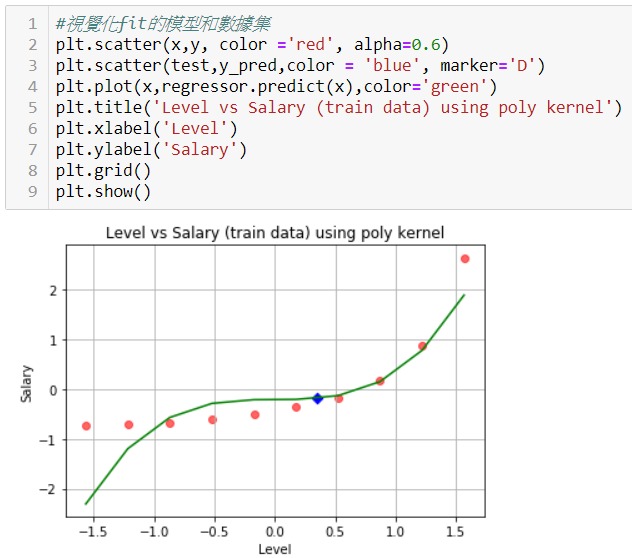
利用reshape更改數组的形状，利用StandardScaler來計算在一個訓練集上的平均值和標準差，使用這模組的好處是可以保存訓練集中的參數直接使用其物件轉換測試集資料。



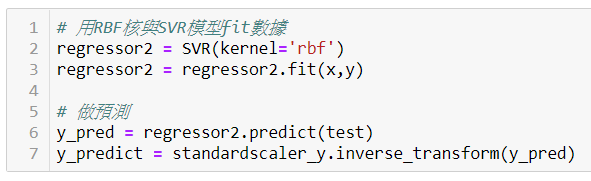
1. **數據處理(poly kernel、縮放測試數據)**



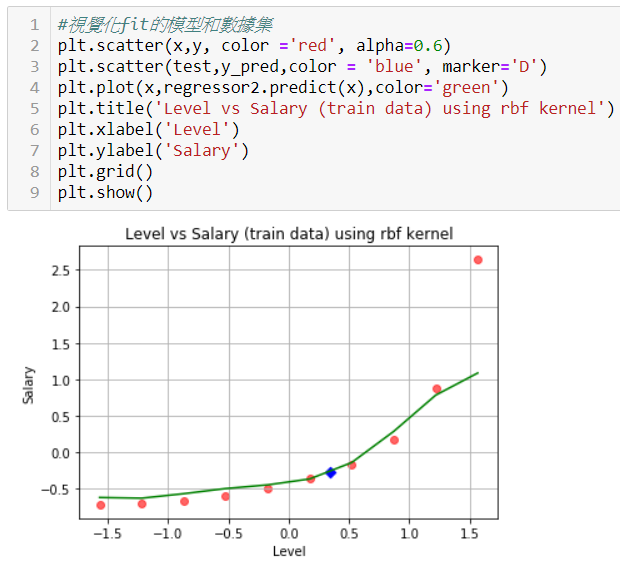
1. **poly kernel視覺化**



1. **數據處理(RBF kernel)**



1. **RBF kernel視覺化**

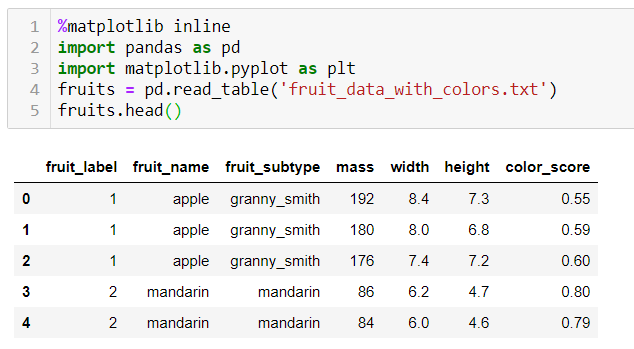
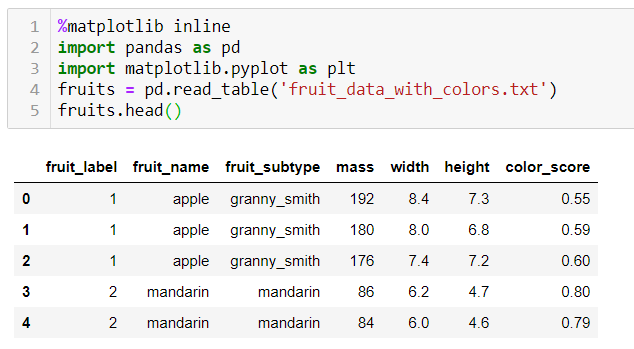


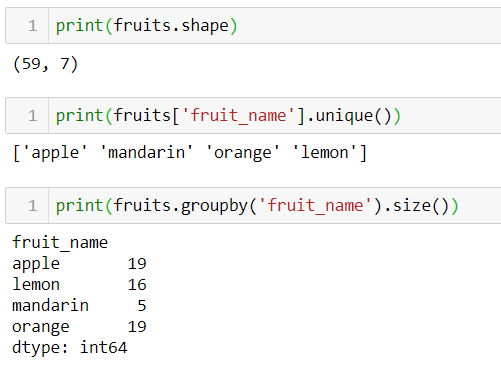
* **支援向量分類 (Support Vector Classification，SVC)**

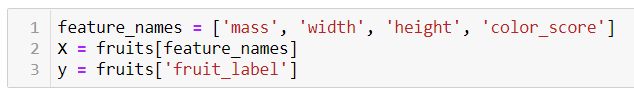
它是使用SVM來擬合曲線，做分類分析。

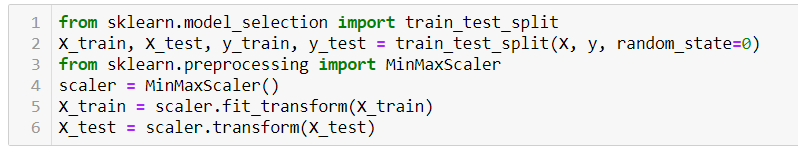
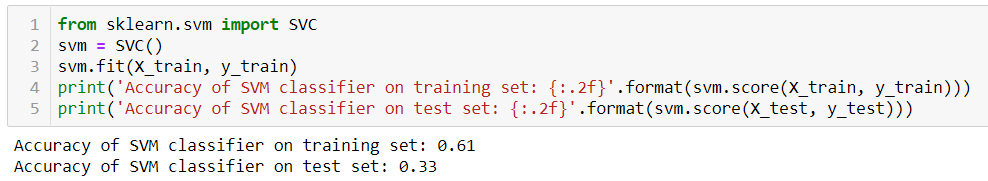
1. **匯入模組、載入數據集**

以fruit\_data\_with\_colors.txt數據集為例



1. **數據集資料統計(資料維度、欄位資料查詢)**
2. **定義新欄位**

新增的欄位命名為feature\_names

1. **分割訓練集和數據集**
2. **計算SVC的準確率**