RegressionTree

2018 年 6 月 22 日

1 決策樹 (迴歸)

1.1 介紹

在我們的上一節，提到 CART 決策樹不僅可以用於分類，還可以用在迴歸問題，我們就一起來試試看

1.2 資料集

scikit-learn 內建的 Boston 地產資料集

1.3 開始撰寫程式

1.3.1 Step 0. 讀入我們的波士頓地產數據集作為練習

```
In [1]: from sklearn.datasets import load_boston
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        # 為了顯示的漂亮，我刻意的把印出來的 row 只顯示 15 個和 column 只顯示十個
        # 大家練習的時候可以去掉下面兩行
```

```
pd.set_option('display.max_rows', 15)
pd.set_option('display.max_columns', 10)

boston = load_boston()
df = pd.DataFrame(boston['data'], columns = boston['feature_names'])
df['target'] = boston['target']
# 如果你想要的話, 你可以把他輸出成 csv 觀察看看
# df.to_csv('boston.csv')
df
```

Out[1]:

	CRIM	ZN	INDUS	CHAS	NOX	...	TAX	PTRATIO	B	LSTAT	\
0	0.00632	18.0	2.31	0.0	0.538	...	296.0	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0.0	0.469	...	242.0	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0.0	0.469	...	242.0	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0.0	0.458	...	222.0	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0.0	0.458	...	222.0	18.7	396.90	5.33	
5	0.02985	0.0	2.18	0.0	0.458	...	222.0	18.7	394.12	5.21	
6	0.08829	12.5	7.87	0.0	0.524	...	311.0	15.2	395.60	12.43	
..	...	...	...	...	...	...	...	...	...	...	
499	0.17783	0.0	9.69	0.0	0.585	...	391.0	19.2	395.77	15.10	
500	0.22438	0.0	9.69	0.0	0.585	...	391.0	19.2	396.90	14.33	
501	0.06263	0.0	11.93	0.0	0.573	...	273.0	21.0	391.99	9.67	
502	0.04527	0.0	11.93	0.0	0.573	...	273.0	21.0	396.90	9.08	
503	0.06076	0.0	11.93	0.0	0.573	...	273.0	21.0	396.90	5.64	
504	0.10959	0.0	11.93	0.0	0.573	...	273.0	21.0	393.45	6.48	
505	0.04741	0.0	11.93	0.0	0.573	...	273.0	21.0	396.90	7.88	

	target
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2
5	28.7
6	22.9
..	...
499	17.5

```
500    16.8
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9

[506 rows x 14 columns]
```

1.3.2 Step 1. 先畫個圖

相關係數: 兩個東西的相關性

完全正相關 (兩個東西總是一起上升): 1

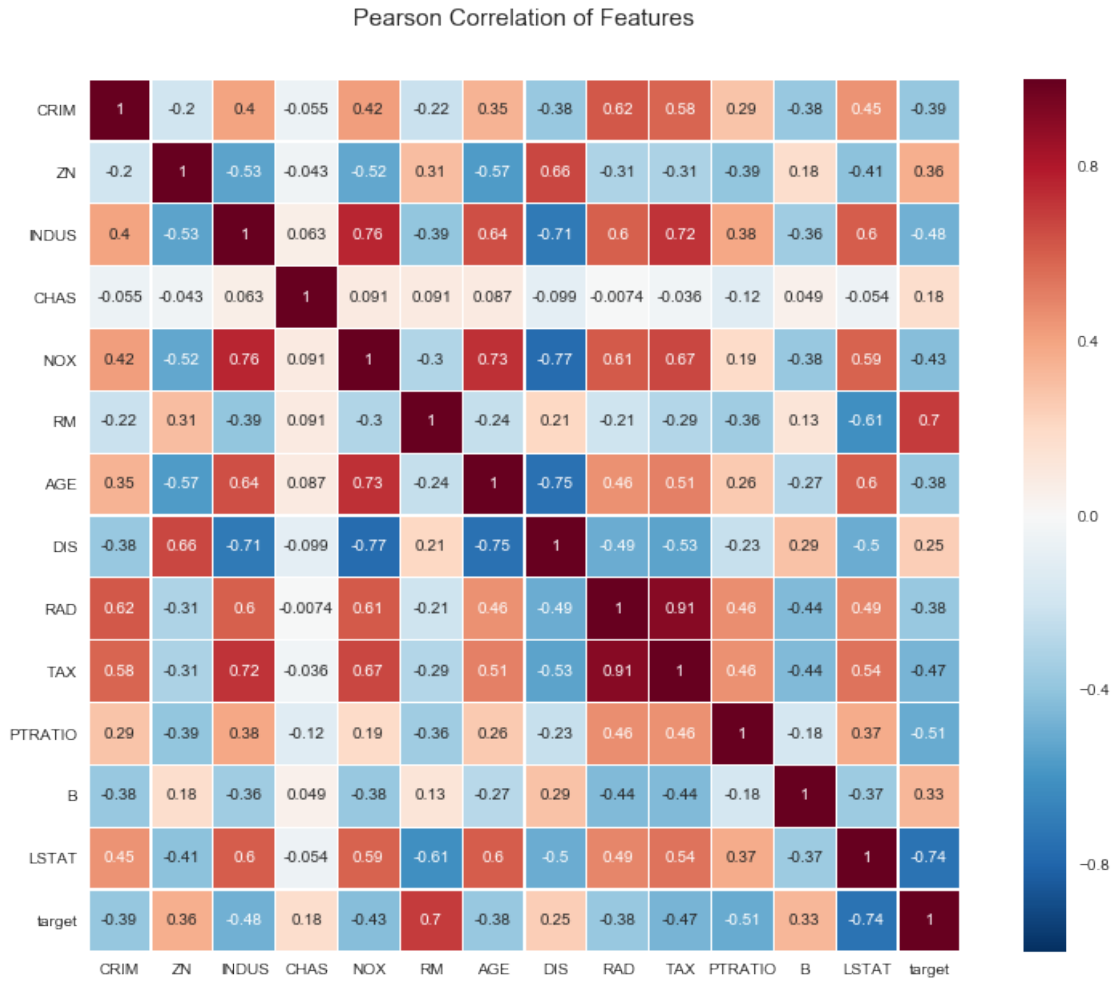
完全正相關 (一個東西上升的時候, 另一個總是下降): -1

一樣先畫個圖試試看

補充：你可以使用 https://matplotlib.org/2.0.2/examples/color/colormaps_reference.html 選擇一個你喜歡的調色盤放入 cmap 參數

```
In [2]: plt.figure(figsize=(14,10))
        plt.title('Pearson Correlation of Features', y=1.05, size=15)
        sns.heatmap(df.astype(float).corr(),linewidths=0.1,
                    square=True, linecolor='white', annot=True)
```

```
Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x10de5dd30>
```



一樣你發現了，像 CHAS, DIS 這些特徵影響的幅度應該都是比較小的 RM 和 LSTAT 的幅度應該是比較大的

1.3.3 Step 2. 訓練模型

我們使用 DecisionTreeRegressor 來訓練
一樣的步驟

1. 創好一個 Regressor
2. 使用 fit 將你要訓練的數據餵進來

```
In [3]: # 我們把我們擁有的資料集分成兩份，一份測試，一份訓練
        from sklearn.model_selection import train_test_split
        # 把資料分成兩部分 (1. 訓練資料 2. 測試資料)
```

```
        data_train, data_test, target_train, target_test = train_test_split(boston['data'],
                                                                            boston['target'],
                                                                            test_size=0.1)
```

```
In [4]: from sklearn.tree import DecisionTreeRegressor
        regr = DecisionTreeRegressor(max_depth = 3)
        regr.fit(data_train, target_train)
```

```
Out[4]: DecisionTreeRegressor(criterion='mse', max_depth=3, max_features=None,
                    max_leaf_nodes=None, min_impurity_split=1e-07,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                    splitter='best')
```

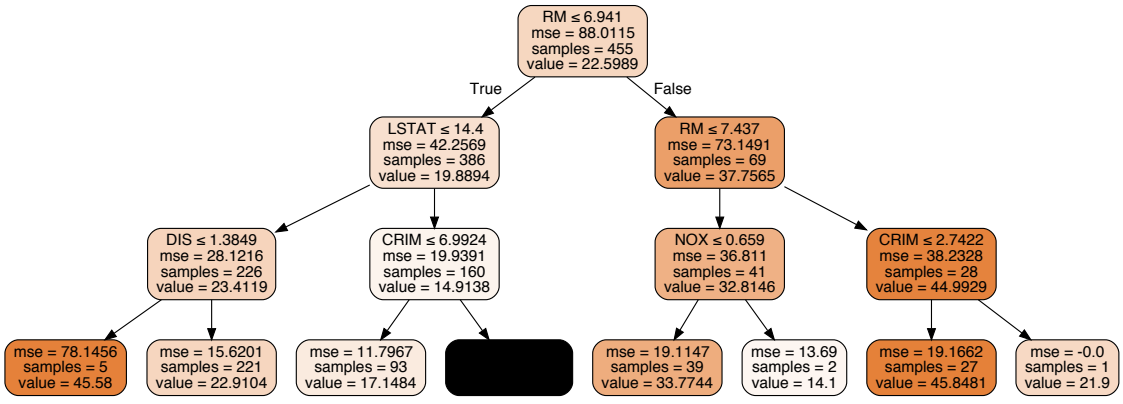
訓練完畫個圖吧，我們使用第三方軟體 graphviz 以及 python 函式庫來操作 graphviz

請完成下面的安裝步驟 1. 請來到 http://www.graphviz.org/download/ 安裝 graphviz 到你的電腦 2. 請使用 pip 或者是 pycharm 安裝好 graphviz 函式庫

```
In [5]: from sklearn.tree import export_graphviz
        import graphviz

        dot_data = export_graphviz(regr, out_file=None,
                                   feature_names=df.columns,
                                   filled=True, rounded=True,
                                   special_characters=True)
        graph = graphviz.Source(dot_data)
        # 你可以把註解解除, 輸出一個 pdf
        # graph.render("boston")
        graph
```

Out[5]:



```
In [6]: # 我們可以直接使用 numpy 來實現兩個 list 的直接相減
        import numpy as np
        predict = regr.predict(data_test)
        print("實際的價錢:", target_test)
        print("預測的價錢:", predict)
        interval = np.subtract(predict, target_test)
        print("差異:", interval)
```

```
實際的價錢: [20.3 19.3 28.2 24.5 29.4 29.  19.9 21.  21.  21.6 43.1 19.2 12.7 27.5
 14.3 22.4 10.2 32.2 13.4 19.3 22.6  9.7 27.5 22.3 19.4 20.7 24.8 14.4
 18.2 21.9 31.7 50.  22.  15.  18.8 22.  23.6 26.5 20.4 16.7 23.9 15.
 26.4 14.8 13.1 16.1 26.4 24.8 21.2 20.6 20.1]
預測的價錢: [22.91040724 22.91040724 22.91040724 22.91040724 22.91040724 33.77435897
 22.91040724 22.91040724 22.91040724 22.91040724 45.84814815 22.91040724
 17.1483871  22.91040724 17.1483871  17.1483871  11.8119403  33.77435897
 11.8119403  22.91040724 22.91040724 11.8119403  22.91040724 22.91040724
 17.1483871  22.91040724 22.91040724 17.1483871  22.91040724 22.91040724
 33.77435897 45.84814815 22.91040724 17.1483871  17.1483871  22.91040724
 22.91040724 22.91040724 22.91040724 11.8119403  33.77435897 14.1
 22.91040724 17.1483871  11.8119403  17.1483871  22.91040724 22.91040724
 22.91040724 22.91040724 11.8119403 ]
差異: [ 2.61040724  3.61040724 -5.28959276 -1.58959276 -6.48959276  4.77435897
  3.01040724  1.91040724  1.91040724  1.31040724  2.74814815  3.71040724
  4.4483871  -4.58959276  2.8483871  -5.2516129   1.6119403   1.57435897
 -1.5880597   3.61040724  0.31040724  2.1119403  -4.58959276  0.61040724
 -2.2516129   2.21040724 -1.88959276  2.7483871   4.71040724  1.01040724
  2.07435897 -4.15185185  0.91040724  2.1483871  -1.6516129   0.91040724
 -0.68959276 -3.58959276  2.51040724 -4.8880597   9.87435897 -0.9
 -3.48959276  2.3483871  -1.2880597   1.0483871  -3.48959276 -1.88959276
  1.71040724  2.31040724 -8.2880597 ]
```

1.3.4 Step 4. 確認一下預測差異

這裡我們不使用混淆矩陣了，而是使用 r2 score，r2 score 的公式如下

(擷取自 wiki https://zh.wikipedia.org/wiki/%E5%86%B3%E5%AE%9A%E7%B3%BB%E6%95%B0)

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

於是可以得到總平方和

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

回歸平方和

$$SS_{\text{reg}} = \sum_i (f_i - \bar{y})^2,$$

殘差平方和

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

由此，決定係數可定義為

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

1 代表最佳，0 代表普通
我們也不用太數學的方式解釋
我們只看兩個特殊值

1. r2 = 1，代表 res = 0，也就是你所有猜的數字都跟真正的數字一樣!
2. r2 = 0，代表 res = 1，也就是你所有猜的數字都是平均值！跟我們之前分類的時候 50% 50% 是一模一樣的意思

當然跟我們之前分類的一樣，訓練的時候如果你得到 r2 = 1，未必是最佳的，也就是分類所說的過擬合 (Overfitting)

我們寧願放棄一點點，讓 r2 大於 0.7 其實就是可以接受的值了
在預測的時候，r2 大於 0.5 其實就是可以接受的值了
當然，r2 的選擇依不同領域，不同情況可能會需要更高一點

```
In [7]: from sklearn.metrics import r2_score
        print("訓練資料 r2 score:", r2_score(target_train,regr.predict(data_train)))
        print("測試資料 r2 score:", r2_score(target_test,regr.predict(data_test)))
```

```
訓練資料 r2 score: 0.8198453341022476
測試資料 r2 score: 0.7707977640804566
```

觀察一下哪個特徵最重要，看起來跟我們一開始畫的圖大致符合，RM 的重要性最高 (0.6)，再接下來是 LSTAT(0.2)

```
In [13]: regr.feature_importances_

Out[13]: array([0.0506248 , 0.        , 0.        , 0.        , 0.02242999,
                0.64434195, 0.        , 0.07653469, 0.        , 0.        ,
                0.        , 0.        , 0.20606857])
```