第九章 决策树

1、已知如表9.1所示的训练数据，生成一个二叉回归树。

**表9.1　训练数据表**

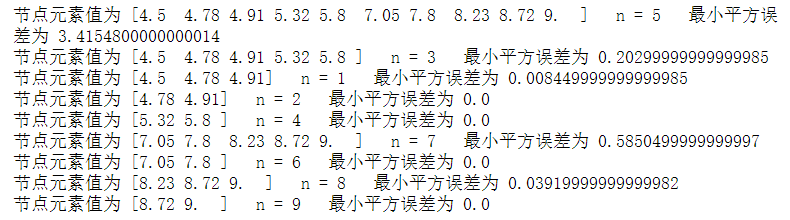
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *xi* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| *yi* | 4.50 | 4.78 | 4.91 | 5.32 | 5.80 | 7.05 | 7.80 | 8.23 | 8.72 | 9.00 |

解：在Jupyer Notebook中利用CART建立回归树求解，代码为：

1. # 第一题
2. # 导入包
3. **import** numpy as np
5. # 节点定义
6. **class** TreeNode(object):
7. **def** \_\_init\_\_(self, tempR, tempc):
8. self.R = tempR
9. self.c = tempc
10. self.left = None
11. self.right = None
13. # 输入y
14. y = np.array([4.5, 4.78, 4.91, 5.32, 5.8, 7.05, 7.8, 8.23, 8.72, 9])
16. # 利用CART算法建立回归树
17. # 变换切分点n，选择使得平方误差最小的切分点
18. **def** CART(start, end):
19. # 切点n的选择表示R1为x值小于等于n的点，R2为大于n的点
20. **if**(end - start >= 1):
21. result = []
22. **for** n **in** range(start+1, end+1): # n在(start, end]之间取值
23. y1 = y[start:n]  # y1取索引为[start, n]之间的值
24. y2 = y[n:end+1]  # y2取索引为[n+1, end]之间的值
25. result.append((y1.std()\*\*2)\*y1.size + (y2.std()\*\*2)\*y2.size)
26. # std即标准差函数，求标准差的时候默认除以元素的个数
27. # 因此平方后乘以元素个数才是要求的平方差
28. index1 = result.index(min(result)) + start # 取平方差误差最小的索引值
29. root = TreeNode(y[start:end+1], min(result))
30. # 索引值为0-9，x值为1-10，即n的值比求的索引值多1
31. **print**("节点元素值为",y[start:end+1], "  n =",index1+1,
32. "  最小平方误差为",min(result)) # 输出n值和最小平方误差
33. root.left = CART(start, index1)  # 对列表的左侧生成左子树
34. root.right = CART(index1+1, end)  # 对列表的右侧生成右子树
35. **else**:
36. root = None
37. **return** root

40. **if** \_\_name\_\_ == "\_\_main\_\_":
41. root = CART(0, 9)

结果为：



可知每次二叉树如何划分，以及划分后包含的数据和平方误差值。根据此划分结果，可自行绘制完整的回归树图，如下图所示：



2、已知如表9.2所示交通状态数据集，利用拥挤指数CI将路段交通拥挤状态划分为畅通、轻度拥挤、拥挤、严重拥挤4个状态．拥挤指数（CI）为系统延误时间和总的旅行时间的比值。规定0 ≤ CI ≤ 0.15时，交通状态为畅通；0．15 < CI ≤ 0．4时，交通状态为轻度拥挤；0．4 < CI ≤ 0.8时，交通状态为拥挤；CI > 0．8时，交通状态为严重拥挤。为简单起见，此处统一将轻度拥挤和畅通归为交通状况良好，将拥挤和严重拥挤归为交通状况不好。要求如下：

（1）求不同分支方式的基尼指数。

（2）求得完整的决策树。

**表9.2 交通状态数据集**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **天气** | **时间段** | **节假日** | **道路质量** | **特殊天气** | **CI** | **拥挤等级** |
| 晴朗 | 高峰期 | 是 | 好 | 否 | 0.18 | 轻度拥挤 |
| 晴朗 | 非高峰期 | 是 | 坏 | 是 | 0.49 | 拥挤 |
| 晴朗 | 非高峰期 | 否 | 好 | 是 | 0.26 | 轻度拥挤 |
| 晴朗 | 非高峰期 | 否 | 好 | 否 | 0.70 | 畅通 |
| 下雨 | 非高峰期 | 否 | 好 | 否 | 0.11 | 畅通 |
| 下雨 | 非高峰期 | 否 | 好 | 是 | 0.23 | 轻度拥挤 |
| 起雾 | 非高峰期 | 否 | 好 | 是 | 0.45 | 拥挤 |
| 下雪 | 非高峰期 | 否 | 好 | 是 | 0.92 | 严重拥挤 |

1. 解：

基尼指数计算：

* 1. 按照天气划分：

子节点晴朗基尼得分

子节点下雨基尼得分

子节点起雾基尼得分

子节点下雪基尼得分

天气分支加权基尼得分

天气分支基尼指数

* 1. 按照时间段划分：

子节点高峰期基尼得分

子节点非高峰期基尼得分

时间段分支加权基尼得分

时间段分支基尼指数

* 1. 按照节假日划分：

子节点节假日基尼得分

子节点非节假日基尼得分

时间段分支加权基尼得分

时间段分支基尼指数

* 1. 按照道路质量划分：

子节点好基尼得分

子节点坏基尼得分

时间段分支加权基尼得分

时间段分支基尼指数

* 1. 按照特殊天气划分：

子节点是特殊天气基尼得分

子节点非节假日基尼得分

时间段分支加权基尼得分

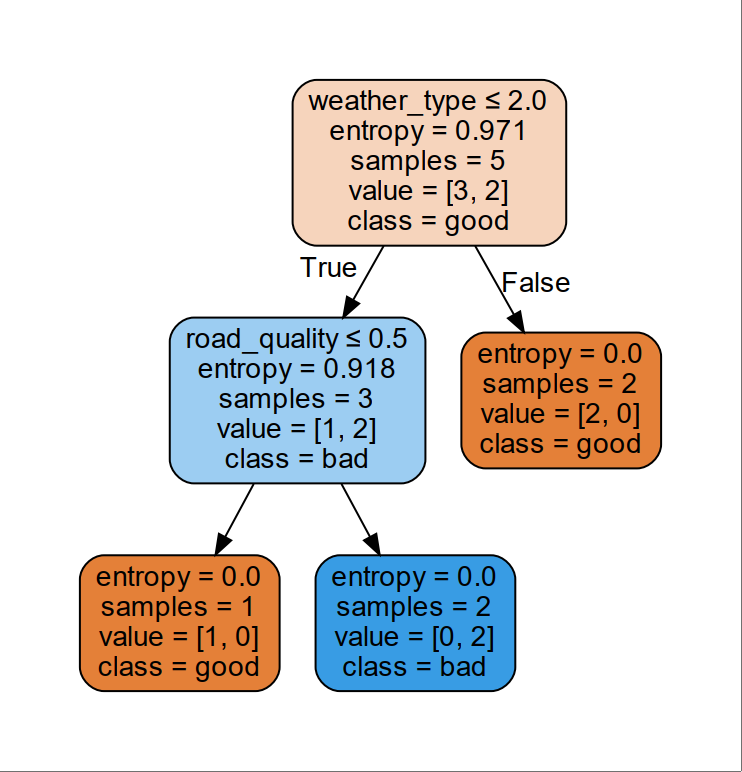
时间段分支基尼指数

综上，按照天气划分的基尼指数较小，分类后纯度大，因此首先应该按照天气进行分支。

1. 解：读者可以在第一题的基础上继续手算求解下一次划分所用特征，此处利用Python求解。为了便于读者练习，本书提供了可以直接使用的数据（traffic\_condition.csv）。代码如下：
2. # 导入包
3. **import** numpy as np
4. **from** sklearn **import** tree
5. **from** sklearn.model\_selection **import** train\_test\_split
6. **import** pydotplus
7. **import** pandas as pd
9. # 定义转换函数
10. **def** weather\_type(s):
11. it = {'sunny':1, 'rainy':2, 'foggy':3, 'snowy':4}
12. **return** it[s]
13. **def** time(s):
14. it = {'peak\_hour':1, 'non\_peak\_hour':2}
15. **return** it[s]
16. **def** holiday(s):
17. it = {'yes':1, 'no':0}
18. **return** it[s]
19. **def** road\_quality(s):
20. it = {'good':1, 'bad':0}
21. **return** it[s]
22. **def** special\_weather(s):
23. it = {'yes': 1, 'no': 0}
24. **return** it[s]
25. **def** traffic\_condition(s):
26. it = {'good': 1, 'bad': 0}
27. **return** it[s]
29. traffic\_feature\_E = 'weather\_type', 'time', 'holiday','road\_quality', 'special\_weather'
30. traffic\_class = 'good', 'bad'
32. # 导入数据
33. data = pd.read\_csv("C:/Users/11/Desktop/DT/traffic\_condition.csv")
35. # 将原始数据中的数据转换为数字形式
36. **for** i **in** range(8):
37. data.iloc[i]['weather\_type'] = weather\_type(data.iloc[i]['weather\_type'])
38. data.iloc[i]['time'] = time(data.iloc[i]['time'])
39. data.iloc[i]['holiday'] = holiday(data.iloc[i]['holiday'])
40. data.iloc[i]['road\_quality'] = road\_quality(data.iloc[i]['road\_quality'])
41. data.iloc[i]['special\_weather'] = special\_weather(data.iloc[i]['special\_weather'])
42. data.iloc[i]['traffic\_condition'] = traffic\_condition(data.iloc[i]['traffic\_condition'])
44. # 划分x、y
45. y = data['traffic\_condition']
46. x = data.drop(['traffic\_condition'],axis = 1)
48. # 拆分训练数据与测试数据
49. x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3)
51. # 使用信息熵作为划分标准，对决策树进行训练
52. clf = tree.DecisionTreeClassifier(criterion='entropy')
53. **print**(clf)
54. clf.fit(x\_train, y\_train.astype('int'))
56. # 把决策树结构写入文件
57. dot\_data = tree.export\_graphviz(clf, out\_file=None, feature\_names=traffic\_feature\_E, class\_names=traffic\_class,
58. filled=True, rounded=True, special\_characters=True)
59. graph = pydotplus.graph\_from\_dot\_data(dot\_data)
60. graph.write\_pdf('play1.pdf')
62. # 系数反映每个特征的影响力。越大表示该特征在分类中起到的作用越大
63. **print**(clf.feature\_importances\_)
65. # 转换数据类型
66. y\_list = y\_train.to\_list()
67. y\_array = np.array(y\_list)
68. y\_train = y\_array.reshape(len(y\_train),-1)
70. y\_list = y\_test.to\_list()
71. y\_array = np.array(y\_list)
72. y\_test = y\_array.reshape(len(y\_test),-1)
74. # 使用训练数据预测
75. answer = clf.predict(x\_train)
76. **print**(answer)
77. **print**(y\_train)
78. **print**(np.mean(answer == y\_train))
80. # 对测试数据进行预测
81. answer = clf.predict(x\_test)
82. **print**(answer)
83. **print**(y\_test)
84. **print**(np.mean(answer == y\_test))

读者可以自行调整划分规则以及各参数。但是，由于此处只有八条数据，数据量过小所以导致该模型效果欠佳，且训练特征并未得到充分利用，在实际应用中很少会出现这种情况。

最终得到的决策树为：



3、在附件中的交通流车速数据集上训练一个决策树模型，并预测数据集所包含时间段之后半个小时的车速。

解：代码如下：

1. # 导入包
2. **from** sklearn.tree **import** DecisionTreeRegressor
3. **import** pandas as pd
4. **import** numpy as np
5. **from** sklearn.model\_selection **import** train\_test\_split
6. **from** sklearn.metrics **import** r2\_score
8. # 导入数据
9. original\_data = pd.read\_csv('C:/Users/11/Desktop/DT/Speed\_data.csv')
10. data = pd.DataFrame(original\_data)
12. # 新增一列索引列
13. data['index'] = data.index
15. # 设置自变量X和因变量y
16. X = data['index']
17. y = data['Speed']
19. # 划分训练集和测试集
20. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,random\_state = 0,train\_size = 0.8)
22. # 转换数据类型
23. X\_list = X\_train.to\_list()
24. X\_array = np.array(X\_list)
25. X\_train = X\_array.reshape(len(X\_train),-1)
27. X\_list = X\_test.to\_list()
28. X\_array = np.array(X\_list)
29. X\_test = X\_array.reshape(len(X\_test),-1)

32. y\_list = y\_train.to\_list()
33. y\_array = np.array(y\_list)
34. y\_train = y\_array.reshape(len(y\_train),-1)
36. y\_list = y\_test.to\_list()
37. y\_array = np.array(y\_list)
38. y\_test = y\_array.reshape(len(y\_test),-1)
40. # 模型训练
41. regressor = DecisionTreeRegressor(random\_state=0)
42. regressor.fit(X\_train, y\_train)
44. # 模型预测
45. regressor\_predict = regressor.predict(X\_test)
47. #  模型评估
48. **print**("回归树的R\_squared值为：", r2\_score(y\_test, regressor\_predict))
50. # 预测之后半小时的车速
51. X\_predict = np.array([2976,2977])
52. X\_predict = X\_predict.reshape(-1, 1)
53. y\_dt = regressor.predict(X\_predict)
54. **print**(y\_dt)

因此，之后半个小时的车速为[82.42 82.42]（以15min为间隔）。

4、在附件中的交通流数据集上训练一个决策树模型，并预测2014年4月2日1:00的统计交通流量。

解：代码类似于上一题，不同代码为：

1. X\_predict = np.array([2980])
2. X\_predict = X\_predict.reshape(-1, 1)
3. y\_dt = regressor.predict(X\_predict)
4. **print**(y\_dt)

结果为：[37.]，即2014年4月2日1:00的统计交通流量预测值为37。读者可以进一步优化调整。