8003119100丁俊

1、导入数据

In [1]:

```
import os
import pandas as pd
```

In [2]:

```
dir_ds = os.path.join(os.path.curdir,'lenses.txt')
df_data = pd.read_csv(dir_ds,sep='\t',header=None,names=['age','spectacle','astigmatic','tear','cla
df_data
```

Out[2]:

	age	spectacle	astigmatic	tear	class
0	young	myope	no	reduced	no lenses
1	young	myope	no	normal	soft
2	young	myope	yes	reduced	no lenses
3	young	myope	yes	normal	hard
4	young	hyper	no	reduced	no lenses
5	young	hyper	no	normal	soft
6	young	hyper	yes	reduced	no lenses
7	young	hyper	yes	normal	hard
8	pre	myope	no	reduced	no lenses
9	pre	myope	no	normal	soft
10	pre	myope	yes	reduced	no lenses
11	pre	myope	yes	normal	hard
12	pre	hyper	no	reduced	no lenses
13	pre	hyper	no	normal	soft
14	pre	hyper	yes	reduced	no lenses
15	pre	hyper	yes	normal	no lenses
16	presbyopic	myope	no	reduced	no lenses
17	presbyopic	myope	no	normal	no lenses
18	presbyopic	myope	yes	reduced	no lenses
19	presbyopic	myope	yes	normal	hard
20	presbyopic	hyper	no	reduced	no lenses
21	presbyopic	hyper	no	normal	soft
22	presbyopic	hyper	yes	reduced	no lenses
23	presbyopic	hyper	yes	normal	no lenses

2、决策树算法

In [3]:

```
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

In [4]:

```
ds = LabelEncoder()
# 定义数据的标签
ds.fit(['no lenses','hard','soft','young','pre','presbyopic','myope','hyper','no','yes','reduced','n
arr_data = np.array([ds.transform(r) for i,r in df_data.iterrows()])
arr_data
```

Out[4]:

```
array([[11,  2,  3,  8,  4],
       [11,  2,  3,  5,  9],
       [11,  2, 10,  8,  4],
       [11,  2, 10,  5,  0],
       [11,  1,  3,  8,  4],
       [11,  1,  3,  5,  9],
       [11,  1, 10,  8,  4],
       [11,  1, 10,  5,  0],
       [ 6,  2,  3,  8,  4],
       [ 6,  2,  3,  5,  9],
       [ 6,  2, 10,  8,  4],
       [ 6,  2, 10,  5,  0],
       [ 6,  1,  3,  8,  4],
       [ 6,  1,  3,  5,  9],
       [ 6,  1, 10,  8,  4],
       [ 6,  1, 10,  5,  4],
       [ 7,  2,  3,  8,  4],
       [ 7,  2,  3,  5,  4],
       [ 7,  2, 10,  8,  4],
       [ 7,  2, 10,  5,  0],
       [ 7,  1,  3,  8,  4],
       [ 7,  1,  3,  5,  9],
       [ 7,  1, 10,  8,  4],
       [ 7,  1, 10,  5,  4]])
```

In [5]:

```
from sklearn.model_selection import train_test_split
```

In [6]:

```
training_set,testing_set = train_test_split(arr_data,test_size = 0.4) # 分割训练集
training_set
```

Out[6]:

```
array([[ 6,  2,  3,  8,  4],
       [ 7,  2,  3,  8,  4],
       [11,  1, 10,  5,  0],
       [ 6,  2, 10,  5,  0],
       [ 6,  2,  3,  5,  9],
       [11,  1,  3,  5,  9],
       [ 6,  1, 10,  5,  4],
       [ 6,  1,  3,  8,  4],
       [ 7,  2, 10,  8,  4],
       [11,  1, 10,  8,  4],
       [ 6,  1, 10,  8,  4],
       [ 6,  2, 10,  8,  4],
       [11,  2,  3,  8,  4],
       [ 7,  1, 10,  5,  4]])
```

In [7]:

```
testing_set
```

Out[7]:

```
array([[11,  2, 10,  5,  0],
       [ 7,  2,  3,  5,  4],
       [11,  2, 10,  8,  4],
       [ 7,  1, 10,  8,  4],
       [ 7,  2, 10,  5,  0],
       [ 7,  1,  3,  5,  9],
       [ 7,  1,  3,  8,  4],
       [11,  1,  3,  8,  4],
       [ 6,  1,  3,  5,  9],
       [11,  2,  3,  5,  9]])
```

In [8]:

```
X_train = training_set[:,0:4]
Y_train = training_set[:,4]
X_test = testing_set[:,0:4]
Y_test = testing_set[:,4]
```

In [9]:

```
from sklearn import tree
```

In [10]:

```
model_dt = tree.DecisionTreeClassifier(criterion='entropy')
model_dt = model_dt.fit(X_train,Y_train)
```

In [11]:

```
y_pre = model_dt.predict(X_test)
y_pre
```

Out[11]:

```
array([0, 9, 4, 4, 0, 9, 4, 4, 9, 9])
```

In [12]:

```
Y_test
```

Out[12]:

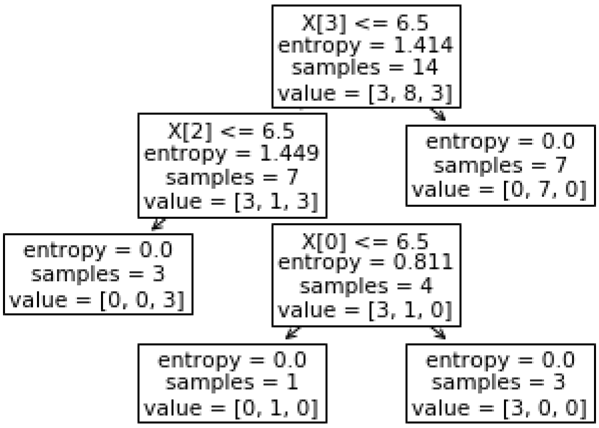
```
array([0, 4, 4, 4, 0, 9, 4, 4, 9, 9])
```

In [44]:

```
tree.plot_tree(model_dt) # 使用sklearn的决策树算法
```

Out[44]:

```
[Text(200.88000000000002, 190.26, ’X[3] <= 6.5\nentropy = 1.414\nsamples = 14\nvalue
= [3, 8, 3]’),
 Text(133.92000000000002, 135.9, ’X[2] <= 6.5\nentropy = 1.449\nsamples = 7\nvalue =
[3, 1, 3]’),
 Text(66.96000000000001, 81.53999999999999, ’entropy = 0.0\nsamples = 3\nvalue = [0,
0, 3]’),
 Text(200.88000000000002, 81.53999999999999, ’X[0] <= 6.5\nentropy = 0.811\nsamples
= 4\nvalue = [3, 1, 0]’),
 Text(133.92000000000002, 27.180000000000007, ’entropy = 0.0\nsamples = 1\nvalue =
[0, 1, 0]’),
 Text(267.84000000000003, 27.180000000000007, ’entropy = 0.0\nsamples = 3\nvalue =
[3, 0, 0]’),
 Text(267.84000000000003, 135.9, ’entropy = 0.0\nsamples = 7\nvalue = [0, 7, 0]’)]
```



3、对实验数据进行回归和预测

In [45]:

```
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_score,f1_score
```

In [46]:

```
confusion_matrix(Y_test,y_pre,labels=[0,4,9]) # 混淆矩阵
```

Out[46]:

```
array([[0, 1, 0],
       [1, 5, 1],
       [0, 0, 2]], dtype=int64)
```

In [47]:

```
accuracy_score(Y_test,y_pre) # 分类正确的百分比
```

Out[47]:

```
0.7
```

In [48]:

```
precision_score(Y_test,y_pre,labels=[0,4,9],average='micro') # 分类的计算精度
```

Out[48]:

```
0.7
```

In [49]:

```
recall_score(Y_test,y_pre,labels=[0,4,9],average='micro') # 召回率：提取出的正确信息条数/样本中的信
```

Out[49]:

```
0.7
```

In [50]:

```
f1_score(Y_test,y_pre,labels=[0,4,9],average='micro') # f1值：精确率和召回率的调和平均数
```

Out[50]:

```
0.7
```

In [ ]:

```

```