第9章　決策樹

第9章　決策樹

- 決策樹的優點
- 決策樹預測模型易有過度擬合的問題
- 如何解決決策樹過度擬合的問題
- 用決策樹探索特徵值的重要性
- 決策樹圖的繪製
- 決策樹圖的解讀

- 到目前為止所學習到的機器演算法都能有相當不錯的預測結果，但始終可惜的是，我們並不曉得這些演算法是如何做預測的，它的判斷準則又是什麼？
- 對人類而言，我們想知道的不僅是結果，更想知道判斷的準則是什麼。這種可以內化和推理的知識，可以幫助我們在未來解決類似的問題。
- 機器學習裡能夠產生判斷準則的演算法叫做決策樹。在本章就要介紹這個神奇的演算法。

範例9-1 載入資料

程式碼

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['font.sans-serif'] = ['DFKai-sb']
plt.rcParams['axes.unicode_minus'] = False
%config InlineBackend.figure_format = 'retina'
import warnings
warnings.filterwarnings('ignore')
# 資料載入
df = pd.read_csv('titanic_train.csv')
df = df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)
```

範例9-1 載入資料

承接上一頁

```
# 欄位設定
X_col_num = ['Age', 'SibSp', 'Parch', 'Fare']
X_col_cat = ['Pclass', 'Sex', 'Embarked']
X_cols = X_col_num + X_col_cat
y_col = 'Survived'
# 資料切割成訓練集和測試集
from sklearn.model_selection import train_test_split
X = df[X_cols]
y = df[y_col]
X_train, X_test, y_train, y_test = train_test_split(X, y,
                              est_size=0.33, random_state=42)
df.head()
```

執行結果

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

範例9-2 先實作資料管道器，並檢視是可用的程式碼

```
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
num_pl = make_pipeline(
    SimpleImputer(strategy='median')
)
cat_pl = make_pipeline(
    SimpleImputer(strategy='most_frequent'),
    OneHotEncoder(sparse=False)
)
```

範例9-2 先實作資料管道器，並檢視是可用的

承接上一頁

```
data_pl = ColumnTransformer([
    ('num_pl', num_pl, X_col_num),
    ('cat_pl', cat_pl, X_col_cat)
])
data_pl.fit_transform(X_train)[:1]
```

執行結果

```
array([[54.     ,  0.     ,  0.     , 51.8625,  1.     ,  0.     ,  0.     ,
         0.     ,  1.     ,  0.     ,  0.     ,  1.     ]])
```

範例9-3 製作決策樹預測模型，用「訓練集」來觀察結果

程式碼

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
model_pl_tree = make_pipeline(data_pl,
                    DecisionTreeClassifier(random_state=42))
model_pl_tree.fit(X_train, y_train)
y_pred = model_pl_tree.predict(X_train)
print('正確率：', accuracy_score(y_train, y_pred).round(2))
print('混亂矩陣')
print(confusion_matrix(y_train, y_pred))
```

執行結果

```
正確率： 0.98
混亂矩陣
[[373   1]
 [ 11 211]]
```

範例9-4　用決策樹模型來預測「測試集」的結果

程式碼

```
y_pred = model_pl_tree.predict(X_test)
print('正確率：', accuracy_score(y_test, y_pred).round(2))
print('混亂矩陣')
print(confusion_matrix(y_test, y_pred))
```

執行結果

```
正確率： 0.74
混亂矩陣
[[133  42]
 [ 35  85]]
```

範例9-5 決策樹的「深度」設定為4

程式碼

```
model_pl_tree = make_pipeline(
    data_pl,
    DecisionTreeClassifier(max_depth=4, random_state=42)
)
model_pl_tree.fit(X_train, y_train)
print('「訓練集」的正確率：', model_pl_tree.score(X_train,
y_train).round(2))
print('「測試集」的正確率：', model_pl_tree.score(X_test,
y_test).round(2))
```

執行結果

```
「訓練集」的正確率： 0.84
「測試集」的正確率： 0.81
```

範例9-6　用不同的決策樹深度來觀察「訓練集」和「測試集」的結果變化

程式碼

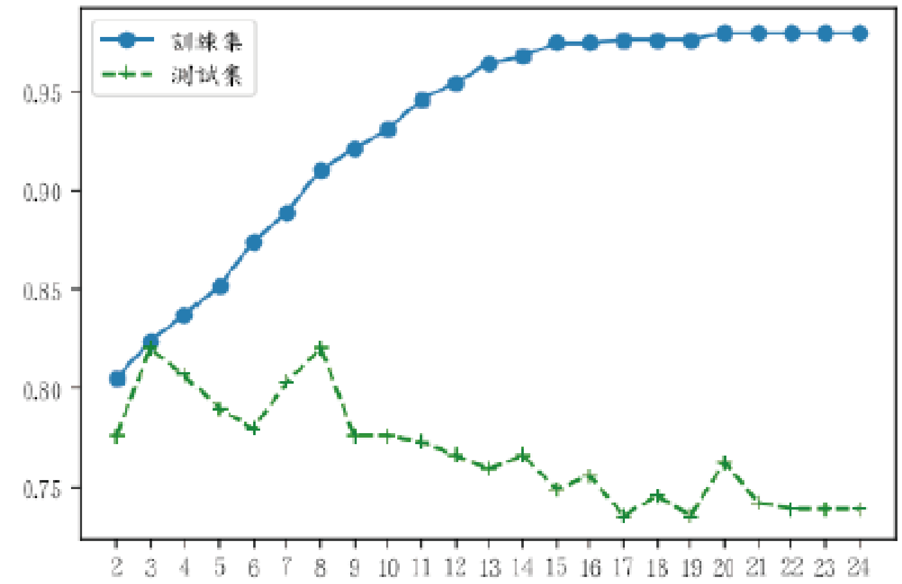
```
acc_train = []
acc_test = []
n_depth = range(2,25)
for n in n_depth:
    model_pl_tree = make_pipeline(
        data_pl,
        DecisionTreeClassifier(max_depth=n, random_state=42)
    )
    model_pl_tree.fit(X_train, y_train)
    acc_train.append(model_pl_tree.score(X_train, y_train))
    acc_test.append(model_pl_tree.score(X_test, y_test))
```

範例9-6　用不同的決策樹深度來觀察「訓練集」和「測試集」的結果變化

承接上一頁

```
# 繪圖開始
plt.plot(n_depth, acc_train,
marker='o', label='訓練集')
plt.plot(n_depth, acc_test,
c='green',
        marker='+', ls='--', label='
測試集')
plt.xticks(n_depth, n_depth)
plt.legend();
```

執行結果



範例9-7　透過min_samples_split來避免過度擬合發生

程式碼

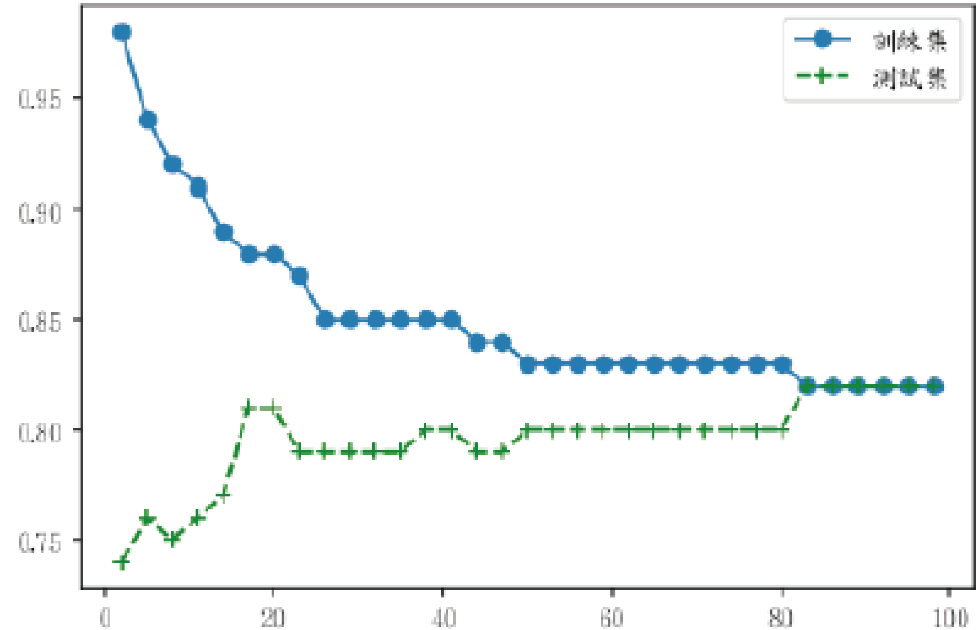
```
acc_train = []
acc_test = []
n_range = range(2,100,3)
for n in n_range:
    model_pl_tree = make_pipeline(data_pl,
                        DecisionTreeClassifier(random_
                        state=42, min_samples_split=n))
    model_pl_tree.fit(X_train, y_train)
    acc_train.append(model_pl_tree.score(X_train,
y_train).round(2))
    acc_test.append(model_pl_tree.score(X_test, y_test).round(2))
```

範例9-7　透過min_samples_split來避免過度擬合發生

承接上一頁

```
plt.plot(n_range, acc_train, marker='o', label='訓練集')
plt.plot(n_range, acc_test, c='green', marker='+', ls='--', label='測試集')
plt.legend();
```

執行結果



範例9-8 用決策樹了解哪些特徵值是重要的

程式碼

```
model_pl_tree = make_pipeline(
    data_pl,
    DecisionTreeClassifier(max_depth=4, random_state=42)
)
model_pl_tree.fit(X_train, y_train)
tree = model_pl_tree.named_steps['decisiontreeclassifier']
feature_importance = tree.feature_importances_.round(3)
feature_importance
```

執行結果

```
array([0.099, 0.043, 0.   , 0.093, 0.   , 0.   , 0.185, 0.561, 0.   ,
       0.   , 0.   , 0.019])
```

範例9-9 取出係數的對應欄位

程式碼

```
print(f'數值型特徵值{X_col_num}')
print(f'類別型特徵值{X_col_cat}')
cat_pl = data_pl.named_transformers_['cat_pl']
oh_cols = cat_pl.named_steps['onehotencoder'].\
get_feature_names(X_col_cat)
print(f'獨熱編碼後的特徵值。{oh_cols}')
cols = X_col_num + oh_cols.tolist()
print(f'所有欄位{cols}')
```

執行結果

```
數值型特徵值 ['Age', 'SibSp', 'Parch', 'Fare']
類別型特徵值 ['Pclass', 'Sex', 'Embarked']
獨熱編碼後的特徵值。['Pclass_1' 'Pclass_2' 'Pclass_3' 'Sex_female'
 'Sex_male' 'Embarked_C' 'Embarked_Q' 'Embarked_S']
所有欄位 ['Age', 'SibSp', 'Parch', 'Fare', 'Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male', 'Embarked_C', 'Embarked_Q', 'Embarked_S']
```

範例9-10 將係數和特徵值名稱結合起來，並依係數大小排序

程式碼

```
pd.DataFrame(feature_importance,
index=cols, columns=['係數']).\
sort_values(by='係數', ascending=False)
```

執行結果

	係數
Sex_female	0.561
Pclass_3	0.185
Age	0.099
Fare	0.093
SibSp	0.043
Embarked_S	0.019
Parch	0.000
Pclass_1	0.000
Pclass_2	0.000
Sex_male	0.000
Embarked_C	0.000
Embarked_Q	0.000

範例9-11 將結果用決策樹圖來呈現

程式碼

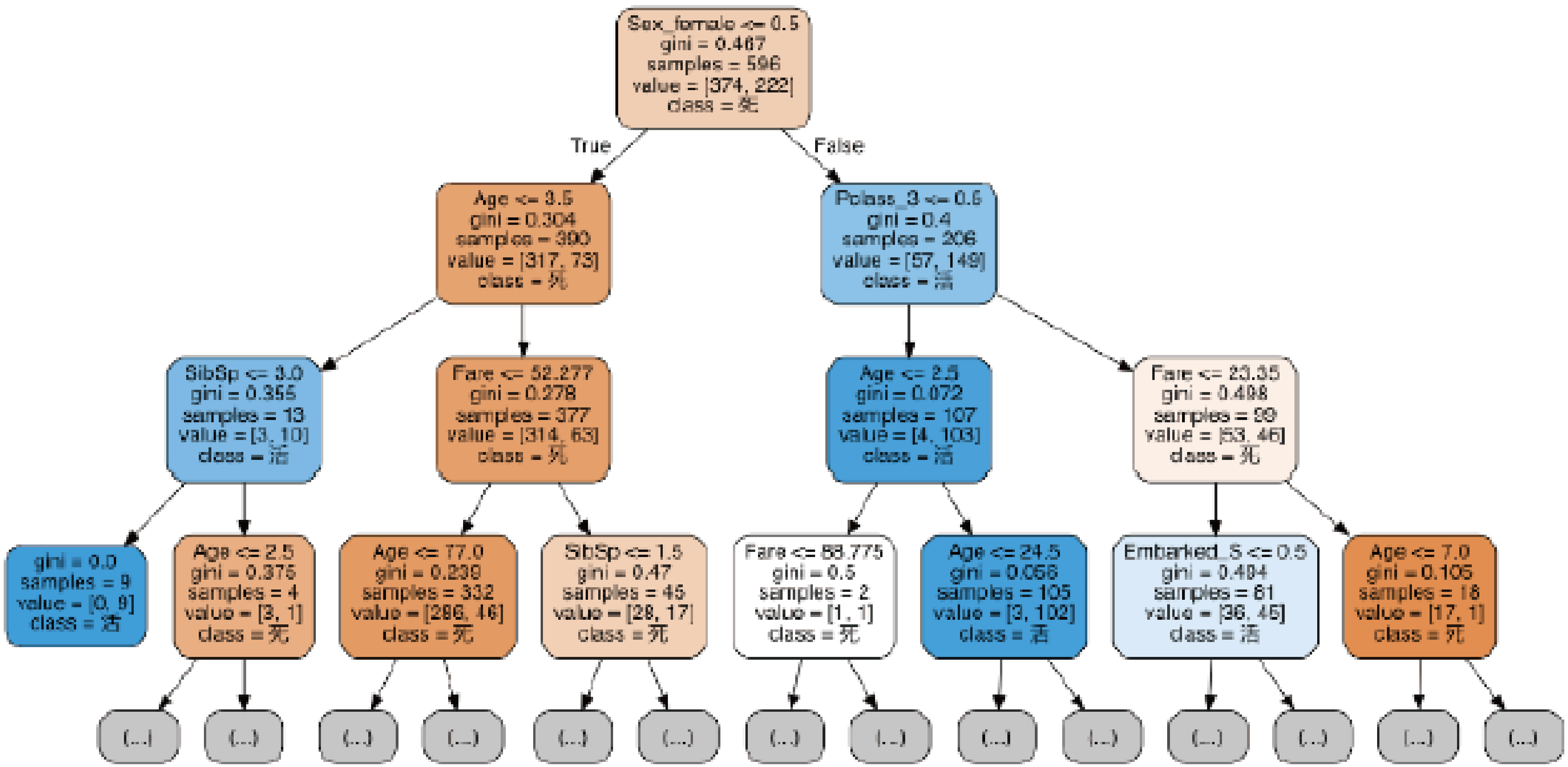
```
from sklearn.tree import export_graphviz
import pydot
from IPython.display import Image
# features 變數存放所有欄位名稱
features = cols
# class_names 變數存放目標值表呈現的文字意義
class_names = ['死', '活']
# export_graphviz 的第一個參數是決策樹模型的預測結果
# max_depth=3 可設定決策樹呈現的深度，其餘參數讀者可自
己測試
dot_data = export_graphviz(
    model_pl_tree.named_steps['decisiontreeclassifier'],
    out_file=None,
```

範例9-11 將結果用決策樹圖來呈現

承接上一頁

```
    feature_names=features,
    class_names = class_names,
    proportion = False,
    max_depth=3,
    filled=True,
    rounded=True
)
graph = pydot.graph_from_dot_data(dot_data)
# 也將結果存到 tree.png檔案裡
graph[0].write_png('tree.png')
Image(graph[0].create_png())
```

執行結果


Sex_female <= 0.5
gini = 0.467
samples = 596
value = [374, 222]
class = 死
True
False
Age <= 3.5
gini = 0.304
samples = 390
value = [317, 73]
class = 死
Pclass_3 <= 0.5
gini = 0.4
samples = 206
value = [57, 149]
class = 活
SibSp <= 3.0
gini = 0.355
samples = 13
value = [3, 10]
class = 活
Fare <= 52.277
gini = 0.278
samples = 377
value = [314, 63]
class = 死
Age <= 2.5
gini = 0.072
samples = 107
value = [4, 103]
class = 活
Fare <= 23.35
gini = 0.498
samples = 99
value = [53, 46]
class = 死
gini = 0.0
samples = 9
value = [0, 9]
class = 活
Age <= 2.5
gini = 0.375
samples = 4
value = [3, 1]
class = 死
Age <= 77.0
gini = 0.239
samples = 332
value = [286, 46]
class = 死
SibSp <= 1.5
gini = 0.47
samples = 45
value = [28, 17]
class = 死
Fare <= 88.775
gini = 0.5
samples = 2
value = [1, 1]
class = 死
Age <= 24.5
gini = 0.056
samples = 105
value = [3, 102]
class = 活
Embarked_S <= 0.5
gini = 0.494
samples = 81
value = [36, 45]
class = 活
Age <= 7.0
gini = 0.105
samples = 18
value = [17, 1]
class = 死
(...)