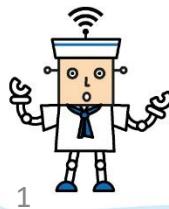




第9章

機器學習實戰（二）： 用 K最近鄰法做分類

機器學習除了做趨勢的預測，另外一個重要的功用則是做分類 (Classification) 的預測。我們可以運用機器學習的分類讓機器進行物體辨識，只要給予電腦觀察物的特徵資料，選定並訓練好模型後就可以進行物體辨識。至今開發出來用於分類的演算法眾多，本章將使用知名的 KNN 分類演算法來實作機器學習的分類。



資料
科學

0

問個感興趣的問題

- 1 照片中的鳶尾花，該如何分辨它是屬於哪一個類別呢？
- 2 假設我們實際觀測一朵鳶尾花的資料，訓練出來的機器學習模型有沒有辦法進行分類的預測？

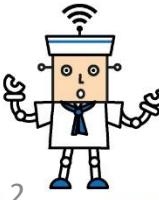
這朵鳶尾花
是屬於哪一個
類別？



哪一類別？



- Iris-Setosa (山鳶尾)
- Iris-Versicolor (變色鳶尾)
- Iris-Virginica (維吉尼亞鳶尾)





1

資料取得

- 從kaggle 網站https://www.kaggle.com/uciml/iris，下載鳶尾花資料集「Iris.csv」，再將「Iris.csv」上傳到Google 雲端硬碟。
- 讀取Google 雲端硬碟的CSV 檔案「Iris.csv」並轉成資料框。
- 刪除「Id」整欄的資料。
- 此資料集共有150 筆記錄，每筆記錄分別有5 個欄位。

```
1 from google.colab import drive  
2 drive.mount('/content/MyGoogleDrive')  
3 import pandas as pd  
4 df=pd.read_csv(i filepath + 'Iris.csv')  
5 df=df.drop('Id', axis=1) ←  
6 df.head()
```

刪除 Id 行資料

↳ Drive already mounted at /content/MyGoogleDrive; to attempt to forcibly remount,

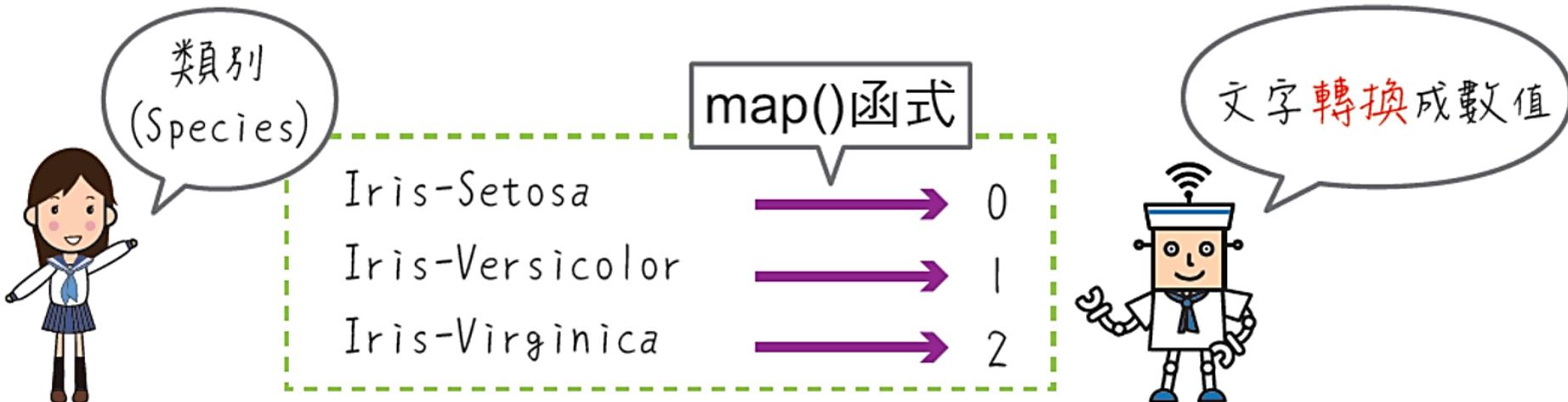
| SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---------------|--------------|---------------|--------------|---------|
|---------------|--------------|---------------|--------------|---------|

| | | | | |
|---|-----|-----|-----|-----------------|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 Iris-setosa |



資料處理

- 先瞭解各行的標題與資料型別
- → 檢查無缺失值的問題
- → 刪除重複值 (共3筆) 後重新編號列索引
- → 將Species (類別) 欄位中的文字轉換成數值



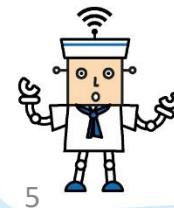
刪除重複列

```
1 df = df.drop_duplicates()  
2 df.reset_index(drop=True)  
3 s = {'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2 }  
4 df['Species']=df['Species'].map(s)  
5 df.info()
```

將列索引重新編號

```
[1]: <class 'pandas.core.frame.DataFrame'>  
Int64Index: 147 entries, 0 to 149  
Data columns (total 5 columns):  
 #   Column           Non-Null Count  Dtype     
---  --  
 0   SepalLengthCm    147 non-null    float64  
 1   SepalWidthCm     147 non-null    float64  
 2   PetalLengthCm    147 non-null    float64  
 3   PetalWidthCm     147 non-null    float64  
 4   Species          147 non-null    int64  
 dtypes: float64(4), int64(1)  
memory usage: 6.9 KB
```

文字轉換成數值

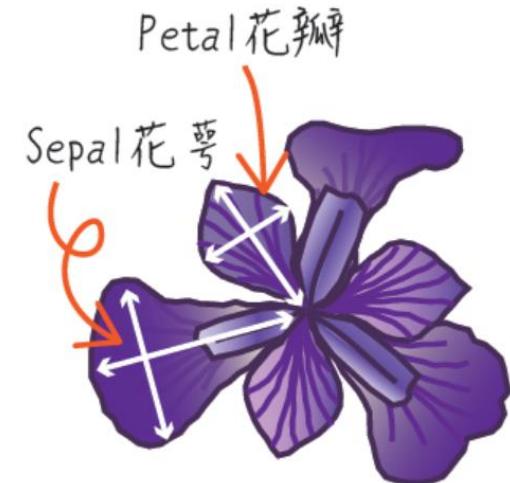




3

探索性資料分析

- 探索性資料分析得知，花萼及花瓣的長度與寬度等4個特徵與類別具有重要的關聯性
- 採用這4個行資料做為機器學習的特徵值 (Feature)
- 類別 (Species) 作為標籤 (Label)。

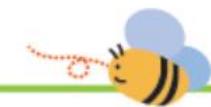


特徵值(X)

- SepalLengthCm(花萼長度)
- SepalWidthCm(花萼寬度)
- PetalLengthCm(花瓣長度)
- PetalWidthCm(花瓣寬度)

標籤(y)

- Species (類別)：
 - 0 : Iris-Setosa (山鳶尾)
 - 1 : Iris-Versicolor (變色鳶尾)
 - 2 : Iris-Virginica (維吉尼亞鳶尾)





1 df.head()

看看前 5 筆資料的特徵值和標籤

特徵值 $X=[['SepalLengthCm', \dots]]$

標籤 $y=['Species']$



| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---------------|--------------|---------------|--------------|---------|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0 |



資料
科學

4

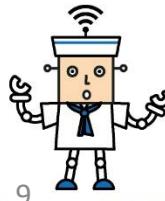
機器學習做資料分析

- 取得資料的特徵值 (X) 和標籤 (y) 之後，便能進行機器學習實作步驟
- 挑選模型 → 學習訓練 → 測試評估 → 決定模型。
- 本例以sklearn 提供的KNN 分類演算法實作機器學習模型。



9-2-1 提出具體的假設

- 由探索性資料分析設定特徵值 (Feature) 和標籤 (Label) 。
- 提出假設：「利用Iris 的花萼及花瓣的長度與寬度可以有效的辨識其類別！」
- 經過機器學習的過程，再把一朵Iris 的資料輸入到訓練後產生的模型中，看看訓練出來的機器學習模型是否能進行分類和辨識。
- 驗證提出的假設是否成立？準確性會是多少？



9-2-2 找出機器學習模型



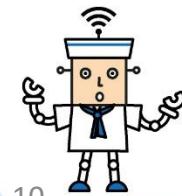
1

挑選模型：匯入 KNN 模型

首先匯入 KNN 機器學習模型。



```
1 from sklearn.neighbors import KNeighborsClassifier
```





學習訓練：建立並訓練 KNN 模型

設定特徵值及標籤

- 將鳶尾花的花萼及花瓣的長度與寬度4個行資料做成**特徵值資料框**`df_X`
- 類別欄位做成**標籤序列**`df_y`。

雙層的中括號，
設定成資料框（特徵值）

```
1 df_X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]  
2 df_y = df['Species']
```

單層的中括號，
設定成序列（標籤）

分割資料集

- 機器學習用的資料集通常會分為兩堆
- 一堆稱為「訓練用資料(Training Data)」，另一堆稱為「測試用資料 (Test Data)」。
- 通常前者資料量會佔 80%，而後者則佔 20%。

分割資料集
train_test_split()

隨機劃分訓練用、測試用資料

匯入 train_test_split() 函式

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

原始資料集的標籤序列 設定測試用資料比例
原始資料集的特徵值資料框 (介於0~1之間的數值)

X_train, X_test, y_train, y_test = train_test_split(train_data, train_target, test_size = 0.2)

訓練用資料

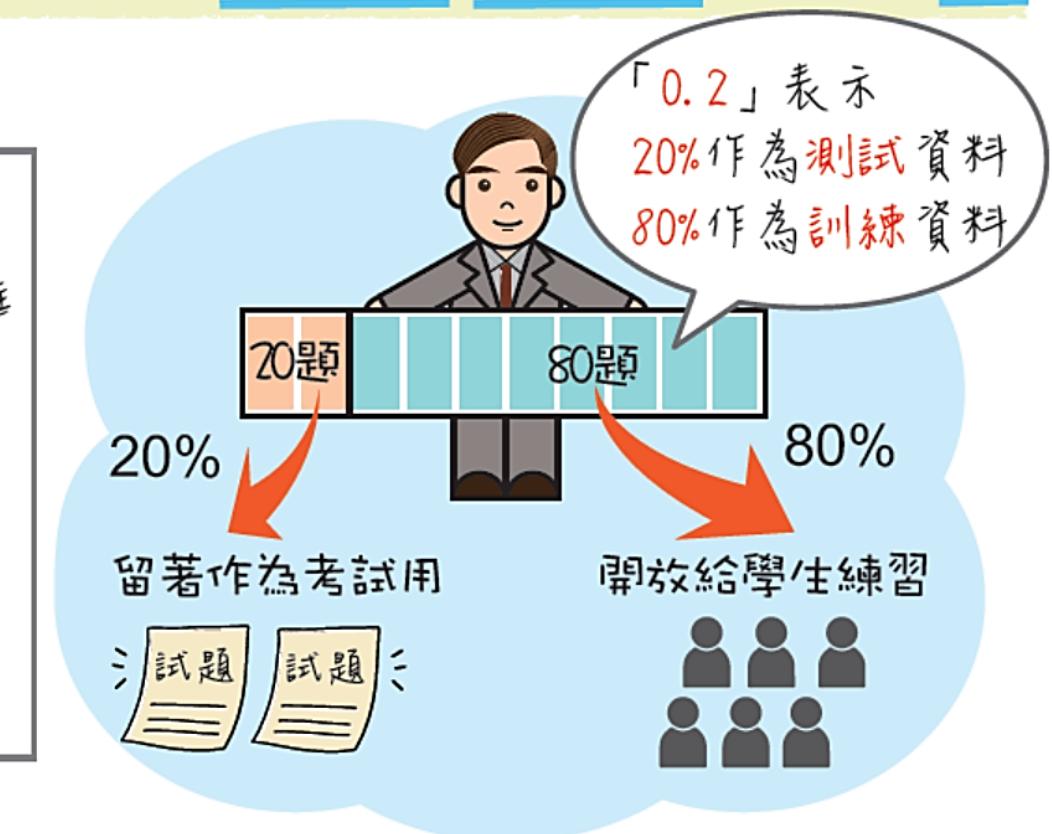
X_train : 訓練用特徵值資料框

y_train : 訓練用標籤序列

測試用資料

X_test : 測試用特徵值資料框

y_test : 測試用標籤序列



- 匯入並使用`train_test_split()`函式將特徵值以及標籤資料皆以80%、20% 的比例分割成訓練用資料 以及測試用資料。

```
▶ 1 from sklearn.model_selection import train_test_split  
2 X_train, X_test, y_train, y_test = 接下  
    train_test_split(df_X, df_y, test_size = 0.2)
```

分割出這 2 個供訓練用

分割出這 2 個供測試用



建立及訓練 KNN 模型

建立
KNN模型

KNN模型名稱 = **KNeighborsClassifier(n_neighbors=k)**

knn = KNeighborsClassifier(n_neighbors=k)

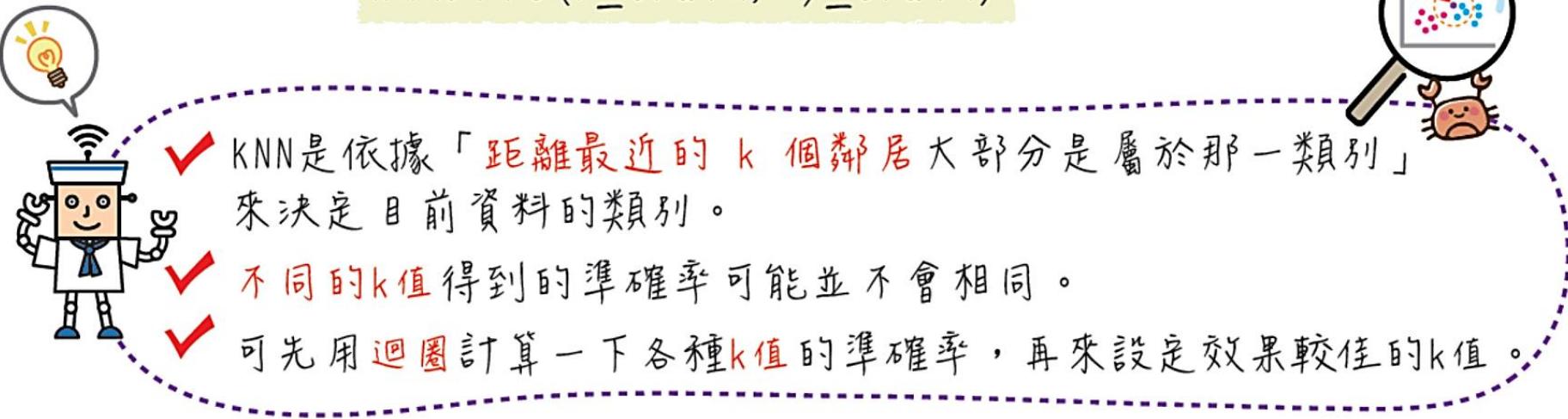
 k 只能正整數



訓練 KNN模型

KNN模型名稱.**fit**(訓練特徵值資料框名稱, 訓練標籤序列名稱)

```
knn.fit(X_train, y_train)
```

- 
- ✓ KNN是依據「**距離最近的 k 個鄰居大部分是屬於那一類別**」來決定目前資料的類別。
 - ✓ 不同的k值得到的準確率可能並不會相同。
 - ✓ 可先用**迴圈**計算一下各種k值的準確率，再來設定效果較佳的k值。



- 使用 KNeighborsClassifier() 函式建立模型knn。
- 先試用「 $k=1$ 」來訓練模型，再視需要進行調校。
- 指定訓練資料（含特徵值及標籤）給 fit() 函式來訓練模型knn。

指定 k 為 1

1 $k = 1$

2 knn=KNeighborsClassifier(n_neighbors=k)

建立新模型 knn

訓練用特徵值

訓練用標籤

1 knn.fit(X_train, y_train)

用 training data 去訓練模型

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=1, p=2,
weights='uniform')
```



測試評估

計算模型的準確率

- 使用**score()** 函式來計算KNN 模型的準確率。

計算
KNN準確率
score()

將測試資料集特徵值(X_{test})和標籤(y_{test})做計算
KNN模型名稱.**score**(測試特徵值資料框名稱, 測試標籤序列名稱)

knn. score(X_{test} , y_{test})

測試用特徵值

測試用標籤



使用模型之前，可以
先利用測試資料集來
評估模型的準確率！

- 在產生理想的KNN 模型之前，先看一看 $k=1$ 的準確率！

```
1 print('----KNN模式訓練後，取test data 進行分類的正確率計算-----')  
2 print('準確率:',knn.score(X_test,y_test))
```

測試用特徵值

測試用標籤

→ ----KNN模式訓練後，取test data 進行分類的正確率計算-----
準確率： 0.9666666666666667

準確率有 96.7%



- 試試別的 **k** 值是否可以得到更佳的效果！
- 一般可以透過 **迴圈** 來計算和觀察不同的參數 **k** 時所得到的模型準確率。

```

1 s = []
2 for i in range(3,11):
3     k=i
4     knn=KNeighborsClassifier(n_neighbors=k)
5     knn.fit(X_train, y_train) # 用 training data 去訓練模型
6     print('k =',k,' 準確率:',knn.score(X_test,y_test)) #用 test data
7     s.append(knn.score(X_test,y_test)) #檢測模型的準確率

```

→ k = 3 準確率: 0.9666666666666667
 k = 4 準確率: 0.9666666666666667
k = 5 準確率: 0.9666666666666667
k = 6 準確率: 0.9666666666666667
k = 7 準確率: 0.9666666666666667
 k = 8 準確率: 1.0
k = 9 準確率: 1.0
 k = 10 準確率: 1.0

- 由結果得知「在不同 k 值時所得到的準確率不一定會相同」。
- 採用 $k = 8$ ，建立並訓練Knn模型。

建立並訓練 $k=8$ 的 k_{nn} 模型

```
1  k = 8
2  knn=KNeighborsClassifier(n_neighbors=k)
3  knn.fit(X_train, y_train)
```

```
→ KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                      weights='uniform')
```



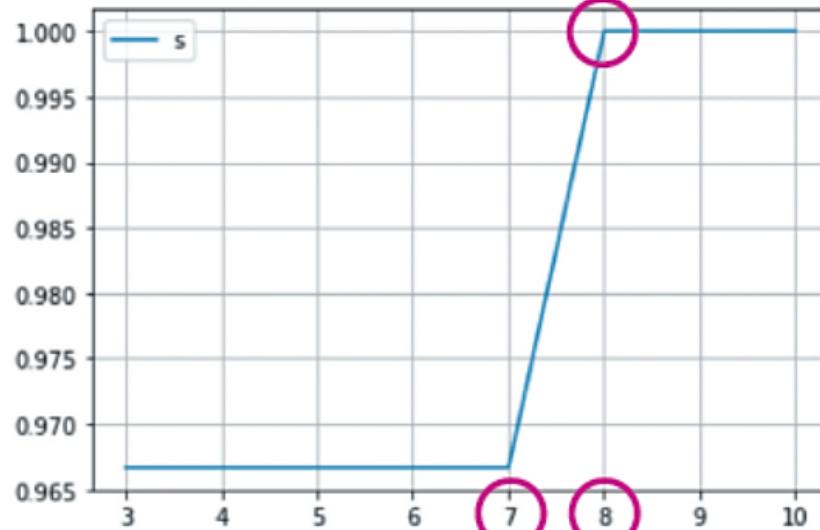


視覺化圖表來顯示準確率

將準確率由數字改成用視覺化圖表來顯示，可以更容易比較出不同 k 值的差別。

```
▶ 1 df_knn = pd.DataFrame()  
2 df_knn['s'] = s  
3 df_knn.index = [3,4,5,6,7,8,9,10]  
4 df_knn.plot(grid=True)
```

```
⇨ <matplotlib.axes._subplots.AxesSubplot at 0x7f81c7638160>
```

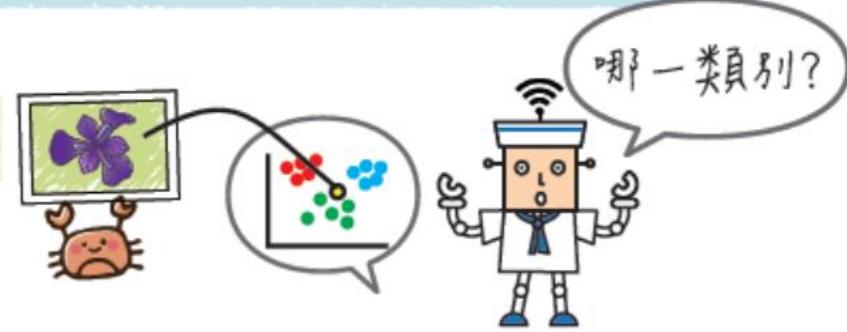


以測試集進行預測

KNN模型
分類預測
`predict()`

預測結果序列 = KNN 模型.`predict`(測試特徵值資料框)

```
pred=knn.predict(X_test)
```



(1) 測試的特徵值資料 `X_test` 以 `predict()` 函式進行預測各筆資料的分類。

```
1 print('分類的預測結果：')
```

```
2 pred = knn.predict(X_test)
```

3 pred

產生 Test data 的預測結果

C → 分類的預測結果：

```
array([2, 1, 0, 2, 0, 2, 1, 0, 2, 0, 1, 0, 0, 0, 0, 2, 2, 1, 2, 0, 2, 2, 2,  
      1, 0, 1, 1, 2, 1, 0, 1])
```

機器學習模型所預測的分類結果

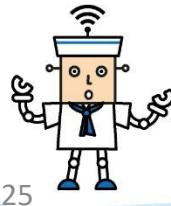


(2) 再把此預測結果序列 (pred) 和測試資料標籤(y_test) 做比對，看看預測結果會是如何。

1 y_test.values ← 觀察 Test data 真實數據 (標籤)

```
array([2, 1, 0, 2, 0, 2, 1, 0, 2, 0, 1, 0, 0, 0, 2, 2, 1, 2, 0, 1, 2, 2,  
1, 0, 1, 1, 2, 1, 0, 1])
```

真正的答案





加深
知識

利用values 屬性做橫式顯示

- 上一頁最底下若直接使用序列名稱 (`y_test`) 會連同索引一起印出來，並且印成直式，這樣會不方便比對。
- 加上「**values**」屬性之後，就可以將序列的值以**橫式**顯示。



1 `y_test`



137

93

11

105

28

135

95

49

104

47

87

16

2

1

0

2

0

2

1

0

2

0

1

0

索引

值



1 `y_test.values`

```
↳ array([2, 1, 0, 2, 0, 2, 1, 0, 2, 0, 1, 0,  
        1, 0, 1, 1, 2, 1, 0, 1])
```

▲ 橫印

▲ 直印

匯入及計算測試集分類的準確率

計算

分類準確率

accuracy_score()

accuracy_score(真實數據分類序列, 預測分類序列)

匯入 accuracy_score 函式

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, pred)
```

真實數據 (標籤) 分類序列

預測分類序列

accuracy_score()

vs.

score()

accuracy_score() 和 9-11頁所介紹的 score() 基本上用途相同，都是將測試集的預測分類結果與真實數據分類 (標籤) 進行比對，然後計算準確率。唯一的差別是兩個函式所傳入的參數不太一樣，例如使用accuracy_score() 前需先用 predict() 計算出預測分類結果 (pred)，才能用accuracy_score() 計算準確率。

- 決入並使用accuracy_score() 函式計算預測結果(分類)的準確率。

▶ 1 from sklearn.metrics import accuracy_score
2 accuracy_score(y_test, pred)

⇒ 0.9666666666666667



準確率(每次執行結果可能不同)



匯入及計算混淆矩陣

- 以人工來做測試集的預測結果和測試集標籤的比對工作，真可說是既辛苦而且又容易出錯。
- 利用Python所提供的「**混淆矩陣 (confusion_matrix)**」直接交給電腦去進行判讀，分析模型在每個類別上的表現狀況，能快速的找出有哪一些分類出了狀況。

WHAT?
混淆矩陣



```
y1=pd.Series([A,A,A,G,C,C,G,G,G,C,C,C,A,A,G])
```

真實數據分類序列

```
p1=pd.Series([A,G,A,G,C,C,G,G,A,C,C,C,G,A,G])
```

預測分類序列

```
confusion_matrix(y1,p1) <span style="color:red;">產生混淆矩陣
```

```
array([[3, 0, 2],  
       [0, 5, 0],  
       [1, 0, 4]])
```

混淆
矩陣


| | | 預測分類 | | |
|------|---|------|---|---|
| | | A | C | G |
| 真實數據 | A | 3 | 0 | 2 |
| | C | 0 | 5 | 0 |
| G | 1 | 0 | 4 | |

2個蘋果被誤判為芭樂了！

1個芭樂被誤判為蘋果了！

混淆
矩陣



將預測分類序列和真實數據分類序列依序逐一比對，並且將比對的結果以表格的方式來呈現。

分類正確

: 預測分類和真實數據分類相同

分類錯誤

: 預測分類和真實數據分類不同

計算
混淆矩陣
`confusion_matrix()`

`confusion_matrix(真實數據分類序列, 預測分類序列)`

匯入 `confusion_matrix` 函式
`from sklearn.metrics import confusion_matrix`

`confusion_matrix(y_test, pred)`

真實數據分類序列

預測分類序列



- 汇入及使用函式`confusion_matrix()`來計算混淆矩陣，可以清楚顯示3種鳶尾花類別的預測結果。

```
1 from sklearn.metrics import confusion_matrix  
2 confusion_matrix(y_test, pred)
```

```
array([[10, 0, 0],  
       [0, 9, 1],  
       [0, 0, 10]])
```

對角線代表分類正確個數



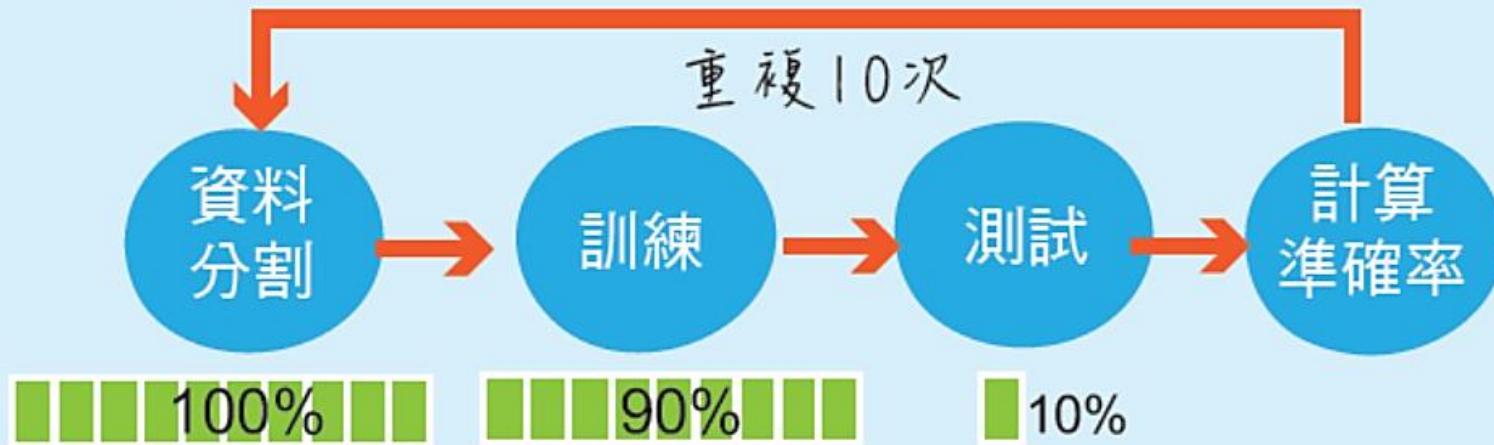


交叉驗證概念

- 為了避免在分割資料集時 “剛好” 挑選到非常適合的訓練資料所決定的k值，最終得到了“**極高**”的準確率。
- 專家提出「**交叉驗證 (CrossValidation)**」的概念，就是餵給模型不同的訓練資料後再觀察結果，並且藉此去推測及**選用最佳的k值**，希望能得到真正理想的模型。



將所有資料分割成數等份（例如10份），每1份就是10%的資料量。
第一次驗證拿第1份為測試資料，其餘9份為訓練資料。
以同樣的方式，重複做10次，得到10次正確率之值。



`cross_val_score()`函式會傳回每次測試所得之正確率，再利用`mean()`函式計算平均的正確率，對照使用`score()`函式得到的KNN模型準確率，透過二項數值來推測及選擇最佳的k值，得到真正理想的模型。



交叉驗證
計算準確率

cross_val_score() 函式

匯入 cross_val_score 函式

```
from sklearn.model_selection import cross_val_score
```

使用前先匯入



```
cross_val_score(模型名稱, 特徵值資料框, 標籤序列, scoring='accuracy', cv=正整數)
```

每一次特徵值資料框、標籤序列
都是未分割的所有資料

↑
重複次數



運算過程

```
cross_val_score(knn, df_x, df_y, scoring='accuracy', cv=10)
```

第 1 次驗證：(訓練資料 = 白底、測試資料 = 黃底)

| | | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| 10% (1) | 10% (2) | 10% (3) | 10% (4) | 10% (5) | 10% (6) | 10% (7) | 10% (8) | 10% (9) | 10% (10) |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|

→ cross_val_score → 1.

第 2 次驗證：(訓練資料 = 白底、測試資料 = 黃底)

| | | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| 10% (1) | 10% (2) | 10% (3) | 10% (4) | 10% (5) | 10% (6) | 10% (7) | 10% (8) | 10% (9) | 10% (10) |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|

→ cross_val_score → 0.93

.....

第 10 次驗證：(訓練資料 = 白底、測試資料 = 黃底)

| | | | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| 10% (1) | 10% (2) | 10% (3) | 10% (4) | 10% (5) | 10% (6) | 10% (7) | 10% (8) | 10% (9) | 10% (10) |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|

→ cross_val_score → 1.



```
1 from sklearn.model_selection import cross_val_score  
2 s = cross_val_score(knn, df_X, df_y, scoring='accuracy', cv=10)  
3 print('交叉驗證每次的準確率：',s)  
4 print('交叉驗證得到的平均準確率：',s.mean())
```

⇨ 交叉驗證每次的準確率： [1. 0.93333333 1. 1. 0.86666667 0.9333333
0.93333333 1. 1. 1.]
交叉驗證得到的平均準確率： 0.9666666666666668

交叉驗證得到的平均準確率





4 決定模型

- 如果評估後，模型的**準確性**已可接受，則模型建立完成
- 否則，重新選擇模型或調整KNN參數後再重新訓練。





5 進行分類預測

- 給一朵鳶尾花的4個特徵值：
- 花萼長度 6.6公分、花萼寬度 3.1公分、花瓣長度 5.2公分、花瓣寬度 2.4公分
- 試試我們建立的辨識模型，看看這朵花應該屬於那一個類別。

```
1 new = [[6.6,3.1,5.2,2.4]]  
2 v=knn.predict(new)  
3 if v==0:  
4     s='Iris-Setosa'  
5 elif v==1:  
6     s='Iris-Versicolour'  
7 elif v==2:  
8     s='Iris-Virginica'  
9 else:  
10    s='錯誤'  
11 print('預測結果為：', s)
```

某一朵花的 4 個特徵值

呼叫 knn 模型，進行預測

→ 預測結果為： Iris-Virginica

9-2-3 提出機器學習後的資料分析結果

- 以機器學習演算法KNN，輸入歷史資料進行訓練，利用完成的模型進行分類是可行的。
- 以本例而言，訓練完模型後我們給了一筆新的特徵值資料 ([[6.6, 3.1, 5.2, 2.4]])，預測出它是「Iris-Virginica」。
- 如果不採用全部4 個特徵值來實作的話，準確率可能就不會很高。





加廣
知識

這樣的機器學習可以進一步應用於以下情境：

- 貓狗的識別。
- 人臉、車牌等識別。
- Titanic（鐵達尼號）之生還預測。





問個感興趣的問題

如果時光真的能夠倒流，當鐵達尼號船難發生時：

- ① 哪些乘客們可以「生還」？例如：以下的這些人能存活下來嗎？

電影中的男主角、女主角



Jack



Rose

梅西百貨公司創辦夫婦，Mrs. Straus 自始至終一直陪伴著 Mr. Straus



Mr. & Mrs. Straus

電影中同情男主角的暴發戶，鐵達尼倖存委員會創辦人



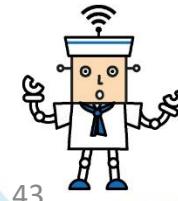
Mrs. Brown

- ② 如果我們也搭上鐵達尼號呢，我們能夠存活下來嗎？



資料取得

- 從kaggle 網站
<https://www.kaggle.com/c/titanic/data> 下載「titanic.zip」
- 解壓縮後將「train.csv」上傳到 Google 雲端硬碟。
- 首先讀取Google drive 的CSV 檔案「train.csv」並轉成**資料框**型別。
- 共有891筆記錄，每筆記錄分別有12個欄位。





```
1 from google.colab import drive  
2 drive.mount('/content/MyGoogleDrive')  
3 import pandas as pd  
4 df=pd.read_csv(i_filepath + 'train.csv')  
5 df.head()
```

↳ Drive already mounted at /content/MyGoogleDrive; to attempt to forcibly remount, call drive.mount("/content/MyGoogleDr

| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|-------------|----------|--------|--|--------|------|-------|-------|---------------------|---------|-------|----------|
| 0 | 1 | 0 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| 1 | 2 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th...) | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 2 | 3 | 1 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| 3 | 4 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| 4 | 5 | 0 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |





資料處理

先瞭解各行的標題與資料型別

- → 缺失值的補值或刪除
- → 刪除重複值或異常值
- → 資料轉換





1 df.info()

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  --  
 0   PassengerId 891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object  
 4   Sex          891 non-null    object  
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object  
 9   Fare          891 non-null    float64 
 10  Cabin         204 non-null    object  
 11  Embarked     889 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

共 891 筆列資料

不到 891 筆，有缺失值



刪除

補值

資料轉換

```
1 df['Age']=df['Age'].fillna(df['Age'].mean())
2 df['Embarked']=df['Embarked'].fillna('S')
3 df=df.drop('Cabin', axis=1)
4 print('重複值：', df[df.duplicated()]) #檢查有無重複值
5 df['Sex']=df['Sex'].map({'female':0, 'male':1})
6 df['Embarked']=df['Embarked'].map({'S':0, 'C':1, 'Q':2})
7 df.head()
```

重複值： Empty DataFrame

Columns: [PassengerId, Survived, Pclass, Name, Parch, Ticket, Fare, Embarked]
Index: []

| | PassengerId | Survived | Pclass | | Sex | Age | SibSp | Parch | Ticket | Fare | Embarked |
|---|-------------|----------|--------|---|-------------|-----|-------|-------|--------|-----------|----------|
| 0 | | 1 | 0 | 3 | | 1 | 22.0 | 1 | 0 | A/5 21171 | 7.2500 |
| 1 | | 2 | 1 | 1 | Cumings, Mr | 0 | 38.0 | 1 | 0 | PC 17599 | 71.2833 |
| 2 | | 3 | 1 | 3 | | 0 | 26.0 | 0 | 0 | STON/O2. | 7.9250 |

處理好的資料





3 探索性資料分析

- 依探索性資料分析得知，「性別 (Sex)」和「艙等 (Pclass)」與乘客的生還狀況 (Survived) 具有重要的關聯性。
- 採用這2個行資料做為機器學習的特徵值 (Features)；「生還狀況 (Survived)」則為標籤 (Label)。





1 df.head()

□

| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch |
|-------------|----------|--------|--|-----|------|-------|-------|
| 0 | 1 | 0 | Braund, Mr. Owen Harris | 1 | 22.0 | 1 | 0 |
| 1 | 2 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th...) | 0 | 38.0 | 1 | 0 |
| 2 | 3 | 1 | Heikkinen, Miss. Laina | 0 | 26.0 | 0 | 0 |
| 3 | 4 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | 0 | 35.0 | 1 | 0 |
| 4 | 5 | 0 | Allen, Mr. William Henry | 1 | 35.0 | 0 | 0 |

標籤 $y=[\text{'Survived'}]$

特徵值 $X=[[\text{'Sex'}, \text{'Pclass'}]]$



9-4

機器學習前實作 —以 Titanic 為例



4

機器學習做資料分析

- 如何建立及訓練KNN 模型？
- 完成後試試KNN 模型用在鐵達尼號的生還預測效果如何。



資料科學 × 機器學習

實戰探索

Practical Exploration



50

9-4-1 提出具體的假設

- 由探索性資料分析選定特徵值之後，我們可以提出如此的假設：
「採用KNN 模型，利用性別 (Sex) 和艙等 (Pclass) 的特徵值、並且以生還狀況(Survived) 為標籤，進行訓練後，將可以有效的預測生還狀況。」





特徵值(X)

- Sex(性別)

1 : 男 0 : 女

- Pclass(艙等)

1 : 一等艙 2 : 二等艙 3 : 三等艙

標籤(y)

- Survived(生還狀況)

0 : 死亡

1 : 生還



9-4-2 找出機器學習模型



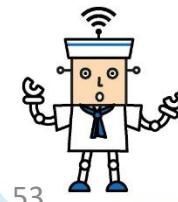
①

挑選模型：匯入 KNN 模型

首先挑選並**匯入**常用的KNN機器學習模型。



```
1 from sklearn.neighbors import KNeighborsClassifier
```





學習訓練：建立並訓練 KNN 模型

設定特徵值及標籤

- 將「性別 (Sex)」和「艙等 (Pclass)」 2個行資料做成特徵值資料框df_X
- 「生還狀況 (Survived)」 做成標籤序列df_y。



```
1 df_X = df[['Sex', 'Pclass']]  
2 df_y = df['Survived']
```



分割資料集

- 匯入並使用train_test_split() 函式將全部891筆中分割出20% (即179筆)為測試資料，所以就會有80% (即712筆) 做為訓練資料。

```
分割出 2 個測試用  
1 from sklearn.model_selection import train_test_split  
2 X_train, X_test, y_train, y_test =  
    train_test_split(df_X, df_y, test_size = 0.2)
```

分割出 2 個訓練用

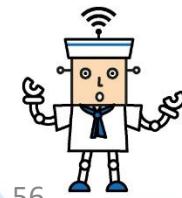


建立 KNN 模型

- 建立KNN 模型的訓練。在此先試用「 $k=1$ 」來訓練模型。

指定 k 為 1

```
1 k = 1
2 knn=KNeighborsClassifier(n_neighbors=k)
```



進行訓練

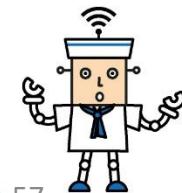
- 指定訓練資料（含特徵值及標籤）給fit()函式來訓練模型knn。

訓練用特徵值 訓練用標籤



```
1 knn.fit X_train, y_train
```

```
↳ KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                      weights='uniform')
```





③ 測試評估

計算模型的準確率

- 使用score() 函式計算模型準確率。
(1) 以「 $k=1$ 」訓練出來的模型進行測試。



```
1 print('----KNN模式訓練後，取test data 進行分類的準確率計算-----')  
2 print('準確率:',knn.score(X_test,y_test))
```



測試用特徵值



測試用標籤

```
----KNN模式訓練後，取test data 進行分類的準確率計算-----  
準確率: 0.36312849162011174
```

(2) 透過迴圈計算和觀察不同的參數 k 時所得到的模型準確率，試試別的 k 值是否可以得到更佳的效果。

```
1 s = []
2 for i in range(3,11):
3     k=i
4     knn=KNeighborsClassifier(n_neighbors=k)
5     knn.fit(X_train, y_train) # 用 training data 去訓練模型
6     print('k =',k,' 準確率:',knn.score(X_test,y_test))
    #用 test data 檢測模型的準確率
7     s.append(knn.score(X_test,y_test))
```

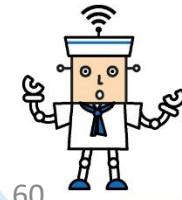
```
↳ k = 3 準確率: 0.7262569832402235
    k = 4 準確率: 0.7262569832402235
    k = 5 準確率: 0.7262569832402235
    k = 6 準確率: 0.7932960893854749
    k = 7 準確率: 0.7932960893854749
    k = 8 準確率: 0.7932960893854749
    k = 9 準確率: 0.7262569832402235
    k = 10 準確率: 0.7262569832402235
```

(3) 由結果得到 k 在不同值時分別的準確率，
在此採用 $k = 7$ 。



```
1 k = 7  
2 knn=KNeighborsClassifier(n_neighbors=k) ← 建立並訓練  $k=7$  的  $k_{nn}$  模型  
3 knn.fit(X_train, y_train)
```

```
↳ KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                      metric_params=None, n_jobs=None, n_neighbors=7, p=2,  
                      weights='uniform')
```



以測試集進行預測

- 以「 $k=7$ 」為KNN模型的參數，拿測試資料透過KNN預測後產生的生還值和真實標籤來進行比對，可找出判斷錯誤的資料。

```
1 print('分類的預測結果：')
2 pred = knn.predict(X_test) ←
3 print(pred)
4 print('真實數據：')
5 print(y_test.values) ←
```

產生 Test data 的預測結果

觀察 Test data 真實數據（標籤）

分類的預測結果：

```
[0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0
 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0]
```

真實數據：

```
[0 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0
 1 0 0 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 1
 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0
 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0]
```

黃底的為錯誤的分類

(每次執行的結果可能不同)

匯入及計算測試集分類的準確率

- 匯入並使用accuracy_score() 函式計算預測結果(分類)的準確率。

▶ 1 from sklearn.metrics import accuracy_score
2 accuracy_score(y_test, pred)

⇨ 0.7932960893854749

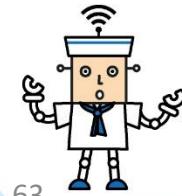


匯入及計算混淆矩陣

- 使用**混淆矩陣**協助快速分析辨識正確和錯誤的情形。

(1) 初步得到如下頁的數據。

- 全部測試資料為 $113 + 3 + 34 + 29 = 179$ 筆
- 正確判斷的筆數是 $113 + 29 = 142$
- 準確率約為 $142 / 179 = 79.33\%$ 。



```

▶ 1 from sklearn.metrics import confusion_matrix
  2 confusion_matrix(y_test, pred)

```

⇒ array([[113, 3],
[34, 29]])

34 及 3 代表分類錯誤個數
113 及 29 代表分類正確個數

| KNN預測 | | 0 死亡 | 1 生還 |
|-------|------|------|------|
| 真實數據 | 0 死亡 | 113 | 3 |
| 0 死亡 | 34 | | 29 |
| 1 生還 | | | |

「死亡」誤判為
「生還」有 3 人

「生還」誤判為
「死亡」有 34 人

正確預測

$$113 + 29 = 142 \text{ 人}$$



(2) 接下來進行10次交叉驗證的預測

- 檢視我們採用及訓練的模型在這10次中準確率的差異。
- 結果為： 平均準確率76.99%，最好達82.02%、最差有74.16%。



```
1 from sklearn.model_selection import cross_val_score
2 s=cross_val_score(knn, df_X, df_y, scoring='accuracy', cv=10)
3 print('準確率：',s)
4 print('平均準確率：',s.mean())
5 print('最高：',s.max())
6 print('最差：',s.min())
```

```
→ 準確率： [0.74444444 0.79775281 0.76404494 0.75280899 0.82022472 0.7752809
          0.76404494 0.74157303 0.7752809 0.76404494]
平均準確率： 0.7699500624219726
最高： 0.8202247191011236
最差： 0.7415730337078652
```



4

決定模型

- 如果評估後，模型的**準確性**已可接受，則模型建立完成
- 否則，重新選擇模型或調整KNN參數後再重新訓練。



進行分類預測

- 使用我們建立的KNN 辨識模型，針對以下這些人物來進行生還預測：

| 人物 | 說明 | 特徵資料 | 請勾選 |
|--|-------------------------------------|---|---|
|  <p>Jack Rose</p> <p>傑克 (Jack) 蘿絲 (Rose)</p> | <p>1997 年 詹姆斯· 卡麥隆導演的鐵達尼號劇中男女主角</p> | <p>傑克： Sex = 1 (男) Pclass = 3 (三等艙)</p> <p>蘿絲： Sex = 0 (女) Pclass = 1 (一等艙)</p> | <p>傑克 (Jack) <input type="checkbox"/> 生還 <input type="checkbox"/> 死亡</p> <p>蘿絲 (Rose) <input type="checkbox"/> 生還 <input type="checkbox"/> 死亡</p> |

特徵值

```
1 print('----- (1) 電影中兩位主角的生還推測 -----')  
2 Rose=[[0,1]] #女性 頭等艙 蘿絲 (Rose DeWitt Bukater)  
3 Jack=[[1,3]] #男性 三等艙 傑克 (Jack Dawson)  
4 v=knn.predict(Rose)  
5 if v==1:  
6     s='生還'  
7 else:  
8     s='死亡'  
9 print('Rose能生還嗎 ? ', s)  
10  
11 v=knn.predict(Jack)  
12 if v==1:  
13     s='生還'  
14 else:  
15     s='死亡'  
16 print('Jack能生還嗎 ? ', s)
```

呼叫模型，進行預測

#Rose為女性，及坐頭等艙

#Jack為男性，及坐三等艙

→ ----- (1) 電影中兩位主角的生還推測 -----

Rose能生還嗎 ? 生還
Jack能生還嗎 ? 死亡

| 人物 | 說明 | 特徵資料 | 請勾選 |
|--|---|--|---|
|  <p>Mr. & Mrs. Straus 伊西多·史特勞斯 (Isidor Straus) 伊達·史特勞斯 (Ida Straus)</p> | <p>伊西多是美國梅西百貨公司創辦人，1912年夫婦兩人搭鐵達尼號坐頭等艙返回美國。當船失事，伊達很快地得以登上救生艇，但她堅持不離開伊西多，雖然伊西多頻催促太太坐上救生艇，但她始終選擇陪伴伊西多，致後來雙雙喪身海裡！</p> | <p>伊西多： Sex = 1 (男) Pclass = 1 (一等艙)</p> <p>伊達： Sex = 0 (女) Pclass = 1 (一等艙)</p> | <p>伊西多·史特勞斯 (Isidor Straus) <input type="checkbox"/>生還 <input type="checkbox"/>死亡</p> <p>伊達·史特勞斯 (Ida Straus) <input type="checkbox"/>生還 <input type="checkbox"/>死亡</p> |

特徵值

```
1 # 真實的伊西多和伊達·斯特勞斯 (Isidor and Ida Straus) 夫婦 (You stay, I stay)
2 # http://www.epochtimes.com/b5/17/12/6/n9931745.htm
3 # Isidor 美國梅西百貨創辦人之一
4 #
5 print('----- (2) 真實的伊西多和伊達·斯特勞斯夫婦的生還推測 -----')
6 Mrs=[[0,1]] #女性 頭等艙 Straus, Mrs. Isidor (Rosalie Ida Blum)
7 Mr=[[1,1]] #男性 頭等艙 Straus, Mr. Isidor
8 v=knn.predict(Mrs)
9 if v==1:
10     s='生還'
11 else:
12     s='死亡'
13 print('Mrs. Straus能生還嗎 ? ', s)
14 v=knn.predict(Mr)
15 if v==1:
16     s='生還'
17 else:
18     s='死亡'
19 print('Mr. Straus能生還嗎 ? ', s)
```

呼叫模型，進行預測

#Ida為女性，及坐頭等艙，可優先搭乘救生艇存活
#Isidor的生存率有多高呢？

→ ----- (2) 真實的伊西多和伊達·斯特勞斯夫婦的生還推測 -----
Mrs. Straus能生還嗎 ? 生還
Mr. Straus能生還嗎 ? 死亡

| 人物 | 說明 | 特徵資料 | 請勾選 |
|---|--|--|---|
|  Mrs. Brown | 布朗夫人的家庭是丹佛淘金熱的富豪，1912年她離開遠在英國的女兒搭鐵達尼號坐頭等艙返美。失事時她順利登上救生艇，但她的救生艇還有四十多個空位，她堅持再多讓多一點人上船，但未受允許。生還返美後她籌措成立鐵達尼號倖存者委員會，積極參加社會慈善活動。 | 布朗夫人： Sex = 0 (女) Pclass = 1 (一等艙) | 布朗夫人 (Mrs. Brown) <input type="checkbox"/> 生還 <input type="checkbox"/> 死亡 |
| 如果當年你是一位搭二等艙的男士呢？ | | Sex = 1 (男) Pclass = 2 (二等艙) | <input type="checkbox"/> 生還 <input type="checkbox"/> 死亡 |

特徵值

```
1 # 真實的 Mrs. Brown  
2 # https://hokkfabrica.com/her-story-margaret-brown-from-titanic/  
3 #  
4 print('----- (3) 真實的Mrs. Brown的生還推測 -----')  
5 #女性 頭等艙 Brown, Mrs. James Joseph (Margaret Tobin) 故事中的暴發戶 對Jack很友善  
6 Brown=[[0,1]]  
7 v=knn.predict(Brown)  
8 if v==1:  
9     s='生還'  
10 else:  
11     s='死亡'  
12 print('Mrs. Brown能生還嗎 ? ', s)
```

#Mrs. Brown呢？

呼叫模型，進行預測

→ ----- (3) 真實的Mrs. Brown的生還推測 -----
Mrs. Brown能生還嗎 ? 生還





```
1 print('----- (5)若你也搭上了鐵達尼號呢？ -----')
2 s=input('您的性別 (0：女，1：男)，請輸入代碼？ ')
3 c=input('搭乘的船艙艙等 (1：S艙，2：C艙，3：Q艙)，請輸入代碼？ ')
4 you=[[int(s),int(c)]]
5 v=knn.predict(you)
6 if v==1:
7     print('預測為：幸運生還')
8 else:
9     print('預測為：無法生還')
```

特徵值

呼叫模型，進行預測

→ ----- (5)若你也搭上了鐵達尼號呢？ -----
 您的性別 (0：女，1：男)，請輸入代碼？ 1
 搭乘的船艙艙等 (1：S艙，2：C艙，3：Q艙)，請輸入代碼？ 1
 預測為：無法生還



9-4-2 提出機器學習後的資料分析結果

- 以**機器學習演算法KNN**，輸入Titanic 歷史資料進行訓練，利用完成的模型進行分類的準確性蠻高的。
- 以本例而言，我們最後給了Jack、Rose... 等6 筆新的特徵值進行分析，模型預測出的生還率符合預期。
- 如果重複幾次的預測時，發現偶爾會有「**意外的結果**」，如Jack 也可能被分類為「生還」，畢竟上述模型的準確度不到百分百。
- 有進一步興趣的話，請前往**kaggle**，有更多網友分享他們的作法。

