

1-2-4 節

Bonus

決策樹與隨機森林

決策樹 Decision Tree

決策樹 (Decision Tree) 有別於神經網路或 SVM 是使用決策邊界或超平面對資料進行分類 (代表分類的依據是一個方程式), 而是使用一連串的問題將資料進行分類, 這樣的好處就是我們能很容易的理解決策樹是怎麼對資料進行分類的。

你可以想一下以下的情境：假設你是一位咖啡店的資料工程師, 店長拿了一份資料給你, 內容是他記錄下各種泡咖啡的方法, 並請人試喝而統計下來的結果：

溫度 (°C)	氣壓 (bar)	喜好
85	8	不好喝
80	10	不好喝
85	10	好喝
90	10	不好喝
85	18	不好喝
85	15	好喝
90	9	不好喝

他希望你能發揮專長, 找出一個判斷如何煮出好喝咖啡的標準, 於是很開心地把資料拿去給神經網路訓練, 並且跑出了一個相當不錯的成果, 所以你把標準寫了下來, 遞給店長：

$$x = [\text{溫度} \quad \text{氣壓}]$$

$$w_0 = \begin{bmatrix} -2.62 & -0.75 & -3.56 & 6.46 & -2.47 \\ 18.84 & -3.58 & -2.62 & -17.24 & 18.27 \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 19.6 \\ 10.14 \\ 7.72 \\ 25.13 \\ 17.37 \end{bmatrix}$$

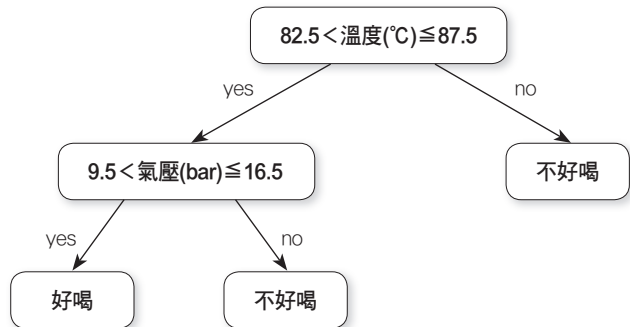
$$b_0 = [0.59 \quad -2.06 \quad -4.83 \quad -2.95 \quad 0.5]$$

$$b_1 = [-7.62]$$

$$y = \text{sigmoid}\{[\tanh(x \cdot w_0 + b_0)] \cdot w_1 + b_1\}$$

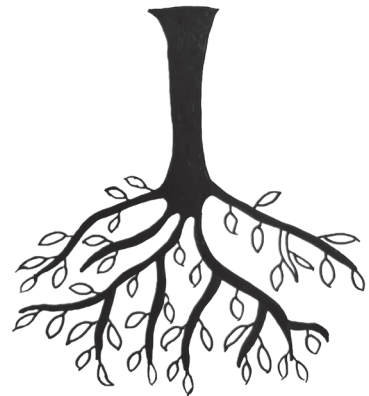
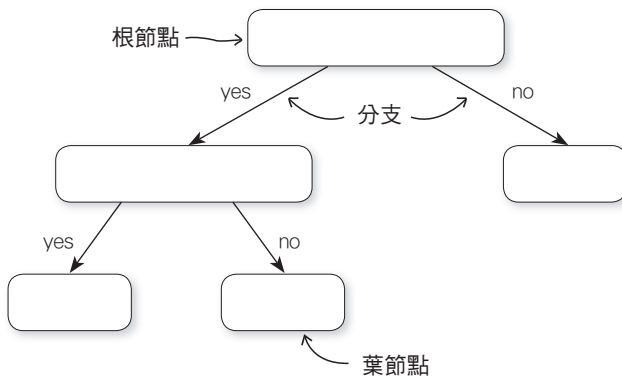
$$f(y) = \begin{cases} \text{好喝} & \text{for } y \geq 0.5 \\ \text{不好喝} & \text{for } y < 0.5 \end{cases}$$

店長看了後卻覺得相當頭痛，因為他根本搞不懂你在寫什麼，氣得要你回去重做，於是你改用決策樹，得出了以下結果：



店長看了之後滿意多了，因為他可以很清楚的知道，只要泡咖啡的溫度保持在 82.5~87.5°C，並把氣壓控制在 9.5~16.5bar 就能煮出好喝的咖啡。

以上就是一個決策樹的例子，它最大的優點就是能很好的解釋分類的標準，並且將其視覺化，透過一個一個問題把輸入的資料分類成你期望的樣子，之所以被稱為決策樹就是因為它生成的分類標準長得像一顆倒過來的樹：



我們稱輸入資料為**根節點**、問題為**分支**、結果為**葉節點**，資料進入決策樹後就會透過分支進行分類，最終輸出到葉節點，即為分類結果。

不過機器要怎麼知道該問什麼問題，以及先問什麼問題呢？首先我們得先了解熵是什麼。

熵 Entropy

熵(Entropy)是化學及熱力學中用來衡量系統混亂程度的指標，簡單來說熵越大，就代表越混亂。而在資訊領域中熵為**資訊的平均量**，或是**訊息不確定性的量度**，因此你也可以把它理解成是亂度，因為當資料越混亂，其資訊量也越大。例如一張九九乘法表對比一本深度學習的書，後者的資訊含量一定更大也更混亂，我們就說深度學習的書有比較大的熵。

那麼要怎麼把熵量化呢？我們可以透過以下問題進行發想：假設你是某某學校 A 班的學生，而你想知道 B 班此次的班排第一名是誰，你有一個朋友在 B 班，但他只告訴你前 4 名的名字(甲、乙、丙、丁)，剩下的要你自己去猜，他讓你可以問他是非題問題，不過每問一次就要請一杯飲料，那你至少要請他幾杯飲料，才能確保會得到正解呢？

答案是 2 杯飲料，這兩個問題是：是甲或乙嗎？、是丙或丁嗎？用數學式計算就是： $\log_2 4 = 2$ ，因為每個是非題可以將資料一分為二，而 4 個資料只要 2 個問題就能分完，得出來的數值其實就是熵，即你要知道這個資訊的平均代價。然而這個計算方式是假設每個人得第一名的機率都一樣，如果你先前就知道某一最人最常得第一名，那你就有機會用更少的飲料得到答案，因此應該把機率也考量進公式中，修正後的式子為： $\frac{1}{4}\log_2 4 + \frac{1}{4}\log_2 4 + \frac{1}{4}\log_2 4 + \frac{1}{4}\log_2 4 = 2$ (機率一樣的情況)，所以如果甲比其他人多一倍的機率得第一，答案就會是： $\frac{2}{5}\log_2 \frac{5}{2} + \frac{1}{5}\log_2 \frac{5}{1} + \frac{1}{5}\log_2 \frac{5}{1} + \frac{1}{5}\log_2 \frac{5}{1} \cong 1.92$ ，會得到比較低的熵。最後我們得知熵的公式為：

$$\text{Entropy} = \text{機率}_0 \times \log_2 \frac{1}{\text{機率}_0} + \text{機率}_1 \times \log_2 \frac{1}{\text{機率}_1} \dots + \text{機率}_n \times \log_2 \frac{1}{\text{機率}_n}$$

正式的寫法為：

$$\text{Entropy} = - \sum_{i=0}^c p(i|t) \times \log_2 p(i|t)$$

熵在決策樹之中是很重要的指標，決策樹的目的就是讓熵最小化，我們期望未經過分類的資料從根結點到葉節點的過程中，能不斷下降熵，這就是決定決策樹該問什麼問題的關鍵。

我們實際用一組資料來計算看看，以下是隨機挑選 10 個路人，並詢問他們對於超級英雄電影中的角色偏好：

性別	大於 20 歲	比較喜歡的角色
男	是	鋼鐵人
男	是	鋼鐵人
女	是	美國隊長
男	否	鋼鐵人
女	是	美國隊長
女	否	鋼鐵人
男	否	鋼鐵人
女	是	美國隊長
男	是	鋼鐵人
女	是	美國隊長

這組資料中偏好鋼鐵人與美國隊長的人數是 6：4，因此未經過分類的熵為：

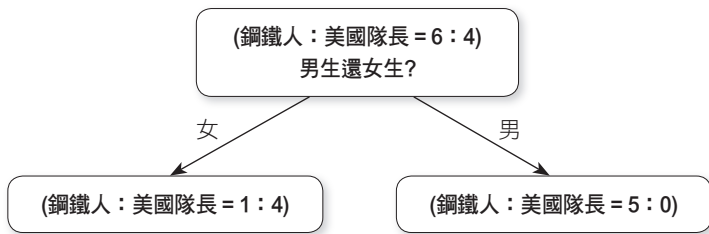
$$\frac{6}{10} \log_2 \frac{10}{6} + \frac{4}{10} \log_2 \frac{10}{4} \cong 0.97 \quad (\text{這裡的機率要代入該資料的占比})$$

\uparrow
 喜歡鋼鐵人

\uparrow
 喜歡美國隊長

得知熵為 0.97 後，接下來決策樹就要找出能將熵下降最多的問題，而這個判斷標準就稱為**資訊增益**。

.....



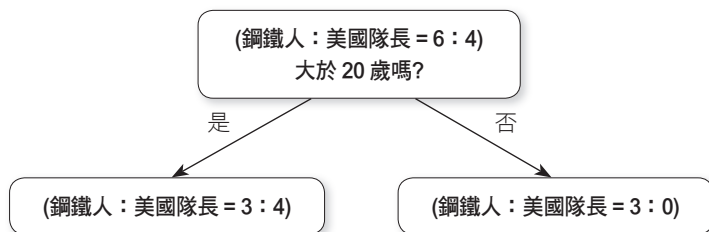
$$\text{女生的熵} = \frac{1}{5} \log_2 \frac{5}{1} + \frac{4}{5} \log_2 \frac{5}{4} \cong 0.72$$

$$\text{男生的熵} = \frac{5}{5} \log_2 \frac{5}{5} + \frac{0}{5} \log_2 0 = 0$$

$$\text{資訊增益} = 0.97 - \left(\frac{5}{10} * 0.72 + \frac{5}{10} * 0 \right) = 0.61$$

原本的熵 女生的熵 男生的熵
 女生的比例 男生的比例

問題 2：大於 20 歲嗎？



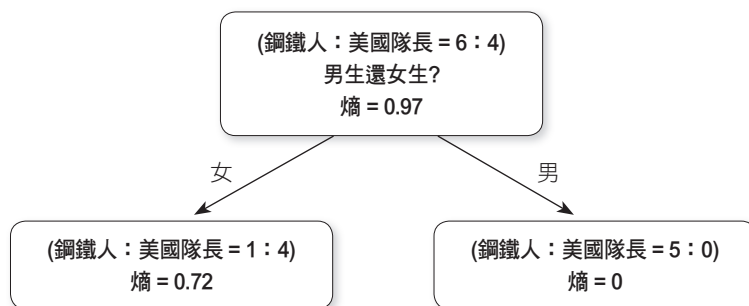
$$\text{大於 20 歲的熵} = \frac{3}{7} \log_2 \frac{7}{3} + \frac{4}{7} \log_2 \frac{7}{4} \cong 0.98$$

$$\text{小於 20 歲的熵} = \frac{3}{3} \log_2 \frac{3}{3} + \frac{0}{3} \log_2 0 = 0$$

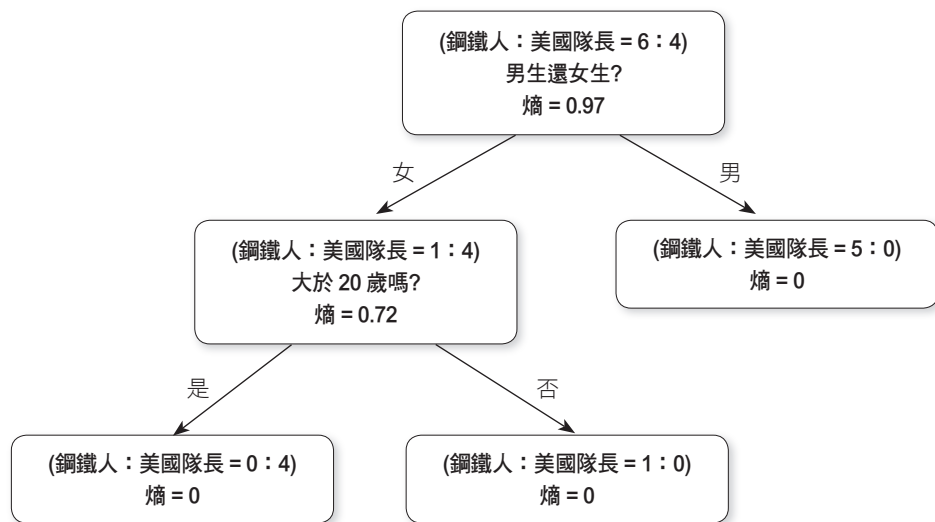
$$\text{資訊增益} = 0.97 - \left(\frac{7}{10} * 0.98 + \frac{3}{10} * 0 \right) \cong 0.28$$

原本的熵 大於 20 歲比例 大於 20 歲的熵 小於 20 歲比例 小於 20 歲的熵

從上面的計算可以知道問題 1 的資訊增益比較高，所以應該優先選擇問題 1，因此我們的決策樹目前會長這樣：



當節點的熵為 0 時即為葉節點，不再進行分類。接著我們把問題 2 加進熵還不是 0 的節點下面：



現在所有節點的熵都為 0，代表此決策樹已建立成功。最後我們得知，這組資料中只要是男生都喜歡鋼鐵人，而女生如果大於 20 歲喜歡美國隊長、小於 20 歲則喜歡鋼鐵人。

不過以上的資料有一個主觀判斷：年齡是否大於 20 歲，真正的資料應該更接近右表：

性別	年齡 (歲)	比較喜歡的角色
男	32	鋼鐵人
男	25	鋼鐵人
女	26	美國隊長
男	19	鋼鐵人
女	28	美國隊長
女	18	鋼鐵人
男	17	鋼鐵人
女	22	美國隊長
男	29	鋼鐵人
女	30	美國隊長

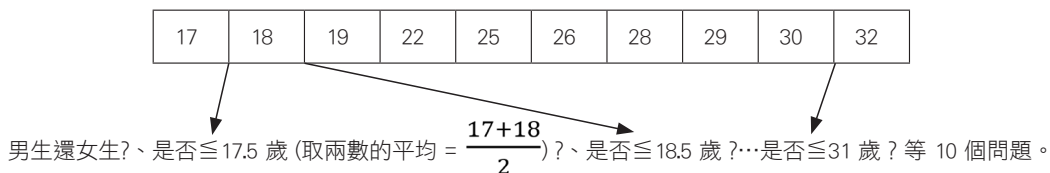
此時決策樹必須先將數值資料排序, 例如年齡資料原本為：

32	25	26	19	28	18	17	22	29	30
----	----	----	----	----	----	----	----	----	----

要排序成：

17	18	19	22	25	26	28	29	30	32
----	----	----	----	----	----	----	----	----	----

而問題就不會是原本的 2 個, 會增加成：



最後整理一下決策樹的生成流程：

- ① 將數值資料排序
- ② 提出所有問題
- ③ 一一計算各個問題的資訊增益
- ④ 選擇資訊增益最大的問題作為分支
- ⑤ 節點熵為 0 時, 成為葉節點
- ⑥ 熵不為 0 的節點重複步驟 2~5

以下我們使用 Python 的機器學習套件, scikit-learn 來實作一次。首先確保你的 Python 環境有安裝 scikit-learn, 如果沒有, 請使用 pip3 安裝：

```
$ pip3 install scikit-learn
```

確認安裝好後，在 Python 環境中輸入以下指令，導入決策樹函式：

```
from sklearn.tree import DecisionTreeClassifier
```

然後我們手動創造以上英雄偏好的資料，首先創造特徵資料，由於 scikit-learn 的特徵資料僅允許輸入數值資料，因此我們用 0 代表女生，用 1 代表男生：

```
feature = [[1, 32], [1, 25], [0, 26], [1, 19], [0, 28], [0, 18], [1, 17],
           [0, 22], [1, 29], [0, 30]]
```

接著創造目標資料 (正確答案)，雖然目標資料允許輸入非數值資料，但為了節省時間，和方便之後視覺化，我們用 0 代表美國隊長，用 1 代表鋼鐵人：

```
target = [1, 1, 0, 1, 0, 1, 1, 0, 1, 0]
```

使用 scikit-learn 的決策樹物件：


```
tree = DecisionTreeClassifier(criterion='entropy')
```

使用 entropy 作為分類標準

使用創造的資料生成決策樹：

```
tree.fit(feature, target)
```

訓練完後，使用以下指令輸入特徵資料來看結果：

特徴資料 

```
prediction = tree.predict(feature)
```

```
print(prediction) ← 輸出: [1 1 0 1 0 1 1 0 1 0]
```

可以看到結果完全相符，如果想看決策樹的分類過程可以用：

```
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file='hero.dot',
                feature_names=['gender', 'age'],
                class_names=['Captain America', 'Iron Man'])
```

特徵名稱 檔名 第 0 個特徵名

from sklearn.tree import export_graphviz ← 導入視化工具

export_graphviz(tree, out_file='hero.dot', 第 1 個特徵名
 feature_names=['gender', 'age'],
 class_names=['Captain America', 'Iron Man'])

類別名稱 第 0 個類別名 第 1 個類別名

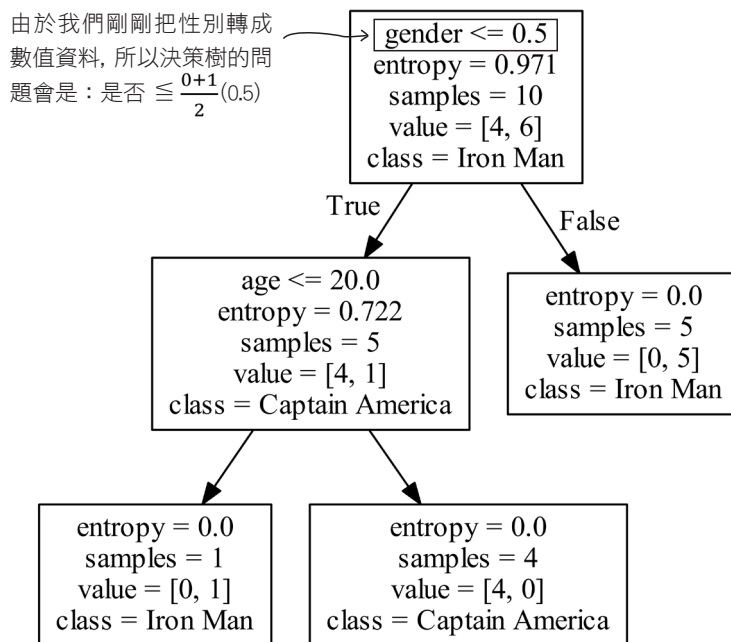
這樣就會生成一個 "hero.dot" 的檔案, 不過直接看的話可能不好理解, 因此可以使用 graphviz 工具幫忙視覺化, graphviz 需要先安裝, 使用以下連結並選擇自己的系統進行安裝: <https://www.graphviz.org/download/>, 如果使用 Anaconda 的話可以直接在 Anaconda Prompt 中輸入:

```
conda install graphviz
```

接著開啟 terminal 或 Anaconda Prompt, 並移動到 "hero.dot" 的目錄下, 輸入以下指令, 即可生成視覺化的決策樹:

```
$ dot -Tpdf hero.dot -o hero.pdf
```

開啟 "hero.pdf" 能看到以下圖片:



這個決策樹與預期的一模一樣, 完整的程式碼請參考:

https://www.flag.com.tw/download/decision_tree_heros.zip

隨機森林

俗語云：「三個臭皮匠，勝過一個諸葛亮」，這句話主要是想表達：人多力量大，雖然一己之力可能不怎麼樣，但只要集眾人之力，便能成就強大的力量。其實在機器學習領域中，也有類似的例子。

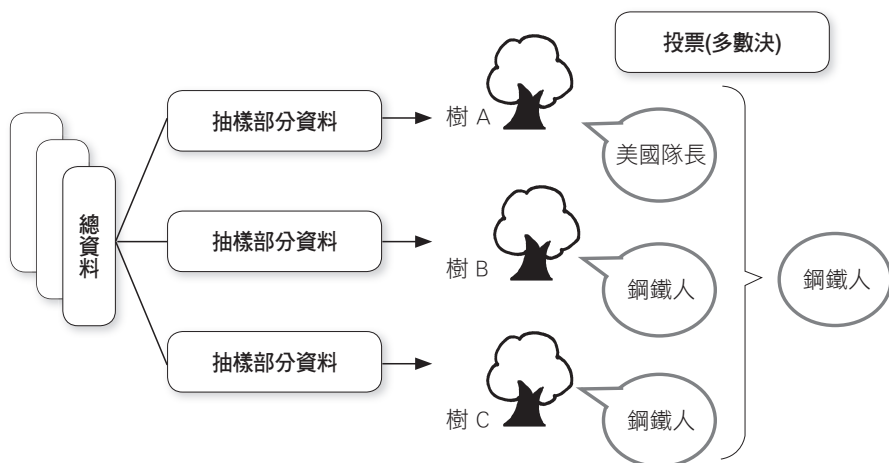
很多人，稱為人群，很多樹稱為什麼？答案就是森林！通常一棵決策樹能帶來的效果很有限，不過如果好幾棵決策樹同時發威，效果就不同凡響了，這個前提是每棵決策樹的分類方法不能一模一樣，否則也沒什麼意義，因此我們要加入一些隨機性，讓每棵決策樹都有不同的生長環境，這樣一來，就能集結眾樹的力量，打造強大的分類器，這就是**隨機森林 (Random Forest)**。

隨機森林的隨機是指：

- ① 從總資料中隨機抽取部分樣本 (例如總資料有 10 筆，從中取 5 筆)。
- ② 從一個樣本中隨機抽取部分特徵來生成一棵決策樹 (例如原本有 5 個特徵：性別、年齡、身高、體重、職業，從中取 3 個特徵：性別、年齡、職業)。

如果重複以上的步驟 n 次就能生成 n 棵決策樹，由於每棵樹的資料都不太一樣，所以樹的樣子也會不太一樣。你可以理解成，一棵樹只用了部分資料學習，所以只專精在部分的資料分類，而一整個森林就有各個領域的好手，只要把它們的意見統合起來，就是一個思考周全的結果。

要分類資料的時候，會把資料丟進每棵樹中，不同的樹可能會有不同的結果，最後讓它們來投票，採多數決為隨機森林的最終結果，我們可以透過下圖來理解：



以下我們使用 scikit-learn 內建的葡萄酒資料庫, 來建立一個隨機森林分類器, 並與決策樹比較結果。

決策樹 VS 隨機森林

首先載入葡萄酒資料 wine：

```
from sklearn import datasets
wine_data = datasets.load_wine()
```

其中 wine_data.feature_names 是特徵名稱、wine_data.target_names 是類別名稱、wine_data.data 是特徵資料、wine_data.target 是目標資料。輸入以下程式碼來查看一些資訊：

```
print(wine_data.data.shape) ← 輸出：(178, 13)
```

```
print('特徵名稱:', wine_data.feature_names) ←
```

輸出： 特徵名稱: ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']

```
print('類別名稱:', wine_data.target_names) ← 輸出： 類別名稱:
['class_0' 'class_1' 'class_2']
```

從輸出中我們可以得知, 總共有 178 筆資料, 一筆資料有 13 個特徵, 分別是: alcohol (酒精濃度)、malic_acid (蘋果酸含量)、ash (灰分量)…等, 而這些資料被分為 3 個類別。

為了讓我們之後能評估分類器的效果, 所以要先將資料分割成訓練資料和測試資料, 我們會使用訓練資料來生成決策樹和隨機森林, 並用測試資料來測試分類效果。以下直接使用 scikit-learn 提供的資料分割函式：

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

訓練資料特徵	測試資料特徵	訓練資料目標	測試資料目標
↓	↓	↓	↓
train_feature, test_feature, train_target, test_target = train_test_split(
wine_data.data, wine_data.target, test_size=0.3			
)			
	↑	↑	↑
	總資料特徵	總資料目標	使用 30% 的資料作為測試用

用分割好的資料先建立一個決策樹：

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree = DecisionTreeClassifier(criterion='entropy', max_depth=4) ←  
tree.fit(train_feature, train_target)
```

限制決策樹的深度為 4 層，
避免樹生長過大

小編補充：其它更多決策樹的參數，可以參考以下網址：<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

接著建立一個隨機森林：

```
from sklearn.ensemble import RandomForestClassifier
```

```
forest = RandomForestClassifier(criterion='entropy', n_estimators=10,  
                               max_depth=4)
```

此參數代表一個森林
中要有幾棵樹

```
forest.fit(train_feature, train_target)
```

一樣限制每棵決策
樹的深度為 4 層

最後我們利用分類器的 `score()` 函式來評估效果，此函式要傳入特徵資料和目標，它會將特徵資料進行分類，並比較與結果的差異，然後輸出分類的準確率，以下我們分別評估決策樹和隨機森林的準確率：

```
accuracy_tree = tree.score(test_feature, test_target)  
accuracy_forest = forest.score(test_feature, test_target)
```

↑ 測試資料的特徵 ↑ 測試資料的目標

```
print('決策樹的準確率:', accuracy_tree) ← 輸出： 決策樹的準確率:  
print('隨機森林的準確率:', accuracy_forest) ← 0.8888888888888888
```

輸出： 隨機森林的準確率:
0.9814814814814815

完整程式碼請參考：https://www.flag.com.tw/download/random_forest.zip
從以上的輸出可以看到隨機森林的效果明顯要優於決策樹，可見眾樹的力量不可小覷。隨機森林是很強大的機器學習方法，在深度學習出現之前，它也曾經是一方霸主。

為什麼每次執行的結果都不一樣？

你可能會發現每次執行的結果都不太一樣, 這是因為 `train_test_split()` 會隨機選擇資料進行分割, 此外, `scikit-learn` 為了加速決策樹的生成, 也加入了隨機因素 (隨機抽取部分資料), 雖然可能會降低準確率, 卻能大幅縮短訓練時間, 不過大多數結果中, 隨機森林都還是會比決策樹好, 如果想讓每次的結果一樣, 可以在各函式中指定 `random_state` 這個參數：

```
train_feature, test_feature, train_target, test_target = train_test_split(
    wine_data.data, wine_data.target, test_size=0.3, random_state=0
)
tree = DecisionTreeClassifier(criterion='entropy', max_depth=4,
                             random_state=0)
forest = RandomForestClassifier(criterion='entropy', n_estimators=10,
                               max_depth=4, random_state=0)
```

—— 以上 3 個函式中都指定 `random_state=0`

