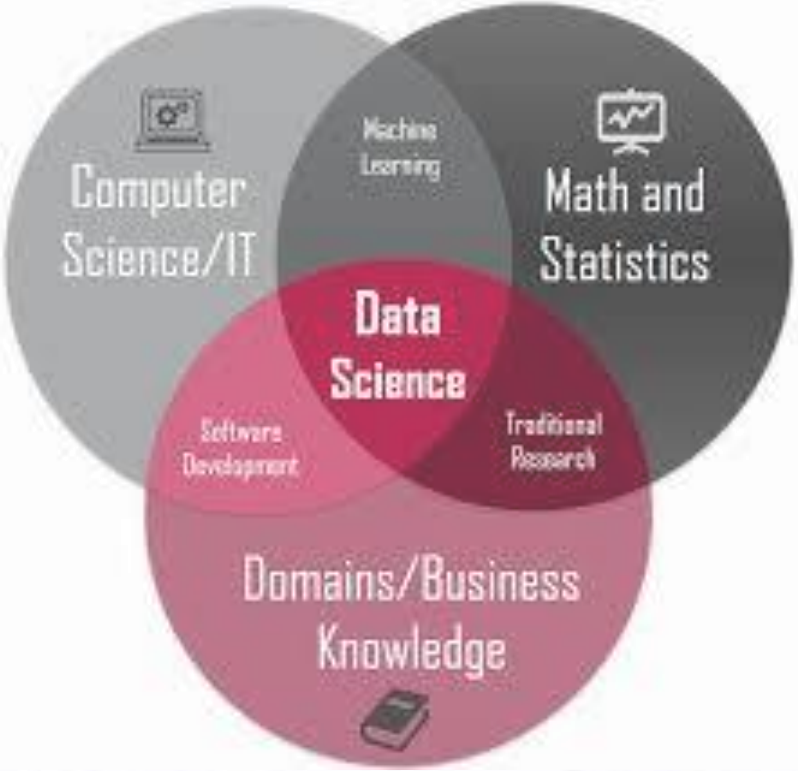
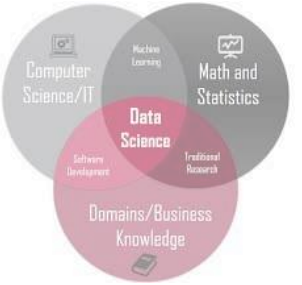
AI & DS



U08-機器學習:監督式學習分類演算法

2023.05_V1.1

單元大綱



- Scikit-Learn 資料集
- K 近鄰演算法
- 單純貝氏演算法
- 決策樹演算法
- 隨機森林演算法

Part 1 Scikit-Learn 資料集

Scikit-Learn 資料集

- sklearn.datasets 提供了一些資料集，可在〈Dataset loading utilities〉取得資料集的相關說明，透過 load_*、fetch_*、make_* 函式取得。
- sklearn.datasets 提供有四類資料集：

1.Toy datasets(玩具資料集)：

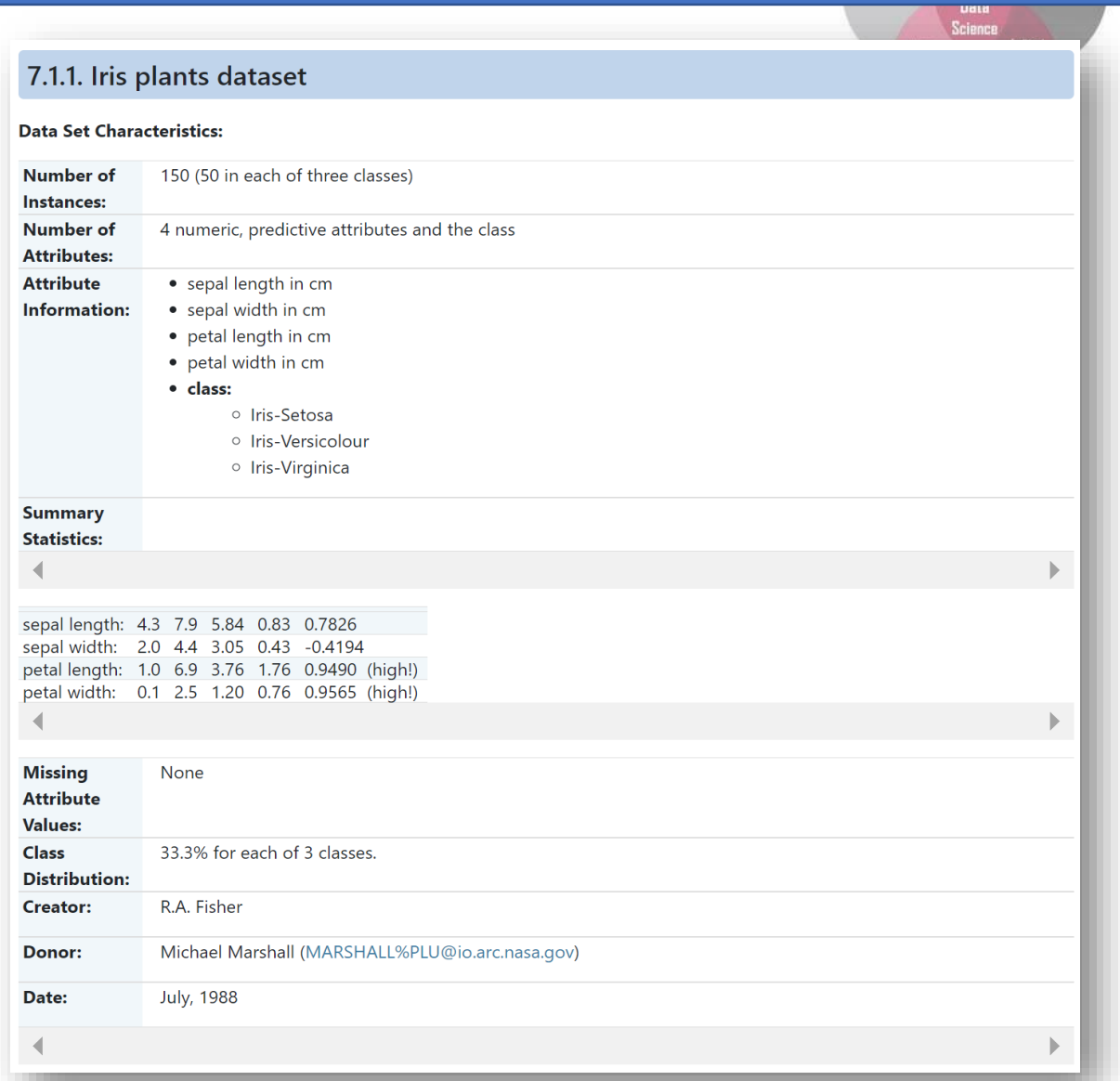
Ex:鳶尾花數據集, 手寫數字數據集的光學識別, 葡萄酒識別, 乳腺癌(診斷) 等

2.Real world datasets(真實世界的數據集)：

Ex:Olivetti 人臉數據集, 20 個新聞組文本數據集,加州住房數據集等。

3. Generated datasets (生成數據集)

4. Loading other datasets(加載其他數據集)



7.1.1. Iris plants dataset

Data Set Characteristics:

Number of Instances:	150 (50 in each of three classes)
Number of Attributes:	4 numeric, predictive attributes and the class

Attribute Information:	• sepal length in cm
	• sepal width in cm
	• petal length in cm
	• petal width in cm
	• **class:**
	  ○ Iris-Setosa
	  ○ Iris-Versicolour
	  ○ Iris-Virginica

Summary Statistics:	

sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

Missing Attribute Values:	None
Class Distribution:	33.3% for each of 3 classes.
Creator:	R.A. Fisher
Donor:	Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
Date:	July, 1988

[Ref]: https://scikit-learn.org/stable/datasets.html

Scikit-Learn 資料集的分類 (1/2)

- Scikit-Learn 資料集分為小規模與大規模兩類:

1.小規模資料集:已經安裝在Scikit-Learn模組內，資料數量少

2.大規模資料集:需要從網路上下載，資料數量大

小規模資料集

- 小規模資料集的模組為: load_ 名稱

  Ex:鳶尾花資料集為: load_ iris ,波士頓房價資料集為: load_ boston

1.首先載入資料集模組, 語法為:

```
from sklearn.datasets import 資料集模組
```

- Ex: from sklearn.datasets import load_ iris

2.取得資料集, 語法為:

```
資料集變數 = 資料集模組()
```

Ex: iris = load_ iris

3.資料集變數常用屬性:

data : 特徵值串列　　target: 目標值串列　　DESCR: 資料集描述

feature_names: 特徵欄位名稱　　target_names: 目標值名稱

Scikit-Learn 資料集的分類 (2/2)

大規模資料集

- 大規模資料集的模組為: fetch_ 名稱
  Ex:新聞資料集為: fetch_ 20newsgroups

1.首先載入資料集模組, 語法為:

```
from sklearn.datasets import 資料集模組
```

- Ex: from sklearn.datasets import fetch_ 20newsgroups

2.取得大規模資料集, 語法為:

```
資料集變數 = 資料集模組(data_home=存檔路徑, subset=資料種類)
```

3.資料集變數常用屬性:

data_home :此參數設定下載資料集檔案儲存的路徑, 預設值為: **None**。

subset :此參數設定下載資料集的種類, 有三種:

- **train**: 只下載訓練資料
- **test**: 只下載測試資料
- **all(預設值)**: 下載全部資料

※ 注意: 執行下載資料集時, 系統會檢視儲存路徑中是否已經有資料集檔案存在, 若沒有就會由網路下載, 並存於指定路徑。

Scikit-Learn 資料集的分割(1/2)

- **訓練集（Training set）**
  用來訓練的已知資料，用來訓練機器學習，主要是靠答案的回饋來調整函數的參數來矯正誤差。
- **測試集（Test set）**
  用來測試的未知資料，讓完成學習的機器使用函數做預測，測試其準確性。
- 訓練集有可能再細分出驗證集（Validation set），如果沒有要調整 Hyperparameter（由人自由決定的參數）的話可以不用驗證集。
- 如何分成訓練集和驗證集, 方法有下列3種：

1.Holdout 驗證

直接用比例分成訓練集和測試集，通常測試集會取比較小的部分，像是30%或20%。

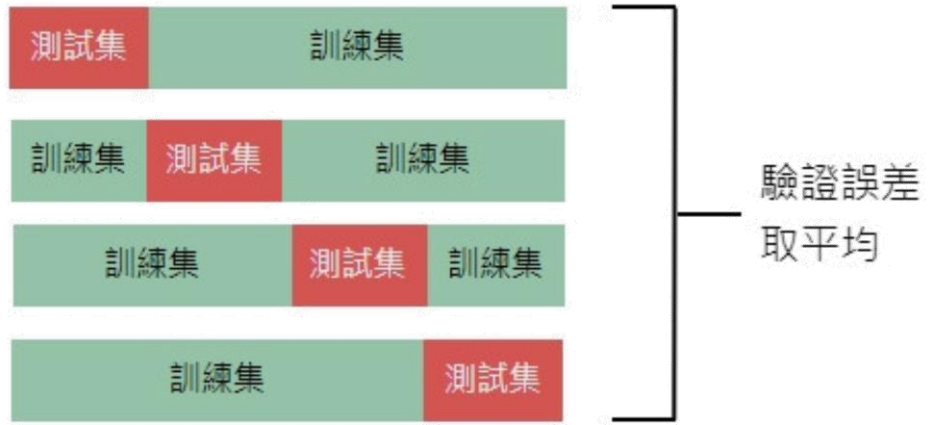
2.K 折交叉驗證（K-Fold Cross Validation）

將資料分成 K 等分，一次取一份測試，剩下做訓練，最後取 K 次的驗證誤差（Validation Error）的平均。

3.Leave one out

其實就是 K 折交叉驗證，只是 K = 所有資料數，一次取一筆資料當測試，剩下做訓練。

比如說100筆資料取4等分做4折交叉驗證：



100筆資料做100折交叉驗證就是 Leave one out

[Ref] https://ithelp.ithome.com.tw/m/articles/10298747

Scikit-Learn 資料集的分割(2/2)

- Scikit-Learn提供: train_test_split 模組, 可隨機分割資料集。

1.首先載入 train_test_split 模組, 語法為:

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

2. train_test_split 模組進行資料集分割語法為:

```
訓練特徵, 測試特徵, 訓練目標, 測試目標 = train_test_split ( 原始特徵,
                原始目標, test_size = 數值, random_state = 數值)
```

訓練特徵, 測試特徵, 訓練目標, 測試目標: 分割後傳回的訓練及測試資料

原始特徵, 原始目標: 分割前的原始資料

test_size : 設定測試資料的比例, 通常數值在 0.1 - 0.3之間

random_state :亂數種子(可省略)。若設定此參數, 代表使用者希望以相同的資料進行訓練,以比較模型準確度時使用。

Part 2

K 近鄰演算法

K近鄰演算法(KNN, K Nearest Neighbor)介紹

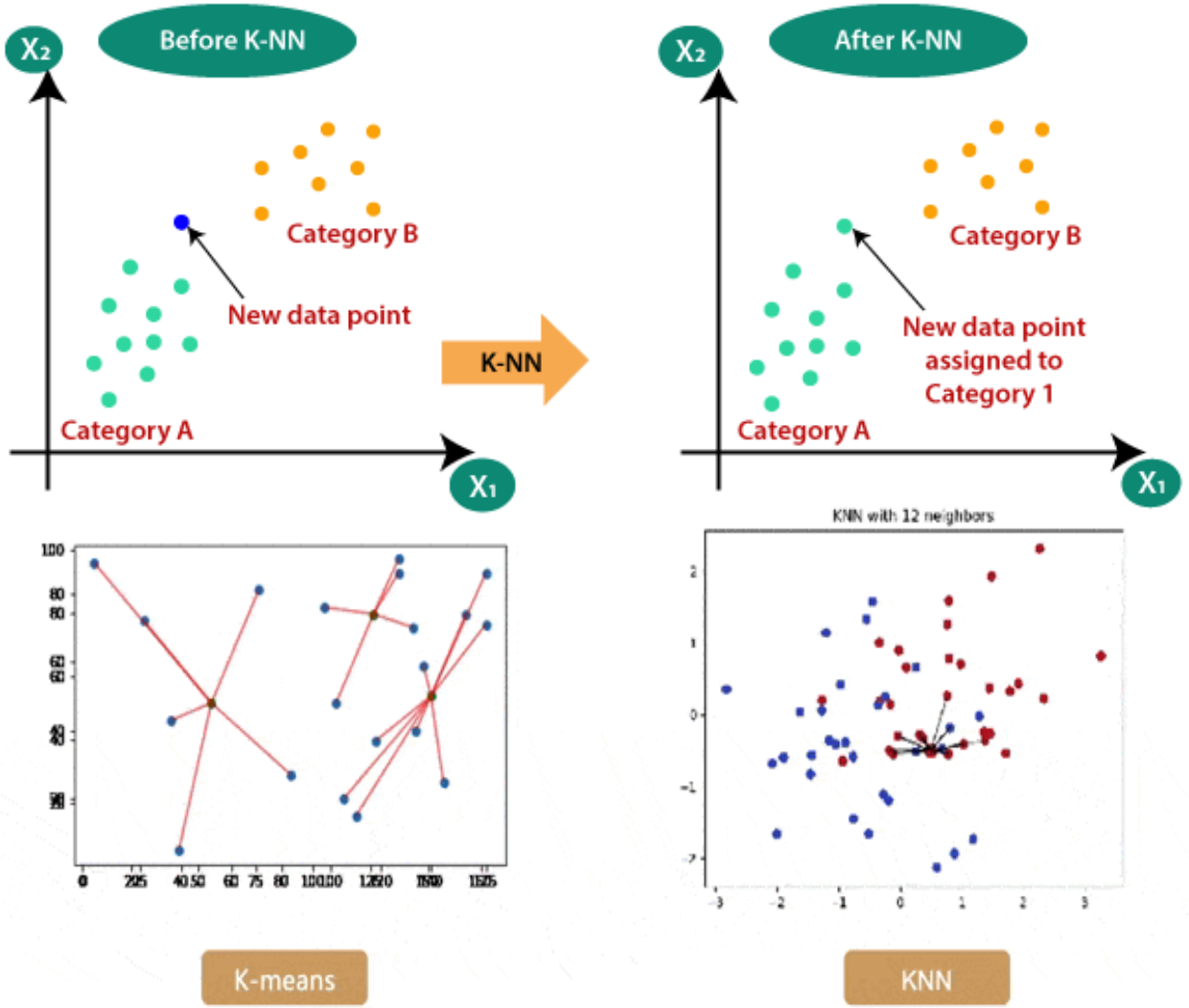
- KNN 的全名 K Nearest Neighbor 是屬於機器學習中的 Supervised learning 其中一種算法，顧名思義就是 k 個最接近你的鄰居。分類的標準是由鄰居「多數表決」決定。
- **KNN的運算步驟:**

步驟1. 計算每個點之間的距離
步驟2. 用K值決定鄰居數目，並進行投票
(在連續型資料中，則是計算平均數)
步驟3. 以投票結果決定類別

- **KNN 的缺點**:計算量龐大,且對資料的局部結構非常敏感，因此調整適當的 k 值極為重要。
- **KNN(監督式) v.s K-means(非監督式)**

KNN 的 k 是設定鄰居的數量採多數決作為輸出的依據。而 K-means 的 k 是設定集群的類別中心點數量。

※特徵的數值大小會影響歐式距離, 如果各特徵值差異較大時,在K近鄰演算法計算歐式距離前,須將資料進行特徵標準化。



KNN 演算方式

S-1.衡量尺度(Scalling):

1.**標準化(Standardization)**:

適用於近似常態分配的定量變數，計算:

x_i 離 $\bar{x}$ 有多少標準差

其中x_i :原始值, $\bar{x}$:平均數, s:標準差

臨界值 $z = \frac{x_i - \bar{x}}{s}$, $s \neq 0$

2.**正規化(Normalization)**:

定量變數若未近似常態分配也可適用，此方法會將數據收斂到0到1之間。

$$\frac{x_i - \text{最小值}}{\text{最大值} - \text{最小值}} \in [0,1]$$

3.**平均值正規化**:

此方法會將數據收斂到-1到1之間, 且平均值為0。

$$\frac{x_i - \text{平均值}}{\text{最大值} - \text{最小值}} \in [-1,1]$$

S-2.決定權重:

目的是越接近的樣本影響力愈大。

S-3.計算距離:

最常使用的是**歐基里德距離(Euclidean Distance)**:

兩點間最短的距離，也就是斜邊距離。

AB距離 $= \sqrt{(X_B - XA)^2 + (Y_B - YA)^2}$

Scikit-Learn 的KNN模組(1/2)

1. 載入Scikit-Learn的KNN模組

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

2. 建立KNeighborsClassifier物件

```
近鄰變數= KNeighborsClassifier (n_neighbors=數值,
                algorithm=演算法,  weights=權重計算方式)
```

n_neighbors: 此參數即k值,就是設定取幾個最接近的資料。

algorithm: 設定使用的演算法,可以設定: auto(預設值), ball_tree, kd_tree及brute。
若設為auto, 表示由系統根據資料特性自動判斷使用何種演算法。

weights: 設定資料的權重。設定值有:
uniform(預設值): 所有資料的權重都相同。
distance: 歐式距離越小的資料,其權重越大。

Scikit-Learn 的KNN模組(2/2)

3.利用fit ()方法進行訓練

近鄰變數. fit (訓練資料, 訓練目標值)

4. 使用predict方法對未知資料進行預測

預測變數= 近鄰變數.predict(預測資料)

5. 或使用score()方法對未知資料進行預測, 並計算準確率

準確率變數= 近鄰變數.score(預測資料, 預測目標值)

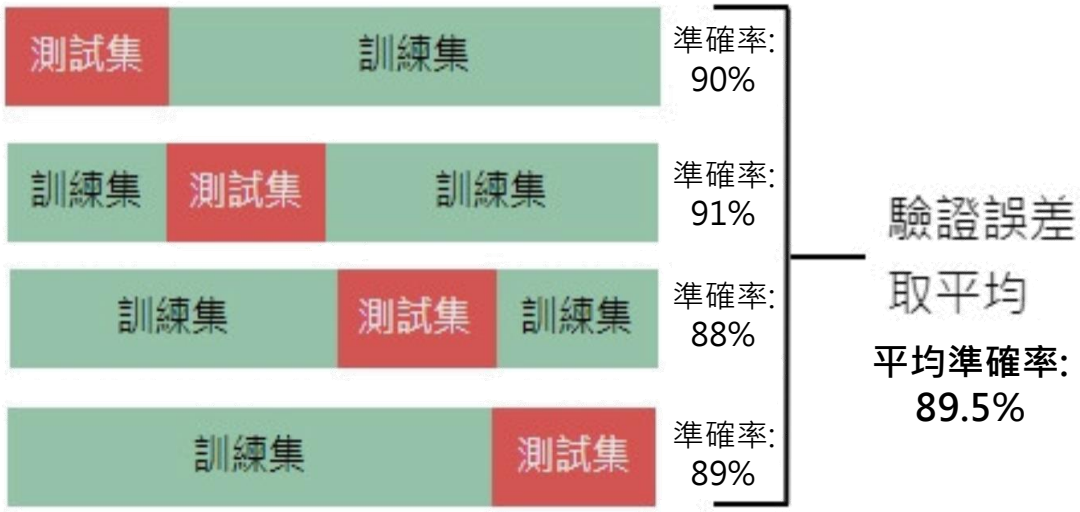
交叉驗證與網格搜索

- **交叉驗證(Cross Validation)**
  是將訓練資料分為為N等份,其中一份作為驗證集,其他資料做為訓練集,進行訓練並驗證得到準確率。依序再將另一份做為驗證集,其他資料做為訓練集,如此反覆進行N次得到N個準確率。
- **K 折交叉驗證 （ K-Fold Cross Validation ）**
  將資料分成 K 等分，一次取一份測試，剩下做訓練，最後取K次的驗證誤差（ Validation Error ）的平均。
- **網格搜索(Grid Search)**
  是一種遍歷搜尋法,在所可能參數值組合中, 嘗試每一種可能的參數值組合,並採取交叉驗證進行評估。以KNN為例:
  - n_neighbors要測試:3, 5, 10 三個值
  - weights: 要測試uniform, distance 兩個值

  組合後共有: (3 uniform), (3 distance)...(10 distance)
  共六組參數設定,利用網格搜索逐一進行交叉驗證,藉以找出做最佳化的參數組合。

※網格搜索適用於少於3-4個參數,若數量太大,會耗費太多時間。

比如說100筆資料取4等分做4折交叉驗證：



[Ref] https://ithelp.ithome.com.tw/m/articles/10298747

Scikit-Learn 的網格搜索

1. 載入Scikit-Learn同時執行交叉驗及網格搜索的模組: GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

2. 建立參數值字典:

```
參數變數={參數1:[值1, 值2, ...],參數2:[值1, 值2, ...],...}
```

Ex: parm={ 'n_neighbors' :[3,5,8,10] , ' weights' :['uniform' , 'distance']

3. 建立GridSearchCV物件:

```
網格變數=GridSearchCV(演算法物件變數, param_grid=參數變數, cv=數值)
```

4. 利用fit ()方法進行網格搜索

```
網格變數. fit (訓練值特徵, 訓練目標值)
```

搜索後結果有三:

- best_score: 交叉驗證中最佳準確率
- best_estimator:最佳參數組合
- cv_results_: 每次交叉驗證準確率

模型的儲存與讀取

模型建立後可儲存起來,無須每次都重新訓練,只需載入模型,即可對未知資料進行預測。

1. 儲存與讀取的模組: joblib (※ Colab 已經預設安裝,只需載入即可)

```
import  joblib
```

2. 使用joblib 的 dump方法,就可以儲存模型:

```
Joblib.dump (演算法物件變數, 儲存路徑)
```

Ex: joblib.dump(knn, ‘mnist500.pkl’) (※通常模型的副檔名為*.pkl)

3. 讀取模型則使用joblib 的 load方法

```
模型變數=Joblib.load (儲存路徑)
```

Ex: knnmodel= joblib.load (‘mnist500.pkl’)

Part 3

單純貝氏演算法

貝氏定理 Bayes’ Theorem介紹

- 貝氏定理描述在一些已知的條件下，某件事情發生的機率。
- 例如:已經知道房價與房子的區域位置有關，使用貝氏定理可透過得知房子的位置，更準確地推估房子的價格。
- [公式] $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$
- [意義] 若想要得知在已知B事件發生下，A事件發生的機率為何?
- [解釋]
  1. P(A) : 事前機率。在事件B發生之前，對事件A有一個基本的機率判斷。
  2. P(B) : 事前機率。在事件A發生之前，對事件B有一個基本的機率判斷。
  3. P(A|B): 在事件B發生之後，會對事件A發生的機率重新評估，稱P(A|B)為事件A的事後機率。
  4. P(B|A): 在事件A發生之後，會對事件B發生的機率重新評估，稱P(B|A)為事件B的事後機率。
- [綜合]
$$P(\text{假設}|\text{資料}) = \frac{P(\text{資料}|\text{假設}) \times P(\text{假設})}{P(\text{資料})}$$
- [意義]在貝氏學習裡，想要得到究竟是怎麼樣的假設，比較符合觀察到的資料，也就是要找到一個讓假設資料最高的”假設”。

[Ref] https://pyecontech.com/2020/02/27/bayesian_classifier/

單純貝氏分類(Naïve Bayes Classifier)演算法

- 單純是表示特徵獨立,即特徵之間沒有關聯。
- 貝氏演算法是基於機率理論的分類演算法。
- 常用於:文章分類, 新聞判別與垃圾郵件判別
- [原理]

$$P(R_i) = P(C_i) \times P(A1|C_i) \times P(A2|C_i)...$$

$P(R_i)$: 第i個類別的機率
$P(C_i)$: 第i個類別在原始資料中的比率。
$P(A1|C_i)$:第1個特徵在第i個類別的比率,餘則類推

- [範例]下表是某業務員推銷商品給客戶的資料表。請問:若有一新客戶,男性, 26歲, 已婚, 這位新客戶是否可賣出產品?

客戶	性別	年齡	婚姻	賣出商品
1	男	21	Y	N
2	男	28	N	Y
3	女	62	Y	N
4	男	19	N	N
5	女	27	Y	Y
6	女	35	Y	Y
7	女	42	N	Y

1.先計算賣出商品的機率$P(R_1)$: 紅字為合乎預測

客戶	性別	年齡	婚姻	賣出商品
2	男	28	N	Y
5	女	27	Y	Y
6	女	35	Y	Y
7	女	42	N	Y

$P(C_1)$: 賣出商品類別在原始資料中的比率: 4/7
$P(A_1|C_1)$: 性別特徵在賣出商品中的比率: 1/4
$P(A_2|C_1)$: 年齡特徵在賣出商品中的比率: 2/4
$P(A_3|C_1)$: 已婚特徵在賣出商品中的比率: 2/4

$$P(R_1) = (4/7) \times (1/4) \times (2/4) \times (2/4) = 0.035714$$

2.先計算未賣出商品的機率$P(R_1)$: 紅字為合乎預測

客戶	性別	年齡	婚姻	賣出商品
1	男	21	Y	N
3	女	62	Y	N
4	男	19	N	N

$P(C_2)$: 賣出商品類別在原始資料中的比率: 3/7
$P(A_1|C_2)$: 性別特徵在未賣出商品中的比率: 2/3
$P(A_2|C_2)$: 年齡特徵在未賣出商品中的比率: 1/3
$P(A_3|C_2)$: 已婚特徵在未賣出商品中的比率: 2/3

$$P(R_2) = (3/7) \times (2/3) \times (1/3) \times (2/3) = 0.063492$$

$\because P(R_2) > P(R_1)$ $\therefore$ 不會賣出商品!!

英文文句特徵處理

1. 載入Scikit-Learn的文句特徵處理模組- CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
```

2. 建立CountVectorizer物件

```
文句變數= CountVectorizer()
```

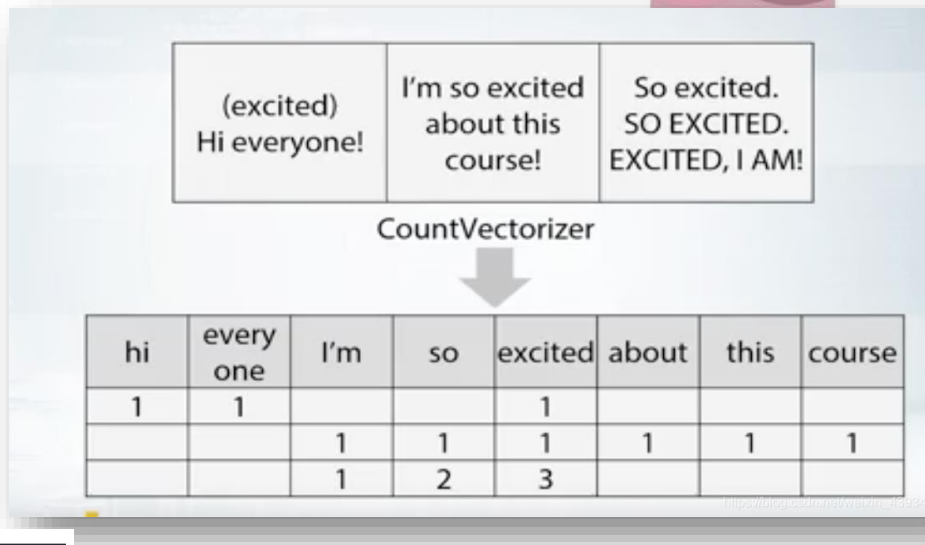
3. 利用fit_transform()方法進行轉換

```
數值變數=文句變數. fit_transform (文句串列)
```

※ 注意!! 這些單詞並不包含單一字母的單詞,例如: i, a 等。

從文本中提取特徵-詞袋法(Bag of word)

- **詞袋法(Bag of words)**
- 詞袋模型將文本轉化為向量，它不考慮文本中單詞的順序，只統計單詞在詞表中出現的次數，在sklearn中由CountVectorizer()函數實現。
- 用詞袋法統計出來的向量，每個單詞出現的次數相差可能很大，有個可能就幾次，有的可能幾十上百次，數字相差過大不利於機器學習訓練，因此需要做特徵縮放(feature scaling),這就引出了TF-IDF，它用的是頻率，而不是頻次，把兩個單詞的統計次數縮放到可比較的相同範圍內。



hi	every one	I'm	so	excited	about	this	course
1	1			1			
		1	1	1	1	1	1
		1	2	3			

```
1  from sklearn.feature_extraction.text import CountVectorizer
2
3  sentences = ['If equal affection cannot be, let the more loving be me',
4               'Life is a load of running, it is necessary to constantly in eac
5  vectorizer = CountVectorizer()
6  vector = vectorizer.fit_transform(sentences)
7  print(vectorizer.get_feature_names())
8  print(vector.toarray())
9
10 ['affection', 'be', 'cannot', 'choose', 'constantly', 'each', 'equal', 'fork
11 [[1 2 1 0 0 0 1 0 1 0 0 0 1 0 0 1 1 1 0 0 0 0 1 0]
12  [0 0 0 1 1 1 0 1 0 2 2 1 0 1 1 0 0 0 1 1 1 1 1 1]]
```

原句

特徵傳回值

與原句比對結果

[Ref] https://blog.csdn.net/weixin_43934593/article/details/103304983

中文文句特徵處理

1. 載入jieba的模組(Colab預設已經安裝)

```
import jieba
```

2. 利用 cut() 方法進行分詞,分詞後回傳值為產生器,轉換為串列。

```
串列變數= list(jieba.cut(中文文句段落))
```

3. 利用join()方法將單詞以空格分開的方式結合起來

```
中文變數= ‘ ’ .join (串列變數)
```

tf-idf文句處理

- CountVectorizer 模組統計單詞數量的方法有一個問題: 每一個單詞的重要性都相等, 這容易造成誤判。畢竟像因為, 所以, 我們...等單詞,在一篇文章出現的次數接近, 也無法說兩篇文章的內容就相似。
- 使用 tf-idf模組進行改善:
- tf : term frequence, 單詞頻率。表示單詞在一個文句中出現的次數。
- Idf: inverse document frequence, 逆文件頻率。若出現文句數量越大, 表示此單詞重要性越低。

1. 載入tf-idf的模組

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

2. 建立TfidfVectorizer 物件

```
文句變數= TfidfVectorizer()
```

3. 利用fit_transform()方法進行轉換

```
數值變數=文句變數. fit_transform (文句串列)
```

單純貝氏分類模組

1. 載入Scikit-Learn的單純貝氏演算法模組- MultinominalNB

```
from sklearn.naive_bayes import MultinominalNB
```

2. 建立MultinominalNB物件

```
貝氏變數= MultinominalNB (alpha=數值)
```

- alpha: 拉普拉絲平滑係數。用於修正當某個特徵在第i個類別的比率為0時, 會使第i個類別的機率為0的缺失。**預設值=1**。

3. 利用fit ()方法進行訓練

```
貝氏變數. fit (訓練資料,訓練目標值)
```

4.訓練完後,利用predict ()方法對未知資料進行預測

```
預測變數 = 貝氏變數. predict (預測資料)
```

5.使用score ()方法對未知資料進行預測, 並計算準確率

```
準確率變數 = 貝氏變數. score (預測資料, 預測目標值)
```

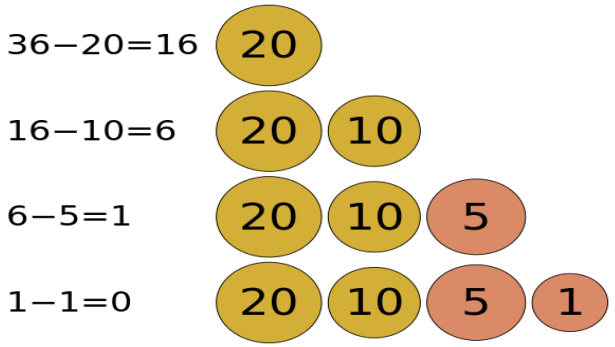
※ 優點: 依據古典機率理論, 穩定分類效率, 對缺失值不敏感。
缺點: 僅適用於特徵獨立的情況,若特徵之間具有高度相關,準確率會下降。

Part 4

決策樹演算法

決策樹(Decision Tree)介紹(1/6)

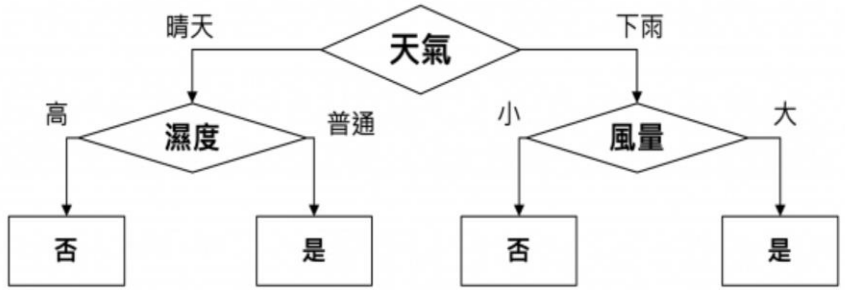
- 決策樹會根據訓練資料產生一棵樹，依據訓練出來的規則來對新樣本進行預測。
- **貪婪法則**：決策樹的生成是以一個貪婪法則來決定每一層要問什麼問題，目標是分類過後每一群能夠很明顯知道是屬於哪一種類別。
- [貪婪演算法] (greedy algorithm)，又稱貪心演算法，是一種在每一步選擇中都採取在當前狀態下最好或最佳（即最有利）的選擇，從而希望導致結果是最好或最佳的演算法。
- Ex:一般人換零錢的時候也會應用到貪婪演算法。把$36換散：$20 > $10 > $5 > $1



- [Ref] 1. https://ithelp.ithome.com.tw/articles/10271143
  2. https://ithelp.ithome.com.tw/articles/10204450

- [案例]
- 評估今天比賽是否舉行，天氣因子可能站比較大的因素，而 Co2 的濃度高低可能佔的因子程度較低。
- 第一層的決策中以天氣的特徵先進行第一次的決策判斷。
- 第二層再從所有特徵中尋找最適合的決策因子，直到設定的最大樹的深度即停止樹的生長。

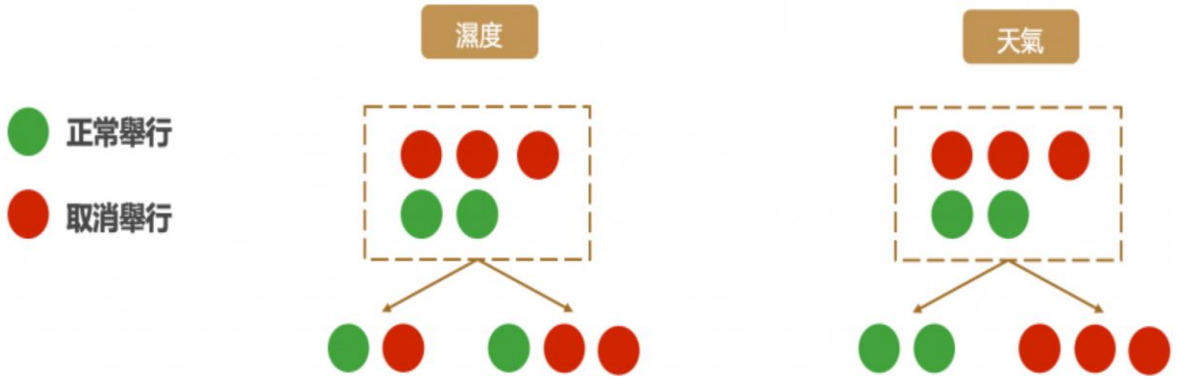
天氣	濕度	風量	是否舉行
晴天	高	大	否
陰天	低	小	是



決策樹(Decision Tree)介紹(2/6)

- 上述案例就是將天氣作為這一層判斷的因子。這就是決策樹在生成中的貪婪機制。要如何去判斷每次決策的好壞，就必須依靠亂度的評估指標。
- **評估指標**:
  - **Information gain** (資訊獲利)
  - Gain ratio (吉尼獲利)
  - **Gini index** (吉尼係數)
  = Gini Impurity (吉尼不純度)
- 依據訓練資料找出合適的規則，最終生成一個規則樹來決策所有事情，其目的使每一個決策能夠使訊息增益最大化。
- 常用 Information Gain 及 Gini Index 來定義分的好壞程度。

- [承上題]
- 以分類問題來說假設要評估明天比賽是否舉行。在樹的第一層節點中要從已知的兩個特徵分別是濕度與天氣選一個作為該層的決策因子。
- 假設目前訓練集有五筆資料，其中正常舉行的有兩筆資料，取消舉行的有三筆資料。
- 在樹的結構中左子樹為決策正常取行，而右子樹是決策取消舉行。可以發現當特徵為天氣的時候可以一很清楚的將這兩類別完整分開，因此會將天氣作為這一層判斷的因子。



[Ref] 1. https://ithelp.ithome.com.tw/articles/10271143
2. https://ithelp.ithome.com.tw/articles/10204450

決策樹(Decision Tree)介紹(3/6)

- **評估分割資訊量**:
- **Information Gain** 透過從訓練資料找出規則，讓每一個決策能夠使訊息增益最大化。其算法主要是計算熵，因此經由決策樹分割後的資訊量要越小越好。

$$Entropy = -\sum_j p_j \log_2 p_j$$

- **Gini** 的數值越大代表序列中的資料亂，數值皆為 0~1 之間，其中 0 代表該特徵在序列中是完美的分類。

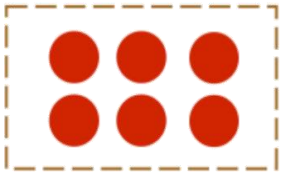
$$Gini = 1 - \sum_j p_j^2$$

- **熵 (Entropy)**
- 熵 (Entropy) 是計算 Information Gain 的一種方法。
- 在下圖公式中 p 代表是的機率、q 代表否的機率。從圖中範例很清楚地知道:
- 當所有的資料都被分類一致時➔ Entropy 即為 0
- 當資料各有一半不同時➔ Entropy 即為 1。

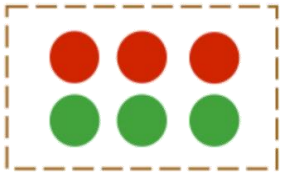
$$\text{Entropy} = -\sum p_j \log_2 p_j$$

$$\text{Information Gain} = -p * \log_2 p - q * \log_2 q$$

p：是的機率　q：否的機率



$$\text{Info}(6,0) = -\frac{6}{6}\log_2\left(\frac{6}{6}\right) - \frac{0}{6}\log_2\left(\frac{0}{6}\right) = 0$$



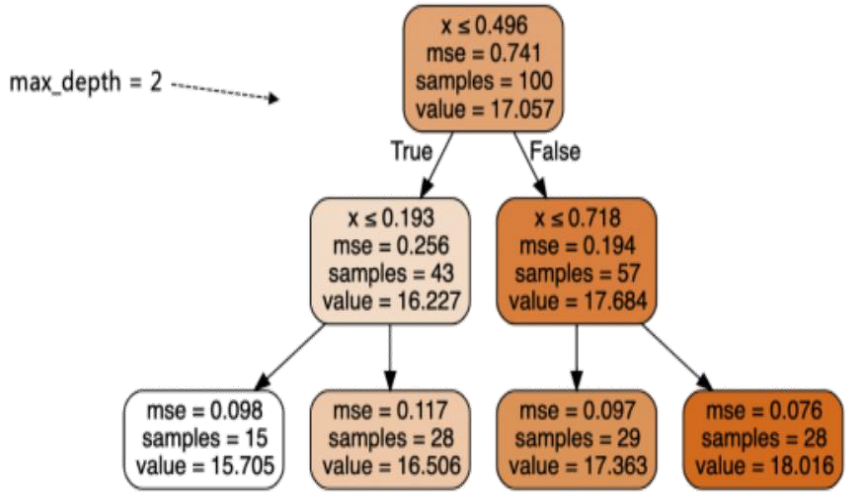
$$\text{Info}(3,3) = -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right) = 1$$

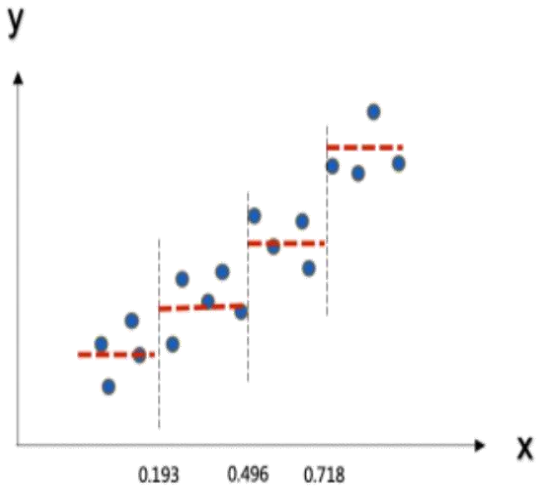
[Ref] 1. https://ithelp.ithome.com.tw/articles/10271143
2. https://ithelp.ithome.com.tw/articles/10204450

決策樹(Decision Tree)介紹(4/6)

- **迴歸樹**:
- 決策樹迴歸方法與分類有點類似差別僅在於評估分枝好壞的方式不同，稱作迴歸樹。當數據集的輸出爲連續性數值時，該樹算法就是一個迴歸樹。
- 透過樹的展開，並用葉節點的均值作爲預測值從根節點開始，對樣本的某一特徵進行測試。經過評估後，將樣本分配到其子結點。
- 每一個子節點對應著該特徵的一個值。依照這樣方式進行，直至到達葉結點。此時誤差值要最小化，並且越接近零越好。
- 樹越深模型越複雜, 且會有過度擬合的問題。

- [範例]
- 假設 x 是輸入 y 是輸出，在一個平面上繪製出資料與正確答案間的分佈。
- 假設迴歸樹的最大深度設定兩層。首先在第一層中會將所有的資料從中間切一刀此斷點為 x=0.496。當大於設定的值的數據點會繼續往右子樹下去延伸，反之小於 0.496 的資料點會往左子樹走。
- 一直不斷持續拓展直到設定的最大深度終止，此時的節點即為葉節點也就是最終的模型輸出值。





[Ref] 1. https://ithelp.ithome.com.tw/articles/10271143
2. https://ithelp.ithome.com.tw/articles/10204450

決策樹(Decision Tree)介紹(6/6)

- 迴歸樹該如何選擇切割點?:
- 在分類模型中決策樹是以亂度作為決策樹生成時候的評估指標。
- 迴歸樹透過是 MSE 或 MAE 來評估模型，並找出誤差最小的值作為樹的特徵選擇與切割點。

- Mean Square Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

- Mean Absolute Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

- 其中前者是均方差，後者是和均值之差的絕對值之和。

優點

- 簡單且高度可解釋性
- 低計算時間複雜度
- 每個決策階段都相當的明確清楚
- 幾乎沒有要調整的超參數

缺點

- 模型容易過度擬合
- 當標籤類別種類多時樹會很複雜

- 決策樹(Decision Tree)常見的三種算法(ID3、C4.5、CART)：

1. ID3:
   訊息熵表示的是所有樣本中各種類別出現的不確定性之和，熵越大代表不確定性就越大。透過計算**資訊增益**，將較高性質的資料置放於相同的類別，已產生各個分支，本方法可以判斷多個類別。
2. C4.5:
   選擇**訊息增益率**來對ID3做改進
3. CART (Classification and Regression tree)
   和熵一樣越大代表不確定性就越大。透過吉尼指數計算吉尼不純度，並進行二元分類。支援分類(不連續項目)與迴歸(連續數據)。

- 吉尼指數的效果,比ID3好,但因計算繁瑣,容易產生過度擬合。此為Scikit 預設的演算法。

[Ref] 1. https://ithelp.ithome.com.tw/articles/10271143
2. https://ithelp.ithome.com.tw/articles/10204450

回歸模型的衡量標準：MSE. RMSE. MAE. MPE

- MSE（Mean Square Error）

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- MSE主要以平方來避免誤差正負的互相抵銷，但也因為平方的特性，所以當單一bias大的時候會有懲罰作用，也就是說MSE對於極值（outliers)會相對敏感。

- **RMSE（Root Mean Square Error）**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

- RMSE主要就是MSE拿去取根號，取根號的目的是讓他與y的單位變得一致，所以解釋起來會比較直觀！

- MAE（Mean Absolute Error）

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

- 抵銷正負誤差的方式，除了平方之外，還有取絕對值，MAE就是取絕對值來計算平均誤差，相對之下對極值比較不敏感，如果training data裡面極值很多，那可以考慮用MAE來當作指標。

- **MAPE（Mean Absolute Percentage Error）**

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

- 將誤差轉換為百分比（徹底擺脫單位），主要目的也是解釋起來比較直觀，要記得如果y含有0就不能使用MAPE!

資料來源：https://ithelp.ithome.com.tw/articles/10274551?sc=hot

Scikit-Learn 的決策樹模組(1/2)

1. 載入Scikit-Learn模組: DecisionTreeClassifier

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

2. 建立DecisionTreeClassifiern 物件:

```
決策樹變數= DecisionTreeClassifier(criterion=演算方式, max_depth=數值)
```

- criterion:
  - gini(預設): 使用基尼指數演算法
  - emtropy:使用ID3演算法
- max_depth: 此參數非必填, 是設定決策樹最大樹層。預設為: None，表示盡可能擴展決策樹的層數。

3.利用fit ()方法進行訓練:

```
決策樹變數. fit (訓練值資料, 訓練目標值)
```

Scikit-Learn 的決策樹模組(2/2)

4.訓練完後,可使用 predict ()方法對未知資料進行預測

```
預測變數 = 決策樹變數. predict (預測資料)
```

5.或使用score ()方法對未知資料進行預測, 並計算準確率

```
準確率變數 = 決策樹變數. score (預測資料, 預測目標值)
```

[Lab] 以決策樹建立鐵達尼號生存判斷

Part 5

隨機森林演算法

隨機森林(Random forest)

- **隨機森林**是一種監督型機器學習演算法，由於具備準確性、簡單性、靈活性，它是最常使用的演算法之一。它可用於分類和迴歸分析，加上其非線性特性，非常能夠適應各種資料和情況。
- 「隨機決策森林」一是由何天琴於 1995 年首次提出，她發明一個公式來使用隨機資料進行預測。之後在 2006 年，Leo Breiman 和 Adele Cutler 擴展了這個演算法，創造出今天所知的隨機森林。
- 之所以被稱為「森林」，是因為它是長著一片決策樹的森林，藉由將來自這些樹的資料合併在一起，來確保獲得最準確的預測。由於它可以在隨機特徵子集中找到最佳特徵，為模型添加隨機性，因此優點更多
- 使隨機森林表現良好，它們需要三個條件：

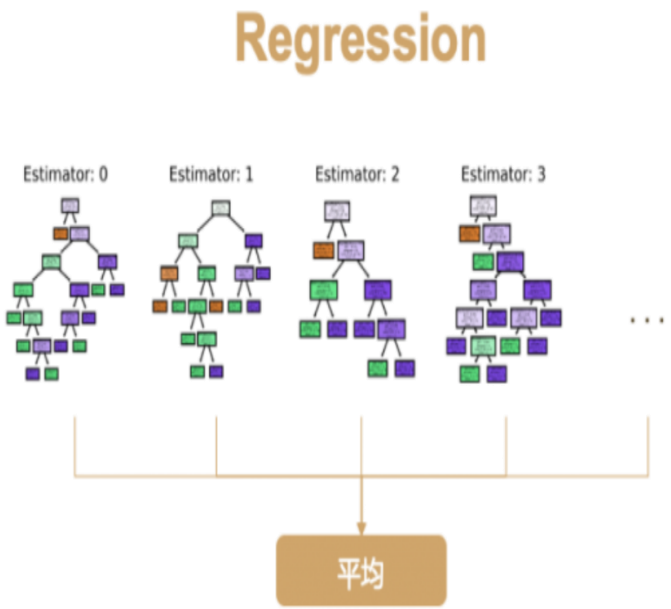
1.一個具有辨識度的訊號，使模型不僅是猜測而已
2.每棵決策樹做出的預測必須與其他樹保持較低相關性
3.特徵需具有一定程度的預測力：GI=GO

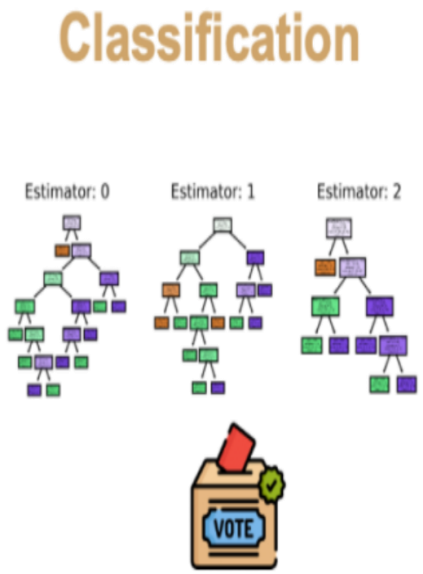
- [Ref] https://www.tibco.com/zh-hant/reference-center/what-is-a-random-forest

- 隨機森林的生成方法

1.從訓練集中抽取 n’ 筆資料出來
2.n’ 筆資料隨機挑選 k 個特徵做樣本
3.重複 m 次，產生 m 棵決策樹
4.分類: 多數投票機制進行預測、迴歸: 平均機制進行預測

※在 sklearn 中，最多隨機選取 log2N 個特徵





Scikit-Learn 的隨機森林模組(1/2)

1. 載入Scikit-Learn模組: RandomForestClassifier

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

2. 建立RandomForestClassifier物件:

```
隨機森林變數= RandomForestClassifier (criterion=演算方式,
                n_ estimators=數值, max_features=最大特徵,
                min_samples_split=數量, random_state=數值 )
```

- criterion:
  - gini(預設): 使用基尼指數演算法
  - entropy:使用ID3演算法
- n_ estimators: 非必填,設定隨機森林決策樹的數量,預設值=100。
- max _features : 此參數非必填，是設定決策樹最大特徵數。設定值有:
  - auto(預設): 使用全部特徵
  - sqrt: 使用特徵平方根
  - 浮點數: 使用特徵比例, Ex:0.7 代表使用特徵數的70%
- min_samples_split:非必填,設定決策樹的最小資料數量,預設值=2。
- random_state:非必填,若設定此參數則每次執行的決策樹都相同。

Scikit-Learn 的隨機森林模組(2/2)

3.利用fit ()方法進行訓練:

```
森林變數. fit (訓練值資料, 訓練目標值)
```

4.訓練完後,可使用 predict ()方法對未知資料進行預測

```
預測變數 = 森林變數. predict (預測資料)
```

5.或使用score ()方法對未知資料進行預測, 並計算準確率

```
準確率變數 = 森林變數. score (預測資料, 預測目標值)
```