書目：Python資料科學與人工智慧 應用實務

# 資料科學

## 資料採礦（Data Mining）

* 預測
* 識別
* 分類
* 最佳化

### 步驟：

1. 篩選 ：使用一些標準或準則來獲取或篩選資料
2. 前置處理 ：資料清理
3. 轉換 ：資料分割合併轉換，轉成可用和操作的資料
4. 資料採礦 ：資料分析，從資料找出樣式和規則
5. 直譯與評估 ：找出的樣式和規則是否具有參考價值的知識

## 資料科學

### 處理步驟

1. 詢問有興趣的問題
2. 取得資料
3. 探索資料
4. 模型資料
5. 溝通和視覺化結果

# 取得資料

## 資料擷取(網路爬蟲)

## 資料清理

# 探索資料

## 機率

P(A)=發生次數/重複試驗的次數

組合事件：P(A∩B) 事件A, B同時發生的機率

條件機率：當一件事情已經發生，在此事件上發生另一事件的機率

P(A|B)= P(A∩B) / P(B) = 事件A, B同時發生的機率 / 事件B發生的機率

例如：骰子出現偶數的機率，和出現4以下的機率

### 機率定理

#### 加法定理

P(A∪B) = P(A) + P(B) - P(A∩B)

例如：事件A是骰子出現點數4以下的事件，事件B是出現點數是偶數的條件

#### 乘法定理

P(A∩B) = P(A) \* P(B|A)

#### 排列組合

### 統計基礎

統計分為

1. 敘述統計：觀察樣本資料，取得特徵摘要與特性

例如：從去年整年週日飲料銷售量的樣本資料，取得資料特徵為「週日平均可賣出100杯飲料」

1. 推論統計：從資料分析趨勢，從部份資料推論全部(母體)

例如：針對數年週日營收，預測下個週日可賣出80~120杯飲料

#### 眾數、中位數、四分位數

|  |
| --- |
| import pandas as pd  df=pd.read\_csv("../input/titanic/train\_and\_test2.csv")  s=pd.Series([30, 1,5, 10, 30, 50, 30, 15, 40, 45, 30])  #眾數  print(df["Age"].mode())  print(s.mode())  #中位數  print(df["Age"].median())  print(s.median())  #四分位數  print(df["Age"].quantile(q=0.25))  print(s.quantile(q=0.25)) |
| 0 28.0  dtype: float64  0 30  dtype: int64  28.0  30.0 |

#### 算術平均數

|  |
| --- |
| print(df["Age"].mean)  print(s.mean) |
| <bound method NDFrame.\_add\_numeric\_operations.<locals>.mean of 0 22.0  1 38.0  2 26.0  3 35.0  4 35.0  ...  1304 28.0  1305 39.0  1306 38.5  1307 28.0  1308 28.0  Name: Age, Length: 1309, dtype: float64>  <bound method NDFrame.\_add\_numeric\_operations.<locals>.mean of 0 30  1 1  2 5  3 10  4 30  5 50  6 30  7 15  8 40  9 45  10 30  dtype: int64> |

#### 離散量數

* 全距=樣本最大值-最小值，表示資料分配中的最大值和最小值之間的距離

|  |
| --- |
| #全距  print(s.max()-s.min())  #四分位差  print(s.quantile(q**=**0.75)**-**s.quantile(q**=**0.25)) |

* 變異數(Variance)

使用全部樣本表示資料離散程度，每個值-平均數=偏差(Deviations)

偏差=

偏差平方和=(2+(2+…+(2

變異數=偏差平方和 / 資料量n

df["Age"].var()

* 標準差(Standard Deviation)

標準差=變異數平方根

panda std()函式計算DataFrame物件指定欄位, 或 Series物件的標準差

df["Age"].std()

df["Age"].describe() 包含資料長度、平均值、標準差、最小值、中位數、最大值

### 機率分配

隨機變數種類：離散型變數(1,2,3..)、連續型變數(溫度)

機率分配方法：常態分配、二項分配、波瓦松分配等

機率分配用途：將複雜現象用簡單數學模型表示，如個別事件發生機率、重複試驗中事件發生次數、事件多久會發生

#### 離散型隨機變數

離散型隨機變數屬性：期望值μ、變異數

μ= x1p1+x2p2+…+xnpn x隨機變數, p機率, 隨機變數\*機率的總和

σ2=(2 \*p +(2\*p +…+(2\*p

例：骰子機率為1/6, 重複多次試驗後(大數法則)，期望值會逐漸接近算術平均值

1\*1/6+2\*1/6+3\*1/6+4\*1/6+5\*1/6+6\*1/6=3.5

(1+2+3+4+5+6)/6=3.5

變異數σ2 = (1-3.5)2\*1/6 + (2-3.5)2\*1/6+…+(6-3.5)2\*1/6=2.9

標準差σ= 1.7

#### 連續型隨機變數

連續型隨機變數是連續無法計數的數值，要使用機率密度函數（Probability Density Function, PDF）來取得機率

使用積分來計算面積值

期望值μ=[隨機變數X與機率密度函數乘積] 的積分

變異數σ2 ＝[(X值 – 期望值μ) 與機率密度函式乘積]的積分

#### 累積分配函數

類似統計累積次數表，將次數和相對次數分別累加

|  |  |  |
| --- | --- | --- |
| 類型 | 次數 | 累積次數 |
| 1~1000 | 22 | 22 |
| 1001~2000 | 45 | 67 |
| 2001~3000 | 57 | 124 |

* 離散型隨機變數：所有x之前的機率總和
* 連續型隨機變數：所有x之前機率密度函數的積分

用途：找出特定值之上之下或兩個值之間的機率，如找出小狗體重大於3公斤的機率，在2~3公斤之間的機率

#### 二項分配

是離散型隨機變數的機率分配

二項隨機變數：在重複發生單一事件中，計算出成功的次數

需符合條件：

1. 樣本尺寸固定，固定的試驗次數
2. 對每一次試驗，成功情況一定會出現或沒有出現
3. 每個事件的機率必須相等
4. 每次試驗都是獨立事件，試驗之間沒有關聯

二項分配重要特性：

1. 固定試驗次數n
2. 每次試驗的成功機率p

二項隨機變數的機率質量函數(PMF)

使用scipy套件的stats統計模組：(擲骰子)

|  |
| --- |
| from scipy import stats  n = 5  p = 1/6  for k in range(n+1):  v = stats.binom.pmf(k, n, p)  print(k, v) |
| import pandas as pd  import matplotlib.pyplot as plt  from scipy import stats  fair\_dice\_rolls = stats.binom.rvs(n=5, k=1/6, size=10000)  print(fair\_dice\_rolls)  df = pd.DataFrame(fair\_dice\_rolls)  df.hist(rnage=(-0.5, 5.5), bins=6) |

norm常態分配

binom 二項分配

rvs 隨機資料, n每次試驗次數, p成功機率, size總共試驗次數

#### 常態分配

高斯分配，常見連續型隨機變數的機率分配

配合平均數和標準差，可進行精確的描述和推論

特性：

1. 以平均值為中心，左右對稱的鐘形曲線(對稱不一定是常態分配，常態分配一定是對稱)
2. 常態曲線的眾數、中位數和平均數是三合一
3. 常態曲線的形狀完全是以平均數和標準差來決定

用途：發現真實世界的現象，IQ測驗、身高、體重、收入支出等。大部分檢定都假設資料分配是常態分配

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  def normal\_pdf(x, mu, sigma):  pi =3.1415926  e = 2.718281  f = (1./np.sqrt(2\*pi\*sigma\*\*2))\*e\*\*(-(x-mu)\*\*2/(2.\*sigma\*\*2)  return f  ax = np.linspace(-5, 5, 100)  ay = [normal\_pdf(x, 0, 1) for x in ax]  plt.plot(ax, ay)  plt.show() |
| from scipy import stats  imort matplotlib.pyplot as plt  import numpy as np  x = [ x/10.0 for x in range(-50, 60)]  plt.plot(x, stats.norm.pdf(x, 0, 1), 'r-', lw=1, alpha=0.6, label='mu=0', sigma=1')  plt.legend()  plt.title("Various Normal PDF")  plt.show() |

## 估計與檢定

#### 抽樣與抽樣分配

推論統計可從部份資料的樣本推論出全部資料的母體

* 估計：從樣本資料推論樣本來源的母體特徵(平均數和標準差)的過程
* 假設檢定：先針對母體提出假設，然後透過分析比較樣本來驗證提出的假設是否有效

## ★探索性資料分析(EDA, Exploratory Data Analysis)

### 找出資料關聯性

使用散佈圖 – 找出正相關、負相關、無相關：觀察特徵分佈狀況，重要程度越高，群聚效果越顯著

使用共變異數 – 測量2個隨機變數之間的關係，但無法辨識強弱

* 負相關：共變異數<0
* 正相關：共變異數>0
* 無相關：共變異數=0

使用相關係數 – 計算2個變數的線性相關性有多強（-1~1）

* 相關性(Correlation): -1 ~ 1, 測量2個變數走勢是如何相關和其強度, 1表示2個變數都增加, -1表1個變數增加, 1個減少
* 因果關係(Causation): 1個變數可決定另一個變數的值

(補充)

* **直方圖**  
  對數據分布情況的圖形表示，是一種二維統計圖表
* **核密度估計**  
  圖分爲兩部分，分別有對角線部分和非對角線部分。在對角線部分是以核密度估計圖（Kernel Density Estimation）的方式呈現，也就是用來看某一個特徵的分佈情況，x軸對應著該特徵的數值，y軸對應著該特徵的密度也就是特徵出現的頻率。在非對角線的部分為兩個特徵之間分佈的關聯散點圖。將任意兩個特徵進行配對，以其中一個爲橫座標，另一個爲縱座標，將所有的數據點繪製在圖上，用來衡量兩個變量的關聯程度。
* **箱形圖**  
  分析每個特徵的分布狀況以及是否有離群值。我們利用箱形圖來表示四分位數來觀察數據分散情況。箱形的兩端為第一個四分位數涵蓋25%之資料(Q1)與第三個四分位數涵蓋75%之資料(Q3)，而箱形圖的中間線為中位數顯示涵蓋前50%資料之位置。箱形上虛線的端點為極大值，箱型下虛線的點為極小值。

### 特徵縮放與標準化

資料單位不同，標準化在同一標準下進行比較

#### 資料標準化（Standardization）

Z分數：可以位移資料分配的平均值是0, 標準差是1

1. 使用公式執行資料標準化
2. 使用Scikit-learn preprocessing執行資料標準化

ex. FB朋友追蹤數和快樂程度的調查資料

#### 最小最大值縮放（Min-max Scaling）

常見特徵縮放方法，又稱正規化(Normalization)，將數值資料轉換成0~1區間，

1. 使用線性回歸法
2. KNN演算法

### 資料清理與轉換

處理遺漏值

* 刪除遺漏值：如果資料量夠大，直接刪除
* 補值：補成固定值、平均值、中位數、亂數值

處理重複資料

處理分類資料

* 使用對應表進行分類資料轉換
* 使用Scikit-learn套件LabelEncoder

### 資料前處理與探索性資料分析(EDA)

* 資料前處理
* 探索性資料分析
  + 是否是有組織的資料
  + 資料的每一列代表什麼
  + 資料的每一欄代表什麼
  + 是否有遺漏值
  + 是否需要執行欄位資料轉換
  + 資料描述是什麼？資料是如何分佈？
  + 資料之間是否存在關係

### ★實做案例：Titanic Dataset

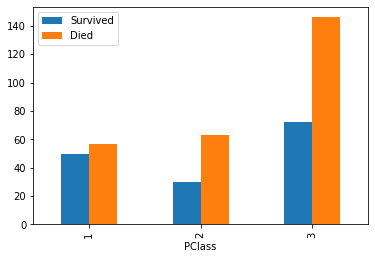
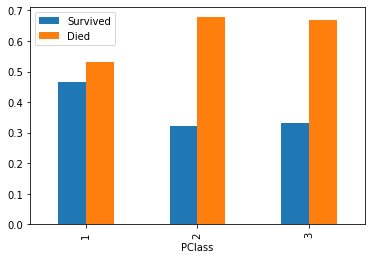
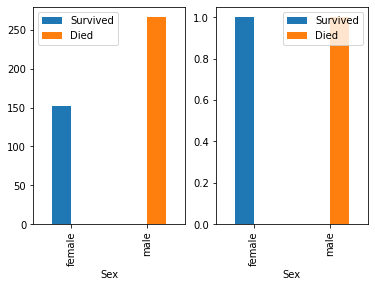
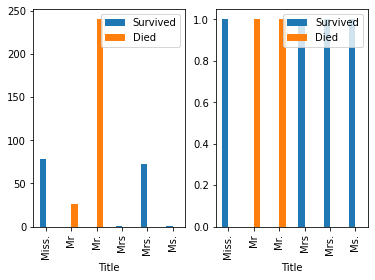
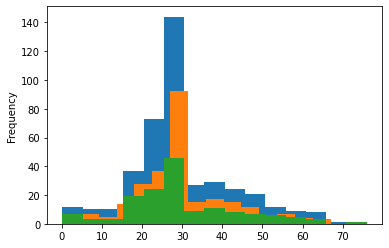
|  |
| --- |
| import numpy as np # linear algebra  import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)  titanic=pd.read\_csv("../input/titantic/train.csv")  #資料描述 Data Describe  print(titanic.shape)  print(titanic.head())        #顯示前五筆  print(titanic.describe())     #檢視資料量、平均值、標準差、最大小值  print(titanic.info())         #檢視是否有遺漏值 |
| /kaggle/input/test-file/tested.csv  /kaggle/input/titantic/train.csv  (418, 12)  PassengerId Survived Pclass \  0 892 0 3  1 893 1 3  2 894 0 2  3 895 0 3  4 896 1 3  Name Sex Age SibSp Parch \  0 Kelly, Mr. James male 34.5 0 0  1 Wilkes, Mrs. James (Ellen Needs) female 47.0 1 0  2 Myles, Mr. Thomas Francis male 62.0 0 0  3 Wirz, Mr. Albert male 27.0 0 0  4 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0 1 1  Ticket Fare Cabin Embarked  0 330911 7.8292 NaN Q  1 363272 7.0000 NaN S  2 240276 9.6875 NaN Q  3 315154 8.6625 NaN S  4 3101298 12.2875 NaN S  PassengerId Survived Pclass Age SibSp \  count 418.000000 418.000000 418.000000 332.000000 418.000000  mean 1100.500000 0.363636 2.265550 30.272590 0.447368  std 120.810458 0.481622 0.841838 14.181209 0.896760  min 892.000000 0.000000 1.000000 0.170000 0.000000  25% 996.250000 0.000000 1.000000 21.000000 0.000000  50% 1100.500000 0.000000 3.000000 27.000000 0.000000  75% 1204.750000 1.000000 3.000000 39.000000 1.000000  max 1309.000000 1.000000 3.000000 76.000000 8.000000  Parch Fare  count 418.000000 417.000000  mean 0.392344 35.627188  std 0.981429 55.907576  min 0.000000 0.000000  25% 0.000000 7.895800  50% 0.000000 14.454200  75% 0.000000 31.500000  max 9.000000 512.329200  <class 'pandas.core.frame.DataFrame'>  RangeIndex: 418 entries, 0 to 417  Data columns (total 12 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 PassengerId 418 non-null int64  1 Survived 418 non-null int64  2 Pclass 418 non-null int64  3 Name 418 non-null object  4 Sex 418 non-null object  5 Age 332 non-null float64  6 SibSp 418 non-null int64  7 Parch 418 non-null int64  8 Ticket 418 non-null object  9 Fare 417 non-null float64  10 Cabin 91 non-null object  11 Embarked 418 non-null object  dtypes: float64(2), int64(5), object(5)  memory usage: 39.3+ KB  None |

#### #資料前處理 Data preprocess

|  |  |
| --- | --- |
| print(np.unique(titanic["PassengerId"].values).size) #檢查是否為unique  titanic.set\_index(["PassengerId"], inplace=True) #指定欄位為索引  print(titanic.head)  # titanic=titanic.drop(labels=['PassengerId'], axis=1) #移除欄位  titanic["SexCode"] = np.where(titanic["Sex"]=="female",1,0) #新增SexCode欄位  print(titanic.head())  # 深度學習時，神經網路只能處理數值資料，需轉換非數值特徵  # PCass欄位轉換為數值資料  class\_mapping = {"1st": 1,                   "2nd": 2,                   "3rd": 3}  titanic["PClass"] = titanic["PClass"].map(class\_mapping) | 418  <bound method NDFrame.head of Survived Pclass Name \  PassengerId  892 0 3 Kelly, Mr. James  893 1 3 Wilkes, Mrs. James (Ellen Needs)  894 0 2 Myles, Mr. Thomas Francis  895 0 3 Wirz, Mr. Albert  896 1 3 Hirvonen, Mrs. Alexander (Helga E Lindqvist)  ... ... ... ...  1305 0 3 Spector, Mr. Woolf  1306 1 1 Oliva y Ocana, Dona. Fermina  1307 0 3 Saether, Mr. Simon Sivertsen  1308 0 3 Ware, Mr. Frederick  1309 0 3 Peter, Master. Michael J  Sex Age SibSp Parch Ticket Fare Cabin \  PassengerId  892 male 34.5 0 0 330911 7.8292 NaN  893 female 47.0 1 0 363272 7.0000 NaN  894 male 62.0 0 0 240276 9.6875 NaN  895 male 27.0 0 0 315154 8.6625 NaN  896 female 22.0 1 1 3101298 12.2875 NaN  ... ... ... ... ... ... ... ...  1305 male NaN 0 0 A.5. 3236 8.0500 NaN  1306 female 39.0 0 0 PC 17758 108.9000 C105  1307 male 38.5 0 0 SOTON/O.Q. 3101262 7.2500 NaN  1308 male NaN 0 0 359309 8.0500 NaN  1309 male NaN 1 1 2668 22.3583 NaN  Embarked  PassengerId  892 Q  893 S  894 Q  895 S  896 S  ... ...  1305 S  1306 C  1307 S  1308 S  1309 C  [418 rows x 11 columns]>  Survived Pclass Name \  PassengerId  892 0 3 Kelly, Mr. James  893 1 3 Wilkes, Mrs. James (Ellen Needs)  894 0 2 Myles, Mr. Thomas Francis  895 0 3 Wirz, Mr. Albert  896 1 3 Hirvonen, Mrs. Alexander (Helga E Lindqvist)  Sex Age SibSp Parch Ticket Fare Cabin Embarked \  PassengerId  892 male 34.5 0 0 330911 7.8292 NaN Q  893 female 47.0 1 0 363272 7.0000 NaN S  894 male 62.0 0 0 240276 9.6875 NaN Q  895 male 27.0 0 0 315154 8.6625 NaN S  896 female 22.0 1 1 3101298 12.2875 NaN S  SexCode  PassengerId  892 0  893 1  894 0  895 0  896 1 |
| # 檢查Age欄位的遺漏值有多少  print(titanic.isnull().sum())  print(sum(titanic["Age"].isnull()))  # 補值成平均值  avg\_age = titanic["Age"].mean()  titanic["Age"].fillna(avg\_age, inplace=True)  print(sum(titanic["Age"].isnull()))  # 顯示性別人數和計算平均年齡  print("性別人數:")  print(titanic["Sex"].groupby(titanic["Sex"]).size())  print(titanic.groupby("Sex")["Age"].mean())  # 處理姓名欄位  import re  patt = re.compile(r"\,\s(\S+\s)")  titles = []  for index, row in titanic.iterrows():      m = re.search(patt, row["Name"])      if m is None:          title = "Mrs" if row["SexCode"] == 1 else "Mr"      else:          title = m.group(0)          title = re.sub(r",", "", title).strip()          if title[0] != "M":              title = "Mrs" if row["SexCode"] == 1 else "Mr"          else:             if title[0] == "M" and title[1] == "a":              title = "Mrs" if row["SexCode"] == 1 else "Mr"      titles.append(title)  titanic["Title"] = titles  print("Title類別:")  print(np.unique(titles).shape[0], np.unique(titles))  # 修正類別的錯誤  titanic["Title"] = titanic["Title"].replace("Mlle","Miss")  titanic["Title"] = titanic["Title"].replace("Ms","Miss")  titanic.to\_csv("titanic\_pre.csv", encoding="utf8")  print("Title人數:")  print(titanic["Title"].groupby(titanic["Title"]).size())  print("平均生存率:")  print(titanic[["Title","Survived"]].groupby(titanic["Title"]).mean()) | # 檢查Age欄位的遺漏值有多少  Survived 0  Pclass 0  Name 0  Sex 0  Age 86  SibSp 0  Parch 0  Ticket 0  Fare 1  Cabin 327  Embarked 0  SexCode 0  PClass 0  dtype: int64  86  0 # 補值成平均值，再次檢查無null值了  性別人數:  Sex  female 152  male 266  Name: Sex, dtype: int64  Sex  female 30.272400  male 30.272699  Name: Age, dtype: float64  Title類別:  6 ['Miss.' 'Mr' 'Mr.' 'Mrs' 'Mrs.' 'Ms.']  Title人數:  Title  Miss. 78  Mr 26  Mr. 240  Mrs 1  Mrs. 72  Ms. 1  Name: Title, dtype: int64  平均生存率:  Survived  Title  Miss. 1.0  Mr 0.0  Mr. 0.0  Mrs 1.0  Mrs. 1.0  Ms. 1.0 |

#### #探索性資料分析

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  titanic = pd.read\_csv("titanic\_pre.csv")  titanic["Died"] = np.where(titanic["Survived"]==0, 1, 0)  print(titanic.head())  titanic.head().to\_html("Ch13\_5c\_01.html")  # 繪出直方圖的年齡分佈, 生存或死亡  titanic["Age"].plot(kind="hist", bins=15)  df = titanic[titanic.Survived == 0]  df["Age"].plot(kind="hist", bins=15)  df = titanic[titanic.Survived == 1]  df["Age"].plot(kind="hist", bins=15)  # 分類顯示Title欄位的生存和死亡數  fig, axes = plt.subplots(nrows=1, ncols=2)  df = titanic[["Survived","Died"]].groupby(titanic["Title"]).sum()  df.plot(kind="bar", ax=axes[0])  df = titanic[["Survived","Died"]].groupby(titanic["Title"]).mean()  df.plot(kind="bar", ax=axes[1])  # 分類顯示Sex欄位的生存和死亡數  fig, axes = plt.subplots(nrows=1, ncols=2)  df = titanic[["Survived","Died"]].groupby(titanic["Sex"]).sum()  df.plot(kind="bar", ax=axes[0])  df = titanic[["Survived","Died"]].groupby(titanic["Sex"]).mean()  df.plot(kind="bar", ax=axes[1])  # 分類顯示PClass欄位的生存和死亡數  df = titanic[['Survived',"Died"]].groupby(titanic["PClass"]).sum()  df.plot(kind="bar")  df = titanic[['Survived',"Died"]].groupby(titanic["PClass"]).mean()  df.plot(kind="bar")  # 計算相關係數  df = titanic.drop("PassengerId", axis=1)  df = df.drop("Died", axis=1)  df = df.drop("Title", axis=1)  print(df.corr())  df.to\_csv("titanic\_train.csv", encoding="utf8")  df.corr().to\_html("Ch13\_5c\_02.html") | PassengerId Survived Pclass \  0 892 0 3  1 893 1 3  2 894 0 2  3 895 0 3  4 896 1 3  Name Sex Age SibSp Parch \  0 Kelly, Mr. James male 34.5 0 0  1 Wilkes, Mrs. James (Ellen Needs) female 47.0 1 0  2 Myles, Mr. Thomas Francis male 62.0 0 0  3 Wirz, Mr. Albert male 27.0 0 0  4 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0 1 1  Ticket Fare Cabin Embarked SexCode PClass Title Died  0 330911 7.8292 NaN Q 0 3 Mr. 1  1 363272 7.0000 NaN S 1 3 Mrs. 0  2 240276 9.6875 NaN Q 0 2 Mr. 1  3 315154 8.6625 NaN S 0 3 Mr. 1  4 3101298 12.2875 NaN S 1 3 Mrs. 0  Survived Pclass Age SibSp Parch Fare \  Survived 1.000000 -0.108615 -0.000011 0.099943 0.159120 0.191514  Pclass -0.108615 1.000000 -0.440782 0.001087 0.018721 -0.577147  Age -0.000011 -0.440782 1.000000 -0.079535 -0.045073 0.329071  SibSp 0.099943 0.001087 -0.079535 1.000000 0.306895 0.171539  Parch 0.159120 0.018721 -0.045073 0.306895 1.000000 0.230046  Fare 0.191514 -0.577147 0.329071 0.171539 0.230046 1.000000  SexCode 1.000000 -0.108615 -0.000011 0.099943 0.159120 0.191514  PClass -0.108615 1.000000 -0.440782 0.001087 0.018721 -0.577147  SexCode PClass  Survived 1.000000 -0.108615  Pclass -0.108615 1.000000  Age -0.000011 -0.440782  SibSp 0.099943 0.001087  Parch 0.159120 0.018721  Fare 0.191514 -0.577147  SexCode 1.000000 -0.108615  PClass -0.108615 1.000000 |



* SecCode和Survived相關係數0.5(高度相關)
* PClass -0.36(中等相關)
* Age -0.04(低度相關)

#### [機器學習專案] Kaggle競賽-鐵達尼號生存預測(Top 3%)

<https://yulongtsai.medium.com/https-medium-com-yulongtsai-titanic-top3-8e64741cc11f>

# 預測資料

## AI概論

參考書目：Python資料科學與人工智慧 應用實務 CH14

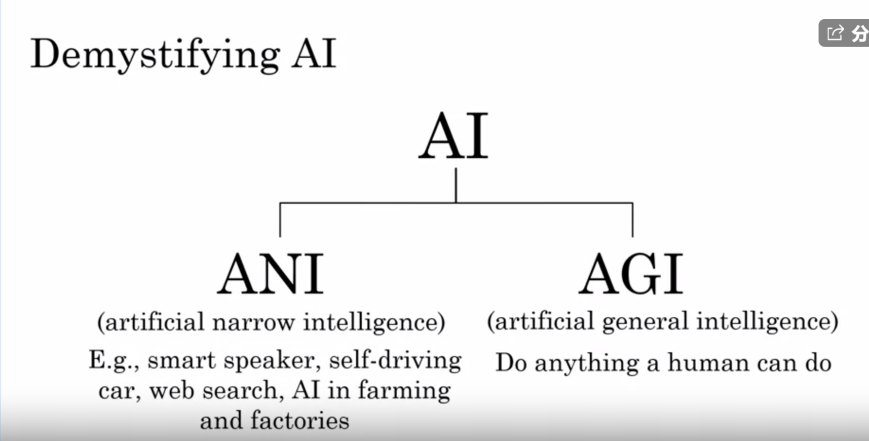
人工智慧>機器學習>深度學習

彙總"[CS\_AI for everyone.docx](file:///D:\doc_office_IT_training_AI\course\TA\CS_AI%20for%20everyone.docx)"

[十三分鐘略懂 AI 技術：機器學習、深度學習技術原理及延伸應用](https://www.youtube.com/watch?v=UGdG4WpluJ8&ab_channel=jasonmel)

[Elon Musk 的機器人好笨拙？ 究竟特斯拉 Optimus 厲害在哪裡？AI Day 大揭密](https://www.youtube.com/watch?v=_HjGppyK0Po&ab_channel=PanSci%E6%B3%9B%E7%A7%91%E5%AD%B8)

### 人工智慧



#### 應用領域

* 手寫辨識
* 語音辨識
* 電腦視覺 – 自動駕駛, 瑕疵檢測
* 專家系統
* 自然語言處理
* 電腦遊戲
* 智慧機器人

[最美機器人曝光！【AI化時代-你我的幫手】特別報導](https://www.youtube.com/watch?v=_nSUwIc89Mw&ab_channel=TVBS%E5%84%AA%E9%81%B8%E9%A0%BB%E9%81%93)

* (對話系統、推薦系統、決策系統)

**補充：人工智慧(AI)於車載方面應用**

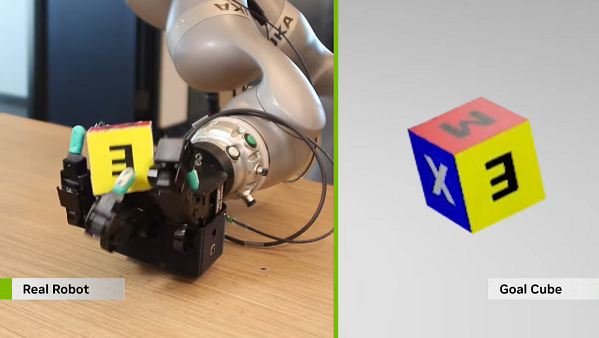
1. 先進駕駛輔助系統(ADAS)衍伸出的Level 5以上完全自駕車
2. 由智慧化中控台、儀表板、車載娛樂系統(IVI)所延伸出的智慧座艙系統
3. 為因應智慧城市、智慧交通需求所衍生的智慧化 / 自動化商用及大眾運輸車隊管理系統

* 3大感測基礎：光達、雷達、攝影機

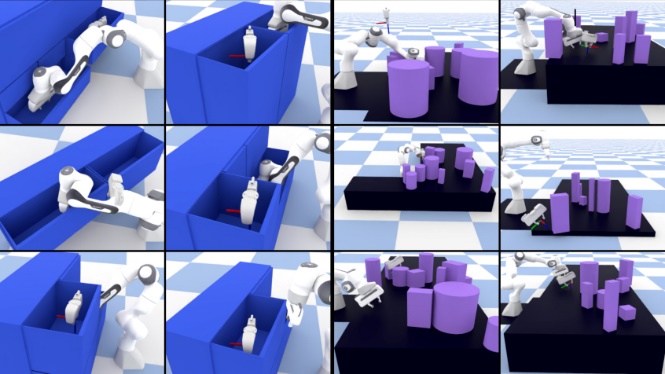
**補充：2022發展**

* 傳統業者為因應數位金融的浪潮，積極導入AI應用比如Chatbot、生物辨識，或是運用AI評估貸款資格等。
* 金融市場中也出現眾多金融科技新創，其中不乏許多運用AI技術開發不同解決方案，比方說AI身分驗證、信用風險評估等
* Martech行銷科技，除早期使用者遊戲、電商之外，現正在迅速導入的中間主流產業，比如製造、零售、金融等
* 醫療產業，台灣一些醫院已開始在部分科技實驗或進行輔助判決、AI預判分析，整體來說，仍在觀望與初步導入階段。新藥開發方面，蛋白質摺疊上幾乎已全面AI化。蛋白質折疊在開放新藥過程中的主要難題，因為組合有無限多種，要組合成可用的東西需要極大的運算量，運用AI做大量的預測與運算。
* 國防領域。2021年美國國防部將AI納入NDAA的法律當中，過去AI應用在軍事上是禁止的，如今美國將軍事AI納入法律，就能夠透過有道德且負責任的方式取得AI技術，等同於授權軍方在法律保護下使用AI
* 元宇宙。現今的機器翻譯(Machine Translating；MT)系統快速進步，但仍極度依賴從大量文本資料中學習，導致對低資料來源語言，也就是缺少訓練資料的語言，以及沒有標準化書寫系統的語言通常成效不佳。Meta日前宣布一項打造語言集MT工具的長期計畫，將涵蓋大多數的語言，並會分為2個新專案。
  1. 第一個專案「任何語言皆重要(No Language Left Behind)」，Meta正在建立新的進階人工智慧模型，能夠從實例較少的語言中學習、訓練，將會用以達成數百種語言的專業及翻譯。
  2. 第二個專案「通用語音翻譯(Universal Speech Translator)」，Meta正在設計新的模式，將一種語言的語音內容即時翻譯成另一種語言，支援沒有標準書寫系統的語言，以及有書面又有口語系統的語言。
* 視訊服務：娛樂用換臉與時尚業換衣化妝技術，也可用在視訊會議改善形象
* AIOT：智慧物聯

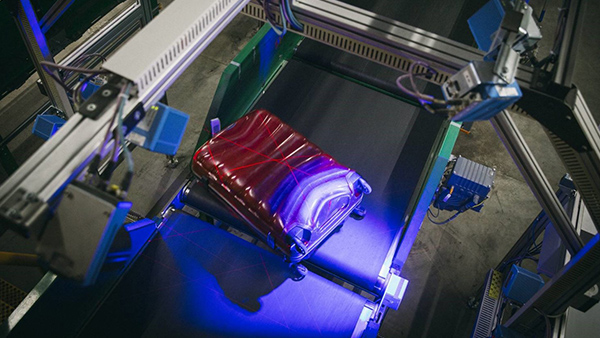
[運用模擬技術來教導實際的機器人手學習靈巧精細的動作](https://developer.nvidia.com/blog/reinforcing-the-value-of-simulation-by-teaching-dexterity-to-a-real-robot-hand/?ncid=em-news-915636-vt48#cid=em07_em-news_en-us)

[](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgo.nvidianews.com%2FMTU2LU9GTi03NDIAAAGJOGIa8w0qNtSXrQ4WX1qnPm2g2yQsIsHSh2Fxyt0SaUzxIuGsB5trYrYsg__BQZ28uzCCZ7U%3D&data=05%7C01%7CJohnny_kuo%40pegatroncorp.com%7Cd19eeb1007cb468837a408daf2aea18c%7C5ab9aaef2fe64d66a5f2129a6671846b%7C1%7C0%7C638089129117207839%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=nw1iQuJ7M6H1gXWJXow6vAt4BlYC8IZiAZRexgyhqtw%3D&reserved=0)

[使用運動策略網路進行避撞的機器人移動](https://developer.nvidia.com/blog/improving-robot-motion-generation-with-motion-policy-networks/?ncid=em-news-780022-vt48#cid=em07_em-news_en-us)

[](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgo.nvidianews.com%2FMTU2LU9GTi03NDIAAAGJOGIa86PUIgIyNO0RSa33YCfSAKOyQ5QIKvcag--u2wj8BNDXrrN7RHoZvqjLSSMRLD1pPqM%3D&data=05%7C01%7CJohnny_kuo%40pegatroncorp.com%7Cd19eeb1007cb468837a408daf2aea18c%7C5ab9aaef2fe64d66a5f2129a6671846b%7C1%7C0%7C638089129117207839%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=o6nvXEXwqnT%2Fv0B1yigQIOkQlXlV6xXuuhezL%2BDgBVI%3D&reserved=0)

[利用視覺人工智慧技術來減少機場行李延誤](https://resources.nvidia.com/en-us-metropolis-software-success-stories/trifork?ncid=em-news-656035-vt48#page=1)

[](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgo.nvidianews.com%2FMTU2LU9GTi03NDIAAAGJOGIa8zsD7KV0keUBBUEtCIC_dBDdYopyPq7TU9N7eUKUxywHVtntqN-J0b9Wwme0Owt2SvA%3D&data=05%7C01%7CJohnny_kuo%40pegatroncorp.com%7Cd19eeb1007cb468837a408daf2aea18c%7C5ab9aaef2fe64d66a5f2129a6671846b%7C1%7C0%7C638089129117364093%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=KdPLAL4ygIN5ozmlCAXfIanhn0BiEZ3Ms3GFVLial5w%3D&reserved=0)

[使用可攜式掃描器快速診斷腦中風](https://blogs.nvidia.com/blog/2022/12/05/emvision-portable-brain-scanner/?ncid=em-news-437733-vt48#cid=em07_em-news_en-us)

[](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgo.nvidianews.com%2FMTU2LU9GTi03NDIAAAGJOGIa8zkDS3ZvA6YIbkTVxTrHIlyzt8ATk9D6AZKV-h_X2kH4nS12nIokOtVwVjtqSdvkvDU%3D&data=05%7C01%7CJohnny_kuo%40pegatroncorp.com%7Cd19eeb1007cb468837a408daf2aea18c%7C5ab9aaef2fe64d66a5f2129a6671846b%7C1%7C0%7C638089129117364093%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=g0%2FWSXlHycrg%2BNrrHNSc%2FTTMH4RmEH1TrnWfWEepgxo%3D&reserved=0)

[「可無人駕駛」的智慧牽引機](https://blogs.nvidia.com/blog/2022/12/01/mondavi-monarch-smart-electric-jetson-tractor/?ncid=em-news-830358-vt48#cid=em07_em-news_en-us)

[](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgo.nvidianews.com%2FMTU2LU9GTi03NDIAAAGJOGIa8-Qii4hReWv3IHGSNTR2gk5MTN5K0smyXAcqCjYVR4yaYWmZfAlyxxBT2DBJXbNxD8E%3D&data=05%7C01%7CJohnny_kuo%40pegatroncorp.com%7Cd19eeb1007cb468837a408daf2aea18c%7C5ab9aaef2fe64d66a5f2129a6671846b%7C1%7C0%7C638089129117364093%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=8%2F1Z8wdblCXhZCQHyRLhd3vwE7gIWkt75GrScTOg00Y%3D&reserved=0)

#### 研究領域

* 機器學習和樣式識別
* 邏輯基礎AI (Logic-based)
* 搜尋(Search)
* 知識表示法(Knowledge Representation, KR)
* AI規劃(Planning)
* 啟發法(Heuristics)
* 基因程式設計(Genetic Programming, GP)

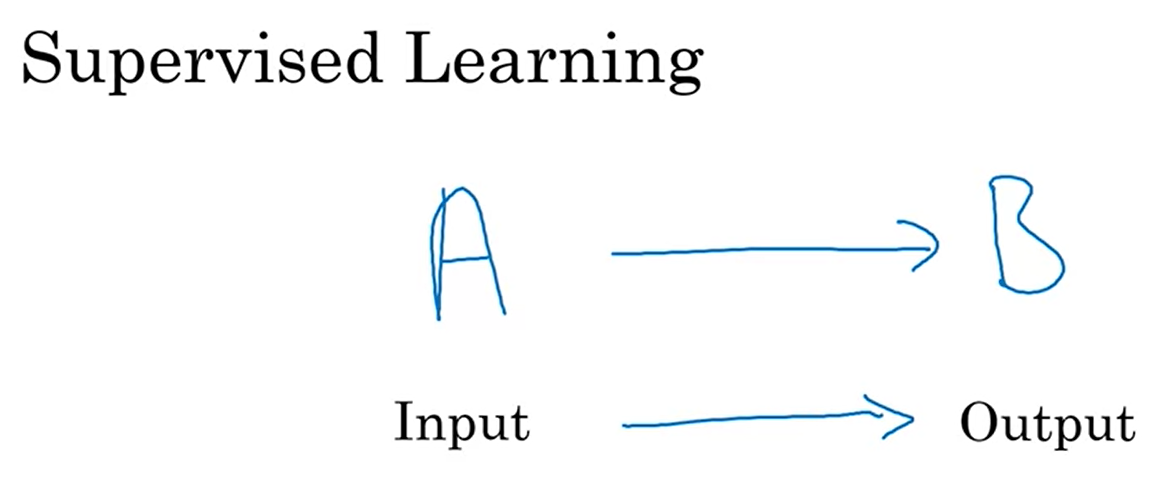
### 機器學習

應用統計學習技術來自動找出資料中隱藏的規則和關聯性

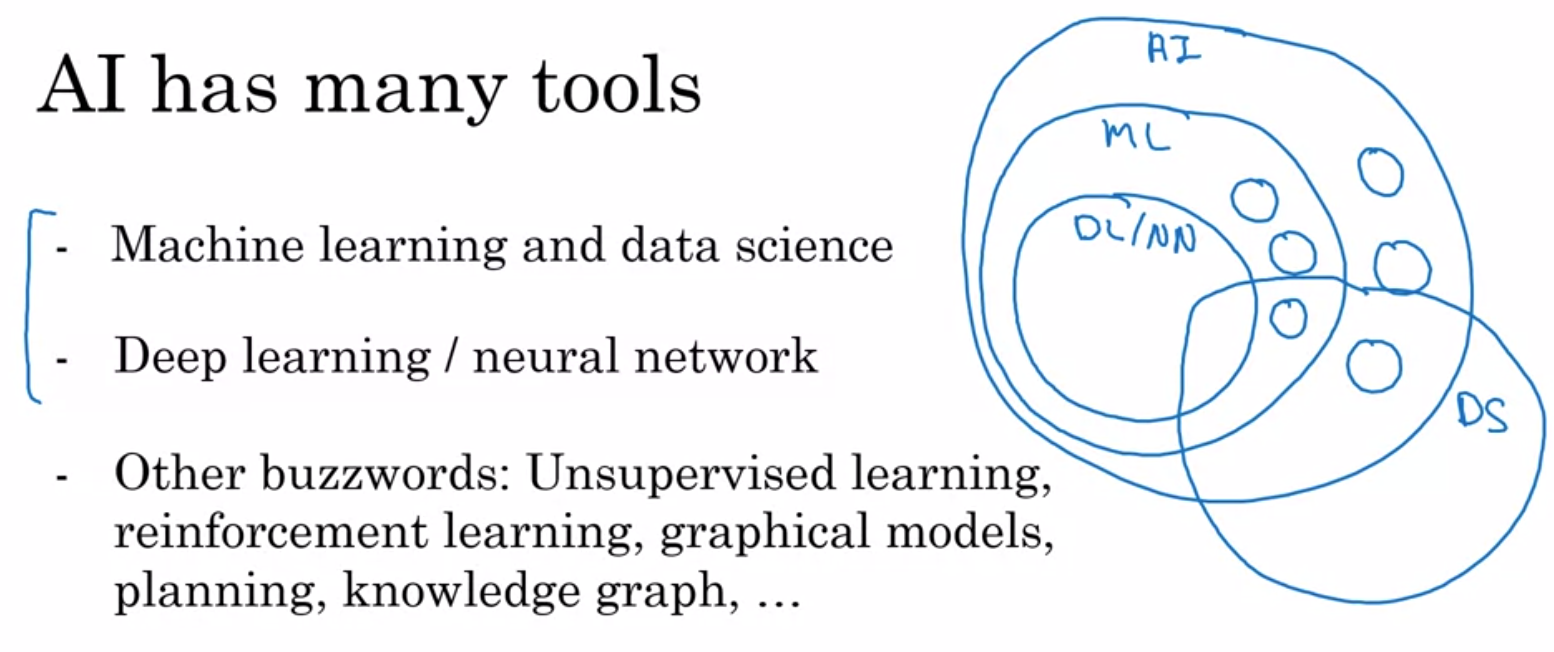
定義：從過往資料和經驗中自我學習並找出其運行的規則，以達到人工智慧的方法。

🡺機器學習就是利用歷史資料(Data)找出一個函數 f(X)=y

機器學習是一種人工智慧 ,學習 a到b 的方法或 輸入到輸出。這就是所謂的監督學習



* 機器學習：監督式學習 (A to B)
* 如何獲取資料（手動標記、觀察使用者行為或其他類型的行為、下載從網站上或是夥伴那裏）
* 資料誤用 （資料收集後才建立AI團隊, 濫用資料, 凌亂資料）
* 機器學習專案通常會產生一個運行的軟體,從給定的a輸出b,
* 資料科學是從資料中提取知識和見解的科學,輸出是一套見解,可以幫助您做出商業決策
* 神經網路=深度學習
* 神經網路是一個非常有效的技術學習a到b或輸入輸出映射



|  |  |
| --- | --- |
| Workflow of ML project   * 收集資料 * 訓練模型 * 部署模型 | Workflow of DS project   * 收集資料 * 分析資料 * 提出假設和行動 * 繼續拿回資料 並定期重新分析資料 |

AI團隊將資料分組到兩個主要資料集:訓練集和測試集

* 訓練集:利用訓練集 機器學習演算法將做的是學習計算出一些映射從 A 到 B,
* 測試集:學習演算法性能是給測試集中的圖像 到AI軟體中並查看輸出的內容

AI軟體可能無法實現 100% 準確的一些原因

* 有局限性
* 資料不足
* 資料混亂: 資料被錯誤地標記
* 資料不明確

#### 解決的問題

* **分類**  
  客戶是否續約、圖片是貓是狗、回饋或打折哪種促銷方法能提昇業績、哪種動物哪種植物、雷達訊號來自哪種飛機、錄音說話的是誰
* **異常值判斷**  
  偵測信用卡盜刷、網路訊息是否正常、這些消費和以前消費行為是否落差很大、管路壓力是否異常
* **預測性分析（回歸）**  
  下星期四氣溫幾度、台北第二季銷售量多少、下週FB會新增多少追蹤者、下週日可賣出多少產品
* **分群**  
  哪些消費者對水果有相似喜好、哪些觀眾喜歡同類型電影、哪些型號手機有相似故障、部落格訪客可分成哪些不同類別群組
* **協助決策**  
  網頁廣告至於哪個位置才容易點選、看到黃燈應保持速度或煞車或加速通過、溫度調低或調高、圍棋決定下一步位置

#### 種類

* 監督式學習(Supervised learning) 🡺基於經驗，來自過往數據，需要訓練

分類(Classification)：預測不連續資料，演算法：決策樹、K鄰近、CART、樸素貝葉斯、CART

迴歸(Regression)：預測連續資料，線性回歸、SVR、Logistic回歸、

* 非監督式學習(Unsupervised learning)：不需要label 🡺不依賴經驗，無須訓練

關聯：找出各種現象同時出現的機率，購物籃分析（買米可能同時會買雞蛋），演算法：Apriori

分群：(聚類) 將樣本分成相似群組，演算法：K-Mean

降維：減少資料集中變數個數(減少特徵)，但仍保留主要資料不失真，演算法：主成份分析

* 半監督式學習：少量資料有label 🡸Google相簿
* 增強學習：無明確答案，一序列的連續決策，決定下一步。邊做邊學

協助決策

分類和迴歸算法幾乎重合，分類是對個體樣本做出定性判定，迴歸是對個體樣本做出定量判定

有經驗的老師預測某學生考試及格或不及格，這是分類；預測某學生能考多少分，這是迴歸；不管是預測是否及格還是預測考多少分，老師的經驗數據和思考方法是相同的，只是最後的表述不同而已。

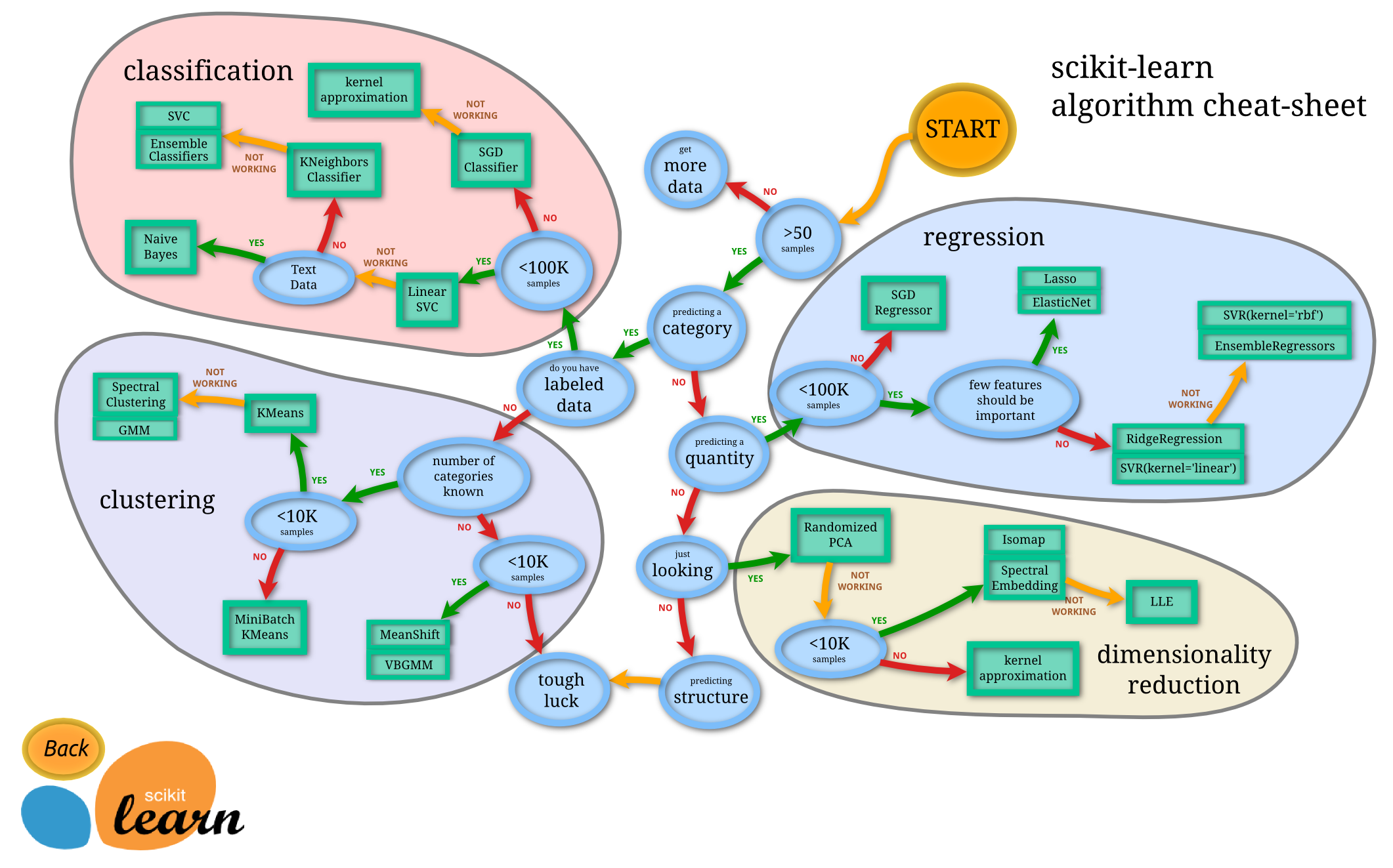
### 深度學習（Deep Learning）

定義：一種實現機器學習的技術。使用模仿人類大腦功能的類神經網路演算法

* DNN
* CNN
* RNN
* LSTM

Scikit-Learning 機器學習地圖

<https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html>



## ML演算法-回歸

### 線性回歸

簡單線性回歸：y=ax+b =>g(x) = w1x1+w2x2+w0

a: coef\_ (coefficient 係數)🡸斜率

b: intercept\_ (intercept 截距)🡸常數

#### 範例1：使用當日氣溫預測當日業績

|  |  |  |
| --- | --- | --- |
| 氣溫 | … | … |
| 營業額 | … | … |

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn.linear\_model import LinearRegression  import matplotlib.pyplot as plt  temperatures = np.array([29, 28, 34, 31, 25, 29, 32, 31, 24, 33, 25, 31, 26, 30])  drink\_sales = np.array([7.7, 6.2, 9.3, 8.4, 5.9, 6.4, 8.0, 7.5, 5.8, 9.1, 5.1, 7.3, 6.5, 8.4])  X = pd.DataFrame(temperatures, columns=["Temperature"])  target = pd.DataFrame(drink\_sales, columns=["Drink\_Sales"])  y = target["Drink\_Sales"]  lm = LinearRegression()  lm.fit(X, y)  print("迴歸係數:", lm.coef\_)  print("截距:", lm.intercept\_ )  # 預測氣溫26, 30度的業績  #new\_temperatures = pd.DataFrame(np.array([26, 30]))  new\_temperatures = np.array([26, 30])  new\_X = pd.DataFrame(new\_temperatures)  predicted\_sales = lm.predict(new\_X)  print(predicted\_sales) |
| 迴歸係數: [0.37378855]  截距: -3.6361233480176187  [6.08237885 7.57753304] <= 溫度[26, 30]分別預測的銷售量 |

|  |
| --- |
| plt.scatter(temperatures, drink\_sales)  # 繪點  regression\_sales = lm.predict(X)  plt.plot(temperatures, regression\_sales, color="blue")  plt.plot(new\_temperatures, predicted\_sales, color="red", marker="o", markersize=10)  plt.show() |
|  |

#### 範例2：使用學生身高來預測體重

|  |  |  |
| --- | --- | --- |
| 身高 | … | … |
| 體重 | … | … |

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn.linear\_model import LinearRegression  import matplotlib.pyplot as plt  heights = np.array([147.9, 163.5, 159.8, 155.1, 163.3, 158.7, 172.0, 161.2, 153.9, 161.6])  weights = np.array([41.7, 60.2, 47.0, 53.2, 48.3, 55.2, 58.5, 49.0, 46.7, 52.5])  X = pd.DataFrame(heights, columns=["Height"])  target = pd.DataFrame(weights, columns=["Weight"])  y = target["Weight"]  lm = LinearRegression()  lm.fit(X, y)  print("迴歸係數:", lm.coef\_)  print("截距:", lm.intercept\_ )  # 預測身高150, 160, 170的體重  new\_heights = np.array([150, 160, 170]))  new\_X = pd.DataFrame(new\_heights)  predicted\_weights = lm.predict(new\_X)  print(predicted\_weights)  plt.scatter(heights, weights)  # 繪點  regression\_weights = lm.predict(X)  plt.plot(heights, regression\_weights, color="blue")  plt.plot(new\_heights, predicted\_weights,           color="red", marker="o", markersize=10)  plt.show() |
| 迴歸係數: [0.62513172]  截距: -48.60353530031602  [45.16622234 51.41753952 57.66885669] |

### 複回歸

y=x1, x2, x3, …

#### 範例1：使用身高和腰圍來預測體重

|  |  |  |
| --- | --- | --- |
| 腰圍 | … | … |
| 身高 | … | … |
| 體重 | … | … |

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn.linear\_model import LinearRegression  waist\_heights = np.array([[67,160], [68,165], [70,167],                            [65,170], [80,165], [85,167],                            [78,178], [79,182], [95,175],                            [89,172]])  weights = np.array([50, 60, 65, 65,                      70, 75, 80, 85,                      90, 81])  X = pd.DataFrame(waist\_heights, columns=["Waist", "Height"])  target = pd.DataFrame(weights, columns=["Weight"])  y = target["Weight"]  lm = LinearRegression()  lm.fit(X, y)  print("迴歸係數:", lm.coef\_)  print("截距:", lm.intercept\_ )  # 預測腰圍和身高[66,164],[82,172]的體重  new\_waist\_heights = pd.DataFrame(np.array([[66, 164],                                             [82, 172]]))  predicted\_weights = lm.predict(new\_waist\_heights)  print(predicted\_weights) |
| 迴歸係數: [0.71013574 1.07794276]  截距: -166.36459730650571  [57.28697457 77.2726885 ] |

#### 範例2：使用店面面積和車站距離來預測單月營業額

|  |  |  |
| --- | --- | --- |
| 店面積 | … | … |
| 距捷運 | … | … |
| 月營收 | … | … |

|  |
| --- |
| import numpy as np  import pandas as pd  from sklearn.linear\_model import LinearRegression  area\_dists = np.array([[10,80], [8,0], [8,200],                         [5,200], [7,300], [8,230],                         [7,40], [9,0], [6,330],                         [9,180]])  sales = np.array([46.9, 36.6, 37.1, 20.8,                      24.6, 29.7, 36.6, 43.6,                      19.8, 36.4])  X = pd.DataFrame(area\_dists, columns=["Area", "Distance"])  target = pd.DataFrame(sales, columns=["Sales"])  y = target["Sales"]  lm = LinearRegression()  lm.fit(X, y)  print("迴歸係數:", lm.coef\_)  print("截距:", lm.intercept\_ )  # 預測腰面積和距離[10,100]的營業額  new\_area\_dists = pd.DataFrame(np.array([[10, 100]]))  predicted\_sales = lm.predict(new\_area\_dists)  print(predicted\_sales) |
| 迴歸係數: [ 4.12351586 -0.03452946]  截距: 6.845523384392724  [44.62773616] |

#### 範例3：使用波士頓資料集預測房價

|  |
| --- |
| import pandas as pd  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  import matplotlib.pyplot as plt  boston = datasets.load\_boston()  #探索資料  print(boston.keys())  print(boston.data.shape)  print(boston.feature\_names)  print(boston.DESCR)  X = pd.DataFrame(boston.data, columns=boston.feature\_names)  target = pd.DataFrame(boston.target, columns=["MEDV"])  y = target["MEDV"]  #訓練預測模型  lm = LinearRegression()  lm.fit(X, y)  print("迴歸係數:", lm.coef\_)  print("截距:", lm.intercept\_ )  coef = pd.DataFrame(boston.feature\_names, columns=["features"])  coef["estimatedCoefficients"] = lm.coef\_  print(coef)  coef.to\_html("Ch15\_3\_2b.html")  #觀察輸出看出RM特徵係數最大, 表示與房價高度相關  plt.scatter(X.RM, y)  plt.xlabel("Average numbwer of rooms per dwelling(RM)")  plt.ylabel("Housing Price(MEDV)")  plt.title("Relationship between RM and Price")  plt.show()  #使用預測模型預測房價  predicted\_price = lm.predict(X)  print(predicted\_price[0:5])  plt.scatter(y, predicted\_price)  plt.xlabel("Price")  plt.ylabel("Predicted Price")  plt.title("Price vs Predicted Price")  plt.show() |
| dict\_keys(['data', 'target', 'feature\_names', 'DESCR', 'filename'])  (506, 13)  ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'  'B' 'LSTAT']  .. \_boston\_dataset:  Boston house prices dataset  ---------------------------  \*\*Data Set Characteristics:\*\*  :Number of Instances: 506  :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.  :Attribute Information (in order):  - CRIM per capita crime rate by town  - ZN proportion of residential land zoned for lots over 25,000 sq.ft.  - INDUS proportion of non-retail business acres per town  - CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)  - NOX nitric oxides concentration (parts per 10 million)  - RM average number of rooms per dwelling  - AGE proportion of owner-occupied units built prior to 1940  - DIS weighted distances to five Boston employment centres  - RAD index of accessibility to radial highways  - TAX full-value property-tax rate per $10,000  - PTRATIO pupil-teacher ratio by town  - B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town  - LSTAT % lower status of the population  - MEDV Median value of owner-occupied homes in $1000's  :Missing Attribute Values: None  :Creator: Harrison, D. and Rubinfeld, D.L.  This is a copy of UCI ML housing dataset.  <https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>  This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.  The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic  prices and the demand for clean air', J. Environ. Economics & Management,  vol.5, 81-102, 1978. Used in Belsley, Kuh & Welsch, 'Regression diagnostics  ...', Wiley, 1980. N.B. Various transformations are used in the table on  pages 244-261 of the latter.  The Boston house-price data has been used in many machine learning papers that address regression  problems.    .. topic:: References  - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.  - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.  迴歸係數: [-1.08011358e-01 4.64204584e-02 2.05586264e-02 2.68673382e+00  -1.77666112e+01 3.80986521e+00 6.92224640e-04 -1.47556685e+00  3.06049479e-01 -1.23345939e-02 -9.52747232e-01 9.31168327e-03  -5.24758378e-01]  截距: 36.45948838509002  features estimatedCoefficients  0 CRIM -0.108011  1 ZN 0.046420  2 INDUS 0.020559  3 CHAS 2.686734  4 NOX -17.766611  5 RM 3.809865  6 AGE 0.000692  7 DIS -1.475567  8 RAD 0.306049  9 TAX -0.012335  10 PTRATIO -0.952747  11 B 0.009312  12 LSTAT -0.524758    [30.00384338 25.02556238 30.56759672 28.60703649 27.94352423] |

輸出結果右方看起來是錯誤資料，可使用殘差圖找出「異常值(Outlier)」的錯誤資料

#### 訓練和測試資料集

實務上會隨機切割「訓練資料集」和「測試資料集」

* 使用訓練資料集訓練預模型
* 使用測試資料集驗證模型績效

|  |
| --- |
| import pandas as pd  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt  boston = datasets.load\_boston()  X = pd.DataFrame(boston.data, columns=boston.feature\_names)  target = pd.DataFrame(boston.target, columns=["MEDV"])  y = target["MEDV"]  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.33, random\_state=5)  lm = LinearRegression()  lm.fit(XTrain, yTrain)  pred\_test = lm.predict(XTest)  plt.scatter(yTest, pred\_test)  plt.xlabel("Price")  plt.ylabel("Predicted Price")  plt.title("Price vs Predicted Price")  plt.show() |
|  |

#### 預測模型績效

評量訓練出的模型是否是好的預測模型

* **MSE(Mean Squared Error)**：說明資料集的點是如何接近回歸線，即測量各點至回歸線距離的平方和、計算平均值。因為是誤差，所以值越小模型越好
* **R Squared(R2)**：也稱決定係數，說明資料集是如何符合回歸線，值是0~1，反應變數的變異比例，使用score()計算，值越大模型越好

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  boston = datasets.load\_boston()  X = pd.DataFrame(boston.data, columns=boston.feature\_names)  target = pd.DataFrame(boston.target, columns=["MEDV"])  y = target["MEDV"]  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.33,                                                  random\_state=5)  lm = LinearRegression()  lm.fit(XTrain, yTrain)  pred\_train = lm.predict(XTrain)  pred\_test = lm.predict(XTest)  MSE\_train = np.mean((yTrain-pred\_train)\*\*2)  MSE\_test = np.mean((yTest-pred\_test)\*\*2)  print("訓練資料的MSE:", MSE\_train)  print("測試資料的MSE:", MSE\_test)  print("訓練資料的R-squared:", lm.score(XTrain, yTrain))  print("測試資料的R-squared:", lm.score(XTest, yTest)) |

波士頓房價程式，增加MSE計算

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  boston = datasets.load\_boston()  X = pd.DataFrame(boston.data, columns=boston.feature\_names)  target = pd.DataFrame(boston.target, columns=["MEDV"])  y = target["MEDV"]  lm = LinearRegression()  lm.fit(X, y)  predicted\_price = lm.predict(X)  print(predicted\_price[0:5])  MSE = np.mean((y-predicted\_price)\*\*2)  print("MSE:", MSE)  print("R-squared:", lm.score(X, y)) |
| 訓練資料的MSE: 19.54675847353467  測試資料的MSE: 28.530458765974725  訓練資料的R-squared: 0.7551332741779998  測試資料的R-squared: 0.6956551656111593 |

#### 殘差圖

異常值(Outlier)會大幅影響模型績效，使用殘差圖(Residual Plots)找出這些異常值

殘差值=觀察值-預測值

* =0 預測值符合測試資料，最佳情形
* >0 預測值太低
* <0 預測值太高

|  |
| --- |
| import pandas as pd  from sklearn import datasets  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt  boston = datasets.load\_boston()  X = pd.DataFrame(boston.data, columns=boston.feature\_names)  target = pd.DataFrame(boston.target, columns=["MEDV"])  y = target["MEDV"]  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.33,                                                  random\_state=5)  lm = LinearRegression()  lm.fit(XTrain, yTrain)  pred\_train = lm.predict(XTrain)  pred\_test = lm.predict(XTest)  #殘差值作為Y軸，預設值X軸  plt.scatter(pred\_train, pred\_train-yTrain, c="b", s=40, alpha=0.5, label="Training Data")  plt.scatter(pred\_test, pred\_test-yTest, c="r", s=40, label="Test Data")  plt.hlines(y=0, xmin=0, xmax=50) #在y=0處，畫出0~50水平線  plt.title("Residual Plot")  plt.ylabel("Residual Value")  plt.legend()  plt.show() |
|  |

找到資料偏離中間水平線很遠的點，就是異常值(Outlier)

### Logistic回歸

線性回歸 解決連續數值的評估和預測

Logistic回歸 解決分類問題，應用二元性資料，如男女、成功失敗、真假等

和線性回歸相同觀念，結果需要使用Logistic函數(Sigmoid函數)轉換為0~1之間的機率

#### S函數圖形

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  t = np.arange(-6, 6, 0.1)  S = 1/(1+(np.e\*\*(-t)))  plt.plot(t, S)  plt.title("sigmoid function")  plt.show() |
|  |

#### 範例：Titanic生存預測

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  from sklearn import preprocessing, linear\_model  titanic = pd.read\_csv("titanic.csv")  print(titanic.info())  #預處理-------------------------  # 將年齡的空值填入年齡的中位數  age\_median = np.nanmedian(titanic["Age"])  print("年齡中位數", age\_median)  new\_age = np.where(titanic["Age"].isnull(),                     age\_median, titanic["Age"])  titanic["Age"] = new\_age  titanic["SexCode"] = np.where(titanic["Sex"]=="female",1,0)  print(titanic)  # 轉換欄位值成為數值  label\_encoder = preprocessing.LabelEncoder()  encoded\_class = label\_encoder.fit\_transform(titanic["PClass"])  #------------------------------  X = pd.DataFrame([encoded\_class, titanic["SexCode"], titanic["Age"]]).T  y = titanic["Survived"]  logistic = linear\_model.LogisticRegression()  logistic.fit(X, y)  print("迴歸係數:", logistic.coef\_)  print("截距:", logistic.intercept\_ )  preds = logistic.predict(X)  print(pd.crosstab(preds, titanic["Survived"]))  pd.crosstab(preds, titanic["Survived"]).to\_html("Ch15\_4\_2a.html")  print((804+265)/(804+185+59+265))  print(logistic.score(X, y)) | /kaggle/input/testtitanic/titanic\_data.csv  <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  None  年齡中位數 28.0  PassengerId Survived Pclass \  0 1 0 3  1 2 1 1  2 3 1 3  3 4 1 1  4 5 0 3  .. ... ... ...  886 887 0 2  887 888 1 1  888 889 0 3  889 890 1 1  890 891 0 3  Name Sex Age SibSp \  0 Braund, Mr. Owen Harris male 22.0 1  1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1  2 Heikkinen, Miss. Laina female 26.0 0  3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1  4 Allen, Mr. William Henry male 35.0 0  .. ... ... ... ...  886 Montvila, Rev. Juozas male 27.0 0  887 Graham, Miss. Margaret Edith female 19.0 0  888 Johnston, Miss. Catherine Helen "Carrie" female 28.0 1  889 Behr, Mr. Karl Howell male 26.0 0  890 Dooley, Mr. Patrick male 32.0 0  Parch Ticket Fare Cabin Embarked SexCode  0 0 A/5 21171 7.2500 NaN S 0  1 0 PC 17599 71.2833 C85 C 1  2 0 STON/O2. 3101282 7.9250 NaN S 1  3 0 113803 53.1000 C123 S 1  4 0 373450 8.0500 NaN S 0  .. ... ... ... ... ... ...  886 0 211536 13.0000 NaN S 0  887 0 112053 30.0000 B42 S 1  888 2 W./C. 6607 23.4500 NaN S 1  889 0 111369 30.0000 C148 C 0  890 0 370376 7.7500 NaN Q 0  [891 rows x 13 columns]  迴歸係數: [[-1.14143901 2.51942976 -0.03270198]]  截距: [0.92404868]  Survived 0 1  row\_0  0 454 94  1 95 248  0.8141660319878141  0.7878787878787878 |

crosstab()交叉分析表

* 左上：預測死亡，實際死亡人數，預測正確
* 右上：預測死亡，實際存活人數，預測錯誤
* 左下：預測生存，實際死亡人數，預測錯誤
* 右下：預測生存，實際存活人數，預測正確

ps. 從探索性資料分析可知Age和Survived相關係數低，故只須使用PClass和SexCode就有不錯的正確率

|  |  |
| --- | --- |
| import pandas as pd  from sklearn import preprocessing, linear\_model  import numpy as np  titanic = pd.read\_csv("titanic.csv")  print(titanic.info())  # 轉換欄位值成為數值  label\_encoder = preprocessing.LabelEncoder()  encoded\_class = label\_encoder.fit\_transform(titanic["PClass"])  titanic["SexCode"] = np.where(titanic["Sex"]=="female",1,0)  X = pd.DataFrame([encoded\_class,                    titanic["SexCode"]]).T  y = titanic["Survived"]  logistic = linear\_model.LogisticRegression()  logistic.fit(X, y)  print("迴歸係數:", logistic.coef\_)  print("截距:", logistic.intercept\_ )  preds = logistic.predict(X)  print(pd.crosstab(preds, titanic["Survived"]))  pd.crosstab(preds, titanic["Survived"]).to\_html("Ch15\_4\_2b.html")  print((840+228)/(840+222+23+228))  print(logistic.score(X, y)) | <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  None  迴歸係數: [[-0.93682895 2.55361267]]  截距: [-0.30131852]  Survived 0 1  row\_0  0 468 109   1. 81 233   0.8134044173648134  0.7867564534231201 |

正確率：0.8134

## ML演算法-分類與分群

決策樹、K鄰近演算法：是多元分類(Multiclass Classification)，可執行多種類別分類

### 決策樹

由一序列條件決策組成，擁有不同的機率和分類來決定下一步

#### 範例：使用決策樹的Titanic生存預測

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  from sklearn import preprocessing, tree  from sklearn.model\_selection import train\_test\_split  titanic = pd.read\_csv("titanic.csv")  # 轉換欄位值成為數值  label\_encoder = preprocessing.LabelEncoder()  encoded\_class = label\_encoder.fit\_transform(titanic["PClass"])  titanic["SexCode"] = np.where(titanic["Sex"]=="female",1,0)  X = pd.DataFrame([titanic["SexCode"], encoded\_class]).T  X.columns = ["SexCode", "PClass"]  y = titanic["Survived"]  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.25, random\_state=1)  dtree = tree.DecisionTreeClassifier()  dtree.fit(XTrain, yTrain)  print("準確率:", dtree.score(XTest, yTest))  #建立交叉分析表，回傳n行k列矩陣，i行j列的值是模型預測第i個樣本為j的機率  preds = dtree.predict\_proba(X=XTest)  print(pd.crosstab(preds[:,0], columns=[XTest["PClass"], XTest["SexCode"]]))  pd.crosstab(preds[:,0], columns=[XTest["PClass"], XTest["SexCode"]]).to\_html("Ch16\_1\_2.html")  #使用GraphViz輸出決策樹圖形  with open("tree.dot", "w") as f:      f = tree.export\_graphviz(dtree,                               feature\_names=["Sex", "Class"],                               out\_file=f) | 準確率: 0.7533632286995515  PClass 0 1 2  SexCode 0 1 0 1 0 1  row\_0  0.029851 0 27 0 0 0 0  0.084746 0 0 0 17 0 0  0.532710 0 0 0 0 0 37  0.655556 32 0 0 0 0 0  0.837500 0 0 28 0 0 0  0.871698 0 0 0 0 82 0  add Codeadd Markdown |

從交叉分析表看出，第3等級女性死亡率超過60%，第1等級10%，所有男性都超過66%

#### 範例：使用決策樹分類IRIS鳶尾花

使用花瓣和花萼來分類IRIS

|  |
| --- |
| import pandas as pd  from sklearn import datasets  from sklearn import tree  from sklearn.model\_selection import train\_test\_split  iris = datasets.load\_iris()  #探索資料集  print(iris.keys())  print(iris.data.shape)  print(iris.feature\_names)  print(iris.DESCR)  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  target = pd.DataFrame(iris.target, columns=["target"])  y = target["target"]  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.33, random\_state=1)  dtree = tree.DecisionTreeClassifier(max\_depth = 8) #決策樹最大深度8  dtree.fit(XTrain, yTrain)  print("準確率:", dtree.score(XTest, yTest))  print(dtree.predict(XTest))  print(yTest.values)  with open("tree2.dot", "w") as f:      f = tree.export\_graphviz(dtree,                               feature\_names=iris.feature\_names,                               out\_file=f) |
| dict\_keys(['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_names', 'filename'])  (150, 4)  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']  .. \_iris\_dataset:  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:  ============== ==== ==== ======= ===== ====================  Min Max Mean SD Class Correlation  ============== ==== ==== ======= ===== ====================  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  The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken  from Fisher's paper. Note that it's the same as in R, but not as in the UCI  Machine Learning Repository, which has two wrong data points.  This is perhaps the best known database to be found in the  pattern recognition literature. Fisher's paper is a classic in the field and  is referenced frequently to this day. (See Duda & Hart, for example.) The  data set contains 3 classes of 50 instances each, where each class refers to a  type of iris plant. One class is linearly separable from the other 2; the  latter are NOT linearly separable from each other.  .. topic:: References  - Fisher, R.A. "The use of multiple measurements in taxonomic problems"  Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to  Mathematical Statistics" (John Wiley, NY, 1950).  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.  (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.  - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System  Structure and Classification Rule for Recognition in Partially Exposed  Environments". IEEE Transactions on Pattern Analysis and Machine  Intelligence, Vol. PAMI-2, No. 1, 67-71.  - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions  on Information Theory, May 1972, 431-433.  - See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s AUTOCLASS II  conceptual clustering system finds 3 classes in the data.  - Many, many more ...  準確率: 0.96  [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1  0 1 2 2 0 1 2 1 2 0 0 0 1]  [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1  0 1 2 2 0 2 2 1 2 0 0 0 1] |

### K鄰近演算法（KNN）

KNN(K Nearest Nieghbor Alogrithm)

使用K個最接近目標資料的資料來預測目標資料所屬的類別

#### 範例：分類面紙是好是壞

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import neighbors  X = pd.DataFrame({     "durability": [7, 7, 3, 1],     "strength":   [7, 4, 4, 4]  })  y = np.array([0, 0, 1, 1])  k = 3  knn = neighbors.KNeighborsClassifier(n\_neighbors=k)  knn.fit(X, y)  # 預測新產品[3,7]的分類 1:好 0:壞  new\_tissue = pd.DataFrame(np.array([[3, 7]]))  pred = knn.predict(new\_tissue)  print(pred) |
| [1] |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 編號 | 耐酸性 | 強度 | 分類 | 距離(3, 7) |
| 1 | 7 | 7 | 壞 | (7-3)2+(7-7)2=16 |
| 2 | 7 | 4 | 壞 | (7-3)2+(4-7)2=25 |
| 3 | 3 | 4 | 好 | (3-3)2+(4-7)2=9 |
| 4 | 1 | 4 | 好 | (1-3)2+(4-7)2=13 |

因K=3, 所以只取三個鄰近值, 編號2距離最遠不計算, 其他編號1,3,4兩個好, 一個壞, 故判斷分類結果為「好」

#### 範例：使用KNN分類IRIS

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  import matplotlib.pyplot as plt  from sklearn import datasets  from sklearn import neighbors  from sklearn.model\_selection import train\_test\_split  iris = datasets.load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  X.columns = ["sepal\_length","sepal\_width","petal\_length","petal\_width"]  target = pd.DataFrame(iris.target, columns=["target"])  y = target["target"]  colmap = np.array(["r", "g", "y"])  plt.figure(figsize=(10,5))  plt.subplot(1, 2, 1)  plt.subplots\_adjust(hspace = .5)  plt.scatter(X["sepal\_length"], X["sepal\_width"], color=colmap[y])  plt.xlabel("Sepal Length")  plt.ylabel("Sepal Width")  plt.subplot(1, 2, 2)  plt.scatter(X["petal\_length"], X["petal\_width"], color=colmap[y])  plt.xlabel("Petal Length")  plt.ylabel("Petal Width")  plt.show()  #從資料探索觀察可看出Sentosas花萼(Sepal)明顯比另兩類的尺寸短且寬  #建立模型  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.33, random\_state=1)  k = 3  knn = neighbors.KNeighborsClassifier(n\_neighbors=k)  knn.fit(X, y)  print("準確率:", knn.score(XTest, yTest))  print(knn.predict(XTest))  print(yTest.values) |
| 準確率: 0.98  [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1  0 1 2 2 0 1 2 1 2 0 0 0 1]  [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1  0 1 2 2 0 2 2 1 2 0 0 0 1] |

#### 如何選擇K值

K鄰近演算法的K值會影響分類的準確度🡺可以使用迴圈執行多次不同的K值的分類來找出最佳K值。

一般來說，K值的上限是訓練資料集的20%

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn import neighbors  from sklearn.model\_selection import train\_test\_split  import matplotlib.pyplot as plt  iris = datasets.load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  X.columns = ["sepal\_length","sepal\_width","petal\_length","petal\_width"]  target = pd.DataFrame(iris.target, columns=["target"])  y = target["target"]  XTrain, XTest, yTrain, yTest = train\_test\_split(X, y, test\_size=0.33, random\_state=1)  Ks = np.arange(1, round(0.2\*len(XTrain) + 1))  accuracies=[]  for k in Ks:      knn = neighbors.KNeighborsClassifier(n\_neighbors=k)      knn.fit(X, y)      accuracy = knn.score(XTest, yTest)      accuracies.append(accuracy)    plt.plot(Ks, accuracies)  plt.show() |
|  |

可看出很多K值準確率都在98%以上，變動不大，不容易看出K值走勢找出最佳K值區間，可使用交叉驗證找出最佳K值

#### 交叉驗證(Cross Validation)的K值最佳化

交叉驗證目的：使用資料集的所有資料來訓練和建立模型

* K-fold交叉驗證

將資料集分割成2或更多的分隔區（Partitions），並將每一個分割區一一作為測試資料集，將其他分隔區作為訓練資料集

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn import neighbors  from sklearn.model\_selection import cross\_val\_score  import matplotlib.pyplot as plt  iris = datasets.load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  X.columns = ["sepal\_length","sepal\_width","petal\_length","petal\_width"]  target = pd.DataFrame(iris.target, columns=["target"])  y = target["target"]  Ks = np.arange(1, round(0.2\*len(X) + 1))  accuracies=[]  for k in Ks:      knn = neighbors.KNeighborsClassifier(n\_neighbors=k)      scores = cross\_val\_score(knn, X, y, scoring="accuracy", cv=10)      accuracies.append(scores.mean())  plt.plot(Ks, accuracies)  plt.show() |
|  |

可看出K值在12~18之間最好，超過18準確率開始下降

### K-means演算法

K-means Clustering: 又稱K平均數分群

* 分類：在已知資料集分類的情況，替新東西進行分類
* 分群：在未知資料集分類的情況，直接使用特徵來進行分類 🡸非監督式學習

步驟：

1. 先找出K個群組的重心(Centroid)，
2. 資料集就以距離最近重心來分成群組後，
3. 重新計算群組的新重心後，
4. 再分群一次，
5. 重複操作3~4完成分群

#### 範例：依據動物的體重和身長來分群

|  |  |  |
| --- | --- | --- |
| 身長 |  |  |
| 體重 |  |  |

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  from sklearn import cluster  import matplotlib.pyplot as plt  df = pd.DataFrame({     "length": [51, 46, 51, 45, 51, 50, 33,                38, 37, 33, 33, 21, 23, 24],     "weight": [10.2, 8.8, 8.1, 7.7, 9.8, 7.2, 4.8,                4.6, 3.5, 3.3, 4.3, 2.0, 1.0, 2.0]  })  k = 3  kmeans = cluster.KMeans(n\_clusters=k, random\_state=12)  kmeans.fit(df)  print(kmeans.labels\_)  colmap = np.array(["r", "g", "y"])  plt.scatter(df["length"], df["weight"], color=colmap[kmeans.labels\_])  plt.show() | [1 1 1 1 1 1 2 2 2 2 2 0 0 0] |

K-means就是在分類，只是不知道各群組是哪一類

#### 範例：使用K-means演算法分群IRIS鳶尾花

因非監督式學習，只要訓練資料X即可

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn import cluster  import matplotlib.pyplot as plt  iris = datasets.load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  X.columns = ["sepal\_length","sepal\_width","petal\_length","petal\_width"]  y = iris.target  k = 3  kmeans = cluster.KMeans(n\_clusters=k, random\_state=12)  kmeans.fit(X)  print(kmeans.labels\_)  print(y)  colmap = np.array(["r", "g", "y"])  plt.figure(figsize=(10,5))  plt.subplot(1, 2, 1)  plt.subplots\_adjust(hspace = .5)  plt.scatter(X["petal\_length"], X["petal\_width"],              color=colmap[y])  plt.xlabel("Petal Length")  plt.ylabel("Petal Width")  plt.title("Real Classification")  plt.subplot(1, 2, 2)  plt.scatter(X["petal\_length"], X["petal\_width"],              color=colmap[kmeans.labels\_])  plt.xlabel("Petal Length")  plt.ylabel("Petal Width")  plt.title("K-means Classification")  plt.show() | [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2  2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2  2 1]  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2  2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  2 2] |

##### 修正分群標籤錯誤重繪散佈圖

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn import cluster  import matplotlib.pyplot as plt  iris = datasets.load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  X.columns = ["sepal\_length","sepal\_width","petal\_length","petal\_width"]  y = iris.target  k = 3  kmeans = cluster.KMeans(n\_clusters=k, random\_state=12)  kmeans.fit(X)  print("K-means Classification:")  print(kmeans.labels\_)  # 修正標籤錯誤  pred\_y = np.choose(kmeans.labels\_, [1,0,2]).astype(np.int64)  print("K-means Fix Classification:")  print(pred\_y)  print("Real Classification:")  print(y)  colmap = np.array(["r", "g", "y"])  plt.figure(figsize=(10,5))  plt.subplot(1, 2, 1)  plt.subplots\_adjust(hspace = .5)  plt.scatter(X["petal\_length"], X["petal\_width"],              color=colmap[y])  plt.xlabel("Petal Length")  plt.ylabel("Petal Width")  plt.title("Real Classification")  plt.subplot(1, 2, 2)  plt.scatter(X["petal\_length"], X["petal\_width"],              color=colmap[pred\_y])  plt.xlabel("Petal Length")  plt.ylabel("Petal Width")  plt.title("K-means Classification")  plt.show() | K-means Classification:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2  2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2  2 1]  K-means Fix Classification:  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2  2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2  2 0]  Real Classification:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2  2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  2 2] |

#### K-means模型的績效測量

|  |  |
| --- | --- |
| import pandas as pd  import numpy as np  from sklearn import datasets  from sklearn import cluster  import sklearn.metrics as sm  iris = datasets.load\_iris()  X = pd.DataFrame(iris.data, columns=iris.feature\_names)  X.columns = ["sepal\_length","sepal\_width","petal\_length","petal\_width"]  y = iris.target  k = 3  kmeans = cluster.KMeans(n\_clusters=k, random\_state=12)  kmeans.fit(X)  # 修正標籤錯誤  pred\_y = np.choose(kmeans.labels\_, [1,0,2]).astype(np.int64)  # 積效矩陣（準確度：Accuracy）  print(sm.accuracy\_score(y, pred\_y))  # 混淆矩陣(Confusion Matrix)  print(sm.confusion\_matrix(y, pred\_y)) | 0.24  [[ 0 50 0]  [48 0 2]  [14 0 36]] |

## Dataset資料集

### **SKlearn**資料集

函數名以 load 開頭的是模塊內置的小型數據集；函數名以 fetch 開頭，是需要從外部數據源下載的大型數據集

每個二維的數據集對應着一個一維的標籤集，用於標識每個樣本的所屬類別或屬性值。通常數據集用大寫字母 X 表示，標籤集用小寫字母 y 表示。

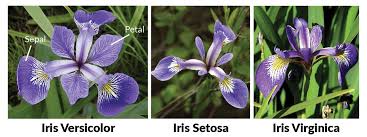
* datasets.load\_boston([return\_X\_y]) ：加載波士頓房價數據集
* datasets.load\_breast\_cancer([return\_X\_y]) ：加載威斯康星州乳腺癌數據集
* datasets.load\_diabetes([return\_X\_y]) ：加載糖尿病數據集
* datasets.load\_digits([n\_class, return\_X\_y]) ：加載數字數據集
* datasets.load\_iris([return\_X\_y]) ：加載鳶尾花數據集。
* datasets.load\_linnerud([return\_X\_y]) ：加載體能訓練數據集
* datasets.load\_wine([return\_X\_y]) ：加載葡萄酒數據集
* datasets.fetch\_20newsgroups([data\_home, …]) ：加載新聞文本分類數據集
* datasets.fetch\_20newsgroups\_vectorized([…]) ：加載新聞文本向量化數據集
* datasets.fetch\_california\_housing([…]) ：加載加利福尼亞住房數據集
* datasets.fetch\_covtype([data\_home, …]) ：加載森林植被數據集

**共同屬性：**

* data # 資料(X)
* feature\_names # 特徵(X.name)
* target # 分類(y)
* target\_name # 分類名稱(y.name)
* DESC # 資料集說明, filename 檔名, frame??

print(dir(iris))：查詢哪些屬性

#### IRIS Dataset(鳶尾花)



鳶尾花IRIS資料集:

<https://zhuanlan.zhihu.com/p/108393576>

包含了150個鳶尾花樣本，對應3種鳶尾花，各50個樣本，以及它們各自對應的4種關於花外形的資料 ，適用於分類任務。

from sklearn import datasets

iris = datasets.load\_iris() # 導入鳶尾花數據

* data # 資料
* feature\_names # 特徵：[Sepal花萼長，花萼寬，Petal花瓣長，花瓣寬]
* target # 分類(0,1,2)
* target\_name # 分類名稱(setosa, versicolor, virginica)
* DESC # 資料集說明, filename 檔名, frame??

|  |
| --- |
| from sklearn import datasets  iris=datasets.load\_iris()  print(type(iris))  print('KEY=', iris.keys())  print('Data=', type(iris.data), iris.data.shape)  print('feature=',iris.feature\_names)  species = iris.target   #Y:分三類  print('Target\_Name=', iris.target\_names)  print('Target=', type(species), species.shape)  print(iris.DESCR) |

ps. 可去kaggle下載dataset

|  |
| --- |
| import numpy  FILENAME = "data/diabetes.csv"  dataset1 = numpy.loadtxt(FILENAME, delimiter=",", skiprows=1)  print(type(dataset1))  print(dataset1.shape) |

最後一欄是Y

### **Tensorflow**資料集

|  |
| --- |
| from tensorflow.keras.datasets import imdb  (X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data()  X = numpy.concatenate((X\_train, X\_test), axis=0)  y = numpy.concatenate((y\_train, y\_test), axis=0)  print(X.shape)  print(y.shape) |

### csv轉成dataset

|  |
| --- |
| import numpy as np # linear algebra  import csv  #from sklearn.datasets.base import Bunch  from sklearn.utils import Bunch  def load\_my\_dataset():      with open('../input/iris-flower-dataset/IRIS.csv') as csv\_file:          data\_file = csv.reader(csv\_file)            head = next(data\_file)          n\_samples = 150 #number of data rows, don't count header          n\_features = 4 #number of columns for features, don't count target column            feature\_names = head[:-1]          #feature\_names = ['f1','f2','f3','f4'] #adjust accordingly          target\_names = ['setosa' 'versicolor' 'virginica'] #adjust accordingly            data = np.empty((n\_samples, n\_features))          target = np.empty((n\_samples,), dtype='S20')          for i, sample in enumerate(data\_file):                  data[i] = np.array(sample[:-1], dtype=np.float64)                  target[i] = np.array(sample[-1], dtype='S20')      return Bunch(data=data, target=target, feature\_names = feature\_names, target\_names = target\_names)  iris = load\_my\_dataset()  # get iris dataset  print(type(iris))  print('KEY=', iris.keys())  print('Data=', type(iris.data), iris.data.shape)  print('feature=',iris.feature\_names)  print('Target\_Name=', iris.target\_names)  species = iris.target   #Y:分三類  print('Target=', type(species), species.shape)  #print(iris.DESCR) |
|  |

# 其他

## ****產業AI化正面臨的三大挑戰****

### 挑戰一、重新建立對人才資本的概念

未來AI所需的是「整合性人才」

### 挑戰二、數據的「質」比量更重要

倘若瑕疵樣態不足或過於單一，便容易使AI在「欠缺經驗」下而影響其辨識精準度。這也是為什麼大多數的企業通常需要花80%的時間用於整理資料，然後才能開始訓練AI的原因之一。只要有工具或流程確保資料夠乾淨，模型也能有不錯的結果。

### 挑戰三、沒有一勞永逸的AI模型

後期在AI模型管理上，許多企業甚至會以為管理AI模型跟管理網頁一樣，但其實兩者間有很大的落差，AI模型在開發完成後，通常會隨著時間失去它的精準度，即使已經有訓練好的模型，企業還是必須持續地訓練，反覆校正，後續的維運仍然相當重要。