線性回歸模型與評估

教授 ： 黃鈺晴

監督式學習
模型概覽

線性迴歸
Linear Regression

用於預測連續數值，模型簡單、計算快速，是最佳入門模型。

決策樹
Decision Tree

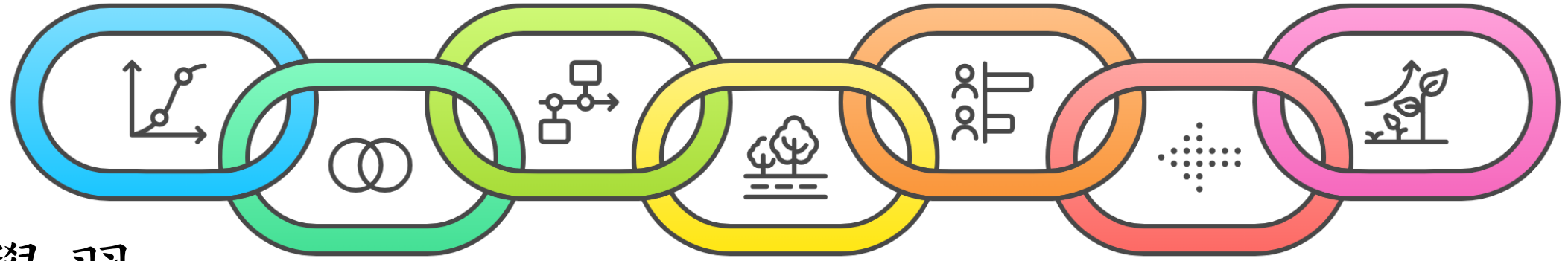
使用條件分支進行預測，易於解釋，適用於分類與迴歸問題。

支持向量機
Support Vector Machine

找出區分不同類別的「最佳邊界」，適用於高維資料，分類效果佳。

梯度提升樹
Gradient Boosting(XGBoost)

將多個弱模型逐步強化疊加成強模型。準確率高，但訓練耗時長。



羅吉斯迴歸
Logistic Regression

輸出機率並轉換為類別，被用於解決分類問題。

隨機森林
Random Forest

多棵決策樹組成的集成模型，提升穩定性與準確率，有效減少過擬合。

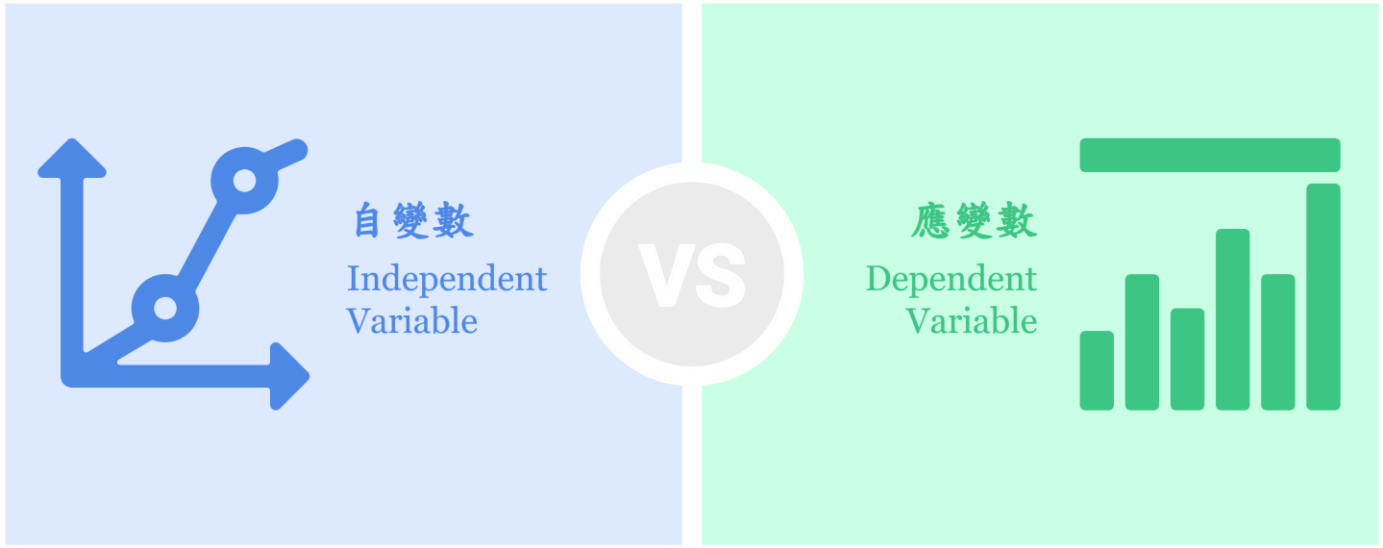
K 最近鄰
K-Nearest Neighbors

依據鄰居進行預測，無需訓練，但對資料規模與特徵比例敏感。

線性迴歸介紹

- 線性迴歸 (Linear Regression) 是一種常被用來處理連續變數 (Continuous Variable) 的迴歸模型。
- 它的主要目的是探討自變數 (Independent Variable) 與應變數 (Dependent Variable) 之間的關係。

- 「因」，影響別人
- 圖表中的X軸
- 機器學習中的「特徵 (Features)」



- 「果」，被別人影響
- 圖表中的Y軸
- 機器學習中的「目標 (Target)」

線性迴歸的「因果關係」迷思

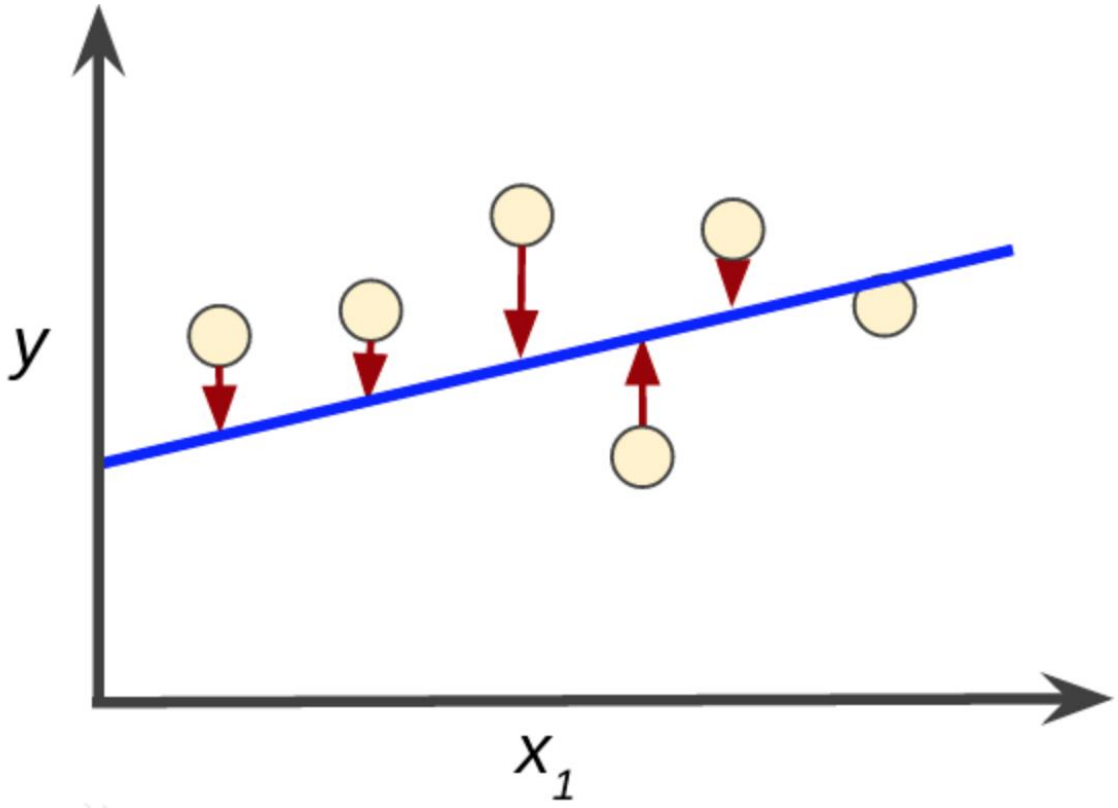
- 線性迴歸並非一個因果關係的分析，而是告訴我們**兩個變數之間的統計關係**。

- 舉例來說，我們可以用體重預測身高，也可以用身高預測體重。這兩個變數之間存在關係，但我們不能說體重是身高的「因」，或身高是體重的「因」。

線性回歸的目標

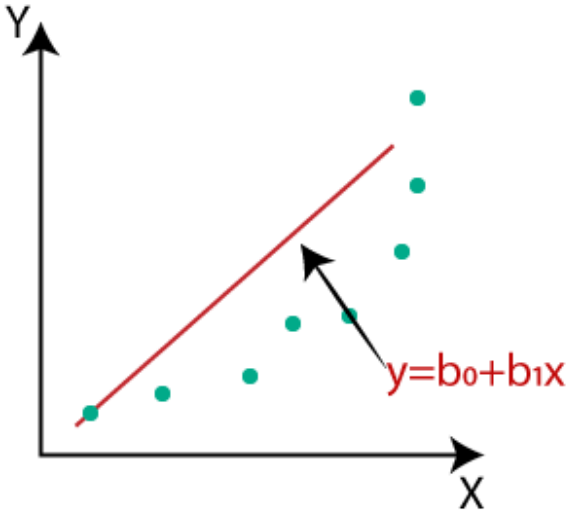
找到一個「最好的線性迴歸預測式」，來盡可能地接近真實的目標式

- 目標式 (Target Formula)：這是理想中存在於兩個變數之間的關係，但通常是我們觀測不到的。
- 預測式 (Prediction Formula)：這是我們透過模型來估計和逼近目標式的公式。



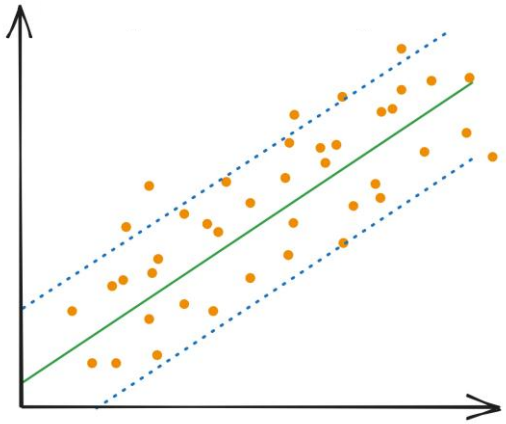
線性迴歸的類型

簡單線性回歸
(Simple Linear Regression)



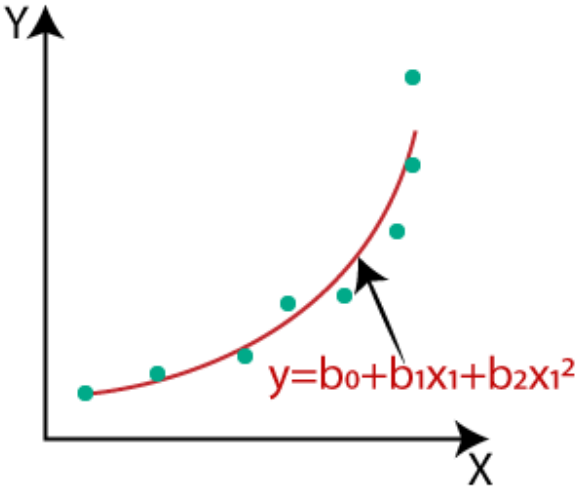
- 一個自變數 影響 一個應變數。
- 例如：房子大小 (自變數) 影響 房價 (應變數)。

多元線性回歸
(Multiple Linear Regression)



- 多個自變數 影響 一個應變數。
- 例如：房子大小、地點、屋齡 (多個自變數) 共同影響 房價 (應變數)。

多項式回歸
(Polynomial Regression)



- 當特徵與目標之間的關係並非單純的線性時使用。模型結果會呈現曲線，而非直線。
- 透過高維度的多項式來建構模型，例如自變數可能會出現高次方 (X^2, X^3等)。

線性迴歸的預測公式

簡單線性回歸
(Simple Linear Regression)

$$y = b_0 + b_1 + x_1$$

多元線性回歸
(Multiple Linear Regression)

$$y = b_0 + b_1 + x_1 + b_2 + x_2 + \ldots + b_n + x_n$$

多項式回歸
(Polynomial Regression)

$$y = b_0 + b_1 + x_1 + b_2 + x^2_2 + \ldots + b_n + x^n_n$$

簡單線性回歸 (Simple Linear Regression)

$$y = b_0 + b_1 + x_1$$

- y: 預測的應變數 (目標) 值。
- $x1$：是自變數 (特徵) 值。
- b_0：稱為截距項 (Intercept)。
  - 它代表當 X 為 0 時，Y 的預測值。
- b_1 ：稱為係數項 (Coefficient) 或斜率 (Slope)。
  - 它代表當 X 變動一個單位時，Y 會變動多少。
- 當我們有多個自變數時，公式會擴展為

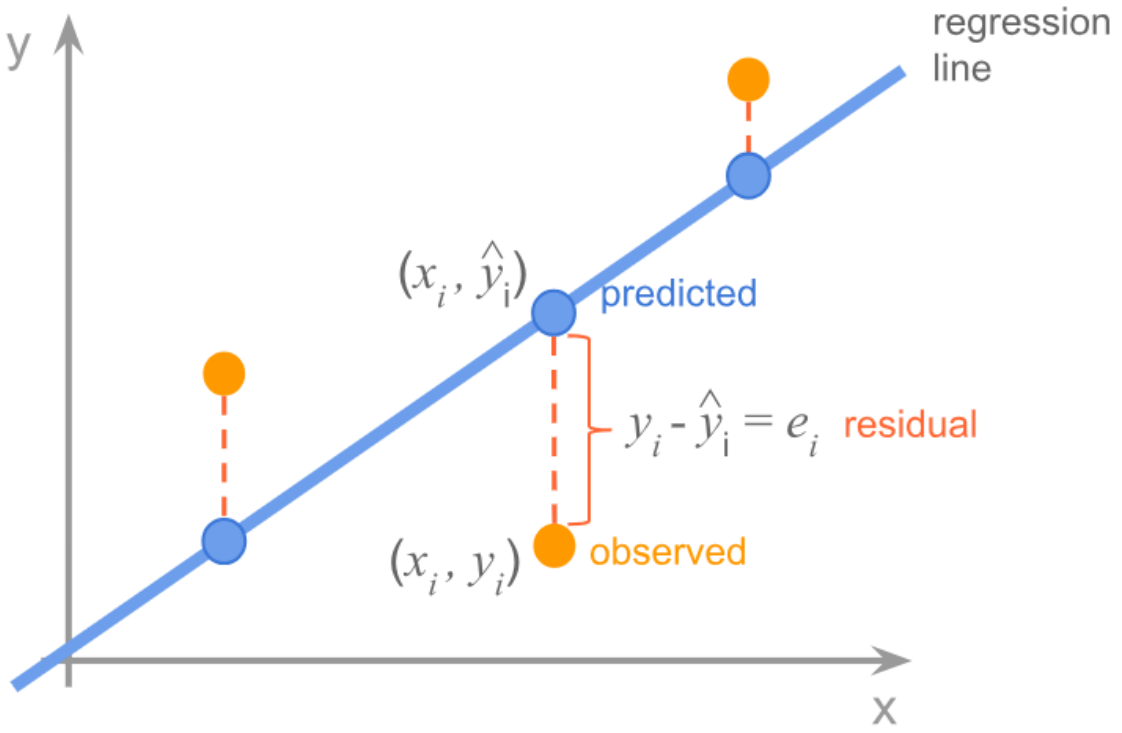
多元線性回歸 (Multiple Linear Regression)

$$y = b_0 + b_1 + x_1 + b_2 + x_2 + \ldots + b_n + x_n$$

- 其中 $b_1, b_2, \ldots, b_n$ 分別是各個自變數的係數，它們代表在其他自變數保持不變的情況下，該自變數每增加一個單位，應變數的變動量。

模型訓練階段: 如何找到「最好」的預測式係數?

- 透過觀察 誤差 (Error) 或 殘差 (Residual) 來決定。
  - 誤差 (Error)：觀測值與真實值之間的差，由於真實值通常觀測不到，所以很難計算誤差。
  - 殘差 (Residual): 觀測值與預測值之間的差，殘差會根據我們建立的模型而有所改變，在訓練模型時通常是分析與處理「殘差」而非「誤差」。



最小平方法（Least Squares Estimate）

- 目標: 找到最佳的 b_0 和 b_1 值，讓預測線盡可能地貼近實際資料點。
- 核心理念: 最小化所有資料點的殘差平方和（SSE）。
  - 殘差(e): 模型預測值（$y^$）與真實觀測值（y）之間的差距
  - 為什麼要平方?
    - 防止正負差異互相抵銷（有的預測高，有的低）。
    - 強調大錯誤：差距越大，平方後懲罰越重。
  - $SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{b}_0 - \hat{b}_1 x_i)^2$
  - 殘差平方和（SSE）越小，表示模型對資料的擬合 (fit) 越好
- 實務中我們常會進一步使用一些標準化或比較性更強的指標來評估模型表現，包含平均平方誤差（MSE）、平均平方誤差（MSE）與決定係數（R-squared）

模型評估指標

- 平均絕對誤差（MAE, Mean Absolute Error）
  - 將每個資料點的殘差取絕對值後平均，不用平方

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i^2\right|$$

  - 對極端值（離群值）較不敏感，能反映平均預測偏差。
  - 易於理解與解釋
- 平均平方誤差（MSE, Mean Squared Error）
  - 計算每個資料點殘差的平方平均值。
  - 本質上就是 SSE 除以資料筆數（n）

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

  - 越小越好，代表模型預測值與真實值越接近。

模型評估指標

- 均方根誤差（RMSE, Root Mean Squared Error）
  - MSE 的平方根

$$RMAE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i^2\right|}$$

  - 結合了 MSE 的特性，對誤差大小變化敏感，常用於比較不同模型之間的預測表現。
- 決定係數（R-squared）

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum(y_i - \hat{y_i})^2}{\sum(y_i - \bar{y})^2}$$

  - R^2範圍 0～1，越接近 1 表示模型越能解釋資料。
    - $R^2 = 1$：完美預測
    - $R^2 = 0$：模型沒有解釋力
  - model.score() 方法可以直接計算 R 平方值。

評估指標比較

- MAE：
  - 優點：直觀、穩定，適合對異常值不敏感的情境
  - 缺點：無法反映誤差的變異程度
  - 適合需要簡單直觀結果的應用，如房價、銷售量預測。
- MSE：
  - 優點：能強調大的預測誤差，對模型調整更有指引性
  - 缺點：單位是平方，不易直接解釋
  - 適用於希望強調大誤差的場景，如品質控制或金融風險評估。
- RMSE：
  - 優點：結合 MAE 與 MSE 的優點，單位可解釋
  - 缺點：相較 MAE 計算略為複雜
  - 廣泛用於許多回歸問題，如天氣預報、能源預測等。

線性回歸的優缺點

Pros

1. **易於實作**
   概念直觀，實作相對簡單。
2. **計算速度快**
   大型資料集處理效率高。
3. **高解釋性**
   係數項直接反映變數關係。

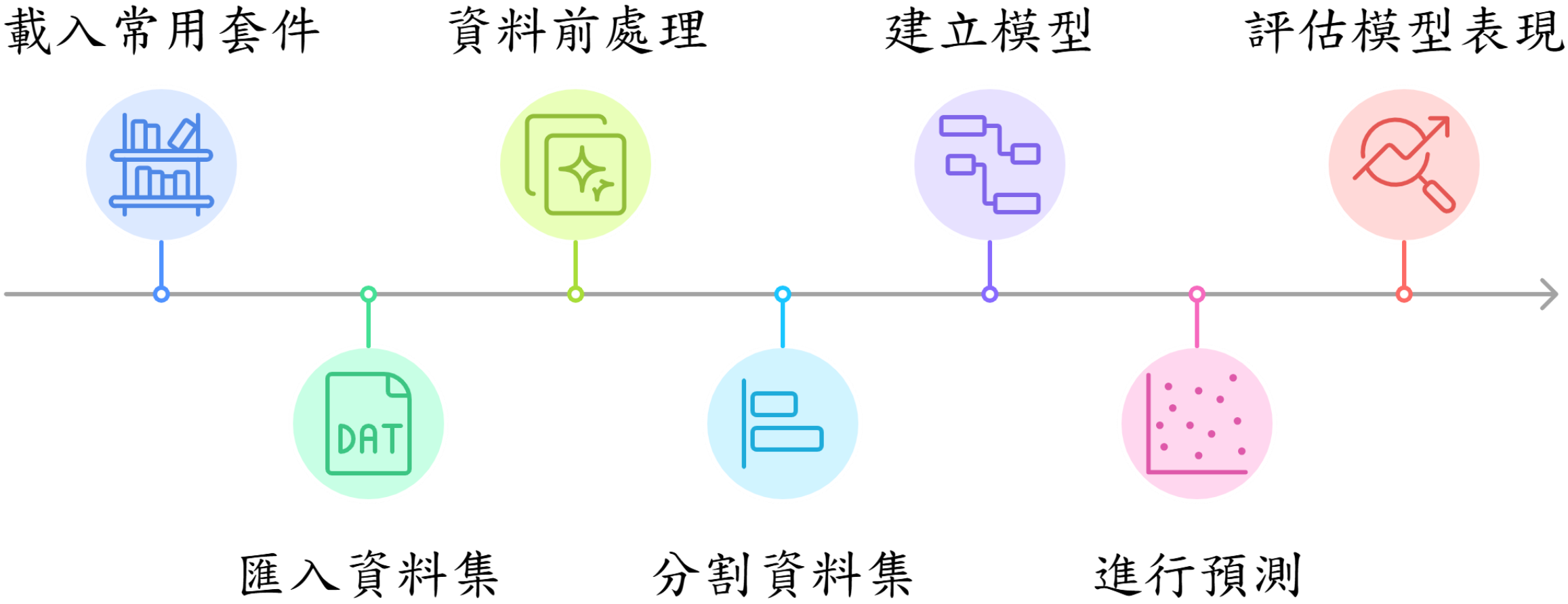
Cons

1. **僅適用於線性關係**
   僅適用於線性關係，非線性資料效果不佳。
2. **不適用於非線性資料**
   無法擬合曲線分佈的非線性資料。
3. **易受共線性影響**
   自變數高度相關時，結果難以解釋。

監督式學習 Python實作流程

- NumPy：用於數值計算。
- Pandas：用於資料處理和分析。
- Matplotlib 和 Seaborn：用於資料視覺化。

- 從 scikit-learn 中匯入對應的模型（例如 LinearRegression）。
- 使用訓練資料來「訓練 (fit)」模型



線性迴歸實作：波士頓房價預測

Step1: 載入常用套件

```
[3] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split # 用於切割資料集
    from sklearn.linear_model import LinearRegression # 線性迴歸模型
    from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score # 評估指標
```

Step2: 匯入資料集

```
[4] from sklearn.datasets import fetch_openml
    boston = fetch_openml(name='boston', version=1, as_frame=True)
    df = boston.frame
```

占住宅用地超過 25,000 平方英尺的比例　　老屋比例　　城鎮師生比

```
df.head()
```

犯罪率 工業用地比率 空汙量 房間數 距市中心 房屋稅 黑人比 貧困指數 房價中位數(目標值)

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6

查看完整資料資訊

在進行任何資料前處理或建模之前，了解資料的基本結構與內容是非常關鍵的第一步。透過df.info()快速查看:

- 欄位名稱與總數
- 每個欄位的資料型態: 本資料集中有兩個類別型欄位需要處理
- 是否有缺失值(null / NaN): 本資料集中無缺失值需要處理
- 總筆數（行數）與記憶體使用量：有助於判斷資料集大小與是否需優化效率

```
print("資料集基本資訊：")
df.info()
```

```
資料集基本資訊:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    category
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    category
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: category(2), float64(12)
memory usage: 49.0 KB
```

資料前處理

- 使用 One-Hot Encoding ，將每個類別值轉換為一個獨立的二元欄位（binary）
- drop_first=True: 將原本欄位刪除，避免變數彼此之間有完全線性關係(多重共線性），在迴歸模型中特別重要，能提高模型的穩定性與解釋能力。

One-Hot Encoding

```
df[['CHAS','RAD']].head(3)
```

	CHAS	RAD
0	0	1
1	0	2
2	0	2

```
#  處理類別型欄位（CHAS,  RAD）
df  =  pd.get_dummies(df,  drop_first=True)

#  顯示轉換後的欄位名稱（確認是否處理成功）
print("\nOne-Hot  Encoding後的相關欄位：")
rad_chas_columns  =  [col  for  col  in  df.columns  if  col.startswith('RAD_')  or  col.startswith('CHAS_')]
df[rad_chas_columns].head(3)
```

```
One-Hot Encoding後的相關欄位：
```

	CHAS_1	RAD_2	RAD_24	RAD_3	RAD_4	RAD_5	RAD_6	RAD_7	RAD_8
0	False	False	False	False	False	False	False	False	False
1	False	True	False	False	False	False	False	False	False
2	False	True	False	False	False	False	False	False	False

分割資料集

- 定義X(特徵)和Y(目標)，本資料集中 房價中位數(MEDV)為預測目標，須從訓練資料集中移除

```
Y = df['MEDV']  # 目標欄位(房價中位數)
X = df.drop('MEDV', axis=1)  # 移除目標欄位，其餘為特徵
```

- 切割資料集
  - test_size=0.3 表示 30% 用於測試，70% 用於訓練
  - random_state=42 確保每次執行結果一致

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)

print(f"訓練集特徵數量: {X_train.shape}")
print(f"測試集特徵數量: {X_test.shape}")
```

```
訓練集特徵數量: (354, 20)
測試集特徵數量: (152, 20)
```

70%資料列 * 20個特徵

30%資料列 * 20個特徵

建立模型

- scikit-learn（常縮寫為 sklearn）是 Python 中最常用的機器學習套件之一，提供了豐富且簡單易用的工具來完成線性模型的建立、訓練與評估。
- 先從 sklearn.linear_model 模組中匯入 LinearRegression這個用來建立線性迴歸模型的工具

```
from sklearn.linear_model import LinearRegression # 線性迴歸模型
```

```
model = LinearRegression()    # 建立一個線性迴歸模型物件
model.fit(X_train, Y_train)  # 使用訓練資料 (X_train, Y_train) 來訓練模型
print("模型訓練完成！")
```

```
模型訓練完成！
```

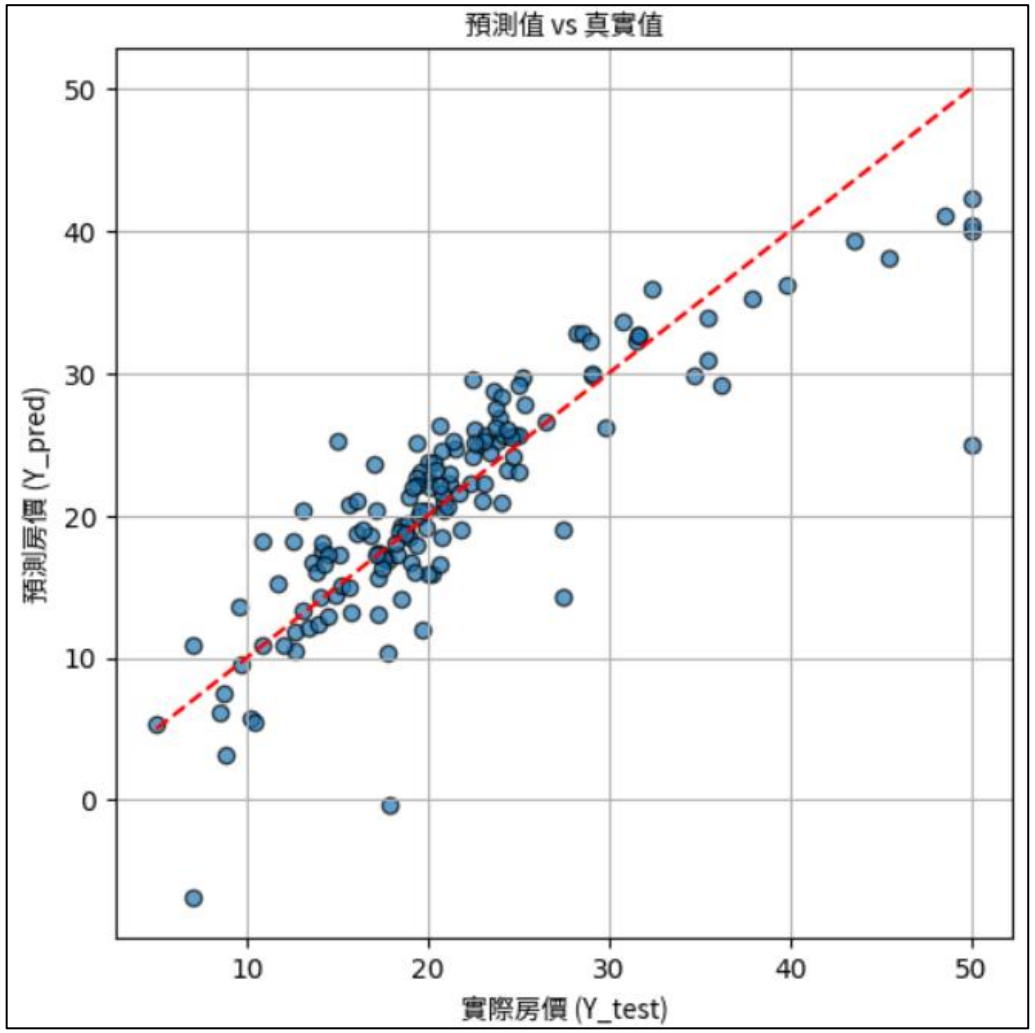
- 這樣model就可以根據輸入資料學習出一條最佳的直線，用來預測目標(本例為房價中位數)

進行預測

```
#  使用訓練好的模型對測試資料  (X_test)  進行預測
Y_pred  =  model.predict(X_test)

#  查看部分預測結果與真實值比較
results_df  =  pd.DataFrame({
        '真實房價  (Y_test)':  Y_test,
        '預測房價  (Y_pred)':  Y_pred})
print(results_df.head())  #  列印前幾筆比較結果
```

	真實房價 (Y_test)	預測房價 (Y_pred)
173	23.6	28.707646
274	32.4	35.884340
491	13.6	16.727016
72	22.8	24.967590
452	16.1	18.785488



評估模型表現

```python
# 計算 R 平方值
r2 = model.score(X_test, Y_test) # 可以直接用 model.score 評估 R 平方
print(f"\n模型的 R 平方 (R-squared): {r2:.4f}")

# 計算均方誤差 (MSE)
mse = mean_squared_error(Y_test, Y_pred) # 從 sklearn.metrics 匯入
print(f"模型的均方誤差 (MSE): {mse:.4f}")

# 計算均方根誤差 (RMSE)
rmse = np.sqrt(mse) # RMSE 是 MSE 的平方根
print(f"模型的均方根誤差 (RMSE): {rmse:.4f}")

# 計算平均絕對誤差 (MAE)
mae = mean_absolute_error(Y_test, Y_pred) # 從 sklearn.metrics 匯入
print(f"模型的平均絕對誤差 (MAE): {mae:.4f}")
```

```
模型的 R 平方 (R-squared): 0.7147
模型的均方誤差 (MSE): 21.2598
模型的均方根誤差 (RMSE): 4.6108
模型的平均絕對誤差 (MAE): 3.1741
```

總結

- 線性迴歸是一種預測連續型目標變數的監督式學習方法
- 自變數（Features）預測 應變數（Target），但不代表因果關係
- 目標是建立一條最佳擬合線，最小化殘差平方和（SSE）
- Python 中使用 LinearRegression 模型進行訓練與預測
- 模型好壞可用 R^2、MAE、MSE、RMSE 等指標評估
- 單一特徵模型雖簡單，但表現常不如全特徵模型
- 若訓練集表現好但測試集差，可能有 過擬合（Overfitting）