

W3. 資料前處理



資訊科學系 助理教授

楊景元

篤行樓337室

(02)2732-1104 #55337

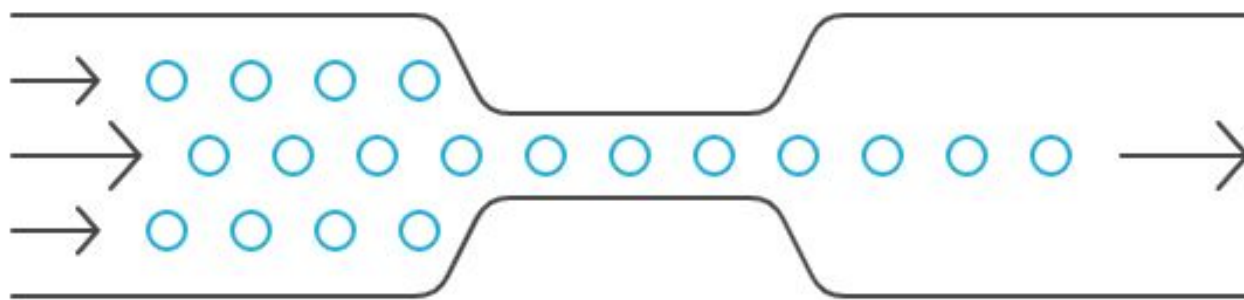
cyyang@mail.ntue.edu.tw

機器學習中的資料前處理

資料前處理是機器學習流程中至關重要的一步，涉及將原始數據轉換為適合模型訓練的格式。這一過程直接影響模型的效能和準確性，因為不良的數據質量會導致模型的預測結果不準確。

資料前處理的定義

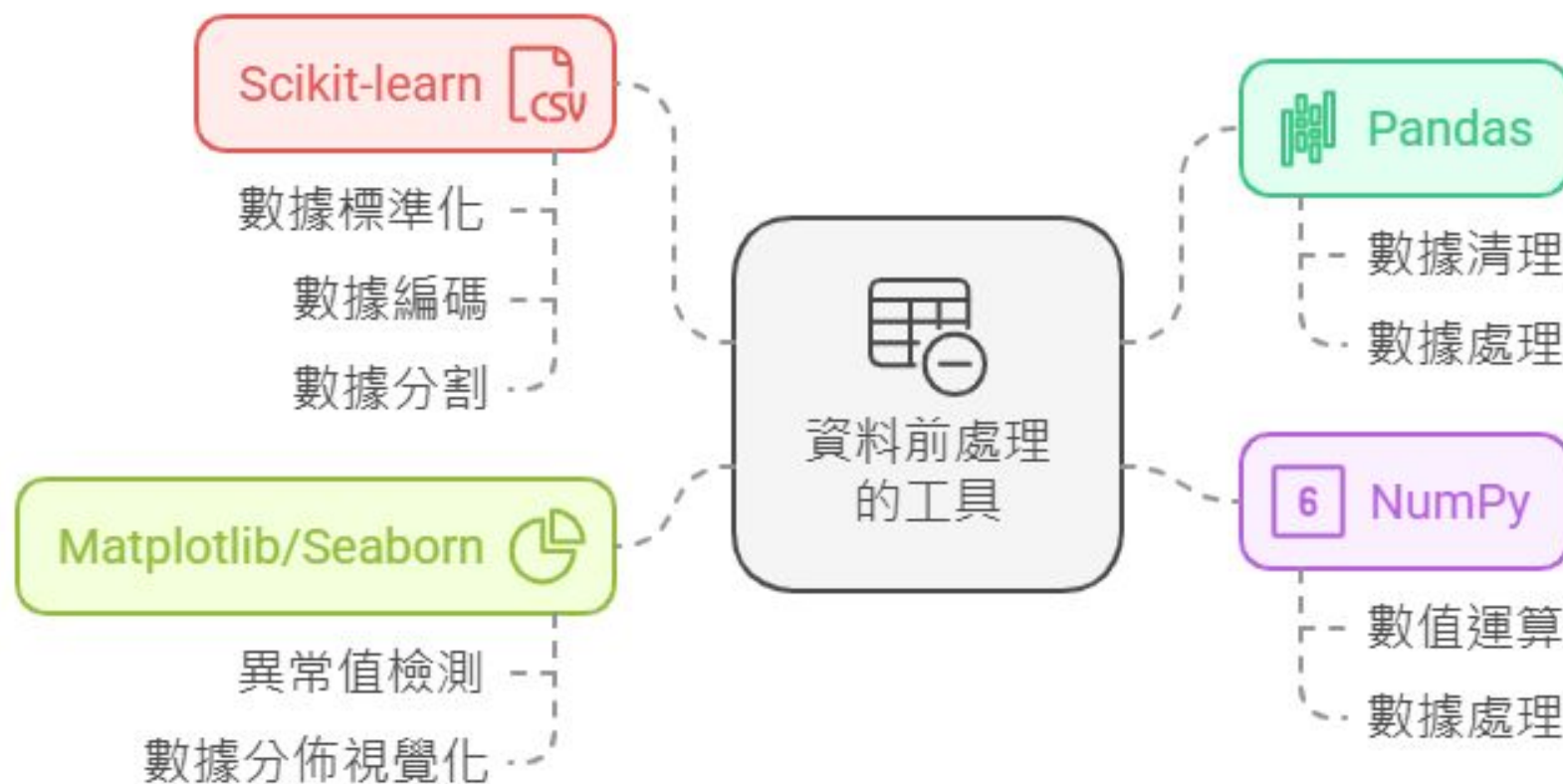
將原始資料進行清理、轉換和整理，以提高數據質量和一致性，減少不確定性和噪音。主要目的為確保數據適合進行後續的分析和建模工作。



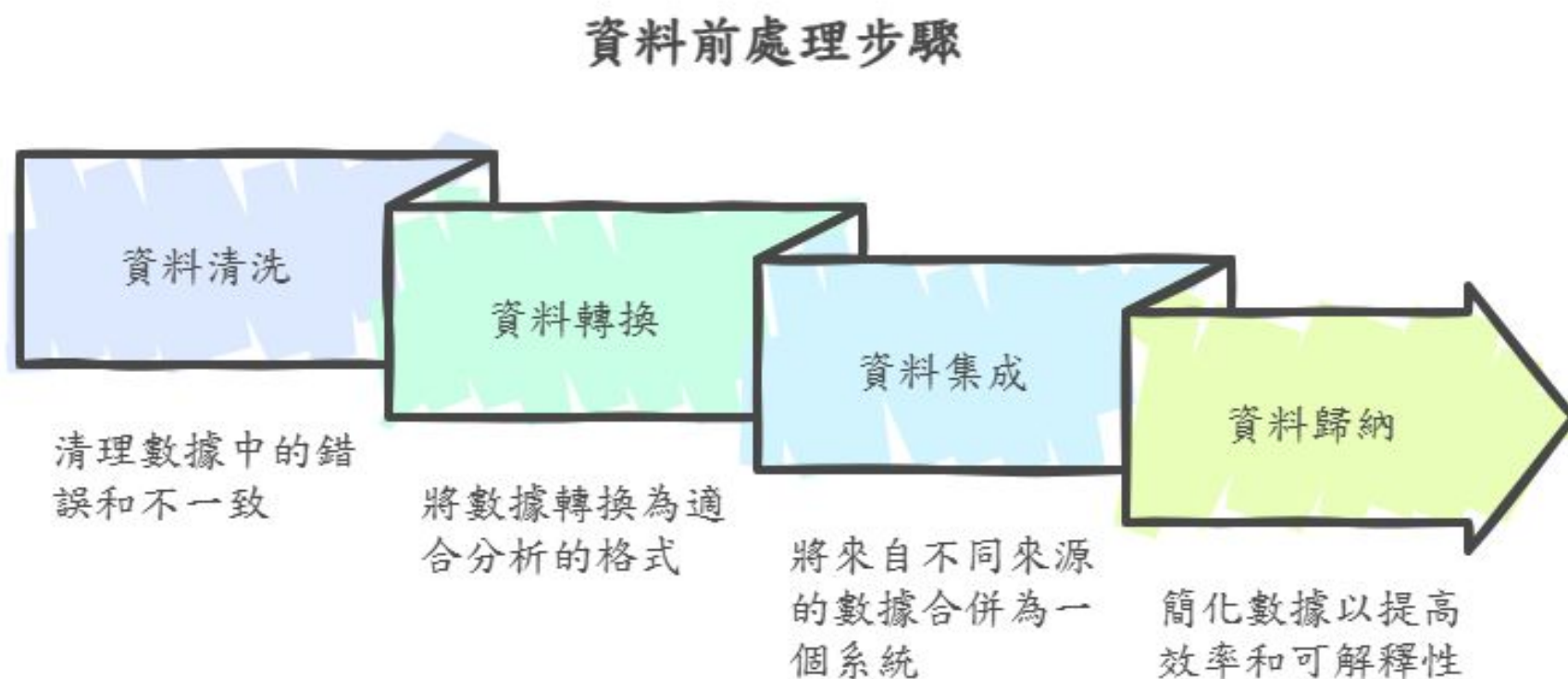
資料前處理的重要性

- 影響模型效能：高品質的數據能夠提高模型的準確度和泛化能力。
- 解決常見問題：如缺失值、異常值、類別不平衡等。

資料前處理的常見工具



資料前處理的主要步驟

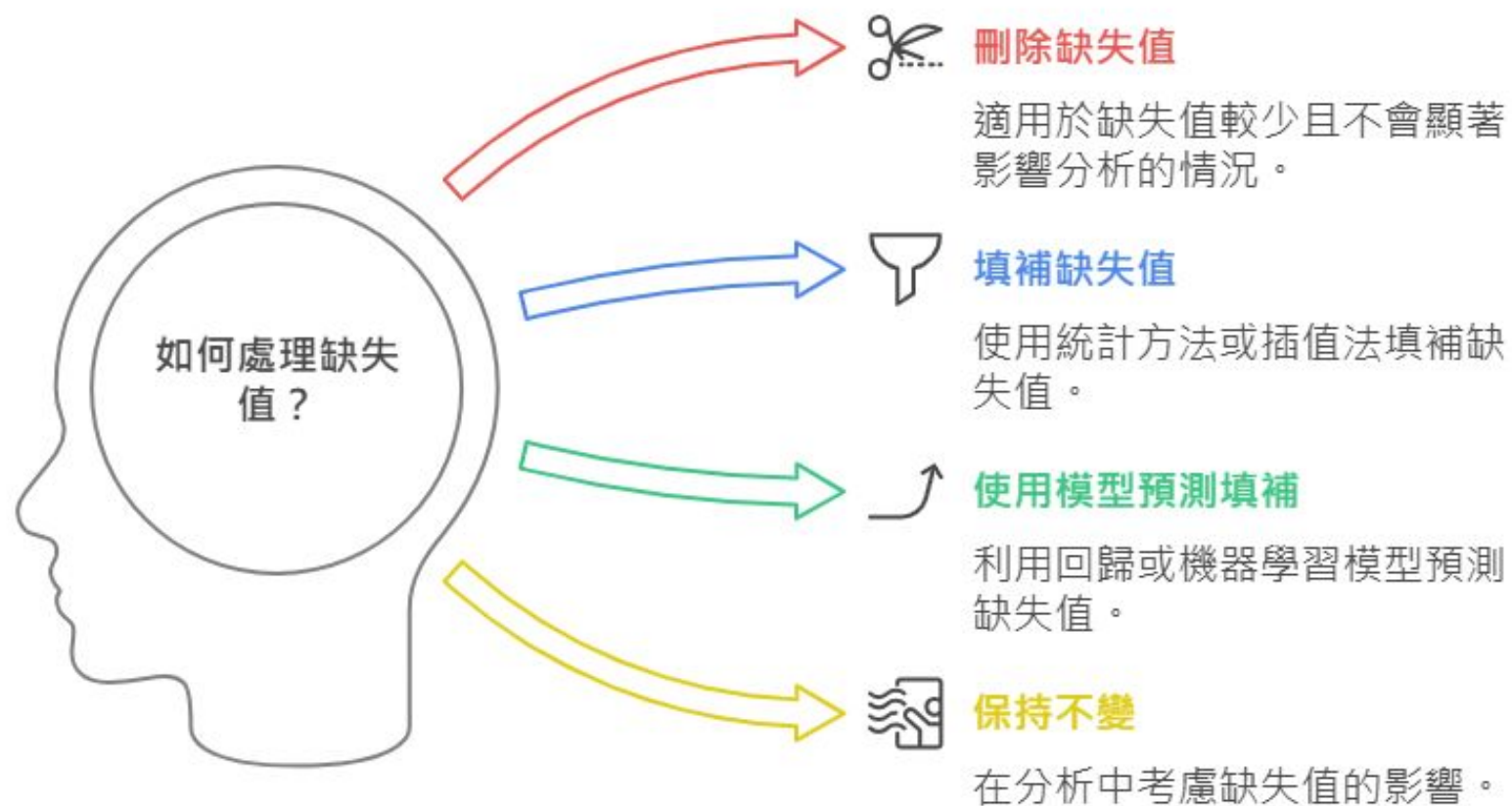


資料清洗 (Data Cleaning)

這一步驟主要是清理數據中的錯誤、缺失值和不一致性。常見的操作包括填補缺失值、刪除重複或無效的記錄,以及修正錯誤的數值或標籤。

- 處理缺失值:
 - 檢視缺失值:使用 `isnull()` 方法。
 - 刪除缺失值:使用 `dropna()` 方法。
 - 填補缺失值:使用均值、中位數或其他方法填補。
- 去除重複資料:
 - 使用 `drop_duplicates()` 方法刪除重複行。
- 修正錯誤數據:
 - 檢查並修正不一致性和錯誤標籤。

資料清洗 (Data Cleaning): 處理缺失值



資料清洗 (Data Cleaning): 處理缺失值

1. 刪除缺失值

- **適用情況** : 當缺失值的數量較少,且不會對分析結果造成重大影響時,可以考慮刪除含有缺失值的行或列。
- **優點** : 簡單直接,能夠保證分析數據的完整性。
- **缺點** : 可能導致信息損失,尤其是當缺失不是隨機發生時,刪除可能會引入偏差。

2. 填補缺失值

- **使用統計量填補** :
 - **均值填補** : 用該列的平均值填補缺失值。
 - **中位數填補** : 用該列的中位數填補,對於有極端值的數據集更為穩健。
 - **眾數填補** : 對於類別型變數,可以使用該列的眾數進行填補。
- **插值法** :
 - 適用於連續型數據,通過已知數據點的信息來估算未知點的值。常見方法包括線性插值和多項式插值。

資料清洗 (Data Cleaning): 處理缺失值

3. 使用模型預測填補

- **基於回歸模型** : 利用其他特徵建立回歸模型來預測缺失值。
- **機器學習算法** : 使用如 KNN (K-Nearest Neighbors) 等算法, 根據相似記錄來預測缺失值。

4. 保持不變

- 在某些情況下, 如果資料收集過程中規則上允許存在空值, 可以選擇不處理缺失值。在後續分析中考慮缺失值的影響, 或在報告結果時說明缺失值的存在和處理方式。

5. 標記缺失值

- 對於某些分析, 可以將缺失值標記為特定標籤 (如 "Missing"), 這樣在後續分析中可以考慮到這些缺失的情況。

資料清洗 (Data Cleaning): 練習

```
import pandas as pd
import numpy as np

# 原始數據
data = {
    "Name": ["Alice", "Bob", "Charlie", "David", "Alice", np.nan],
    "Age": [25, np.nan, 35, 40, 25, 30],
    "City": ["New York", "Los Angeles", "Chicago", "Houston", "New York", "Houston"],
    "Salary": [50000, 60000, 70000, np.nan, 50000, 80000]
}

df = pd.DataFrame(data)
print("原始數據:\n", df)
```

資料清洗 (Data Cleaning): 刪除缺失值

1. 檢視數據中的缺失值, 並計算每列缺失值的數量

```
missing_values = df.isnull().sum()  
print("\n每列缺失值的數量:\n", missing_values)
```

2. 刪除包含缺失值的行

```
df_dropped_na = df.dropna()  
print("\n刪除缺失值後的數據:\n", df_dropped_na)
```

資料清洗 (Data Cleaning): 填補缺失值

3. 填補缺失值

年齡 (Age) 列填補為平均值

薪水 (Salary) 列填補為中位數

```
df["Age"].fillna(df["Age"].mean(), inplace=True)
```

```
df["Salary"].fillna(df["Salary"].median(), inplace=True)
```

```
print("\n填補缺失值後的數據:\n", df)
```

資料清洗 (Data Cleaning): 去除重複資料

4. 刪除數據中的重複行

```
df_no_duplicates = df.drop_duplicates()  
print("\n刪除重複行後的數據:\n", df_no_duplicates)
```

資料清洗 (Data Cleaning): 修正錯誤數據

5. 修正錯誤數據

假設 Name 列中應該唯一, 修正重複名字

```
df["Name"] = df["Name"].fillna("Unknown") # 填補缺失名稱
```

```
df.loc[df["Name"].duplicated(), "Name"] += "_Duplicate" # 添加後綴區分重複名稱
```

6. 假設 Salary 的範圍應該在 30000 到 100000 之間, 修正不符合條件的數據

```
df.loc[df["Salary"] < 30000, "Salary"] = 30000
```

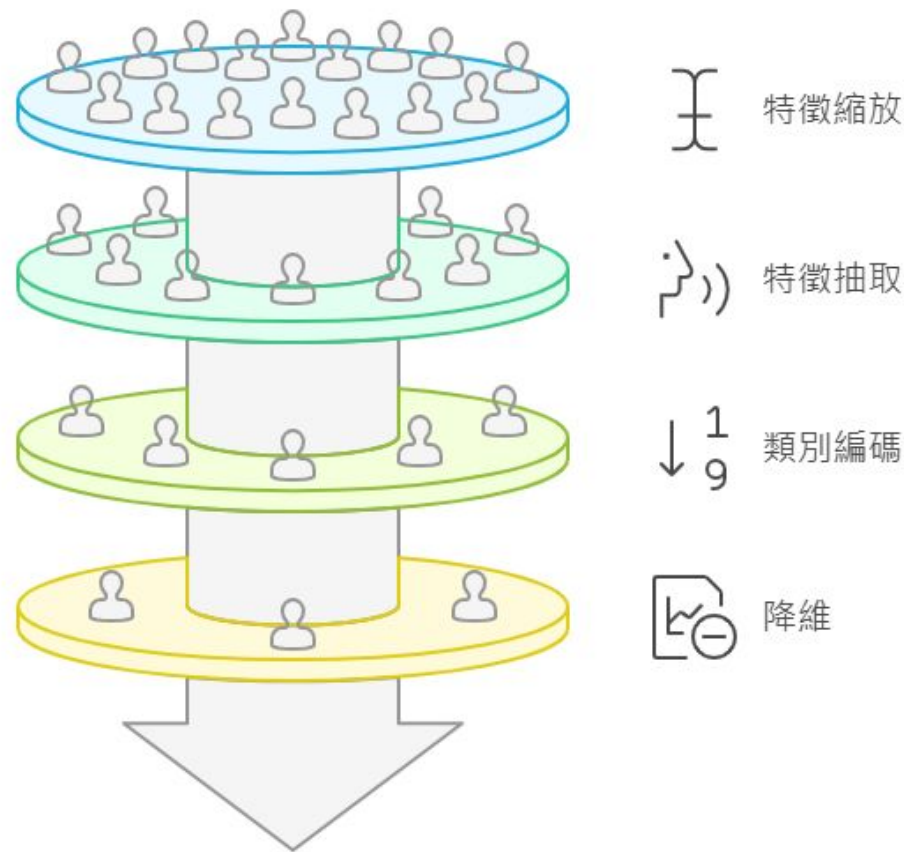
```
df.loc[df["Salary"] > 100000, "Salary"] = 100000
```

```
print("\n修正錯誤數據後的數據:\n", df)
```

資料轉換 (Data Transformation)

資料轉換是對數據進行轉換,以使其更適合進行分析。這包括

- 特徵縮放(如標準化或正規化)
- 特徵抽取(從原始數據中提取有意義的特徵)
- 類別編碼(將類別型數據轉換為數值型)
- 降維(例如主成分分析)。



資料轉換 (Data Transformation)

1. 特徵縮放 (Feature Scaling):

- **目的:** 縮小特徵之間的範圍差異, 避免某些特徵因量級較大而主導模型。
- **方法:**
 - **標準化 (Standardization):**
 - 將數據縮放到均值為 0, 標準差為 1 的分佈。
 - 常用於需保持數據分佈的模型, 如線性回歸、支持向量機。
 - **正規化 (Normalization):**
 - 將數據縮放到 $[0, 1]$ 範圍內。
 - 常用於需簡化計算的場景, 如神經網絡。

資料轉換 (Data Transformation): 特徵縮放

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas as pd
```

```
# 數據
```

```
data = pd.DataFrame({
    "Age": [20, 25, 30, 35, 40],
    "Income": [20000, 30000, 40000, 50000, 60000]
})
```

```
print("原始數據:\n", data)
```

```
# a. 標準化
```

```
scaler_standard = StandardScaler()
data_standardized = scaler_standard.fit_transform(data)
print("標準化後的數據:\n", data_standardized)
```

```
# b. 正規化
```

```
scaler_minmax = MinMaxScaler()
data_normalized = scaler_minmax.fit_transform(data)
print("正規化後的數據:\n", data_normalized)
```

資料轉換 (Data Transformation)

2. 特徵抽取 (Feature Extraction):

- **目的:** 從原始數據中提取對模型有用的信息, 減少噪聲或冗餘。
- **應用:**
 - 新建衍生特徵(例如: 收入與年齡的比值)。
 - 在圖像處理中提取邊緣、顏色等特徵。

資料轉換 (Data Transformation): 特徵抽取

```
import pandas as pd
```

```
# 數據
```

```
data = pd.DataFrame({  
    "Age": [20, 25, 30, 35, 40],  
    "Income": [20000, 30000, 40000, 50000, 60000]  
})
```

```
# 新建特徵
```

```
data["Income_per_Age"] = data["Income"] / data["Age"]  
print("新增特徵後的數據:\n", data)
```

資料轉換 (Data Transformation)

3. 類別編碼 (Categorical Encoding):

- **目的:** 將類別型數據轉換為模型能理解的數值型數據。
- **方法:**
 - **標籤編碼 (Label Encoding):** 將每個類別映射為數字。
 - **獨熱編碼 (One-Hot Encoding):** 為每個類別創建一個二進制特徵列。

資料轉換 (Data Transformation)

1. 標籤編碼 (Label Encoding)

- **定義** : 將每個類別值映射到一個唯一的整數。例如,對於類別 "紅"、"綠" 和 "藍",可以分別映射為 0、1 和 2。
- **適用情況** : 適合有序類別 (如等級) 或類別數量較少的情況。
- **優點** : 簡單易用,能夠保留類別之間的順序關係。
- **缺點** : 對於無序類別,可能會引入不必要的順序性,影響模型效果。

2. 獨熱編碼 (One-Hot Encoding)

- **定義** : 為每個類別創建一個新的二進位特徵,對於每一個觀測值,僅有一個二進位特徵為 1,其餘為 0。
- **適用情況** : 適合無序類別或類別數量較少的情況。
- **優點** : 消除了類別之間的順序關係,避免了模型誤解。
- **缺點** : 當類別數量非常多時,會導致維度爆炸,增加計算成本。

資料轉換 (Data Transformation): 類別編碼

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# 數據
data = pd.DataFrame({
    "City": ["New York", "Los Angeles", "Chicago", "Houston", "Phoenix"]
})

# a. 標籤編碼
label_encoder = LabelEncoder()
data["City_LabelEncoded"] = label_encoder.fit_transform(data["City"])
print("標籤編碼後的數據:\n", data)

# b. 獨熱編碼
data_onehot = pd.get_dummies(data, columns=["City"])
print("獨熱編碼後的數據:\n", data_onehot)
```

資料轉換 (Data Transformation)

4. 降維 (Dimensionality Reduction):

- **目的:** 簡化數據結構, 降低特徵數量, 提升模型計算效率。
- **方法:**
 - **主成分分析 (PCA):** 通過線性變換找到數據的主要方向, 降低維度。
 - **線性判別分析 (LDA):** 使用標籤信息來找到最區分類別的方向。

資料轉換 (Data Transformation): 降維

```
from sklearn.decomposition import PCA
```

```
# 數據
```

```
data = pd.DataFrame({  
    "Feature1": [2, 4, 6, 8, 10],  
    "Feature2": [1, 3, 5, 7, 9],  
    "Feature3": [10, 20, 30, 40, 50],  
    "Feature4": [5, 10, 15, 20, 25]  
})
```

```
# PCA 降維到 2 個主要成分
```

```
pca = PCA(n_components=2)  
data_pca = pca.fit_transform(data)
```

```
# 結果
```

```
print("原始數據:\n", data)  
print("\n降維後的數據:\n", data_pca)
```

資料集成 (Data Integration)

此步驟涉及將來自不同來源、不同格式與結構的數據整合到統一的數據資料庫中。這需要確保數據的一致性和可用性,並解決格式不一致的問題。

- **整合不同來源的數據：**
 - 將來自不同來源、不同格式的數據整合到統一的資料庫中。
- **確保一致性：**
 - 處理不同系統數據格式不一致的問題, 確保所有數據具有相同的結構。

資料集成 (Data Integration)

數據整合挑戰的解決方案



資料集成 (Data Integration)

1. 制定統一的接口規範

- 定義通用數據交換標準: 確保所有參與系統遵循相同的規則和標準進行信息交流, 這可以減少格式不一致的問題。
- 建立數據模型: 制定一套通用的數據模型, 以簡化不同系統之間的信息轉換。

2. 引入適配器或轉換器技術

- 開發適配器: 針對特定需求開發適配器(Adapter)或轉換器(Transformer), 將不同類型和不兼容的結構化數據進行相互映射, 消除格式上的障礙。
- 自動化轉換工具: 使用自動化工具來處理不同格式之間的轉換, 例如將 CSV 格式轉換為 JSON 格式。

3. 構建統一的數據平台

- 建立數據湖或數據倉庫: 搭建一個集成各類數據源的統一平台, 確保各數據集的一致性與可比性。
- 實時監控和質量控制: 提供實時監控功能以確保數據質量和完整性, 並對跨庫/跨部門共享進行管理控制。

資料集成 (Data Integration)

4. 使用資料字典和映射表

- 建立資料字典：對不同數據源中的字段進行標準化命名和定義，消除混淆和歧義。
- 字段映射規則：制定清晰的字段映射規則，確保源數據庫的字段正確對應到目標數據庫的字段。

5. 數據格式轉換

- 日期格式轉換：統一日期格式，例如將 YYYY-MM-DD 轉換為 DD/MM/YYYY。
- 單位轉換：在不同領域中，確保將數值轉換為相同單位，以保持數據的一致性。

6. 利用人工智能進行異常檢測

- 自動識別異常值：通過訓練 AI 模型來自動檢測因格式不匹配而導致的潛在風險項和異常點，這不僅降低了人力成本，還提高了整體分析效率。

資料集成 (Data Integration): 字典和映射表

```
import pandas as pd
```

```
# 原始數據
```

```
data = {  
    "Emp_ID": [101, 102, 103],  
    "F_Name": ["Alice", "Bob", "Charlie"],  
    "Dept": ["HR", "Engineering", "Finance"]  
}
```

```
df = pd.DataFrame(data)
```

```
print("原始數據:\n", df)
```

資料集成 (Data Integration): 字典和映射表

a. 字段名稱標準化

```
df.rename(columns={"Emp_ID": "Employee_ID", "F_Name": "First_Name", "Dept": "Department"},  
inplace=True)
```

```
print("\n字段名稱標準化後:\n", df)
```

b. 部門名稱映射

```
department_mapping = {  
    "HR": "Human Resources",  
    "Engineering": "Engineering Department",  
    "Finance": "Financial Department"  
}
```

```
df["Department"] = df["Department"].map(department_mapping)
```

```
print("\n字段值映射後:\n", df)
```

資料集成 (Data Integration): 數據格式轉換

原始數據

```
data = {  
    "Date": ["2023-01-01", "2023-02-15", "2023-03-20"],  
    "Weight (kg)": [70, 65, 80]  
}  
df = pd.DataFrame(data)  
print("原始數據:\n", df)
```

a. 日期格式轉換

```
df["Date"] = pd.to_datetime(df["Date"]).dt.strftime("%d/%m/%Y")  
print("\n日期格式轉換後:\n", df)
```

b. 單位轉換

```
df["Weight (lbs)"] = df["Weight (kg)"] * 2.20462  
print("\n單位轉換後:\n", df)
```

資料歸納 (Data Reduction)

最後,資料歸納旨在減少數據的維度、大小或複雜度,同時保留主要特徵和資訊。這有助於降低計算成本、減少噪音影響,提高模型效率和可解釋性。

- 減少數據維度:
 - 使用主成分分析(PCA)等技術減少特徵空間的維度。
- 降低計算成本:
 - 減少數據量以提高運算效率,降低噪音影響。

資料歸納 (Data Reduction): 減少數據量

```
import pandas as pd
```

```
# 數據
```

```
data = pd.DataFrame({  
    "Feature1": [2, 4, 6, 8, 10],  
    "Feature2": [1, 3, 5, 7, 9],  
    "Feature3": [10, 20, 30, 40, 50],  
    "Feature4": [5, 10, 15, 20, 25],  
    "Target": [1, 0, 1, 0, 1]  
})
```

資料歸納 (Data Reduction): 減少數據量

a. 隨機抽樣 60% 的數據

```
sampled_data = data.sample(frac=0.6, random_state=42)  
print("\n隨機抽樣後的數據:\n", sampled_data)
```

b. 根據相關性矩陣選擇相關性較高的特徵

```
correlation_matrix = data.corr()  
print("\n相關性矩陣:\n", correlation_matrix)
```

選擇兩個相關性較高的特徵(假設我們選擇 Feature1 和 Feature3)

```
selected_features = data[["Feature1", "Feature3", "Target"]]  
print("\n選擇的特徵作為輸入:\n", selected_features)
```

W3. 實作練習

資料清洗 (Data Cleaning)

使用以下數據：

```
data = {  
    "Name": ["Alice", "Bob", "Charlie", "David", "Eve", "Frank"],  
    "Age": [25, None, 35, 40, -5, 30],  
    "City": ["New York", "Los Angeles", None, "Houston", "Chicago", "New York"],  
    "Salary": [50000, 60000, None, 80000, 90000, -10000]  
}
```

a. 填補缺失值：

- 使用中位數填補 Age 列。

- 將 City 列的缺失值填補為 "Unknown"。

b. 修正錯誤數據：

- 將 Age 中小於 0 的值設置為 None，然後再用中位數填補。

- 將 Salary 中小於 0 的值設置為 0。

c. 刪除不完整的行：

- 刪除 Age 和 City 中同時缺失值的行。

資料轉換 (Data Transformation)

使用以下數據：

```
data = {  
    "Feature1": [10, 20, 30, 40, 50],  
    "Feature2": [1, 2, 3, 4, 5],  
    "Category": ["A", "B", "A", "B", "C"]  
}
```

a. 將 Feature1 和 Feature2 進行標準化 (均值為 0, 標準差為 1)。

b. 將 Category 列進行類別編碼, 使用 One-Hot Encoding。

c. 使用 PCA 將 Feature1 和 Feature2 降維到一個主成分。

資料集成 (Data Integration)

使用以下數據：

```
data = {  
    "ID": [1, 2, 3],  
    "Emp_Name": ["John", "Jane", "Doe"],  
    "Dept_Code": ["IT", "HR", "FIN"]  
}
```

a. 將字段名稱標準化，將以下字段進行重命名：

```
# - "ID" -> "Employee_ID"  
# - "Emp_Name" -> "Employee_Name"  
# - "Dept_Code" -> "Department"
```

使用以下數據：

```
data = {  
    "Date_Joined": ["2022-01-01", "2022-05-15", "2022-09-20"],  
    "Height (cm)": [170, 165, 180]  
}
```

a. 將 Date_Joined 列的日期格式從 YYYY-MM-DD 轉換為 MM/DD/YYYY。

b. 將 Height (cm) 的單位從公分轉換為英吋 (inch)，並新增一列 "Height (inch)"。

提示：1 cm = 0.393701 inch。

資料歸納 (Data Reduction)

使用以下數據：

```
data = {  
    "Feature1": [1, 2, 3, 4, 5],  
    "Feature2": [2, 4, 6, 8, 10],  
    "Feature3": [5, 10, 15, 20, 25]  
}
```

a. 將數據按 80% 抽樣，減少數據量。

b. 選擇兩個相關性較高的特徵。