



BỘ GIÁO DỤC ĐÀO TẠO **TRƯỜNG ĐẠI HỌC SÀI GÒN**

Khoa Công nghệ Thông tin

PHÂN TÍCH DỮ LIỆU - BUỔI 02
Tiền Xử Lý Dữ Liệu

CBGD: Phan Thành Huấn

✉ ⚡ : pthuan112358@gmail.com

📞 : 097 882 8842

Nội dung

1. Tổng quan về giai đoạn tiền xử lý dữ liệu
2. Tóm tắt mô tả về dữ liệu
3. Làm sạch dữ liệu
4. Tích hợp dữ liệu
5. Biến đổi dữ liệu
6. Thu giảm dữ liệu
7. Rời rạc hóa dữ liệu
8. Tạo cây phân cấp ý niệm
9. Biểu diễn dữ liệu

1-Tổng quan về giai đoạn tiền xử lý dữ liệu

(Data Preprocessing)

Giai đoạn tiền xử lý dữ liệu là một bước quan trọng trong quá trình PTDL - thu thập, chuẩn hóa, lọc, biến đổi và tạo dữ liệu để chuẩn bị cho các bước xử lý và phân tích tiếp theo.

1. **Thu thập dữ liệu:** Thu thập dữ liệu từ các nguồn khác nhau như cơ sở dữ liệu, file văn bản, các hệ thống đo đạc, các trang web, và các nguồn dữ liệu công cộng;
2. **Xóa dữ liệu không hợp lệ:** Loại bỏ các dữ liệu không hợp lệ, thiếu sót hoặc trùng lặp từ tập dữ liệu;

1-Tổng quan về giai đoạn tiền xử lý dữ liệu

(Data Preprocessing)

3. **Chuẩn hóa dữ liệu:** Đảm bảo rằng dữ liệu được chuẩn hóa theo đúng định dạng, đơn vị và quy tắc đã được xác định trước. Ví dụ: chuẩn hóa định dạng ngày tháng, đơn vị đo lường, viết hoa chữ cái đầu tiên trong tên, .v.v.;
4. **Lọc dữ liệu:** Lựa chọn và loại bỏ các dữ liệu không cần thiết hoặc không quan trọng từ tập dữ liệu. Điều này giúp giảm kích thước tập dữ liệu và tăng tốc độ xử lý;

1-Tổng quan về giai đoạn tiền xử lý dữ liệu

(Data Preprocessing)

5. **Xử lý dữ liệu thiếu:** Xử lý các giá trị thiếu trong tập dữ liệu bằng các phương pháp như điền giá trị trung bình, giá trị trung vị hoặc giá trị phổ biến nhất vào chỗ trống;
6. **Biến đổi dữ liệu:** Chuyển đổi dữ liệu từ dạng gốc sang dạng phù hợp cho việc phân tích. Ví dụ: chuyển đổi biến định danh thành biến số, tạo biến mới từ các biến hiện có, .v.v.;

1-Tổng quan về giai đoạn tiền xử lý dữ liệu

(Data Preprocessing)

7. Tạo dữ liệu mới: Tạo ra các biến mới từ dữ liệu hiện có bằng cách tính toán, kết hợp hoặc phân loại. Điều này có thể giúp tăng cường khả năng dự đoán và hiểu biết từ dữ liệu.

Giai đoạn tiền xử lý dữ liệu đóng vai trò quan trọng trong việc đảm bảo chất lượng và sẵn sàng của dữ liệu cho các bước xử lý và phân tích tiếp theo \Rightarrow cải thiện độ tin cậy và hiệu quả của kết quả phân tích dữ liệu.

1-Tổng quan về giai đoạn tiền xử lý dữ liệu

(Data Preprocessing)

Ví dụ: Loại bỏ danh sách học sinh không hợp lệ.

```
# Danh sách các học sinh
students = [
    {'name': 'John', 'age': 15, 'grade': 9},
    {'name': 'Alice', 'age': 17, 'grade': 11},
    {'name': 'Bob', 'age': 14, 'grade': 8},
    {'name': 'Charlie', 'age': 16, 'grade': 10},
    {'name': 'Dave', 'age': 18, 'grade': 12},
    {'name': 'Eve', 'age': 13, 'grade': 7},
    {'name': 'Frank', 'age': 15, 'grade': 9},
]

# Hàm kiểm tra dữ liệu hợp lệ
def is_valid_student(student):
    if not isinstance(student, dict):
        return False
    if 'name' not in student or 'age' not in student or 'grade' not in student:
        return False
    if not isinstance(student['name'], str) or not isinstance(student['age'], int) or not isinstance(student['grade'], int):
        return False
    if student['age'] < 13 or student['age'] > 18 or student['grade'] < 7 or student['grade'] > 12:
        return False
    return True

# Xóa dữ liệu không hợp lệ
valid_students = [student for student in students if is_valid_student(student)]

# In danh sách học sinh hợp lệ
for student in valid_students:
    print(student)
```

1-Tổng quan về giai đoạn tiền xử lý dữ liệu

(Data Preprocessing)

Ví dụ: Chuẩn hóa danh sách HS – Chữ cái đầu Tên; tuổi, điểm là chuỗi.

```
# Danh sách các học sinh
students = [
    {'name': 'John', 'age': 15, 'grade': 9},
    {'name': 'Alice', 'age': 17, 'grade': 11},
    {'name': 'Bob', 'age': 14, 'grade': 8},
    {'name': 'Charlie', 'age': 16, 'grade': 10},
    {'name': 'Dave', 'age': 18, 'grade': 12},
    {'name': 'Eve', 'age': 13, 'grade': 7},
    {'name': 'Frank', 'age': 15, 'grade': 9},
]

# Hàm chuẩn hóa dữ liệu
def normalize_student(student):
    student['name'] = student['name'].capitalize()
    student['age'] = str(student['age'])
    student['grade'] = str(student['grade'])
    return student

# Chuẩn hóa dữ liệu
normalized_students = [normalize_student(student) for student in students]

# In danh sách học sinh đã chuẩn hóa
for student in normalized_students:
    print(student)
```


2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Trong giai đoạn TXLDL, quá trình tóm tắt và mô tả dữ liệu là một bước quan trọng để hiểu rõ về tính chất và cấu trúc của dữ liệu. Quá trình này giúp xác định các thông tin cần thiết và tạo sự nhất quán trong dữ liệu trước khi tiếp tục với các bước xử lý và phân tích.

- 1. Kiểm tra thông tin cơ bản:** Xem xét các thuộc tính cơ bản của dữ liệu như tên biến, loại dữ liệu, kích thước, đơn vị đo lường, phạm vi giá trị, vv. Điều này giúp xác định các thuộc tính và đặc điểm quan trọng của dữ liệu;

2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

2. **Thống kê mô tả:** Tính toán các thống kê mô tả như giá trị trung bình, độ lệch chuẩn, phương sai, phân vị, tỷ lệ phần trăm, vv. Điều này giúp hiểu rõ hơn về phân phối và biến động của dữ liệu;
3. **Khám phá dữ liệu:** Trực quan hóa dữ liệu bằng các biểu đồ, đồ thị hoặc biểu đồ hộp. Điều này giúp phát hiện các mẫu, xu hướng, biên độ và các giá trị ngoại lệ trong dữ liệu;

2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

4. **Xác định dữ liệu thiếu:** Kiểm tra xem dữ liệu có giá trị thiếu không và xác định tỷ lệ dữ liệu thiếu trong mỗi biến. Điều này giúp quyết định cách xử lý dữ liệu thiếu trong các bước tiếp theo;
5. **Xác định và loại bỏ nhiễu:** Phát hiện và xử lý các giá trị nhiễu hoặc không hợp lệ trong dữ liệu. Điều này giúp đảm bảo tính chính xác và đáng tin cậy của dữ liệu;

2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

6. Tạo biến mới: Dựa trên các phân tích và hiểu biết về dữ liệu, tạo ra các biến mới từ dữ liệu hiện có. Điều này có thể giúp mở rộng khả năng phân tích và khám phá dữ liệu.

Quá trình tóm tắt và mô tả dữ liệu trong giai đoạn TXLDL giúp xác định và hiểu rõ hơn về tính chất, đặc điểm và cấu trúc của dữ liệu ⇒

Cơ sở để tiếp tục với các bước xử lý và PTDL tiếp theo.

2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Thống kê mô tả dữ liệu


```
import numpy as np
import pandas as pd

# Tạo dữ liệu mẫu
data = np.random.randint(0, 100, size=(100,))

# Tạo DataFrame từ dữ liệu
df = pd.DataFrame(data, columns=['Value'])

# Tóm tắt dữ liệu
summary = df.describe()

# In ra kết quả
print(summary)
```



	Value
count	100.000000
mean	51.210000
std	28.592679
min	1.000000
25%	31.000000
50%	47.000000
75%	74.500000
max	99.000000

2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Tóm tắt dữ liệu (Bar Chart)

```
diem_toan = [9, 8, 7, 6, 9, 10, 5, 7, 8, 9]
from statistics import mean

diem_trung_binh = mean(diem_toan)
print("Điểm trung bình là:", diem_trung_binh)

diem_cao_nhat = max(diem_toan)
diem_thap_nhat = min(diem_toan)

print("Điểm cao nhất là:", diem_cao_nhat)
print("Điểm thấp nhất là:", diem_thap_nhat)
```

Điểm trung bình là: 7.8
Điểm cao nhất là: 10
Điểm thấp nhất là: 5

2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Trực quan hóa dữ liệu

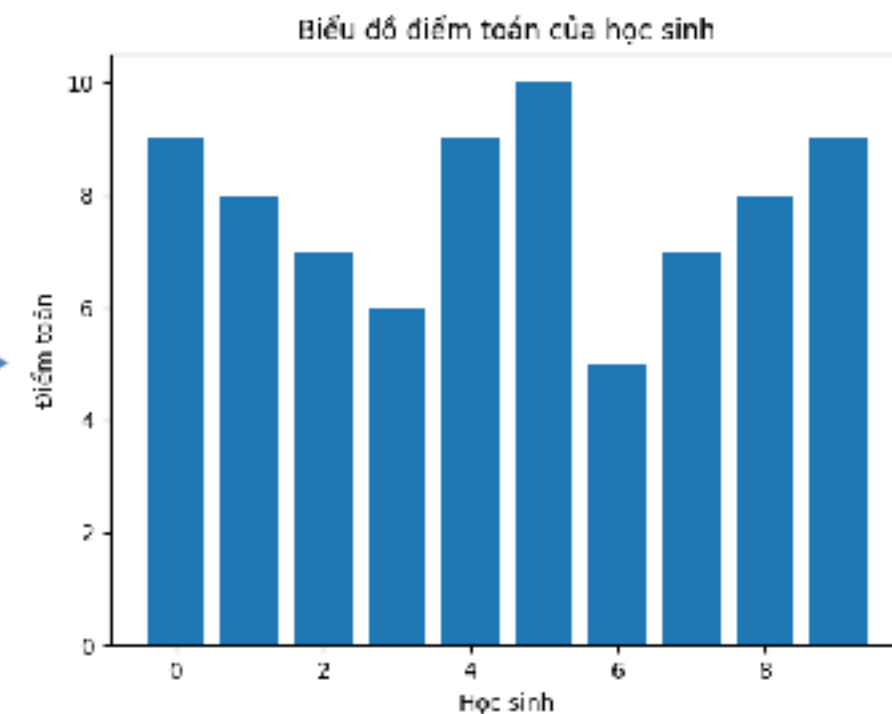
```
import matplotlib.pyplot as plt

diem_toan = [9, 8, 7, 6, 9, 10, 5, 7, 8, 9]

#Vẽ biểu đồ cột
plt.bar(range(len(diem_toan)), diem_toan)

plt.xlabel("Học sinh")
plt.ylabel("Điểm toán")

plt.title("Biểu đồ điểm toán của học sinh")
plt.show()
```



2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Trực quan hóa dữ liệu (Box Plot)

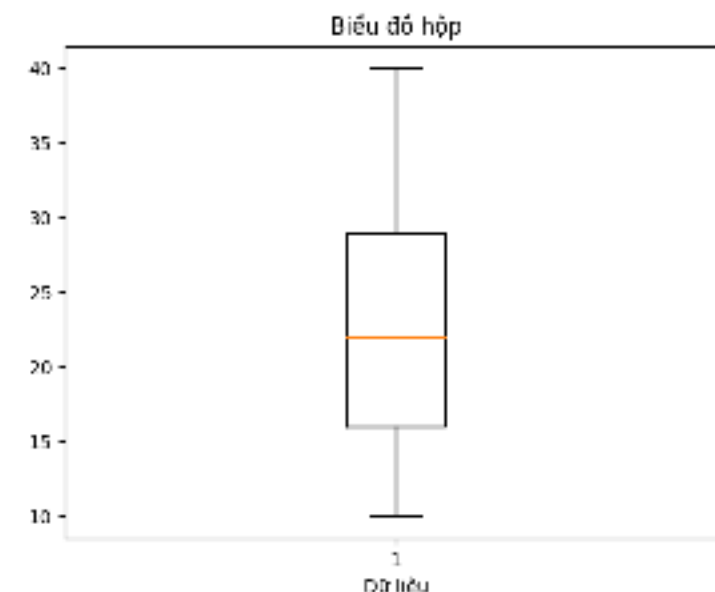
```
import matplotlib.pyplot as plt

# Dữ liệu mẫu
data = [10, 12, 15, 17, 20, 22, 25, 28, 30, 35, 40]

# Vẽ biểu đồ hộp
plt.boxplot(data)

# Đặt tiêu đề và nhãn cho biểu đồ
plt.title("Biểu đồ hộp")
plt.xlabel("Dữ liệu")

# Hiển thị biểu đồ
plt.show()
```



2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Trực quan hóa dữ liệu (Scatter Plot)

```
import pandas as pd
import matplotlib.pyplot as plt

# Đọc dữ liệu từ tệp CSV vào DataFrame
df = pd.read_csv('Vidu-phantan.csv')

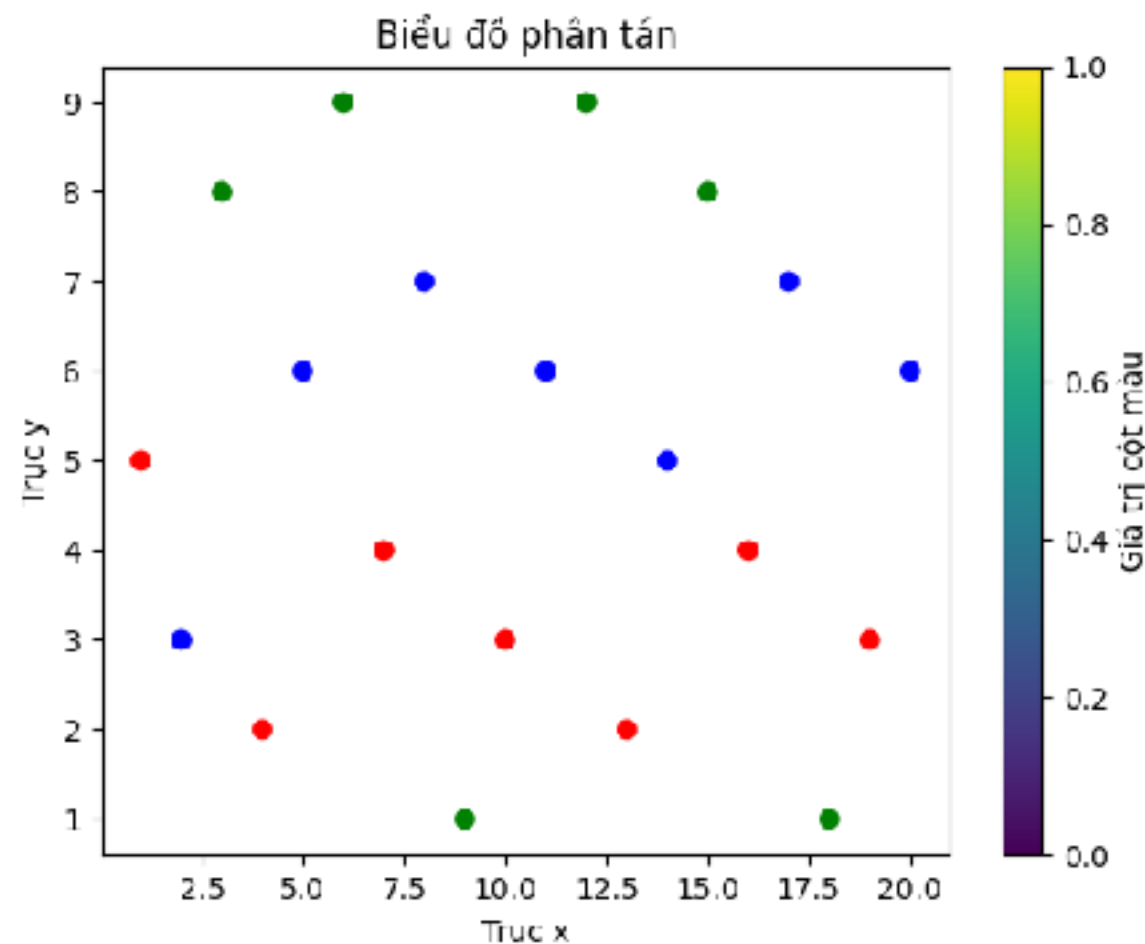
# Lấy các cột dữ liệu cần trực quan hóa
x = df['cot_x']
y = df['cot_y']
colors = df['cot_mau']

# Vẽ biểu đồ phân tán
plt.scatter(x, y, c=colors, cmap='viridis')

# Đặt tiêu đề và nhãn trục
plt.title('Biểu đồ phân tán')
plt.xlabel('Trục x')
plt.ylabel('Trục y')

# Thêm colorbar để hiển thị giá trị của cột màu
cbar = plt.colorbar()
cbar.set_label('Giá trị cột màu')

# Hiển thị biểu đồ
plt.show()
```



2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Trực quan hóa dữ liệu (Network Graph)

```
import networkx as nx
import matplotlib.pyplot as plt

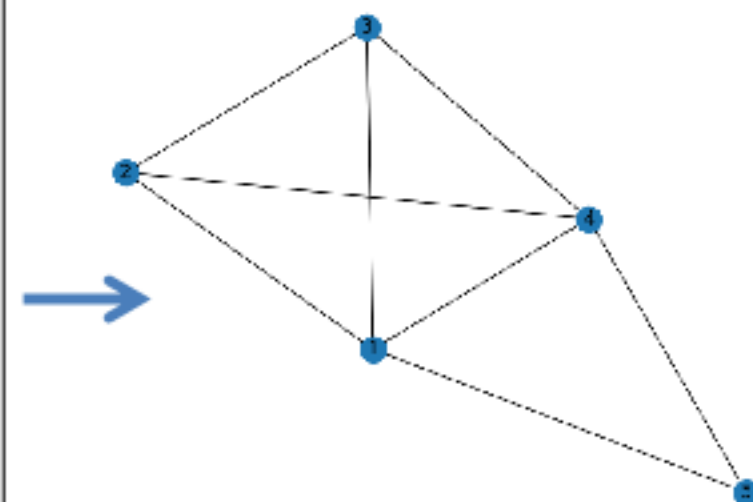
# Tạo đồ thị mạng
G = nx.Graph()

# Thêm các đỉnh vào đồ thị
G.add_nodes_from([1, 2, 3, 4, 5])

# Thêm các cạnh vào đồ thị
G.add_edges_from([(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4), (4, 5), (5, 1)])

# Vẽ đồ thị mạng
nx.draw(G, with_labels=True)

# Hiển thị đồ thị
plt.show()
```



2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Trực quan hóa dữ liệu thiếu (Missing)

```
import pandas as pd
import matplotlib.pyplot as plt

# Dữ liệu mẫu có dữ liệu thiếu
data = {'Tên': ['John', 'Mike', 'Sarah', 'Emily', 'David'],
        'Tuổi': [30, 25, None, 35, 28],
        'Lương': [5000, 4000, None, None, 4500]}

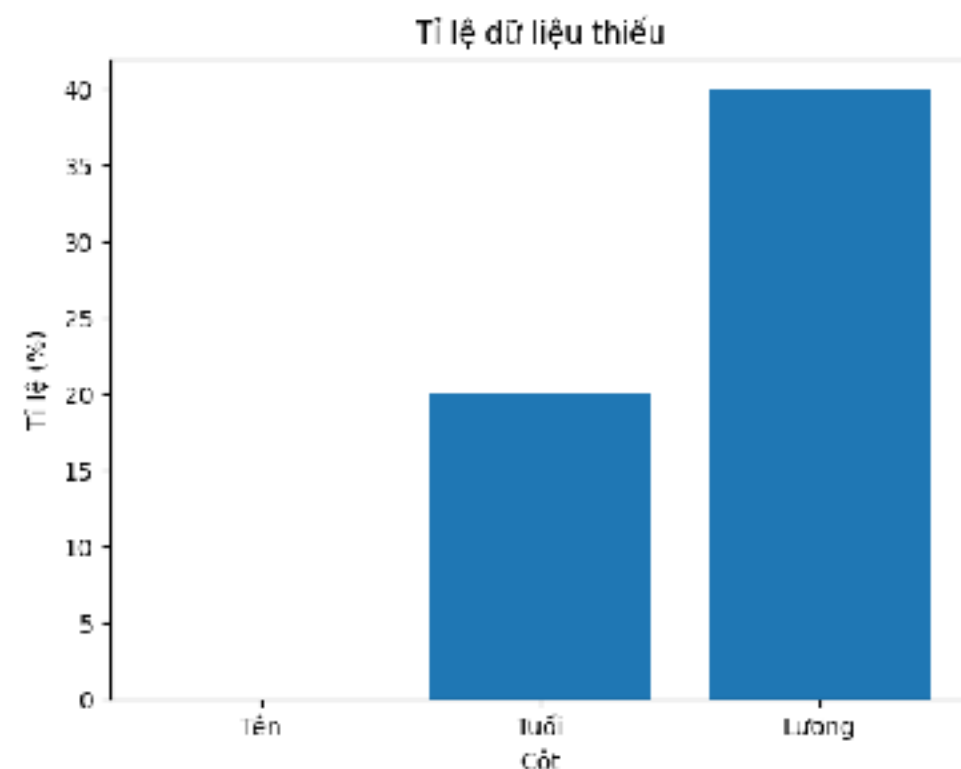
# Tạo DataFrame từ dữ liệu
df = pd.DataFrame(data)

# Tính tỷ lệ dữ liệu thiếu trong từng cột
missing_data = df.isnull().sum() / len(df) * 100

# Vẽ biểu đồ cột tỷ lệ dữ liệu thiếu
plt.bar(missing_data.index, missing_data.values)

# Đặt tiêu đề và nhãn cho biểu đồ
plt.title("Tỷ lệ dữ liệu thiếu")
plt.xlabel("Cột")
plt.ylabel("Tỷ lệ (%)")

# Hiển thị biểu đồ
plt.show()
```



2-Tóm tắt mô tả về dữ liệu

(Data Summarization & Data Description)

Ví dụ: Loại bỏ dữ liệu thiếu (Missing)

```
import pandas as pd
import numpy as np

# Tạo một DataFrame chứa dữ liệu có nhiều
data = {'A': [1, 2, np.nan, 4, 5],
        'B': [6, np.nan, 8, np.nan, 10],
        'C': [11, 12, 13, 14, 15]}
df = pd.DataFrame(data)

summary = df.describe()
print(summary)

# Xác định và loại bỏ các giá trị NaN (nhiều)
df_cleaned = df.dropna()

# Tóm tắt và mô tả dữ liệu sau khi loại bỏ nhiều
summary = df_cleaned.describe()
print(summary)
```



	A	B	C
count	4.000000	3.0	5.000000
mean	3.000000	8.0	13.000000
std	1.825742	2.0	1.581139
min	1.000000	6.0	11.000000
25%	1.750000	7.0	12.000000
50%	3.000000	8.0	13.000000
75%	4.250000	9.0	14.000000
max	5.000000	10.0	15.000000

	A	B	C
count	2.000000	2.000000	2.000000
mean	3.000000	8.000000	13.000000
std	2.828427	2.828427	2.828427
min	1.000000	6.000000	11.000000
25%	2.000000	7.000000	12.000000
50%	3.000000	8.000000	13.000000
75%	4.000000	9.000000	14.000000
max	5.000000	10.000000	15.000000

3-Làm sạch dữ liệu (Data Cleaning)

Làm sạch dữ liệu là một bước quan trọng trong giai đoạn TXLDL - loại bỏ các dữ liệu không chính xác, không hoàn chỉnh hoặc không cần thiết, đồng thời cải thiện chất lượng và độ tin cậy của dữ liệu:

1. **Loại bỏ dữ liệu trùng lặp:** Kiểm tra và loại bỏ các bản ghi trùng lặp trong tập dữ liệu - tìm các giá trị trùng lặp và xóa bỏ;
2. **Xử lý dữ liệu thiếu:** Kiểm tra và xử lý các giá trị thiếu trong tập dữ liệu - phương pháp như điền giá trị trung bình, giá trị trung vị hoặc giá trị phổ biến vào các ô thiếu, hoặc xóa bỏ các hàng hoặc cột chứa dữ liệu thiếu;

3-Làm sạch dữ liệu (Data Cleaning)

3. **Kiểm tra và xử lý dữ liệu ngoại lệ:** Kiểm tra và xử lý các giá trị ngoại lệ hoặc không hợp lệ trong tập dữ liệu - thay thế các giá trị ngoại lệ bằng giá trị trung bình, giới hạn hoặc xóa bỏ các hàng hoặc cột chứa dữ liệu ngoại lệ;
4. **Chuẩn hóa dữ liệu:** Đảm bảo rằng dữ liệu trong tập dữ liệu được chuẩn hóa và đồng nhất - chuyển đổi các giá trị về cùng một đơn vị đo lường, đơn vị tiền tệ hoặc định dạng;
5. **Loại bỏ các ký tự đặc biệt và dấu câu:** Loại bỏ các ký tự đặc biệt và dấu câu không cần thiết - làm sạch và chuẩn hóa văn bản;

3-Làm sạch dữ liệu (Data Cleaning)

6. **Kiểm tra và xử lý dữ liệu không hợp lệ:** Kiểm tra và xử lý các giá trị không hợp lệ hoặc không phù hợp - kiểm tra các ràng buộc dữ liệu, như kiểm tra kiểu dữ liệu, giới hạn giá trị và quy tắc kinh nghiệm;
7. **Xác định và xử lý dữ liệu dư thừa:** Kiểm tra và xử lý các dữ liệu dư thừa hoặc không cần thiết trong tập dữ liệu - xóa bỏ các cột hoặc hàng chứa dữ liệu không cần thiết hoặc không ảnh hưởng đến quá trình phân tích.

3-Làm sạch dữ liệu (Data Cleaning)

Ví dụ 1: Loại bỏ dữ liệu trùng lặp

```
import numpy as np
# Tạo một mảng 2 chiều với dữ liệu trùng lặp
data = np.array([[1, 2, 3],
                 [4, 2, 3],
                 [5, 6, 1],
                 [4, 7, 8],
                 [1, 2, 3],
                 [9, 5, 6]])
# Chuyển mảng thành một tập hợp 2D duy nhất
unique_data = np.unique(data, axis=0)
print(unique_data)
```



```
[[1 2 3]
 [4 2 3]
 [4 7 8]
 [5 6 1]
 [9 5 6]]
```

Hàm `np.unique()` để tìm các hàng duy nhất trong mảng `data`. Tham số `axis=0` tìm các hàng duy nhất; `axis=1` tìm các cột duy nhất.

3-Làm sạch dữ liệu (Data Cleaning)

Ví dụ 2: Xử lý dữ liệu thiếu – thay thế bằng giá trị trung bình

```
import pandas as pd
import numpy as np
# Tạo một DataFrame với dữ liệu thiếu
data = {'A': [1, 2, np.nan, 4, 5],
        'B': [6, np.nan, 8, np.nan, 10],
        'C': [11, 12, 13, np.nan, 15]}
df = pd.DataFrame(data)
# Điền giá trị trung bình vào các ô thiếu
df_filled = df.fillna(df.mean())

print(df_filled)
```



	A	B	C
0	1.0	6.0	11.0
1	2.0	NaN	12.0
2	NaN	8.0	13.0
3	4.0	NaN	NaN
4	5.0	10.0	15.0

	A	B	C
0	1.0	6.0	11.00
1	2.0	8.0	12.00
2	3.0	8.0	13.00
3	4.0	8.0	12.75
4	5.0	10.0	15.00

Hàm `fillna()` điền giá trị trung bình vào các ô thiếu trong DataFrame;

Hàm `mean()` tính giá trị trung bình của từng cột.

3-Làm sạch dữ liệu (Data Cleaning)

Ví dụ 3: Xử lý dữ liệu ngoại lệ – thay thế bằng giá trị trung bình

```
import pandas as pd
import numpy as np

# Tạo một Series với dữ liệu ngoại lệ
data = pd.Series([1, 2, 3, 100, 5, 6, 200, 8, 9, 50])

# Tính giá trị trung bình và độ lệch chuẩn
mean = data.mean()
std = data.std()

# Xác định ngưỡng trên và ngưỡng dưới
threshold = 2 * std
lower_threshold = mean - threshold
upper_threshold = mean + threshold

# Kiểm tra và điền giá trị trung bình cho dữ liệu ngoại lệ
data_cleaned = data.copy()
data_cleaned[(data_cleaned < lower_threshold) | (data_cleaned > upper_threshold)] = mean

print(data_cleaned)
```

0	1
1	2
2	3
3	100
4	5
5	6
6	200
7	8
8	9
9	50

mean: 38.40	
std: 64.96	
0	1.0
1	2.0
2	3.0
3	100.0
4	5.0
5	6.0
6	38.4
7	8.0
8	9.0
9	50.0

Quy tắc 3σ cho phân phối chuẩn:

- 68% các điểm dữ liệu nằm trong khoảng $\mu \pm \sigma$
- 95% các điểm dữ liệu nằm trong khoảng $\mu \pm 2\sigma$
- 99.7% các điểm dữ liệu nằm trong khoảng $\mu \pm 3\sigma$

4-Tích hợp dữ liệu (Data Integration)

Tích hợp dữ liệu trong giai đoạn TXLDL là quan trọng và cần thiết để đảm bảo tính toàn vẹn, đồng nhất và sẵn sàng của dữ liệu để sử dụng trong các bước tiếp theo của quy trình XLDL.

1. **Tạo tập dữ liệu duy nhất:** Tích hợp dữ liệu từ các nguồn khác nhau tạo ra một tập dữ liệu duy nhất, chứa thông tin từ nhiều nguồn khác nhau - *đảm bảo tính toàn vẹn và đồng nhất của dữ liệu*;
2. **Loại bỏ dữ liệu trùng lặp:** Xác định và loại bỏ dữ liệu trùng lặp, từ đó giảm thiểu sự lặp lại và tiết kiệm không gian lưu trữ;

4-Tích hợp dữ liệu (Data Integration)

3. **Đồng nhất dữ liệu:** đồng nhất các định dạng và cấu trúc dữ liệu từ các nguồn \neq ; đảm bảo rằng các thuộc tính và các giá trị dữ liệu có cùng ý nghĩa và đúng định dạng;
4. **Tạo mối liên kết giữa các nguồn dữ liệu:** tạo mối liên kết giữa các nguồn dữ liệu \neq dựa trên các thuộc tính chung - tăng khả năng sử dụng và tìm kiếm dữ liệu \Rightarrow cái nhìn toàn diện hơn về thông tin;
5. **Tăng tính nhất quán và sẵn sàng của dữ liệu:** tính nhất quán và sẵn sàng của dữ liệu bằng cách kiểm tra và đảm bảo rằng các ràng buộc dữ liệu được duy trì và không có mất mát thông tin quan trọng.

4-Tích hợp dữ liệu (Data Integration)

Ví dụ: Trộn 2 tập dữ liệu

```
import pandas as pd


# Đọc dữ liệu từ tệp tin data1.csv và data2.csv
data1 = pd.read_csv("data1.csv")
data2 = pd.read_csv("data2.csv")

print(data1)

print(data2)
# Ghép nối dữ liệu từ hai tệp tin dựa trên cột "ID"
merged_data = pd.merge(data1, data2, on="ID")

# In ra dữ liệu đã tích hợp
print(merged_data)

merged_data.to_csv("merged_data.csv", index=False)
```



	ID	Ten	So dien thoai
0	1	Nhat	123456789
1	2	Tam	234567891
2	3	Luc	345678912

	ID	Tuoi
0	2	25
1	3	31

	ID	Ten	So dien thoai	Tuoi
0	2	Tam	234567891	25
1	3	Luc	345678912	31

Phương thức `merge()` để ghép nối dữ liệu dựa trên cột "ID". Kết quả là một tệp tin dữ liệu mới có dữ liệu từ cả hai nguồn.

5-Biến đổi dữ liệu (Data Transformation)

Biến đổi dữ liệu là bước quan trọng để chuẩn bị dữ liệu ban đầu cho các bước tiếp theo của quy trình XLDL - thay đổi cấu trúc, định dạng hoặc giá trị của dữ liệu \Rightarrow phiên bản mới của dữ liệu ban đầu, là dạng mà thuận tiện để phân tích hoặc sử dụng cho mô hình học máy.

1. **Chuẩn hóa dữ liệu:** đưa các giá trị về cùng một khoảng hoặc phạm vi, giúp loại bỏ các biến động không cần thiết trong dữ liệu;
2. **Xử lý dữ liệu thiếu:** xử lý các giá trị thiếu hoặc không hợp lệ trong dữ liệu - điền giá trị trung bình, trung vị hoặc phổ biến vào các giá trị thiếu, hoặc loại bỏ các hàng hoặc cột có dữ liệu thiếu;

5-Biến đổi dữ liệu (Data Transformation)

3. **Loại bỏ nhiễu:** loại bỏ các giá trị nhiễu hoặc không hợp lệ trong dữ liệu - gồm việc áp dụng các quy tắc hoặc ngưỡng để loại bỏ các giá trị ngoại lai hoặc không hợp lệ;
4. **Rút trích đặc trưng:** để rút trích các đặc trưng (features) quan trọng từ dữ liệu ban đầu - tạo ra các biến đặc trưng mới hoặc trích xuất thông tin quan trọng từ các biến hiện có;
5. **Mã hóa dữ liệu:** mã hóa các biến hạng mục (categorical variables) thành dạng số để có thể sử dụng trong mô hình học máy - mã hóa one-hot, label hoặc frequency.

5-Biến đổi dữ liệu (Data Transformation)

Ví dụ: Mã hóa one-hot, label hoặc frequency.

```
import pandas as pd

# Tạo DataFrame mẫu
data = pd.DataFrame({'color': ['Red', 'Blue', 'Green', 'Red', 'Blue', 'Green']})

# Mã hóa dữ liệu hạng mục sử dụng mã hóa one-hot
encoded_data = pd.get_dummies(data, columns=['color'])

# In kết quả mã hóa dữ liệu
print(encoded_data)
```

	color_Blue	color_Green	color_Red
0	0	0	1
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	1	0

5-Biến đổi dữ liệu (Data Transformation)

Ví dụ: Mã hóa one-hot, label hoặc frequency.

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

# Tạo DataFrame mẫu
data = pd.DataFrame({'color': ['Red', 'Blue', 'Green', 'Red', 'Blue', 'Green']})

# Khởi tạo đối tượng LabelEncoder
label_encoder = LabelEncoder()

# Mã hóa dữ liệu hạng mục
data['color_encoded'] = label_encoder.fit_transform(data['color'])

# In kết quả mã hóa dữ liệu
print(data)
```

	color	color_encoded
0	Red	2
1	Blue	0
2	Green	1
3	Red	2
4	Blue	0
5	Green	1

Mã hóa nhãn chỉ phù hợp khi các giá trị hạng mục có một *thứ tự ngầm định*

5-Biến đổi dữ liệu (Data Transformation)

Ví dụ: Mã hóa one-hot, label hoặc frequency.

```
import pandas as pd

# Tạo DataFrame mẫu
data = pd.DataFrame({'color': ['Red', 'Blue', 'Green', 'Red', 'Blue', 'Green']})

# Tính tần suất xuất hiện của mỗi nhãn
frequency = data['color'].value_counts(normalize=True)

# Mã hóa dữ liệu hạng mục sử dụng tần suất
data['color_encoded'] = data['color'].map(frequency)

# In kết quả mã hóa dữ liệu
print(data)
```

	color	color_encoded
0	Red	0.333333
1	Blue	0.333333
2	Green	0.333333
3	Red	0.333333
4	Blue	0.333333
5	Green	0.333333

Mã hóa tần suất không yêu cầu các nhãn có thứ tự ngầm định và giúp giữ lại thông tin về *tần suất xuất hiện* của mỗi nhãn.

6-Thu giảm dữ liệu (Data Reduction)

Thu giảm dữ liệu (data reduction) là giảm kích thước của dữ liệu ban đầu mà vẫn giữ được phần lớn thông tin quan trọng \Rightarrow giảm độ phức tạp tính toán, tiết kiệm tài nguyên và tăng hiệu suất của XLDL.

1. **Giảm chiều dữ liệu:** giảm số lượng biến đặc trưng (features) trong dữ liệu, từ đó giảm chiều dữ liệu - giảm độ phức tạp tính toán, tăng tốc độ xử lý và giảm khả năng overfitting trong mô hình hóa;
2. **Loại bỏ các biến không quan trọng:** loại bỏ các biến không quan trọng hoặc ít có ảnh hưởng đến kết quả của mô hình;

6-Thu giảm dữ liệu (Data Reduction)

3. **Mẫu lấy mẫu:** lấy mẫu (sampling) để giảm kích thước dữ liệu –có thể được thực hiện ngẫu nhiên hoặc theo các phương pháp như mẫu lấy mẫu cân bằng (balanced sampling) để đảm bảo sự đại diện của các lớp hoặc nhãn trong dữ liệu;
4. **Kỹ thuật nén dữ liệu:** kỹ thuật nén dữ liệu để lưu trữ và xử lý dữ liệu hiệu quả hơn - nén *lossless* hoặc nén *lossy*;
5. **Phân cụm dữ liệu:** phân cụm dữ liệu (data clustering) để tạo ra các nhóm dữ liệu tương tự - giảm kích thước dữ liệu bằng cách thay thế các nhóm dữ liệu tương tự bằng một đại diện duy nhất.

6-Thu giảm dữ liệu (Data Reduction)

Ví dụ: Lấy mẫu (sampling) ngẫu nhiên.

```
import pandas as pd
from sklearn.utils import resample

# Đọc dữ liệu từ file CSV
data = pd.read_csv('phim.csv')
print(data)

# Lấy mẫu ngẫu nhiên từ tập dữ liệu
n_samples = 3 # Số lượng mẫu mới cần lấy
sampled_data = resample(data, n_samples=n_samples)
```

```
# In kết quả lấy mẫu
print(sampled_data)
```

	title	genre	...	actor_3	rating
0	The Avengers	Action	...	Mark Ruffalo	8.0
1	Titanic	Romance	...	Billy Zane	7.8
2	The Hangover	Comedy	...	Zach Galifianakis	7.7
3	Inception	Sci-Fi	...	Ellen Page	8.8
4	The Dark Knight	Action	...	Aaron Eckhart	9.0

[5 rows x 6 columns]

	title	genre	...	actor_3	rating
1	Titanic	Romance	...	Billy Zane	7.8
0	The Avengers	Action	...	Mark Ruffalo	8.0
3	Inception	Sci Fi	...	Ellen Page	8.8

[3 rows x 6 columns]

7-Rời rạc hóa dữ liệu (Data Discretization)

Rời rạc hóa dữ liệu là chuyển đổi các giá trị dữ liệu liên tục thành các giá trị rời rạc.

1. **Giảm độ phức tạp dữ liệu:** giảm độ phức tạp của dữ liệu bằng cách chuyển đổi các giá trị liên tục thành các giá trị rời rạc - đơn giản hóa quá trình xử lý dữ liệu và tính toán;
2. **Tạo danh mục:** tạo ra các danh mục hoặc nhóm dữ liệu dựa trên các giá trị rời rạc. (Ví dụ, thuộc tính "tuổi", chia dữ liệu thành các nhóm tuổi như "trẻ em", "thanh niên", "người trung niên" và "người già" giúp phân loại và phân tích dữ liệu dễ dàng hơn;

7-Rời rạc hóa dữ liệu (Data Discretization)

3. **Hỗ trợ các thuật toán máy học:** Một số thuật toán máy học yêu cầu dữ liệu đầu vào là dạng rời rạc \Rightarrow chuyển đổi dữ liệu liên tục thành dạng phù hợp cho các thuật toán này;
4. **Bảo mật dữ liệu:** bảo vệ thông tin nhạy cảm trong dữ liệu. Thay vì lưu trữ giá trị chính xác \Rightarrow chuyển đổi chúng thành các giá trị rời rạc để giảm khả năng nhận dạng thông tin cá nhân.

7-Rời rạc hóa dữ liệu (Data Discretization)

Ví dụ: Chuyển đổi dữ liệu liên tục thành dạng rời rạc.


```
import pandas as pd

# Tạo DataFrame từ dữ liệu
data = pd.DataFrame({
    'age': [25, 30, 35, 40, 45, 50, 55, 60],
    'income': [50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000]
})
print(data)

# Rời rạc hóa thuộc tính 'age' thành các nhóm tuổi
bins = [20, 30, 40, 50, 60, 70]
labels = ['20-30', '30-40', '40-50', '50-60', '60+']
data['age_group'] = pd.cut(data['age'], bins=bins, labels=labels)

# Rời rạc hóa thuộc tính 'income' thành các mức thu nhập
bins = [0, 60000, 80000, 100000, 120000, float('inf')]
labels = ['Low', 'Medium', 'High', 'Very High', 'Super High']
data['income_level'] = pd.cut(data['income'], bins=bins, labels=labels)

# In kết quả sau khi rời rạc hóa
print(data)
```



	age	income		age_group	income_level
0	25	50000		20-30	Low
1	30	60000		20-30	Low
2	35	70000		30-40	Medium
3	40	80000		30-40	Medium
4	45	90000		40-50	High
5	50	100000		40-50	High
6	55	110000		50-60	Very High
7	60	120000		50-60	Very High

8-Tạo cây phân cấp ý niệm

(Concept Hierarchy Generation)

Cây phân cấp ý niệm giúp tổ chức và biểu diễn mối quan hệ phân cấp giữa các ý niệm trong XLDL \Rightarrow hiểu rõ các mức độ và mối quan hệ giữa các ý niệm, tạo ra một cái nhìn tổng quan và chi tiết về dữ liệu.

1. **Tổ chức dữ liệu:** giúp việc quản lý và tìm kiếm dữ liệu dễ dàng hơn.
 \Rightarrow tạo ra một khung làm việc để tổ chức, phân loại và xác định mối quan hệ giữa các ý niệm trong dữ liệu;
2. **Tạo phân cấp ý niệm:** xác định mối quan hệ phân cấp giữa các ý niệm trong dữ liệu \Rightarrow hiểu rõ các mức độ, cấp độ và mối quan hệ giữa các ý niệm;

8-Tạo cây phân cấp ý niệm

(Concept Hierarchy Generation)

3. **Tối ưu hóa xử lý dữ liệu:** tối ưu hóa quá trình xử lý dữ liệu. Bằng cách phân loại và nhóm các ý niệm có mối quan hệ tương tự \Rightarrow giảm độ phức tạp tính toán và tăng hiệu suất trong việc XLDL;
4. **Tăng khả năng tương tác:** cung cấp một cấu trúc tương tác giữa các ý niệm và bước xử lý dữ liệu \Rightarrow người dùng tương tác với từng ý niệm và bước xử lý một cách linh hoạt, giúp thích nghi và điều chỉnh quá trình XLDL dựa trên nhu cầu cụ thể.

8-Tạo cây phân cấp ý niệm

(Concept Hierarchy Generation)

Ví dụ: Tạo cây phân cấp

```
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

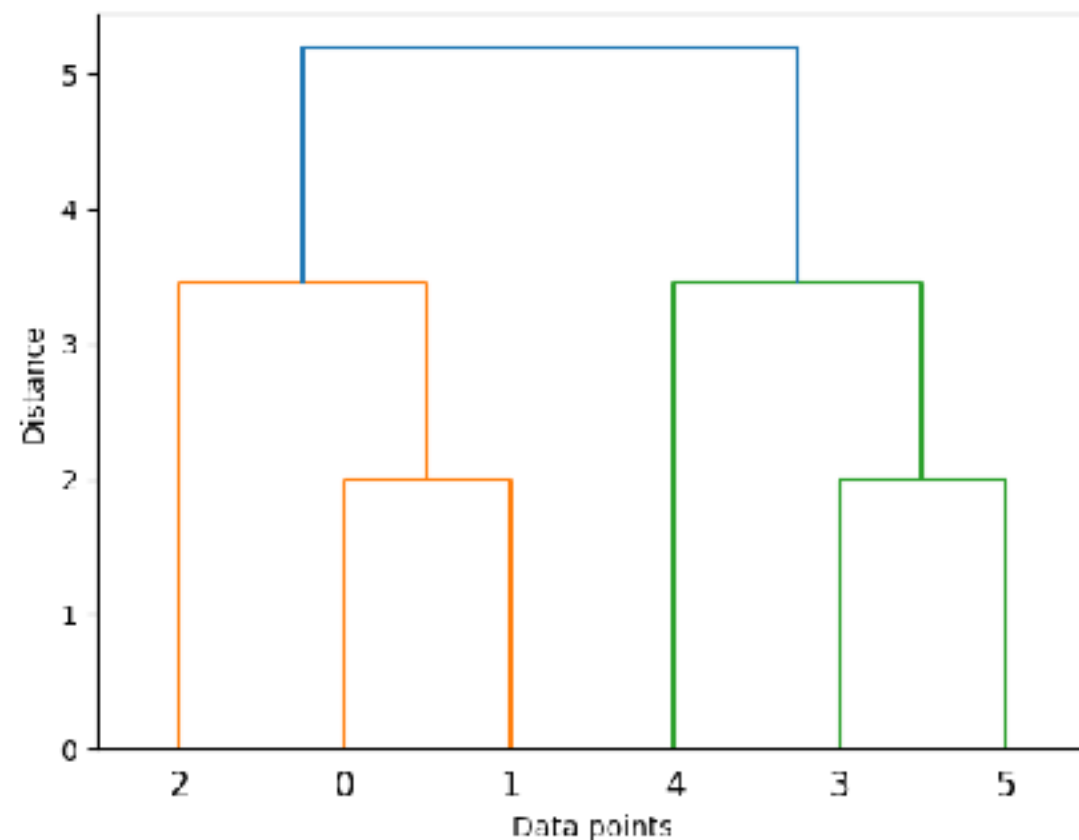
# Tạo dữ liệu giả định
X = np.array([[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]])

# Tạo ma trận khoảng cách
dist_matrix = linkage(X, method='ward')

# Vẽ biểu đồ dendrogram
dendrogram(dist_matrix)

# Đặt tên cho các trục
plt.xlabel('Data points')
plt.ylabel('Distance')

# Hiển thị biểu đồ dendrogram
plt.show()
```



9-Biểu diễn dữ liệu (Data Representation)

Biểu diễn dữ liệu (Data Representation) là quan trọng để đảm bảo dữ liệu được hiểu và xử lý một cách hiệu quả.

1. **Định dạng dữ liệu:** cho phép xác định định dạng của dữ liệu - đảm bảo rằng dữ liệu được tổ chức và biểu thị một cách rõ ràng. Ví dụ, dữ liệu có thể được biểu diễn dưới dạng văn bản, số, hình ảnh, âm thanh, video, hoặc các định dạng khác;
2. **Chuẩn hóa dữ liệu:** giúp chuẩn hóa dữ liệu thành dạng chuẩn, đồng nhất - xử lý các giá trị ngoại lệ, chuyển đổi đơn vị đo, định dạng hoặc chuẩn hóa dữ liệu;

9-Biểu diễn dữ liệu (Data Representation)

3. **Xử lý dữ liệu thiếu:** giúp xử lý dữ liệu thiếu bằng cách đánh dấu hoặc thay thế các giá trị thiếu - dữ liệu đầy đủ và đáng tin cậy hơn;
4. **Mã hóa dữ liệu:** mã hóa dữ liệu thành các biểu diễn số hoặc ký tự. Ví dụ, dữ liệu hình ảnh có thể được mã hóa dưới dạng các ma trận số, trong khi dữ liệu văn bản có thể được biểu diễn dưới dạng các ký tự hoặc mã hóa thành số;
5. **Giảm chiều dữ liệu:** giúp giảm chiều dữ liệu, tức là giảm số lượng biến hoặc thuộc tính của dữ liệu.