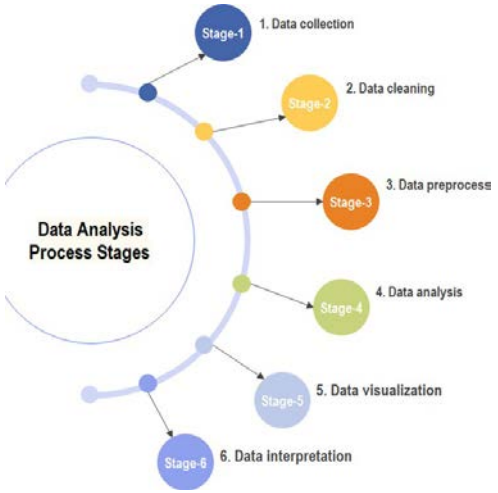
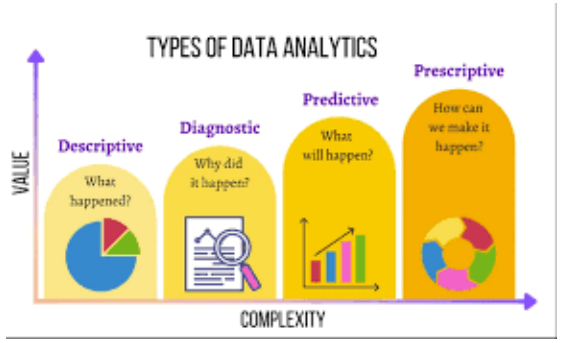
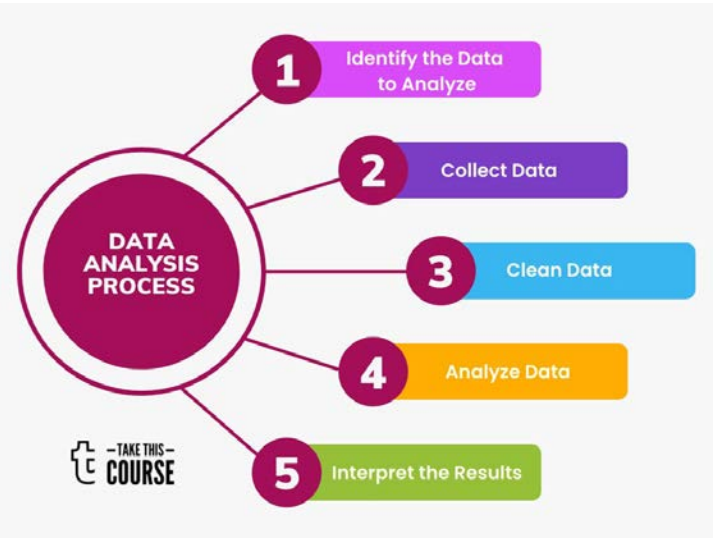




PHÂN TÍCH DỮ LIỆU

(Data Analysis)



Lê Văn Hạnh
levanhanhvn@gmail.com

NỘI DUNG MÔN HỌC

PHẦN 1 TỔNG QUAN & THU THẬP DỮ LIỆU CHO VIỆC PHÂN TÍCH

1. Khoa học dữ liệu
2. Thu thập dữ liệu
3. Tìm hiểu dữ liệu

PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU *(Data Preprocessing)*

4. Nhiệm vụ chính trong tiền xử lý dữ liệu
5. PANDAS
6. Thao tác với các định dạng khác nhau của tập tin dữ liệu
7. Làm sạch và Chuẩn bị dữ liệu
8. Sắp xếp dữ liệu: nối, kết hợp và định hình lại
9. Tổng hợp dữ liệu và các tác vụ trên nhóm

PHẦN 3 TRỰC QUAN HÓA DỮ LIỆU *(Data Visualization)*

10. Đồ thị và Biểu đồ
11. Vẽ đồ thị và Trực quan hóa

PHẦN 2
TIỀN XỬ LÝ DỮ LIỆU *(Data Preprocessing)*

Chương 4

CÁC KỸ THUẬT TIỀN XỬ LÝ DỮ LIỆU

Lê Văn Hạnh
levanhanhvn@gmail.com

NỘI DUNG CHƯƠNG 4

1. Tổng quan về tiền xử lý dữ liệu
2. Làm sạch dữ liệu *(Data Cleaning)*
3. Tích hợp dữ liệu *(Data Integration)*
4. Giảm thiểu dữ liệu *(Data Reduction)*
5. Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*
6. Thực hành

1. TỔNG QUAN VỀ TIỀN XỬ LÝ DỮ LIỆU

1.1. Tổng quan về tiền xử lý dữ liệu

1.1.1. Chất lượng dữ liệu & Tại sao phải xử lý thực hiện tiền xử lý dữ liệu?

- Cơ sở dữ liệu trong thế giới thực thường
  - Có thể có nguồn gốc từ nhiều nguồn không đồng nhất nên rất dễ bị nhiễu (*noisy*), thiếu (*missing data*) dữ liệu và không nhất quán (*inconsistent*)
  - Kích thước thường rất lớn của chúng (thường là vài gigabyte trở lên).

  $\Rightarrow$ Chất lượng của dữ liệu thấp

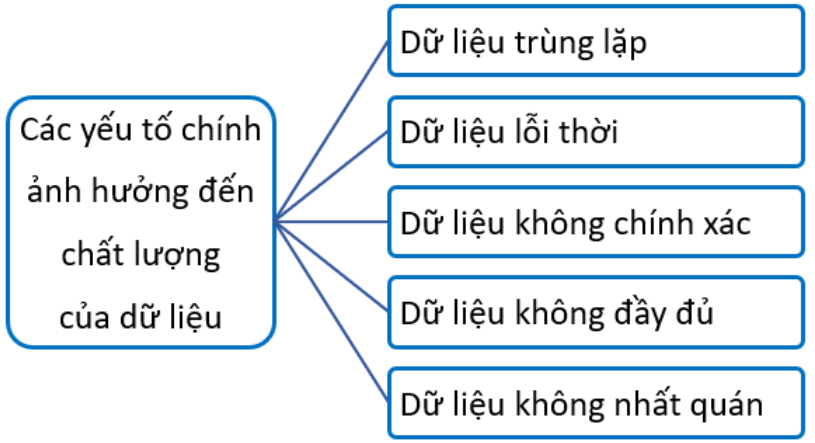
  $\Rightarrow$ Chất lượng kết quả khai thác thấp.

  $\Rightarrow$ **Làm thế nào dữ liệu có thể được xử lý trước để giúp cải thiện chất lượng dữ liệu và do đó, cải thiện kết quả khai thác?**

1.1. Tổng quan về tiền xử lý dữ liệu

1.1.2. Chất lượng dữ liệu & Tại sao phải xử lý thực hiện tiền xử lý dữ liệu?

- *Các yếu tố chính ảnh hưởng đến chất lượng dữ liệu*

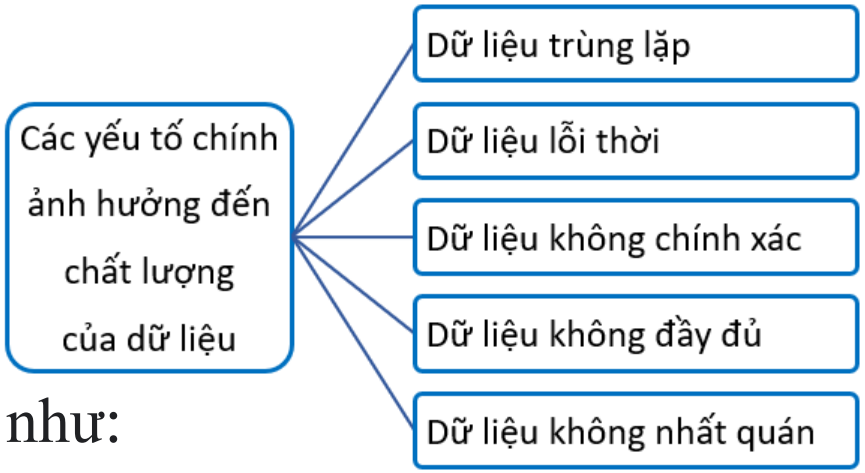


  - ***Dữ liệu trùng lặp*** (*Duplicated data*): Thường xảy ra do người dùng cập nhật cùng 1 nội dung/sự việc nhiều lần.

  - ***Dữ liệu lỗi thời*** *(Outdated data)*: tùy vào nhu cầu sử dụng của người phân tích mà dữ liệu có thể bị xem là lỗi thời hay không?

1.1. Tổng quan về tiền xử lý dữ liệu

1.1.2. Chất lượng dữ liệu & Tại sao phải xử lý thực hiện tiền xử lý dữ liệu?



- *Các yếu tố chính ảnh hưởng đến chất lượng dữ liệu*

  - ***Dữ liệu không chính xác*** (*inaccuracy*):

    Có nhiều lý do có thể dẫn đến dữ liệu không chính xác như:

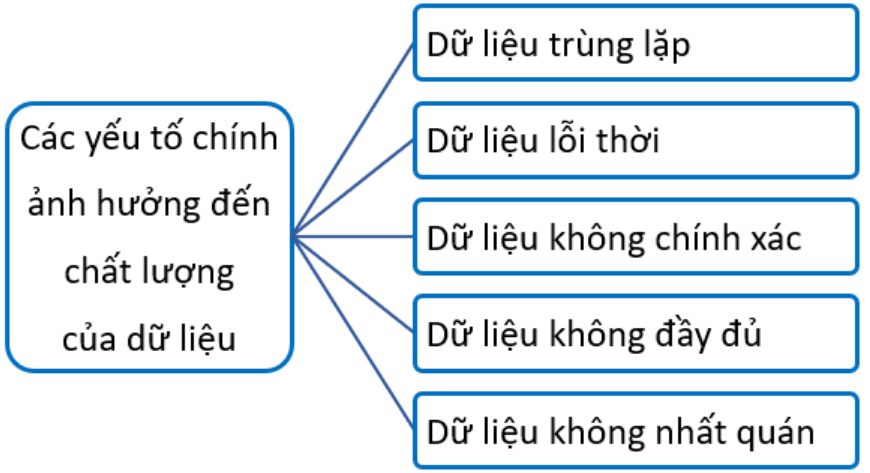
    - Lỗi do các công cụ thu thập dữ liệu
    - Lỗi do con người hoặc máy tính khi nhập dữ liệu (như trùng lặp).
    - Lỗi do người dùng cố tình cung cấp dữ liệu không chính xác.
    - Lỗi trong quá trình truyền dữ liệu
    - Lỗi do những hạn chế về công nghệ như kích thước bộ đệm hạn chế để điều phối việc truyền và tiêu thụ dữ liệu được đồng bộ hóa.
    - Lỗi do không nhất quán trong quy ước đặt tên hoặc mã dữ liệu hoặc định dạng không nhất quán đối với dữ liệu đầu vào (ví dụ: ngày).

1.1. Tổng quan về tiền xử lý dữ liệu

1.1.2. Chất lượng dữ liệu & Tại sao phải xử lý thực hiện tiền xử lý dữ liệu?

- *Các yếu tố chính ảnh hưởng đến chất lượng dữ liệu*

  - ***Dữ liệu không đầy đủ*** (*incomplete data*) có thể xảy ra vì một số lý do:
    - Dữ liệu bị xem là không đầy đủ đơn giản vì chúng không được coi là quan trọng tại thời điểm nhập dữ liệu. Do đó, các thuộc tính quan tâm hiện tại thuộc dạng này có thể không có sẵn.
    - Dữ liệu liên quan có thể không được ghi lại do hiểu lầm hoặc do trục trặc của thiết bị.



  - ***Dữ liệu không nhất quán*** (*inconsistency*): Chủ yếu do chưa hoặc không có quy định thống nhất chung khi nhập dữ liệu như tên gọi cho đối tượng (sản phẩm, tên địa danh, …), đơn vị tính, …

1.1. Tổng quan về tiền xử lý dữ liệu

1.1.2. Chất lượng dữ liệu & Tại sao phải xử lý thực hiện tiền xử lý dữ liệu?

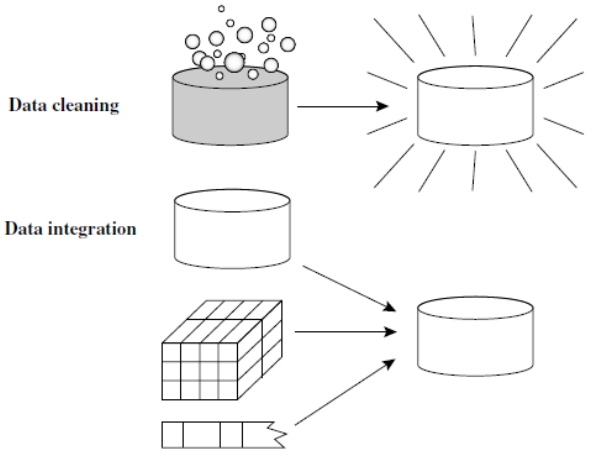
- *Các yếu tố khác ảnh hưởng đến chất lượng dữ liệu*

  - ***Tính kịp thời*** (*timeliness*) cũng ảnh hưởng đến chất lượng dữ liệu. Giả sử cần phân tích việc biến động giá của xăng dầu, nhưng mặt hàng này có thể biến động nhiều lần trong ngày. Vì vậy, nếu việc cập nhật số liệu không kịp thời chắc chắn sẽ ảnh hưởng đến việc phân tích dữ liệu.

  - ***Độ tin cậy*** (*believability*): phản ánh mức độ tin cậy của dữ liệu đối với người dùng. Nếu người dung phát hiện ra 1 vài lỗi về mức độ tin cậy của dữ liệu, thì người dung đó có thể không bao giờ "dám" sử dụng bộ dữ liệu đó nữa.

  - ***Khả năng diễn giải*** (*interpretability*): phản ánh mức độ dễ hiểu của dữ liệu. Do đó dù cơ sở dữ liệu hiện tại chính xác, đầy đủ, nhất quán và kịp thời nhưng do quá khó hiểu nên có khả năng người dùng sẽ không sử dụng.

1.2. Nhiệm vụ chính trong tiền xử lý dự liệu



- Tiền xử lý dữ liệu
  - **Làm sạch dữ liệu** *(data cleaning)*
    - Điền vào các giá trị còn thiếu
    - Làm mịn dữ liệu nhiễu
    - Xác định hoặc loại bỏ các giá trị ngoại lệ
    - Giải quyết sự không nhất quán
  - **Tích hợp dữ liệu** *(data integration)*
    - Nhận dạng thực thể
    - Phân tích dư thừa và tương quan
    - Trùng lặp dữ liệu
    - Phát hiện và giải quyết xung đột giá trị dữ liệu
  - **Giảm thiểu dữ liệu** *(data reduction)*
    - Giảm kích thước
    - Giảm số lượng
  - **Chuyển đổi dữ liệu** *(data transformation)*
    - Làm mịn
    - Xây dựng thuộc tính
    - Tổng hợp
    - Chuẩn hóa
    - Rời rạc hóa
    - Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa

1.2. Nhiệm vụ chính trong tiền xử lý dự liệu

1.2.1. Làm sạch dữ liệu (Data cleaning)

- Nếu người dùng tin rằng dữ liệu bị bẩn (dirty) ⇒ họ khó có thể tin tưởng vào kết quả của bất kỳ hoạt động khai thác dữ liệu nào đã được áp dụng vì kết quả đầu ra không đáng tin cậy. Do đó, bước tiền xử lý hữu ích là chạy dữ liệu của bạn thông qua một số quy trình làm sạch dữ liệu.
- Quy trình làm sạch dữ liệu gồm:
  - Điền vào các giá trị còn thiếu (*filling in missing values*).
  - Làm mịn dữ liệu nhiễu (*smoothing noisy data*)
  - Xác định hoặc loại bỏ các giá trị ngoại lệ (*identifying or removing outliers*)
  - Giải quyết sự không nhất quán (*resolving inconsistencies*).

1.2.2. Tích hợp dữ liệu (Data integration)

- Xảy ra khi muốn đưa dữ liệu từ nhiều nguồn vào phân tích của mình. Điều này sẽ liên quan đến việc tích hợp nhiều cơ sở dữ liệu (*databases*), khối dữ liệu (*data cubes*) hoặc tập tin (*files*).
- Thông thường, việc làm sạch dữ liệu và tích hợp dữ liệu được thực hiện như một bước tiền xử lý khi chuẩn bị dữ liệu cho kho dữ liệu. Việc làm sạch dữ liệu bổ sung có thể được thực hiện để phát hiện và loại bỏ những phần dư thừa có thể xảy ra do tích hợp dữ liệu.

1.2.2. Tích hợp dữ liệu (Data integration)

- Các vấn đề có thể gặp khi tích hợp dữ liệu:
  - Một số thuộc tính đại diện cho một khái niệm nhất định có thể có tên khác nhau trong các cơ sở dữ liệu khác nhau, gây ra sự thiếu nhất quán (*inconsistencies*) và dư thừa (*redundancies*). Ví dụ: cùng là thuộc tính nhận dạng khách hàng có thể được gọi là *custom_id* trong một kho dữ liệu này và *id-cust* trong một kho dữ liệu khác.
  - Sự không nhất quán:
    - Trong việc đặt tên. Ví dụ: cùng một tên có thể được đăng ký là “Bill” trong một CSDL 1, “William” trong CSDL 2 và “B.” trong CSDL 3.
    - Định dạng ngày/tháng/năm
    - …
  - Có thể có một số thuộc tính có thể được suy ra từ những thuộc tính khác (ví dụ: doanh thu hàng năm).

⇒ Việc có một lượng lớn dữ liệu dư thừa có thể làm chậm hoặc gây nhầm lẫn cho quá trình khám phá tri thức. Rõ ràng, ngoài việc làm sạch dữ liệu, các bước phải được thực hiện để tránh sự dư thừa trong quá trình tích hợp dữ liệu.

1.2.3. Giảm dữ liệu (Data reduction)

- Sau khi làm sạch và tích hợp dữ liệu, Lúc này tập dữ liệu đã chọn để phân tích trở nên LỚN (*huge*), điều này chắc chắn sẽ làm chậm quá trình khai thác.
- Giảm dữ liệu giúp tập dữ liệu có khối lượng nhỏ hơn nhiều nhưng vẫn tạo ra kết quả phân tích giống nhau (hoặc gần như giống nhau).
- Các chiến lược giảm thiểu dữ liệu bao gồm
  - ***Giảm kích thước*** (*Dimensionality reduction*): Trong việc giảm kích thước, các sơ đồ mã hóa dữ liệu được áp dụng để thu được biểu diễn được giảm bớt hoặc "nén" của dữ liệu gốc. Ví dụ
    - Các kỹ thuật nén dữ liệu như: sử dụng biến đổi wavelet (*wavelet transforms*) và phân tích thành phần chính (*principal components analysis*).
    - Lựa chọn tập hợp con thuộc tính cần dung (ví dụ: loại bỏ các thuộc tính không liên quan)
    - Xây dựng thuộc tính là một tập hợp nhỏ các thuộc tính hữu ích hơn được lấy từ tập hợp ban đầu.
  - ***Giảm số lượng*** (*Numerosity reduction)*: Trong việc giảm số lượng, dữ liệu được thay thế bằng các biểu diễn thay thế, nhỏ hơn bằng cách sử dụng:
    - Các mô hình tham số.
    - Chuyển các giá trị số lớn về thành phạm vi nhỏ hơn, chẳng hạn như [0,0; 1,0].

1.2.3. Giảm dữ liệu (Data reduction)

- Sau khi làm sạch và tích hợp dữ liệu, Lúc này tập dữ liệu đã chọn để phân tích trở nên LỚN (*huge*), điều này chắc chắn sẽ làm chậm quá trình khai thác.
- Giảm dữ liệu giúp tập dữ liệu có khối lượng nhỏ hơn nhiều nhưng vẫn tạo ra kết quả phân tích giống nhau (hoặc gần như giống nhau).
- Các chiến lược giảm thiểu dữ liệu bao gồm
  - ***Giảm kích thước*** (*Dimensionality reduction*): Trong việc giảm kích thước, các sơ đồ mã hóa dữ liệu được áp dụng để thu được biểu diễn được giảm bớt hoặc "nén" của dữ liệu gốc. Ví dụ
    - Các kỹ thuật nén dữ liệu như: sử dụng biến đổi wavelet (*wavelet transforms*) và phân tích thành phần chính (*principal components analysis*).
    - Lựa chọn tập hợp con thuộc tính cần dung (ví dụ: loại bỏ các thuộc tính không liên quan)
    - Xây dựng thuộc tính là một tập hợp nhỏ các thuộc tính hữu ích hơn được lấy từ tập hợp ban đầu.
  - ***Giảm số lượng*** (*Numerosity reduction)*: Trong việc giảm số lượng, dữ liệu được thay thế bằng các biểu diễn thay thế, nhỏ hơn bằng cách sử dụng:
    - Các mô hình tham số.
    - Chuyển các giá trị số lớn về thành phạm vi nhỏ hơn, chẳng hạn như [0,0; 1,0].

1.2.4. Chuyển đổi dữ liệu (Data transformation)

- Sự rời rạc hóa (*discretization*) và tạo hệ thống phân cấp khái niệm (*concept hierarchy generation*) là những công cụ mạnh mẽ để khai thác dữ liệu ở chỗ chúng cho phép khai thác dữ liệu ở nhiều mức độ trừu tượng.
- Các hình thức chuyển đổi dữ liệu:
  - Chuẩn hóa (*normalization*)
  - Rời rạc hóa dữ liệu (*data discretization*)
  - Tạo phân cấp khái niệm (*concept hierarchy generation*)

1.3. Những lỗi trong quá trình làm sạch dữ liệu

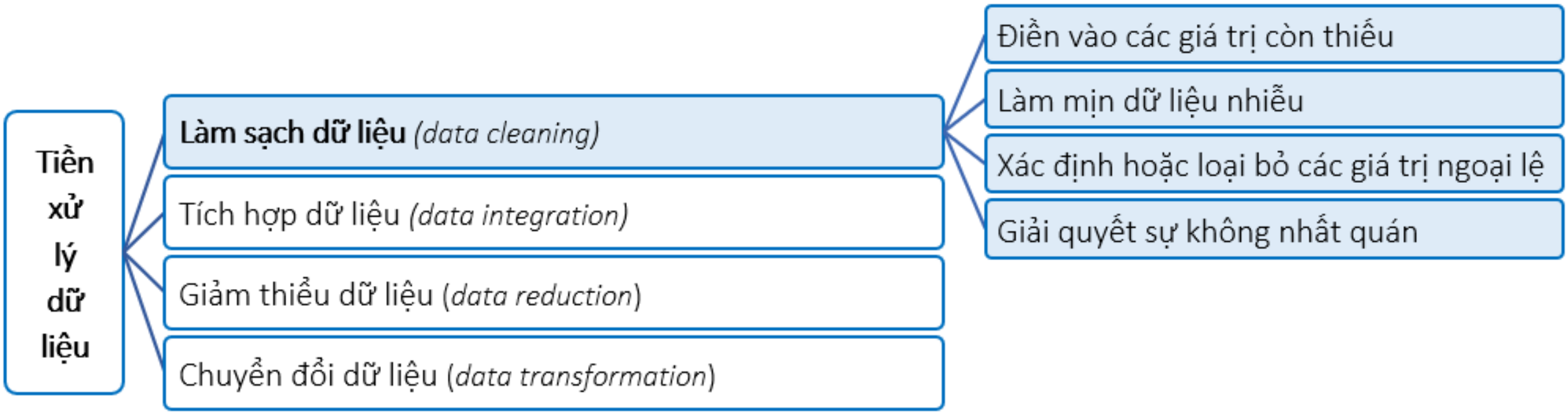
- *Không kiểm tra lỗi chính tả*: Lỗi chính tả có thể đơn giản như lỗi khi nhập liệu
- *Quên ghi lại cách sửa lỗi tài liệu*: Ghi lại cách sửa lỗi giúp tiết kiệm thời gian, vì nó giúp tránh những lỗi đó trong tương lai.
- *Không kiểm tra các giá trị sai*: Giá trị sai xảy ra khi các giá trị được nhập vào trường (field) bị sai. Các giá trị này vẫn có thể được định dạng chính xác, điều này khiến chúng khó phát hiện hơn nếu không cẩn thận.
- *Bỏ qua các giá trị bị thiếu*: Các giá trị bị thiếu trong tập dữ liệu có thể tạo ra lỗi và cung cấp cho bạn kết luận không chính xác. Cách tốt nhất, hãy cố gắng giữ cho dữ liệu sạch nhất có thể bằng cách duy trì tính đầy đủ và nhất quán.
- *Chỉ nhìn vào một tập hợp con của dữ liệu*: Điều quan trọng là phải suy nghĩ về tất cả các dữ liệu liên quan khi đang làm sạch. Điều này giúp đảm bảo phân tích viên hiểu toàn bộ câu chuyện mà dữ liệu đang kể và phân tích viên đang chú ý đến tất cả các lỗi có thể xảy ra.

NỘI DUNG CHƯƠNG 4

1. Tổng quan về tiền xử lý dữ liệu
2. Làm sạch dữ liệu *(Data Cleaning)*
3. Tích hợp dữ liệu *(Data Integration)*
4. Giảm thiểu dữ liệu *(Data Reduction)*
5. Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*
6. Thực hành

2. LÀM SẠCH DỮ LIỆU (Data Cleaning)

Dữ liệu trong thế giới thực có xu hướng không đầy đủ, nhiễu và không nhất quán. Các quy trình làm sạch dữ liệu (*Data cleaning* - hoặc dọn dẹp dữ liệu - *Data cleansing*) cố gắng điền vào các giá trị còn thiếu, làm giảm nhiễu trong khi xác định các giá trị ngoại lệ và sửa các điểm không nhất quán trong dữ liệu



2.1.Thiếu vắng giá trị (Missing Values)

Các phương pháp xử lý đối với dữ liệu bị thiếu

i. Bỏ qua bộ dữ liệu

- Khi bỏ qua bộ dữ liệu ⇒ bỏ qua giá trị của các thuộc tính còn lại trong bộ dữ liệu. Những giá trị này lại có thể hữu ích cho nhiệm vụ hiện tại.
- Phương pháp này không hiệu quả lắm, trừ khi tỷ lệ dữ liệu sẽ bỏ qua chiếm tỷ lệ thấp.
- Phương pháp này đặc biệt kém khi tỷ lệ bỏ qua cao.

ii. Điền giá trị còn thiếu theo cách thủ công

Nhìn chung, cách tiếp cận này tốn thời gian và có thể không khả thi với một tập dữ liệu lớn có nhiều giá trị bị thiếu.

2.1.Thiếu vắng giá trị (Missing Values)

Các phương pháp xử lý đối với dữ liệu bị thiếu

iii. *Sử dụng hằng số chung để điền giá trị còn thiếu*

- Thay thế tất cả các giá trị thuộc tính bị thiếu bằng cùng một hằng số, chẳng hạn như nhãn như "*Không biết*" (*unknown*) hoặc $-\infty$.
- Nếu các giá trị bị thiếu được thay thế bằng "*Không biết*" thì chương trình khai thác có thể nhầm tưởng rằng chúng tạo thành một khái niệm vì tất cả chúng đều có một giá trị chung là "*Không biết*".
- Vì vậy, phương pháp này tuy đơn giản nhưng không hề dễ thực hiện.

iv.*Sử dụng thước đo xu hướng trung tâm cho thuộc tính*

- Giá trị trung bình (*mean*): sử dụng khi phân phối dữ liệu bình thường (đối xứng - *symmetric*)
- Giá trị trung vị (*median*): sử dụng khi phân phối dữ liệu lệch (*skewed data distribution*)

2.1. Thiếu vắng giá trị (Missing Values)

Các phương pháp xử lý đối với dữ liệu bị thiếu

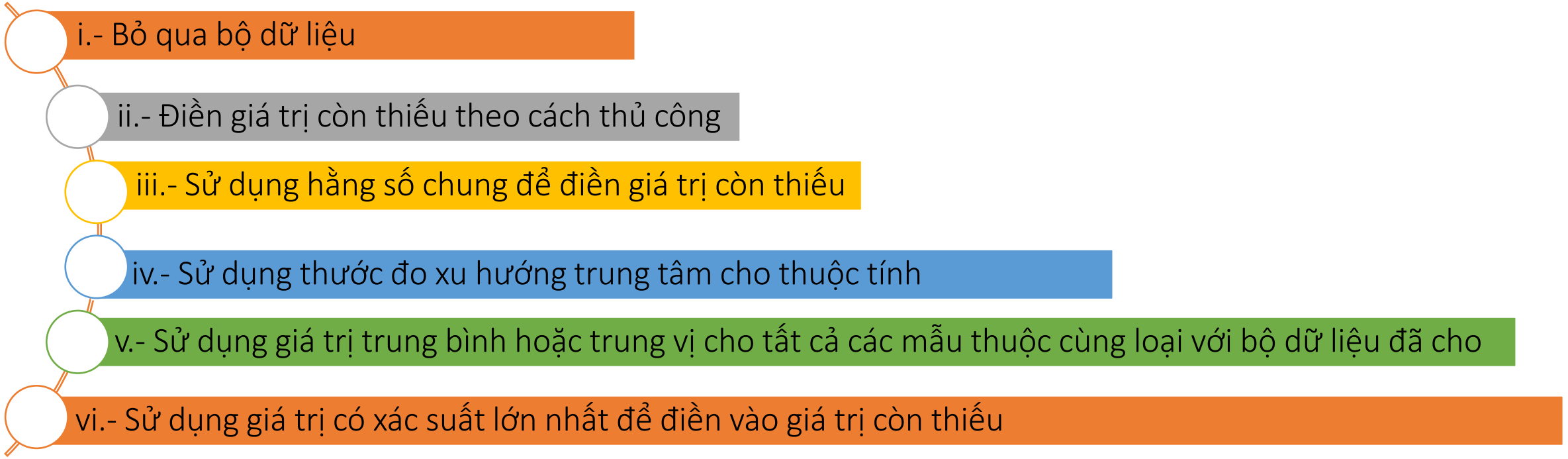
v. Sử dụng giá trị trung bình hoặc trung vị cho tất cả các mẫu thuộc cùng loại với bộ dữ liệu đã cho

Sử dụng trung bình hoặc trung vị tương tự như phương pháp ***iv*** ở trên cho những bộ dữ liệu cùng bị thiếu giá trị trên cùng 1 thuộc tính và có cùng phân loại (như cùng mức thu nhập, cùng ngành nghề, cùng khu vực, …

vi. Sử dụng giá trị có xác suất lớn nhất để điền vào giá trị còn thiếu

- Giá trị này có thể được xác định bằng hồi quy, Naïve Bayes hoặc quy nạp cây quyết định (*decision tree induction*).
- Ví dụ: bằng cách sử dụng các thuộc tính khách hàng khác trong tập dữ liệu để xây dựng cây quyết định nhằm dự đoán các giá trị còn thiếu của thuộc tính thu nhập.

2.1.Thiếu vắng giá trị (Missing Values)



- Phương pháp từ (***iii***) đến (***vi***) làm sai lệch dữ liệu – do giá trị được điền vào có thể không chính xác.
- Tuy nhiên, phương pháp (*vi*) là một chiến lược phổ biến. So với các phương pháp khác, nó sử dụng nhiều thông tin nhất từ dữ liệu hiện tại để dự đoán các giá trị còn thiếu.

2.1. Thiếu vắng giá trị (Missing Values)

- **Lưu ý**: trong một số trường hợp, giá trị bị thiếu có thể không có nghĩa là dữ liệu có lỗi! Ví dụ: khi đăng ký thẻ tín dụng, ứng viên có thể được yêu cầu cung cấp số giấy phép lái xe của họ. Những ứng viên không có giấy này đương nhiên có thể để trống thuộc tính này.
- Lý tưởng nhất là các quy trình phần mềm/các biểu mẫu cần có:
  - Có thể chỉ định liệu giá trị *null* có được phép hay không?
  - Mỗi thuộc tính nên có một hoặc nhiều quy tắc liên quan đến điều kiện *null* như "không áp dụng" (*"not applicable"*), "không biết" (*"don't know"*), "?" hoặc "không có" (*"none"*), …
  - Các quy tắc xử lý hoặc chuyển đổi các giá trị *null*.
  - Các thuộc tính cũng có thể được cố ý để trống nếu chúng được cung cấp ở bước sau của quy trình công việc.

2.1. Thiếu vắng giá trị (Missing Values)

⇒ Do đó:

- Mặc dù có thể đã cố gắng hết sức để làm sạch dữ liệu sau khi dữ liệu được lưu giữ, nhưng dữ liệu vẫn có thể vẫn "chưa sạch".
- Việc thiết kế cơ sở dữ liệu và quy trình nhập dữ liệu tốt sẽ giúp giảm thiểu số lượng giá trị bị thiếu hoặc lỗi ngay từ đầu.

2.2. Dữ liệu nhiễu (Noisy Data)

Nhiễu là sai số hoặc phương sai ngẫu nhiên của một biến đo được. Các kỹ thuật làm mịn dữ liệu:

2.2.1. Tạo nhóm (Binning techniques)

Là sắp xếp dữ liệu ban đầu tăng dần, sau đó phân phối đều chúng vào các "nhóm" (*buckets*) hoặc thùng (*bins*). Giá trị của các phần tử trong cùng bin sẽ quy về chung 1 giá trị bằng cách tham khảo "*vùng lân cận*" (*"neighborhood"*), tức là các giá trị có trong từng bin theo 1 trong 3 cách sau:

i. Thay thế bằng giá trị trung bình (mean) của bin.

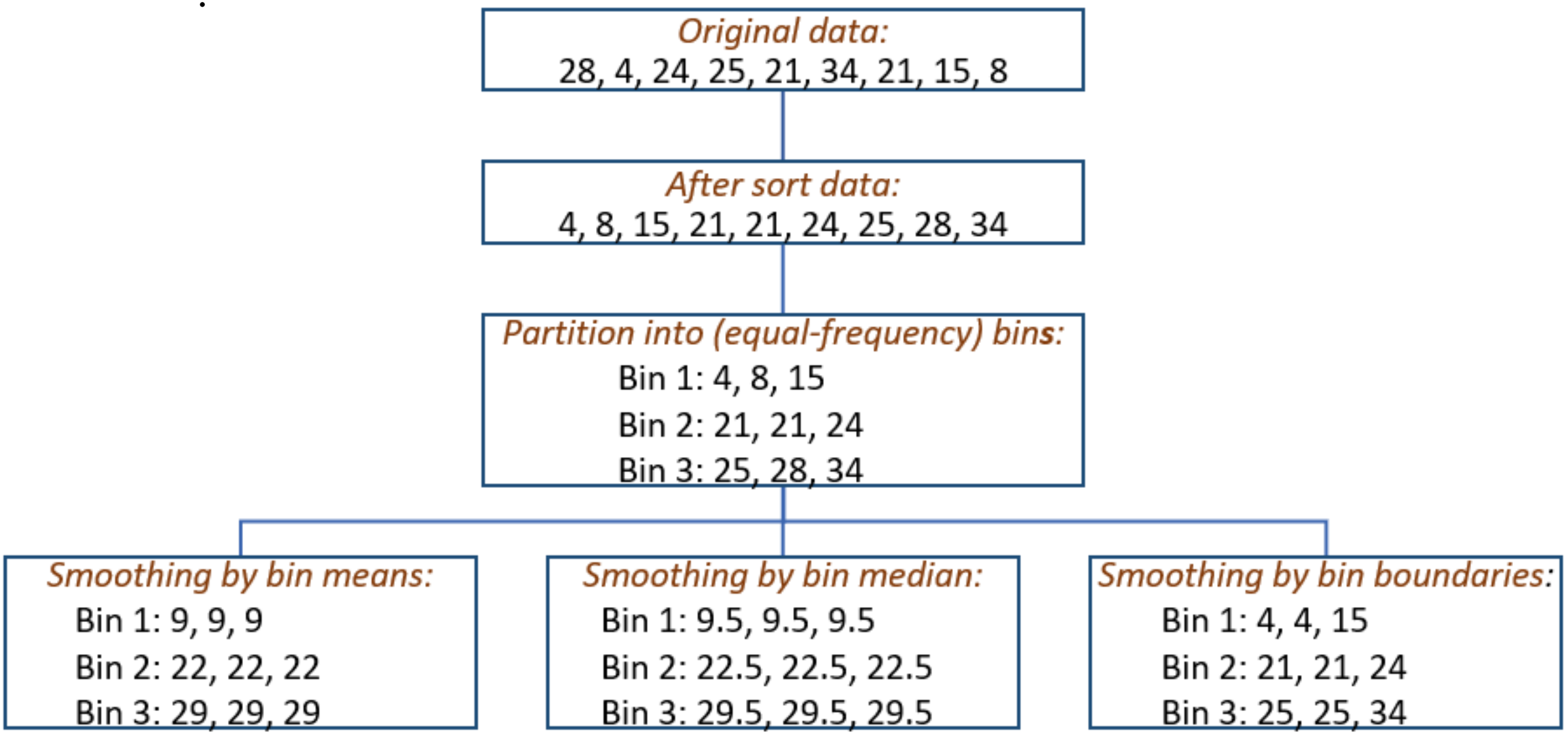
ii. Thay thế bằng giá trị trung vị (median)

iii. Làm mịn theo ranh giới của bin (*smoothing by bin boundaries*): giá trị tối thiểu và tối đa trong một bin nhất định được xác định là ranh giới ngăn của *bin*. Mỗi giá trị còn lại trong bin được thay thế bằng giá trị biên gần nhất.

2.2. Dữ liệu nhiễu (Noisy Data)

2.2.1. Tạo nhóm (Binning techniques)

Ví dụ



2.2. Dữ liệu nhiễu (Noisy Data)

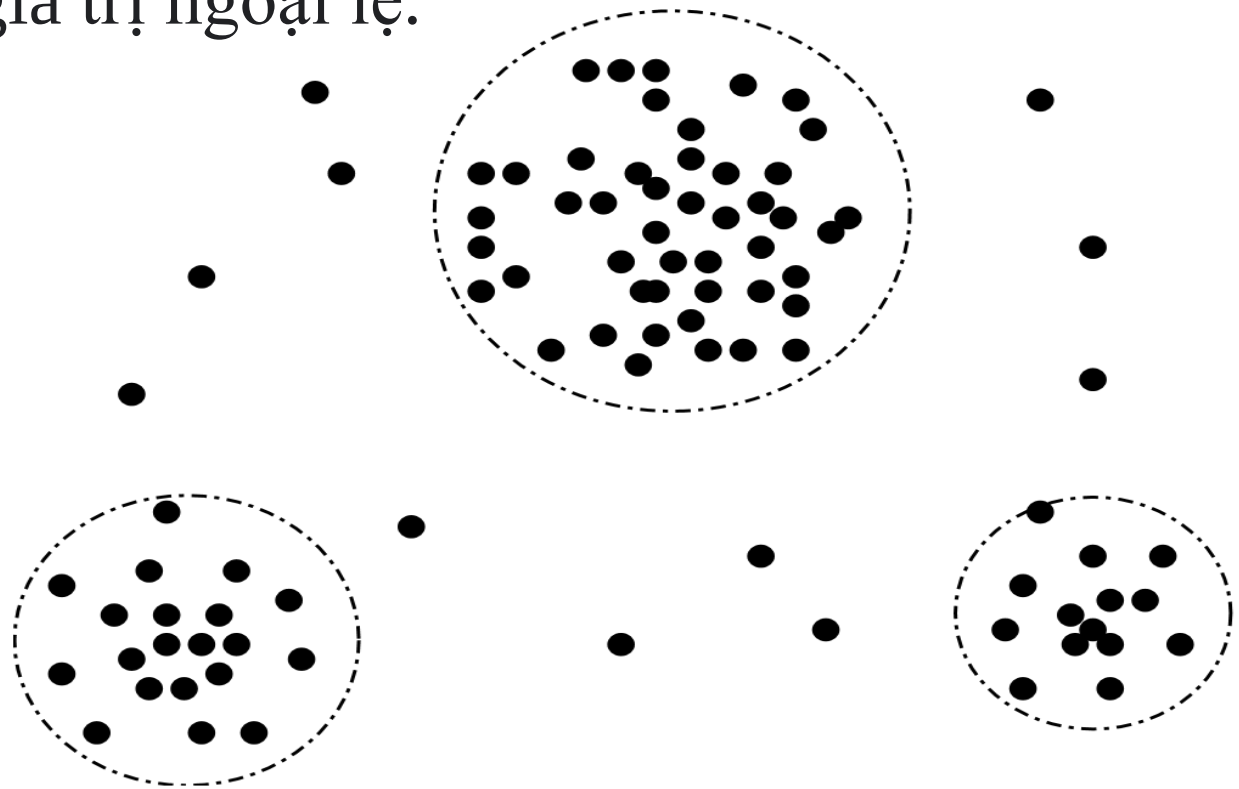
2.2.2. Hồi quy (regression)

- Làm mịn dữ liệu cũng có thể được thực hiện bằng hồi quy, một kỹ thuật làm phù hợp các giá trị dữ liệu với một hàm.
- *Hồi quy tuyến tính* (*linear regression*) liên quan đến việc tìm ra đường "tốt nhất" phù hợp với hai thuộc tính (hoặc biến) để một thuộc tính có thể được sử dụng để dự đoán thuộc tính kia.
- *Hồi quy tuyến tính bội* (*multiple linear regression*) là phần mở rộng của hồi quy tuyến tính, trong đó có nhiều hơn hai thuộc tính có liên quan và dữ liệu phù hợp với bề mặt đa chiều.

2.2. Dữ liệu nhiễu (Noisy Data)

2.2.3. Phân tích ngoại lệ (Outlier analysis)

Các ngoại lệ có thể được phát hiện bằng cách phân cụm (*clustering*), ví dụ: trong đó các giá trị tương tự được tổ chức thành các nhóm (*groups*) hoặc "cụm" (*clusters*). Theo trực giác, các giá trị nằm ngoài tập hợp các cụm có thể được coi là các giá trị ngoại lệ.



2.2. Dữ liệu nhiễu (Noisy Data)

2.2.4. Chuyển đổi dữ liệu để giảm số lượng giá trị riêng biệt

Nhiều phương pháp làm mịn dữ liệu cũng được sử dụng để rời rạc hóa dữ liệu (một dạng chuyển đổi dữ liệu) và giảm bớt dữ liệu:

- *Kỹ thuật tạo nhóm* (*binning techniques*) sẽ giảm số lượng giá trị riêng biệt cho mỗi thuộc tính. Điều này hoạt động như một hình thức rút gọn dữ liệu cho các phương pháp khai thác dữ liệu dựa trên logic (*logic-based data mining methods*),
- *Quy nạp cây quyết định* (*decision tree induction*) liên tục thực hiện so sánh giá trị trên dữ liệu đã được sắp xếp.
- *Hệ thống phân cấp khái niệm* (*Concept hierarchies*): Ví dụ, hệ thống phân cấp khái niệm về giá có thể ánh xạ các giá trị giá thực thành giá rẻ tiền, giá vừa phải và đắt, do đó làm giảm số lượng giá trị dữ liệu được xử lý trong quá trình khai thác.

2.3. Quá trình làm sạch dữ liệu (Data Cleaning as a Process)

Quy trình 2 bước để làm sạch dữ liệu:

B1:Phát hiện sự khác biệt

- *Sự khác biệt có thể do một số yếu tố gây ra*, bao gồm:
  - Các biểu mẫu nhập dữ liệu được thiết kế kém có nhiều field tùy chọn.
  - Lỗi của con người khi nhập dữ liệu, lỗi cố ý (ví dụ: không muốn tiết lộ thông tin về bản thân) và dữ liệu lỗi thời (ví dụ: địa chỉ đã thay đổi nhưng không được cập nhật).
  - Lỗi do việc trình bày dữ liệu không nhất quán và việc sử dụng mã không nhất quán (ví dụ: “2010/12/25” và “25/12/2010” cho kiểu dữ liệu ngày).
  - Các nguyên nhân khác gây ra lỗi khác biệt gồm:
    - Lỗi trong thiết bị đo đạc ghi lại dữ liệu.
    - Dữ liệu được sử dụng (không đầy đủ) cho các mục đích khác với dự định ban đầu.
    - Có thể có sự mâu thuẫn do tích hợp dữ liệu (ví dụ: cùng một thuộc tính nhưng có thể có các tên khác nhau trong các cơ sở dữ liệu khác nhau).

2.3. Quá trình làm sạch dữ liệu (Data Cleaning as a Process)

B1:Phát hiện sự khác biệt

- *Làm thế nào để có thể tiến hành phát hiện sự khác biệt?*
  - Sử dụng bất kỳ kiến thức đã có về các thuộc tính của dữ liệu. Ví dụ: kiểu dữ liệu và miền giá trị của từng thuộc tính là gì? Các giá trị được chấp nhận cho mỗi thuộc tính là gì?
  - Vận dụng các mô tả dữ liệu thống kê cơ bản để nắm bắt xu hướng dữ liệu và xác định các điểm bất thường. Ví dụ:
    - Tìm các giá trị trung bình, trung vị và mode.
    - Dữ liệu có đối xứng hay bị lệch không?
    - Phạm vi của các giá trị là gì? Tất cả các giá trị có nằm trong phạm vi dự kiến không?
    - Độ lệch chuẩn của từng thuộc tính là bao nhiêu? Các giá trị có nhiều hơn hai độ lệch chuẩn so với giá trị trung bình của một thuộc tính nhất định có thể được gắn cờ là các giá trị ngoại lệ tiềm năng.
    - Có bất kỳ sự phụ thuộc nào đã biết giữa các thuộc tính không?

2.3. Quá trình làm sạch dữ liệu (Data Cleaning as a Process)

B1:Phát hiện sự khác biệt

- *Làm thế nào để có thể tiến hành phát hiện sự khác biệt?*
  - Dữ liệu cũng cần được kiểm tra về các quy tắc sau:
    - *Quy tắc duy nhất* (*unique rules*): giá trị trên cùng 1 thuộc tính không được trùng nhau.
    - *Quy tắc liên tiếp* (*consecutive rules*): không được thiếu giá trị nào giữa giá trị thấp nhất và cao nhất của thuộc tính và tất cả các giá trị cũng phải là duy nhất (ví dụ: như trong số hóa đơn).
    - *Quy tắc vô hiệu* (*null rules*): chỉ định việc sử dụng khoảng trống, dấu chấm hỏi, ký tự đặc biệt hoặc các chuỗi khác có thể chỉ ra điều kiện null. Các lý do thiếu giá trị có thể bao gồm:
      - Người được yêu cầu cung cấp giá trị cho thuộc tính từ chối và/hoặc thấy rằng thông tin được yêu cầu không áp dụng được;
      - Người nhập dữ liệu không biết giá trị chính xác;
      - Giá trị sẽ được cung cấp ở bước sau của quy trình. Quy tắc null phải chỉ định cách ghi lại điều kiện null, chẳng hạn như lưu trữ số 0 cho thuộc tính số, khoảng trống cho thuộc tính kiểu chuỗi hoặc bất kỳ quy ước nào khác có thể được sử dụng (ví dụ: các mục nhập như “không biết” hoặc “?” nên được chuyển thành null).

2.3. Quá trình làm sạch dữ liệu (Data Cleaning as a Process)

B2: Chuyển đổi dữ liệu để sửa chữa sự khác biệt của dữ liệu

Có thể sử dụng 1 số công cụ thương mại hoặc tự viết tập lệnh tùy chỉnh cho bước này của quy trình làm sạch dữ liệu như:

- *Các công cụ lọc dữ liệu* sử dụng kiến thức miền đơn giản (ví dụ: kiến thức về địa chỉ bưu chính và kiểm tra chính tả) để phát hiện lỗi và sửa dữ liệu. Những công cụ này dựa vào kỹ thuật phân tích cú pháp (*parsing*) và kết hợp mờ (*fuzzy matching*) khi làm sạch dữ liệu từ nhiều nguồn.
- *Các công cụ kiểm tra dữ liệu* tìm ra sự khác biệt bằng cách phân tích dữ liệu để khám phá các quy tắc và mối quan hệ, đồng thời phát hiện dữ liệu vi phạm các điều kiện đó. Chúng là các biến thể của công cụ khai thác dữ liệu. Ví dụ: có thể sử dụng các mô tả dữ liệu thống kê cơ bản hoặc phân tích thống kê để tìm ra mối tương quan hoặc phân cụm để xác định các ngoại lệ.

2.3. Quá trình làm sạch dữ liệu (Data Cleaning as a Process)

B2: Chuyển đổi dữ liệu để sửa chữa sự khác biệt của dữ liệu

Có thể sử dụng 1 số công cụ thương mại hoặc tự viết tập lệnh tùy chỉnh cho bước này của quy trình làm sạch dữ liệu như:

- *Các công cụ chuyển đổi dữ liệu*. Các công cụ di chuyển dữ liệu cho phép chỉ định các phép biến đổi đơn giản, chẳng hạn như thay thế chuỗi "gender" bằng "sex". Các công cụ ETL (*extraction/transformation/loading* - trích xuất/chuyển đổi/tải) cho phép người dùng chỉ định các phép biến đổi thông qua giao diện người dùng đồ họa (GUI). Tuy nhiên, các công cụ này thường chỉ hỗ trợ một tập hợp biến đổi bị hạn chế.
- *Chuyển đổi thủ công* bằng cách sử dụng các tham chiếu bên ngoài. Ví dụ: các lỗi xảy ra khi nhập dữ liệu có thể được sửa bằng cách thực hiện theo dõi trên giấy. Tuy nhiên, hầu hết các lỗi sẽ yêu cầu chuyển đổi dữ liệu. Nghĩa là, một khi đã tìm thấy sự khác biệt, thường cần xác định và áp dụng (một loạt) phép biến đổi để sửa chúng.

2.4. Tính tương tác của quá trình làm sạch dữ liệu

- *Quá trình hai bước ở trên vẫn có những hạn chế*:
  - Dễ xảy ra sai sót và tốn thời gian.
  - Một số chuyển đổi có thể gây ra nhiều khác biệt hơn.
  - Một số khác biệt lồng nhau chỉ có thể được phát hiện sau khi những khác biệt khác đã được khắc phục. Ví dụ: lỗi đánh máy như “20010” trong trường năm chỉ có thể xuất hiện sau khi tất cả các giá trị ngày đã được chuyển đổi sang định dạng thống nhất.
  - Việc chuyển đổi thường được thực hiện theo quy trình hàng loạt. Chỉ sau khi quá trình chuyển đổi hoàn tất, người dùng mới có thể quay lại và kiểm tra xem có bất thường mới nào được tạo nhầm hay không.
  - Thông thường, cần phải lặp lại nhiều lần trước khi người dùng hài lòng. Bất kỳ bộ dữ liệu nào không thể được xử lý tự động bằng một phép chuyển đổi nhất định thường được ghi vào một tập tin mà không có bất kỳ lời giải thích nào về lý do đằng sau sự thất bại của chúng. Kết quả là toàn bộ quá trình làm sạch dữ liệu cũng bị thiếu tính tương tác.

2.4. Tính tương tác của quá trình làm sạch dữ liệu

- *Quá trình hai bước ở trên vẫn có những hạn chế*:
  - Dễ xảy ra sai sót và tốn thời gian.
  - Một số chuyển đổi có thể gây ra nhiều khác biệt hơn.
  - Một số khác biệt lồng nhau chỉ có thể được phát hiện sau khi những khác biệt khác đã được khắc phục. Ví dụ: lỗi đánh máy như “20010” trong trường năm chỉ có thể xuất hiện sau khi tất cả các giá trị ngày đã được chuyển đổi sang định dạng thống nhất.
  - Việc chuyển đổi thường được thực hiện theo quy trình hàng loạt. Chỉ sau khi quá trình chuyển đổi hoàn tất, người dùng mới có thể quay lại và kiểm tra xem có bất thường mới nào được tạo nhầm hay không.
  - Thông thường, cần phải lặp lại nhiều lần trước khi người dùng hài lòng. Bất kỳ bộ dữ liệu nào không thể được xử lý tự động bằng một phép chuyển đổi nhất định thường được ghi vào một tập tin mà không có bất kỳ lời giải thích nào về lý do đằng sau sự thất bại của chúng. Kết quả là toàn bộ quá trình làm sạch dữ liệu cũng bị thiếu tính tương tác.

2.4. Tính tương tác của quá trình làm sạch dữ liệu

- *Các phương pháp mới để làm sạch dữ liệu nhấn mạnh đến tính tương tác tăng lên*

Ví dụ: Potter's Wheel là một công cụ làm sạch dữ liệu có sẵn công khai, tích hợp tính năng phát hiện và chuyển đổi sự khác biệt.

Người dùng dần dần xây dựng một loạt các phép biến đổi bằng cách soạn thảo và gỡ lỗi các phép biến đổi riêng lẻ, từng bước một, trên giao diện giống như bảng tính. Các phép biến đổi có thể được xác định bằng đồ họa hoặc bằng cách cung cấp các ví dụ. Kết quả được hiển thị ngay lập tức trên các bản ghi hiển thị trên màn hình. Người dùng có thể chọn hoàn tác các phép biến đổi để các phép biến đổi gây ra các lỗi bổ sung có thể được "xóa". Công cụ này tự động thực hiện kiểm tra sự khác biệt ở chế độ nền trên chế độ xem dữ liệu được chuyển đổi mới nhất. Người dùng có thể dần dần phát triển và tinh chỉnh các phép biến đổi khi tìm thấy sự khác biệt, dẫn đến việc làm sạch dữ liệu hiệu quả hơn.

2.4. Tính tương tác của quá trình làm sạch dữ liệu

- *Cách tiếp cận khác*
  - Phát triển các ngôn ngữ khai báo để đặc tả các toán tử chuyển đổi dữ liệu.
  - Công việc như vậy tập trung vào việc xác định các phần mở rộng mạnh mẽ cho SQL và các thuật toán cho phép người dùng thể hiện các thông số kỹ thuật làm sạch dữ liệu một cách hiệu quả..

2.5. Thực hành

Dựa trên bộ dữ liệu Mental_Health

2.5.1. Điền giá trị còn thiếu:

Thực hiện điền giá trị NULL cho các field theo quy ước như sau:

- *Occupation*: sử dụng giá trị mode.
- *Gender*: sử dụng giá trị của mục chọn có số lượng xuất hiện là ít nhất
- *Sleep_Hours*: sử dụng giá trị nhỏ nhất trong 3 giá trị: mean, median, mode
- *Mental_Health_Condition*: dựa vào số lần xuất hiện cao nhất của các nhóm được kết hợp bởi 2 thuộc tính "*Mental_Health_Condition*" "*Severity* “ (có 10 nhóm)
- *Work_Hours*: = *Work_Hours* trung bình của những người có cùng *Age*, *Gender*, *Occupation*

2.5. Thực hành

Dựa trên bộ dữ liệu *Mental_Health*

2.5.2. Làm mịn dữ liệu

Giả sử chia thuộc tính *Work_Hours* thành các nhóm: *30-39, 40-49, 50-59, 60-69, 70-80*. Lần lượt copy dữ liệu thành 3 bảng khác nhau (gọi là *X*, *Y* và *Z*). Đối với mỗi bảng, cần thực hiện định lại giá trị thuộc tính:

i. Bảng *X*: Thay thế bằng giá trị trung bình (mean) của nhóm.

ii. Bảng *Y*: Thay thế bằng giá trị trung vị (median) của nhóm.

iii. Bảng *Z*: Làm mịn theo ranh giới của nhóm (*smoothing by bin boundaries*)

2.5. Thực hành

Dựa trên bộ dữ liệu Mental_Health

2.5.3. Xác định giá trị ngoại lệ

- Xác định giá trị ngoại lệ cho tất cả các field kiểu số (nguyên, thực) có trong dữ liệu
- Đối với field *Work_Hours*, chuyển các giá trị outlier như sau:
  - Nếu giá trị <min => chuyển thành min
  - Nếu giá trị >max => chuyển thành max

2.5. Thực hành

Dựa trên bộ dữ liệu Mental_Health

2.5.4. Xử lý dữ liệu không nhất quán

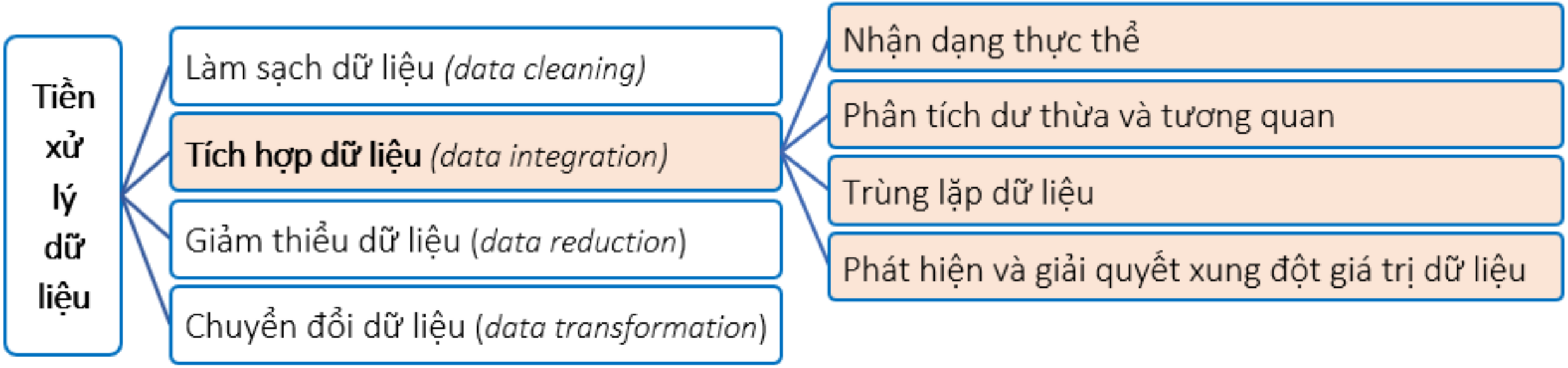
- Trong field *Occupation*, cùng ý nghĩa như nhau nhưng có 3 giá trị khác nhau là "*IT*", "*Information Technology*" và "*Information – Technology*". Thực hiện chuyển thống nhất thành "*IT*".
- Tương tự, field *Stress_Level*, cùng ý nghĩa như nhau nhưng có 2 trường hợp:
  - 2 giá trị khác nhau là "*Low*" và "*Sometimes*". Thực hiện chuyển thống nhất thành "*Low*".
  - 2 giá trị khác nhau là "*High*" và "*very stressful*". Thực hiện chuyển thống nhất thành "*High*".
- Field *Recording_date*: (nhắc lại, bộ dữ liệu này được ghi nhận trong tháng 3 năm 2023) kiểu dữ liệu ngày hiện tại không thống nhất và bị ghi nhầm năm nên đang được xem là kiểu chuỗi. Hảy chỉnh sửa để thống nhất định dạng thành m/d/yyyy. Sau đó chuyển kiểu dữ liệu của field thành ngày.

NỘI DUNG CHƯƠNG 4

1. Tổng quan về tiền xử lý dữ liệu
2. Làm sạch dữ liệu *(Data Cleaning)*
3. Tích hợp dữ liệu *(Data Integration)*
4. Giảm thiểu dữ liệu *(Data Reduction)*
5. Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*
6. Thực hành

3. TÍCH HỢP DỮ LIỆU (Data Integration)

Khai thác dữ liệu thường yêu cầu tích hợp dữ liệu (hợp nhất dữ liệu từ nhiều kho dữ liệu). Việc tích hợp cẩn thận có thể giúp giảm thiểu và tránh sự dư thừa cũng như sự không nhất quán trong tập dữ liệu thu được. Điều này có thể giúp cải thiện độ chính xác và tốc độ của quá trình khai thác dữ liệu tiếp theo.



3.1. Nhận dạng thực thể *(Entity Identification Problem)*

Làm cách nào để có thể khớp các thực thể trong thế giới thực tương đương từ nhiều nguồn dữ liệu? Điều này được gọi là vấn đề nhận dạng thực thể.

- *Tích hợp lược đồ* (*Schema integration*) và *kết hợp đối tượng* (*object matching*)

Ví dụ: làm thế nào nhà phân tích dữ liệu hoặc máy tính có thể chắc chắn rằng id khách hàng trong một cơ sở dữ liệu và mã số quản lý trong cơ sở dữ liệu khác tham chiếu đến cùng một thuộc tính?

Ví dụ về siêu dữ liệu (*metadata*) cho từng thuộc tính bao gồm tên, ý nghĩa, loại dữ liệu và phạm vi giá trị (*range of values*) được phép dùng cho thuộc tính đó cũng như các quy tắc null để xử lý các giá trị trống, 0 hoặc null. Siêu dữ liệu như vậy có thể được sử dụng để giúp tránh lỗi trong việc tích hợp lược đồ. Siêu dữ liệu cũng có thể được sử dụng để giúp chuyển đổi dữ liệu (ví dụ: trong đó mã dữ liệu cho loại thanh toán trong một cơ sở dữ liệu có thể là “H” và “S” nhưng lại có giá trị là 1 và 2 trong cơ sở dữ liệu khác). Do đó, bước này cũng liên quan đến việc làm sạch dữ liệu.

3.1. Nhận dạng thực thể (Entity Identification Problem)

- *Chú ý đến cấu trúc của dữ liệu trong quá trình tích hợp*

Khi khớp các thuộc tính từ cơ sở dữ liệu này với cơ sở dữ liệu khác trong quá trình tích hợp, phải đặc biệt chú ý đến cấu trúc của dữ liệu.

Điều này nhằm đảm bảo rằng mọi phụ thuộc chức năng của thuộc tính và các ràng buộc tham chiếu trong hệ thống nguồn đều khớp với các thuộc tính trong hệ thống đích.

Ví dụ: trong một hệ thống, chiết khấu có thể được áp dụng cho đơn đặt hàng, trong khi ở hệ thống khác, chiết khấu được áp dụng cho từng chi tiết đơn hàng riêng lẻ trong đơn đặt hàng. Nếu điều này không được phát hiện trước khi tích hợp, các vật phẩm trong hệ thống mục tiêu có thể bị giảm giá không đúng cách.

3.2. Phân tích dư thừa và tương quan *(Redundancy and Correlation Analysis)*

- *Dư thừa* (*Redundancy*) là một vấn đề quan trọng khác trong tích hợp dữ liệu:
  - Một thuộc tính (chẳng hạn như *Thành tiền*) có thể dư thừa nếu nó có thể được tính toán từ một hoặc tập hợp các thuộc tính khác.
  - Sự không nhất quán trong cách đặt tên thuộc tính hoặc chiều cũng có thể gây ra sự dư thừa trong tập dữ liệu kết quả.
- *Phân tích tương quan* (*correlation analysis*):
  - Một số dư thừa có thể được phát hiện bằng phân tích tương quan. Với hai thuộc tính, phân tích như vậy có thể đo lường mức độ ảnh hưởng của một thuộc tính đến thuộc tính kia dựa trên dữ liệu có sẵn.
  - Đối với dữ liệu danh nghĩa (*nominal data*): sử dụng phép kiểm tra χ^2 (chi-square).
  - Đối với các thuộc tính số: sử dụng hệ số tương quan (*correlation coefficient*) và hiệp phương sai (*covariance*), cả hai đều truy cập xem giá trị của một thuộc tính khác nhau như thế nào so với giá trị của thuộc tính khác.

3.2. Phân tích dư thừa và tương quan (Redundancy and Correlation Analysis)

3.2.1.Phép kiểm tra tương quan χ2 *(chi-square)* cho dữ liệu danh nghĩa

Đối với dữ liệu danh nghĩa, mối quan hệ tương quan giữa hai thuộc tính A và B có thể được phát hiện bằng phép kiểm tra χ^2. Giả sử A có c giá trị phân biệt, cụ thể là $a_1, a_2, ..., a_c$. B có r giá trị riêng biệt, cụ thể là $b_1, b_2, ..., b_r$. Các bộ dữ liệu được mô tả bởi A và B có thể được hiển thị dưới dạng bảng thống kê, với các giá trị c của A tạo thành các cột và giá trị r của B tạo thành các hàng. Gọi (A_i, B_j) biểu thị sự kiện chung mà thuộc tính A nhận giá trị a_i và thuộc tính B nhận giá trị b_j, nghĩa là trong đó $(A = a_i, B = b_j)$. Mọi cặp giá trị chung có thể có (A_i, B_j) đều có ô (hoặc vị trí) riêng trong bảng. Giá trị χ^2 (còn được gọi là thống kê *Pearson* χ^2) được tính như sau:

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$ *Công thức 3-1*

trong đó:
- o_{ij} là tần suất quan sát được (tức là số lượng thực tế) của sự kiện chung (A_i, B_j).
- e_{ij} là tần suất dự kiến của (A_i, B_j), có thể được tính như sau

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_i)}{n}$$ *Công thức 3-2*

Với
- n là số bộ dữ liệu;
- $count(A = ai)$ là số bộ có giá trị ai cho A;
- $count(B = bj)$ là số bộ có giá trị bj cho B.

3.2. Phân tích dư thừa và tương quan (Redundancy and Correlation Analysis)

3.2.1.Phép kiểm tra tương quan χ2 (chi-square) cho dữ liệu danh nghĩa

Ví dụ: Cho số liệu khảo sát trên 1500 người (nam+nữ) về việc thăm dò xem loại truyện ưa thích của họ là hư cấu hay phi hư cấu. Số lượng quan sát được tóm tắt trong bảng bên

	male	*female*	*Total*
fiction	250	200	450
non_fiction	50	1000	1050
Total	300	1200	1500

Sử dụng công thức (2), tính được tần suất dự kiến cho từng ô.

$$e_{11} = \frac{count(male) \times count(fiction)}{n} = \frac{300 \times 450}{1500} = 90$$

$$e_{12} = \frac{count(female) \times count(fiction)}{n} = \frac{1200 \times 450}{1500} = 360$$

$$e_{21} = \frac{count(male) \times count(non\_fiction)}{n} = \frac{300 \times 1050}{1500} = 210$$

$$e_{22} = \frac{count(female) \times count(non\_fiction)}{n} = \frac{1200 \times 1050}{1500} = 840$$

	male		*female*		*Total*
fiction	250	(90)	200	(360)	450
non_fiction	50	(210)	1000	(840)	1050
Total	300		1200		1500

Sử dụng công thức (1) để tính χ^2, ta có

$$\chi^2 = \sum_{i=1}^{c}\sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}} = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840}$$

$$= 284.44 + 121.90 + 71.11 + 30.48 = 507.93$$

3.2.1.Phép kiểm tra tương quan χ2 (chi-square) cho dữ liệu danh nghĩa

Đối với bảng 2×2 này, bậc tự do (*degrees of freedom*) là (2−1)(2−1) = 1.

Đối với 1 bậc tự do, giá trị χ^2 cần thiết để bác bỏ giả thuyết ở mức ý nghĩa 0,001 là 10.828 (lấy từ bảng điểm phần trăm trên của phân bố χ^2)

Vì giá trị tính toán của ta (507.93) cao hơn giá trị này nên ta có thể bác bỏ giả thuyết rằng giới tính và cách đọc ưa thích là độc lập ⇒ kết luận rằng hai thuộc tính này có mối tương quan (mạnh mẽ) đối với một nhóm người nhất định.

Critical values of the Chi-square distribution with *d* degrees of freedom

	Probability of exceeding the critical value						
d	0.05	0.01	0.001	*d*	0.05	0.01	0.001
1	3.841	6.635	10.828	11	19.675	24.725	31.264
2	5.991	9.210	13.816	12	21.026	26.217	32.910
3	7.815	11.345	16.266	13	22.362	27.688	34.528
4	9.488	13.277	18.467	14	23.685	29.141	36.123
5	11.070	15.086	20.515	15	24.996	30.578	37.697
6	12.592	16.812	22.458	16	26.296	32.000	39.252
7	14.067	18.475	24.322	17	27.587	33.409	40.790
8	15.507	20.090	26.125	18	28.869	34.805	42.312
9	16.919	21.666	27.877	19	30.144	36.191	43.820
10	18.307	23.209	29.588	20	31.410	37.566	45.315

3.2.2.Phép kiểm tra tương quan χ2 *(chi-square)* cho dữ liệu kiểu số

Có thể đánh giá mối tương quan giữa hai thuộc tính A và B bằng cách tính hệ số tương quan (còn được gọi là *hệ số mômen sản phẩm Pearson - Pearson's product moment coefficient*) theo công thức sau:

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i-\bar{A})(b_i-\bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_ib_i)-n\bar{A}\bar{B}}{n\sigma_A\sigma_B} \qquad \textit{Công thức 3}$$

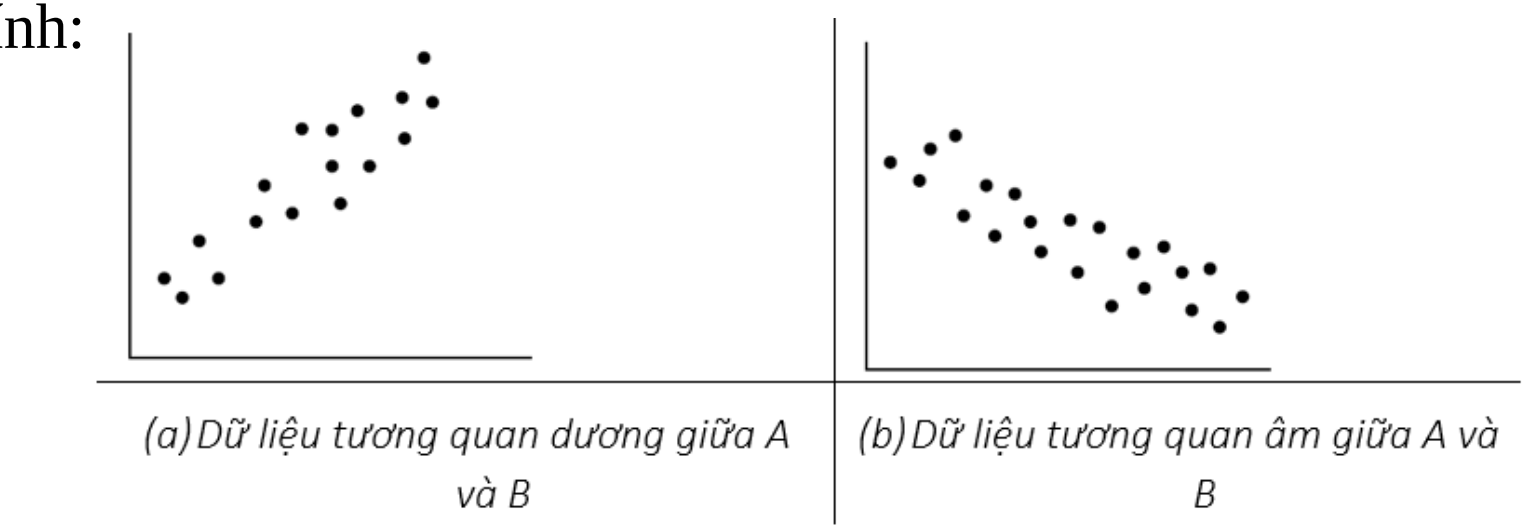
trong đó
- n là số bộ dữ liệu.
- a_i và b_i là giá trị tương ứng của A và B trong bộ i,
- $\bar{A}$ và $\bar{B}$ là giá trị trung bình tương ứng của A và B
- σ_A và σ_B là độ lệch chuẩn (*standard deviations*) tương ứng của A và B.
- $\Sigma(a_ib_i)$ là tổng của các tích giữa A và B (tức là, đối với mỗi bộ dữ liệu, giá trị của A được nhân với giá trị của B trong bộ dữ liệu đó).

Lưu ý rằng $-1 \le r_{A,B} \le +1$. Nếu:

- $r_{A,B} > 0$: A và B có mối tương quan dương, nghĩa là giá trị của A tăng khi giá trị của B tăng.
- $r_{A,B} = 0$: A và B độc lập và không có mối tương quan giữa chúng.
- $r_{A,B} < 0$: A và B có mối tương quan nghịch, trong đó giá trị của một thuộc tính tăng khi giá trị của thuộc tính kia giảm.

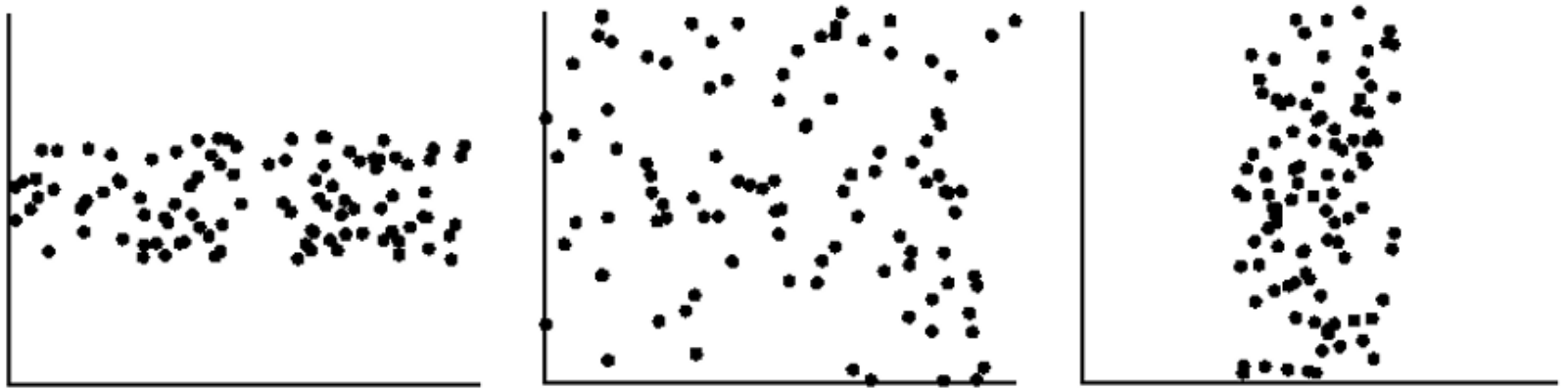
3.2.2.Phép kiểm tra tương quan χ2 (chi-square) cho dữ liệu kiểu số

Biểu đồ phân tán (*Scatter plots*) cũng có thể được sử dụng để xem mối tương quan giữa các thuộc tính:



(a) Dữ liệu tương quan dương giữa A và B

(b) Dữ liệu tương quan âm giữa A và B



(c) Dữ liệu A và B không tương quan

Sử dụng Scatter plots để biểu diễn mối tương quan giữa các thuộc tính.

3.2.3.Hiệp phương sai của dữ liệu số (Covariance of Numeric Data)

Trong lý thuyết xác suất và thống kê, mối tương quan (*correlation*) và hiệp phương sai (*covariance*) là hai thước đo tương tự nhau để đánh giá mức độ thay đổi của hai thuộc tính cùng nhau. Xét hai thuộc tính số *A* và *B* và một tập hợp *n* quan sát $\{(a_1, b_1), \ldots, (a_n, b_n)\}$. Giá trị trung bình của *A* và *B* tương ứng còn được gọi là giá trị kỳ vọng (*expected values*) của *A* và *B*, nghĩa là

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n} \quad \text{và} \quad E(B) = \bar{B} = \frac{\sum_{i=1}^{n} b_i}{n}$$

Hiệp phương sai giữa A và B được định nghĩa theo công thức sau:

$$Cov(A,B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})-(b_i - \bar{B})}{n} \qquad \textit{Công thức 3-4}$$

Nếu so sánh công thức (3) đối với $r_{A,B}$ (hệ số tương quan), với công thức (4) đối với hiệp phương sai, ta thấy rằng

$$r_{A,B} = \frac{Cov(A,B)}{\sigma_A \sigma_B} \qquad \textit{Công thức 3-5}$$

trong đó σA và σB lần lượt là độ lệch chuẩn của A và B. Nó cũng có thể được chỉ ra rằng

$$Cov(A,B) = E(A \times B) - \bar{A}\bar{B} \qquad \textit{Công thức 3-6}$$

Phương trình này có thể đơn giản hóa việc tính toán.

3.2.3.Hiệp phương sai của dữ liệu số (Covariance of Numeric Data)

Đối với hai thuộc tính A và B có xu hướng thay đổi cùng nhau, nếu A lớn hơn $\bar{A}$ (giá trị kỳ vọng của A) thì B có khả năng lớn hơn $\bar{B}$ (giá trị kỳ vọng của B). Do đó, hiệp phương sai giữa A và B là dương. Mặt khác, nếu một trong các thuộc tính có xu hướng cao hơn giá trị mong đợi của nó trong khi thuộc tính kia thấp hơn giá trị mong đợi của nó thì hiệp phương sai của A và B là âm.

Nếu A và B độc lập (*independent* - tức là chúng không có tương quan) thì:

$$E(A \times B) = E(A) \times E(B)$$

Do đó, hiệp phương sai là $Cov(A, B) = E(A \times B) - \bar{A} \times \bar{B} = E(A) \times E(B) - \bar{A} \times \bar{B} = 0$.

Tuy nhiên, điều ngược lại là không đúng. Một số cặp biến ngẫu nhiên (thuộc tính) có thể có hiệp phương sai bằng 0 nhưng không độc lập. Chỉ theo một số giả định bổ sung

3.2.3.Hiệp phương sai của dữ liệu số (Covariance of Numeric Data)

Ví dụ 3-2: Xem xét Bảng 2, trình bày một ví dụ đơn giản về giá cổ phiếu được quan sát tại năm thời điểm của 2 công ty ABC (gọi tắt là A) và XYZ (gọi tắt là X). Nếu các cổ phiếu bị ảnh hưởng bởi cùng xu hướng của ngành, giá của chúng sẽ tăng hay giảm cùng nhau?

·Giá cổ phiếu của 2 công ty ABC và XYZ

Time point	*ABC*	*XYZ*
t1	6	20
t2	5	10
t3	4	14
t4	3	5
t5	2	5

$$E(A) = \bar{A} = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{6+5+4+3+2}{5} = \frac{20}{5} = 4$$

và $$E(X) = \bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} = \frac{20+10+14+5+5}{5} = \frac{54}{5} = 10.8$$

Sử dụng công thức 6, ta có:

$$\text{Cov(A, X)} = E(A \times X) - (\bar{A} \times \bar{X})$$
$$= \frac{(6 \times 20) + (5 \times 10) + (4 \times 14) + (3 \times 5) + (2 \times 5)}{5} - (4 \times 10.8)$$
$$= 50.2 - 43.2 = 7$$

$\Rightarrow$ với hiệp phương sai dương, có thể nói rằng giá cổ phiếu của cả hai công ty đều tăng cùng nhau.

3.3. Trùng lặp dữ liệu *(Tuple Duplication)*

- Là trùng lặp giữa các dòng trong dữ liệu, ví dụ: khi có hai hoặc nhiều bộ dữ liệu giống hệt nhau cho một trường hợp nhập dữ liệu duy nhất.
- Việc sử dụng các bảng không chuẩn hóa (thường được thực hiện để cải thiện hiệu suất bằng cách tránh các phép nối) là một trường hợp khác. Sự không nhất quán thường phát sinh giữa các bản sao khác nhau, do nhập dữ liệu không chính xác hoặc cập nhật một số lần nhưng không phải tất cả dữ liệu. Ví dụ: nếu cơ sở dữ liệu đơn đặt hàng chứa các thuộc tính cho tên và địa chỉ của người mua thay vì khóa thông tin này trong cơ sở dữ liệu người mua thì có thể xảy ra sự khác biệt, chẳng hạn như tên của cùng một người mua xuất hiện với các địa chỉ khác nhau trong cơ sở dữ liệu đơn đặt hàng.

3.4. Phát hiện và giải quyết xung đột giá trị dữ liệu

- Là trường hợp đối với cùng một thực thể trong thế giới thực, các giá trị thuộc tính từ các nguồn khác nhau có thể khác nhau. Điều này có thể là do sự khác biệt trong cách trình bày, chia tỷ lệ hoặc mã hóa.
- Một số ví dụ:
  - Thuộc tính đơn vị tính trong hệ thống X sử dụng là Kg, hệ thống Y sử dụng là pounds.
  - Đối với một chuỗi khách sạn, giá phòng ở các thành phố khác nhau có thể không chỉ liên quan đến các loại tiền tệ khác nhau mà còn liên quan đến các dịch vụ khác nhau (ví dụ: bữa sáng miễn phí) và thuế.

3.4. Phát hiện và giải quyết xung đột giá trị dữ liệu

- Một số ví dụ:
  - Khi trao đổi thông tin giữa các trường học, mỗi trường có thể có chương trình giảng dạy và hệ thống chấm điểm riêng. Một trường đại học X có thể áp dụng hệ thống 4 học kỳ/ năm học, và chấm điểm theo thang điểm từ A+ đến F, trong khi trường đại học Y có thể áp dụng hệ thống 2 học kỳ/năm học và chấm điểm theo thang điểm từ 1 đến 10. Rất khó để làm việc đưa ra các quy tắc chuyển đổi chính xác từ khóa học sang cấp lớp giữa hai trường đại học, khiến việc trao đổi thông tin trở nên khó khăn.
  - Các thuộc tính khác nhau về mức độ trừu tượng: trong đó một thuộc tính trong một hệ thống được ghi lại ở mức độ trừu tượng thấp hơn thuộc tính "tương tự" trong một hệ thống khác. Ví dụ: trường A xếp loại học sinh theo các mức: xuất sắc, giỏi, khá, trung bình, kém, *yếu*; nhưng trường Y lại xếp loại học sinh theo các mức: xuất sắc, giỏi, khá, *trung bình-khá*, trung bình, kém.

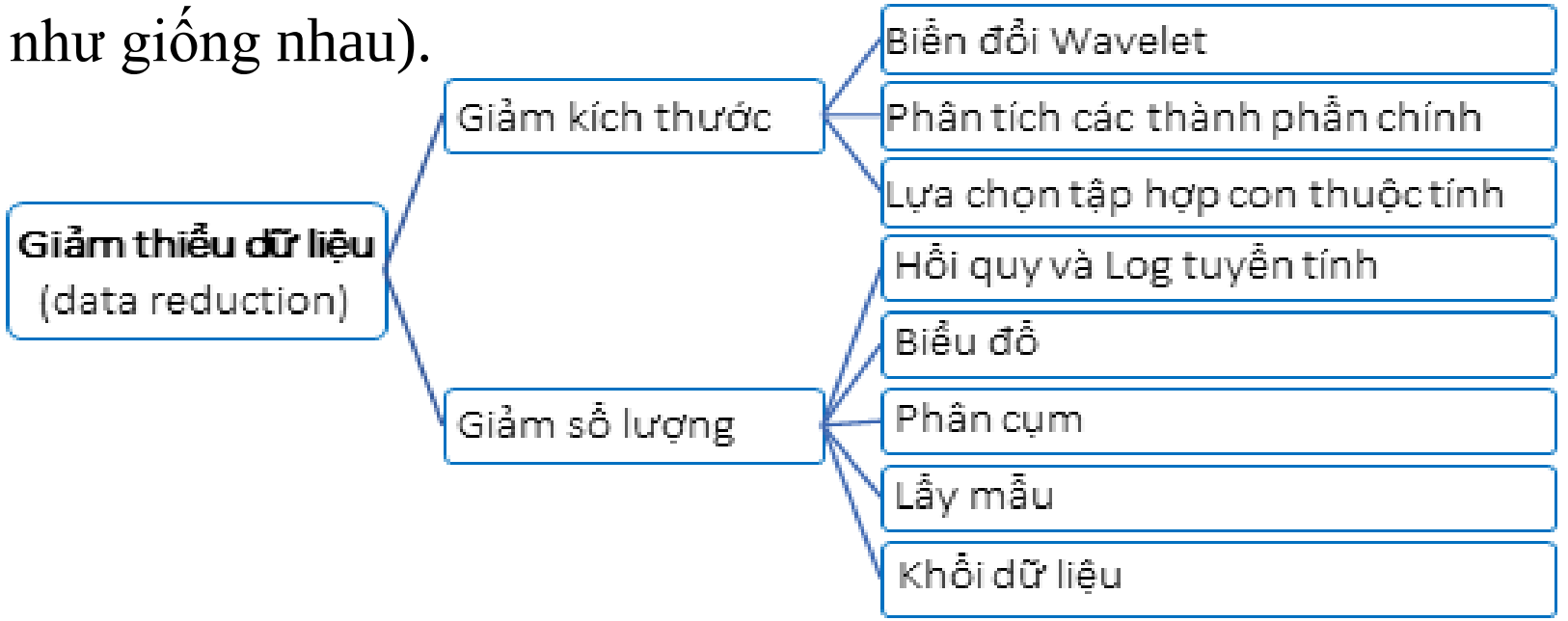
NỘI DUNG CHƯƠNG 4

1. Tổng quan về tiền xử lý dữ liệu
2. Làm sạch dữ liệu *(Data Cleaning)*
3. Tích hợp dữ liệu *(Data Integration)*
4. Giảm thiểu dữ liệu *(Data Reduction)*
5. Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*
6. Thực hành

4. GIẢM THIỂU DỮ LIỆU *(Data Reduction)*

Hãy tưởng tượng rằng bạn đã chọn dữ liệu từ kho dữ liệu của công ty để phân tích. Tập dữ liệu có thể sẽ rất lớn! Việc phân tích và khai thác dữ liệu phức tạp trên lượng dữ liệu khổng lồ có thể mất nhiều thời gian, khiến việc phân tích đó trở nên không thực tế hoặc không khả thi.

Các kỹ thuật giảm dữ liệu có thể được áp dụng để thu được biểu diễn giảm của tập dữ liệu có khối lượng nhỏ hơn nhiều nhưng vẫn duy trì chặt chẽ tính toàn vẹn của dữ liệu gốc. Nghĩa là, việc khai thác trên tập dữ liệu rút gọn sẽ hiệu quả hơn nhưng vẫn tạo ra kết quả phân tích giống nhau (hoặc gần như giống nhau).



4.1. Giảm kích thước (Dimensionality reduction)

4.1.1. Kỹ thuật biến đổi Wavelet (Wavelet Transforms)

- Biến đổi *wavelet* (*discrete wavelet transform - DWT*) là một kỹ thuật xử lý tín hiệu tuyến tính, khi áp dụng cho vectơ dữ liệu *X*, sẽ biến đổi nó thành một vectơ khác (X_0). Khi áp dụng kỹ thuật này để giảm dữ liệu, mỗi bộ dữ liệu sẽ được xem là một vectơ dữ liệu X gồm ***n*** chiều ($X = (x_1, x_2, ..., x_n)$), mô tả *n* phép đo được thực hiện trên bộ dữ liệu từ *n* thuộc tính cơ sở dữ liệu.
- Sự hữu ích của dữ liệu biến đổi *wavelet* là có thể được cắt bớt. Dữ liệu gần đúng đã được nén có thể được giữ lại bằng cách chỉ lưu trữ một phần nhỏ hệ số sóng con mạnh nhất. Ví dụ: tất cả các hệ số *wavelet* lớn hơn một số ngưỡng do người dùng chỉ định có thể được giữ lại. Tất cả các hệ số khác được đặt thành *0*. Do đó, việc biểu diễn dữ liệu thu được rất thưa thớt, nhờ vậy các thao tác có thể tận dụng tính thưa thớt của dữ liệu sẽ được tính toán rất nhanh nếu được thực hiện trong không gian *wavelet*. Kỹ thuật này cũng có tác dụng loại bỏ nhiễu mà không làm mịn các tính năng chính của dữ liệu, giúp việc làm sạch dữ liệu trở nên hiệu quả.

4. Giảm thiểu dữ liệu (Data Reduction)

4.1. Giảm kích thước (Dimensionality reduction)

4.1.1. Kỹ thuật biến đổi Wavelet (Wavelet Transforms)

Phương pháp Wavelet:

i. Độ dài L của vectơ dữ liệu đầu vào phải có giá trị là lũy thừa nguyên của 2. Điều kiện này có thể được đáp ứng bằng cách đệm vectơ dữ liệu bằng các số 0 nếu cần ($L \geq n$).

ii. Mỗi phép biến đổi liên quan đến việc áp dụng hai hàm.

- Hàm đầu tiên áp dụng một số thao tác làm mịn dữ liệu, chẳng hạn như tính tổng hoặc trung bình có trọng số.
- Hàm thứ hai thực hiện sự khác biệt có trọng số, có tác dụng làm nổi bật các tính năng chi tiết của dữ liệu.

iii. Hai hàm này được áp dụng cho các cặp điểm dữ liệu trong X, nghĩa là cho tất cả các cặp số đo (x_{2i}, x_{2i+1}). Điều này dẫn đến hai bộ dữ liệu có độ dài $L/2$. Nói chung, chúng thể hiện phiên bản được làm mịn (*smoothed*) hoặc tần suất thấp (*low-frequency*) của dữ liệu đầu vào và nội dung tần suất cao (*high-frequency*) của nó tương ứng.

iv. Hai hàm này được áp dụng đệ quy cho các tập dữ liệu thu được ở vòng lặp trước, cho đến khi các tập dữ liệu thu được có độ dài 2.

v. Các giá trị được chọn từ các tập dữ liệu thu được trong các lần lặp trước được chỉ định là hệ số *wavelet* của dữ liệu được chuyển đổi.

4.1.2. Phân tích các thành phần chính *(Principal Components Analysis - PCA)*

Quy trình cơ bản:

i. Dữ liệu đầu vào được chuẩn hóa để mỗi thuộc tính nằm trong cùng một phạm vi. Bước này giúp đảm bảo rằng các thuộc tính có miền giá trị lớn sẽ không lấn át các thuộc tính có miền giá trị nhỏ hơn.

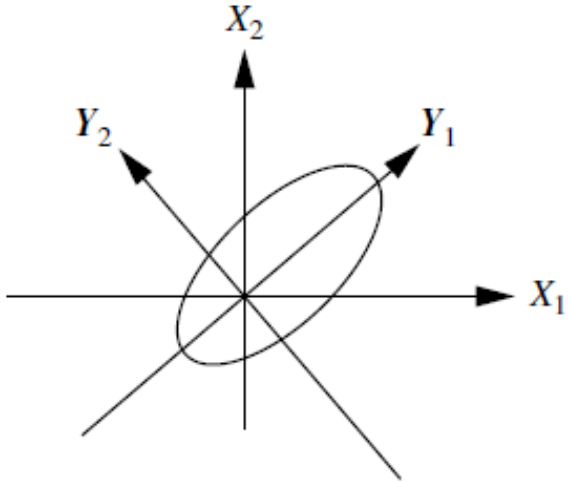
ii. *PCA* tính toán *k* vectơ trực giao (*orthonormal vectors*) làm cơ sở cho dữ liệu đầu vào được chuẩn hóa. Đây là các vectơ đơn vị mà mỗi vectơ hướng vuông góc với các vectơ khác. Các vectơ này được gọi là các thành phần chính (*principal components*). Dữ liệu đầu vào là sự kết hợp tuyến tính của các thành phần chính.

4.1. Giảm kích thước (Dimensionality reduction)

4.1.2. Phân tích các thành phần chính (Principal Components Analysis - PCA)

Quy trình cơ bản:

iii. Các thành phần chính được sắp xếp giảm dần theo mức độ quan trọng của "*ý nghĩa*" hoặc "*độ mạnh*". Các thành phần chính về cơ bản đóng vai trò như một bộ trục mới cho dữ liệu, cung cấp thông tin quan trọng về phương sai. Nghĩa là, các trục được sắp xếp sao cho trục đầu tiên hiển thị phương sai lớn nhất trong số dữ liệu, trục thứ hai hiển thị phương sai cao nhất tiếp theo, v.v. Ví dụ, Hình sau cho thấy hai thành phần chính đầu tiên, Y_1 và Y_2, đối với tập hợp dữ liệu đã cho ban đầu được ánh xạ tới trục X_1 và X_2. Thông tin này giúp xác định các nhóm hoặc mẫu trong dữ liệu.



4.1.2. Phân tích các thành phần chính (*Principal Components Analysis - PCA*)

Quy trình cơ bản:

iv. Do các thành phần được sắp xếp theo thứ tự giảm dần về "mức độ quan trọng" (*significance*) nên kích thước dữ liệu có thể giảm bằng cách loại bỏ các thành phần yếu hơn (*weaker components*), tức là những thành phần có phương sai thấp. Bằng cách sử dụng các thành phần chính mạnh nhất, có thể xây dựng lại dữ liệu gốc gần đúng.

4.1. Giảm kích thước (Dimensionality reduction)

4.1.2. Phân tích các thành phần chính (Principal Components Analysis - PCA)

- *PCA* có thể được áp dụng cho:
  - Các thuộc tính có thứ tự và không có thứ tự
  - xử lý dữ liệu thưa thớt (*sparse data*)
  - dữ liệu sai lệch (*skewed data*).
  - Dữ liệu đa chiều (*Multidimensional data* - có nhiều hơn hai chiều) có thể được xử lý bằng cách giảm số chiều xuống còn hai chiều.
- Các thành phần chính có thể được sử dụng làm đầu vào cho phân tích cụm (*multiple regression*) và hồi quy bội (*multiple regression*).
- So với các phép biến đổi *wavelet*, *PCA* có xu hướng xử lý dữ liệu thưa thớt tốt hơn, trong khi các phép biến đổi *wavelet* phù hợp hơn với dữ liệu có nhiều chiều.

4.1.3. *Lựa chọn tập hợp con thuộc tính* (Attribute Subset Selection)

- Mục tiêu của việc lựa chọn tập hợp con thuộc tính là tìm một tập hợp các thuộc tính tối thiểu sao cho phân bố xác suất thu được của các lớp dữ liệu càng gần với phân bố ban đầu thu được bằng cách sử dụng tất cả các thuộc tính.
- Lợi ích của việc khai thác trên một tập hợp thuộc tính đã rút gọn:
  - Giảm số lượng thuộc tính xuất hiện trong các mẫu được phát hiện, giúp làm cho các mẫu dễ hiểu, giảm sự nhầm lẫn cho thuật toán khai thác được sử dụng. Nói cách khác là tang chất lượng của các mẫu.
  - Giảm thời gian khai thác dữ liệu (nhanh hơn).

4.1. Giảm kích thước (Dimensionality reduction)

4.1.3. Lựa chọn tập hợp con thuộc tính (Attribute Subset Selection)

- "Làm thế nào ta có thể tìm được tập hợp con 'tốt' của các thuộc tính ban đầu?"
  - Với n thuộc tính, có thể có 2^n tập con. Việc tìm kiếm toàn diện tập hợp con các thuộc tính tối ưu có thể cực kỳ tốn kém, đặc biệt khi n và số lượng lớp dữ liệu tăng lên.
  - Mặc dù chuyên gia về lĩnh vực có thể chọn ra một số thuộc tính hữu ích nhưng đây có thể là một nhiệm vụ khó khăn và tốn thời gian, đặc biệt khi ý nghĩa của dữ liệu chưa được biết rõ.
  - Do đó, các phương pháp heuristic khám phá không gian tìm kiếm rút gọn thường được sử dụng để lựa chọn tập hợp con thuộc tính. Các phương pháp này thường tham lam (*greedy*) ở chỗ, trong khi tìm kiếm trong không gian thuộc tính, chúng luôn đưa ra những gì có vẻ là lựa chọn tốt nhất vào thời điểm đó. Chiến lược đó nhằm đưa ra lựa chọn tối ưu cục bộ với hy vọng rằng điều này sẽ dẫn đến giải pháp tối ưu toàn cục. Những phương pháp tham lam như vậy có hiệu quả trong thực tế và có thể tiến gần đến việc ước tính một giải pháp tối ưu.

4.1. Giảm kích thước (Dimensionality reduction)

4.1.3. Lựa chọn tập hợp con thuộc tính (Attribute Subset Selection)

- ***Phương pháp tham lam*** (*greedy-heuristic*)
  - Các thuộc tính "tốt nhất" (*best* và "tệ nhất" - *worst*) thường được xác định bằng cách sử dụng các thử nghiệm có ý nghĩa thống kê, giả định rằng các thuộc tính này độc lập với nhau. Nhiều thước đo đánh giá thuộc tính khác có thể được sử dụng như thước đo thu được thông tin (*information gain measure*) được sử dụng trong việc xây dựng cây quyết định để phân loại.

4.1. Giảm kích thước (Dimensionality reduction)

4.1.3. Lựa chọn tập hợp con thuộc tính (Attribute Subset Selection)

- ***Phương pháp tham lam*** (*greedy-heuristic*)

  Gồm các kỹ thuật:

  i. *Forward selection* (*Lựa chọn tiến*): Quy trình bắt đầu với một tập thuộc tính trống (rỗng) là tập rút gọn. Thuộc tính tốt nhất ban đầu được xác định và thêm vào tập rút gọn. Ở mỗi bước lặp hoặc bước tiếp theo, thuộc tính gốc tốt nhất còn lại sẽ được thêm vào tập hợp.

  ii. *Backward elimination* (*Loại bỏ lùi*): Quy trình bắt đầu với tập hợp đầy đủ các thuộc tính. Ở mỗi bước, sẽ loại bỏ thuộc tính xấu nhất còn lại trong tập hợp.

  iii. *Combination of forward selection and backward elimination* (Kết hợp lựa chọn tiến và loại bỏ lùi): ở mỗi bước, quy trình sẽ chọn thuộc tính tốt nhất và loại bỏ thuộc tính xấu nhất trong số các thuộc tính còn lại.

4.1. Giảm kích thước (Dimensionality reduction)

4.1.3. Lựa chọn tập hợp con thuộc tính (Attribute Subset Selection)

- ***Phương pháp tham lam*** (*greedy-heuristic*)

  Gồm các kỹ thuật:

  iv. *Decision tree induction* (*Tạo cây quyết định*):

  - Các thuật toán cây quyết định (ví dụ: ID3, C4.5 và CART) ban đầu được dùng để phân loại.
  - Quy nạp cây quyết định xây dựng một cấu trúc giống như sơ đồ trong đó mỗi nút bên trong (không phải nút lá) biểu thị một thử nghiệm trên một thuộc tính, mỗi nhánh tương ứng với một kết quả của thử nghiệm và mỗi nút (lá) bên ngoài biểu thị một dự đoán lớp. Tại mỗi nút, thuật toán chọn thuộc tính "tốt nhất" để phân chia dữ liệu thành các lớp riêng lẻ.
  - Tất cả các thuộc tính không xuất hiện trong cây được coi là không liên quan. Tập hợp các thuộc tính xuất hiện trên cây tạo thành tập con rút gọn của các thuộc tính.

4.1. *Giảm kích thước* (Dimensionality reduction)

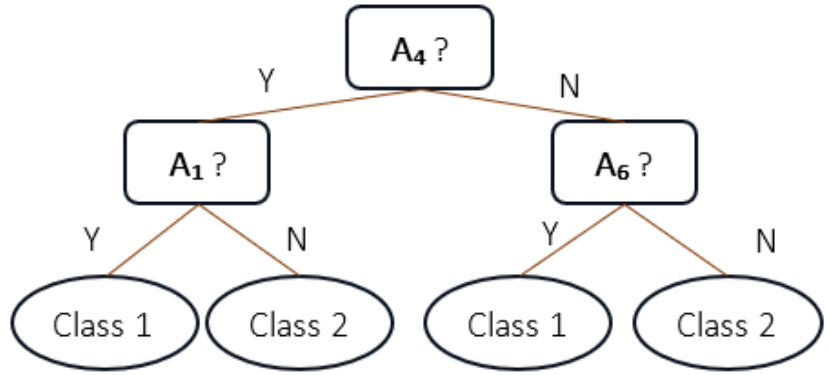
4.1.3. *Lựa chọn tập hợp con thuộc tính* (Attribute Subset Selection)

- ***Phương pháp tham lam*** (*greedy-heuristic*)

Forward selection	*Backward elimination*	*Combination of forward & backward*
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ ***Initial reduced set***: {} => $\{A_1\}$ => $\{A_1, A_4\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	***Initial attribute set***: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => ***Reduced attribute set***: $\{A_1, A_4, A_6\}$	***Initial attribute set***: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ Step 1 => $\{A_1\}$; $\{A_2, A_3, A_4, A_6\}$ Step 2 => $\{A_1, A_4\}$; $\{A_2, A_6\}$ Step 3 => $\{A_1, A_4, A_6\}$; { } => ***Reduced attribute set***: $\{A_1, A_4, A_6\}$

Decision tree induction

Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$



=> ***Reduced attribute set***: $\{A_1, A_4, A_6\}$

4.2. Giảm số lượng (Numerosity reduction)

- Các kỹ thuật giảm số lượng thay thế khối lượng dữ liệu gốc bằng các dạng biểu diễn dữ liệu thay thế, nhỏ hơn.
- Những kỹ thuật này có thể là tham số hoặc không tham số. Đối với các phương pháp tham số, một mô hình được sử dụng để ước tính dữ liệu, do đó, thông thường chỉ cần lưu trữ các tham số dữ liệu thay vì dữ liệu thực tế (các giá trị ngoại lệ cũng có thể được lưu trữ).

4.2. Giảm số lượng (Numerosity reduction)

4.2.1. Mô hình hồi quy và Log-tuyến tính: Giảm dữ liệu tham số (Regression and Log-Linear Models: Parametric Data Reduction)

- ***Mô hình hồi quy** (Regression Models)*
  - ***Hồi quy tuyến tính*** (*linear regression*): dữ liệu được mô hình hóa để khớp với một phương trình đường thẳng, với phương trình:

$$y = wx + b$$

trong đó

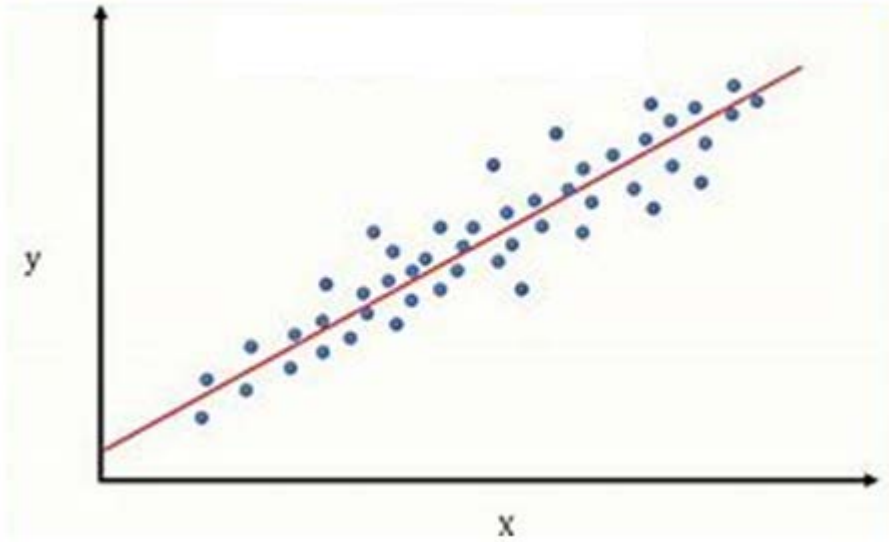
- y: được gọi là *biến phản hồi - response variable*), có thể được mô hình hóa dưới dạng hàm tuyến tính của một biến ngẫu nhiên khác
- x: (được gọi là biến dự đoán - *predictor variable*)
- Phương sai (*variance*) của y được coi là không đổi (*constant*).
- Trong bối cảnh khai thác dữ liệu, x và y là các thuộc tính cơ sở dữ liệu số. Các hệ số w và b (được gọi là hệ số hồi quy - *regression coefficients*), chỉ định độ dốc của đường thẳng và điểm chặn y tương ứng.

Các hệ số này có thể giải bằng phương pháp bình phương tối thiểu (*method of least squares*), giúp giảm thiểu sai số giữa đường thực tế phân tách dữ liệu (*the actual line separating the data*) và ước lượng của đường thẳng (*the estimate of the line*).

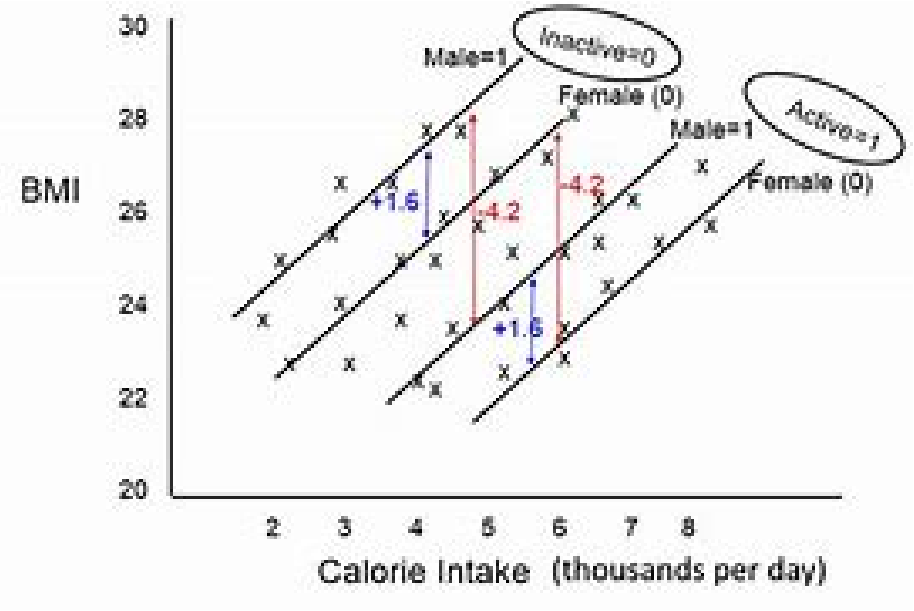
4.2. Giảm số lượng (Numerosity reduction)

4.2.1. Mô hình hồi quy và Log-tuyến tính: Giảm dữ liệu tham số (Regression and Log-Linear Models: Parametric Data Reduction)

- ***Mô hình hồi quy** (Regression Models)*
  - ***Hồi quy tuyến tính bội*** (*Multiple linear regression*): là phần mở rộng của hồi quy tuyến tính (đơn giản), cho phép một biến phản hồi, *y*, được mô hình hóa như một hàm tuyến tính (*linear function*) của hai hoặc nhiều biến dự đoán (*predictor variables*).



Simple Linear Regression



Multi Regression Linear

4.2. Giảm số lượng (Numerosity reduction)

4.2.1. Mô hình hồi quy và Log-tuyến tính

- ***Mô hình log-tuyến tính** (Log-linear model)*
  - Cho một tập hợp các bộ dữ liệu có *n* chiều (ví dụ: được mô tả bởi *n* thuộc tính), có thể coi mỗi bộ dữ liệu là một điểm trong không gian *n* chiều. Các mô hình log-tuyến tính có thể được sử dụng để ước tính xác suất của từng điểm trong không gian đa chiều cho một tập hợp các thuộc tính rời rạc, dựa trên một tập hợp con nhỏ hơn của các tổ hợp chiều (*dimensional combinations*). Điều này cho phép không gian dữ liệu có chiều cao hơn (*higher-dimensional data space*) được xây dựng từ không gian có chiều thấp hơn (*lower-dimensional spaces*).
  - Các mô hình log-tuyến tính cũng hữu ích cho việc:
    - *Giảm kích thước* (vì các điểm có chiều thấp hơn thường chiếm ít không gian hơn các điểm dữ liệu ban đầu).
    - *Làm mịn dữ liệu* (*data smoothing* - vì các ước tính tổng hợp trong không gian có chiều thấp hơn ít chịu các biến thể lấy mẫu hơn các ước tính trong không gian nhiều chiều hơn).

4.2. Giảm số lượng (Numerosity reduction)

4.2.1. Mô hình hồi quy và Log-tuyến tính

- Cả mô hình hồi quy và log-tuyến tính đều có thể được sử dụng trên dữ liệu thưa thớt, mặc dù ứng dụng của chúng có thể bị hạn chế.
- Mặc dù cả hai phương pháp đều có thể xử lý dữ liệu sai lệch (*skewed data*), nhưng hồi quy lại hoạt động rất tốt. Hồi quy có thể cần nhiều tính toán khi áp dụng cho dữ liệu nhiều chiều, trong khi các mô hình log-tuyến tính cho thấy khả năng mở rộng tốt cho tối đa 10 chiều.
- Một số gói phần mềm để giải quyết các vấn đề hồi quy:
  - SAS (www.sas.com)
  - SPSS (www.spss.com)
  - S-Plus (www.insightful.com).

4.2. Giảm số lượng (Numerosity reduction)

4.2.2. Biểu đồ (Histograms)

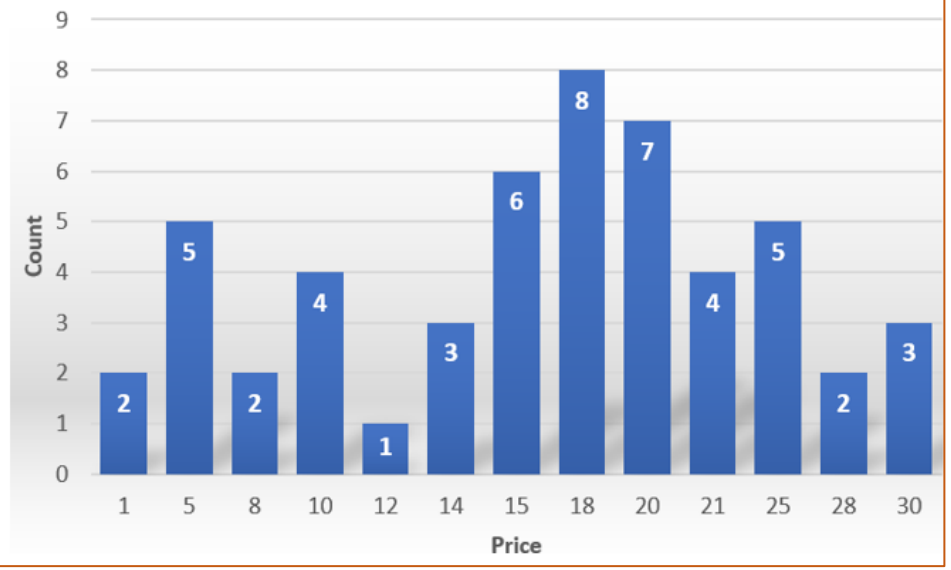
- Biểu đồ sử dụng tính năng tạo nhóm để phân phối dữ liệu gần đúng và là một hình thức giảm dữ liệu phổ biến.
- Biểu đồ có hiệu quả cao trong việc xấp xỉ cả dữ liệu thưa thớt và dữ liệu dày đặc, cũng như dữ liệu có độ lệch cao và đồng nhất.
- Biểu đồ được mô tả trước đây cho các thuộc tính đơn lẻ có thể được mở rộng cho nhiều thuộc tính. Biểu đồ đa chiều có thể nắm bắt được sự phụ thuộc giữa các thuộc tính. Những biểu đồ này tỏ ra có hiệu quả trong việc xấp xỉ dữ liệu với tối đa năm thuộc tính.
- Biểu đồ cho thuộc tính A, phân vùng phân phối dữ liệu của A thành các tập con rời rạc, được gọi là buckets (nhóm) hoặc bins (thùng).
- Phân loại nhóm:
  - Nhóm đơn: Nếu mỗi nhóm chỉ đại diện cho một cặp thuộc tính–giá trị/tần suất duy nhất.
  - Nhóm biểu thị giá trị của phạm vi liên tục các cho thuộc tính đã cho.

4.2. Giảm số lượng (Numerosity reduction)

4.2.2. Biểu đồ (Histograms)

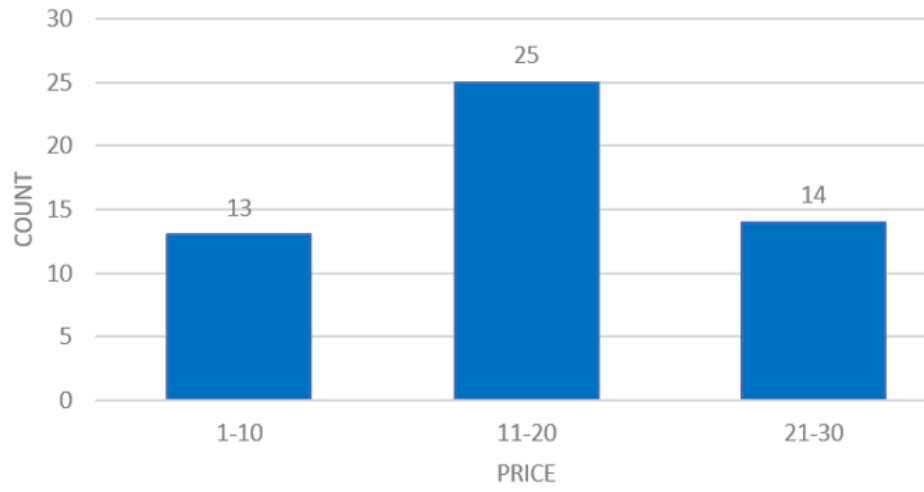
- Ví dụ: Cho dữ liệu về giá của các mặt hàng thường được bán (đã được sắp xếp tăng dần): 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30, 30.

Price	Count
1	2
5	5
8	2
10	4
12	1
14	3
15	6
18	8
20	7
21	4
25	5
28	2
30	3



Nhóm đơn

Price	Count
1-10	13
11-20	25
21-30	14

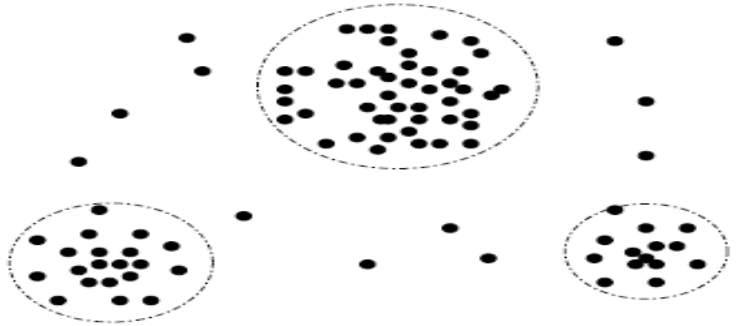


Nhóm biểu thị giá trị của phạm vi liên tục

4.2. Giảm số lượng (Numerosity reduction)

4.2.3. Phân cụm (Clustering)

- Kỹ thuật phân cụm coi các bộ dữ liệu là đối tượng. Chúng phân chia các đối tượng thành các nhóm (*groups*) hoặc cụm (*cluster*), sao cho các đối tượng trong một cụm là "tương tự" (*similar*) với nhau và "không giống" (*dissimilar*) với các đối tượng trong các cụm khác.
- Sự tương đồng thường được định nghĩa theo mức độ "gần gũi" (*close*) của các vật thể trong không gian, dựa trên hàm khoảng cách (*distance function*).
- "Chất lượng" (*quality*) của một cụm có thể được biểu thị bằng đường kính (*diameter*) của nó, tức là khoảng cách tối đa giữa hai đối tượng bất kỳ trong cụm.
- Khoảng cách trung tâm (*centroid distance*) là một thước đo thay thế về chất lượng cụm và được định nghĩa là khoảng cách trung bình của từng đối tượng cụm từ tâm cụm (biểu thị "đối tượng trung bình" (*average object*) hoặc điểm trung bình (*point object*) trong không gian của cụm).



4.2. Giảm số lượng (Numerosity reduction)

4.2.4. Lấy mẫu (Sampling)

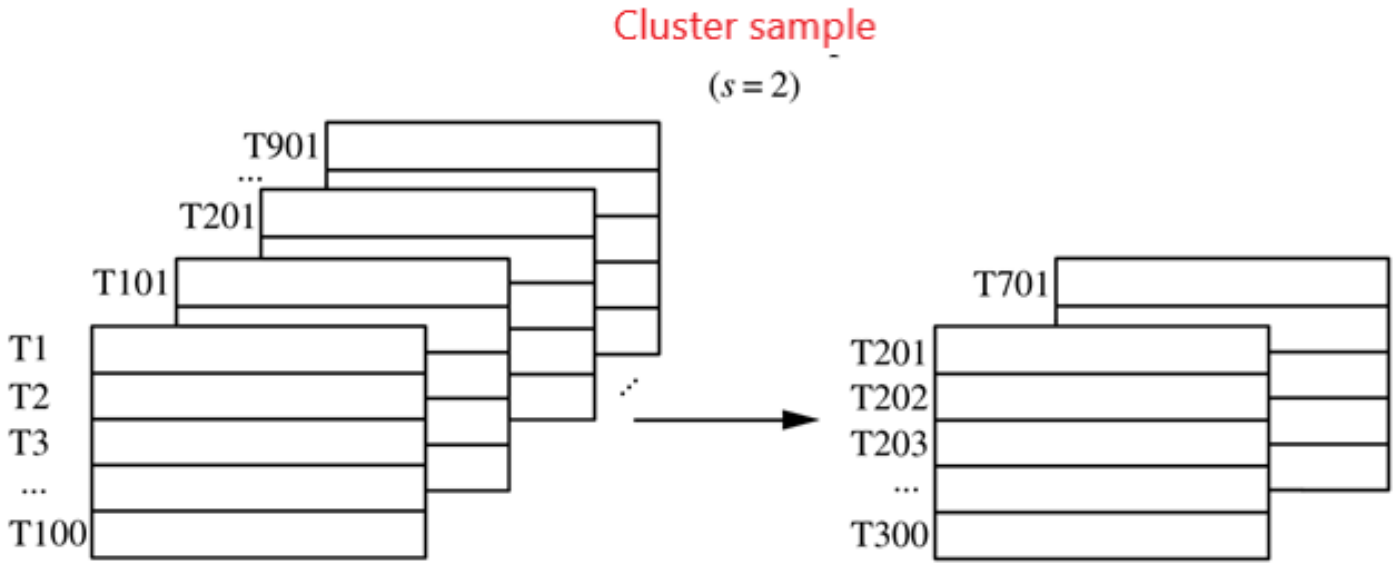
- Lấy mẫu cho phép một tập dữ liệu lớn được biểu diễn bằng một mẫu (hoặc tập hợp con) dữ liệu ngẫu nhiên nhỏ hơn nhiều.
- Phân loại:
  - i. ***Mẫu ngẫu nhiên đơn giản không thay thế*** (*Simple random sample without replacement - SRSWOR*) có kích thước *s*: Mẫu này được tạo bằng cách vẽ *s* của *N* bộ dữ liệu từ *D* (*s<N*), trong đó xác suất lấy được bất kỳ bộ dữ liệu nào trong *D* là *1/N*, nghĩa là tất cả các bộ dữ liệu đều có khả năng được lấy mẫu như nhau.
  - ii. ***Mẫu ngẫu nhiên đơn giản có thay thế*** (*Simple random sample with replacement - SRSWR*) có kích thước *s*: Tương tự như *SRSWOR*, ngoại trừ việc mỗi lần lấy một bộ từ D, nó sẽ được ghi lại và sau đó được thay thế. Nghĩa là, sau khi một bộ được rút ra, nó sẽ được đặt lại vào D để có thể được rút lại.

4.2. Giảm số lượng (Numerosity reduction)

4.2.4. Lấy mẫu (Sampling)

iii. Mẫu cụm (*Cluster sample*).

- Là chia các bộ dữ liệu trong D thành M "*cụm*".
- Tiêu chí phân cụm có thể là ngẫu nhiên, theo trọng lượng, theo đơn giá, theo vị trí địa lý, ….



4.2. *Giảm số lượng* (Numerosity reduction)

4.2.4. *Lấy mẫu* (Sampling)

iv. *Mẫu phân tầng* (Stratifiedsample)

Nếu D được chia thành các phần rời rạc lẫn nhau gọi là tầng (*strata*), thì mẫu phân tầng (*stratified sample*) của D được tạo ra bằng cách lấy *SRS* ở mỗi tầng. Điều này giúp đảm bảo mẫu đại diện, đặc biệt khi dữ liệu bị sai lệch.

Ví dụ: một mẫu phân tầng có thể được lấy từ dữ liệu khách hàng, trong đó một phân tầng được tạo cho từng nhóm tuổi khách hàng. Bằng cách này, nhóm tuổi có số lượng khách hàng ít nhất sẽ chắc chắn được đại diện.

Startified sample

(according to *age*)

T38	youth
T256	youth
T307	youth
T391	youth
T96	middle_aged
T117	middle_aged
T138	middle_aged
T263	middle_aged
T290	middle_aged
T308	middle_aged
T326	middle_aged
T387	middle_aged
T69	senior
T284	senior

T38	youth
T391	youth
T117	middle_aged
T138	middle_aged
T290	middle_aged
T326	middle_aged
T69	senior

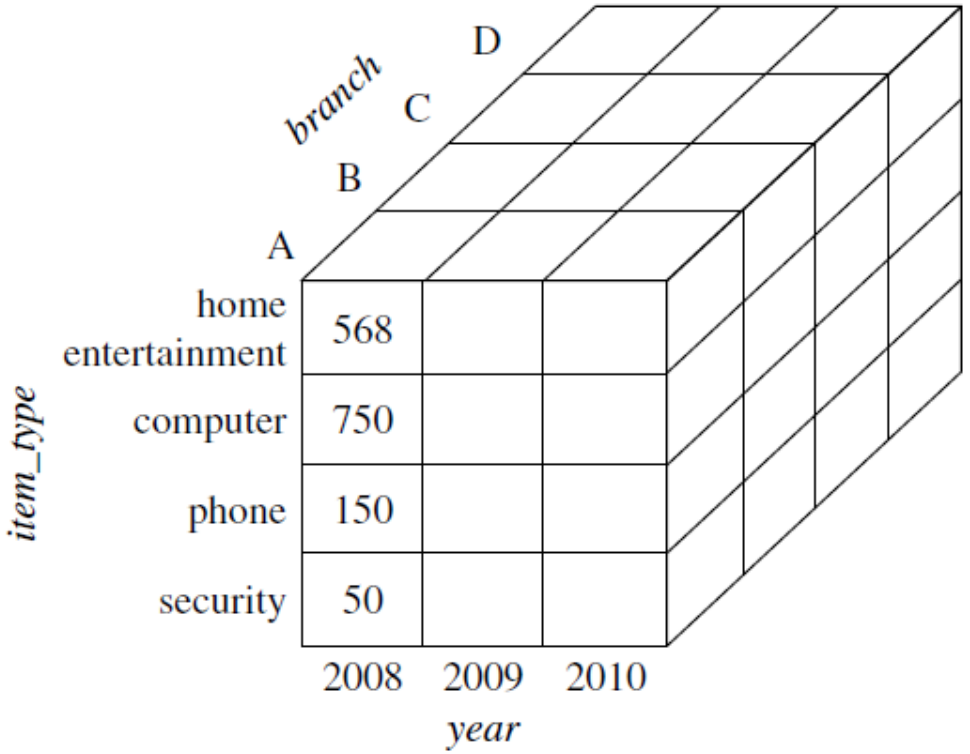
4.2. Giảm số lượng (Numerosity reduction)

4.2.5. Tổng hợp khối dữ liệu (Data Cube Aggregation)

- Khối dữ liệu lưu trữ thông tin tổng hợp đa chiều.

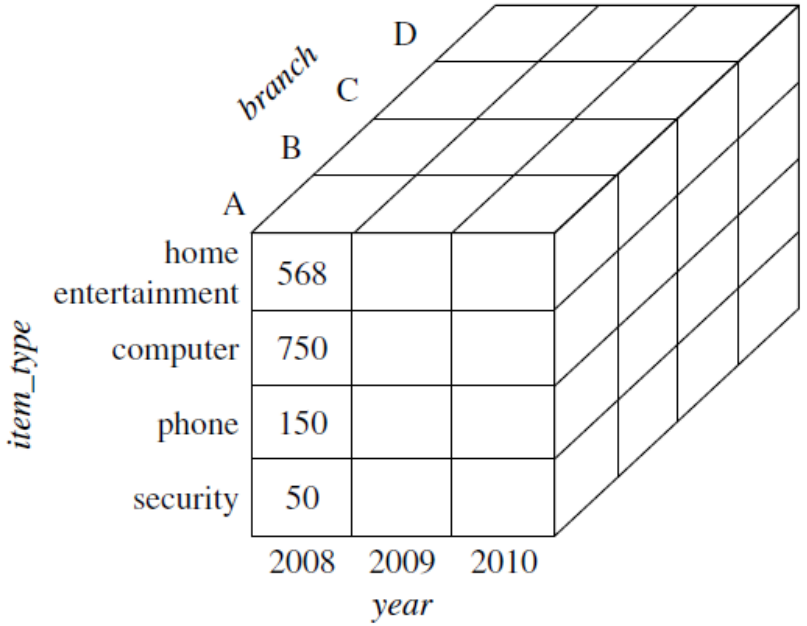
Ví dụ: hiển thị một khối dữ liệu để phân tích đa chiều dữ liệu bán hàng đối với doanh số hàng năm cho mỗi loại mặt hàng cho mỗi chi nhánh của Công ty. Mỗi ô chứa một giá trị dữ liệu tổng hợp, tương ứng với điểm dữ liệu trong không gian đa chiều.

Dựa trên khối dữ liệu này, cho phép phân tích dữ liệu ở nhiều mức độ trừu tượng như thống phân cấp cho chi nhánh có thể cho phép các chi nhánh được nhóm thành các khu vực, dựa trên địa chỉ của chúng. Các khối dữ liệu cung cấp khả năng truy cập nhanh vào dữ liệu tóm tắt, được tính toán trước, từ đó mang lại lợi ích cho việc xử lý phân tích trực tuyến cũng như khai thác dữ liệu.



4.2. Giảm số lượng (Numerosity reduction)

4.2.5. Tổng hợp khối dữ liệu (Data Cube Aggregation)



- Khối được tạo ở mức trừu tượng thấp nhất được gọi là khối cơ sở (*base cuboid*). Hình khối cơ sở phải tương ứng với một thực thể riêng lẻ được quan tâm chẳng hạn như doanh số bán hàng hoặc khách hàng. Nói cách khác, mức thấp nhất phải có thể sử dụng được hoặc hữu ích cho việc phân tích.
- Khối lập phương ở mức độ trừu tượng cao nhất là hình khối đỉnh. Đối với dữ liệu bán hàng trong hình minh họa, hình khối đỉnh sẽ cho một tổng là tổng doanh số bán hàng trong cả ba năm, cho tất cả các loại mặt hàng và cho tất cả các chi nhánh.
- Các khối dữ liệu được tạo ra cho các mức độ trừu tượng khác nhau thường được gọi là các hình khối, do đó, một khối dữ liệu thay vào đó có thể đề cập đến một mạng các hình khối. Mỗi mức độ trừu tượng cao hơn sẽ làm giảm kích thước dữ liệu kết quả hơn nữa. Khi trả lời các yêu cầu khai thác dữ liệu, nên sử dụng hình khối nhỏ nhất có sẵn phù hợp với nhiệm vụ nhất định.

4.3. Nén dữ liệu (data compression)

- Trong nén dữ liệu, các phép biến đổi được áp dụng để thu gọn hoặc "nén" từ dữ liệu gốc để có được dữ liệu nén.
- Nếu dữ liệu gốc có thể được xây dựng lại từ dữ liệu nén mà không bị mất thông tin thì việc giảm dữ liệu được gọi là không mất dữ liệu. Thay vào đó, nếu chỉ có thể xây dựng lại một cách gần đúng dữ liệu gốc thì việc giảm dữ liệu được gọi là mất dữ liệu.
- Có một số thuật toán không mất dữ liệu để nén chuỗi; tuy nhiên, chúng thường chỉ cho phép thao tác dữ liệu một cách hạn chế.
- Kỹ thuật giảm kích thước và giảm số lượng cũng có thể được coi là hình thức nén dữ liệu.

NỘI DUNG CHƯƠNG 4

1. Tổng quan về tiền xử lý dữ liệu
2. Làm sạch dữ liệu *(Data Cleaning)*
3. Tích hợp dữ liệu *(Data Integration)*
4. Giảm thiểu dữ liệu *(Data Reduction)*
5. Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*
6. Thực hành

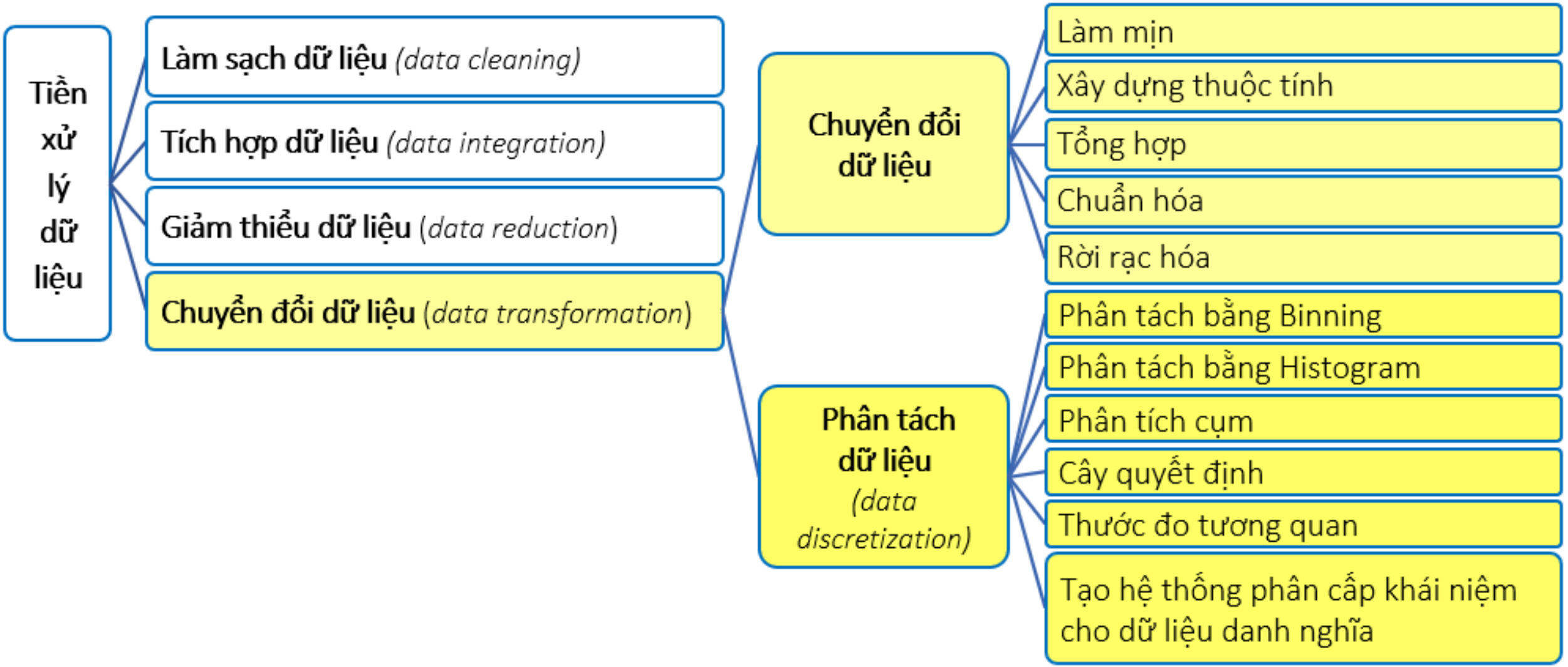
5. CHUYỂN ĐỔI DỮ LIỆU VÀ PHÂN TÁCH DỮ LIỆU

(Data Transformation and Data Discretization)

5.1. Chuyển đổi dữ liệu là gì?

- Chuyển đổi dữ liệu là quá trình thay đổi định dạng, cấu trúc hoặc giá trị của dữ liệu. Là một nhà phân tích dữ liệu, rất có thể sẽ cần chuyển đổi dữ liệu tại một số điểm để giúp việc phân tích dữ liệu dễ dàng hơn.
- Chuyển đổi dữ liệu thường bao gồm:
  - Thêm hoặc sao chép dữ liệu
  - Xóa trường hoặc bản ghi
  - Chuẩn hóa tên của các biến
  - Đổi tên, di chuyển hoặc kết hợp các cột trong cơ sở dữ liệu
  - Kết hợp một bộ dữ liệu với một bộ dữ liệu khác
  - Lưu file ở định dạng khác. Ví dụ: lưu file excel sang file csv, hoặc export dữ liệu trong SQL Server ra file excel, …

5. Chuyển đổi dữ liệu & Phân tách dữ liệu (*Data Transformation and Data Discretization*)



5.2. Mục tiêu của việc chuyển đổi dữ liệu

- *Tổ chức dữ liệu*: dữ liệu được tổ chức tốt hơn sẽ dễ sử dụng hơn.
- *Khả năng tương thích dữ liệu*: các ứng dụng hoặc hệ thống khác nhau sau đó có thể sử dụng cùng một dữ liệu.
- *Di chuyển dữ liệu*: dữ liệu có định dạng phù hợp có thể được di chuyển từ hệ thống này sang hệ thống khác.
- *Hợp nhất dữ liệu*: dữ liệu với cùng một tổ chức có thể được hợp nhất với nhau.
- *Nâng cao dữ liệu*: dữ liệu có thể được hiển thị với các trường chi tiết hơn.
- *So sánh dữ liệu*.

5.3. Các chiến lược chuyển đổi dữ liệu *(Data Transformation Strategies)*

i. ***Làm mịn*** (*Smoothing*)

- Có tác dụng loại bỏ nhiễu khỏi dữ liệu.
- Các kỹ thuật bao gồm *binning*, hồi quy (*regression*) và phân cụm (*regression*).

ii. ***Xây dựng thuộc tính*** (*Attribute construction* - hoặc xây dựng đối tượng - *feature construction*), trong đó các thuộc tính mới được xây dựng và bổ sung từ tập hợp các thuộc tính đã cho để hỗ trợ quá trình khai thác.

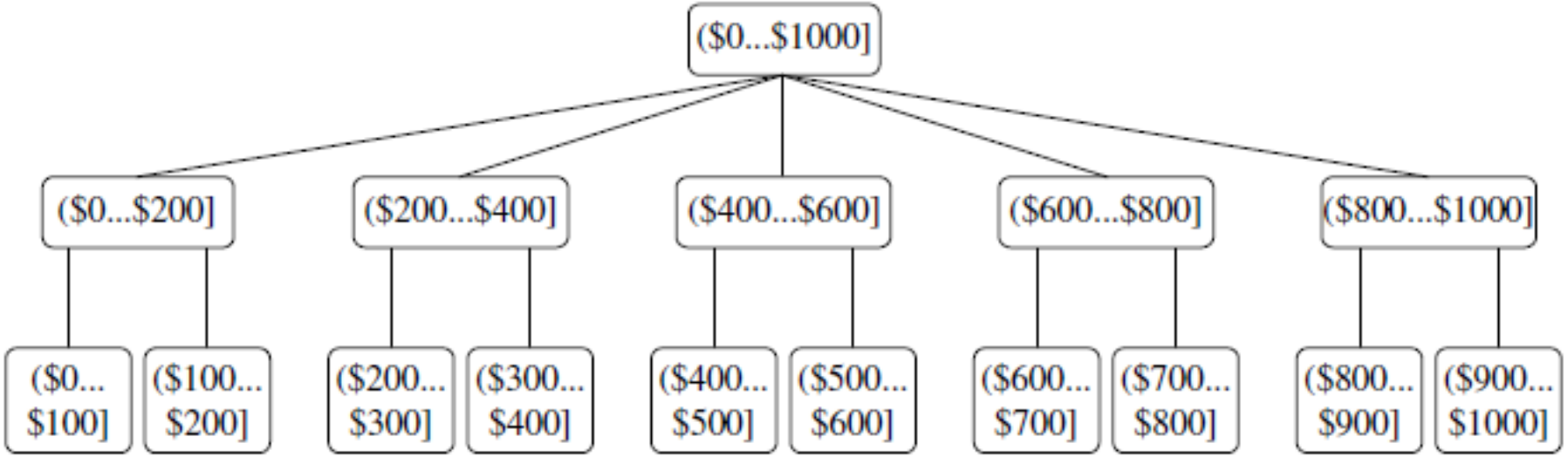
iii. ***Tổng hợp*** (*Aggregation*)

- Các hoạt động tóm tắt (*summary*) hoặc tổng hợp (*aggregation*) được áp dụng cho dữ liệu. Ví dụ: dữ liệu bán hàng hàng ngày có thể được tổng hợp để tính tổng số tiền hàng tháng và hàng năm.
- Thường được sử dụng trong việc xây dựng khối dữ liệu để phân tích dữ liệu ở nhiều mức độ trừu tượng (*abstraction levels*).

5.3. Các chiến lược chuyển đổi dữ liệu (Data Transformation Strategies)

iv. ***Chuẩn hóa*** (*Normalization*): trong đó dữ liệu thuộc tính được chia tỷ lệ để nằm trong phạm vi nhỏ hơn, chẳng hạn như −1,0 đến 1,0 hoặc 0,0 đến 1,0.

v. ***Rời rạc hóa*** (*Discretization*): trong đó các giá trị thô của thuộc tính số (ví dụ: tuổi) được thay thế bằng các nhãn khoảng (ví dụ: 0–10, 11–20, v.v...) hoặc các nhãn khái niệm (ví dụ: thanh niên, người lớn, người cao tuổi). Ngược lại, các nhãn có thể được tổ chức đệ quy thành các khái niệm cấp cao hơn, dẫn đến hệ thống phân cấp khái niệm cho thuộc tính số.



Minh họa hệ thống phân cấp khái niệm cho thuộc tính giá. Nhiều hệ thống phân cấp khái niệm có thể được xác định cho cùng một thuộc tính để đáp ứng nhu cầu của nhiều người dùng khác nhau.

5.3. Các chiến lược chuyển đổi dữ liệu (Data Transformation Strategies)

vi. *Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa (Concept hierarchy generation for nominal data)*

- Các thuộc tính cấp dưới (như đường phố) có thể được khái quát hóa thành các khái niệm cấp cao hơn (như thành phố hoặc quốc gia).
- Nhiều hệ thống phân cấp cho các thuộc tính danh nghĩa được ẩn trong lược đồ cơ sở dữ liệu và có thể được xác định tự động ở cấp độ định nghĩa lược đồ.

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

- Đơn vị đo được sử dụng có thể ảnh hưởng đến việc phân tích dữ liệu (Ví dụ: thay đổi từ mét sang inch hoặc từ kilogam sang pound). Nói chung, việc thể hiện một thuộc tính theo đơn vị nhỏ hơn sẽ dẫn đến phạm vi lớn hơn cho thuộc tính đó và do đó có xu hướng mang lại cho thuộc tính đó hiệu ứng (*effect*) hay "*trọng lượng*" (*weight*) lớn hơn.
- Để tránh sự phụ thuộc vào việc lựa chọn đơn vị đo lường, dữ liệu phải được chuẩn hóa (*normalized*) hoặc tiêu chuẩn hóa (*standardized*). Điều này liên quan đến việc chuyển đổi dữ liệu để nằm trong phạm vi nhỏ hơn hoặc phổ biến hơn, chẳng hạn như [−1, 1] hoặc [0,0; 1,0].
- Các thuật ngữ chuẩn hóa (*normalized*) và tiêu chuẩn hóa (*standardized*) được sử dụng thay thế cho nhau trong quá trình tiền xử lý dữ liệu, mặc dù trong thống kê, thuật ngữ chuẩn hóa (*normalized*) cũng có ý nghĩa khác.

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

- Đơn vị đo được sử dụng có thể ảnh hưởng đến việc phân tích dữ liệu (Ví dụ: thay đổi từ mét sang inch hoặc từ kilogam sang pound). Nói chung, việc thể hiện một thuộc tính theo đơn vị nhỏ hơn sẽ dẫn đến phạm vi lớn hơn cho thuộc tính đó và do đó có xu hướng mang lại cho thuộc tính đó hiệu ứng (*effect*) hay "*trọng lượng*" (*weight*) lớn hơn.

  ⇒Việc chuẩn hóa dữ liệu cố gắng cung cấp cho tất cả các thuộc tính một trọng số bằng nhau (*equal weight*).

- Chuẩn hóa đặc biệt hữu ích cho các thuật toán phân loại:
  - Liên quan đến mạng neural (*neural networks*)
  - Đo khoảng cách (*distance-based methods*) như phân loại và phân cụm láng giềng gần nhất (*nearest-neighbor classification and clustering*): việc chuẩn hóa giúp ngăn các thuộc tính có phạm vi lớn ban đầu (ví dụ: thu nhập) vượt trội hơn các thuộc tính có phạm vi nhỏ hơn ban đầu (ví dụ: thuộc tính nhị phân).
  - Khi sử dụng thuật toán lan truyền ngược mạng neural (*neural network backpropagation algorithm*): việc chuẩn hóa các giá trị đầu vào cho từng thuộc tính được đo trong các bộ dữ liệu huấn luyện (*training tuples*) sẽ giúp tăng tốc giai đoạn học.
  - Khi không có kiến thức trước về dữ liệu.

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

Có nhiều phương pháp để chuẩn hóa dữ liệu. Ở đây sẽ thảo luận về chuẩn hóa tối thiểu-tối đa (*min-max normalization*), chuẩn hóa điểm z (z-score normalization) và chuẩn hóa theo tỷ lệ thập phân (*normalization by decimal scaling*).

Để thảo luận, cho A là một thuộc tính số có n giá trị quan sát, $v_1, v_2, ..., v_n$.

5.4.1. Chuẩn hóa tối thiểu-tối đa (min-max normalization)

- Thực hiện chuyển đổi tuyến tính trên dữ liệu gốc. Giả sử min_A và max_A là các giá trị tối thiểu và tối đa của thuộc tính A. Chuẩn hóa tối thiểu - tối đa ánh xạ giá trị v_i của A đến v_{i0} trong phạm vi $[new\_min_A, new\_max_A]$ bằng cách tính toán.

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

- Chuẩn hóa tối thiểu tối đa bảo tồn các mối quan hệ giữa các giá trị dữ liệu ban đầu. Nó sẽ gặp lỗi "ngoài giới hạn" (*out-of-bounds*) sau này giá trị nhập vào nằm ngoài phạm vi dữ liệu ban đầu của A.

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

5.4.1. Chuẩn hóa tối thiểu-tối đa (*min-max normalization*)

- Ví dụ: Chuẩn hóa tối thiểu tối đa. Giả sử giá trị tối thiểu và tối đa cho thuộc tính thu nhập lần lượt là 12.000$ và 98.000$. Ta ánh xạ thu nhập này vào phạm vi [0.0; 1.0]. Bằng cách chuẩn hóa tối thiểu tối đa, dựa trên công thức, giá trị 73,600$ cho thu nhập được chuyển thành.

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

$$\frac{73.600 - 12.000}{98.000 - 12.000}(1.0 - 0) + 0 = 0.716$$

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

5.4.2. Chuẩn hóa điểm z (*z-score normalization* hoặc chuẩn hóa trung bình bằng zero - *zero-mean normalization*)

- Các giá trị cho thuộc tính A, được chuẩn hóa dựa trên giá trị trung bình (*mean* - tức là *average*) và độ lệch chuẩn (*standard deviation*) của ***A***. Một giá trị, v_i, của A được chuẩn hóa thành v_{i0} bằng cách tính toán.

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}$$

Trong đó:

- $\bar{A}$: giá trị trung bình của thuộc tính A. Với $\bar{A} = \frac{1}{n}(v_1 + v_2 + \cdots + v_n)$.
- σ_A : độ lệch chuẩn của thuộc tính A. σ_A được tính là căn bậc hai của phương sai của A ($Cov(A, B) = E(A \times B) - \bar{A}\bar{B}$)

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

5.4.2. Chuẩn hóa điểm z (*z-score normalization* hoặc chuẩn hóa trung bình bằng zero - *zero-mean normalization*)

- ***Ví dụ***: Giả sử rằng giá trị trung bình và độ lệch chuẩn của các giá trị cho thu nhập thuộc tính lần lượt là 54.000$ và 16.000$. Thực hiện chuẩn hóa cho điểm z có giá trị là 73.600$, giá trị này sẽ được chuyển đổi thành

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A} \quad \Rightarrow \frac{73.600 - 54.000}{16.000} = 1.225$$

- ***Một biến thể của Chuẩn hóa điểm z***
  - Thay thế độ lệch chuẩn (σ_A) của công thức $v_i' = \frac{v_i - \bar{A}}{\sigma_A}$ bằng độ lệch tuyệt đối trung bình của *A* (ký hiệu s_A). Độ lệch tuyệt đối trung bình của *A*, là
  - Độ lệch tuyệt đối trung bình của *A*, ký hiệu s_A, được tính theo công thức:

$$S_A = \frac{1}{n}(|v_1 - \bar{A}| + |v_2 - \bar{A}| + \cdots + |v_n - \bar{A}|$$

  - Do đó, chuẩn hóa điểm *z* sử dụng độ lệch tuyệt đối trung bình là $v_i' = \frac{v_i - \bar{A}}{S_A}$

5.4. Chuyển đổi dữ liệu bằng cách chuẩn hóa (Data Transformation by Normalization)

5.4.3. Chuẩn hóa theo tỷ lệ thập phân *(normalization by decimal scaling)*

- Thực hiện chuẩn hóa bằng cách di chuyển dấu thập phân của các giá trị thuộc tính *A*. Số dấu thập phân di chuyển phụ thuộc vào giá trị tuyệt đối tối đa của *A*. Một giá trị, v_i, của *A* được chuẩn hóa thành v_{i0} bằng cách tính toán theo công thức:

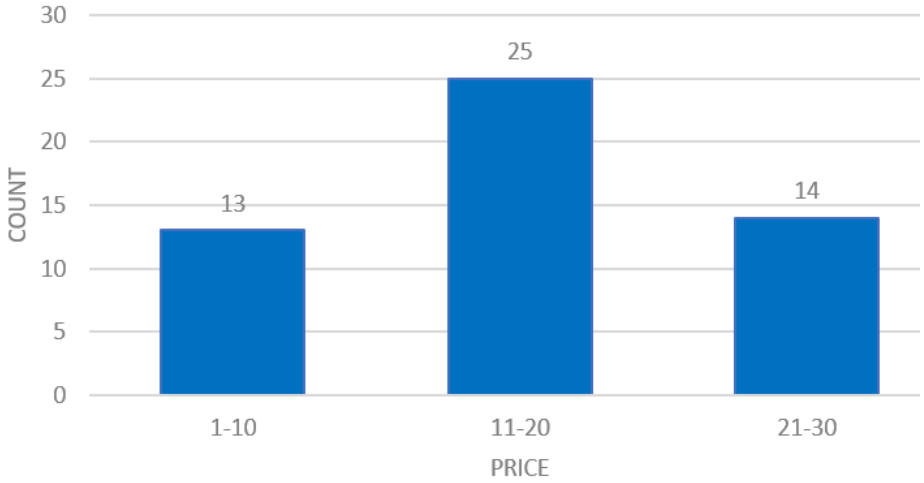
$$v_i' = \frac{v_i}{10^j}$$

trong đó *j* là số nguyên nhỏ nhất sao cho $\max(|v_i'|) < 1$

Ví dụ : Giả sử rằng các giá trị được ghi lại của *A* nằm trong khoảng từ −986 đến 917. Giá trị tuyệt đối tối đa của *A* là 986. Để chuẩn hóa theo tỷ lệ thập phân, do đó ta chia mỗi giá trị cho 1000 (tức là $j = 3$) để −986 trở thành −0,986 và 917 thành 0,917.

5.5. Phân tách bằng Binning (Discretization by Binning)

- Giống như *binning*, phân tích biểu đồ là một kỹ thuật phân tách không giám sát vì nó không sử dụng thông tin lớp. Biểu đồ phân chia các giá trị của một thuộc tính, A, thành các phạm vi rời rạc được gọi là *buckets* (xô) hoặc *bins* (thùng).
- Các quy tắc phân vùng khác nhau có thể được sử dụng để xác định biểu đồ sao cho dữ liệu được phân vùng hoặc phạm vi có kích thước bằng nhau (tức là các cột trong biểu đồ có chiều rộng bằng nhau - *equal-width histogram*) như phân vùng về giá, trong đó mỗi thùng có chiều rộng là $10).



5.6. Phân tách theo Phân tích Cụm, Cây Quyết định và Tương quan (Discretization by Cluster, Decision Tree, and Correlation Analyses)

5.6.1. Phân tích cụm (Cluster analysis)

- Thuật toán phân cụm có thể được áp dụng để phân tách một thuộc tính số, *A*, bằng cách phân vùng các giá trị của *A* thành các cụm (*clusters*) hoặc nhóm (*groups*). Việc phân cụm xem xét dựa trên sự phân bố của *A*, cũng như sự gần gũi của các điểm dữ liệu, và do đó có thể tạo ra kết quả phân tách chất lượng cao.
- Phân cụm có thể được sử dụng để tạo hệ thống phân cấp khái niệm cho A bằng cách tuân theo chiến lược chia tách từ trên xuống (*top-down splitting*) hoặc chiến lược hợp nhất từ dưới lên (*bottom-up merging*), trong đó mỗi cụm tạo thành một nút của hệ thống phân cấp khái niệm. Sau đó, các cụm được hình thành bằng cách liên tục nhóm các cụm lân cận để hình thành các khái niệm ở cấp cao hơn.

5.6. Phân tách theo Phân tích Cụm, Cây Quyết định và Tương quan

5.6.2. Kỹ thuật tạo cây quyết định để phân loại (generate decision trees for classification)

- Sử dụng cách tiếp cận chia tách từ trên xuống (*top-down splitting approach*).
- Được thực hiện có giám sát, nghĩa là chúng có sử dụng thông tin nhãn lớp. Ví dụ: có thể có một tập dữ liệu về các triệu chứng của bệnh nhân (các thuộc tính) trong đó mỗi bệnh nhân có nhãn lớp chẩn đoán liên quan.
- Thông tin phân phối lớp được sử dụng trong tính toán và xác định các điểm phân chia (giá trị dữ liệu để phân vùng phạm vi thuộc tính). Ý tưởng chính là chọn các điểm chia sao cho một phân vùng kết quả nhất định chứa càng nhiều tuple của cùng một lớp càng tốt.
- *Entropy* là thước đo được sử dụng phổ biến nhất cho mục đích này. Để phân tách một thuộc tính số, A, phương pháp chọn giá trị của A có *entropy* tối thiểu (*minimum entropy*) làm điểm tách (*split-point*) và phân vùng đệ quy (*recursively partitions*) các khoảng kết quả để đi đến sự phân tách phân cấp (*hierarchical discretization*). Sự riêng biệt như vậy tạo thành một hệ thống phân cấp khái niệm (*concept hierarchy*) cho A.

5.6. Phân tách theo Phân tích Cụm, Cây Quyết định và Tương quan

5.6.3. Thước đo tương quan *(measures of correlation)*

- Các thước đo tương quan có thể được sử dụng để phân tách. *ChiMerge* là một phương pháp phân tách dựa trên χ^2.
- Các phương pháp phân tách đã nghiên cứu cho đến thời điểm này đều sử dụng một chiến lược chia tách từ trên xuống. Điều này trái ngược với *ChimerGE*, sử dụng cách tiếp cận từ dưới lên bằng cách tìm các khoảng lân cận tốt nhất và sau đó hợp nhất chúng bằng cách đệ quy để tạo thành các khoảng lớn hơn.
- Cũng như phân tích cây quyết định, *ChiMerge* được giám sát ở chỗ nó sử dụng thông tin lớp. Về cơ bản, để phân tách chính xác, các tần suất lớp tương đối phải khá nhất quán trong một khoảng thời gian. Do đó, nếu hai khoảng liền kề có phân phối các lớp rất giống nhau, thì các khoảng có thể được hợp nhất. Nếu không, chúng vẫn tách biệt.

5.6. Phân tách theo Phân tích Cụm, Cây Quyết định và Tương quan

5.6.3. Thước đo tương quan (measures of correlation)

- *ChiMerge* tiến hành như sau:
  - Ban đầu, mỗi giá trị riêng biệt của một thuộc tính số *A* được coi là một khoảng.
  - Các kiểm tra χ^2 được thực hiện cho mỗi cặp khoảng liền kề. Các khoảng liền kề với giá trị χ^2 nhỏ nhất được hợp nhất với nhau, bởi vì giá trị χ^2 thấp cho một cặp khoảng chỉ ra các phân phối lớp tương tự.
  - Quá trình hợp nhất này tiến hành đệ quy cho đến khi đáp ứng tiêu chí dừng được xác định trước.

5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa (Concept Hierarchy Generation for Nominal Data)

- Định nghĩa thủ công về hệ thống phân cấp khái niệm có thể là một nhiệm vụ tẻ nhạt và tốn thời gian đối với người dùng hoặc chuyên gia về lĩnh vực. May mắn thay, nhiều hệ thống phân cấp ẩn chứa trong lược đồ cơ sở dữ liệu và có thể được xác định tự động ở cấp độ định nghĩa lược đồ. Hệ thống phân cấp khái niệm có thể được sử dụng để chuyển đổi dữ liệu thành nhiều cấp độ chi tiết. Ví dụ, các mô hình khai thác dữ liệu liên quan đến doanh số có thể được tìm thấy liên quan đến các khu vực hoặc quốc gia cụ thể, ngoài các địa điểm chi nhánh riêng lẻ.
- Có bốn phương pháp để tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa.

5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa *(Concept Hierarchy Generation for Nominal Data)*

Có bốn phương pháp để tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa.

i. *Đặc tả thứ tự của các thuộc tính một cách rõ ràng ở cấp độ lược đồ bởi người dùng hoặc chuyên gia* (*Specification of a partial ordering of attributes explicitly at the schema level by users or experts*)

Hệ thống phân cấp khái niệm cho các thuộc tính hoặc kích thước danh nghĩa thường liên quan đến một nhóm thuộc tính. Người dùng hoặc chuyên gia có thể dễ dàng xác định một hệ thống phân cấp khái niệm bằng cách chỉ định thứ tự một phần hoặc toàn bộ của các thuộc tính ở cấp lược đồ.

Ví dụ: Giả sử cơ sở dữ liệu quan hệ (hoặc một vị trí trong kho dữ liệu) chứa nhóm thuộc tính sau: đường phố, thành phố, tỉnh hoặc tiểu bang và quốc gia. Một hệ thống phân cấp có thể được xác định bằng cách chỉ định thứ tự giữa các thuộc tính này ở cấp lược đồ như đường phố < thành phố < tỉnh hoặc tiểu bang < quốc gia.

5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa (*Concept Hierarchy Generation for Nominal Data*)

ii. *Đặc tả một phần của hệ thống phân cấp bằng cách nhóm dữ liệu rõ ràng*

(*Specification of a portion of a hierarchy by explicit data grouping*)

- Về cơ bản đây là định nghĩa thủ công của một phần của hệ thống phân cấp khái niệm.
- Trong một cơ sở dữ liệu lớn, việc xác định toàn bộ hệ thống phân cấp khái niệm bằng cách liệt kê giá trị rõ ràng là không thực tế. Ngược lại, có thể dễ dàng chỉ định các nhóm rõ ràng cho một phần nhỏ dữ liệu cấp trung gian.

Ví dụ, sau khi chỉ định rằng tỉnh và quốc gia tạo thành một hệ thống phân cấp ở cấp lược đồ, người dùng có thể xác định một số cấp trung gian theo cách thủ công, chẳng hạn như:

- *"{Kon Tum, Gia Lai, Đắk Lắk, Đắk Nông, Lâm Đồng } ⊂ Tây nguyên"*
- hoặc *"Bắc Trung Bộ, Nam Trung Bộ, Tây nguyên" ⊂ Miền Trung*

5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa (*Concept Hierarchy Generation for Nominal Data*)

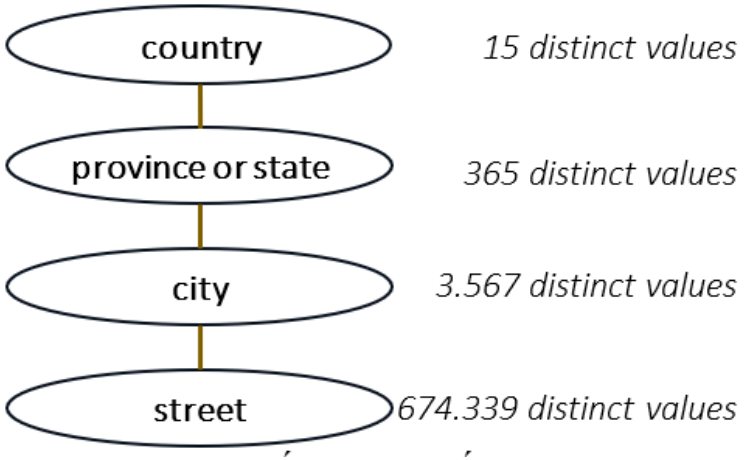
iii. *Đặc điểm kỹ thuật của một tập hợp các thuộc tính, nhưng không phải thứ tự một phần của chúng* (*Specification of a set of attributes, but not of their partial ordering*)

- Người dùng có thể chỉ định một tập hợp các thuộc tính tạo thành một hệ thống phân cấp khái niệm. Sau đó, hệ thống có thể tự động tạo thứ tự thuộc tính để xây dựng một hệ thống phân cấp khái niệm có ý nghĩa.
- Một thuộc tính xác định cấp khái niệm cao (ví dụ: quốc gia) thường sẽ chứa một số lượng nhỏ hơn các giá trị riêng biệt so với thuộc tính xác định cấp khái niệm thấp hơn (ví dụ: đường phố). Dựa trên quan sát này, một hệ thống phân cấp khái niệm có thể được tạo tự động dựa trên số lượng giá trị riêng biệt cho mỗi thuộc tính trong tập hợp thuộc tính đã cho. Thuộc tính có các giá trị khác biệt nhất được đặt ở cấp độ phân cấp thấp nhất. Số lượng giá trị riêng biệt của một thuộc tính càng thấp, nó càng cao trong hệ thống phân cấp khái niệm được tạo ra. Quy tắc heuristic này hoạt động tốt trong nhiều trường hợp.
- Một số hoán đổi hoặc điều chỉnh cấp địa phương có thể được áp dụng bởi người dùng hoặc chuyên gia, khi cần thiết, sau khi kiểm tra hệ thống phân cấp đã tạo.

5.7. *Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa* *(Concept Hierarchy Generation for Nominal Data)*

iii. *Đặc điểm kỹ thuật của một tập hợp các thuộc tính, nhưng không phải thứ tự một phần của chúng* (*Specification of a set of attributes, but not of their partial ordering*)

- Ví dụ: Giả sử trong CSDL có các thuộc tính như đường-phố, quốc gia, tỉnh hoặc tiểu bang và thành phố, nhưng không chỉ định thứ tự phân cấp giữa các thuộc tính.
  - ***B1***: sắp xếp các thuộc tính theo thứ tự tăng dần dựa trên số lượng giá trị riêng biệt trong mỗi thuộc tính. Kết quả thu được: quốc gia (có 15 giá trị), tỉnh hoặc tiểu bang (có 365 giá trị), thành phố (có 3.567 giá trị) và đường phố (có 674.339 giá trị).
  - ***B2***: tạo hệ thống phân cấp từ trên xuống theo thứ tự sắp xếp, với thuộc tính đầu tiên ở cấp trên cùng và thuộc tính cuối cùng ở cấp dưới cùng. Cuối cùng, người dùng có thể kiểm tra hệ thống phân cấp được tạo và khi cần thiết, sửa đổi nó để phản ánh các mối quan hệ ngữ nghĩa mong muốn giữa các thuộc tính.



5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa *(Concept Hierarchy Generation for Nominal Data)*

iii. Đặc điểm kỹ thuật của một tập hợp các thuộc tính, nhưng không phải thứ tự một phần của chúng (*Specification of a set of attributes, but not of their partial ordering*)

- Lưu ý rằng quy tắc heuristic này không phải là hoàn hảo. Ví dụ: chiều thời gian trong cơ sở dữ liệu có thể chứa 20 năm riêng biệt, 12 tháng riêng biệt và 7 ngày riêng biệt trong tuần. Tuy nhiên, điều này không gợi ý rằng hệ thống phân cấp thời gian phải là "*năm < tháng < các ngày trong tuần*", với các ngày trong tuần ở đầu hệ thống phân cấp.

5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa (*Concept Hierarchy Generation for Nominal Data*)

iii. *Đặc tả chỉ một phần của bộ thuộc tính* (*Specification of only a partial set of attributes*)

- Đôi khi người dùng có thể bất cẩn khi xác định một hệ thống phân cấp hoặc chỉ có một ý tưởng mơ hồ về những gì nên được đưa vào hệ thống phân cấp. Do đó, người dùng có thể chỉ bao gồm một tập hợp nhỏ của các thuộc tính có liên quan trong đặc tả phân cấp. Ví dụ: thay vì bao gồm tất cả các thuộc tính có liên quan theo thứ bậc cho vị trí, người dùng có thể chỉ định đường phố và thành phố.
- Để xử lý các hệ thống phân cấp được chỉ định một phần như vậy, điều quan trọng là phải nhúng ngữ nghĩa dữ liệu vào lược đồ cơ sở dữ liệu để các thuộc tính có kết nối ngữ nghĩa chặt chẽ có thể được ghim lại với nhau.
- Theo cách này, đặc tả của một thuộc tính có thể kích hoạt cả một nhóm các thuộc tính được liên kết chặt chẽ về mặt ngữ nghĩa bị "kéo vào" để tạo thành một hệ thống phân cấp hoàn chỉnh.
- Tuy nhiên, người dùng nên có tùy chọn ghi đè tính năng này, nếu cần thiết.

5.7. Tạo hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa (*Concept Hierarchy Generation for Nominal Data*)

iii. *Đặc tả chỉ một phần của bộ thuộc tính* (*Specification of only a partial set of attributes*)

- ***Ví dụ***:
  - Giả sử một chuyên gia khai thác dữ liệu (phục vụ như một quản trị viên) đã ghim năm thuộc tính *số*, *đường phố*, *thành phố*, *tỉnh* hoặc *tiểu bang* và *quốc gia*, bởi vì chúng được liên kết chặt chẽ về mặt ngữ nghĩa liên quan đến khái niệm vị trí. Nếu người dùng chỉ định thuộc tính *thành phố* cho vị trí xác định thứ bậc, hệ thống có thể tự động kéo tất cả năm thuộc tính có liên quan về ngữ nghĩa để tạo thành một hệ thống phân cấp.
  - Người dùng có thể chọn loại bỏ bất kỳ thuộc tính nào trong số này (ví dụ: số và đường phố) khỏi hệ thống phân cấp, giữ thành phố là cấp khái niệm thấp nhất.

NỘI DUNG CHƯƠNG 4

1. Tổng quan về tiền xử lý dữ liệu
2. Làm sạch dữ liệu *(Data Cleaning)*
3. Tích hợp dữ liệu *(Data Integration)*
4. Giảm thiểu dữ liệu *(Data Reduction)*
5. Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*
6. Thực hành

6. THỰC HÀNH

i. Với file về sức khỏe tâm thần (Metal_Health) như mô tả trong các file được gởi kèm theo bài giảng (dữ liệu đã được xóa bớt 5 giá trị trên các field)

SV lần lượt thực hiện :

- Làm sạch dữ liệu *(Data Cleaning)*
- Tích hợp dữ liệu *(Data Integration)*
- Giảm thiểu dữ liệu (Data Reduction)
- Chuyển đổi dữ liệu và phân tách dữ liệu *(Data Transformation and Data Discretization)*

6. THỰC HÀNH

ii. Sinh viên truy cập ứng dụng MS Teams để down load và thực hiện các yêu cầu có trong 2 file sau:

- "*Exercise_Data of Patients_For Medical Field.pdf*"
- "*Data of Patients_For Medical Field.csv*"

iii. Dựa vào 2 bài thực hành trên, và dựa trên dữ liệu SV đã chọn cho đề tài của mình, SV cũng thực hiện các công việc như mô tả trong 2 bài tập (1) và (11)


Q&A