ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỌC MÁY TÍNH

KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG – CS313.N22

Bài tập 2.2: *Cách sử dụng ngôn ngữ Python trong quá trình chuẩn bị dữ liệu (**Biến đổi dữ liệu** và **Rút gọn dữ liệu**)*

Nhóm 3:

Bùi Nguyễn Anh Trung (NT)	20520332
Hồ Thanh Tịnh	20520813
Nguyễn Trần Minh Anh	20520394
Lê Nguyễn Bảo Hân	20520174
Nguyễn Văn Đức Ngọc	20521666

Hồ Chí Minh, 13 tháng 03 năm 2023

Nội dung

Bảng phân công

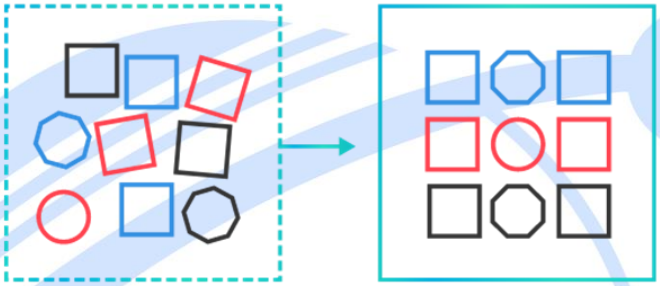
Thành viên / Công Việc	Bùi Nguyễn Anh Trung	Hồ Thanh Tịnh	Nguyễn Trần Minh Anh	Lê Nguyễn Bảo Hân	Nguyễn Văn Đức Ngọc
Phân công, quản lý công việc chung	✓				
Tim hiểu nội dung Làm sạch dữ liệu		✓			
Tim hiểu nội dung Tích hợp dữ liệu			✓		
Tìm hiểu nội dung Biến đổi dữ liệu				✓	
Tìm hiểu nội dung Rút gọn dữ liệu					✓
Tổng hợp nội dung báo cáo Định dạng báo cáo				✓	
Chuẩn bị demo Thuyết trình	✓				
Làm slide		✓	✓		✓
Đóng góp, chỉnh sửa và hoàn thiện nội dung	✓				
Mức độ hoàn thiện (%)	100%	100%	100%	100%	100%

I. Quá trình chuẩn bị dữ liệu

1. Biến đổi dữ liệu

1.1 Khái niệm biến đổi dữ liệu

Biến đổi dữ liệu là quá trình chuyển đổi, cấu trúc dữ liệu thành dạng phù hợp để thuận tiện cho các thuật toán khai thác dữ liệu. Từ đó, cải thiện kết quả khai thác và làm cho các mẫu được tìm thấy trở nên dễ hiểu hơn. [1] [2]



Hình 1. Minh họa biến đổi dữ liệu

1.2 Tại sao phải biến đổi dữ liệu?

Biến đổi dữ liệu mang lại nhiều lợi ích:

- **Kết quả khai thác tốt hơn**

  Các phương pháp khai thác sẽ mang lại kết quả tốt hơn khi thực hiện trên các dữ liệu đã được chuẩn hóa.

  Ví dụ: Trong dữ liệu khách hàng chứa thuộc tính *tuổi* và *tiền lương hàng năm*, thuộc tính *tiền lương hàng năm* thường nhận các giá trị lớn hơn nhiều so với *tuổi*. Do đó, nếu các thuộc tính không được chuẩn hóa, các phép đo khoảng cách được thực hiện trên *tiền lương hàng năm* sẽ mang ảnh hưởng lớn hơn so với các phép đo khoảng cách theo *tuổi*, gây mất cân bằng dữ liệu.

- **Đơn giản hóa và giảm kích thước dữ liệu**

  Biến đổi dữ liệu thành một dạng biểu diễn ngắn gọn, dễ sử dụng hơn ở cấp độ kiến thức của các kết quả khai thác.

  Ví dụ: Có thể giảm số lượng giá trị cho thuộc tính *tuổi* bằng cách

- Chia phạm vi tuổi thành các khoảng
- Lưu nhãn thay cho các giá trị thực

Việc thay thế số lượng lớn các giá trị thành số lượng nhỏ các nhãn đã làm giảm và đơn giản hóa dữ liệu gốc.

Age	1,5,9,4,7,11,14,17,13,18, 19,31,33,36,42,44,46,70,74,78,77

↓

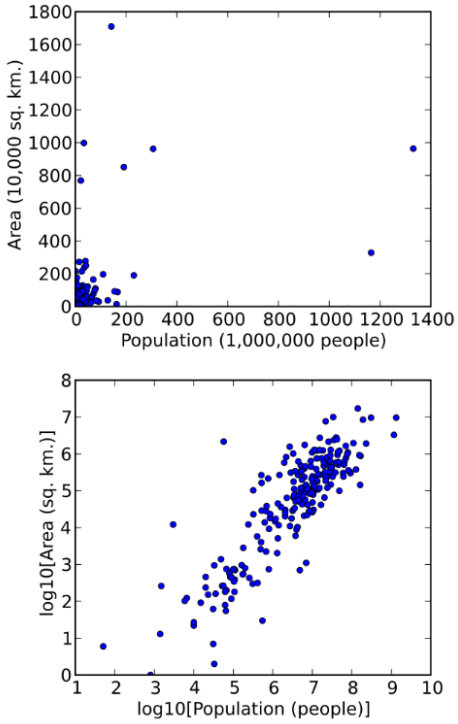
Attribute	Age	Age	Age	Age
	1,5,4,9,7	11,14,17,13,18,19	31,33,36,42,44,46	70,74,77,78
After Discretization	Child	Young	Mature	Old

Hình 2. Các giá trị liên tục của thuộc tính tuổi được chia thành các khoảng

- **Trực quan hóa dữ liệu dễ dàng hơn**

  Dữ liệu có thể được biến đổi để dễ dàng biểu diễn hơn.

  Ví dụ: Giả sử có một biểu đồ phân tán với các điểm là các quốc gia trên thế giới và các giá trị dữ liệu là *diện tích* và *dân số* của mỗi quốc gia. Nếu biểu đồ thể hiện dữ liệu chưa được biến đổi (10.000 km2 cho *diện tích* và 1.000.000 người cho *dân số*), hầu hết các quốc gia sẽ được vẽ thành cụm điểm ở góc dưới bên trái của biểu đồ. Tuy nhiên, sau khi biến đổi logarit, các điểm sẽ được trải đều hơn trên biểu đồ, dữ liệu được trực quan hóa rõ ràng hơn.



Hình 3. Biểu đồ phân tán biểu diễn diện tích lãnh thổ so với dân số của các quốc gia

- **Cải thiện khả năng diễn giải**

  Ngoài ra, biến đổi dữ liệu có thể được áp dụng để cải thiện khả năng diễn giải.

  Ví dụ: Khi so sánh mức tiết kiệm nhiên liệu của ô tô, dữ liệu này thường được trình bày dưới dạng *km/l* hoặc *dặm/gallon*. Tuy nhiên, nếu mục tiêu là đánh giá lượng nhiên liệu phải bổ sung trong một năm khi lái chiếc ô tô này so với ô tô khác, sẽ tự nhiên hơn khi làm việc với *l/km* hoặc *gallon/dặm* bằng cách áp dụng biến đổi đối ứng.

1.3 Các phương pháp biến đổi dữ liệu

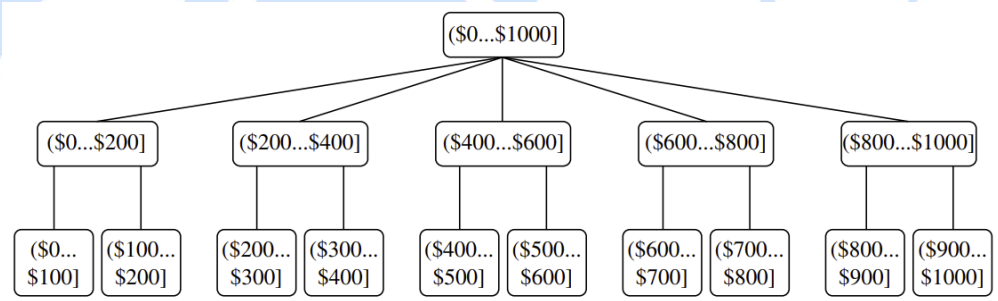
Các phương pháp biến đổi dữ liệu bao gồm:

1. **Làm trơn:** Loại bỏ dữ liệu nhiễu, các kỹ thuật bao gồm chia giỏ, hồi quy và phân cụm.
2. **Xây dựng thuộc tính:** Các thuộc tính mới được xây dựng và thêm vào từ tập hợp các thuộc tính đã cho để hỗ trợ quá trình khai thác.
3. **Tổng hợp:** Áp dụng các hoạt động tổng hợp cho dữ liệu.

Ví dụ: Dữ liệu bán hàng ngày có thể được tổng hợp để tính toán cho hàng tháng và hàng năm. Bước này thường được sử dụng trong việc xây dựng dữ liệu để phân tích dữ liệu ở nhiều cấp độ trừu tượng.

4. **Chuẩn hóa:** thuộc tính dữ liệu được chia tỷ lệ để biến đổi về phạm vi nhỏ hơn, chẳng hạn như *-1.0 đến 1.0* hoặc *0.0 đến 1.0*
5. **Rời rạc hóa:** giá trị gốc của thuộc tính số (ví dụ: *tuổi*) được thay thế bằng nhãn theo khoảng (ví dụ: *0-10, 11-20,...*) hoặc nhãn theo khái niệm (ví dụ: *thanh niên, người trưởng thành, người cao tuổi*).

   Ngược lại, các nhãn cũng có thể được tổ chức đệ quy thành các khái niệm cấp cao hơn, tạo ra một hệ thống phân cấp khái niệm cho thuộc tính số. *Hình 4* dưới đây thể hiện một hệ thống phân cấp khái niệm cho thuộc tính giá tiền. Có thể xác định nhiều hệ thống phân cấp khái niệm cho cùng một thuộc tính để đáp ứng nhu cầu của nhiều người dùng khác nhau.



Hình 4. Hệ thống phân cấp khái niệm cho thuộc tính giá tiền, trong đó một khoảng ($X...$Y] biểu thị phạm vi từ $X (không bao gồm) đến $Y (bao gồm)

6. **Phân cấp khái niệm:** phân cấp khái niệm cho các dữ liệu danh nghĩa, trong đó các thuộc tính như *đường_phố* được khái quát hóa thành các khái niệm cấp cao hơn, như *thành_phố* hoặc *quốc_gia*.

Có nhiều mối liên hệ giữa các tác vụ chính của quá trình tiền xử lý dữ liệu. Ba phương pháp đầu tiên trong số các phương pháp nêu trên đã được đề cập đến ở các mục khác. *Làm trơn* là một hình thức của làm sạch dữ liệu, *xây dựng thuộc tính* và *tổng hợp* sẽ được thảo luận ở rút gọn dữ liệu. Phân mục

này chỉ tập trung chủ yếu vào ba phương pháp: *chuẩn hóa*, *rời rạc hóa* và *phân cấp khái niệm*.

1.3.1 Chuẩn hóa

Vấn đề

Đơn vị đo lường được sử dụng có thể ảnh hưởng đến việc phân tích dữ liệu.

Ví dụ, thay đổi đơn vị từ *mét* sang *inch* cho chiều cao, hoặc từ *kilôgam* sang *pound* cho cân nặng có thể dẫn đến những kết quả rất khác nhau.

Thông thường, việc biểu diễn một thuộc tính bằng các đơn vị nhỏ hơn sẽ đem lại phạm vi lớn hơn và dẫn đến xu hướng cung cấp cho thuộc tính đó "trọng lượng" lớn hơn. Để tránh sự phụ thuộc vào việc lựa chọn đơn vị đo lường, dữ liệu nên được chuẩn hóa. Điều này hàm ý việc chuyển đổi để dữ liệu nằm trong một phạm vi nhỏ hơn hoặc phổ biến hơn, chẳng hạn như [-1,1] hay [0.0, 1.0].

Ngữ cảnh áp dụng

Chuẩn hóa dữ liệu cố gắng cung cấp cho tất cả các thuộc tính một trọng số bằng nhau. Chuẩn hóa đặc biệt hữu ích với:

- Các thuật toán phân loại liên quan đến mạng neural hoặc phép đo khoảng cách, chẳng hạn như phân loại nearest-neighbor và phân cụm lân cận.
- Nếu sử dụng thuật toán lan truyền ngược để khai thác phân loại, việc chuẩn hóa các giá trị đầu vào cho từng thuộc tính trong các bộ dữ liệu huấn luyện sẽ giúp tăng tốc giai đoạn đào tạo.
- Đối với các phương pháp dựa trên khoảng cách, 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 ban đầu nhỏ (ví dụ: *thuộc tính nhị phân*).
- Phương pháp này vẫn hữu ích ngay cả khi không có kiến thức trước về dữ liệu.

Có nhiều phương pháp để chuẩn hóa dữ liệu. Ở đây, ta nghiên cứu *min-max normalization*, *z-score normaliztion*, và *decimal scaling*. Giả sử, A là một thuộc tính số với n giá trị được quan sát, $v_1, v_2, \dots, v_n$.

Phương pháp

1. **Min-max normalization** thực hiện phép biến đổi tuyến tính trên dữ liệu gốc. Giả sử min_A và max_A là giá trị nhỏ nhất và lớn nhất của thuộc tính A. Một chuẩn hóa min-max ánh xạ một giá trị v_i của A đến v_i' trong miền $[new\_min_A, new\_max_A]$ bằng cách tính

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

   Min-max normalization bảo toàn mối quan hệ giữa các giá trị dữ liệu gốc. Nếu giá trị đầu vào trong tương lai nằm ngoài phạm vi dữ liệu gốc cho A, ta sẽ gặp phải lỗi "out-of-bounds".

   Ví dụ min-max normalization. Giả sử giá trị nhỏ nhất và lớn nhất cho thuộc tính *thu nhập* lần lượt là \$12.000 và \$98.000, ta muốn biến đổi *thu nhập* về miền giá trị [0.0, 1.0]. Với min-max normalization, một giá trị \$73.600 cho *thu nhập* sẽ được biến đổi thành

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

2. **z-score normalization** (hay *zero-mean normalization*), các giá trị của thuộc tính A được chuẩn hóa dựa trên giá trị trung bình và độ lệch chuẩn của A. Giá trị v_i được chuẩn hóa thành v_i' bằng cách tính

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

   Trong đó $\bar{A}$ và σ_A lần lượt là giá trị trung bình và độ lệch chuẩn của thuộc tính A. Với $\bar{A} = \frac{1}{n}(v_1 + v_2 + \cdots + v_n)$ và σ_A là căn bậc hai phương sai của A.

Phương pháp chuẩn hóa này rất hữu ích khi giá trị tối thiểu và tối đa thực tế của thuộc tính A không xác định hoặc khi các giá trị ngoại lai chiếm ưu thế trong min-max normalization.

Ví dụ z-score normalization. Giả sử giá trị trung bình và độ lệch chuẩn của các giá trị thuộc tính *thu nhập* lần lượt là $54.000 và $16.000, với giá trị là $73.600 cho *thu nhập* được biến đổi thành

$$\frac{73.600 - 54.000}{16.000} = 1.225$$

Một biến thể của phương pháp này thay thế *độ lệch chuẩn* bằng *độ lệch tuyệt đối trung bình* của A, kí hiệu s_A

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

z-score normalization sử dụng độ lệch tuyệt đối trung bình:

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

Độ lệch tuyệt đối trung bình s_A mạnh mẽ hơn đối với các ngoại lai so với độ lệch chuẩn σ_A. Khi tính toán độ lệch tuyệt đối trung bình, độ lệch trung bình (tức $|x_i - \bar{x}|$) không được bình phương, do đó, ảnh hưởng của các ngoại lai phần nào được giảm đi.

3. **Decimal scaling** chuẩn hóa bằng cách di chuyển dấu thập phân của các giá trị thuộc tính A. Số lượng dấu thập phân được di chuyển phụ thuộc vào giá trị tuyệt đối tối đa A. v_i được chuẩn hóa thành v_i' bằng cách tính

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

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

Ví dụ decimal scaling. Giả sử các giá trị của A được ghi nhận nằm trong khoảng -986 đến 917. Giá trị tuyệt đối tối đa của A là 986. Để chuẩn hóa theo decimal scaling, ta chia từng giá trị cho 1000 (tức j=3) để -986 chuẩn hóa thành -0.986 và 917 thành 0.917.

Lưu ý rằng quá trình chuẩn hóa có thể làm thay đổi khá nhiều dữ liệu gốc, đặc biệt là khi sử dụng z-score normalization hay decimal scaling, cần lưu lại các tham số chuẩn hóa (như giá trị trung bình và độ lệch chuẩn nếu sử dụng z-score normalization) để dữ liệu trong tương lai có thể được chuẩn hóa một cách thống nhất.

1.3.2 Rời rạc hóa

Rời rạc hóa thực hiện chuyển đổi thuộc tính số (liên tục) thành thuộc tính phân loại để đáp ứng nhu cầu của các thuật toán khai thác.

Trong rời rạc hóa:

- Phạm vi của một thuộc tính liên tục được chia thành các khoảng
- Sau đó, các giá trị dữ liệu thực có thể được thay thế bằng nhãn của khoảng để thu được thuộc tính dạng phân loại.

Có thể thực hiện rời rạc hóa dữ liệu bằng các phương pháp:

1. **Chia giỏ (Binning)**

   Chia giỏ là một kỹ thuật phân chia top-down dựa trên số lượng giỏ được chỉ định. Ta đã thảo luận về các phương pháp chia giỏ ở bước *làm sạch dữ liệu*. Các phương pháp này cũng được sử dụng cho *rời rạc hóa* để rút gọn dữ liệu và tạo hệ thống phân cấp khái niệm.

   Chẳng hạn, các giá trị thuộc tính có thể được phân chia bằng cách áp dụng chia nhóm có độ rộng hoặc tần số bằng nhau. Sau đó, thay thế từng giá trị giỏ bằng giá trị trung bình hoặc trung tuyến giỏ. Những kỹ thuật này có thể được áp dụng đệ quy đến các phân vùng kết quả để tạo phân cấp khái niệm.

   Chia giỏ không sử dụng thông tin phân lớp và vì vậy là một kỹ thuật ***rời rạc hóa không giám sát.***

   Ví dụ rời rạc hóa bằng phương pháp chia giỏ theo độ rộng. Giả sử thuộc tính *giá tiền* có 12 giá trị của đã được sắp xếp:

price	5	10	11	13	15	35	50	55	72	89	204	215

- Thực hiện phân chia theo độ rộng (equal-width), ta có độ rộng mỗi khoảng bằng

$$\frac{215 - 5}{3} = 70$$

- Phân chia vào *ba giỏ*

bin1	5, 10, 11, 13, 15, 35, 50, 55, 72
bin2	89
bin3	204, 215

- Thay thế từng giá trị bằng nhãn giỏ tương ứng

price	5	10	11	13	15	35	50	55	72	89	204	215
categorical attr.	1	1	1	1	1	1	1	1	1	2	3	3

Ví dụ rời rạc hóa bằng phương pháp chia giỏ theo độ sâu. Giả sử với thuộc tính *giá tiền* như trên.

- Phân chia vào *ba giỏ*: độ sâu bằng 4 (mỗi khoảng chứa 4 giá trị)

bin1	5, 10, 11, 13
bin2	15, 35, 50, 55
bin3	72, 89, 204, 215

- Thay thế từng giá trị bằng nhãn giỏ tương ứng

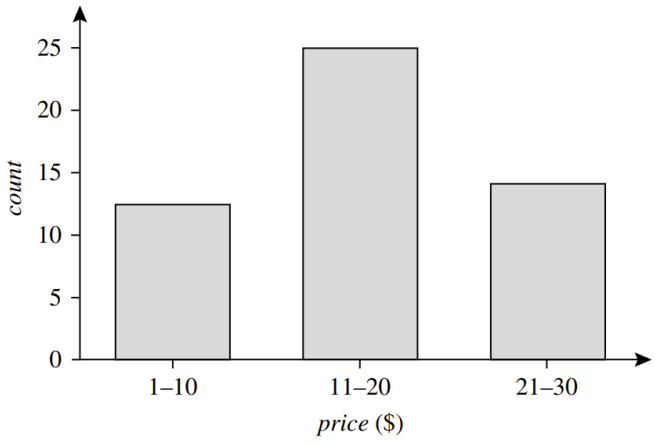
price	5	10	11	13	15	35	50	55	72	89	204	215
categorical attr.	1	1	1	1	2	2	2	2	3	3	3	3

2. Phân tích biểu đồ (Histogram Analysis)

Tương tự chia giỏ, phân tích biểu đồ là một kỹ thuật rời rạc hóa không giám sát bởi vì không sử dụng thông tin phân lớp. Một biểu đồ phân chia

các giá trị của một thuộc tính A thành các phạm vi rời rạc được gọi là nhóm hoặc giỏ.

Có nhiều quy tắc phân chia khác nhau có thể được sử dụng để xác định biểu đồ. Ví dụ, trong một biểu đồ có chiều rộng bằng nhau, các giá trị được phân chia thành các phân vùng hoặc phạm vi có kích thước bằng nhau (ví dụ: trong *Hình 5* dưới đây về *giá tiền*, mỗi nhóm có chiều rộng là $10).



Hình 5. Biểu đồ giá tiền có chiều rộng bằng nhau, các giá trị được tổng hợp sao cho mỗi nhóm có chiều rộng đồng nhất là $10

Với biểu đồ tần số bằng nhau, các giá trị được phân vùng sao cho lý tưởng nhất là mỗi phân vùng chứa cùng một số bộ dữ liệu. Thuật toán phân tích biểu đồ có thể được áp dụng đệ quy cho từng phân vùng để tự động tạo hệ thống phân cấp khái niệm đa cấp, quy trình sẽ kết thúc sau khi đạt đến số lượng cấp khái niệm được chỉ định trước.

Kích thước khoảng tối thiểu cũng có thể được chỉ định cho mỗi cấp độ để kiểm soát quy trình đệ quy. Điều này tạo nên chiều rộng tối thiểu của phân vùng hoặc số lượng giá trị tối thiểu cho mỗi phân vùng ở từng cấp độ. Biểu đồ cũng có thể được phân vùng dựa trên phân tích cụm phân phối dữ liệu, như được mô tả ở phần tiếp theo.

3. Phân cụm, cây quyết định và phân tích tương quan (Cluster, Decision Tree, Correlation Analyses)

Phân cụm, phân tích cây quyết định và phân tích tương quan có thể được sử dụng để thực hiện rời rạc hóa dữ liệu.

Cluster

Phân tích cụm là một phương pháp rời rạc hóa dữ liệu phổ biến. Một thuật toán phân cụm có thể được áp dụng để rời rạc hóa một thuộc tính số *A* bằng cách phân chia các giá trị của *A* thành các cụm hoặc nhóm. Phân cụm xem xét phân phối *A*, cũng như mức độ gần của các điểm dữ liệu và do đó có thể tạo ra kết quả rời rạc chất lượng cao.

Phân cụm có thể được dùng để tạo phân cấp khái niệm cho *A* bằng cách tuân theo chiến lược phân chia top-down hoặc chiến lược hợp nhất bottom-up, trong đó mỗi cụm tạo thành một nút của phân cấp khái niệm.

Ban đầu, mỗi cụm hoặc phân vùng ban đầu có thể được phân tách thành nhiều cụm con, tạo thành cấp thấp hơn của hệ thống phân cấp. Sau đó, các cụm được hình thành bằng cách nhóm liên tục các cụm lân cận để tạo thành các khái niệm cấp cao hơn.

Ví dụ phân cụm đơn giản. Phân chia dữ liệu bằng khoảng cách giá trị.

- Phân chia dữ liệu từ 2 khoảng cách giá trị lớn nhất thành ba giỏ

bin1	5, 10, 11, 13, 15
bin2	35, 50, 55, 72, 89
bin3	204, 215

- Thay thế từng giá trị bằng nhãn giỏ tương ứng

price	5	10	11	13	15	35	50	55	72	89	204	215
categorical attr.	1	1	1	1	1	2	2	2	2	2	3	3

Decision Tree

Các kỹ thuật tạo cây quyết định cho phân lớp có thể được áp dụng cho việc rời rạc hóa. Những kỹ thuật như vậy sử dụng cách tiếp cận phân chia top-down. Không giống như các phương pháp khác đã được đề cập cho

đến nay, các phương pháp tiếp cận Decision Tree cho rời rạc hóa là phương pháp có giám sát, nghĩa là chúng sử dụng thông tin nhãn lớp.

Ví dụ, có một tập hợp dữ liệu về các triệu chứng của bệnh nhân (các thuộc tính), trong đó mỗi bệnh nhân được gán một nhãn lớp chẩn đoán. Thông tin phân phối của các lớp được sử dụng trong tính toán và xác định điểm phân chia split-point (giá trị dữ liệu để phân vùng phạm vi thuộc tính).

Ý tưởng chính là chọn các điểm phân chia sao cho một phân vùng kết quả đã cho chứa càng nhiều bộ dữ liệu của cùng một lớp càng tốt. **Entropy** là biện pháp được sử dụng phổ biến nhất cho mục đích này. Để rời rạc hóa một thuộc tính số A, phương pháp sẽ chọn giá trị của A có entropy nhỏ nhất làm điểm phân chia và phân vùng đệ quy các khoảng kết quả để đạt sự rời rạc hóa theo cấp bậc. Sự rời rạc hóa như vậy tạo thành một hệ thống phân cấp khái niệm cho *A*.

Bởi vì sự rời rạc hóa dựa trên cây quyết định sử dụng thông tin phân lớp, nhiều khả năng các ranh giới khoảng (điểm phân chia split-points) được chọn xảy ra ở những nơi có thể giúp cải thiện độ chính xác của phân loại.

Correlation Analyses

Các biện pháp tương quan có thể được sử dụng để rời rạc hóa. ChiMerge là một phương pháp rời rạc hóa dựa trên $\chi 2$.

Các phương pháp rời rạc hóa mà ta đã nghiên cứu đều sử dụng chiến lược chia chia top-down. ChiMerge, ngược lại, sử dụng cách tiếp cận bottom-up bằng cách tìm các khoảng lân cận tốt nhất rồi hợp nhất chúng để tạo thành các khoảng lớn hơn theo cách đệ quy.

Cũng như phân tích cây quyết định, ChiMerge được giám sát ở chỗ nó sử dụng thông tin phân lớp. Lưu ý cơ bản để rời rạc hóa được chính xác là tần số lớp phải tương đối nhất quán trong một khoảng. Do đó, nếu hai khoảng liền kề có sự phân bố các lớp rất giống nhau, thì các khoảng đó có thể được hợp nhất. Nếu không, nó vẫn nên tách biệt.

ChiMerge được thực hiện như sau. Ban đầu, mỗi giá trị riêng biệt của thuộc tính số A được coi là một khoảng. Phép thử χ2 được thực hiện cho mọi cặp khoảng liền kề. Các khoảng liền kề có giá trị χ2 nhỏ nhất được hợp nhất với nhau, bởi vì giá trị χ2 thấp cho một cặp biểu thị sự phân bố lớp tương tự. Quá trình hợp nhất này tiến hành đệ quy cho đến khi đáp ứng tiêu chí dừng được xác định trước.

1.3.3 Phân cấp khái niệm

Biến đổi dữ liệu bằng phương pháp xây dựng hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa. Thuộc tính danh nghĩa có số lượng giá trị riêng biệt hữu hạn (nhưng có thể tương đối lớn) và không có thứ tự giữa các giá trị. Ví dụ bao gồm *vị_trí_địa_lý*, *loại_công_việc* và *loại_mục*.

Định nghĩa thủ công các hệ thống phân cấp khái niệm có thể là một nhiệm vụ khá nhàm chán và tốn thời gian. May mắn thay, nhiều hệ thống phân cấp ẩn bên trong lược đồ cơ sở dữ liệu và có thể được xác định tự động ở mức 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 mẫu khai thác dữ liệu liên quan đến *doanh số bán hàng* có thể liên quan đến các khu vực hoặc quốc gia cụ thể, ngoài các địa điểm chi nhánh riêng lẻ.

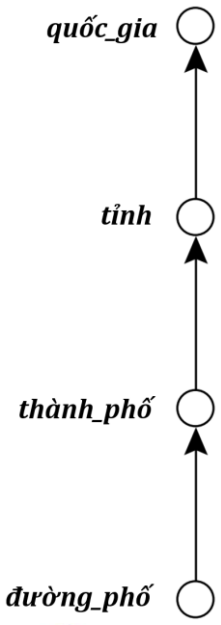
Ở đây, ta nghiên cứu bốn phương pháp để tạo ra các hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa.

1. **Đặc tả một phần thứ tự của các thuộc tính**

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

   Giả sử cơ sở dữ liệu quan hệ chứa nhóm thuộc tính sau: ***đường_phố***, ***thành_phố***, ***tỉnh*** và ***quốc_gia***. Một hệ thống phân cấp có thể được xác

định bằng cách chỉ định tổng thứ tự giữa các thuộc tính này ở cấp lược đồ, chẳng hạn như ***đường_phố < thành_phố < tỉnh < quốc_gia***.



Hình 6. Hệ thống phân cấp khái niệm vị trí

2. **Đặc tả một phần cấu trúc phân cấp bằng cách nhóm dữ liệu**

   Về cơ bản, phương pháp này thực hiện định nghĩa thủ công 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 các giá trị một cách rõ ràng là không thực tế. Ngược lại, chúng ta có thể dễ dàng chỉ định các nhóm cho một phần nhỏ dữ liệu ở cấp trung gian.

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

   "{*Alberta*, *Saskatchewan*, *Manitoba*} ⊂ *thảo_nguyên_Canada*" và **"{*British_Columbia*, *thảo_nguyên_Canada*} ⊂ *Western_Canada*"**.

3. **Đặc tả một tập hợp các thuộc tính nhưng không phải thứ tự từng phần của chúng**

   Người dùng có thể chỉ định một tập hợp các thuộc tính tạo thành một hệ thống phân cấp khái niệm nhưng bỏ qua việc nêu rõ thứ tự từng phần

của chúng. Sau đó, hệ thống có thể cố gắng tự động tạo thứ tự thuộc tính để xây dựng một hệ thống phân cấp khái niệm có ý nghĩa.

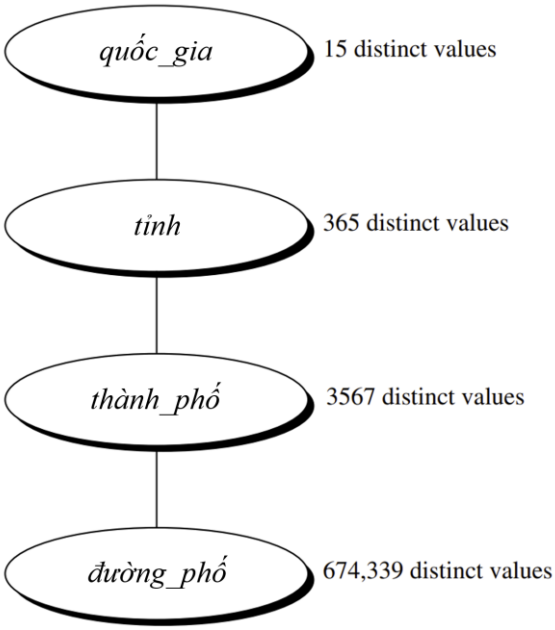
"*Nếu không có kiến thức về ngữ nghĩa của dữ liệu, làm thế nào để tìm ra thứ tự phân cấp cho một tập hợp các thuộc tính danh nghĩa bất kì?*"

Có thể thấy, các khái niệm cấp cao hơn thường sẽ bao gồm nhiều khái niệm cấp thấp hơn, một thuộc tính định nghĩa cấp độ khái niệm cao (ví dụ: ***quốc_gia***) thường sẽ chứa một số lượng giá trị riêng biệt nhỏ hơn so với một thuộc tính định nghĩa cấp độ khái niệm thấp hơn (ví dụ: ***đường_phố***). Dựa trên quan sát này, 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 thuộc tính đã cho. Thuộc tính có các giá trị khác biệt nhất sẽ được đặt ở mức phân cấp thấp nhất. Số lượng giá trị riêng biệt mà một thuộc tính có càng thấp thì thuộc tính đó ở vị trí càng cao trong hệ thống phân cấp khái niệm được tạo.

Quy tắc heuristic này hoạt động tốt trong nhiều trường hợp. Sau khi kiểm tra hệ thống phân cấp được tạo, nếu cần thiết, người dùng hoặc chuyên gia có thể áp dụng một số hoán đổi hoặc điều chỉnh ở cấp độ cục bộ.

Ví dụ Tạo hệ thống phân cấp khái niệm dựa trên số lượng giá trị riêng biệt của mỗi thuộc tính. Giả sử người dùng chọn một tập hợp các thuộc tính hướng vị trí (***đường_phố***, ***quốc_gia***, ***tỉnh*** và ***thành_phố***) từ *AllElectronics* database, nhưng không chỉ định thứ tự phân cấp giữa các thuộc tính.

Hệ thống phân cấp khái niệm cho vị trí có thể được tạo tự động, như được minh họa trong *Hình 6*.



Hình 7. Hệ thống phân cấp khái niệm được tạo tự động dựa trên số lượng giá trị riêng biệt của thuộc tính

- Trước tiên, sắp xếp các thuộc tính theo thứ tự tăng dần dựa trên số lượng giá trị riêng biệt của mỗi thuộc tính. Điều này dẫn đến kết quả sau (trong đó số lượng giá trị riêng biệt cho mỗi thuộc tính được hiển thị trong ngoặc đơn): ***quốc_gia*** (15), ***tỉnh*** (365), ***thành_phố*** (3.567) và ***đường_phố*** (674.339).
- Sau đó, tạo hệ thống phân cấp từ trên xuống theo thứ tự được sắp xếp, với thuộc tính đầu tiên ở cấp cao nhất và thuộc tính cuối cùng ở cấp dưới cùng.
- Cuối cùng, người dùng có thể kiểm tra hệ thống phân cấp được tạo sửa đổi nếu cần để phản ánh các mối quan hệ ngữ nghĩa mong muốn giữa các thuộc tính. Trong ví dụ này, có thể thấy không cần sửa đổi cấu trúc phân cấp đã tạo.

Lưu ý rằng quy tắc heuristic này vẫn tồn tại những mặt hạn chế. Chẳng hạn, một 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 có nghĩa là hệ thống phân cấp thời gian nên là ***"năm < tháng < ngày_trong_tuần"*** với ***ngày_trong_tuần*** ở đầu hệ thống phân cấp.

4. Đặc tả một phần tập hợp các thuộc tính

Đôi khi người dùng có thể bất cẩn khi xác định cấu trúc phân cấp hoặc mới chỉ có một ý tưởng mơ hồ. Do đó, người dùng có thể chỉ xác định một tập hợp con nhỏ các thuộc tính có liên quan trong đặc tả cấu trúc phân cấp.

Ví dụ, thay vì bao gồm tất cả các thuộc tính có liên quan về mặt phân cấp cho ***vị trí***, người dùng có thể chỉ xác định ***đường_phố*** và ***thành_phố***. Để xử lý trường hợp các hệ thống phân cấp chỉ 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.

Bằng cách này, đặc tả của một thuộc tính có thể kích hoạt toàn bộ nhóm các thuộc tính được liên kết chặt chẽ về mặt ngữ nghĩa để "kéo vào" và tạo thành một hệ thống phân cấp hoàn chỉnh.

Ví dụ Tạo hệ thống phân cấp khái niệm bằng cách sử dụng các kết nối ngữ nghĩa được chỉ định trước. Giả sử một chuyên gia khai thác dữ liệu (đóng vai trò là quản trị viên) đã ghim năm thuộc tính ***số***, ***đường_phố***, ***thành_phố***, ***tỉnh*** và ***quốc_gia***, bởi vì chúng có liên kết chặt chẽ về mặt ngữ nghĩa về khái niệm ***vị trí***. Nếu người dùng chỉ xác định thuộc tính ***thành_phố*** cho một hệ thống phân cấp ***vị trí***, hệ thống có thể tự động kéo vào tất cả năm thuộc tính có liên quan về mặt ngữ nghĩa để tạo thành một thứ bậc. 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*** ở mức khái niệm thấp nhất.

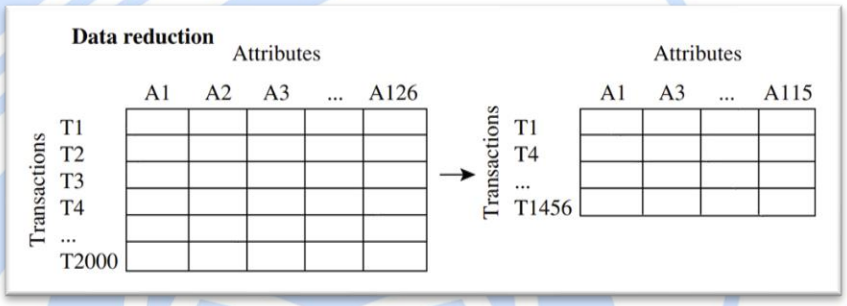
Tóm lại, thông tin ở cấp lược đồ và ở số lượng giá trị của thuộc tính có thể được sử dụng để tạo ra các hệ thống phân cấp khái niệm cho dữ liệu danh nghĩa. Biến đổi dữ liệu danh nghĩa bằng việc sử dụng các hệ thống phân cấp khái niệm cho phép tìm thấy các mẫu kiến thức cấp cao hơn.

Nó cho phép khai thác ở nhiều cấp độ trừu tượng và đây là yêu cầu phổ biến đối với các ứng dụng khai thác dữ liệu.

2. Rút gọn dữ liệu

2.1 Khái niệm rút gọn dữ liệu

Rút gọn dữ liệu làm giảm kích thước của một tập dữ liệu trong khi vẫn giữ được những thông tin quan trọng nhất. Điều này có lợi trong các tình huống mà tập dữ liệu quá lớn để được xử lý hiệu quả, hoặc khi tập dữ liệu chứa một lượng lớn thông tin không liên quan hoặc dư thừa.



Hình 8. Minh họa rút gọn dữ liệu

2.2 Tại sao phải rút gọn dữ liệu?

Ưu điểm chính của rút gọn dữ liệu:

- *Cải thiện hiệu quả* của các thuật toán học máy: giảm kích thước của tập dữ liệu; loại bỏ thông tin không liên quan hoặc dư thừa giúp các thuật toán học máy chạy nhanh hơn, mô hình hoạt động chính xác và mạnh mẽ hơn.
- *Giảm chi phí lưu trữ* liên quan đến các bộ dữ liệu lớn.

Tuy nhiên, phương pháp này vẫn tồn tại những mặt **hạn chế** như:

- *Gây mất mát thông tin* vì dữ liệu bị xóa trong quá trình rút gọn.
- *Ảnh hưởng đến độ chính xác* vì có thể loại bỏ thông tin quan trọng, cần thiết cho các dự đoán.

- *Tác động đến khả năng diễn giải dữ liệu* vì việc xóa thông tin không liên quan hoặc dư thừa có thể vô tình loại bỏ ngữ cảnh cần thiết để hiểu kết quả.
- *Phát sinh thêm chi phí tính toán* cho quá trình khai thác dữ liệu vì đòi hòi thêm thời gian xử lý để rút gọn dữ liệu.

2.3 Các phương pháp rút gọn dữ liệu

Các phương pháp rút gọn dữ liệu phổ biến:

1. **Giảm chiều dữ liệu (Dimensionality Reduction):** Loại bỏ các đặc trưng không liên quan bằng cách kết hợp nhiều đặc trưng thành một đặc trưng duy nhất.
2. **Giảm số lượng dữ liệu (Numerosity Reduction):** Sử dụng các hình thức biểu diễn dữ liệu nhỏ, thay thế để làm giảm khối lượng dữ liệu. Có hai loại phương pháp:
    - *Phương pháp tham số (Parametric)*
    - *Phương pháp phi tham số (Non-Parametric).*
3. **Nén dữ liệu (Data compression):** Chuyển đổi, mã hóa cấu trúc dữ liệu theo cách tiêu tốn ít không gian hơn; xây dựng một biểu diễn thông tin nhỏ gọn bằng cách loại bỏ sự dư thừa và biểu diễn dữ liệu ở dạng nhị phân.
    - *Nén không mất dữ liệu:* dữ liệu có thể được khôi phục thành công từ dạng nén
    - *Nén mất dữ liệu:* không thể khôi phục dạng ban đầu từ dạng nén
4. **Tổng hợp khối dữ liệu (Data Cube Aggregation):** Là một tập hợp đa chiều, tổng hợp ở các cấp độ khác nhau của khối dữ liệu để đại diện cho tập dữ liệu gốc. Trong đó, khối dữ liệu là một cách lưu trữ dữ liệu hiệu quả hơn, do đó đạt được mức giảm dữ liệu, bên cạnh các hoạt động tổng hợp nhanh hơn.

2.3.1 Giảm chiều dữ liệu

Lựa chọn tập thuộc tính con

- **Lựa chọn chuyển tiếp từng bước:** Bắt đầu bằng một tập hợp các thuộc tính rỗng, tìm thuộc tính tốt nhất trong số các thuộc tính ban đầu dựa trên mức độ liên quan của chúng với các thuộc tính khác.

  Giả sử có các thuộc tính sau trong tập dữ liệu, trong đó một vài thuộc tính là dư thừa.

```
Initial attribute Set: {X1, X2, X3, X4, X5, X6}
Initial reduced attribute set:  { }

Step-1: {X1}
Step-2: {X1, X2}
Step-3: {X1, X2, X5}

Final reduced attribute set: {X1, X2, X5}
```

Hình 9. Giảm chiều dữ liệu với lựa chọn chuyển tiếp từng bước

- **Lựa chọn ngược từng bước:** Bắt đầu với một tập hợp các thuộc tính hoàn chỉnh trong dữ liệu gốc và tại mỗi điểm, loại bỏ thuộc tính tồi tệ nhất còn lại trong tập hợp.

  Giả sử có các thuộc tính sau trong tập dữ liệu trong đó một vài thuộc tính là dư thừa.

```
Initial attribute Set: {X1, X2, X3, X4, X5, X6}
Initial reduced attribute set:  {X1, X2, X3, X4, X5, X6 }

Step-1: {X1, X2, X3, X4, X5}
Step-2: {X1, X2, X3, X5}
Step-3: {X1, X2, X5}

Final reduced attribute set: {X1, X2, X5}
```

Hình 10. Giảm chiều dữ liệu với lựa chọn ngược từng bước

2.3.2 Giảm số lượng dữ liệu

1. Phương pháp tham số

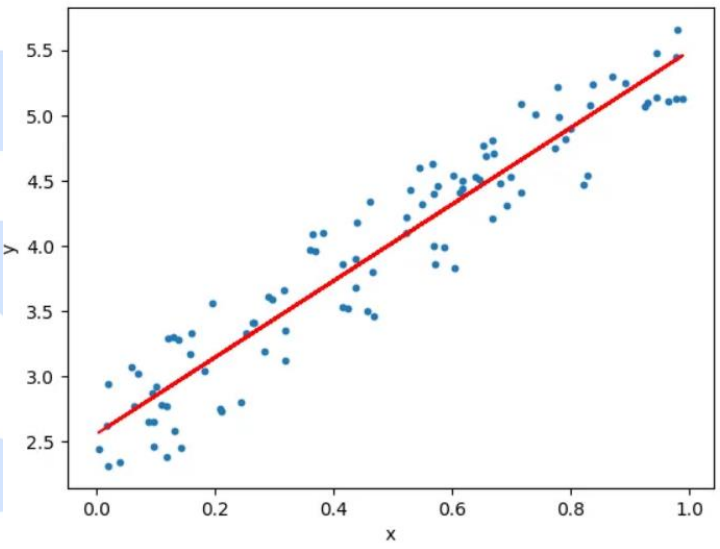
Hồi quy (Regression): Hồi quy có thể là một hồi quy tuyến tính đơn giản hoặc hồi quy đa tuyến tính.

- Khi chỉ có ***một*** thuộc tính độc lập, mô hình hồi quy được gọi là hồi quy tuyến tính đơn giản.

  Trong hồi quy tuyến tính, dữ liệu được mô hình hóa thành một đường thẳng phù hợp. Ví dụ, một biến ngẫu nhiên *y* có thể được mô hình hóa như một hàm tuyến tính của một biến ngẫu nhiên *x* khác với phương trình $y = ax + b$ trong đó *a* và *b* (hệ số hồi quy) chỉ định độ dốc và giao điểm *y* của đường thẳng.

- Nếu có ***nhiều*** thuộc tính độc lập, mô hình hồi quy được gọi là hồi quy đa tuyến tính.

  Trong hồi quy đa tuyến tính, *y* sẽ được mô hình hóa như một hàm tuyến tính của hai hoặc nhiều biến dự đoán (độc lập).



Hình 11. Mô hình Linear Regression

Ở đây, có thể hình dung dữ liệu là các điểm, mô hình cần tìm để lưu trữ các điểm dữ liệu đó chính là một đường thẳng sao cho giá trị trung bình độ lệch chuẩn giữa các điểm dữ liệu thật so với điểm dữ liệu tương ứng trên đường thẳng là nhỏ nhất.

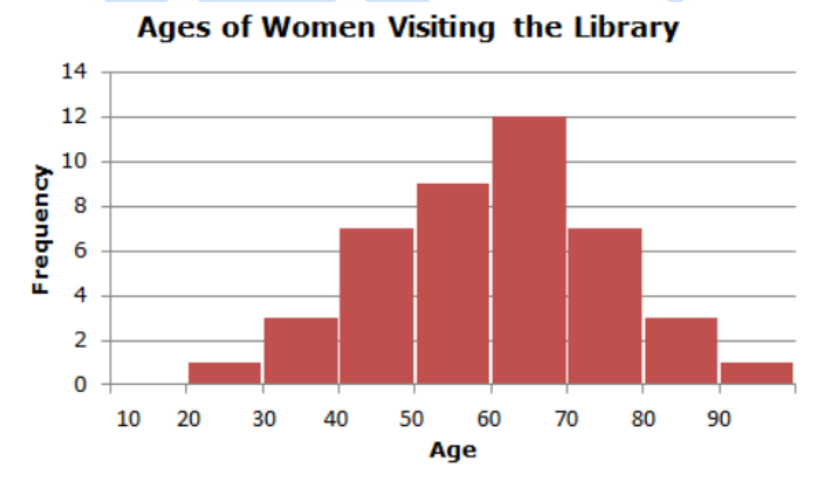
Log-Linear Models: Mô hình Log-Linear có thể được sử dụng để ước tính xác suất của mỗi điểm dữ liệu trong một không gian đa chiều cho một tập hợp các thuộc tính rời rạc, dựa trên một tập hợp con nhỏ hơn của các kết hợp chiều. Điều này cho phép một không gian dữ liệu chiều cao hơn được xây dựng từ các thuộc tính chiều thấp hơn.

Cả hai đều có thể được sử dụng trên dữ liệu thưa thớt, mặc dù ứng dụng của chúng có thể bị hạn chế.

2. Phương pháp phi tham số

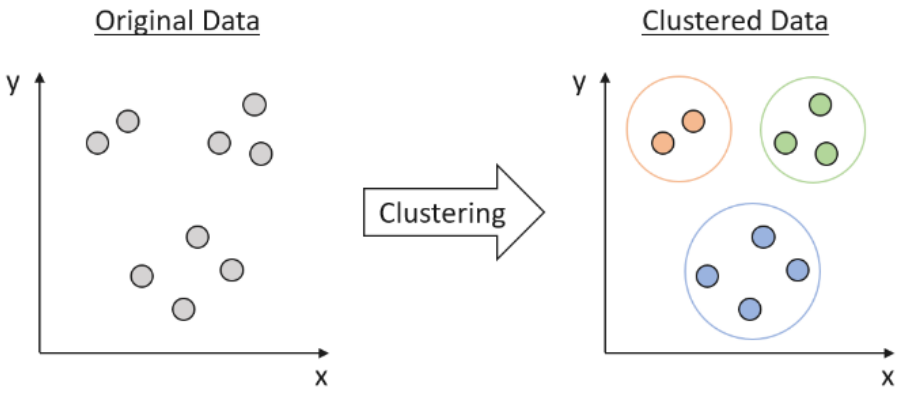
Những phương pháp này được sử dụng để lưu trữ giảm đại diện dữ liệu bao gồm *biểu đồ*, *phân cụm*, *lấy mẫu* và *tổng hợp khối dữ liệu*.

- **Biểu đồ (Histograms):** Biểu đồ tần suất biểu diễn dữ liệu về tần suất, sử dụng phương pháp chia giỏ (binning) để phân phối dữ liệu gần đúng và là một hình thức rút gọn dữ liệu phổ biến.



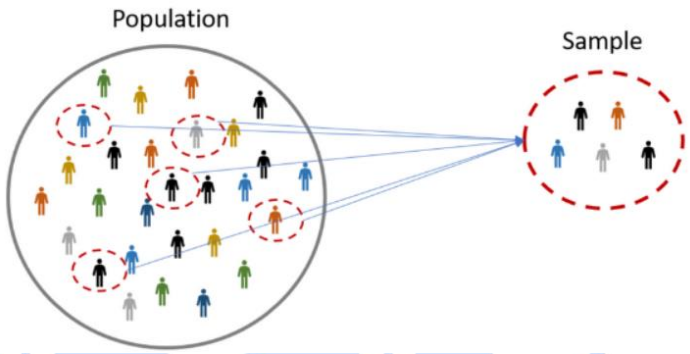
Hình 12. Biểu đồ thể hiện tần suất ghé thăm thư viện của nữ giới theo độ tuổi

- **Gom cụm (Clustering):** Phân vùng toàn bộ dữ liệu thành các nhóm/cụm khác nhau. Trong rút gọn dữ liệu, biểu diễn cụm của dữ liệu được sử dụng để thay thế dữ liệu thực tế. Nó cũng giúp phát hiện các ngoại lệ trong dữ liệu.



Hình 13. Minh họa phương pháp gom cụm

- **Lấy mẫu (Sampling):** Cho phép một tập dữ liệu lớn được biểu diễn bằng một mẫu dữ liệu ngẫu nhiên nhỏ hơn nhiều (hoặc tập hợp con).



Hình 14. Minh họa phương pháp lấy mẫu

- **Tổng hợp khối dữ liệu (Data Cube Aggregation):** Di chuyển dữ liệu từ mức chi tiết sang số lượng thứ nguyên ít hơn. Tập dữ liệu kết quả có khối lượng nhỏ hơn, không mất thông tin cần thiết cho nhiệm vụ phân tích.

2.3.3 Nén dữ liệu

Kỹ thuật nén dữ liệu làm giảm kích thước của các tệp bằng các cơ chế mã hóa khác nhau (Mã hóa Huffman &; Mã hóa độ dài chạy). Có thể phân chia thành hai loại chính:

1. **Nén dữ liệu không mất mát (Lossless)**

   Về cơ bản, kiểu nén Lossless sẽ đơn giản hóa các phần dữ liệu dư thừa, không cần thiết nhưng không làm mất dữ liệu. Dữ liệu gốc được nén lại và sau khi giải nén sẽ thu được bộ dữ liệu y hệt ban đầu.

Có rất nhiều dạng nén Lossless sử dụng các thuật toán khác nhau nhưng về bản chất, các dữ liệu sau khi được nén đều có thể được tái tạo lại toàn bộ như dữ liệu ban đầu sau khi giải nén, không có mất mát và thay đổi gì.

Thông thường, các phần mềm như WinZip đều dựa trên kiểu nén Lossless này. Các file dữ liệu sau khi bị nén sẽ có dung lượng nhỏ hơn, tuy nhiên sau khi giải nén dữ liệu sẽ được khôi phục lại như ban đầu, không có phần dữ liệu nào bị mất đi. Trong các file hình ảnh, PNG cũng là một định dạng được nén theo kiểu Lossless.

Các dữ liệu media hiện nay, đặc biệt là file âm thanh cũng thường được sử dụng kiểu nén Lossless vì nó đảm bảo dữ liệu không bị mất đi và giống như bản gốc. Mặc dù có rất nhiều định dạng Lossless khác nhau (FLAC, APE, TAK, ALAC, TTA, WAV) do đó cũng có nhiều kiểu nén Lossless khác nhau cho ra các file với dung lượng từ cao đến thấp. Nhưng sau khi giải nén, chúng đều cho ra dữ liệu với chất lượng giống nhau và giống bản gốc (nếu cùng một bản gốc). Do đó việc so sánh dung lượng và birate của các file nhạc Lossless là điều không cần thiết.

2. **Nén dữ liệu có mất mát (Lossy)**

Ngược lại với Lossless, kiểu nén Lossy loại bỏ hoàn toàn một phần của dữ liệu, nghĩa là có sự sai khác giữa dữ liệu gốc trước khi nén và sau khi giải nén, có sự mất mát dữ liệu.

Điều này khá tồi tệ đối với các dữ liệu văn bản, vì có thể bị cắt mất một vài dòng văn bản sau khi giải nén do một phần dữ liệu bị loại bỏ trong quá trình nén.

Tuy nhiên, rất nhiều dữ liệu media lại được sử dụng kiểu nén này. Các file MP3 là một ví dụ điển hình, hầu hết các file nhạc số lưu trữ trên internet đều sử dụng định dạng này vì nó rất nhẹ, dung lượng có thể chỉ bằng 1/10 so với bản gốc. Một số âm thanh của bản nhạc sẽ bị loại bỏ, đa số là các âm thanh mà chúng ta khó có thể nghe thấy. Tuy nhiên nếu thực

hiện nén với dung lượng càng nhỏ, lượng dữ liệu mất đi sẽ càng lớn và chất lượng âm thanh sẽ rất kém.

Một định dạng phổ biến khác là file hình ảnh JPEG. Hầu hết các bức ảnh chia sẻ trên internet đều không cần phải có chất lượng cao như phục vụ cho in ấn, mà thay vào đó là dung lượng nhỏ để dễ dàng chia sẻ. Do đó, bạn có thể loại một phần dữ liệu, như độ phân giải và các điểm ảnh giảm đi. Càng nén nhiều lần, chất lượng hình ảnh sẽ càng giảm cùng với dung lượng, tuy nhiên điều đó còn phụ thuộc vào mục đích sử dụng của bức ảnh, nên chất lượng không phải lúc nào cũng được đặt lên hàng đầu.

```
file_path = pathlib.Path("insurance.csv")
file_pathz = pathlib.Path("insurance.csv.gz")
file_size = file_path.stat().st_size
file_size_zip = file_pathz.stat().st_size
print("file size :", file_size, "bytes")
print("file size zip :", file_size_zip, "bytes")
```

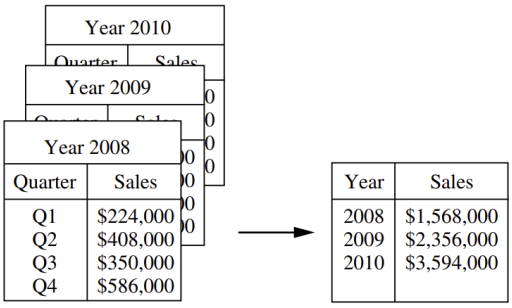
```
file size : 55628 bytes
file size zip : 15639 bytes
```

Hình 15. Ví dụ nén dữ liệu

Theo như hình 15, ta thấy khi nén 1 tập tin (cụ thể là 1 file csv) thì ta thu được kích thước đã giảm đi rất nhiều, khoảng 72% so với kích thước dữ liệu ban đầu. Từ đó, khi lưu trữ khai thác dữ liệu lớn hơn thì ta sẽ tiết kiệm được tài nguyên lưu trữ.

2.3.4 Tổng hợp khối dữ liệu

Xem xét tình hình dữ liệu của một công ty. Dữ liệu này bao gồm doanh số mỗi quý, trong những năm 2008 đến 2010. Bây giờ, ta quan tâm đến doanh thu hàng năm (tổng số mỗi năm), thay vì tổng số mỗi quý. Do đó, dữ liệu có thể được giảm để dữ liệu kết quả tóm tắt tổng doanh thu mỗi năm thay vì mỗi quý.



Hình 16. Minh họa tổng hợp khối dữ liệu

Khối dữ liệu lưu trữ thông tin tổng hợp (tóm tắt) đa chiều. Mỗi ô giữ một giá trị dữ liệu tổng hợp, tương ứng với điểm dữ liệu trong không gian đa chiều.

Bằng cách này, tổng hợp cung cấp dữ liệu cần thiết có kích thước nhỏ hơn nhiều và do đó ta đạt được mức giảm dữ liệu ngay cả khi không mất bất kỳ dữ liệu nào.

Các khối dữ liệu cung cấp quyền truy cập nhanh vào dữ liệu tóm tắt, được tính toán trước, do đó mang lại lợi ích cho quá trình xử lý phân tích trực tuyến cũng như khai thác dữ liệu.

II. Tham khảo

[1] [Online]. Available: http://hanj.cs.illinois.edu/cs412/bk3/03.pdf.

[2] [Online]. Available: https://web.cs.hacettepe.edu.tr/~ilyas/Courses/VBM684/lec03-DataPreprocessing.pdf.