



TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH.
HCMC University of Technology and Education

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO
BỘ MÔN CƠ ĐIỆN TỬ

ĐỒ ÁN MÔN HỌC

ASSOCIATION RULE ALGORITHMS



MÃ MÔN HỌC: GELA220405

GVHD: VŨ QUANG HUY

SVTH: PHẠM CÔNG THÀNH – 18146213

BÙI TRỌNG QUÝ – 18144281

NGUYỄN ANH KHOA – 18146148

TP. HỒ CHÍ MINH, THÁNG 5 NĂM 2021

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN.....	3
1.1. Đặt vấn đề	3
1.2. Mục tiêu.....	3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	4
2.1. Tổng quan về dữ liệu và cơ sở dữ liệu	4
2.2. Tổng quan về tri thức và phát hiện tri thức trong cơ sở dữ liệu	6
2.3. Khai phá dữ liệu	8
2.3.1. Khái niệm về khai phá dữ liệu.....	8
2.3.2. Quá trình khai phá dữ liệu	8
2.3.3. Các kỹ thuật khai phá dữ liệu	9
2.4. Khái quát về khai thác tập phổ biến và luật kết hợp	10
2.4.1. Khái niệm và định nghĩa	10
2.4.2. Các phương pháp khai phá tập phổ biến và luật kết hợp	10
2.5. Cơ sở lý thuyết của giải thuật Apriori.....	11
2.5.1 Các khái niệm	11
2.5.2 Một số luật kết hợp	12
2.5.3 Xây dựng thuật toán Apriori.....	12
2.6. Cơ sở lý thuyết của giải thuật Eclat.....	15
2.7. Cơ sở lý thuyết của giải thuật FP-growth.....	18
CHƯƠNG 3: DỮ LIỆU	22
3.1 Mô tả tập dữ liệu	22
3.2 Tiêu chuẩn đánh giá dữ liệu.....	22
3.3 Quá trình tiền xử lý dữ liệu	23
3.3.1 Chuẩn bị dữ liệu	24
3.3.1.1 Làm sạch dữ liệu	24
3.3.1.1 Chuyển hóa dữ liệu	24
3.3.1.2 Tích hợp dữ liệu	24
3.3.1.3 Chuẩn hóa dữ liệu.....	25

3.3.1.4	Xử lý dữ liệu bị thiếu	25
3.3.1.5	Xác định nhiễu.....	25
3.3.2	Tối giản dữ liệu	25
3.3.2.1	Chọn thuộc tính.....	25
3.3.2.2	Chọn mẫu.....	26
3.3.2.3	Rời rạc hóa dữ liệu.....	26
3.3.2.4	Mở rộng thuộc tính / Tạo mẫu.....	26
CHƯƠNG 4: GIẢI THUẬT VÀ CHƯƠNG TRÌNH.....		26
4.1	Giải thuật Apriori	26
4.2	Giải thuật Eclat	27
4.3	Giải thuật FP-growth.....	30
CHƯƠNG 5: KẾT QUẢ VÀ PHÂN TÍCH		31
5.1	Mô tả cách huấn luyện.....	31
5.2	Phân tích kết quả.....	31
TÀI LIỆU THAM KHẢO		33

CHƯƠNG 1: TỔNG QUAN

1.1. Đặt vấn đề

Thuật ngữ và kỹ thuật khai phá dữ liệu ra đời vào những năm 80 của thế kỷ trước. Khai phá dữ liệu nhằm tìm ra được tri thức ẩn chứa bên trong dữ liệu đang có. Để thực hiện các quá trình khám phá tri thức bên trong dữ liệu, các nhà chuyên gia đã phát minh ra nhiều giải thuật tiên tiến. Các thuật toán được cộng đồng và các chuyên gia tối ưu theo năm tháng. Hiện nay, nhờ có sự phát triển của các ngành khoa học máy tính và các ngành công nghệ thông tin, nên chúng ta có thể áp dụng và sử dụng các giải thuật phức tạp này.

Các kỹ thuật khai phá dữ liệu đang được sử dụng rất phổ biến và rộng rãi. Nó xuất hiện trong nhiều lĩnh vực kinh doanh và đời sống khác nhau như: thương mại (phân tích dữ liệu bán hàng và thị trường, ...), thông tin sản xuất (điều khiển và lập kế hoạch, ...)...

Những thông tin, tri thức thu được từ quá trình khai phá dữ liệu rất đáng giá. Chính vì thế mà trong tương lai, rất có thể sẽ có thêm nhiều thuật toán tối ưu hơn được phát minh bởi những nhà tiên phong trong lĩnh vực này.

1.2. Mục tiêu

Mục tiêu của đồ án này là tìm hiểu kỹ về data mining (khai phá dữ liệu) và các kỹ thuật khai phá dữ liệu kết hợp. Sau đó áp dụng lý thuyết để làm một mô hình phân tích dữ liệu kết hợp trên python.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về dữ liệu và cơ sở dữ liệu

Khái niệm dữ liệu có thể hiểu là thông tin, thường là số, được thu thập thông qua quá trình quan sát, đánh giá. Theo góc nhìn kỹ thuật, đơn cử là ngành khoa học máy tính hoặc các ngành liên quan đến lập trình, ta thường liên tưởng dữ liệu bằng các dãy số gồm 0 và 1, còn gọi là bit. Trong các ngành liên quan đến kinh doanh, để mang lại lợi ích kinh doanh tốt nhất thì những người chủ phải biết được một vài thông tin về những khách hàng mà chủ thể kinh doanh đó tiếp cận, thông tin đó được biết là “dữ liệu khách hàng”.

Từ khái niệm dữ liệu, ta có một cách nhìn tổng quan về "cơ sở dữ liệu". Cơ sở dữ liệu đề cập đến một tập hợp các dữ liệu liên quan và cách thức tổ chức, sắp xếp của dữ liệu. Truy cập vào dữ liệu này thường được cung cấp bởi "hệ thống quản lý cơ sở dữ liệu" (DBMS) bao gồm một bộ phần mềm máy tính tích hợp cho phép người dùng tương tác với một hoặc nhiều cơ sở dữ liệu và cung cấp quyền truy cập vào tất cả dữ liệu có trong cơ sở dữ liệu (mặc dù hạn chế có thể tồn tại giới hạn truy cập vào dữ liệu cụ thể). Hệ thống quản lý cơ sở dữ liệu cung cấp các chức năng khác nhau cho phép nhập, lưu trữ và truy xuất số lượng lớn thông tin và cung cấp các cách để quản lý cách thức tổ chức thông tin đó.

Khái niệm “cơ sở dữ liệu” xuất hiện từ giữa những năm 1960 cùng với sự sẵn có của bộ nhớ, bộ nhớ khả biến (RAM), mà sau đó đã xuất hiện nhiều cột mốc phát triển của cơ sở dữ liệu:

- Những năm 1960, hệ thống quản lý cơ sở dữ liệu hướng đối tượng.
- Những năm 1970, hệ thống quản lý cơ sở dữ liệu quan hệ
- Trong những năm 1970 và 1980, Phương pháp tích hợp
- Cuối những năm 1970, SQL DBMS

Từ đó cho đến nay, rất nhiều cơ sở dữ liệu đã được tổ chức, phát triển và khai thác ở mọi quy mô và các lĩnh vực hoạt động của con người và xã hội. Theo như đánh giá cho thấy, lượng thông tin trên thế giới cứ sau 20 tháng lại tăng lên gấp đôi. Kích thước và số lượng cơ sở dữ liệu thậm chí còn tăng nhanh hơn. Với sự phát triển của công nghệ điện tử, sự phát triển mạnh mẽ của công nghệ phần cứng tạo ra các bộ nhớ có dung lượng lớn, bộ xử lý có tốc độ cao cùng với sự phát triển của các hệ thống viễn thông, người ta đã và đang xây dựng các hệ thống thông tin nhằm tự động hoá mọi hoạt động của con người. Điều này đã tạo ra một dòng dữ liệu tăng lên không ngừng vì ngay cả những hoạt động đơn giản như gọi điện thoại, tra cứu sách trong thư viện, ... đều được thực hiện thông qua máy tính. Cho đến nay, số lượng cơ sở dữ liệu đã trở nên khổng lồ bao gồm các cơ sở dữ

liệu cực lớn cỡ gigabytes và thậm chí terabytes lưu trữ các dữ liệu kinh doanh ví dụ như dữ liệu thông tin khách hàng, dữ liệu bán hàng, dữ liệu các tài khoản, ... Nhiều hệ quản trị cơ sở dữ liệu mạnh với các công cụ phong phú và thuận tiện đã giúp con người khai thác có hiệu quả nguồn tài nguyên dữ liệu. Mô hình cơ sở dữ liệu quan hệ và ngôn ngữ vấn đáp chuẩn (SQL) đã có vai trò hết sức quan trọng trong việc tổ chức và khai thác cơ sở dữ liệu. Cho đến nay, không một tổ chức nào sử dụng tin học trong công việc mà không sử dụng các hệ quản trị cơ sở dữ liệu và các hệ công cụ báo cáo, ngôn ngữ hỏi đáp nhằm khai thác cơ sở dữ liệu phục vụ cho các hoạt động tác nghiệp của mình. Cùng với việc tăng không ngừng khối lượng dữ liệu, các hệ thống thông tin cũng được chuyên môn hoá, phân chia theo lĩnh vực ứng dụng như sản xuất, tài chính, hoạt động kinh doanh, Như vậy bên cạnh chức năng khai thác dữ liệu có tính chất tác nghiệp, sự thành công trong công việc không còn là năng suất của các hệ thống thông tin nữa mà là tính linh hoạt và sẵn sàng đáp lại những yêu cầu trong thực tế, cơ sở dữ liệu cần đem lại những “tri thức” hơn là chính những dữ liệu trong đó. Các quyết định cần phải có càng nhanh càng tốt và phải chính xác dựa trên những dữ liệu sẵn có trong khi khối lượng dữ liệu cứ sau 20 tháng lại tăng gấp đôi làm ảnh hưởng đến thời gian ra quyết định cũng như khả năng hiểu hết được nội dung dữ liệu. Lúc này, các mô hình cơ sở dữ liệu truyền thống và ngôn ngữ SQL đã cho thấy không có khả năng thực hiện công việc này. Để lấy thông tin có tính “tri thức” trong khối dữ liệu khổng lồ này, người ta đã tìm ra những kỹ thuật có khả năng hợp nhất các dữ liệu từ các hệ thống giao dịch khác nhau, chuyển đổi thành một tập hợp các CSDL ổn định, có chất lượng được sử dụng chỉ cho riêng một vài mục đích nào đó. Các kỹ thuật đó gọi chung là kỹ thuật tạo kho dữ liệu (data warehousing) và môi trường các dữ liệu có được gọi là các kho dữ liệu (data warehouse).

Nhưng chỉ có kho dữ liệu thôi chưa đủ để có tri thức. Các kho dữ liệu được sử dụng theo một số cách như:

Theo cách khai thác truyền thống: tức là kho dữ liệu được sử dụng để khai thác các thông tin bằng các công cụ truy vấn và báo cáo.

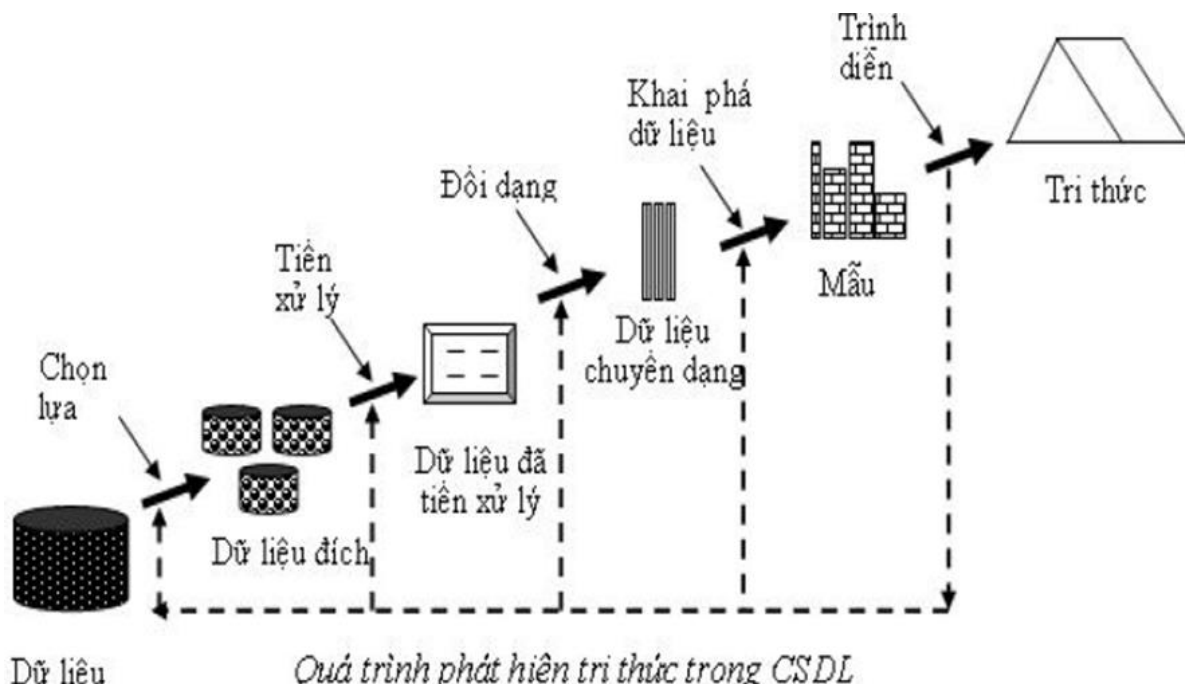
Các kho dữ liệu được sử dụng để hỗ trợ cho phân tích trực tuyến (OLAPOnLine Analytical Processing): Việc phân tích trực tuyến có khả năng phân tích dữ liệu, xác định xem giả thuyết đúng hay sai. Tuy nhiên, phân tích trực tuyến lại không có khả năng đưa ra các giả thuyết.

Công nghệ khai phá dữ liệu (data mining) ra đời đáp ứng những đòi hỏi trong khoa học cũng như trong hoạt động thực tiễn. Đây chính là một ứng dụng chính của kho dữ liệu.

2.2. Tổng quan về tri thức và phát hiện tri thức trong cơ sở dữ liệu

Tri thức là một khái niệm rất trừu tượng, nhưng nói đến tri thức của con người, chúng ta thường nghĩ ngay đến bao gồm những kiến thức, thông tin, sự hiểu biết, hay kỹ năng có được nhờ trải nghiệm, thông qua giáo dục hay tự học hỏi của mỗi người. Tương tự, tri thức trong dữ liệu có thể hiểu đơn giản là kiến thức hay sự hiểu biết mà một cá nhân hay tổ chức đạt được khi sử dụng, phân tích và đánh dữ liệu.

Phát hiện tri thức là quá trình tìm ra những thông tin tiềm ẩn có giá trị mà trước đó chưa được phát hiện, tìm ra những xu hướng phát triển và những yếu tố tác động lên chúng. Đây là một vấn đề rất quan trọng, việc sử dụng thông tin một cách có hiệu quả có thể tác động trực tiếp đến khả năng thành công trong công việc của chúng ta, đặc biệt là các mảng chuyên ngành như machine learning, data analytics,...

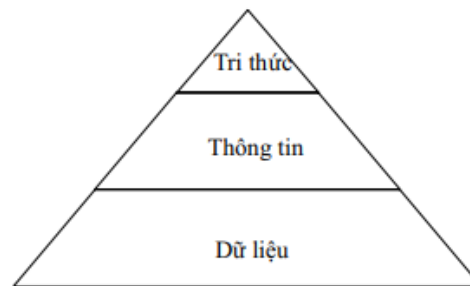


Hình 1. Quá trình phát hiện tri thức

Trước khi đi vào tìm hiểu các giai đoạn trong quá trình phát hiện tri thức ta xem xét một số ví dụ để phân biệt ba khái niệm: dữ liệu, thông tin và tri thức. Như đã đề cập ở trên dữ liệu thường được cho bởi các giá trị mô tả các sự kiện, hiện tượng cụ thể, còn tri thức tri thức là một khái niệm rất trừu tượng.

So với dữ liệu thì tri thức có số lượng ít hơn rất nhiều. Thuật ngữ ít ở đây không chỉ đơn giản là một dấu nhỏ hơn bình thường mà là sự kết tinh hoặc cô đọng lại. Ta hãy hình dung dữ liệu như là những điểm trên mặt phẳng còn tri thức chính là phương trình của

đường cong nối tất cả những điểm này lại. Chỉ cần một phương trình đường cong ta có thể biểu diễn được vô số điểm



Hình 2. Mối quan hệ giữa thông tin, dữ liệu và tri thức

Trong hình 1, ta thấy quá trình phát hiện tri thức gồm nhiều giai đoạn. Đầu ra của giai đoạn này là đầu vào của giai đoạn sau. Trong tiến trình này, người ta đặc biệt quan tâm đến pha khai phá dữ liệu (Data mining). Khai phá dữ liệu chính là sử dụng những kỹ thuật, những phương pháp để đưa ra những thông tin có cấu trúc, những tri thức tiềm ẩn trong lượng dữ liệu. Các kỹ thuật phát hiện tri thức được thực hiện qua nhiều giai đoạn và sử dụng nhiều phương pháp như: phân lớp, gom cụm, phân tích sự tương tự, tổng hợp, phát hiện luật kết hợp và mẫu tuần tự,... Quá trình phát hiện tri thức gồm các bước cơ bản sau:

- Chọn lọc dữ liệu (selection): là quá trình xác định loại dữ liệu và nguồn thích hợp, cũng như các công cụ thích hợp để thu thập dữ liệu.
- Tiền xử lý dữ liệu (preprocessing): lý do phải có quá trình này là phần lớn các cơ sở dữ liệu đều ít nhiều mang tính không nhất quán. Trên thực tế, không có gì là hoàn hảo hay tuyệt đối, chính vì thế khi gom dữ liệu rất có thể mắc một số lỗi như dữ liệu không đầy đủ, chặt chẽ và không logic (bị trùng lặp, giá trị bị sai lệch...). Quá trình tiền xử lý dữ liệu cho một dự án nghiên cứu có thể ảnh hưởng đến kết quả đầu ra.
- Chuyển đổi dữ liệu (transformation): Máy, máy tính... không thể hiểu chữ viết tay, hình ảnh, đoạn phim, chúng chỉ có thể hiểu những dãy số gồm 0 và 1. Trong machine learning, chuyển đổi dữ liệu là bước mà dữ liệu được chuyển đổi hay mã hóa thành dạng mà máy có thể hiểu được.
- Khai phá dữ liệu (data mining): trong giai đoạn này ta sử dụng các kỹ thuật nhằm phát hiện ra các tri thức tiềm ẩn trong dữ liệu. Một số kỹ thuật được sử dụng đó là: phân lớp, gom cụm, luật kết hợp...
- Đánh giá kết quả mẫu: Đây là giai đoạn cuối cùng trong tiến trình phát hiện tri thức. Trong giai đoạn này, các mẫu dữ liệu được chiết xuất bởi các phần mềm khai phá dữ liệu. Không phải bất cứ mẫu nào cũng đều có ích, thậm chí còn bị sai lệch. Chính vì vậy, cần

phải xác định và lựa chọn những tiêu chuẩn đánh giá sao cho sẽ chiết xuất ra các tri thức cần thiết.

2.3. Khai phá dữ liệu

Khai phá dữ liệu (data mining) đã và đang được ứng dụng trong rất nhiều lĩnh vực như:

- Lĩnh vực tài chính
- Lĩnh vực chăm sóc sức khỏe
- Lĩnh vực viễn thông
- Lĩnh vực Marketing và Sales
- Lĩnh vực thương mại điện tử
- Lĩnh vực giáo dục

2.3.1. Khái niệm về khai phá dữ liệu

Khái niệm khai phá dữ liệu ra đời vào những năm 1700s với cột mốc quan trọng đầu tiên là định lý Bayes. Cụ thể hơn, năm 1763, di cảo của Thomas Bayes về định lý xác suất đã được đăng sau hai năm kể từ khi Thomas Bayes qua đời. Định lý Bayes chính là nền tảng cho khai phá dữ liệu và xác suất, định lý này cho chúng ta hiểu được sự phức tạp của thực tế bằng những bài toán ước lượng.

Đến với cột mốc quan trọng thứ hai, năm 1989, Fayyad, Piatetsky-Shapiro và Smyth đã dùng khái niệm phát hiện tri thức trong cơ sở dữ liệu (Knowledge Discovery in Database – KDD) để chỉ toàn bộ quá trình phát hiện các tri thức có ích từ các tập dữ liệu lớn. Trong đó, khai phá dữ liệu là một bước đặc biệt trong toàn bộ tiến trình, sử dụng các giải thuật đặc biệt để chiết xuất ra các mẫu (pattern) (hay các mô hình) từ dữ liệu. Khai phá dữ liệu là một tiến trình sử dụng các công cụ phân tích dữ liệu khác nhau để khám phá ra các mẫu dưới nhiều góc độ khác nhau nhằm phát hiện ra các mối quan hệ giữa các dữ kiện, đối tượng bên trong cơ sở dữ liệu, kết quả của việc khai phá là xác định các mẫu hay các mô hình đang tồn tại bên trong, nhưng chúng nằm ẩn khuất ở các cơ sở dữ liệu. Để từ đó rút trích ra được các mẫu, các mô hình hay các thông tin và tri thức từ các cơ sở dữ liệu.

2.3.2. Quá trình khai phá dữ liệu

Các giải thuật khai phá dữ liệu thường được mô tả như những chương trình hoạt động trực tiếp trên tệp dữ liệu. Với các phương pháp học máy và thống kê trước đây, thường thì bước đầu tiên là các giải thuật nạp toàn bộ tệp dữ liệu vào trong bộ nhớ. Khi chuyển sang các ứng dụng công nghiệp liên quan đến việc khai phá các kho dữ liệu lớn, mô hình này không thể đáp ứng được. Không chỉ bởi vì nó không thể nạp hết dữ liệu vào trong bộ

nhớ mà còn vì khó có thể chiết xuất dữ liệu ra các tệp đơn giản để phân tích được. Quá trình xử lý khai phá dữ liệu bắt đầu bằng cách xác định chính xác vấn đề cần giải quyết. Sau đó sẽ xác định các dữ liệu liên quan dùng để xây dựng giải pháp. Bước tiếp theo là thu thập các dữ liệu có liên quan và xử lý chúng thành dạng sao cho giải thuật khai phá dữ liệu có thể hiểu được. Về lý thuyết thì có vẻ rất đơn giản nhưng khi thực hiện thì đây thực sự là một quá trình rất khó khăn, gặp phải rất nhiều vướng mắc như: các dữ liệu phải được sao ra nhiều bản (nếu được chiết xuất vào các tệp), quản lý tập các tệp dữ liệu, phải lặp đi lặp lại nhiều lần toàn bộ quá trình (nếu mô hình dữ liệu thay đổi),... Bước tiếp theo là chọn thuật toán khai phá dữ liệu thích hợp và thực hiện việc khai phá dữ liệu để tìm được các mẫu (pattern) có ý nghĩa dưới dạng biểu diễn tương ứng với các ý nghĩa đó (thường được biểu diễn dưới dạng các luật xếp loại, cây quyết định, luật sản xuất, biểu thức hồi quy,...). Đặc điểm của mẫu phải là mới (ít nhất là đối với hệ thống đó). Độ mới có thể được đo tương ứng với độ thay đổi trong dữ liệu (bằng cách so sánh các giá trị hiện tại với các giá trị trước đó hoặc các giá trị mong muốn), hoặc bằng tri thức (mối liên hệ giữa phương pháp tìm mới và phương pháp cũ như thế nào). Thường thì độ mới của mẫu được đánh giá bằng một hàm logic hoặc một hàm đo độ mới, độ bất ngờ của mẫu. Ngoài ra, mẫu còn phải có khả năng sử dụng tiềm tàng. Các mẫu này sau khi được xử lý và diễn giải phải dẫn đến những hành động có ích nào đó được đánh giá bằng một hàm lợi ích. Mẫu khai thác được phải có giá trị đối với các dữ liệu mới với độ chính xác nào đó.

Kỹ thuật khai phá dữ liệu thực chất là phương pháp không hoàn toàn mới. Nó là sự kế thừa, kết hợp và mở rộng của các kỹ thuật cơ bản đã được nghiên cứu từ trước như máy học, nhận dạng, thống kê (hồi quy, xếp loại, phân cụm), các mô hình đồ thị, các mạng Bayes, trí tuệ nhân tạo, thu thập tri thức hệ chuyên gia, v.v... Tuy nhiên, với sự kết hợp tài tình của khai phá dữ liệu, kỹ thuật này có ưu thế hơn hẳn các phương pháp trước đó, đem lại nhiều triển vọng trong việc ứng dụng phát triển nghiên cứu khoa học.

2.3.3. Các kỹ thuật khai phá dữ liệu

Khai thác tập phổ biến và luật kết hợp: (frequent itemset) là lớp bài toán rất quan trọng trong lĩnh vực khai phá dữ liệu. Mục tiêu khi dùng bài toán này là tìm tất cả các tập, giá trị hay các mối tương quan, có độ phổ biến cao trong cơ sở dữ liệu. Kỹ thuật khai phá tập phổ biến và luật kết hợp được ứng dụng trong rất nhiều lĩnh vực, nổi tiếng nhất là bài toán Market Basket Analysis, áp dụng vào các khía cạnh như tiếp thị chéo, trưng bày sản phẩm nhằm tăng năng suất kinh doanh.

Phân lớp dữ liệu: (classification) phân lớp dữ liệu là một trong những bài toán lớn của Machine learning. Đây là một kỹ thuật phân loại một hay một nhóm đối tượng vào một hay nhiều lớp đã được cho trước. Kỹ thuật đã, đang và sẽ có nhiều ứng dụng trong các

lĩnh vực thương mại, ngân hàng, y tế, giáo dục. Ví dụ, chúng ta có thể dùng kỹ thuật phân lớp này để phân loại khách hàng, phân loại sản phẩm, ...

Gom cụm: (cluster analysis) khác với phân lớp dữ liệu, dữ liệu đưa vào của bài toán gom cụm không có những lớp, hay nhãn dán cho trước. Mục đích khi dùng thuật toán này là gom nhóm một tập các đối tượng theo cách các đối tượng cùng nhóm sẽ có tính giống nhau (theo các đặc tính nào đó càng giống càng tốt), giữa các nhóm có sự khác biệt (theo các đặc tính nào đó). Phân tích cụm là một tác vụ chính của khai phá dữ liệu, và là một kỹ thuật phổ biến trong thống kê phân tích dữ liệu, nó được ứng dụng trong nhiều lĩnh vực như nhận dạng mẫu, phân tích ảnh, truy hồi thông tin, ...

2.4. Khái quát về khai thác tập phổ biến và luật kết hợp

2.4.1. Khái niệm và định nghĩa

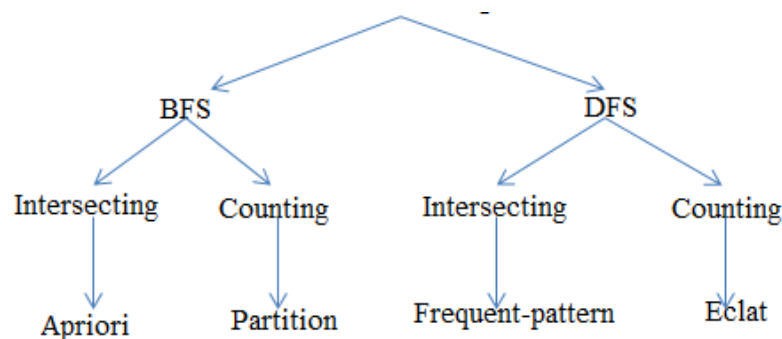
Khai thác tập phổ biến dựa trên quy tắc để khám phá các mối quan hệ thú vị giữa các biến trong cơ sở dữ liệu lớn. Một luật kết hợp $X \rightarrow Y$ phản ánh sự xuất hiện của tập X dẫn đến sự xuất hiện đồng thời của tập Y . Chẳng hạn như việc phân tích cơ sở dữ liệu tại một cửa hàng bán máy tính, ta có thể nhận được thông tin về những khách hàng có khuynh hướng mua máy tính cùng với phần mềm quản lý nhân sự trong cùng lần mua.

2.4.2. Các phương pháp khai phá tập phổ biến và luật kết hợp

Hiện đã có rất nhiều các phương pháp, kỹ thuật khai phá tập phổ biến trong. Ở đây chỉ xét đến ba kỹ thuật, trong đó có một kỹ thuật cơ bản, là kỹ thuật khởi đầu làm nền tảng cho nhiều kỹ thuật khai phá tập phổ biến sau này, đó chính là Apriori.

Ba giải thuật tiêu biểu cho bài toán khai phá tập phổ biến:

- Apriori
- Frequent-pattern
- Eclat



Hình 3. Hệ thống hóa một số giải thuật khai phá tập phổ biến

2.5. Cơ sở lý thuyết của giải thuật Apriori

Thuật toán Apriori được công bố bởi R. Agrawal và R. Srikant vào năm 1994 để tìm các tập phổ biến trong một bộ dữ liệu lớn. Tên của thuật toán là Apriori vì nó sử dụng kiến thức đã có từ trước (prior) về các thuộc tính, vật phẩm thường xuyên xuất hiện trong cơ sở dữ liệu. Để cải thiện hiệu quả của việc lọc các tập thường xuyên xuất hiện, một thuộc tính quan trọng được sử dụng gọi là thuộc tính Apriori giúp giảm phạm vi tìm kiếm của thuật toán.

Việc thuật toán Apriori có thể làm là nhìn vào dữ liệu của những gì đã xảy ra và khẳng định rằng nếu một việc gì đó xảy ra thì sẽ có tỉ lệ bao nhiêu phần trăm sự việc khác sẽ xảy ra. Ví dụ một siêu thị muốn nghĩ cách sắp xếp các gian hàng một cách hợp lý nhất, họ có thể nhìn vào lịch sử mua hàng và sắp xếp các tập sản phẩm thường được mua cùng nhau vào một chỗ. Hoặc một trang web bán hàng có thể áp dụng để giới thiệu sản phẩm mà khách hàng có thể muốn mua.

2.5.1 Các khái niệm

Tiếp theo chúng ta sẽ sơ lược một số khái niệm trong giải thuật Apriori:

- Giả sử thuật toán apriori được áp dụng trong lĩnh vực mua hàng. Cho cơ sở dữ liệu gồm các giao dịch T là tập các giao dịch t_1, t_2, \dots, t_n . $T = \{t_1, t_2, \dots, t_n\}$. T gọi là cơ sở dữ liệu giao dịch

- Mỗi giao dịch t_i bao gồm tập các sản phẩm I gọi là itemset. $I = \{i_1, i_2, \dots, i_m\}$. Một itemset gồm k items gọi là k -itemset.

- Mục đích của luật kết hợp là tìm ra sự kết hợp hay tương quan giữa các items. Những luật kết hợp này có dạng $X \Rightarrow Y$. Luật kết hợp $X \Rightarrow Y$ có thể hiểu rằng những người mua các mặt hàng trong tập X cũng thường mua các mặt hàng trong tập Y . Ví dụ, nếu $X = \{\text{Cam, chuối}\}$ và $Y = \{\text{Dưa hấu, xoài}\}$ và ta có luật kết hợp $X \Rightarrow Y$ thì chúng ta có thể nói rằng những người mua cam và chuối thì cũng thường mua dưa hấu và xoài.

- Độ hỗ trợ (Support) và độ tin cậy (Confidence) là 2 tham số dùng để đo lường luật kết hợp. Độ hỗ trợ (Support) của luật kết hợp $X \Rightarrow Y$ là tần suất của giao dịch chứa tất cả các items trong cả hai tập X và Y . Ví dụ, support của luật $X \Rightarrow Y$ là 5% có nghĩa là 5% các giao dịch X và Y được mua cùng nhau.

- Công thức để tính support của luật $X \Rightarrow Y$ như sau:

$$\text{Supp}(X) = \frac{\text{count}(X)}{D}$$

Trong đó, D là số tổng số lần mua hàng và count(X) là số lần X xuất hiện

- Độ tin cậy (Confidence) của luật kết hợp $X \Rightarrow Y$ là xác suất xảy ra Y khi đã biết X. Ví dụ độ tin cậy của luật kết hợp {bưởi} \Rightarrow {cam} là 80% có nghĩa là 80% khách hàng mua bưởi cũng mua cam. Công thức để tính độ tin cậy của luật kết hợp $X \Rightarrow Y$ là xác suất có điều kiện Y khi đã biết X như sau:

$$\text{Conf}(X) = \frac{\text{count}(X+Y)}{\text{count}(X)}$$

Trong đó, count(X+Y) là số tổng số lần xuất hiện cả X và Y, count(X) là số lần giao dịch chứa X.

- Để tìm ra luật kết hợp, ta thường áp dụng 2 tiêu chí: minimum support (min_sup) và minimum confidence (min_conf). Các luật thỏa mãn có support và confidence thỏa mãn (lớn hơn hoặc bằng) cả Minimum support và Minimum confidence gọi là các luật mạnh. Minimum support và Minimum confidence gọi là các giá trị ngưỡng và phải xác định trước khi sinh các luật kết hợp.

- Một itemsets mà tần suất xuất hiện của nó $\geq \text{min_sup}$ gọi là frequent itemsets

2.5.2 Một số luật kết hợp

- Luật kết hợp nhị phân: táo \Rightarrow chuối

- Luật kết hợp định lượng: Cân nặng [70kg – 90kg] \Rightarrow chiều cao [170cm – 190cm]

- Luật kết hợp mờ: cân nặng lớn \Rightarrow rất cao.

Thuật toán phổ biến nhất tìm các luật kết hợp là luật kết hợp nhị phân

2.5.3 Xây dựng thuật toán Apriori

Ý tưởng của thuật toán:

Tìm tất cả frequent itemsets

K-itemset (itemsets gồm k items) được dùng để tìm (k+1)- itemset.

Đầu tiên tìm 1-itemset (ký hiệu L1). L1 được dùng để tìm L2 (2-itemsets). L2 được dùng để tìm L3 (3-itemset) và tiếp tục cho đến khi không có k-itemset được tìm thấy.

Từ frequent itemsets sinh ra các luật kết hợp thỏa mãn 2 tham số min_sup và min_conf .

Các bước cụ thể:

Bước 1: Quét toàn bộ dữ liệu để có được support của 1-itemset, so sánh support với min_sup để có được L_1

Bước 2: Sử dụng L_{k-1} nối L_{k-1} để sinh ra tập ứng viên C_k là k-itemset. Loại bỏ các itemsets không phải là frequent itemsets thu được k-itemset.

Bước 3: quét dữ liệu để có được support của mỗi tập ứng viên C_k k-itemset, so sánh support với min_sup để thu được frequent k-itemset (L_k).

Bước 4: Lặp lại từ bước 2 cho đến khi tập frequent n-itemset (L_n) trống

Bước 5: Với mỗi frequent itemset I, sinh tất cả các tập con s không rỗng của I

Bước 6: Với mỗi tập con s không rỗng của I, sinh ra các luật $s \Rightarrow (I-s)$ nếu độ tin cậy (Confidence) của nó $\geq \text{min_conf}$.

Chẳng hạn với $I = \{A1, A2, A5\}$, các tập con của I: $\{A1\}, \{A2\}, \{A5\}, \{A1, A2\}, \{A1, A5\}, \{A2, A5\}$. sẽ có các luật sau:

+ $\{A1\} \Rightarrow \{A2, A5\}$

+ $\{A2\} \Rightarrow \{A1, A5\}$

+ $\{A5\} \Rightarrow \{A1, A2\}$

+ $\{A1, A2\} \Rightarrow \{A5\}$


+ $\{A1, A5\} \Rightarrow \{A2\}$


+ $\{A2, A5\} \Rightarrow \{A1\}$

Ví dụ: Giả sử có cơ sở dữ liệu giao dịch bán hàng gồm 5 giao dịch như sau:

Tid	List of Items
1	Beer,Diaper,Baby Powder,Bread,Umbrella
2	Diaper,Baby Powder
3	Beer,Diaper,Milk
4	Diaper,Beer,Detergent
5	Beer,Milk,Coca-Cola

Thuật toán Apriori tìm các luật kết hợp trong giao dịch bán hàng trên như sau:

bước 1 min-sup =40% (2/5)					
C ₁				L ₁	
items	sets	support		items	support
Beer		4/5		Beer	4/5
Diaper		4/5		Diaper	4/5
Baby Powder		2/5		Baby Powder	2/5
Bread		1/5		Milk	2/5
Umbrella		1/5			
Milk		2/5			
Detergent		1/5			
Coca-Cola		1/5			

Bước 2 và 3					
C ₂				L ₂	
items		support		items	support
Beer,Diaper		3/5		Beer,Diaper	3/5
Beer, Baby Powder		0		Beer, Milk	2/5
Beer, Milk		2/5		Diaper, Baby Powder	2/5
Diaper, Baby Powder		2/5			
Diaper, Milk		0			
Baby Powder, Milk		0			

Bước 4		C_3	min-sup = 40%	
items		support		
Beer, Diaper, Milk		1/5		
Beer, Diaper, Baby Powder		1/5		
Diaper, Milk, Baby Powder		0		
Beer, Milk, Baby Powder		0		

→ $L_3 = \text{Empty (Stop)}$

Bước 5			
min_sup = 40% min_conf = 70%			
itemsets	Support (A,B)	Support(A)	Confidence
Beer, Diaper	60%	80%	75.00%
Diaper, Beer	60%	80%	75.00%
Beer, Milk	40%	80%	50.00%
Milk, Beer	40%	40%	100.00%
Diaper, Baby Powder	40%	80%	50.00%
Baby Powder, Diaper	40%	40%	100.00%

Bước 6: Ta có các luật kết hợp sau với min_sup = 40%, min_conf = 70%

R1: Beer => Diaper (support = 60%, confidence = 75%)

R2: Diaper => Beer (support = 60%, confidence = 75%)

R3: Milk => Beer (support = 40%, confidence = 100%)

R4: Baby Powder => Diaper (support = 40%, confidence = 100%)

2.6. Cơ sở lý thuyết của giải thuật Eclat

Thuật toán ECLAT là viết tắt của cụm từ Equivalence Class Clustering và bottom-up Lattice Traversal. Đây là một trong những phương pháp phổ biến của Association Rule Mining. Đây là một phiên bản hiệu quả hơn và có thể mở rộng của thuật toán Apriori.

Trong khi thuật toán Apriori hoạt động theo chiều ngang Breadth-First Search của một biểu đồ, thì thuật toán ECLAT hoạt động theo chiều dọc giống như Tìm kiếm theo Depth-First Search của một biểu đồ. Cách tiếp cận theo chiều dọc này của thuật toán ECLAT làm cho nó trở thành một thuật toán nhanh hơn so với thuật toán Apriori.

Ý tưởng cơ bản của thuật toán Eclat là sử dụng giao điểm Transaction Id Sets (tidsets) để tính toán giá trị hỗ trợ của một ứng cử viên và tránh tạo ra các tập hợp con không tồn

tại trong cây tiền tố. Trong lần gọi đầu tiên của hàm, tất cả các mục đơn lẻ được sử dụng cùng với các bộ tidset của chúng. Sau đó, hàm được gọi đệ quy và trong mỗi lần gọi đệ quy, mỗi cặp item-tidset được xác minh và kết hợp với các cặp item-tidset khác. Quá trình này được tiếp tục cho đến khi không có cặp item-tidset ứng cử nào có thể được kết hợp.

Bây giờ chúng ta hãy hiểu cách hoạt động đã nêu ở trên với một ví dụ:

Xét bảng dữ liệu sau:

Transaction Id	Bread	Butter	Milk	Coke	Jam
T1	1	1	0	0	1
T2	0	1	0	1	0
T3	0	1	1	0	0
T4	1	1	0	1	0
T5	1	0	1	0	0
T6	0	1	1	0	0
T7	1	0	1	0	0
T8	1	1	1	0	1
T9	1	1	1	0	0

Dữ liệu đã cho ở trên là một ma trận boolean trong đó đối với mỗi ô (i, j), giá trị biểu thị liệu mục thứ j có được đưa vào giao dịch thứ i hay không. 1 có nghĩa là đúng trong khi 0 có nghĩa là sai. Nói cách dễ hiểu ở các id giao dịch là các sản phẩm được mua với giá trị là 1 còn các sản phẩm có giá trị 0 là không được mua.

Bây giờ chúng ta gọi hàm lần đầu tiên và sắp xếp từng mục với bộ tidset của nó theo kiểu bảng:

k = 1, minimum support = 2

Item	Tidset
Bread	{T1, T4, T5, T7, T8, T9}
Butter	{T1, T2, T3, T4, T6, T8, T9}
Milk	{T3, T5, T6, T7, T8, T9}
Coke	{T2, T4}
Jam	{T1, T8}

Bây giờ chúng ta gọi hàm đệ quy cho đến khi không thể kết hợp thêm cặp item-tidset nào nữa:

$$k = 2$$

Item	Tidset
{Bread, Butter}	{T1, T4, T8, T9}
{Bread, Milk}	{T5, T7, T8, T9}
{Bread, Coke}	{T4}
{Bread, Jam}	{T1, T8}
{Butter, Milk}	{T3, T6, T8, T9}
{Butter, Coke}	{T2, T4}
{Butter, Jam}	{T1, T8}
{Milk, Jam}	{T8}

$$k = 3$$

Item	Tidset
{Bread, Butter, Milk}	{T8, T9}
{Bread, Butter, Jam}	{T1, T8}

$$k = 4$$

Item	Tidset
{Bread, Butter, Milk, Jam}	{T8}

2.7. Cơ sở lý thuyết của giải thuật FP-growth

Để hiểu giải thuật FP-growth một cách trực quan, chúng ta cùng xem qua ví dụ sau đây, xét một bảng dữ liệu về các giao dịch (transactions):

Transaction ID	Items
T1	{ <u>E</u> ,K,M,N,O,Y}
T2	{D, <u>E</u> ,K,N,O,Y}
T3	{A, <u>E</u> ,K,M}
T4	{ <u>C</u> ,K,M,U,Y}
T5	{ <u>C</u> , <u>E</u> ,I,K,O,O}

Dữ liệu đã cho ở trên là tập dữ liệu giả định về các giao dịch với mỗi chữ cái đại diện cho một mặt hàng. Tần suất của từng mục riêng lẻ được tính như bảng dưới đây:

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

Giả sử chúng ta chọn mức độ ủng hộ (mức hỗ trợ) tối thiểu là 3. Bây giờ chúng ta sẽ tạo một tập hợp L chứa các phần tử có tần số lớn hơn hoặc bằng mức độ ủng hộ, các phần tử này được xếp theo thứ tự tần số giảm dần. Khi hoàn thành chúng ta sẽ có tập hợp như sau:

$$L = \{K : 5, E : 4, M : 3, O : 3, Y : 3\}$$

Tiếp theo chúng ta sẽ lọc bảng chứa các giao dịch, chỉ lấy các mặt hàng được liệt kê trong tập hơn L, và chúng được sắp xếp theo thứ tự tần suất mua giảm dần. Khi hoàn thành bước này, chúng ta sẽ có bảng sau, với Ordered-Item Set là tập hợp gồm những sản phẩm có mức ủng hộ từ 3 trở lên trong từng đợt giao dịch.

Transaction ID	Items	Ordered-Item Set
T1	{ <u>E</u> ,K,M,N,O,Y}	{ <u>K</u> , <u>E</u> ,M,O,Y}
T2	{ <u>D</u> , <u>E</u> ,K,N,O,Y}	{ <u>K</u> , <u>E</u> ,O,Y}
T3	{ <u>A</u> , <u>E</u> ,K,M}	{ <u>K</u> , <u>E</u> ,M}
T4	{ <u>C</u> , <u>K</u> ,M,U,Y}	{ <u>K</u> , <u>M</u> ,Y}
T5	{ <u>C</u> , <u>E</u> ,I,K,O,O}	{ <u>K</u> , <u>E</u> ,O}

Để hoàn thành mô hình FP-growth chúng ta xây dựng FP-tree bởi vì FP-growth biểu diễn dữ liệu dưới dạng một cấu trúc có tên là FP-tree.

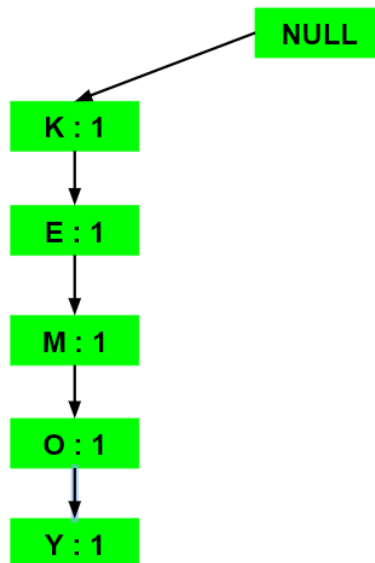
FP-Growth sử dụng FP-Tree để xác định trực tiếp các tập hạng mục phổ biến (không sinh các tập hạng mục ứng viên từ các tập hạng mục ứng viên trước).

Khi một FP-Tree đã được xây dựng, FP-Growth sử dụng cách tiếp cận chia để trị đệ quy để khai thác các tập phổ biến.

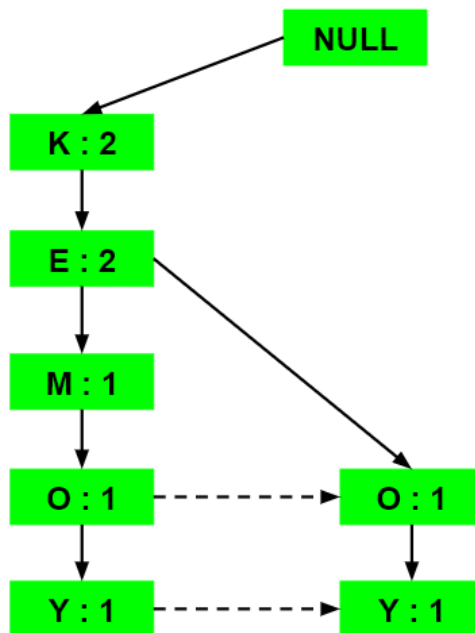
Với mỗi giao dịch, FP-Tree xây dựng một đường đi (path) trong cây.

Hai giao dịch có chứa cùng một số các mục, thì đường đi của chúng sẽ có phần (đoạn) chung. Càng nhiều các đường đi có phần tử chung, thì việc biểu diễn bằng FP-Tree sẽ càng gọn.

Xét giao dịch T1, chúng ta sẽ có tập các sản phẩm sau {K, E, M, O, Y} và biểu diễn chúng dưới dạng tree với giá trị ủng hộ (số lần mua) là 1:

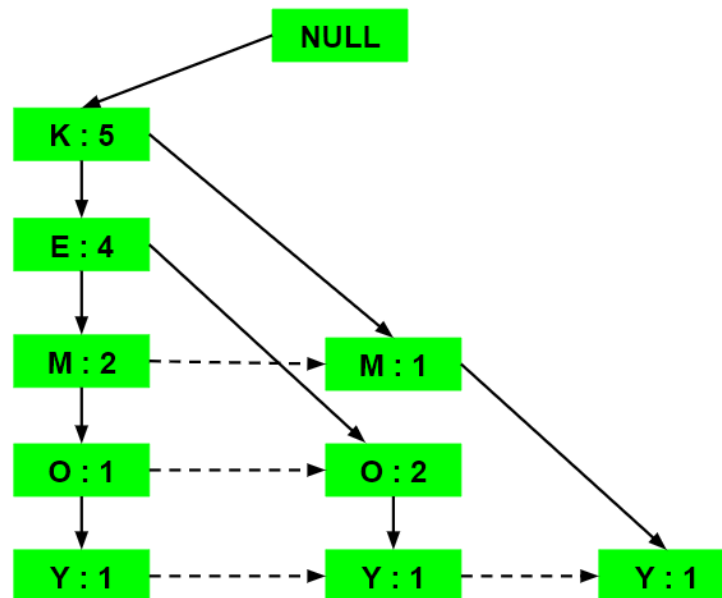


Sau đó chúng ta xét giao dịch T2, và tương tự như bước ở trên, chúng ta sẽ có được tập hợp các sản phẩm {K, E, O, Y} và biểu diễn tập hợp này lên tree ở phía trên. Lưu ý, như đã đề cập, với các giao dịch có một số các mục chung, các mục đó sẽ có chung đường đi (path), hình dưới đây sẽ minh họa điều đó:



Số hỗ trợ (trị số ủng hộ) tăng lên khi sản phẩm được chọn mua một lần, nếu sản phẩm chưa có trong tree, chúng ta sẽ tạo đường đi (path) mới và ô (node) sản phẩm mới.

Lần lượt chúng ta sẽ thêm các giao dịch T3, T4, T5 và có được mô hình sau:



Bây giờ chúng ta có thể tìm ra được các đường để dẫn đến từng ô sản phẩm mong muốn. Liệt kê chúng vào một bảng và chúng ta được:

Items	Conditional Pattern Base
Y	{{ <u>K</u> ,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{ <u>K</u> ,E,M : 1}, {K,E : 2}}
M	{{ <u>K</u> ,E : 2}, {K : 1}}
E	{ <u>K</u> : 4}
K	

Bước tiếp theo trích ra các sản phẩm phổ biến trong tất cả các đường dẫn trong cột Conditional Pattern Base.

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{ <u>K</u> ,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{ <u>K</u> : 3}
O	{{ <u>K</u> ,E,M : 1}, {K,E : 2}}	{K, <u>E</u> : 3}
M	{{ <u>K</u> ,E : 2}, {K : 1}}	{ <u>K</u> : 3}
E	{ <u>K</u> : 4}	{ <u>K</u> : 4}
K		

Từ bảng trên chúng ta có thể lập ra được một bảng với các tập hợp gồm các sản phẩm thường được mua cùng với nhau:

Items	Frequent Pattern Generated
Y	{< <u>K</u> ,Y : 3>}
O	{< <u>K</u> ,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{< <u>K</u> ,M : 3>}
E	{< <u>E</u> ,K : 3>}
K	

CHƯƠNG 3: DỮ LIỆU

3.1 Mô tả tập dữ liệu

Có rất nhiều cách để mô tả dữ liệu, nhưng hầu hết các phương pháp đều tập trung vào hai khía cạnh, đó là số lượng và chất lượng của dữ liệu. Dưới đây là ba đặc tính khi nói về mô tả dữ liệu:

- Kích thước dữ liệu (Amount of data): ví dụ, tại một cửa hàng bán lẻ chuyên bán các loại nước ngọt, dữ liệu bán hàng sau một tháng thu được gồm 500 khách hàng riêng biệt đã tới mua, và tổng số lượt mua là 650 với nhiều loại đồ uống khác nhau. Tùy vào cách trình bày dữ liệu mà ta có thể có được một cơ sở dữ liệu có cấu trúc với kích thước là hàng và cột. Kích thước dữ liệu càng lớn thì kết quả đạt được của dự án càng tốt, nhưng điều đó đòi hỏi thời gian thực thi tìm tri thức trong dữ liệu càng dài.

- Kiểu của giá trị dữ liệu (Value types): có rất nhiều định dạng cho kiểu giá trị dữ liệu, như số, chuỗi, Boolean (đúng hoặc sai hay 0 và 1). Chúng ta cần xác định đúng kiểu dữ liệu trong cơ sở dữ liệu để có thể thực hiện các bước tiền xử lý dữ liệu tốt và cho ra kết quả tốt.

- Mã hóa (coding schemes or encoding): mục đích nhằm chuyển đổi các biến đã được phân loại thành một mẫu, định dạng khác để cung cấp cho các thuật toán, giúp thuật toán có thể thực hiện công việc tốt hơn.

3.2 Tiêu chuẩn đánh giá dữ liệu

Có rất nhiều tiêu chuẩn đánh giá chất lượng của dữ liệu, tùy vào những trường hợp cụ thể và mong muốn đầu ra của dữ liệu mà chúng ta có thể áp dụng linh hoạt các tiêu chuẩn đó.

Định nghĩa chất lượng của data thì có rất nhiều nhưng chúng ta có thể hiểu đơn giản là fitness for use (phù hợp để dùng), phù hợp để dùng trong trường hợp mà chúng ta muốn, thỏa mãn đủ các điều kiện mà chúng ta yêu cầu. Tuy có rất nhiều tiêu chuẩn khác nhau, chung quy lại dữ liệu cần thỏa mãn các điều kiện sau đây để có được một lượng tốt:

- Không có lỗi (free of error): việc một tập dữ liệu lấy từ thực tế không có lỗi là điều bất khả thi, nhưng nhiệm vụ của chúng ta là kiểm tra và chỉnh sửa tập dữ liệu đó để loại bỏ các lỗi. Một số lỗi nghiêm trọng ảnh hưởng rất lớn đến chất lượng của dữ liệu như giá trị ngoài vùng đo (outlier), ví dụ như chúng ta đo nhiệt độ phòng liên tục, nhiệt độ phòng vào khoảng 33 độ C, nhưng giá trị trả về lại có một số dữ liệu có giá trị nhỏ hơn 20 độ C, các giá trị đó chính là giá trị ngoài vùng đo (outlier). Ngoài ra còn các lỗi nghiêm trọng khác mà chúng cần xem xét để loại bỏ nó.

- Chính xác (accurate): độ chính xác của tập dữ liệu đề cập đến việc liệu các giá trị lưu trữ về một đối tượng nào đó có chính xác hay không.

- Hoàn chỉnh (complete): để đánh giá sự hoàn chỉnh của tập dữ liệu khá khó bởi nó bao quát nhiều đặc tính khác nhau của tập dữ liệu đó, tùy vào từng trường hợp cụ thể mà ta có thể đánh giá một tập dữ liệu đã hoàn chỉnh hay chưa. Ví dụ, một cơ sở dữ liệu về khách hàng tại một cửa hàng thú cưng, những thông tin cần thiết là thú cưng mà khách hàng nuôi và những thức và đồ dùng cho thú cưng mà khách hàng thường mua, những thông tin khác như số người hiện tại sống chung với khách hàng đó là những thông tin không cần thiết.

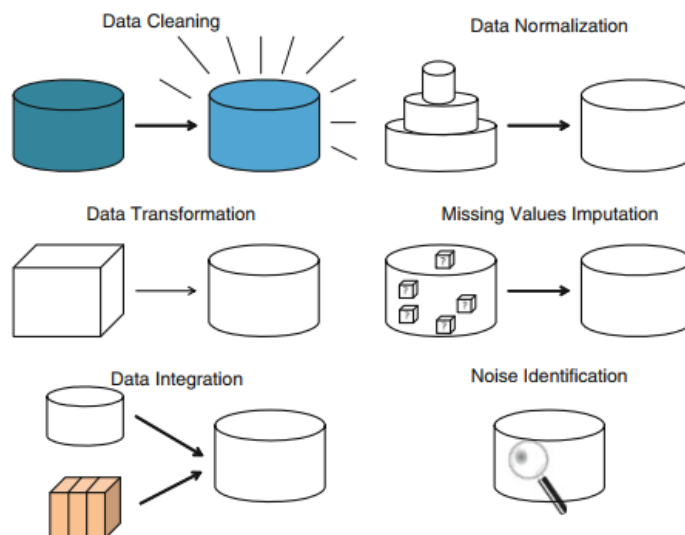
- Liên quan, thích hợp (relevant): khi thu thập dữ liệu, chúng ta cần phải cân nhắc xem dữ liệu đó có liên quan tới mục đích của chúng ta hay không. Ví dụ, đối với chủ của một cửa hàng bán thú cưng thì quan tâm về việc khách hàng thường dùng bữa sáng ở đâu là việc cực kỳ vô lý, chủ cửa hàng chỉ cần quan tâm một số thông tin quan trọng liên quan đến khách hàng và thú cưng khách hàng.

- Có thể tiếp cận được (accessible): dữ liệu có thể dễ dàng truy cập và đọc hiểu là một điều khá quan trọng trong việc nâng cấp chất lượng dữ liệu. Nhiệm vụ của chúng ta là loại bỏ rào cản giữa dữ liệu và người dùng, giúp người dùng có cái nhìn tổng quan và dễ hiểu về dữ liệu.

3.3 Quá trình tiền xử lý dữ liệu

Dữ liệu được lấy từ thực tế không hoàn hảo và còn nhiều lỗi, chính vì thế mà tiền xử lý dữ liệu là một trong những bước quan trọng của quá trình khai phá dữ liệu.

Trong mục 3.3 này sẽ là các kỹ thuật được dùng trong tiền xử lý dữ liệu.



Hình 4. Quá trình tiền xử lý dữ liệu

3.3.1 Chuẩn bị dữ liệu

3.3.1.1 Làm sạch dữ liệu

Đây là giai đoạn xử lý các dữ liệu xấu, không chính xác có trong tập dữ liệu. Hoặc loại bỏ những chi tiết không cần thiết trong dữ liệu. Quá trình làm sạch dữ liệu là một khái niệm tổng quát, và nó bao hàm nhiều kỹ thuật khác trong tiến trình tiền xử lý dữ liệu. Cụ thể hơn, việc xử lý dữ liệu bị thiếu hay xử lý nhiễu cũng sẽ nằm trong quá trình làm sạch dữ liệu. Tuy nhiên, chúng ta sẽ phân nó thành từng mục khác nhau để tìm hiểu kỹ hơn các kỹ thuật này.

3.3.1.1 Chuyển hóa dữ liệu

Tới bước này, dữ liệu được chuyển đổi dữ liệu để dễ dàng hoặc hiệu quả hơn trong việc áp dụng dữ liệu vào các quá trình sau đó. Nhiệm vụ chính của quá trình chuyển hóa dữ liệu là tổng hợp hoặc tóm tắt dữ liệu, làm mịn, xây dựng các đặc tính và tổng quát hóa dữ liệu. Tuy một vài nhiệm vụ ở quá trình này có thể thực hiện bằng các công cụ máy móc, nhưng một số tác vụ khác vẫn cần con người giám sát như tạo báo cáo về dữ liệu, các tác vụ phân loại.

3.3.1.2 Tích hợp dữ liệu

Mục đích của bước tích hợp dữ liệu là tổng hợp, hợp nhất các dữ liệu từ nhiều cơ sở dữ liệu thành một dữ liệu tổng thể. Tuy vậy, việc này cần thực hiện chính xác và cẩn thận để tránh dư thừa dữ liệu và xuất hiện sự không nhất quán trong tập dữ liệu. Các nhiệm vụ điển hình được thực hiện trong tích hợp dữ liệu là xác định và thống nhất các biến và

miền, phân tích mối tương quan thuộc tính, sao chép các bộ giá trị và phát hiện xung đột trong các giá trị dữ liệu của các nguồn khác nhau.

3.3.1.3 Chuẩn hóa dữ liệu

Các đơn vị được sử dụng trong dữ liệu có thể ảnh hưởng đến việc phân tích dữ liệu. Tất cả các giá trị, thuộc tính phải sử dụng cùng một thang đo hoặc có quy ước chung. Nhiệm vụ của chúng ta khi thực hiện bước này là cố gắng cung cấp các giá trị sao cho có trọng số bằng nhau, như đã nói ở trên, dễ hiểu hơn là các giá trị phải có cùng thang đo.

3.3.1.4 Xử lý dữ liệu bị thiếu

Chúng ta có thể sử dụng nhiều cách để xử lý dữ liệu bị thiếu, có thể kể phương pháp trung bình cộng các giá trị có trong tập dữ liệu mà cùng thuộc tính với dữ liệu bị thiếu.

3.3.1.5 Xác định nhiễu

Xác định nhiễu được coi là một bước trong quá trình làm sạch dữ liệu hay còn được biết tới là bước làm mịn trong quá trình chuyển hóa dữ liệu. Mục tiêu chính là phát hiện các giá trị chứa nhiễu trong quá trình đo lường hoặc thu thập dữ liệu. Lưu ý, chúng ta chỉ đang nói tới việc phát hiện dữ liệu bị nhiễu chứ chưa xử lý chúng. Quá trình tối giản dữ liệu ở mục 3.3.2 sẽ đi sâu hơn vào việc xử lý nhiễu.

3.3.2 Tối giản dữ liệu

Trong quá trình tối giản dữ liệu, chúng ta sẽ sử dụng nhiều kỹ thuật khác nhau với mục đích cuối cùng là làm giảm dữ liệu ban đầu. Một mô hình khai phá dữ liệu sẽ không chạy hoặc chắc chắn sẽ cho kết quả sai nếu việc chuẩn bị dữ liệu không được tiến hành đúng cách. Dữ liệu khi trải qua tiến trình tối giản, dữ liệu thường được duy trì cấu trúc thiết yếu và tính toàn vẹn của dữ liệu ban đầu, chỉ khác là dữ liệu được giảm bớt. Nhiệm vụ chính của việc chuẩn bị dữ liệu cho mô hình khai phá dữ liệu là áp dụng dữ liệu hiệu quả vào mô hình và đạt được kết quả chính xác. Do đó, nếu chúng ta cho rằng mình đã thực hiện tốt các bước trong tiến trình tiền xử lý dữ liệu ở trên, chúng ta thường sẽ đặt câu hỏi “Có thể sử dụng dữ liệu gốc mà không cần bước tối giản dữ liệu cho mô hình khai phá dữ liệu hay không?”. Câu trả lời là có thể, tuy nhiên, khi đó sẽ xuất hiện các vấn đề quan trọng khác mà chúng ta cần giải quyết.

Trong 4 phần dưới đây sẽ giải thích ngắn gọn về 4 kỹ thuật được sử dụng trong quá trình tối giản dữ liệu.

3.3.2.1 Chọn thuộc tính

Chọn thuộc tính (Feature Selection, Feature Extraction) là nhiệm vụ rất quan trọng giai đoạn tiền xử lý dữ liệu khi triển khai các mô hình khai phá dữ liệu. Một vấn đề gặp phải là các dataset dùng để xây dựng các Data mining Models thường chứa nhiều thông tin

không cần thiết (thậm chí gây nhiễu) cho việc xây dựng mô hình. Chẵn hạn, một dataset gồm hàng trăm thuộc tính dùng để mô tả về khách hàng của một doanh nghiệp được thu thập, tuy nhiên khi xây dựng một Data mining model nào đó chỉ cần khoảng 50 thuộc tính từ hàng trăm thuộc tính đó. Nếu ta sử dụng tất cả các thuộc tính (hàng trăm) của khách hàng để xây dựng mô hình thì ta cần nhiều CPU, nhiều bộ nhớ trong quá trình Training model, thậm chí các thuộc tính không cần thiết đó làm giảm độ chính xác của mô hình và gây khó khăn trong việc phát hiện tri thức.

3.3.2.2 Chọn mẫu

Chọn các một lượng các thuộc tính, giá trị cần thiết bên trong dữ liệu ban đầu với yêu cầu là các dữ liệu được trích xuất này có thể đại diện cho các đặc tính của dữ liệu ban đầu. Các phương pháp lựa chọn mẫu được áp dụng để giảm tập dữ liệu ban đầu xuống kích thước có thể quản lý được, dẫn đến giảm tài nguyên tính toán cần thiết để thực hiện quá trình học tập. Các thuật toán lựa chọn cá thể cũng có thể được áp dụng để loại bỏ các cá thể nhiễu, trước khi áp dụng các thuật toán học. Bước này có thể cải thiện độ chính xác trong các bài toán phân loại.

3.3.2.3 Rời rạc hóa dữ liệu

Bước này biến đổi dữ liệu định lượng thành dữ liệu định tính, tức là các thuộc tính số thành các thuộc tính rời rạc hoặc danh nghĩa với một số khoảng hữu hạn, thu được một phân vùng không chồng chéo của một miền liên tục.

3.3.2.4 Mở rộng thuộc tính / Tạo mẫu

Chúng ta có thể tự tạo hay điều chỉnh các mẫu, giá trị bên trong cơ sở dữ liệu để máy có thể làm việc tốt hơn trong điều kiện học có giám sát.

CHƯƠNG 4: GIẢI THUẬT VÀ CHƯƠNG TRÌNH

4.1 Giải thuật Apriori

```
import numpy as np

import pandas as pd

from apyori import apriori

# thêm các thư viện cần thiết cho training bao gồm numpy, pandas và apyori để
sử dụng thuật toán apriori.

data = pd.read_csv('database.csv', header=None)
```

đọc dữ liệu từ file databasse.csv với không có tiêu đề các cột. dữ liệu được thu thập trong vòng 1 tuần với 7501 lần giao dịch

```
training= []
```

```
for i in range(0, 7501):
```

```
    training.append([str(data.values[i,j]) for j in range(0, 20)])
```

đọc từng hàng dữ liệu và lưu vào trong mảng record

```
rules = apriori(training, min_support=0.0056, min_confidence=0.5, min_lift=3, min_length=3)
```

```
results = list(rules)
```

giả sử mong muốn mặt hàng được mua ít nhất 6 lần 1 ngày thì độ support là $6 \times 7 / 7501$. Áp dụng thuật toán apriori với min_support là 0.0056, min_confidence là 0.5, khả năng mua cùng nhau là gấp 3 lần so với mua lẻ, số phần tử trong tập phổ biến cuối cùng là 3. Sau đó đưa các luật vào 1 list.

```
results # in ra tất cả kết quả.
```

4.2 Giải thuật Eclat

Đầu tiên, chúng ta import các thư viện cần thiết vào. Trong đó thư viện Mlxtend là thư viện Python với các công cụ thú vị cho các nhiệm vụ khoa học dữ liệu, thư viện này tạo bản ghi phản thực tế, vẽ biểu đồ tương quan PCA và ranh giới quyết định, thực hiện phân

```
Entrée [1]: import numpy as np  
import pandas as pd
```

```
Entrée [ ]: #pip install mlxtend  
#pip install --no-binary :all: mlxtend
```

tách phương sai lệch, khởi động chuỗi, và hơn thế nữa... Thư viện numpy là thư viện lõi phục vụ tính toán các mảng nhiều chiều kích thước lớn. Thư viện pandas cung cấp cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt.

Tiếp theo chúng tạo các mảng 2 chiều:

```
Entrée [2]: transactions = [[1, 2, 5],  
                             [2, 4],  
                             [2, 3],  
                             [1, 2, 4],  
                             [1, 3],  
                             [2, 3],  
                             [1, 3],  
                             [1, 2, 3, 5],  
                             [1, 2, 3]]
```

Sau đó đưa tập dữ liệu được định dạng qua tập .csv lên:

Tiếp theo chúng ta xử lý trước dữ liệu để phù hợp với mô hình:

Ta dùng hàm for để chuyển đổi dữ liệu thành các list.

```
Entrée [5]: transactions = []
            for sublist in dataset.values.tolist():
                clean_sublist = [item for item in sublist if item is not np.nan]
                transactions.append(clean_sublist)
```

Gọi hàm mã hóa dữ liệu giao dịch. Thông qua việc sử dụng TransactionEncoder chúng ta có thể chuyển đổi giao dịch này thành một định dạng mảng phù hợp với các API học

```
Entrée [6]: from mlxtend.preprocessing import TransactionEncoder
            te = TransactionEncoder()
            te_ary = te.fit(transactions).transform(transactions)
            df_x = pd.DataFrame(te_ary, columns=te.columns_) # encode to onehot
```

máy điển hình. Thông qua fit, transactionencoder học các nhãn duy nhất trong transactions qua phương thức transform, nó biến đổi các giao dịch đầu vào thành một mảng boolean được mã hóa (encode to onehot).

Tiến hành import apriori và association_rules.

Sau đó thiết lập các giá trị qua hàm apriori và association vào sets và rules. Apriori trả về chỉ số cột của các mục, đặt use_colnames=True chuyển đổi các giá trị số nguyên này thành các tên mục tương ứng. Hàm kế cho phép bạn xác định số liệu mà bạn quan tâm và ngưỡng theo. Ở đây, quy tắc là support cao hơn ngưỡng 0.5%. Trong các trường hợp này,

```
Entrée [7]: from mlxtend.frequent_patterns import apriori
            from mlxtend.frequent_patterns import association_rules
            df_sets = apriori(df_x, min_support=0.005, use_colnames=True)
            df_rules = association_rules(df_sets, metric='support', min_threshold= 0.005, support_only=True)
```

không phải tất cả các chỉ số đều có thể được tính toán, do DataFrames đầu vào không đầy đủ, bạn có thể sử dụng support_only=True, tùy chọn này sẽ chỉ tính toán cột hỗ trợ của một quy tắc nhất định mà không yêu cầu nhiều thông tin.

Xuất te_ary:

Chúng ta vừa hoàn thành bước tiền xử lý dữ liệu.

```
Entrée [8]: print(te_ary)

[[False True True ... True False False]
 [False False False ... False False False]
 [False False False ... False False False]
 ...
 [False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False True False]]
```

Xuất khung dữ liệu được chuyển đổi:

Entrée [9]: `print(df_x)`

```

      asparagus  almonds  antioxydant juice  asparagus  avocado  babies food \
0      False      True      True      False      False      True      False
1      False      False      False      False      False      False      False
2      False      False      False      False      False      False      False
3      False      False      False      False      False      True      False
4      False      False      False      False      False      False      False
...      ...      ...      ...      ...      ...      ...      ...
7496     False     False      False      False      False      False      False
7497     False     False      False      False      False      False      False
7498     False     False      False      False      False      False      False
7499     False     False      False      False      False      False      False
7500     False     False      False      False      False      False      False

      bacon  barbecue  sauce  black tea  blueberries  ...  turkey \
0      False      False      False      False      False  ...  False
1      False      False      False      False      False  ...  False
2      False      False      False      False      False  ...  False
3      False      False      False      False      False  ...  True
4      False      False      False      False      False  ...  False
...      ...      ...      ...      ...      ...  ...  ...
7496     False      False      False      False      False  ...  False
7497     False      False      False      False      False  ...  False
7498     False      False      False      False      False  ...  False
7499     False      False      False      False      False  ...  False
7500     False      False      False      False      False  ...  False

```

Sau đó tính hỗ trợ:

Entrée [15]: `print(df_sets)`

```

      support  itemsets
0      0.020397      (almonds)
1      0.008932  (antioxydant juice)
2      0.033329      (avocado)
3      0.008666      (bacon)
4      0.010799  (barbecue sauce)
..      ...      ...
720  0.007466  (mineral water, spaghetti, soup)
721  0.009332  (spaghetti, mineral water, tomatoes)
722  0.006399  (mineral water, spaghetti, turkey)
723  0.006266  (mineral water, spaghetti, whole wheat rice)
724  0.005066  (olive oil, pancakes, spaghetti)

```

[725 rows x 2 columns]

Bảng quy tắc

Entrée [17]: `print(df_rules)`

	antecedents	consequents	antecedent support \
0	(burgers)	(almonds)	NaN
1	(almonds)	(burgers)	NaN
2	(almonds)	(chocolate)	NaN
3	(chocolate)	(almonds)	NaN
4	(eggs)	(almonds)	NaN
...
1935	(olive oil, spaghetti)	(pancakes)	NaN
1936	(pancakes, spaghetti)	(olive oil)	NaN
1937	(olive oil)	(pancakes, spaghetti)	NaN
1938	(pancakes)	(olive oil, spaghetti)	NaN
1939	(spaghetti)	(olive oil, pancakes)	NaN

	consequent support	support	confidence	lift	leverage	conviction
0	NaN	0.005199	NaN	NaN	NaN	NaN
1	NaN	0.005199	NaN	NaN	NaN	NaN
2	NaN	0.005999	NaN	NaN	NaN	NaN
3	NaN	0.005999	NaN	NaN	NaN	NaN
4	NaN	0.006532	NaN	NaN	NaN	NaN
...
1935	NaN	0.005066	NaN	NaN	NaN	NaN
1936	NaN	0.005066	NaN	NaN	NaN	NaN
1937	NaN	0.005066	NaN	NaN	NaN	NaN
1938	NaN	0.005066	NaN	NaN	NaN	NaN
1939	NaN	0.005066	NaN	NaN	NaN	NaN

[1940 rows x 9 columns]

Vì chúng ta chỉ sử dụng “support” nên phần code này đã dùng ECLAT.

4.3 Giải thuật FP-growth

Đầu tiên chúng ta sẽ thêm các thư viện cần thiết cho việc thực hiện khai phá dữ liệu:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
import numpy as np
from mlxtend import frequent_patterns
```

Đọc dữ liệu bằng `pd.read_csv`, bởi vì dữ liệu được lưu dưới dạng Comma-separated values (dữ liệu được ngăn cách bằng dấu phẩy).

```
data = pd.read_csv('/content/Market_Basket_Optimisation.csv')
```

Tiếp theo chúng ta dùng cấu trúc các lệnh và sử dụng khái niệm List comprehensive để lấy từng tập hợp sản phẩm được mua trong một giao dịch:

```
transactions = []
for sublist in data.values.tolist():
```

```
transfer_list = [item for item in sublist if item is not np.nan]
transactions.append(transfer_list)
transactions
```

Gọi hàm mã hóa dữ liệu giao dịch. Thông qua việc sử dụng TransactionEncoder chúng ta có thể chuyển đổi giao dịch này thành một định dạng mảng phù hợp với các API học máy điển hình. Thông qua fit, transactionencoder học các nhãn duy nhất trong transactions qua phương thức transform, nó biến đổi các giao dịch đầu vào thành một mảng boolean được mã hóa (encode to onehot).

```
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Cuối cùng là dùng thư viện mlxtend, gọi hàm fpgrowth, đặt hệ số ủng hộ tối thiểu là 0.6 và ta sẽ được kết quả gồm các sản phẩm thường được mua chung với nhau trong một giao dịch.

```
frequent_patterns.fpgrowth(df, min_support=0.6)
```

CHƯƠNG 5: KẾT QUẢ VÀ PHÂN TÍCH

5.1 Mô tả cách huấn luyện

Mô hình sẽ được huấn luyện bằng việc quét qua toàn bộ dữ liệu để tìm độ phổ biến của các tập mặt hàng và so sánh với độ phổ biến nhỏ nhất để tìm ra các tập sản phẩm thỏa mãn yêu cầu.

5.2 Phân tích kết quả

Sau khi tìm hiểu và làm việc với các thuật toán trong khai phá dữ liệu, bao gồm Apriori, Eclat và FP-Growth chúng ta có thể rút ra được một số nhận định và điểm mạnh điểm yếu của từng thuật toán.

Nhìn chung, các thuật toán bên trong các thư viện đã làm tốt vai trò của nó là tìm được các tri thức chứa bên trong dữ liệu, các tri thức ở đây chính là các tập phổ biến, xuất hiện nhiều lần và thường đi chung với nhau.

Như đã giới thiệu, Apriori là một trong những thuật toán nền tảng trong data mining, nó là cơ sở cho một số thuật toán khai phá dữ liệu khác, nên chúng ta sẽ xét ưu nhược điểm của Eclat và FP-growth so với Apriori.

Xét về thuật toán Apriori, nó có một số hạn chế:

- Phải quét cơ sở dữ liệu nhiều lần.
- Số lượng tập ứng viên (C_i) rất lớn
- Thực hiện việc tính độ phổ biến nhiều, đơn điệu.
- Khôn thể phát hiện các luật dạng phủ định

Còn về thuật toán Eclat, so với Apriori, Eclat có một số ưu điểm:

- Yêu cầu về bộ nhớ: Vì thuật toán ECLAT sử dụng cách tiếp cận Depth-First Search, nên nó sử dụng ít bộ nhớ hơn so với thuật toán Apriori.
- Tốc độ: Thuật toán ECLAT thường nhanh hơn thuật toán Apriori.
- Số lần tính toán: Thuật toán ECLAT không liên quan đến việc quét dữ liệu lặp lại để tính toán các giá trị hỗ trợ riêng lẻ.

Khi so sánh FP-growth và Apriori dựa trên 6 tiêu chí: kỹ thuật, chiến lược tìm kiếm, việc sử dụng bộ nhớ, số lần quét cơ sở dữ liệu, thời gian thực hiện và hiệu quả các thuật toán trên các bộ dữ liệu khác nhau chúng ta có thể rút ra:

- Kỹ thuật: Thuật toán Apriori sử dụng hai tính chất của Apriori để thực hiện quá trình kết hợp sinh ra các tập ứng viên và loại bỏ các tập ứng viên không phù hợp, xây dựng dần các tập phổ biến từ dưới lên. Thuật toán FP Growth xây dựng cây FP- tree rồi từ đó xây dựng cơ sở mẫu điều kiện (conditional pattern-base) và các FP-tree có điều kiện (condition FP-tree) thỏa mãn độ hỗ trợ tối thiểu minSup.
- Chiến lược tìm kiếm: Apriori sử dụng chiến lược tìm kiếm theo chiều sâu còn FP Growth sử dụng chiến lược chia để trị.
- Sử dụng bộ nhớ: Thuật toán Apriori đòi hỏi không gian bộ nhớ lớn khi xử lý số lượng các tập ứng cử viên candidate itemsets được tạo ra. Thuật toán FP Growth đòi hỏi ít bộ nhớ hơn do cấu trúc cây nhỏ gọn của nó và khai phá các tập phổ biến frequent itemsets mà không phải thông qua quá trình sinh tập ứng cử viên candidate itemsets.
- Số lần quét cơ sở dữ liệu: Thuật toán Apriori thực hiện nhiều lần quét để có thể tạo ra các tập ứng viên candidate itemsets. Thuật toán FP Growth chỉ cần quét cơ sở dữ liệu đúng hai lần.

- Thời gian thực hiện: Trong Apriori, thời gian thực hiện thuật toán bị lãng phí nhiều ở quá trình mỗi lần sinh ra các tập ứng viên. Với đặc thù của chiến lược sử dụng trong FP Growth đã nêu ở trên, nó yêu cầu ít thời gian thực hiện hơn so với giải thuật Apriori.

- Hiệu quả trên các bộ dữ liệu: Apriori làm việc tốt với cơ sở dữ liệu lớn. FP-Growth làm việc tốt với các cơ sở dữ liệu nhỏ và các tập phổ biến sinh ra không quá dài (độ dài lớn nhất của một tập frequent itemset không quá lớn).

TÀI LIỆU THAM KHẢO

Phần dưới đây sẽ liệt kê các website, sách, tài liệu được tham khảo để viết bài báo cáo này:

[1]https://en.wikipedia.org/wiki/Association_rule_learning

[2]https://lib.hpu.edu.vn/bitstream/handle/123456789/18139/71_NguyenNgocChau_C T1002.pdf

[3]https://vi.wikipedia.org/wiki/C%C6%A1_s%E1%BB%9F_d%E1%BB%AF_li%E1%BB%87u#Nh%E1%BB%AFng_n%C4%83m_1960,_DBMS_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng

[4]<https://www.hauai.edu.vn/media/26/ufpdf26980.pdf>

[5]<https://en.wikipedia.org/wiki/Data>

[6]https://en.wikipedia.org/wiki/Thomas_Bayes

[7]<http://www.jaist.ac.jp/~bao/VNAlectures/AssociationAnalysis-Hieu.pdf>

[8]https://vi.wikipedia.org/wiki/Tri_th%E1%BB%A9c#:~:text=Tri%20th%E1%BB%A9c%20hay%20ki%E1%BA%BFn%20th%E1%BB%A9c,l%C3%BD%20thuy%E1%BA%BFt%20hay%20th%E1%BB%B1c%20h%C3%A0nh.

[9]https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/dstopic.html

<https://www.kdnuggets.com/2016/06/rayli-history-data-mining.html>

- [10]<https://www.kdnuggets.com/2016/06/rayli-history-data-mining.html>
- [11]https://en.wikipedia.org/wiki/Association_rule_learning#:~:text=Association%20rule%20learning%20is%20a,using%20some%20measures%20of%20interestingness.
- [12]https://repository.vnu.edu.vn/bitstream/VNU_123/6401/1/00050001570.pdf
- [13]https://lib.hpu.edu.vn/bitstream/handle/123456789/18139/71_NguyenNgocChauCT1002.pdf
- [14]<https://nhannguyen95.github.io/bai-tap-khai-pha-tap-pho-bien-bang-thuat-toan-apriori/>
- [15]<https://fsppm.fulbright.edu.vn/cache/MPP05-521-R01&02V-2012-10-05-15541737.pdf>
- [16]<https://www.ibm.com/docs/en/spss-modeler/SaaS?topic=understanding-describing-data>
- [17]<http://bis.net.vn/forums/p/505/942.aspx>
- [18]https://en.wikipedia.org/wiki/Instance_selection
- [19]<https://nhannguyen95.github.io/bai-tap-khai-pha-tap-pho-bien-bang-thuat-toan-fp-growth/>
- [20]<https://www.geeksforgeeks.org/ml-frequent-pattern-growth-algorithm/>
- [21]<http://bis.net.vn/forums/p/389/683.aspx>
- [22]Principles of Data Quality by Arthur D. Chapman
- [23]Data Preprocessing in Data mining by Salvador García, Julián Luengo, Francisco Herrera

CODE THAM KHẢO:

Apriori: <https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python>

Eclat: <https://www.i2tutorials.com/machine-learning-tutorial/eclat-algorithm/>

FP-growth: http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/fpgrowth/