BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

ÁP DỤNG THUẬT TOÁN ECLAT VÀO BỘ DỮ LIỆU MARKET BASKET ANALYSIS 4

Học phần: KHAI PHÁ DỮ LIỆU

Chuyên ngành: Khoa học dữ liệu – Khóa 47

Nhóm Sinh Viên:

Họ và tên	MSSV
Trương Thanh Phong	31211027662
Đỗ Quang Thiên Phú	31211024191
Lê Trần Khánh Phú	31211024087
Phạm Minh Phước	31211027663
Dương Mỹ Quỳnh	31211027666

Giảng Viên: TS. Nguyễn An Tế

TP. Hồ Chí Minh, ngày 14 tháng 12 năm 2023

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn đến khoa Công nghệ thông tin kinh doanh – Trường Công nghệ và Thiết kế, Đại học Kinh tế TP. Hồ Chí Minh đã đưa những học phần bổ ích vào chương trình đào tạo để chúng em có cơ hội được tiếp cận với nhiều kiến thức mới lạ, có ý nghĩa cho ngành nghề sau này.

Tiếp theo, chúng em xin bày tỏ lòng biết ơn đến các nguồn học liệu được tham khảo trong đồ án. Những tài liệu này đã cung cấp cho chúng em những thông tin quan trọng, là nền tảng giúp chúng em tiếp cận đề tài dễ dàng hơn và hiểu hơn về chủ đề mà mình. Đồng thời, chúng em xin gửi lời cảm ơn đến tất cả các bạn học đã cùng chúng em chia sẻ, trao đổi và góp ý trong quá trình thực hiện đồ án. Sự giúp đỡ của các bạn đã giúp chúng em hoàn thiện đề tài một cách toàn diện hơn.

Đặc biệt, chúng em muốn gửi lời cảm ơn sâu sắc nhất đến TS. Nguyễn An Tế, giảng viên bộ môn Khai phá dữ liệu. Người đã trực tiếp giảng dạy và hướng dẫn nhóm chúng em trong suốt quá trình thực hiện đồ án. Sự tận tâm, nhiệt tình và kiến thức chuyên môn sâu rộng của thầy đã giúp chúng em hiểu rõ hơn về các khái niệm, phương pháp và các kỹ thuật trong khai phá dữ liệu cũng như các môn học liên quan khác, từ đó có thể hoàn thành đồ án một cách tốt nhất.

Mặc dù chúng em đã cố gắng nhưng chắc chắn đồ án không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được góp ý, đánh giá từ thầy để hoàn thiện đồ án hơn trong tương lai và cải thiện trong những đồ án môn học tiếp theo.

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Từ gốc
ECLAT	Equivalence Class Clustering and Bottom-Up Lattice Traversal
FP-Growth	Frequent Pattern Growth Algorithm

CHƯƠNG 1: MỞ ĐẦU

1. GIỚI THIỆU ĐỀ TÀI

Trong bối cảnh ngày nay, khi dữ liệu ngày càng trở nên phong phú và đa dạng, việc khai phá thông tin ẩn sau những dòng dữ liệu là một thách thức quan trọng. Trong lĩnh vực bán lẻ và thương mại điện tử, phân tích giỏ hàng đã trở thành một công cụ mạnh mẽ để hiểu rõ hành vi mua sắm của khách hàng.

Đề tài này đặt ra mục tiêu nghiên cứu về việc ứng dụng thuật toán ECLAT (Equivalence Class Clustering and Bottom-Up Lattice Traversal) trong phân tích giỏ hàng. ECLAT, một thuật toán quan trọng trong lĩnh vực khai phá dữ liệu, giúp xác định mối liên kết giữa các sản phẩm dựa trên sự xuất hiện chung trong các giao dịch mua sắm.

2. MỤC TIÊU ĐỀ TÀI

Phân tích sâu rộng các giao dịch mua sắm để xác định những mối liên kết mạnh mẽ giữa các sản phẩm. ECLAT sẽ được áp dụng để tìm ra những cặp sản phẩm thường xuất hiện cùng nhau trong giỏ hàng, từ đó xác định các quy luật kết hợp quan trọng. Từ kết quả đó, ta sử dụng để dự đoán hành vi mua sắm của khách hàng. Điều này giúp doanh nghiệp hiểu rõ hơn về sở thích và ưu tiên của khách hàng, từ đó tối ưu hóa chiến lược tiếp thị và cung cấp các sản phẩm tương ứng.

Mục tiêu nghiên cứu này nhằm tạo ra những hướng dẫn cụ thể và chiến lược ứng dụng trong thực tế, giúp doanh nghiệp nắm bắt cơ hội thị trường và cải thiện trải nghiệm mua sắm của khách hàng.

3. PHƯƠNG PHÁP NGHIÊN CỨU

Nhóm chủ yếu dựa trên việc xử lý bộ dữ liệu mua sắm thu thập từ các giao dịch của khách hàng. Đặc biệt, nhóm sẽ áp dụng thuật toán ECLAT để tìm kiếm và phân tích các luật kết hợp giữa các sản phẩm, các quy luật này sẽ được chi tiết phân tích và đánh giá để chúng ta có thể rút ra những thông tin quan trọng về hành vi mua sắm và từ đó phát triển chiến lược kinh doanh hiệu quả. Qua việc khám phá mối liên kết giữa các sản phẩm, nhóm hy vọng sẽ tạo ra những chiến lược thực tế giúp doanh nghiệp tối ưu hóa trải nghiệm mua sắm và tăng cường hiệu suất kinh doanh của họ.

4. KẾT CẤU ĐỀ TÀI

Kết cấu đề tài gồm 5 chương:

- ***Chương 1:*** Trình bày tổng quan đề tài, giới thiệu, mục tiêu, phương pháp nghiên cứu của đề tài.
- ***Chương 2:*** Giới thiệu về Market Basket Analysis, giới thiệu các thuật toán Apriori, ECLAT, FP-Growth. So sánh độ phức tạp của các thuật toán.
- ***Chương 3:*** Trình bày về các bước của thuật toán ECLAT, ý nghĩa các hàm, các công thức, biểu đồ quy trình mô hình, áp dụng thuật toán cho các bài toán đơn giản sau đó đưa ra mô hình.
- ***Chương 4:*** Ứng dụng thuật toán ECLAT vào Market Basket Analysis, đưa ra kết quả và biểu diễn trực quan hóa.
- ***Chương 5:*** Trình bày kết luận, đưa ra các hạn chế và hướng phát triển cho đề tài.

CHƯƠNG 2: MARKET BASKET ANALYSIS

1. GIỚI THIỆU MARKET BASKET ANALYSIS

Market Basket Analysis là một phương pháp quan trọng trong lĩnh vực bán lẻ và tiếp thị. Nó giúp các doanh nghiệp hiểu rõ hơn về hành vi mua hàng của khách hàng và tìm ra những mối liên kết giữa các mặt hàng khác nhau mà khách hàng thường mua cùng nhau. Bằng cách áp dụng Market Basket Analysis, các doanh nghiệp có thể tăng cường doanh số bán hàng và tạo ra doanh thu bằng cách đưa ra các gợi ý mua hàng thông minh và hiệu quả.

Market Basket Analysis hoạt động dựa trên nguyên tắc rằng một khách hàng thường mua những sản phẩm nào cùng một lúc. Bằng cách xác định các mối quan hệ và quy tắc nếu-thì giữa các mặt hàng, Market Basket Analysis xây dựng các hồ sơ mua hàng. Ví dụ, nếu một khách hàng mua bánh mì, có khả năng anh ta cũng sẽ mua bơ, mứt hoặc sữa để kết hợp với bánh mì. Các quy tắc này giúp các doanh nghiệp tìm ra cách kết hợp sản phẩm và dịch vụ để thúc đẩy bán hàng chéo, tăng tỷ lệ chuyển đổi và tăng doanh thu.

Ứng dụng của Market Basket Analysis rất đa dạng. Trong ngành bán lẻ, nó được sử dụng để tối ưu hóa việc đặt sản phẩm và gợi ý mua hàng cho khách hàng. Nó cũng có thể áp dụng trong lĩnh vực chăm sóc sức khỏe để phát hiện các phản ứng phụ của thuốc. Ngoài ra, Market Basket Analysis cũng có thể được sử dụng trong việc phát hiện gian lận và quản lý rủi ro.

Bài toán Market Basket Analysis đóng vai trò quan trọng trong việc tăng cường sự hiểu biết về nhu cầu và hành vi mua hàng của khách hàng. Bằng cách áp dụng các kỹ thuật phân tích dữ liệu thông minh, nó mang lại lợi ích kinh doanh đáng kể cho các doanh nghiệp, giúp tối ưu hóa hoạt động bán hàng và đáp ứng nhu cầu của khách hàng một cách hiệu quả và cá nhân hóa.

2. KHAI PHÁ LUẬT KẾT HỢP

Khai phá luật kết hợp là quá trình phân tích dữ liệu để tìm ra các mẫu tương quan giữa các mục trong tập dữ liệu giao dịch. Giao dịch được đại diện bởi tập hợp các mục, trong đó mỗi mục đại diện cho một danh mục hoặc sản phẩm khác nhau. Các mẫu phổ biến là những mẫu xuất hiện thường xuyên trong các giao dịch.

Có ba loại mẫu phổ biến được quan tâm trong khai phá luật kết hợp. Itemsets là tập hợp các mục mà không quan tâm đến thứ tự của chúng. Subsequences là tập hợp các

mục mà quan tâm đến thứ tự của chúng. Substructures là các loại đồ thị con kết hợp itemsets và subsequences.

Một itemset là một tập con của tất cả các mục có sẵn trong tập hợp danh mục. K itemsets là các itemset có số phần tử bằng k. Các giao dịch được tổ chức thành một cơ sở dữ liệu, trong đó mỗi giao dịch là một tập hợp các mục.

Luật kết hợp X=>Y là một quy tắc mà cả X và Y đều không trống và không giao nhau. Độ hỗ trợ của một itemset được đo bằng số lần xuất hiện của nó trong cơ sở dữ liệu. Độ hỗ trợ được tính bằng tỷ lệ giữa số lần xuất hiện của itemset X và Y cùng với nhau và tổng số giao dịch trong cơ sở dữ liệu.

Độ tin cậy của một luật kết hợp được sử dụng để đo độ chính xác của luật đó. Độ tin cậy được tính bằng tỷ lệ giữa số lần xuất hiện của itemset X và Y cùng với nhau và số lần xuất hiện của itemset X. Độ tin cậy cho biết trong số các lần mà itemset X xuất hiện, có bao nhiêu lần đi kèm với itemset Y.

Tập phổ biến là tập hợp các itemset có độ hỗ trợ lớn hơn hoặc bằng một giá trị ngưỡng được gọi là MinSup. Các itemset trong tập phổ biến đại diện cho các mẫu xuất hiện thường xuyên trong cơ sở dữ liệu.

Cuối cùng, luật kết hợp mạnh là những luật có độ tin cậy lớn hơn hoặc bằng một giá trị ngưỡng được gọi là MinConf. Những luật này biểu thị các mối quan hệ mạnh mẽ giữa các mục trong cơ sở dữ liệu.

3. LỢI ÍCH CỦA SỬ DỤNG LUẬT KẾT HỢP

- **Hỗ trợ doanh nghiệp phát triển chiến lược bán hàng**

Mục tiêu cuối cùng của bất kỳ doanh nghiệp nào là trở nên có lợi nhuận. Điều này đòi hỏi thu hút nhiều khách hàng hơn và tăng doanh số bán hàng. Họ có thể phát triển các chiến lược tốt hơn bằng cách xác định những sản phẩm bán chạy cùng nhau. Ví dụ, biết được rằng những người mua khoai tây chiên hầu như luôn mua Coca-Cola có thể được sử dụng để tăng doanh số bán hàng.

- **Hỗ trợ doanh nghiệp phát triển chiến lược marketing**

Thu hút khách hàng là một yếu tố quan trọng của bất kỳ doanh nghiệp nào. Hiểu được những sản phẩm bán chạy cùng nhau và những sản phẩm không thể thiếu khi phát triển các chiến lược marketing.

Điều này bao gồm kế hoạch bán hàng và quảng cáo, cũng như marketing nhắm mục tiêu. Ví dụ, biết rằng một số đồ trang trí không bán chạy như những món khác trong

mùa lễ có thể giúp người quản lý mở một chiết khấu cho những món trang trí ít bán chạy hơn.

- **Hỗ trợ lập kế hoạch về hạn sử dụng**

Kiến thức về luật kết hợp có thể giúp người quản lý cửa hàng lập kế hoạch về hàng tồn kho và tránh mất tiền do cung cấp quá nhiều hàng hóa có doanh số bán chậm.

Ví dụ, nếu quả oliu không bán chạy, người quản lý sẽ không tích trữ chúng. Tuy nhiên, ông muốn đảm bảo rằng hàng tồn kho hiện tại được bán trước ngày hết hạn. Vì những người mua bột làm bánh pizza cũng mua quả oliu, quả oliu có thể được bán với giá thấp hơn khi mua kèm bột làm bánh pizza.

- **Hỗ trợ tổ chức trong cửa hàng**

Các sản phẩm đã được chứng minh là tăng doanh số bán hàng của các sản phẩm khác có thể được di chuyển gần nhau hơn trong cửa hàng. Ví dụ, nếu doanh số bán bơ tăng nhờ doanh số bán bánh mì, chúng có thể được di chuyển vào cùng một lối đi trong cửa hàng.

Khai phá luật kết hợp cũng được sử dụng trong đề xuất phương tiện truyền thông (phim ảnh, âm nhạc, v.v.), phân tích trang web (người truy cập trang web A có khả năng cao truy cập trang web B), và nhiều lĩnh vực khác.

4. HƯỚNG TIẾP CẬN

4.1. Các thuật toán trong khai phá luật kết hợp

Có nhiều thuật toán được sử dụng để khai phá luật kết hợp. Trong đó có 3 thuật toán phổ biến nhất là Apriori, ECLAT và FP Growth.

- Apriori: Thuật toán này hoạt động bằng cách như sau đầu tiên xác định các tập hợp hàng hóa phổ biến trong tập dữ liệu (các tập hợp hàng hóa xuất hiện trong một số giao dịch nhất định). Sau đó, nó sử dụng các tập hợp hàng hóa phổ biến này để tạo ra các luật kết hợp, đó là các câu lệnh có dạng "nếu mua hàng A, thì rất có thể cũng mua hàng B". Thuật toán Apriori sử dụng một phương pháp từ dưới lên, bắt đầu từ các hàng hóa cá nhân và dần dần xây dựng lên các tập hợp hàng hóa phức tạp hơn.
    - Ưu điểm:
        - Apriori đơn giản và dễ hiểu.
        - Nó hiệu quả trong việc xác định các mẫu phổ biến.
        - Nó giúp giảm không gian tìm kiếm bằng cách loại bỏ các tập hợp

hàng hóa không phổ biến, giảm độ phức tạp tính toán của thuật toán.
        - Nó đã được sử dụng và kiểm tra rộng rãi trong nhiều ứng dụng, là một thuật toán đáng tin cậy và được công nhận.
    - Nhược điểm:
        - Apriori không phù hợp cho các tập dữ liệu rất lớn, vì độ phức tạp tính toán của thuật toán tăng theo cấp số mũ trên kích thước của tập dữ liệu.
        - Nó có thể không hiệu quả trong việc xác định các mẫu trong các tập dữ liệu có nhiều hàng hóa hiếm hoặc giao dịch không thường xuyên.
        - Nó nhạy cảm với ngưỡng hỗ trợ tối thiểu và ngưỡng tin cậy tối thiểu, có thể ảnh hưởng đến chất lượng kết quả.
        - Nó có thể tạo ra một số lượng lớn các luật kết hợp, làm cho việc hiểu kết quả trở nên khó khăn.
- FP Growth: Thuật toán này xây dựng một cấu trúc cây gọi là FP-tree, mã hóa các tập hợp hàng hóa phổ biến trong tập dữ liệu. FP-tree được sử dụng để tạo ra các luật kết hợp tương tự như thuật toán Apriori. Thuật toán FP-Growth thường nhanh hơn thuật toán Apriori, đặc biệt là đối với các tập dữ liệu lớn.
    - Ưu điểm:
        - FP-Growth hiệu quả và nhanh hơn thuật toán Apriori, đặc biệt là đối với các tập dữ liệu lớn.
        - Nó sử dụng cấu trúc cây FP-tree để mã hóa các tập hợp hàng hóa phổ biến, giúp giảm độ phức tạp tính toán của thuật toán.
    - Nhược điểm:
        - FP-Growth đòi hỏi một bước tiền xử lý phức tạp để xây dựng cây FP-tree.
        - Nó có thể không hiệu quả trong việc xác định các mẫu trong các tập dữ liệu có nhiều hàng hóa hiếm hoặc giao dịch không thường xuyên.
- ECLAT: Đây là một biến thể của thuật toán Apriori, sử dụng phương pháp từ trên xuống thay vì từ dưới lên. Nó chia các hàng hóa thành các lớp tương đương dựa trên độ hỗ trợ của chúng. Luật kết hợp được tạo ra bằng cách kết hợp các lớp

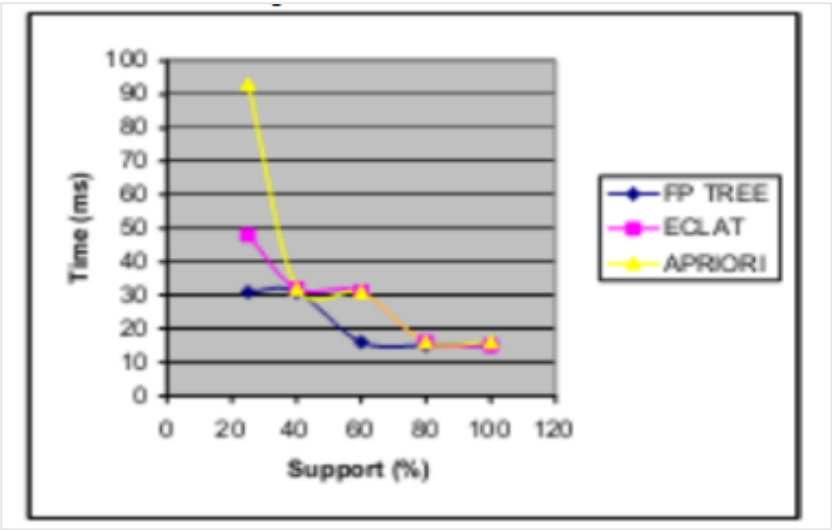
tương đương này thành một cấu trúc giống hình mạng lưới. Đây là một phiên bản thuật toán Apriori hiệu quả và có khả năng mở rộng tốt hơn.

- Ưu điểm:
  - ECLAT giúp giảm độ phức tạp tính toán so với Apriori bằng cách sử dụng phương pháp từ trên xuống.
  - Nó phân loại các hàng hóa thành các lớp tương đương dựa trên độ hỗ trợ của chúng.
- Nhược điểm:
  - ECLAT cần một bước tiền xử lý để phân loại hàng hóa thành các lớp tương đương.
  - Nó có thể không hiệu quả trong việc xác định các mẫu trong các tập dữ liệu có nhiều hàng hóa hiếm hoặc giao dịch không thường xuyên.

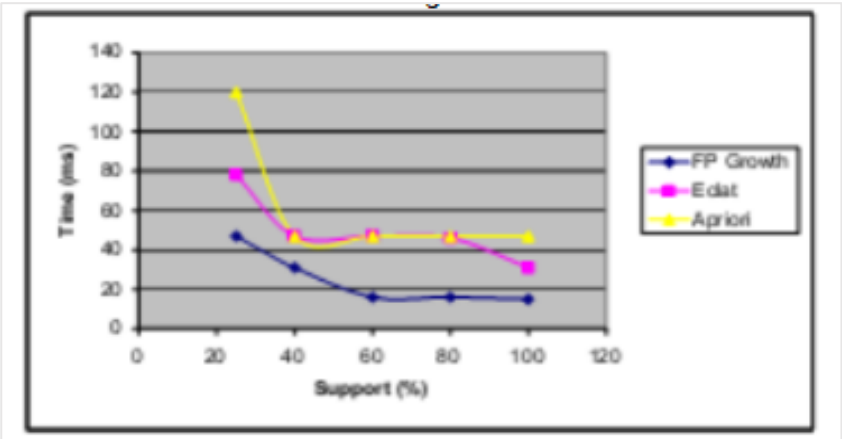
4.2. So sánh thuật toán

So sánh, kết luận theo lý thuyết: Dựa trên ưu và nhược điểm của cả hai thuật toán, có thể nói rằng thuật toán ECLAT là thuật toán tốt nhất trong số ba thuật toán Apriori, FP-Growth và ECLAT. ECLAT giúp giảm độ phức tạp tính toán và có khả năng xác định các mẫu trong các tập dữ liệu lớn có hàng hóa hiếm hoặc giao dịch không thường xuyên. Ngoài ra, ECLAT cũng không đòi hỏi bước tiền xử lý phức tạp như FP-Growth. Tuy nhiên, cần lưu ý rằng sự lựa chọn của thuật toán tốt nhất còn phụ thuộc vào yêu cầu cụ thể của bài toán và tính chất của dữ liệu được sử dụng

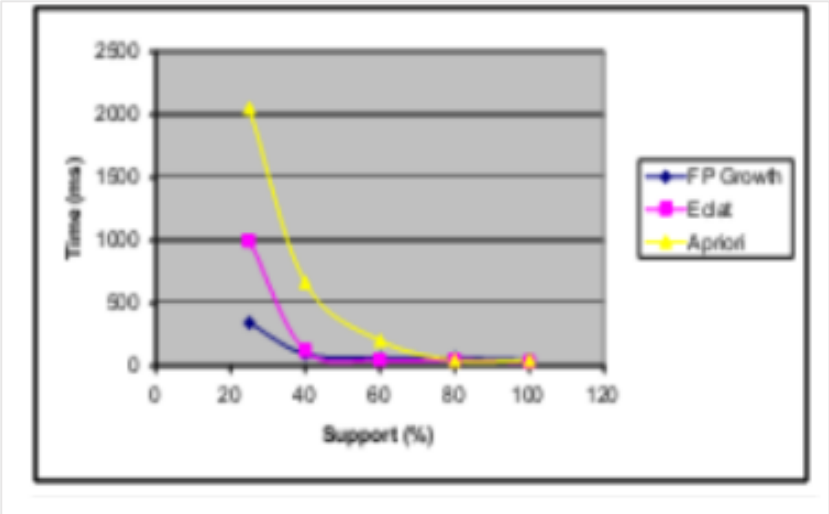
So sánh độ phức tạp của ba thuật toán: nhóm đã tham khảo bài của International Journal of Computer Applications về so sánh độ phức tạp của ba thuật toán và triển khai chúng bằng Java. Tác giả đã đánh giá hiệu suất của các thuật toán trên một tập dữ liệu tổng hợp, bao gồm số lượng thuộc tính và phiên bản khác nhau. Dưới đây là phân tích so sánh của các thuật toán thông qua việc thay đổi các tham số khác nhau.



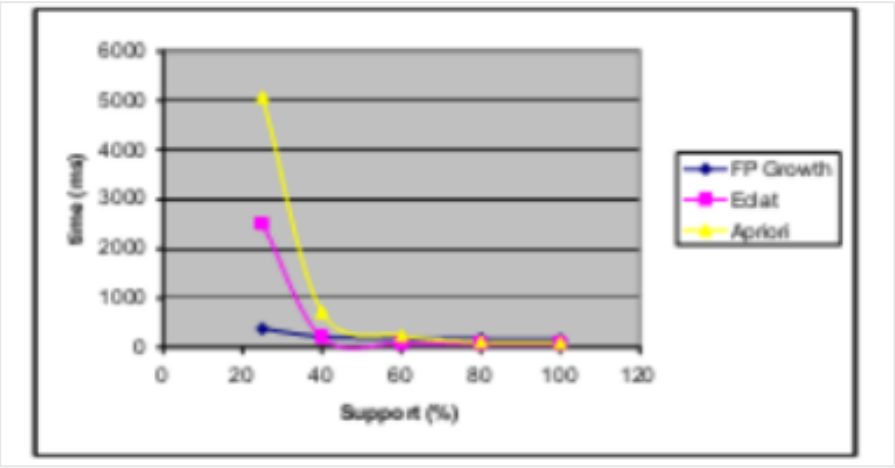
Hình 1. So sánh thuật toán Apriori, ECLAT và FP-Growth trên tập dữ liệu nhân tạo.



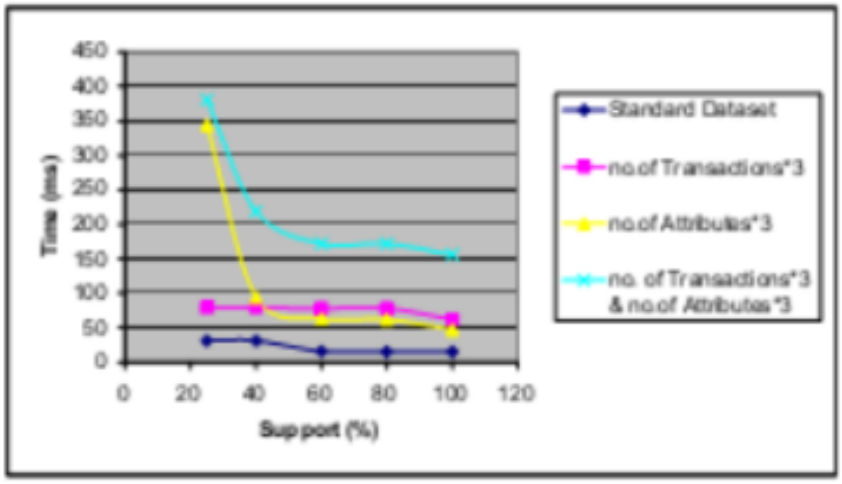
Hình 2. So sánh thuật toán Apriori, ECLAT và FP-Growth trên tập dữ liệu nhân tạo khi số lượng giao dịch gấp ba lần so với ban đầu.



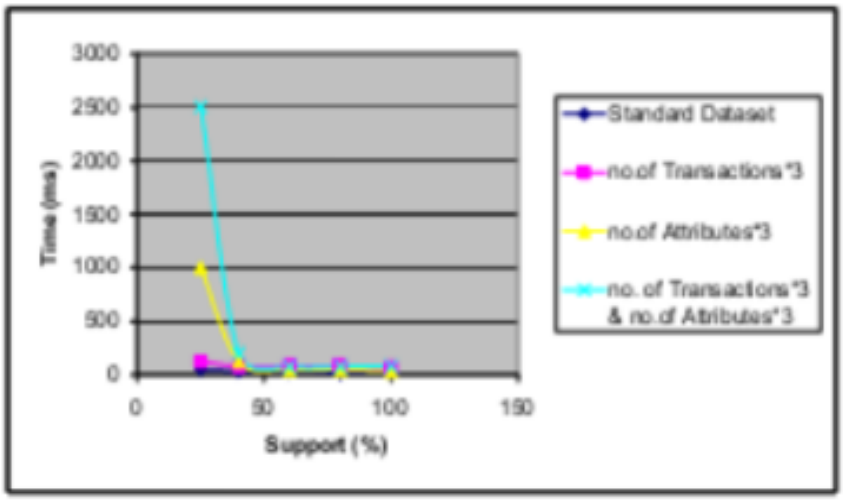
Hình 3. So sánh thuật toán Apriori, ECLAT và FP-Growth trên tập dữ liệu nhân tạo khi số lượng thuộc tính được tạo ra gấp ba lần so với ban đầu.



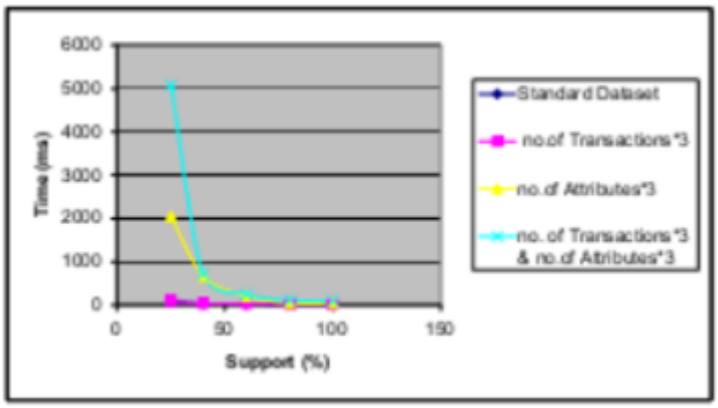
Hình 4. So sánh thuật toán Apriori, ECLAT và FP-Growth trên một tập dữ liệu với số lượng giao dịch gấp ba lần so với ban đầu và số lượng thuộc tính gấp ba lần so với ban đầu.



Hình 5. Sự tỷ lệ của thuật toán FP-Growth đối với số lượng thuộc tính và giao dịch.



Hình 6. Sự tỷ lệ của thuật toán ECLAT đối với số lượng thuộc tính và giao dịch.



Hình 7. Sự tỷ lệ của thuật toán Apriori đối với số lượng thuộc tính và giao dịch.

→ ***Kết luận:*** Trên bộ dữ liệu tiêu chuẩn, thuật toán FP-Growth cho kết quả tốt nhất, trong khi Apriori mất nhiều thời gian nhất. Khi số lượng giao dịch được tăng lên ba lần so với ban đầu, cả FP-Growth và ECLAT đều mất thời gian nhiều hơn Apriori. Khi số lượng thuộc tính được tăng lên ba lần so với ban đầu, FP-Growth cho kết quả tốt nhất, Apriori tăng đột ngột trong khi ECLAT nằm giữa hai thuật toán kia. Ngoài ra, từ hình 2 và hình 3, rõ ràng là việc tăng số lượng thuộc tính ảnh hưởng nhiều hơn đến mỗi thuật toán riêng lẻ so với việc tăng số lượng giao dịch theo cùng tỷ lệ. Khi số lượng giao dịch và số lượng bộ dữ liệu tăng lên ba lần, tất cả các thuật toán đều mất thời gian tăng đột ngột. Ở giai đoạn này, FP-Growth cho kết quả tốt nhất trong khi Apriori cho kết quả tồi nhất, như được thể hiện trên hình 4. Hình 5 cho thấy sự tỷ lệ của thuật toán FP-Growth dưới các điều kiện khác nhau. Hình 6 và hình 7 cho thấy sự tỷ lệ của ECLAT và Apriori dưới các điều kiện khác nhau. Từ sự so sánh giữa hình 5, hình 6 và hình 7, rõ ràng là FP-Growth hoạt động tốt dưới mọi biến thể và do đó là thuật toán tốt nhất trong ba thuật toán đó. Apriori cho kết quả tệ nhất trong ba thuật toán và do đó không có tính mở rộng tốt nhất, trong khi ECLAT nằm ở giữa FP-Growth và Apriori.

Tóm lại từ cả lý thuyết và so sánh độ phức tạp thì ECLAT là một thuật toán tốt để sử dụng trong khai phá luật kết hợp. Vì vậy nhóm quyết định chọn thuật toán ECLAT để tiến hành nghiên cứu cho bộ dữ liệu bài toán Market Basket Analysis.

CHƯƠNG 3: THUẬT TOÁN ECLAT

1. GIỚI THIỆU THUẬT TOÁN ECLAT

Thuật toán ECLAT viết tắt của Equivalence Class Clustering and bottom-up Lattice Traversal, được đề xuất bởi Zaki và cộng sự vào năm 1997. Nó là một trong những phương pháp khai phá luật kết hợp phổ biến nhất. Trong khi thuật toán Apriori hoạt động với dữ liệu được sử dụng dưới định dạng nằm ngang mô phỏng duyệt chiều rộng (Breadth-First Search) của đồ thị, thì thuật toán ECLAT hoạt động với dữ liệu được định dạng theo chiều dọc mô phỏng duyệt chiều sâu (Depth-First Search) của đồ thị.

Vì cách sắp xếp dữ liệu nên ECLAT không cần phải quét dữ liệu toàn bộ dữ liệu cho mỗi lần tính toán như Apriori nên ECLAT chạy nhanh hơn và đơn giản hơn. Hướng tiếp cận theo chiều dọc của thuật toán ECLAT khiến nó nhanh hơn so với thuật toán Apriori.

2. MÔ TẢ THUẬT TOÁN

2.1. Giải thích ký hiệu

Ký hiệu	Ý nghĩa
D	Bộ dữ liệu giao dịch (Transaction Database)
s	Ngưỡng hỗ trợ tối thiểu (minSup)
I	Tập mục (itemsets)
θ	Tập đóng xuống các tập con của `I` với ngưỡng s (được giải thích bên dưới)

Bảng 1. Giải thích ký hiệu trong mã giả của thuật toán ECLAT

2.2. Mã giả (Pseudocode):

```
Input: D, s, I ⊆ θ
Output: F[I](D, s)
1:    F[I] := {}
2:    for all i ϵ θ occurring in D do
3:        F[I] := F[I] ∪ {I ∪ {i}}
4:        //Create D^i
5:        D^i := {}
6:        for all j ϵ θ occurring in D such that j > i do
```

```
7:          C := cover({i}) ∩ cover({j})
8:          if |C| ≥ s then
9:              D^i := D^i ∪ {(j, C)}
10:         end if
11:     end for
12:     //Depth-first recursion
13:     Compute F[I ∪ {i}](D^i, s)
14:     F[I] := F[I] ∪ F[I ∪ {i}]
15: end for
```

Bảng 2. Mã giả (pseudocode) của thuật toán ECLAT

(MM Hlaing, 2019)

Trong đó:

$$cover(X, D) := \{tid | (tid, \theta) \epsilon D, X \subseteq \theta\}$$

$$F(D, s) := \{X \epsilon \theta | support(X, D) \geq s\}$$

3. ĐÁNH GIÁ LUẬT KẾT HỢP

Trong khai phá luật kết hợp, một số chỉ số được dùng để đánh giá chất lượng và tầm quan trọng của các luật kết hợp được phát hiện để có thể chọn ra các luật phù hợp nhất.

A. Utility Function - Hàm hữu dụng

Hữu dụng tiềm ẩn của một mẫu là nhân tố xác định mức độ đáng quan tâm của nó. Có thể đo lường bởi một hàm hữu dụng, như là support. Support là tỷ lệ xuất hiện (tần suất) các items quan tâm trong toàn tập dữ liệu. Nó được tính bằng số lượng giao dịch có một item hoặc các item chia cho tổng số lượng giao dịch trong tập dữ liệu. Chỉ số Support cao chỉ ra rằng item hoặc các item phổ biến trong tập dữ liệu, trong khi chỉ số support thấp cho thấy item hoặc các item rất hiếm.

$$Support(\{A\} \rightarrow \{B\}) = P(A \cup B) = \frac{freq(A \cup B)}{|D|}$$

Ví dụ với luật A→B (với A và B là các tập item) có support là s nếu trong toàn bộ giao dịch có s% giao dịch chứa cả A và B.

B. Certainty Function - Hàm chắc chắn

Hàm chắc chắn trong khai phá luật kết hợp giúp đo lường Confidence. Confidence chỉ sự chắc chắn của các luật kết hợp. Confidence được tính bằng số lượng giao dịch chứa cả hai mục chia cho số lượng giao dịch có chứa mục đầu tiên. Độ tin cậy

(hay chỉ số Confidence) cao cho thấy rằng khi xuất hiện mục đầu tiên thì khả năng cao sẽ xuất hiện mục thứ hai.

$$Confidence(\{A\} \rightarrow \{B\}) = P(B|A) = \frac{freq(A \cup B)}{freq(A)}$$

Với A và B là các tập có độ tin cậy. Luật A → B (A và B là các tập item) có confidence là c nếu trong các giao dịch có chứa A thì có c% giao dịch có chứa B.

C. Lift Function

Hàm Lift là hàm đo mức độ liên kết giữa hai mục, trong đó có tính đến tần suất của hai mục trong tập dữ liệu. Lift được tính bằng độ tin cậy (confidence) của liên kết chia cho độ hỗ trợ (support) của mục thứ hai. Lift được dùng để so sánh độ liên kết giữa hai mục với mức độ liên kết nếu các mục độc lập với nhau.

Nếu giá trị Lift lớn hơn 1 cho thấy mức độ liên kết giữa hai mặt hàng mạnh hơn so với tần suất của từng mặt hàng. Điều này cho thấy liên kết có thể có ý nghĩa. Nếu giá trị Lift nhỏ hơn 1 cho thấy mối liên kết yếu có thể kém tin cậy hoặc ít quan trọng hơn.

Lift({A}{B})=(Transactions containing both A and B)/(Transactions containing A)Fraction of transactions containing B

- Lift < 1 (tương quan âm): Nếu khả năng xảy ra A cao thì ít có khả năng xảy ra B.
- Lift = 1 (không có tương quan): Hai hành động A và B không ảnh hưởng tới nhau.
- Lift > 1 (tương quan dương): Nếu khả năng xảy ra A cao thì sẽ xảy ra B.

4. MÔ HÌNH KHAI PHÁ LUẬT KẾT HỢP

Thuật toán ECLAT (Equivalence Class Clustering and Bottom-Up Lattice Traversal) là một thuật toán khai thác quy luật kết hợp (association rules) trong dữ liệu. Dưới đây là mô tả chi tiết về từng bước của thuật toán ECLAT:

Giai đoạn 1: Tìm tất cả tập phổ biến thỏa mãn minSup.

Bước 1: Quét cơ sở dữ liệu để tạo biểu diễn dạng dọc của cơ sở dữ liệu.

- Tính minFreq theo minSup, với $minFreq = minSup.(số\ lượng\ giao\ dịch)$
- Quét cơ sở dữ liệu để xác định tần suất xuất hiện của từng item riêng lẻ trong cơ sở dữ liệu.
- Tạo một bảng để lưu trữ thông tin về sự xuất hiện của từng item trong các giao dịch.

Bước 2: Tạo ra lớp tương đương đầu tiền.

- Bước này tạo ra các tập tương đương của kích thước 1 (tập chứa một phần tử).

- Mỗi phần tử của tập tương đương là một giao dịch chứa item có sẵn trong cơ sở dữ liệu.
- Loại bỏ những item nào có số lượng giao dịch thấp hơn minFreq.
- Thu được frequent itemset với kích thước 1.

Bước 3: ECLAT kết hợp các itemset của tập tương đương có kích thước k để tạo các tập tương đương có kích thước k + 1.

- Tạo các tập tương đương của kích thước k + 1 bằng cách kết hợp các itemset của tập tương đương hiện tại (kích thước k).
- Cách kết hợp itemset có thể thay đổi tùy thuộc vào quy ước cụ thể của thuật toán và dữ liệu.
- Loại bỏ những itemset nào có số lượng giao dịch thấp hơn minFreq.
- Thu được frequent itemset với kích thước k+1.

Bước 4: ECLAT xử lý đệ quy từng tập tương đương theo cùng cách.

- Áp dụng đệ quy cho mỗi tập tương đương được tạo ra trong bước trước.
- Quy trình này sẽ tạo ra các tập tương đương của kích thước (k + 2), (k + 3), và tiếp tục cho đến khi không còn tập tương đương nào có thể được tạo ra.

Bước 5: Các lớp tương đương khác chứa chỉ một itemset.

- Tất cả các lớp tương đương khác đều chứa chỉ một itemset, do đó không thể tạo ra các tập tương đương mới từ chúng.
- Thuật toán dừng lại khi không còn có tập tương đương nào có thể được tạo ra.

Giai đoạn 2: Tạo các luật kết hợp mạnh từ tập phổ biến thỏa minSup và minConf.

- Từ tập phổ biến, tạo các luật kết hợp có thể.
- So sánh với minSup và loại nếu thấp hơn minSup.

VD: Tìm các mẫu phổ biến với minFreq = 2.

Tid	Items
T1	bánh, chocolate, sữa
T2	bia, chocolate
T3	chocolate, hamburger, sữa
T4	bánh, hamburger, sữa
T5	chocolate, hamburger, sữa

C1

Items	Transactions
bánh	T1, T3
bia	T2
chocolate	T1, T2, T3, T5
hamburger	T3, T5
sữa	T1, T3, T4, T5

L1

Items	Transactions
bánh	T1, T3
chocolate	T1, T2, T3, T5
hamburger	T3, T5
sữa	T1, T3, T4, T5

C2

Items	Transactions
bánh, chocolate	T1, T3
bánh, hamburger	T3
bánh, sữa	T1, T3
chocolate, hamburger	T3, T5
chocolate, sữa	T1, T3, T5
hamburger, sữa	T3, T5

L2

Items	Transactions
bánh, chocolate	T1, T3
bánh, sữa	T1, T3
chocolate, hamburger	T3, T5
chocolate, sữa	T1, T3, T5
hamburger, sữa	T3, T5

C3

Items	Transactions
bánh, chocolate, hamburger	T3
bánh, chocolate, sữa	T1, T3
bánh, hamburger, sữa	T3
chocolate, hamburger, sữa	T3, T5

L3

Items	Transactions
bánh, chocolate, sữa	T1, T3
chocolate, hamburger, sữa	T3, T5

C4 = ∅

Items	Transactions
bánh, chocolate, hamburget, sữa	T3

Bảng 3. Các bước Khai phá mẫu phổ biến theo thuật toán ECLAT

Vậy các tập phổ biến là:

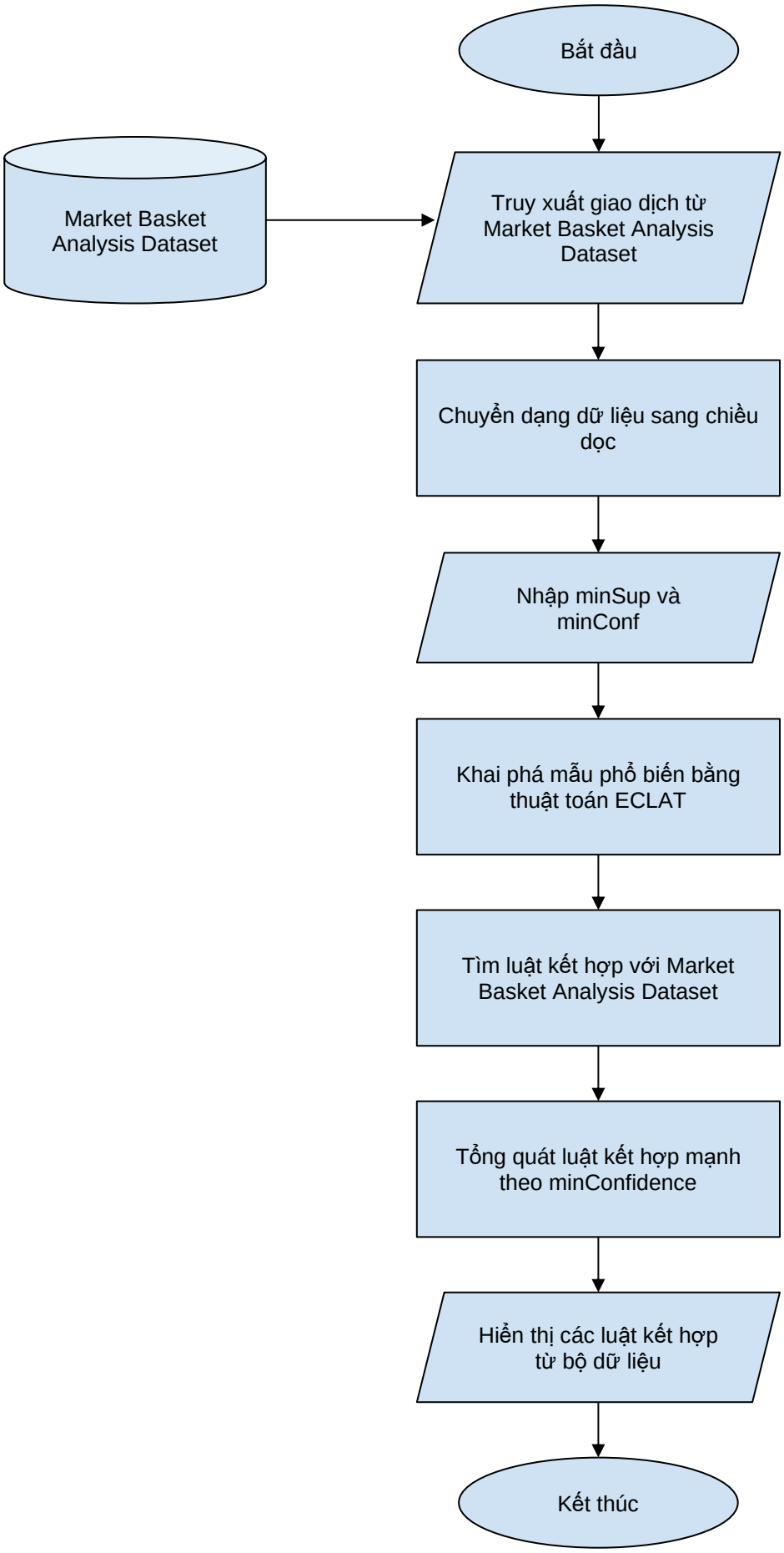
{bánh}	support = 2/5

{chocolate} {hamburger} {sữa}	support = 4/5 support = 2/5 support = 4/5
{bánh, chocolate} {bánh, sữa} {chocolate, hamburger} {chocolate, sữa} {hambuger, sữa}	support = 2/5 support = 2/5 support = 2/5 support = 3/5 support = 2/5
{bánh, chocolate, sữa} {chocolate, hamburger, sữa}	support = 2/5 support = 2/5

Bảng 4. Mẫu phổ biến tìm được thông qua thuật toán ECLAT

Thuật toán ECLAT giúp khai thác quy luật kết hợp trong cơ sở dữ liệu dạng cột (vertical), tập trung vào việc tìm các tập tương đương có sự xuất hiện chung cao trong dữ liệu.

5. ỨNG DỤNG MÔ HÌNH KHAI PHÁ LUẬT KẾT HỢP VÀO MARKET BASKET ANALYSIS



Hình 8. Mô hình Khai phá luật kết hợp cho Market Basket Analysis

CHƯƠNG 4: ÁP DỤNG THUẬT TOÁN ECLAT VÀO MARKET BASKET ANALYSIS

1. MÔ TẢ DỮ LIỆU

Bộ dữ liệu Market Basket Analysis 4 ghi lại mặt hàng trong giỏ hàng siêu thị của các khách hàng với mỗi dòng đại diện cho một giao dịch còn mỗi cột đại diện cho một item trong giỏ hàng.

```
# Kích thước bộ dữ liệu
df.shape
```

```
(14963, 11)
```

```
# Quan sát các cột dữ liệu
df.info()
```

```
RangeIndex: 14963 entries, 0 to 14962
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       14963 non-null  object
 1   1       14963 non-null  object
 2   2       4883 non-null   object
 3   3       2185 non-null   object
 4   4       795 non-null    object
 5   5       451 non-null    object
 6   6       276 non-null    object
 7   7       196 non-null    object
 8   8       51 non-null     object
 9   9       1 non-null      object
 10  10      1 non-null      object
dtypes: object(11)
```

Dựa trên thông tin từ df.shape và df.info(), bộ dữ liệu có 14963 dòng đại diện cho 14963 giao dịch, mỗi hàng chứa các tuy nhiên mỗi sản phẩm trên 1 giao dịch thì nằm ở 1 cột khác nhau. Sau đó,nhóm sử dụng hàm data.head() để xem 5 hàng đầu tiên

của dữ liệu.

```
# Quan sát các dòng dữ liệu đầu
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10
0	whole milk	pastry	salty snack	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	sausage	whole milk	semi-finished bread	yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	soda	pickled vegetables	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	canned beer	misc. beverages	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	sausage	hygiene articles	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Hình 9. Các dòng đầu của bộ dữ liệu.

Dữ liệu NaN thường xuất hiện khi có các giá trị thiếu trong bảng dữ liệu. Tuy nhiên, trong trường hợp của bộ dữ liệu nhóm nghiên cứu, dòng NaN không phải là do dữ liệu bị thiếu mà là kết quả dựa trên quyết định thiết kế cụ thể của dữ liệu, và dòng NaN không phải là kết quả của dữ liệu bị thiếu mà là một biểu diễn của trạng thái cụ thể của giao dịch.

Cụ thể, nhóm xác định rằng nếu một giao dịch có 0 sản phẩm, thì tất cả các cột khác đều được đặt là NaN. Tương tự, nếu một giao dịch có 1 sản phẩm, thì tất cả các cột khác cũng được đặt là NaN ngoại trừ cột đầu tiên chưa thông tin về sản phẩm.

2. PHÂN TÍCH KHÁM PHÁ DỮ LIỆU

2.1. Xử lý trùng trong giao dịch

Bước đầu nhóm tiến hành kiểm tra số lượng xuất hiện của khoảng 15 sản phẩm phổ biến để có cái nhìn tổng quát về bộ dữ liệu.

```
# Tính tổng số lượng từng sản phẩm
items_total = df.apply(pd.Series.value_counts).sum(axis=1)
items_total = pd.DataFrame({'items': items_total.index,
'transactions': items_total.values})
items_total.sort_values('transactions',
ascending=False).head(15).reset_index(drop =
True).style.background_gradient(cmap='Blues')
```

	items	transactions
0	whole milk	2502.000000
1	other vegetables	1898.000000
2	rolls/buns	1716.000000
3	soda	1514.000000
4	yogurt	1334.000000
5	root vegetables	1071.000000
6	tropical fruit	1032.000000
7	bottled water	933.000000
8	sausage	924.000000
9	citrus fruit	812.000000
10	pastry	785.000000
11	pip fruit	744.000000
12	shopping bags	731.000000
13	canned beer	717.000000
14	bottled beer	687.000000

Hình 10. Các item phổ biến nhất của dataset (trước khi xử lý)

Cụ thể, 5 mặt hàng bán chạy nhất là whole milk, other vegetables, rolls/buns, soda, và yogurt. Đây đều là những mặt hàng thiết yếu, tiêu dùng thường xuyên trong các hộ gia đình. Xếp sau nhóm thực phẩm thiết yếu là các loại trái cây và rau củ như trái cây nhiệt đới, và trái cây họ cam quýt (tropical fruit, citrus fruit). Đây là những loại thực phẩm giàu dinh dưỡng, đang được người dân ngày càng quan tâm. Bia và các loại nước đóng chai/lon là những sản phẩm cũng được mua khá nhiều cho thấy nhu cầu về đồ uống giải khát tại cửa hàng cũng khá cao. Các mặt hàng khác như bánh ngọt, túi đựng hàng, xúc xích, trái cây hạt cứng, cho thấy nhu cầu đa dạng của người mua sắm tại cửa hàng.

Sau khi kiểm tra tổng số lượng theo từng sản phẩm thì nhóm muốn kiểm tra trên từng giao dịch.

```
# Kiểm tra tính duy nhất của các item trong mỗi giao dịch
df['num_uniq'] = df.iloc[:,0:11].apply(pd.Series.nunique, axis=1) #Số item duy nhất trong giao dịch
df['num_item'] = df.iloc[:,0:11].count(axis=1) #Số item trong giao dịch
print("--- CÁC GIAO DỊCH CÓ CHỨA ITEM TRÙNG LẶP ---")
df[df['num_uniq'] != df['num_item']]
```

--- CÁC GIAO DỊCH CÓ CHỨA ITEM TRÙNG LẶP ---

	0	1	2	3	4	5	6	7	8	9	10	num_uniq	num_item
15	rolls/buns	rolls/buns	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2
27	rolls/buns	rolls/buns	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2
31	whole milk	frankfurter	chicken	frankfurter	flour	chocolate	bottled beer	rolls/buns	NaN	NaN	NaN	7	8
33	tropical fruit	soda	yogurt	root vegetables	yogurt	domestic eggs	white wine	photo/film	NaN	NaN	NaN	7	8
102	root vegetables	whipped/sour cream	root vegetables	canned beer	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
14841	bottled water	root vegetables	bottled water	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	3
14863	beef	beef	margarine	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	3
14901	frankfurter	margarine	whole milk	soda	margarine	brown bread	margarine	frozen vegetables	NaN	NaN	NaN	6	8
14923	candles	rolls/buns	other vegetables	rolls/buns	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	4
14936	margarine	margarine	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	2

707 rows × 13 columns

Hình 11. Các transaction chứa item trùng lặp

Và vì thuật toán Apriori và ECLAT được tính toán dựa trên độ phổ biến của các item trong tập dữ liệu. Nên sự xuất hiện của các items bị trùng lặp có trong bộ dữ liệu sẽ ảnh hưởng đến kết quả, do đó nhóm sẽ lọc đi những giá trị trùng lặp để đảm bảo mỗi giao dịch chỉ chứa các item duy nhất.

```
# Xử lý các giao dịch và đưa vào df mới sau đó kiểm tra lại tính duy nhất
df_cleaned = []

df = df.replace(np.nan, None)

for _, row in df.iloc[:,0:11].iterrows():
  temp = []
  for column, value in row.iteritems():
    if (value is not None) and (value not in temp):
      i = f"{value}"
      temp.append(i)
  df_cleaned.append(temp)

df_cleaned = pd.DataFrame(df_cleaned)
df_cleaned['num_uniq'] =
df_cleaned.iloc[:,0:10].apply(pd.Series.nunique, axis=1)
df_cleaned['num_item'] = df_cleaned.iloc[:,0:10].count(axis=1)
print('Số giao dịch có chứa item trùng lặp là:',
len(df_cleaned[df_cleaned['num_uniq'] != df_cleaned['num_item']]))
```

	0	1	2	3	4	5	6	7	8	9	num_uniq	num_item
0	whole milk	pastry	salty snack	None	None	None	None	None	None	None	3	3
1	sausage	whole milk	semi-finished bread	yogurt	None	None	None	None	None	None	4	4
2	soda	pickled vegetables	None	None	None	None	None	None	None	None	2	2
3	canned beer	misc. beverages	None	None	None	None	None	None	None	None	2	2
4	sausage	hygiene articles	None	None	None	None	None	None	None	None	2	2
...	...	...	...	...	...	...	...	...	...	...	...	...
14958	butter milk	whipped/sour cream	None	None	None	None	None	None	None	None	2	2
14959	bottled water	herbs	None	None	None	None	None	None	None	None	2	2
14960	fruit/vegetable juice	onions	None	None	None	None	None	None	None	None	2	2
14961	bottled beer	other vegetables	None	None	None	None	None	None	None	None	2	2
14962	soda	root vegetables	semi-finished bread	None	None	None	None	None	None	None	3	3

14963 rows × 12 columns

Hình 12. Bộ dữ liệu sau khi xử lý các item trùng lặp

Dữ liệu thu được sau khi xử lý như sau:

```
# Thu được DataFrame sau khi xử lý (Vì lọc item trùng nên số cột cũng giảm)
df_cleaned = df_cleaned.iloc[:,0:10]
df_cleaned = df_cleaned.fillna(value= float('nan'))
df_cleaned
```

	0	1	2	3	4	5	6	7	8	9
0	whole milk	pastry	salty snack	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	sausage	whole milk	semi-finished bread	yogurt	NaN	NaN	NaN	NaN	NaN	NaN
2	soda	pickled vegetables	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	canned beer	misc. beverages	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	sausage	hygiene articles	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
14958	butter milk	whipped/sour cream	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14959	bottled water	herbs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14960	fruit/vegetable juice	onions	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14961	bottled beer	other vegetables	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14962	soda	root vegetables	semi-finished bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN

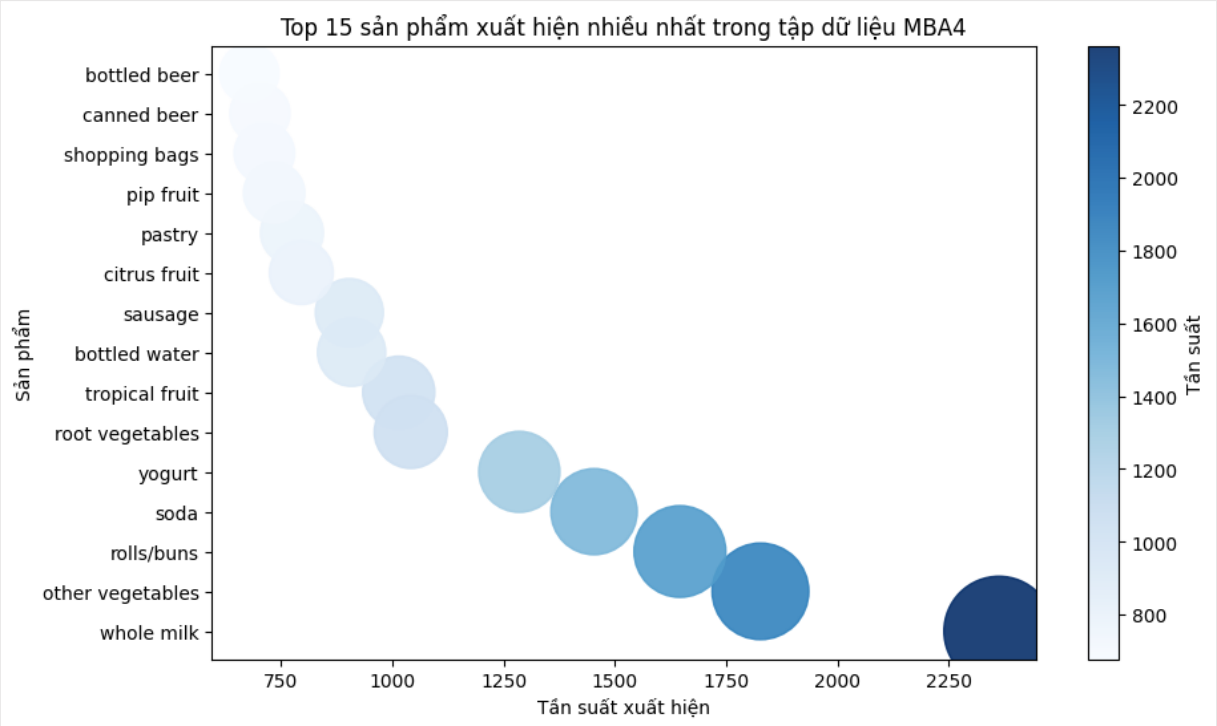
14963 rows × 10 columns

Hình 13. Bộ dữ liệu tên df_cleaned được xử dụng để khai phá luật kết hợp

2.2. Phân tích sản phẩm

Vẽ biểu đồ scatter plot, bar chart và tree map để quan sát những sản phẩm được mua nhiều nhất trong các giao dịch.

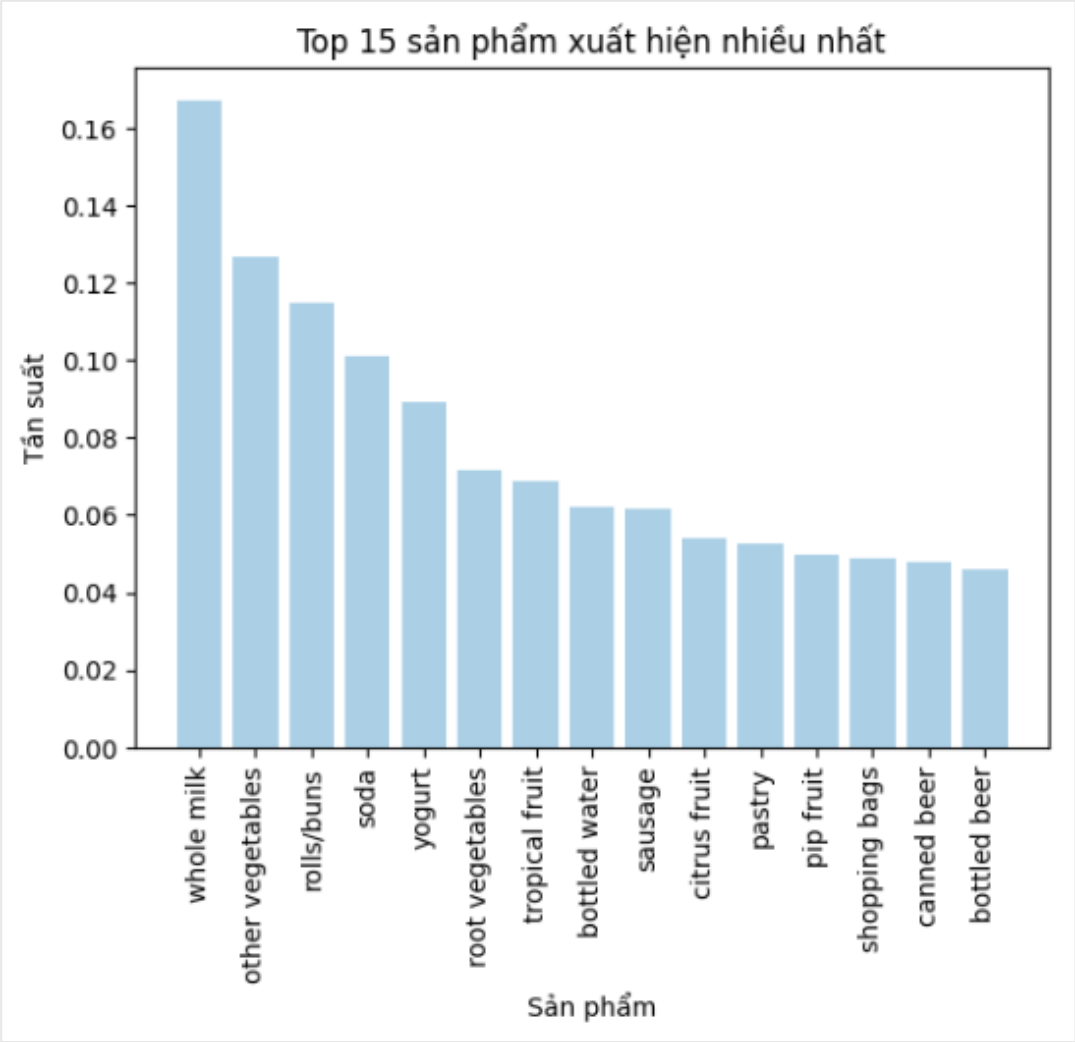
```
# Biểu đồ scarter plot thể hiện top 15 sản phẩm được xuất hiện trong giỏ hàng nhiều nhất
# ----------------------------------------------------------------------
items_list = df_cleaned.values.flatten()
frequency = pd.Series(items_list).value_counts().head(15)
fig, ax = plt.subplots(figsize=(10, 6))
scatter = ax.scatter(frequency.values, frequency.index, s=frequency.values * 1.5, alpha=0.9, c=frequency.values, cmap='Blues')
ax.set_xlabel('Tần suất xuất hiện')
ax.set_ylabel('Sản phẩm')
ax.set_title('Top 15 sản phẩm xuất hiện nhiều nhất trong tập dữ liệu MBA4')
cbar = plt.colorbar(scatter)
cbar.ax.set_ylabel('Tần suất')
plt.show()
```



Hình 14. Biểu đồ scarter plot thể hiện top15 item xuất hiện trong giỏ hàng nhiều nhất

```
# Biểu đồ bar plot thể hiện tần suất của top 15 sản phẩm được xuất hiện trong giỏ hàng nhiều nhất
# ----------------------------------------------------------------------
```

```
product_counts = df_cleaned.apply(pd.Series.value_counts)
total_samples = len(df)
product_frequencies = product_counts / total_samples
top_products =
product_frequencies.sum(axis=1).sort_values(ascending=False).head(15)
plt.bar(top_products.index, top_products.values, color='#ABD0E6')
plt.xlabel('Sản phẩm')
plt.ylabel('Tần suất')
plt.title('Top 15 sản phẩm xuất hiện nhiều nhất')
plt.xticks(rotation=90)
plt.show()
```

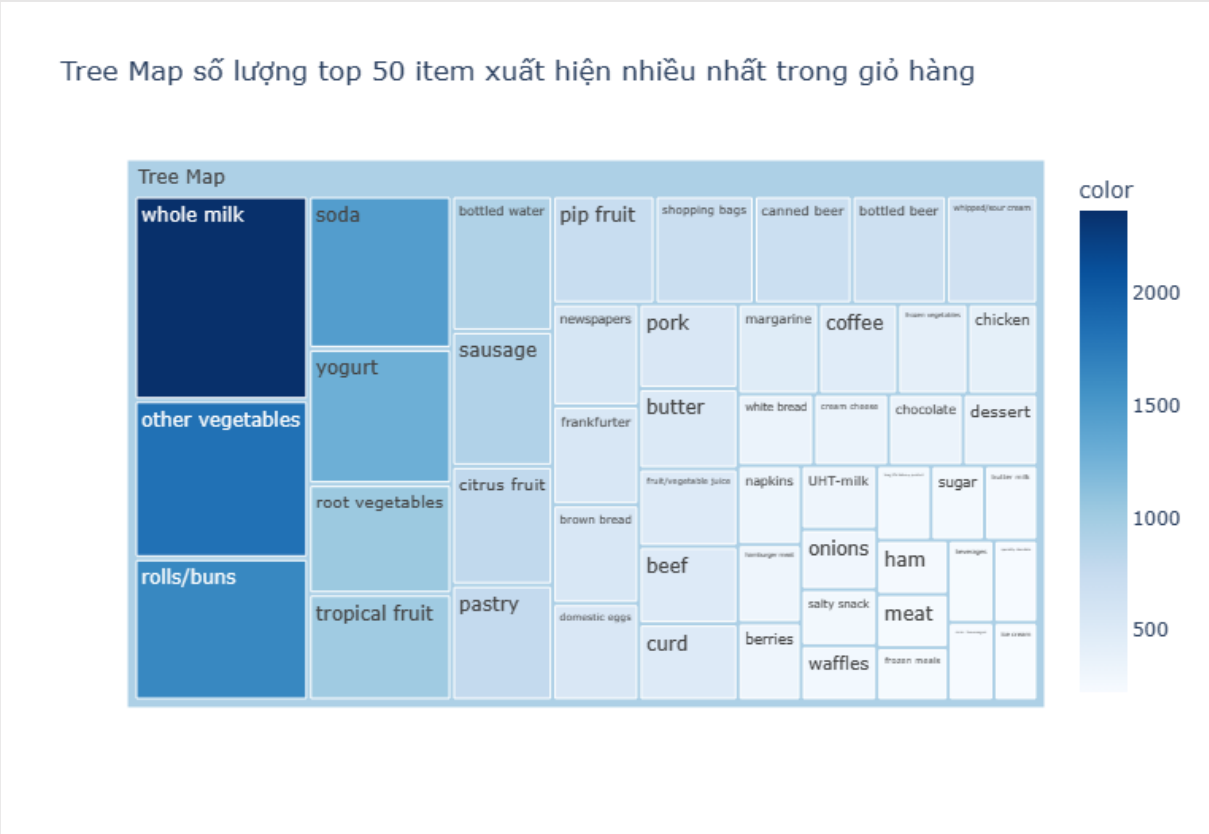


Hình 15. Biểu đồ bar plot thể hiện tần suất top15 item xuất hiện trong giỏ hàng nhiều nhất.

```
# Tree Map biểu diễn số lượng top 50 mặt hàng xuất hiện nhiều nhất
trong giỏ hàng
```

```
# ----------------------------------------------------------------------
items_total_clean["all"] = "Tree Map"
items_total_clean = items_total_clean.sort_values('transactions',
ascending=False)

fig = px.treemap(items_total_clean.head(50), path=['all', "items"],
values='transactions',
                 color=items_total_clean["transactions"].head(50),
hover_data=['items'],
                 color_continuous_scale='Blues',
                 title="Tree Map số lượng top 50 item xuất hiện nhiều
nhất trong giỏ hàng")
fig.show()
```



Hình 16. Tree Map biểu diễn số lượng top 50 item xuất hiện nhiều nhất trong giỏ hàng

Nhận xét:

Ba biểu đồ đều thể hiện mức tiêu thụ của các sản phẩm phổ biến nhất trong bộ dữ liệu Market Basket Analysis.

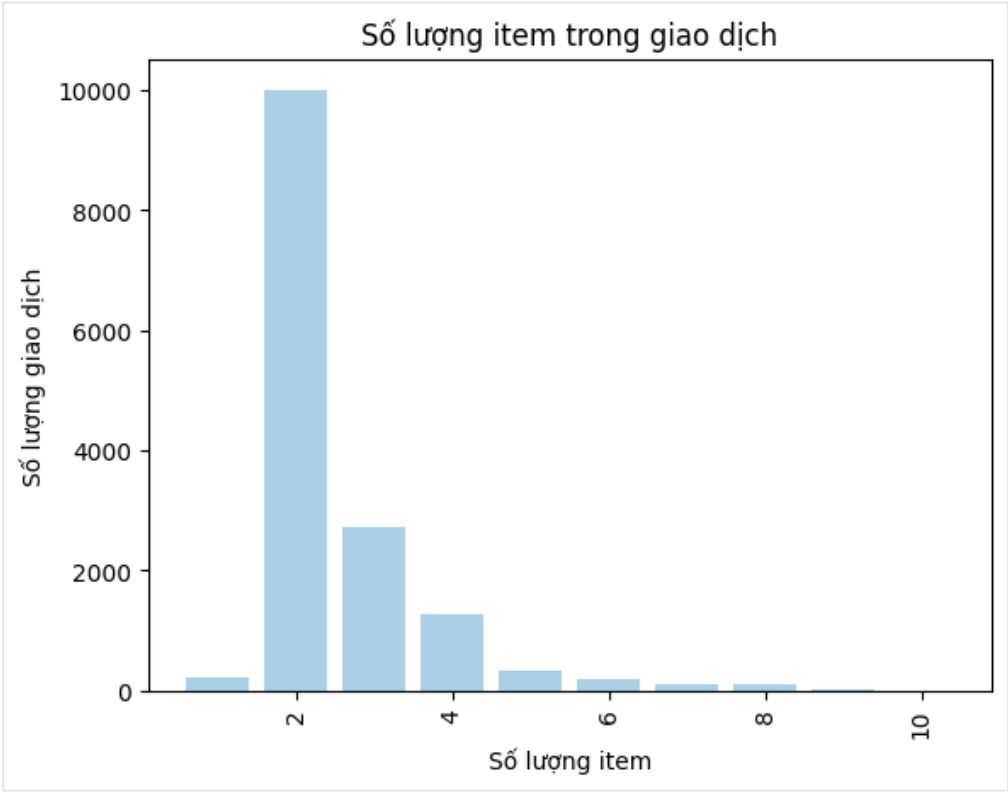
- Whole mile (sữa nguyên kem) là sản phẩm được tiêu thụ nhiều nhất, với mức

tiêu thụ hơn 2.200 đơn vị. Điều này cho thấy sữa là một loại thức uống phổ biến.

- Các sản phẩm khác được tiêu thụ nhiều bao gồm các loại rau củ, rolls/buns (một loại bánh), soda, sữa chua, tropical fruit (trái cây nhiệt đới), bottle water (nước đóng chai.
- Các sản phẩm được tiêu thụ ít hơn bao gồm các loại báo, túi mua sắm, …

Biểu đồ cho thấy sự tiêu thu đa dạng các sản phẩm, bia, đồ uống có ga và các loại thực phẩm là những sản phẩm được tiêu thụ nhiều nhất.

2.3. Phân tích giao dịch



Hình 17. Biểu đồ bar chat thể hiện số lượng item trong các giao dịch

Nhận xét:

Từ biểu đồ thanh, có thể thấy rằng số lượng item trong mỗi giao dịch dao động từ 1 đến 10 items. Có hai loại giao dịch chính, với số lượng giao dịch tương đương nhau:

- Hầu hết mọi giao dịch chứa 2 items, với số lượng gần 10000 giao dịch.
- Các giao dịch có 3, 4, 5 items có chiến số lượng ít hơn.
- Có khá ít các giao dịch từ 6 items trở lên.

3. TIỀN XỬ LÝ DỮ LIỆU

Bài toán khai phá luật kết hợp trong dữ liệu giao dịch Market Basket Analysis 4 đặt

ra mục tiêu tìm kiếm các mối quan hệ kết hợp giữa các mặt hàng, giúp hiểu rõ hơn về hành vi mua sắm của khách hàng. Đối với bộ dữ liệu này, nhóm đã xác định ngưỡng hỗ trợ tối thiểu (minSup) là 0.001 và ngưỡng độ tin cậy tối thiểu (minConf) là 0.03.

Ngưỡng hỗ trợ (minSup) là tỷ lệ giao dịch tối thiểu mà một itemset phải xuất hiện trong đó để được xem xét. Trong trường hợp này, nếu một itemset xuất hiện trong ít nhất 0.1% số lượng giao dịch, thì nó sẽ được xem xét trong quá trình khai phá.

Ngưỡng độ tin cậy (minConf) là một yếu tố quan trọng để lựa chọn những luật kết hợp có độ chắc chắn đủ cao. Trong bối cảnh minConf = 0.03, có nghĩa là chỉ những luật có xác suất xảy ra kết quả (confidence) từ 3% trở lên sẽ được xem xét. Các luật kết hợp thỏa điều kiện trên có thể được sử dụng để tìm hiểu mối quan hệ giữa các mặt hàng, từ đó đưa ra các khuyến nghị sản phẩm cho khách hàng.

```
# Nhập các ngưỡng tối thiểu khi khai phá dữ liệu
min_support = eval(input("Nhập độ hỗ trợ tối thiểu - minSup (e.g. 0.01): "))
min_confidence = eval(input("Nhập độ chắc chắn tối thiểu - minConf (e.g. 0.3): "))
```

```
Nhập độ hỗ trợ tối thiểu - minSup (e.g. 0.01): 0.001
Nhập độ chắc chắn tối thiểu - minConf (e.g. 0.3): 0.03
```

Trong quá trình huấn luyện mô hình khai phá luật kết hợp dựa trên thuật toán ECLAT, nhóm đã triển khai ba phương pháp chính để so sánh và đánh giá hiệu suất của mỗi phương pháp. Đầu tiên, nhóm đã sử dụng thư viện PyECLAT để thực hiện ECLAT, trên cơ sở là một thư viện Python mạnh mẽ được tối ưu hóa cho việc khai phá luật kết hợp. Sự linh hoạt và tiện lợi của PyECLAT giúp nhóm thực hiện quy trình này một cách hiệu quả và tiết kiệm thời gian.

Tiếp theo, nhóm đã xây dựng một phiên bản tùy chỉnh của thuật toán ECLAT, dựa trên ý tưởng cơ bản của nó. Việc xây dựng phiên bản tùy chỉnh này mang lại sự linh hoạt cho nhóm để tinh chỉnh và điều chỉnh thuật toán theo nhu cầu cụ thể của dự án. Điều này giúp đảm bảo rằng thuật toán có thể được điều chỉnh để đáp ứng yêu cầu đặc biệt của tập dữ liệu hoặc mục tiêu của dự án.

Cuối cùng là thuật toán Apriori và thực hiện so sánh với kết quả của hai mô hình trước để đánh giá hiệu suất. So sánh này cung cấp cái nhìn tổng quan về tính hiệu quả của ECLAT trong ngữ cảnh cụ thể của dự án và giúp nhóm xác định thuật toán nào phù

hợp nhất với mục tiêu khai phá dữ liệu của họ.

Tổng cộng, quá trình này không chỉ giúp nhóm hiểu rõ hơn về hiệu suất của thuật toán ECLAT mà còn cung cấp sự linh hoạt và sự tùy chỉnh cho phương pháp triển khai, đồng thời đưa ra quyết định thông minh trong việc lựa chọn thuật toán dựa trên yêu cầu cụ thể của dự án.

3.1. Thuật toán ECLAT (PyECLAT lib)

Nhóm sử dụng hàm ECLAT từ thư viện pyECLAT để chuyển đổi DataFrame df_cleaned thành bảng boolean. Trong bảng này, mỗi hàng biểu diễn một giao dịch và mỗi cột biểu diễn một mặt hàng. Giá trị trong mỗi ô của bảng là 1 nếu mặt hàng đó xuất hiện trong giao dịch và 0 nếu không. Điều này tạo ra một biểu diễn dữ liệu thuận tiện để áp dụng thuật toán ECLAT để khám phá các luật kết hợp trong tập dữ liệu đã được xử lý.

```
df1 = df_cleaned
eclat = ECLAT(data=df1)
eclat.df_bin
```

Sau khi tiến hành chuyển dạng dữ liệu thì tiến hành huấn luyện mô hình khai phá luật kết hợp bằng thuật toán ECLAT. Nhóm sử dụng thư viện PyECLAT để thực hiện thuật toán ECLAT trên dữ liệu đã được làm sạch và chuẩn hóa. Cụ thể, nó tìm kiếm các tập phổ biến có sự kết hợp của 1 đến 2 mặt hàng trong giao dịch. Kết quả bao gồm các tập phổ biến cùng với mức độ hỗ trợ của chúng. Các tham số như ngưỡng hỗ trợ (min_support) và kích thước tập (min_combination và max_combination) đã được đặt trước và ảnh hưởng đến quá trình khai phá.

```
# Khai phá các tập phổ biến có sự kết hợp của 1 đến 2 item
start_time = time.time()
min_combination = 1
max_combination = 2
rule_indices, rule_supports = eclat.fit(min_support=min_support,
                                        min_combination=min_combination,
                                        max_combination=max_combination,
                                           separator = ', ',
                                           verbose = True)
```

```
elapsed_time_eclat1 = time.time() - start_time
```

```
Combination 1 by 1
149it [00:02, 55.43it/s]
Combination 2 by 2
11026it [01:51, 98.45it/s]
```

Đầu tiên, chuyển đổi khóa thành frozenset và tạo DataFrame itemsets với mỗi khóa của tập phổ biến được chuyển từ định dạng chuỗi sang frozenset để có thể sử dụng như một khóa duy nhất không thay đổi được. Và mỗi frozenset được thêm vào danh sách itemsets cùng với giá trị hỗ trợ tương ứng. Cuối cùng, ta tạo DataFrame freq_itemsets với hai cột là 'support' và 'itemsets', trong đó 'support' chứa giá trị hỗ trợ và 'itemsets' chứa các frozenset đại diện cho các tập phổ biến.

Nhóm sử dụng association_rules từ thư viện mlxtend để khai phá các luật từ tập phổ biến đã tìm được trước đó với ngưỡng yêu cầu cho confidence là minConf = 0.03 và lift tối thiểu là 1. Các luật được lọc và sắp xếp dựa trên độ confidence, sau đó được lưu vào data frame rule_ECLAT.

```
# Chuyển các khóa của tập phổ biến thành frozenset và đứa vào một list itemsets
itemsets = []
for i in list(rule_supports.keys()):
  i = i.split(', ')
  i = (frozenset(i))
  itemsets.append(i)
# Ta được tạo được một DataFrame gồm các tập phổ biến và support
freq_itemsets = pd.DataFrame({'support': list(rule_supports.values()),
                      'itemsets': itemsets})
# Sử dụng association_rules để khai phá các luật từ tập phổ biến tìm được
# Ngưỡng yêu cầu là minConf = 0.03 và lift tối thiểu là 1
rule_ECLAT = association_rules(freq_itemsets, metric="confidence", min_threshold= min_confidence)
rule_ECLAT = rule_ECLAT[rule_ECLAT.lift > 1].sort_values('confidence', ascending=False)
```

```
rule_ECLAT = rule_ECLAT.reset_index(drop=True)

rule_ECLAT
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(semi-finished bread)	(whole milk)	0.009490	0.157923	0.001671	0.176056	1.114825	0.000172	1.022008	0.103985
1	(detergent)	(whole milk)	0.008621	0.157923	0.001403	0.162791	1.030824	0.000042	1.005814	0.030162
2	(ham)	(whole milk)	0.017109	0.157923	0.002740	0.160156	1.014142	0.000038	1.002659	0.014188
3	(processed cheese)	(rolls/buns)	0.010158	0.110005	0.001470	0.144737	1.315734	0.000353	1.040610	0.242430
4	(packaged fruit/vegetables)	(rolls/buns)	0.008488	0.110005	0.001203	0.141732	1.288421	0.000269	1.036967	0.225772
...	...	...	...	...	...	...	...	...	...	...
143	(canned beer)	(white bread)	0.046916	0.023993	0.001537	0.032764	1.365573	0.000411	1.009068	0.280885
144	(margarine)	(chicken)	0.032213	0.027869	0.001002	0.031120	1.116675	0.000105	1.003356	0.107962
145	(bottled beer)	(frozen vegetables)	0.045312	0.028002	0.001403	0.030973	1.106100	0.000135	1.003066	0.100475
146	(root vegetables)	(frozen vegetables)	0.069572	0.028002	0.002139	0.030740	1.097751	0.000190	1.002824	0.095705
147	(citrus fruit)	(frozen vegetables)	0.053131	0.028002	0.001604	0.030189	1.078074	0.000116	1.002254	0.076484

148 rows × 10 columns

Hình 18. Luật kết hợp thu được từ thuật toán ECLAT (pyECLAT)

3.2. Thuật toán ECLAT (mô phỏng)

Bước đầu tiên trong quy trình xử lý dữ liệu là thay thế giá trị NaN bằng giá trị None trong DataFrame. Việc này giúp chuẩn bị dữ liệu cho các bước xử lý và phân tích tiếp theo một cách đồng nhất.

```
# Thay thế các giá trị NaN thành các giá trị None
df2 = df_cleaned.replace(np.nan, None)
df2
```

Tiếp theo, để chuẩn bị dữ liệu cho thuật toán ECLAT, nhóm chuyển từ dạng dữ liệu ngang (Horizontal) thành dạng dữ liệu dọc (Vertical), với mỗi dòng đại diện cho một giao dịch và mỗi cột đại diện cho một sản phẩm trong giao dịch. Quá trình chuẩn hóa này giúp thuật toán ECLAT hoạt động hiệu quả và đảm bảo tính nhất quán khi áp dụng các phương pháp khai phá luật kết hợp.

```
# Chuyển Data Fomart từ dạng ngang (Horizontal) sang dạng dọc
(Vertical)
vertical_data={}


for i in range(df2.shape[0]):
  for j in range(10):
    if df2.iloc[i][j] is not None:
      if(vertical_data.get(df2.iloc[i][j])):
```

```
            vertical_data[df2.iloc[i][j]].append(str(i))
        else:
            vertical_data[df2.iloc[i][j]] = [str(i)]
```

Trước hết, một từ điển freq_sup được tạo để lưu trữ tần số hỗ trợ cho mỗi mặt hàng trong vertical_data. Quá trình này tính toán số lần mỗi mặt hàng xuất hiện trong tất cả các giao dịch, cung cấp cái nhìn về độ phổ biến của từng mặt hàng trong tập dữ liệu.

```
# Tần số hỗ trợ (Frequency Support)
freq_sup = {}
for item in vertical_data:
    freq_sup[item] = len(vertical_data[item]
```

Để tiện lợi trong việc xử lý dữ liệu, nhóm đã xây dựng hai hàm đơn giản: list_to_str và str_to_list. Hàm list_to_str chuyển đổi một danh sách thành một chuỗi được sắp xếp, trong khi str_to_list thực hiện chuyển đổi ngược lại.

```
# Xây dựng một số hàm đơn giản để code ngắn gọn hơn và dễ dàng xử lý
các tập phổ biến
#   + Chuyển list thành string
#   + Chuyển string thành list
def list_to_str(l):
    l_list = list(l)
    l_list.sort()
    return ','.join(l_list)


def str_to_list(s):
    return s.split(',')
```

Đến phần tìm kiếm các tập phổ biến nhóm thực hiện dựa trên phương pháp đề xuất bởi Zaki và Hsiao. Và quy trình sẽ này bắt đầu bằng việc tạo các bản sao của vertical_data để thực hiện các thao tác mà không làm thay đổi dữ liệu gốc. Sau đó, các vòng lặp được thực hiện để tìm kiếm các tập phổ biến.

Trong mỗi vòng lặp, danh sách item_list được tạo và các mặt hàng được kiểm tra kết hợp với nhau. Nếu kết hợp này đáp ứng các điều kiện, nó được thêm vào

new_temp_data. Các mặt hàng phổ biến được duyệt qua và các mặt hàng không đáp ứng ngưỡng hỗ trợ tối thiểu (min_support) được loại bỏ khỏi data_temp. Quá trình lặp được thực hiện cho đến khi không còn kết hợp nào mới được tìm thấy.

```
# Hàm tìm kiếm các tập phổ biến theo đề xuất của (Yin & Han, 2003)
start_time = time.time()


new_temp_data = vertical_data.copy()
data_temp = vertical_data.copy()
key_set = set()
item_list = []
frequent_item_set = {}
iteration = 0


for item in new_temp_data:
  if(len(new_temp_data[item]) < (min_support*len(df_cleaned))):
    del data_temp[item]


while True:
  iteration+=1


  temp_temp = new_temp_data.copy()
  item_list = list(temp_temp.keys())
  if len(item_list) == 0:
    break
  new_temp_data = {}


  for i in range(len(item_list)):
    for j in range(i+1, len(item_list)):
      if (iteration == 1) or
(len(list(set(str_to_list(item_list[i])).intersection(set(str_to_list(
item_list[j])))))==iteration-1):
        key_set = set()
        key_set.update(str_to_list(item_list[i]))
```

```
        key_set.update(str_to_list(item_list[j]))
        key_set_str = list_to_str(key_set)
        temp =
list(set(temp_temp[item_list[i]])&set(temp_temp[item_list[j]]))
        if len(temp) >= (min_support*len(df_cleaned)):
          new_temp_data[key_set_str] = temp
          freq_sup[key_set_str] = len(new_temp_data[key_set_str])
  data_temp.update(new_temp_data)

frequent_item_set = data_temp
elapsed_time_eclat2 = time.time() - start_time
```

Sau khi đã xác định các tập phổ biến từ đoạn mã trước, quá trình tìm kiếm luật kết hợp mạnh được thực hiện. Trong bước này sử dụng vòng lặp để duyệt qua các tập phổ biến và tạo ra các luật kết hợp từ chúng.

Đối với mỗi tập phổ biến, các phương án kết hợp của mặt hàng được xây dựng, và độ tin cậy của mỗi luật kết hợp được đánh giá dựa trên ngưỡng độ tin cậy tối thiểu (min_confidence). Các luật kết hợp mạnh, thoả mãn điều kiện độ tin cậy, được thêm vào danh sách ECLAT_association_rules. Thời gian thực thi của quy trình này được tính toán và ghi lại trong biến elapsed_time_eclat2.

```
# Tìm luật kết hợp mạnh từ tập phổ biến tìm được ở trên
start_time = time.time()

frequent_item_set_keys = list(frequent_item_set.keys())
ECLAT_association_rules = []

for frequent_item in frequent_item_set_keys:
  for i in range(1, len(str_to_list(frequent_item))):
    for item_combination in
itertools.combinations(str_to_list(frequent_item), i):
      str_item_combination_A = list_to_str(list(item_combination))
      str_item_combination_B =
list_to_str(list(set(str_to_list(frequent_item))-
```

```
set(list(item_combination)))
      conf = freq_sup[frequent_item]/freq_sup[str_item_combination_A]
      if(conf >= min_confidence):
        ECLAT_association_rules.append((str_item_combination_A,
str_item_combination_B,freq_sup[str_item_combination_A]/len(df),
                                    freq_sup[str_item_combination_B]/len(d
f),len(frequent_item_set[frequent_item])/len(df),conf))


elapsed_time_eclat2 = elapsed_time_eclat2 + time.time() - start_time
```

	antecedents	consequents	antecedent_support	consequent_support	support	confidence	lift
0	semi-finished bread	whole milk	0.009490	0.157923	0.001671	0.176056	1.114825
1	detergent	whole milk	0.008621	0.157923	0.001403	0.162791	1.030824
2	ham	whole milk	0.017109	0.157923	0.002740	0.160156	1.014142
3	pastry	sausage	0.051728	0.060349	0.003208	0.062016	1.027617
4	sausage	pastry	0.060349	0.051728	0.003208	0.053156	1.027617
...	...	...	...	...	...	...	...
482	other vegetables,soda	nan,rolls/buns	0.009691	0.109938	0.001136	0.117241	1.066433
483	rolls/buns,soda	nan,other vegetables	0.008087	0.122034	0.001136	0.140496	1.151281
484	nan,other vegetables,rolls/buns	soda	0.010493	0.097106	0.001136	0.108280	1.115071
485	nan,other vegetables,soda	rolls/buns	0.009691	0.110005	0.001136	0.117241	1.065785
486	nan,rolls/buns,soda	other vegetables	0.008087	0.122101	0.001136	0.140496	1.150651

487 rows × 7 columns

Hình 19. Luật kết hợp thu được từ thuật toán ECLAT mô phỏng

Sau khi đã xác định và đánh giá các luật kết hợp mạnh từ tập phổ biến, quy trình tiếp theo là chuyển đổi các luật này thành một DataFrame để dễ dàng quan sát và phân tích.

Các luật được lọc dựa trên điều kiện lift > 1 để chỉ giữ lại những luật có ảnh hưởng tích cực. Kết quả cuối cùng là DataFrame rule_ECLAT_2 hiển thị các thông tin chi tiết về các luật kết hợp mạnh, cung cấp cơ sở để hiểu rõ mối quan hệ giữa các mặt hàng trong dữ liệu. Thời gian thực thi của quy trình này được tính toán và ghi lại trong biến elapsed_time eclat2.

3.3. Thuật toán Apriori

Trong phần này, nhóm thực hiện việc khai phá tập kết hợp từ dữ liệu giao dịch bằng cách sử dụng thuật toán Apriori.

Đầu tiên, chuyển đổi dữ liệu thành danh sách giao dịch (Transaction List). Mỗi giao dịch trong dataframe df được biến đổi thành một danh sách (itemset) chứa các mặt

hàng mà khách hàng đã mua. Tất cả các giao dịch này được tập hợp thành một danh sách lớn có tên là transaction.

Trong vòng lặp mỗi hàng của DataFrame df được duyệt qua để tạo danh sách các giao dịch (transaction). Với mỗi hàng, một danh sách tạm thời (temp) được tạo ra để chứa tên các mặt hàng có trong giao dịch đó. Nếu giá trị không phải là None, nó được chuyển thành chuỗi và thêm vào danh sách tạm thời.

```
# Đưa các giao dịch vào list
# Mỗi list là một itemset chứa các item là mặt hàng mà khách hàng mua
transaction = []


for _, row in df2.iterrows():
  temp = []
  for column, value in row.iteritems():
    if value is not None:
      i = f"{value}"
      temp.append(i)
  transaction.append(temp)
```

Sử dụng TransactionEncoder từ thư viện mlxtend để chuyển danh sách giao dịch thành một mảng one-hot encoder, trong đó mỗi hàng biểu diễn một giao dịch và mỗi cột biểu diễn một mặt hàng. Nếu mặt hàng có trong giao dịch, giá trị tương ứng là True, ngược lại là False.

```
# TransactionEncoder để chuyển dữ liệu giao dịch thành mảng one-hot
encoder boolean
te = TransactionEncoder()
te_ary = te.fit(transaction).transform(transaction)
df3 = pd.DataFrame(te_ary, columns=te.columns_)
df3
```

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...
0	False	False	False	False	False	False	False	False	False	False	...
1	False	False	False	False	False	False	False	False	False	False	...
2	False	False	False	False	False	False	False	False	False	False	...
3	False	False	False	False	False	False	False	False	False	False	...
4	False	False	False	False	False	False	False	False	False	False	...
...	...	...	...	...	...	...	...	...	...	...	...
14958	False	False	False	False	False	False	False	False	False	False	...
14959	False	False	False	False	False	False	False	False	False	False	...
14960	False	False	False	False	False	False	False	False	False	False	...
14961	False	False	False	False	False	False	False	False	False	False	...
14962	False	False	False	False	False	False	False	False	False	False	...

14963 rows × 168 columns

Hình 20. Tập dữ liệu chuyển dạng theo yêu cầu của thuật toán Apriori

Trong bước này, thuật toán Apriori được áp dụng để tìm các tập phổ biến với mức hỗ trợ tối thiểu là min_support. Kết quả là một DataFrame df_apriori chứa các tập phổ biến và mức hỗ trợ của chúng.

```
# Sử dụng thuật toán Apriori (thư viện mlxtend) để khai pháp tập kết hợp
start_time = time.time()

df_apriori = apriori(df3, min_support=min_support, use_colnames=True)
df_apriori.itemsets

elapsed_time_apriori = time.time() - start_time
```

Sau khi có danh sách các tập phổ biến từ Apriori, đoạn mã sử dụng hàm association_rules để khai phá các luật kết hợp từ các tập phổ biến này. Các luật được lọc dựa trên các tiêu chí như độ tin cậy (confidence) và lift. Trong trường hợp này, ngưỡng độ tin cậy được đặt là min_confidence và ngưỡng lift là 1.

Kết quả cuối cùng là DataFrame rule_APRIORI, trong đó mỗi hàng đại diện cho một luật kết hợp, bao gồm các thông tin như tập đầu (antecedents), tập kết (consequents), độ tin cậy, hỗ trợ, lift, và các thông số khác.

```
# Sử dụng association_rules để khai phá các luật từ tập phổ biến tìm được
# Ngưỡng yêu cầu là minConf = 0.03 và lift tối thiểu là 1
```

```
start_time = time.time()


rule_APRIORI = association_rules(df_apriori, metric="confidence",
min_threshold= min_confidence)
rule_APRIORI = rule_APRIORI[rule_APRIORI.lift > 1]
rule_APRIORI = rule_APRIORI.reset_index(drop=True)


elapsed_time_apriori = elapsed_time_apriori + time.time() - start_time
rule_APRIORI
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(Instant food products)	(nan)	0.004010	0.999933	0.004010	1.000000	1.000067	2.679871e-07	inf	0.000067
1	(UHT-milk)	(nan)	0.021386	0.999933	0.021386	1.000000	1.000067	1.429265e-06	inf	0.000068
2	(UHT-milk)	(tropical fruit)	0.021386	0.067767	0.001537	0.071875	1.060617	8.785064e-05	1.004426	0.058402
3	(abrasive cleaner)	(nan)	0.001470	0.999933	0.001470	1.000000	1.000067	9.826194e-08	inf	0.000067
4	(artif. sweetener)	(nan)	0.001938	0.999933	0.001938	1.000000	1.000067	1.295271e-07	inf	0.000067
...	...	...	...	...	...	...	...	...	...	...
1213	(yogurt, nan, whole milk)	(sausage)	0.011161	0.060349	0.001470	0.131737	2.182917	7.967480e-04	1.082219	0.548014
1214	(nan, sausage, whole milk)	(yogurt)	0.008955	0.085879	0.001470	0.164179	1.911760	7.012151e-04	1.093681	0.481231
1215	(yogurt, sausage)	(nan, whole milk)	0.005748	0.157923	0.001470	0.255814	1.619866	5.626300e-04	1.131541	0.384877
1216	(yogurt, whole milk)	(nan, sausage)	0.011161	0.060282	0.001470	0.131737	2.185337	7.974939e-04	1.082296	0.548527
1217	(sausage, whole milk)	(yogurt, nan)	0.008955	0.085812	0.001470	0.164179	1.913249	7.018136e-04	1.093761	0.481642

1218 rows × 10 columns

Hình 21. Luật kết hợp thu được từ thuật toán Apriori

4. KẾT QUẢ MÔ HÌNH

Sau khi thực hiện chạy ba mô hình khai phá luật kết hợp - ECLAT, ECLAT mô phỏng và Apriori, chúng ta có được cái nhìn toàn diện về các mối liên kết và quy luật trong tập dữ liệu. Trong phần này, nhóm sẽ tổng hợp và so sánh kết quả từ ba mô hình. Qua đó, chúng ta có thể đánh giá hiệu suất của từng thuật toán và đưa ra những kết luận.

Sự hiểu biết sâu rộng về cách mà mỗi thuật toán xử lý và trả về thông tin sẽ giúp nâng cao chất lượng và hiệu suất của quy trình khai phá luật kết hợp trong các tình huống thực tế. Nhóm sẽ tiến hành so sánh 3 thuật toán dựa trên các tiêu chí về thời gian thực hiện và các chỉ số liên quan như: Support, Confidence, Lift.

4.1. Thời gian thực hiện

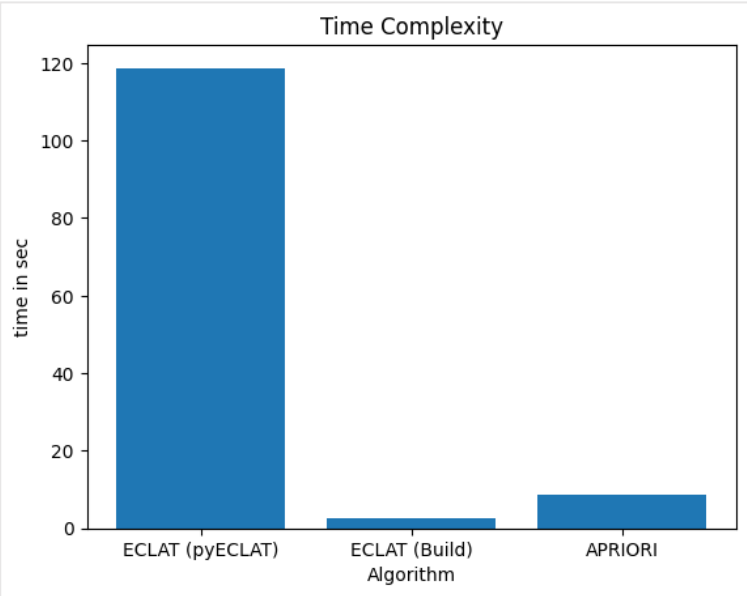
Vẽ biểu đồ thanh để quan sát thời gian chạy của 3 thuật toán, sau đó tiến hành so sánh.

```
# Sử dụng association_rules để khai phá các luật từ tập phổ biến tìm
được
```

```
# Ngưỡng yêu cầu là minConf = 0.03 và lift tối thiểu là 1
start_time = time.time()

rule_APRIORI = association_rules(df_apriori, metric="confidence",
min_threshold= min_confidence)
rule_APRIORI = rule_APRIORI[rule_APRIORI.lift > 1]
rule_APRIORI = rule_APRIORI.reset_index(drop=True)

elapsed_time_apriori = elapsed_time_apriori + time.time() - start_time
rule_APRIORI
```



Hình 22. Thời gian thực hiện của ba thuật toán (pyECLAT, ECLAT, Apriori)

Từ biểu đồ, có thể thấy rằng thuật toán ECLAT (pyECLAT) là thuật toán phức tạp về thời gian nhất. Thuật toán Apriori là thuật toán phức tạp thứ hai. Thuật toán ECLAT mô phỏng đạt hiệu quả về thời gian cao nhất. Đây là lợi thế của thuật toán này.

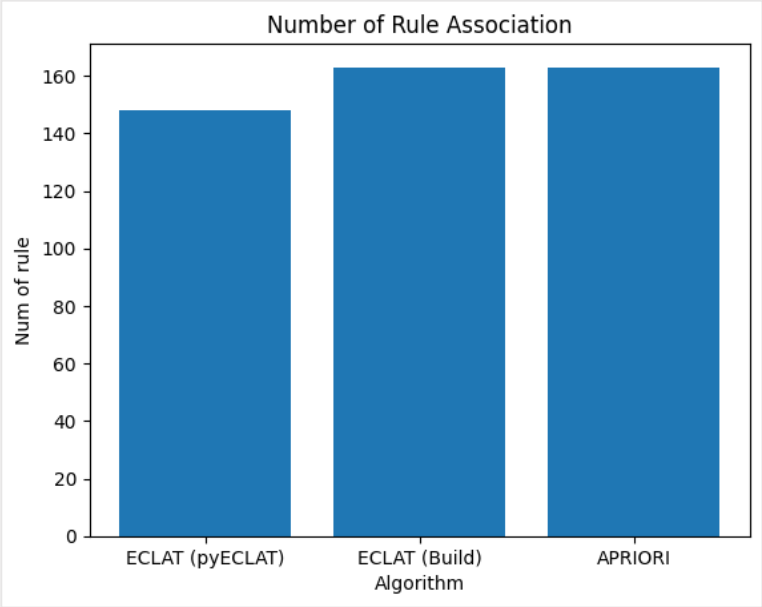
4.2. Số lượng luật kết hợp

Vẽ biểu đồ thanh để quan sát số lượng luật kết hợp được tạo từ 3 thuật toán, sau đó tiến hành so sánh.

```
x = ["ECLAT (pyECLAT)", "ECLAT (Build)", "APRIORI"]
y = [len(rule_ECLAT), len(rule_ECLAT_2), len(rule_APRIORI)]

plt.bar(x, y)
```

```
plt.xlabel("Algorithm")
plt.ylabel("Num of rule")
plt.title("Number of Rule Association")
plt.show()
```



Hình 23. Số lượng luật kết hợp được sinh ra từ ba thuật toán

Từ biểu đồ, có thể thấy rằng thuật toán ECLAT mô phỏng và APRIORI tạo ra số luật kết hợp bằng nhau, còn thuật toán ECLAT (pyECLAT) tạo ra số luật kết hợp ít hơn. Nguyên nhân là do 2 thuật toán ECLAT mô phỏng và APRIORI được thiết kế để tìm ra tất cả các tập phổ biến thỏa mãn minSup và minConf, còn thuật toán ECLAT (pyECLAT) chỉ các tập phổ biến có sự kết hợp của 1 đến 2 mặt hàng trong giao dịch, nên khi tạo luật kết hợp từ các tập phổ biến, số lượng luật của ECLAT (pyECLAT) ít hơn 2 thuật toán còn lại.

4.3. So sánh các chỉ số của ba thuật toán

A. So sánh support của 3 thuật toán

Vẽ biểu đồ đường để quan sát độ hỗ trợ của các luật kết hợp được tạo từ 3 thuật toán, sau đó tiến hành so sánh. Độ hỗ trợ của từng luật kết hợp trong thuật toán được sắp xếp theo thứ tự tăng dần.

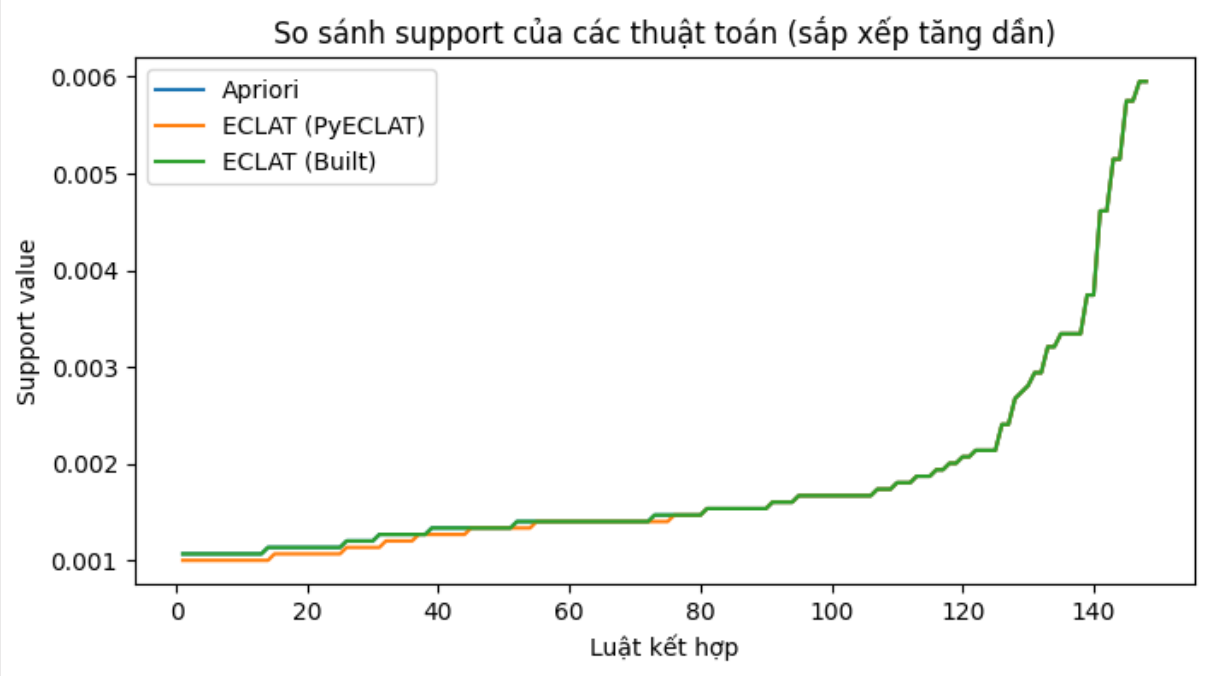
```
k = min(len(rule_APRIORI), len(rule_ECLAT), len(rule_ECLAT_2))
x =  [i for i in range(1, k+1)]
y1 = rule_APRIORI["support"].sort_values().tail(k)
y2 = rule_ECLAT["support"].sort_values().tail(k)
```

```
y3 = rule_ECLAT_2["support"].sort_values().tail(k)

plt.figure(figsize=(8,4))
plt.plot(x, y1, label='Apriori')
plt.plot(x, y2, label='ECLAT (PyECLAT)')
plt.plot(x, y3, label='ECLAT (Built)')

plt.xlabel('Luật kết hợp')
plt.ylabel('Support value')
plt.title('So sánh support của các thuật toán (sắp xếp tăng dần)')

plt.legend()
plt.show()
```



Hình 24. So sánh chỉ số support của các luật sinh ra từ ba thuật toán

Từ biểu đồ, có thể thấy rằng 2 thuật toán APRIORI và ECLAT tạo ra các luật kết hợp giống nhau nên đường biểu diễn độ hỗ trợ của 2 thuật toán này trùng nhau. Còn với độ hỗ trợ dưới 0.2%, các luật do thuật toán ECLAT (PyECLAT) tạo ra có độ hỗ trợ thấp hơn các luật do 2 thuật toán còn lại tạo ra, từ khoảng 0.2% trở lên thì đường biểu diễn độ hỗ trợ của 3 thuật toán này trùng nhau.

B. So sánh lift của 3 thuật toán

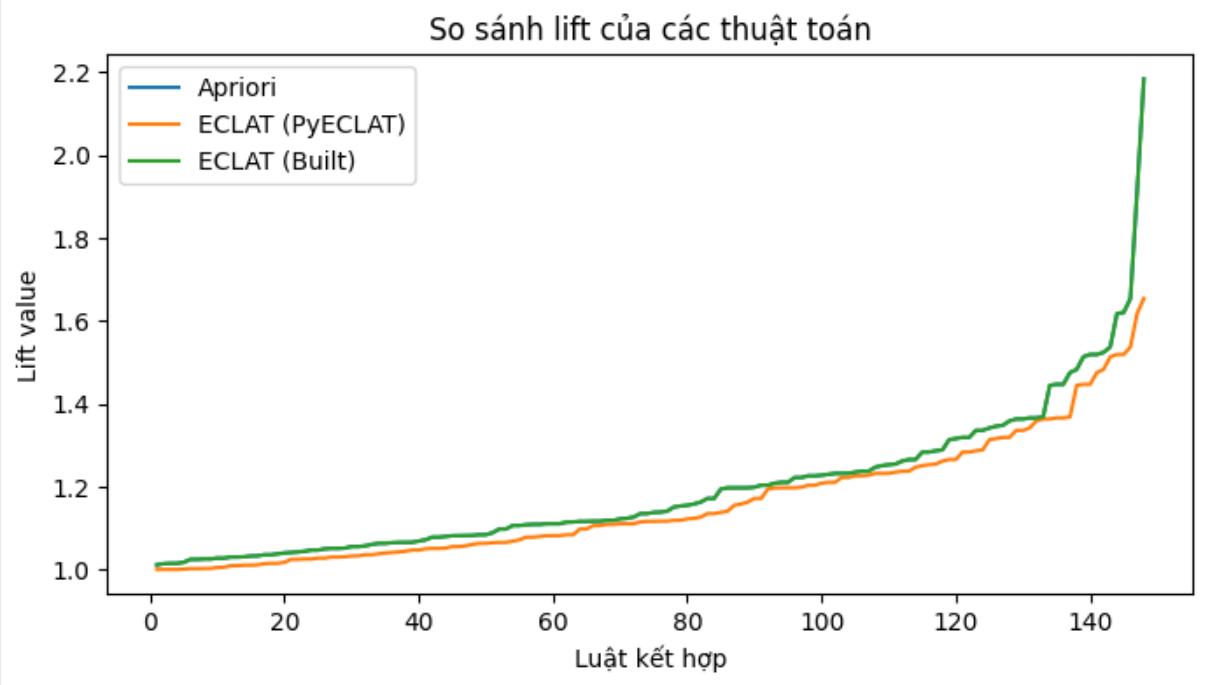
Vẽ biểu đồ đường để quan sát độ liên kết của 2 mục của các luật kết hợp được tạo từ 3 thuật toán, sau đó tiến hành so sánh. Độ liên kết của từng luật kết hợp trong thuật toán được sắp xếp theo thứ tự tăng dần.

```
k = min(len(rule_APRIORI), len(rule_ECLAT), len(rule_ECLAT_2))
x =  [i for i in range(1, k+1)]
y1 = rule_APRIORI["lift"].sort_values().tail(k)
y2 = rule_ECLAT["lift"].sort_values().tail(k)
y3 = rule_ECLAT_2["lift"].sort_values().tail(k)

plt.figure(figsize=(8,4))
plt.plot(x, y1, label='Apriori')
plt.plot(x, y2, label='ECLAT (PyECLAT)')
plt.plot(x, y3, label='ECLAT (Built)')

plt.xlabel('Luật kết hợp')
plt.ylabel('Lift value')
plt.title('So sánh lift của các thuật toán')

plt.legend()
plt.show()
```



Hình 25. So sánh chỉ số lift của các luật sinh ra từ ba thuật toán

Tương tự với biểu đồ trên, vì 2 thuật toán APRIORI và ECLAT tạo ra các luật kết hợp giống nhau nên đường biểu diễn độ liên kết giữa 2 mục của 2 thuật toán này cũng trùng nhau. Còn với thuật toán ECLAT (PyECLAT) có độ liên kết giữa 2 mục thấp hơn so với 2 thuật toán còn lại, điều này cho thấy mức độ liên kết giữa hai mục của thuật toán này thấp hơn. Vậy liên kết giữa 2 mục của 2 thuật toán ECLAT mô phỏng và

APRIORI có ý nghĩa hơn.

C. So sánh lift của 3 thuật toán

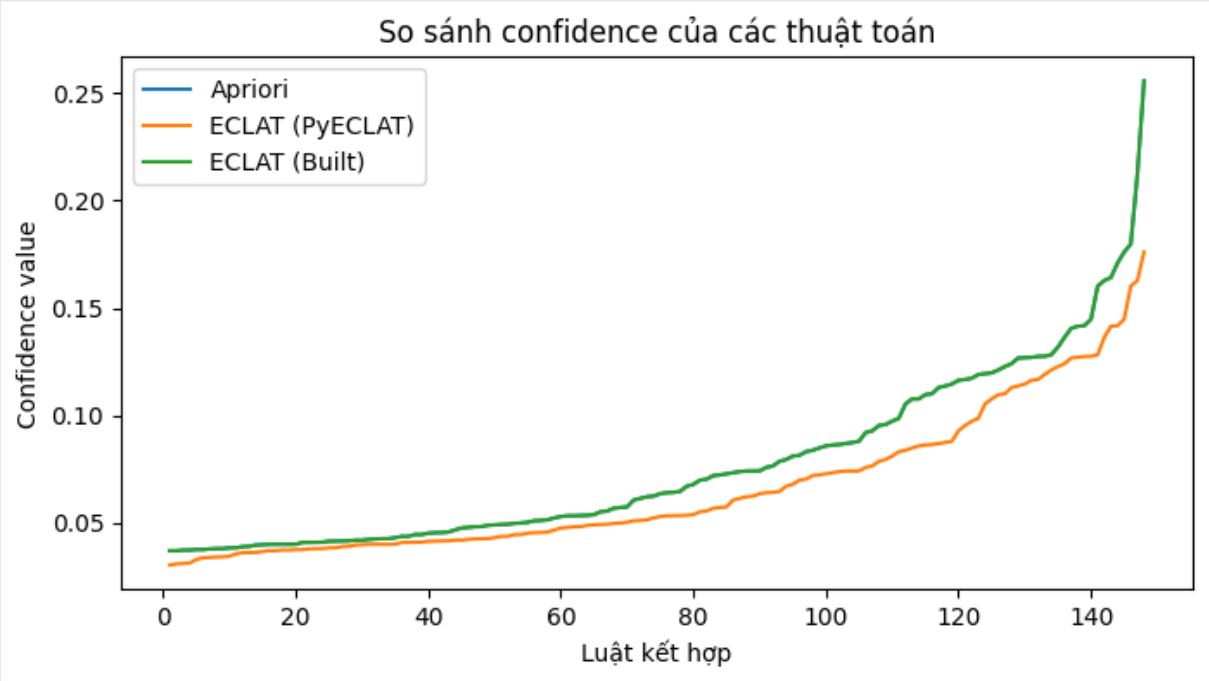
Vẽ biểu đồ đường để quan sát độ tin cậy của các luật kết hợp được tạo từ 3 thuật toán, sau đó tiến hành so sánh. Độ tin cậy của từng luật kết hợp trong thuật toán được sắp xếp theo thứ tự tăng dần.

```
k = min(len(rule_APRIORI), len(rule_ECLAT), len(rule_ECLAT_2))
x =  [i for i in range(1, k+1)]
y1 = rule_APRIORI["confidence"].sort_values().tail(k)
y2 = rule_ECLAT["confidence"].sort_values().tail(k)
y3 = rule_ECLAT_2["confidence"].sort_values().tail(k)

plt.figure(figsize=(8,4))
plt.plot(x, y1, label='Apriori')
plt.plot(x, y2, label='ECLAT (PyECLAT)')
plt.plot(x, y3, label='ECLAT (Built)')

plt.xlabel('Luật kết hợp')
plt.ylabel('Confidence value')
plt.title('So sánh confidence của các thuật toán')

plt.legend()
plt.show()
```



Hình 26. So sánh chỉ số confidence của các luật sinh ra từ ba thuật toán

Tương tự với 2 biểu đồ trên, đường biểu diễn độ tin cậy của 2 thuật toán APRIORI

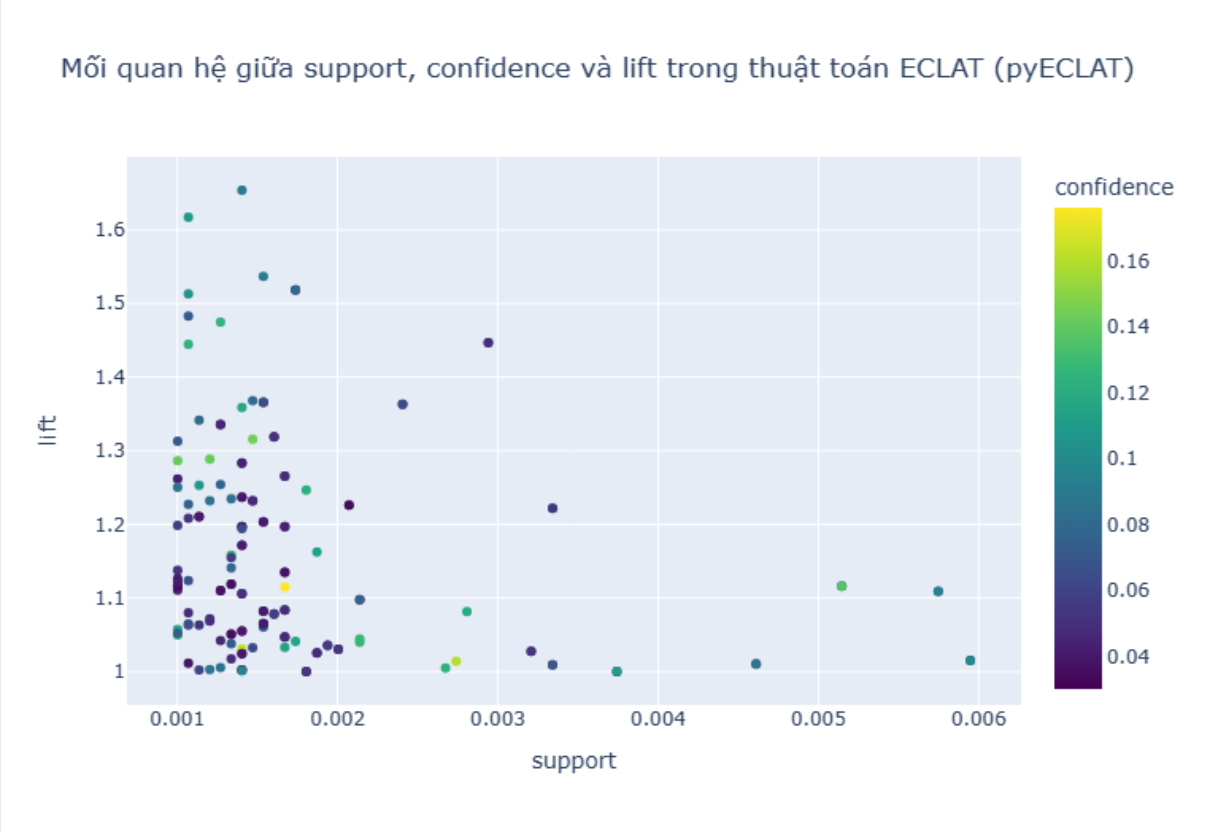
và ECLAT cũng trùng nhau. Còn với thuật toán ECLAT (PyECLAT) có độ tin cậy thấp hơn so với 2 thuật toán còn lại, điều này luật kết hợp của 2 thuật toán ECLAT mô phỏng và APRIORI chắc chắn hơn.

4.4. Mối quan hệ giữa 3 hàm đánh giá của từng thuật toán

A. Mối quan hệ giữa 3 hàm đánh giá của thuật toán ECLAT (PyECLAT)

Vẽ biểu đồ scatter plot để quan sát mối quan hệ giữa support, confidence và lift của các luật kết hợp được tạo ra từ thuật toán ECLAT (PyECLAT).

```
fig = px.scatter(rule_ECLAT, x = 'support', y = 'lift',
                 color = 'confidence',
color_continuous_scale=px.colors.sequential.Viridis,
                 title = 'Mối quan hệ giữa support, confidence và lift
trong thuật toán ECLAT (pyECLAT)')
fig.show()
```



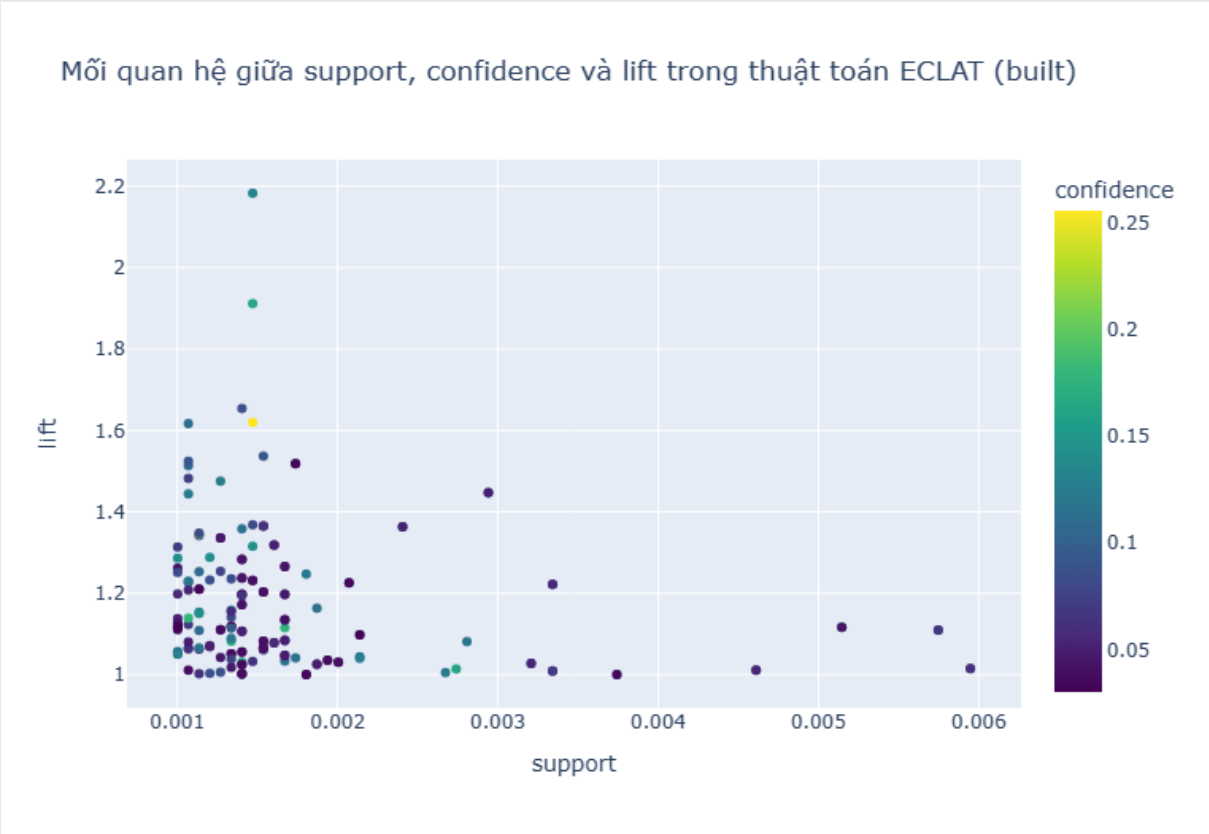
Hình 27. Mối quan hệ giữa support, confidence và lift của các luật kết hợp sinh ra bởi thuật toán ECLAT (PYECLAT)

Từ biểu đồ, có thể thấy rằng các luật kết hợp được tao có độ hỗ trợ khoảng từ 0.1% đến 0.2% chiếm phần lớn, và những luật này có độ tin cậy nằm trong khoảng 3% đến 10%. Có khá ít luật nào có độ hỗ trợ cao hơn 0.4% hay độ liên kết giữa 2 mục cao hơn 1.4 hay độ tin cậy cao hơn 10%.

B. Mối quan hệ giữa 3 hàm đánh giá của thuật toán ECLAT mô phỏng

Vẽ biểu đồ scatter plot để quan sát mối quan hệ giữa support, confidence và lift của các luật kết hợp được tạo ra từ thuật toán ECLAT mô phỏng.

```
fig = px.scatter(rule_ECLAT_2, x = 'support', y = 'lift',
                 color = 'confidence',
color_continuous_scale=px.colors.sequential.Viridis,
                 title = 'Mối quan hệ giữa support, confidence và lift
trong thuật toán ECLAT (built)')
fig.show()
```



Hình 28. Mối quan hệ giữa support, confidence và lift lift của các luật kết hợp sinh ra bởi thuật toán ECLAT mô phỏng

Từ biểu đồ, có thể thấy rằng các luật kết hợp được tao có độ hỗ trợ tập trung khoảng từ 0.1% đến 0.2%, và hầu hết những luật kết hợp có độ hỗ trợ càng thấp thì độ tin cậy và độ liên kết của 2 mục cũng càng thấp. những luật này có độ tin cậy nằm trong khoảng 3% đến 10% và độ liên kết giữa 2 mục từ 1 đến 1.4. Có rải rác các luật có độ hỗ trợ cao hơn 0.2% hay độ liên kết giữa 2 mục cao hơn 1.4 hay độ tin cậy cao hơn 10%. Không có luật kết hợp nào có chỉ số của 3 độ đo đều cao.

Biểu đồ scatter plot thể hiện mối quan hệ giữa support, confidence và lift của các luật kết hợp được tạo ra từ thuật toán APRIORI giống với thuật toán ECLAT mô phỏng.

CHƯƠNG 5: KẾT LUẬN VÀ ĐÁNH GIÁ

1. KẾT LUẬN

Trong bài toán Market Basket Analysis, việc áp dụng thuật toán ECLAT mô phỏng để khai thác luật kết hợp mang lại nhiều lợi ích đáng kể. Thuật toán này không chỉ nhanh chóng xử lý dữ liệu mà còn tạo ra các luật kết hợp với hiệu suất cao hơn sao với 2 thuật toán ECLAT trong thư viện pyECLAT và thuật toán Apriori. Điều này có thể giúp cung cấp thông tin hữu ích về các mối quan hệ giữa các mặt hàng trong giỏ hàng, đồng thời hỗ trợ quản lý cửa hàng và siêu thị trong việc tối ưu hóa bố trí sản phẩm và tăng cường doanh thu.

2. ƯU ĐIỂM

- **Hiệu quả trong thời gian và tốc độ xử lý.**

  Thuật toán ECLAT đã chứng minh được khả năng xử lý tốt với dữ liệu lớn và thưa thớt. Sự tối ưu hóa trong quá trình xử lý giúp giảm độ phức tạp thời gian so với một số thuật toán khác, đồng thời nhanh chóng tạo ra các luật kết hợp quan trọng từ dữ liệu.

- **Độ chính xác và hiệu suất.**

ECLAT không chỉ nổi bật với hiệu suất cao mà còn đảm bảo độ chính xác trong việc phát hiện mối quan hệ giữa các sản phẩm. Các luật kết hợp được tạo ra từ thuật toán này có thể tin cậy và chính xác, giúp người quản lý đưa ra quyết định dựa trên thông tin đáng tin cậy.

- **Hỗ trợ quản lý và tối ưu hóa doanh thu.**

Thành công của ECLAT mang lại lợi ích cho quản lý cửa hàng và siêu thị bằng cách cung cấp thông tin chi tiết về mối quan hệ giữa các mặt hàng. Quản lý có thể tận dụng thông tin này để tổ chức bố trí cửa hàng một cách hiệu quả, đặt các sản phẩm liên quan gần nhau để tối đa hóa khả năng mua sắm và tăng cường doanh thu.

- **Tích hợp thông tin hữu ích.**

Các luật kết hợp do ECLAT tạo ra không chỉ giúp tối ưu hóa doanh thu mà còn cung cấp thông tin chi tiết về sự tương tác giữa các sản phẩm. Điều này có thể hỗ trợ trong việc đưa ra các chiến lược quảng cáo, chương trình khuyến mãi, và cả chiến lược định giá.

Như vậy, sự kết hợp giữa hiệu suất thời gian, độ chính xác, và khả năng hỗ trợ quyết định chiến lược kinh doanh làm cho ECLAT trở thành một công cụ quan trọng trong bài

toán Market Basket Analysis. Thông qua việc phân tích và áp dụng các luật kết hợp, người quản lý có thể thúc đẩy hiệu quả kinh doanh và tối ưu hóa trải nghiệm mua sắm cho khách hàng.

3. HẠN CHẾ

Bên cạnh những ưu điểm trên, thuật toán ECLAT cho bài toán Market Basket Analysis cũng có những hạn chế có thể ảnh hưởng tới doanh nghiệp/ người bán hàng.

- **Yêu cầu bước tiền xử lý**

Một trong nhược điểm chính của thuật toán ECLAT là cần một bước tiền xử lý để phân loại hàng hóa thành các lớp tương đương. Việc này không chỉ tăng độ phức tạp của quá trình triển khai mà còn đòi hỏi kiến thức chuyên sâu về dữ liệu và ngữ cảnh cụ thể của tập dữ liệu. Bước tiền xử lý này có thể làm cho quá trình khai thác luật kết hợp trở nên phức tạp và đòi hỏi nhiều công sức tính toán.

- **Khả năng không hiệu quả với dữ liệu hiếm và giao dịch không thường xuyên**

ECLAT có thể không hiệu quả khi đối mặt với các tập dữ liệu chứa nhiều hàng hóa hiếm hoặc giao dịch không thường xuyên. Trong các trường hợp này, với sự hiếm có của các mục, quá trình xác định các mẫu có thể trở nên phức tạp và đòi hỏi nhiều tài nguyên tính toán. Điều này làm giảm khả năng xác định các quy luật kết hợp và làm mất đi tính ứng dụng của ECLAT trong những tình huống này.

Những hạn chế trên có thể dẫn đến việc doanh nghiệp không thể tận dụng hết tiềm năng của dữ liệu và không có được những chiến lược bán hàng hiệu quả. Điều quan trọng là lựa chọn thuật toán phù hợp với đặc tính của dữ liệu và mục tiêu kinh doanh để đảm bảo rằng quá trình Market Basket Analysis mang lại giá trị tốt nhất cho doanh nghiệp.

4. HƯỚNG PHÁT TRIỂN

- **Kết Hợp Với Các Phương Pháp Khác**

So sánh hiệu suất của thuật toán ECLAT với các phương pháp khác như Apriori, FP-growth để xác định phương pháp nào mang lại kết quả tốt nhất. Ta có thể thực hiện thử nghiệm đối chiếu trực tiếp giữa ECLAT và các thuật toán khác trên cùng bộ dữ liệu để đánh giá tính hiệu quả và tốc độ thực thi.

- **Dự Đoán Xu Hướng Tương Lai**

Phát triển mô hình học máy dựa trên quy luật kết hợp để dự đoán xu hướng mua sắm tương lai của khách hàng. Ta có thể sử dụng thuật toán học máy như Random Forest

hoặc Support Vector Machines để xây dựng mô hình dự đoán dựa trên các quy luật kết hợp và dữ liệu lịch sử.

- **Tối Ưu Hóa Đa Kênh Bán Hàng**

Phân tích quy luật kết hợp trong từng kênh và kết hợp chúng để tạo ra chiến lược toàn diện, đồng nhất cho tất cả các kênh bán hàng. Để có thể tối ưu hóa chiến lược bán hàng trên nhiều kênh, bao gồm cả cửa hàng trực tuyến và offline.

- **Xây Dựng Hệ Thống Tư Vấn**

Phát triển hệ thống tư vấn dựa trên quy luật kết hợp để cung cấp gợi ý sản phẩm thông minh cho khách hàng. Sử dụng công nghệ AI để xây dựng một hệ thống tự động tư vấn dựa trên quy luật kết hợp và lịch sử mua sắm của khách hàng.

- **Nghiên Cứu Về Ảnh Hưởng Của Chiến Lược Giá**

Điều tra tác động của chiến lược giá đối với mối quan hệ giữa các sản phẩm trong giỏ hàng bằng cách phân tích quy luật kết hợp trong các phạm vi giá cả khác nhau để xem xét sự biến động của mối liên kết dựa trên chiến lược giá.

TÀI LIỆU THAM KHẢO

Ali, M. (2023, January). Association Rule Mining in Python Tutorial. Datacamp.com. https://www.datacamp.com/tutorial/association-rule-mining-python?fbclid=IwAR3_cszF8zoCp59cbdUGk9HyluqPLY0lGxrNtQE69tX6-1Nfsq2f4xvrPpE

Fournier-Viger, P. (2019). 数据挖掘 Introduction to Data Mining. Slideplayer.com. https://slideplayer.com/slide/17630911/

Hlaing, M. M. (2019). ECLAT based market basket analysis for electronic showroom. https://www.ijarnd.com/manuscripts/v4i7/V4I7-1140.pdf

Mathur, V. (2022, September 26). Association rule mining: Importance and steps. Analyticssteps.com. https://www.analyticssteps.com/blogs/association-rule-mining-importance-and-steps

Saini, H. (2022, May 10). Market basket analysis: An overview. Analyticssteps.com. https://www.analyticssteps.com/blogs/market-basket-analysis-overview

Tế, N. A. (2023). Data Mining.2023.Ch4 - Luật kết hợp.

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). New Algorithms for Fast Discovery of Association Rules.

Zaki, Mohammed. (2000). Scalable algorithms for association mining. IEEE Trans Knowl Data Eng. Knowledge and Data Engineering, IEEE Transactions on. 12. 372 - 390. 10.1109/69.846291.

Sinthuja, M., Aruna, P., & Puviarasan, N. (2017). EXPERIMENTAL EVALUATION OF APRIORI AND EQUIVALENCE CLASS CLUSTERING AND BOTTOM-UP LATTICE TRAVERSAL (ECLAT) ALGORITHMS.

PHÂN CÔNG CÔNG VIỆC

STT	Họ và tên	MSSV	Công việc	Đánh giá
1	Trương Thanh Phong	31211027662	- Chương 4: Áp dụng thuật toán ECLAT vào Market Basket Analysis - Làm slide	100%
2	Đỗ Quang Thiên Phú	31211024191	- Chương 3: Thuật toán ECLAT - Tổng hợp word	100%
3	Lê Trần Khánh Phú	31211024087	- Chương 4: Áp dụng thuật toán ECLAT vào Market Basket Analysis - Làm slide	100%
4	Phạm Minh Phước	31211027663	- Chương 1: Mở đầu - Chương 5: Kết luận và đánh giá - Tổng hợp word	100%
5	Dương Mỹ Quỳnh	31211027666	- Chương 2: Market Basket Analysis - Tổng hợp word	100%