ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH
UEH UNIVERSITY
Trường Công nghệ và Thiết kế
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

BÁO CÁO CUỐI KỲ

ỨNG DỤNG THUẬT TOÁN ECLAT ĐỂ GIẢI QUYẾT BÀI TOÁN MARKET BASKET ANALYSIS

Môn học: Khai Phá Dữ Liệu

Giảng viên hướng dẫn: TS. Nguyễn An Tế
Mã lớp học phần: 24C1INF50904301
Nhóm sinh viên thực hiện:

- Trầm Thái Tú - 31221022394
- Nguyễn Thành Vinh - 31221025662
- Huỳnh Ngọc Khánh Vy - 88214020004
- Nguyễn Vân Phi Yến - 31221021785

Hồ Chí Minh, ngày 12 tháng 12 năm 2024

LỜI CẢM ƠN

Để có thể hoàn thiện bài luận báo cáo cuối kỳ môn học *Khai phá dữ liệu* với đề tài *"Ứng dụng thuật toán ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) để giải quyết bài toán Market Basket Analysis"*, nhóm chúng em xin gửi lời tri ân sâu sắc đến Thầy TS. Nguyễn An Tế, giảng viên hướng dẫn trực thuộc khoa Công nghệ Thông tin Kinh doanh của Đại học Kinh tế Thành phố Hồ Chí Minh.

Nhóm em chân thành cảm ơn Thầy vì sự tận tâm trong giảng dạy, cũng như những kiến thức quý báu mà Thầy đã truyền đạt trong môn học Khai phá dữ liệu. Không chỉ giới hạn trong khuôn khổ bài giảng, Thầy còn khuyến khích chúng em tìm hiểu và nghiên cứu thêm các thuật toán mới như ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal), đồng thời mở rộng góc nhìn và ứng dụng vào bài toán Market Basket Analysis thực tiễn.

Trong quá trình thực hiện bài tiểu luận, nhóm chúng em đã cố gắng vận dụng những kiến thức đã được học và nghiên cứu thêm nhiều tài liệu để hoàn thiện bài tiểu luận này. Tuy nhiên, do kiến thức còn hạn chế và thiếu kinh nghiệm thực tiễn nên nội dung bài tiểu luận khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp quý báu từ Thầy để nhóm có thể hoàn thiện bài làm hơn.

Nhóm chúng em xin chân thành cảm ơn Thầy vì sự đồng hành và hỗ trợ trong suốt quá trình học tập và nghiên cứu.

Trân trọng,
Nhóm thực hiện

PHÂN CÔNG CÔNG VIỆC

Nhiệm vụ	Thành viên	Mức độ hoàn thành
- Phụ trách nội dung: Chương 4, 3 - Áp dụng ECLAT xử lý bộ dữ liệu	Trầm Thái Tú	100%
- Phụ trách nội dung: Chương 5, 2 - Tìm hiểu thuật toán ECLAT - Phụ trách làm slide thuyết trình	Huỳnh Ngọc Khánh Vy	100%
- Phụ trách nội dung: Chương 4, 3 - Tiền xử lý và trực quan hóa	Nguyễn Thành Vinh	100%
- Phụ trách nội dung: Chương 1, 2 - Tìm hiểu thuật toán ECLAT - Phụ trách làm slide thuyết trình	Nguyễn Vân Phi Yến	100%

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BIỂU ĐỒ

CHƯƠNG 1 - TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Trong bối cảnh thị trường mua sắm toàn cầu không ngừng phát triển mạnh mẽ, đặc biệt là sự bùng nổ của ngành thương mại điện tử (E - Commerce). Năm 2023, theo số liệu thống kê từ Statista, doanh số thương mại điện tử bán lẻ của toàn cầu ước tính đạt 5,8 nghìn tỷ USD, và được dự báo sẽ tăng trưởng 39% trong những năm tới, kỳ vọng vượt hơn 8 nghìn tỷ vào năm 2027. Trong cùng năm, chỉ riêng tại Việt Nam, theo báo cáo của Access Parnership, giá trị xuất khẩu thương mại điện tử đã đạt mốc 86 nghìn tỷ đồng. Điều đáng chú ý là hơn 93% các doanh nghiệp vừa, nhỏ và siêu nhỏ được khảo sát đã cho rằng nếu không có thương mại điện tử thì họ không thể xuất khẩu.

Đứng trước bối cảnh thị trường cạnh tranh gay gắt như thế, một trong những yếu tố quan trọng giúp các sàn thương mại điện tử nói riêng, và các siêu thị, cửa hàng trực tuyến nói chung, cạnh tranh hiệu quả là có thể hiểu rõ được hành vi mua sắm của khách hàng. Từ đó, đưa ra các chiến lược nhằm thu hút và giữ chân người tiêu dùng. Một trong những công cụ quan trọng giúp đạt được mục tiêu này là *'Bài toán phân tích giỏ hàng' (Market Basket Analysis)* - một bài toán quan trọng trong khai phá dữ liệu, nhằm khám phá các mối liên kết hoặc thói quen mua sắm của người tiêu dùng, dựa trên những mặt hàng mà họ thường xuyên mua cùng nhau. Từ đó, đưa ra các quyết định chiến lược như sắp xếp hàng hóa trên kệ hàng, chiến lược khuyến mãi kèm theo để tăng trải nghiệm mua sắm cho khách hàng.

Trong khuôn khổ bài báo cáo này, nhóm sẽ tiến hành sử dụng thuật toán *ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal)* để giải quyết bài toán *Market Basket Analysis*, nhằm khai thác và phát hiện các tập phổ biến và luật kết hợp từ dữ liệu giao dịch.

1.2. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đề tài này, bao gồm:

- ➢ Ứng dụng thuật toán ECLAT để giải quyết bài toán Market Basket Analysis.
- ➢ Đánh giá hiệu suất của thuật toán ECLAT trong việc phân tích giỏ hàng. Đồng thời, so sánh hiệu suất của kết quả với hai thuật toán Apriori và FP-Growth.

1.3. Phương pháp nghiên cứu

Nhằm đạt được các mục tiêu đã đề ra, nhóm tiến hành thực hiện đề tài dựa trên các phương pháp nghiên cứu sau:

- ➢ Phân tích và trực quan hóa dữ liệu để hiểu rõ hơn về tập dữ liệu, cấu trúc, phân phối và tính chất của từng giao dịch mua hàng.
- ➢ Áp dụng thuật toán ECLAT để tìm ra các tập phổ biến và luật kết hợp của bộ dữ liệu.
- ➢ Đánh giá và so sánh các kết quả thu được của thuật toán ECLAT với hai thuật toán khác là Apriori và FP-Growth.

1.4. Tài nguyên sử dụng

- ➢ Nguồn dữ liệu: Bộ dữ liệu *Market Basket Analysis 1* do giảng viên hướng dẫn môn học *Khai phá dữ liệu* cung cấp.
- ➢ Ngôn ngữ và thư viện lập trình:
    - ○ Ngôn ngữ lập trình Python: Dùng để xử lý dữ liệu, phân tích, biểu diễn trực quan và áp dụng thuật toán
    - ○ Các thư viện hỗ trợ:
        - ■ pandas
        - ■ numpy
        - ■ matplotlib
        - ■ seaborn
        - ■ plotly
        - ■ mlxtend

CHƯƠNG 2 - TỔNG QUAN VỀ BÀI TOÁN MARKET BASKET ANALYSIS

2.1. Tổng quan về luật kết hợp

2.1.1. Giới thiệu khai phá luật kết hợp

Khai phá luật kết hợp (Association Rule Mining) là một kỹ thuật quan trọng trong khai phá dữ liệu (Data Mining), tập trung vào việc phát hiện các mối quan hệ, mô hình hoặc mẫu ẩn trong tập dữ liệu lớn. Mô hình đầu tiên của bài toán Khai phá luật kết hợp là mô hình nhị phân được đề xuất bởi R. Agrawal, T. Imielinski và A. Swami vào năm 1993.

Bài toán kinh điển dẫn đến việc khai phá luật kết hợp là bài toán giỏ mua hàng trong siêu thị, với một lượng lớn và đa dạng các mặt hàng được khách hàng bỏ vào giỏ của họ. Các nhà quản lý trên cơ sở muốn tìm hiểu liệu khách hàng thường mua các mặt hàng nào đồng thời hoặc nếu họ mua sản phẩm A, B có tiếp tục mua sản phẩm C nào đó hay không, và những sản phẩm được phân tích chính xác sẽ cho ra các liên kết nhất định. Từ đó nhà quản lý có thể điều chỉnh việc nhập hàng, bố trí các mặt hàng này gần nhau hoặc đơn giản là bán các mặt hàng đó theo một gói hàng.

Khai phá luật kết hợp được mô tả như sự tương quan của những sự kiện xuất hiện thường xuyên một cách đồng thời. Nhiệm vụ chính của khai phá luật kết hợp là phát hiện ra các tập con cùng xuất hiện trong một khối lượng giao dịch lớn của một cơ sở dữ liệu cho trước.

2.1.2. Giới thiệu luật kết hợp

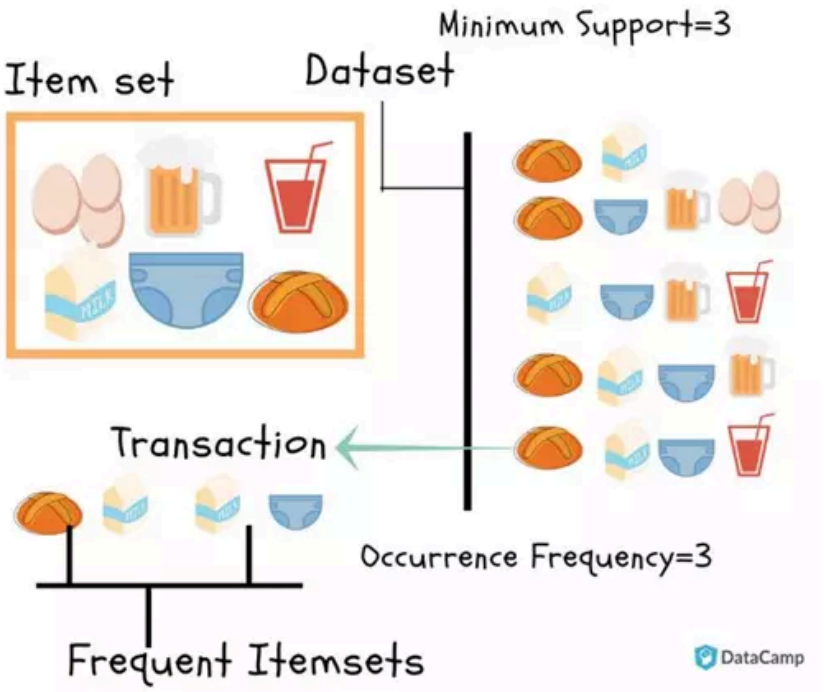
Trong lĩnh vực Data Mining, mục đích của luật kết hợp (Association Rule - AR) là tìm ra các mối kết hợp (Association) hay tương quan (Correlation) giữa các đối tượng trong khối lượng lớn dữ liệu. Ứng dụng của luật kết hợp rất phổ biến trong nhiều lĩnh vực, nhất là trong kinh doanh như Market Basket Analysis (Cross Selling**,** Product Placement, Affinity Promotion, Customer Behavior Analysis).

Khai phá luật kết hợp nhằm tìm ra các **luật kết hợp** dưới dạng: X→Y. Trong đó:

- X và Y là các tập hợp mục (itemsets).
- Quy tắc này ngụ ý rằng nếu một khách hàng mua X, thì họ có khả năng mua Y.

Ví dụ: Trong một siêu thị, luật kết hợp có thể là:

- "Nếu khách hàng mua **bánh mì**, họ có khả năng cao sẽ mua **bơ**."



Quá trình khai thác quy tắc kết hợp gồm hai bước chính:

Bước 1: **Xác định các tập hợp mục thường xuyên (Frequent Itemsets)**:

- Đây là các nhóm mục (itemsets) xuất hiện thường xuyên trong tập dữ liệu giao dịch.
- Một tập hợp mục được coi là thường xuyên nếu tần suất xuất hiện của nó đạt hoặc vượt qua ngưỡng hỗ trợ tối thiểu (**minSup**) đã được định trước.

Bước 2**: Tạo các quy tắc kết hợp mạnh từ các tập hợp mục thường xuyên**:

- Từ các tập hợp mục thường xuyên, ta xây dựng các quy tắc chỉ ra mối quan hệ giữa các sản phẩm.
- Quy tắc kết hợp chỉ được chấp nhận nếu nó đáp ứng cả hai tiêu chí:
    - **Hỗ trợ tối thiểu** (**minSup**): Đảm bảo quy tắc dựa trên các tập hợp mục thường xuyên.
    - **Độ tin cậy tối thiểu** (**minConf**): Đảm bảo mức độ chắc chắn rằng khi một tập hợp mục xuất hiện, tập hợp mục còn lại sẽ xuất hiện theo.

2.1.3 Các chỉ số đo lường luật kết hợp

- ***Support (Độ hỗ trợ):*** Đo lường mức độ phổ biến của một tập hợp mục trong toàn bộ tập dữ liệu.

$$\text{Support}(X \rightarrow Y) = \frac{số\ giao\ dịch\ chứa\ cả\ X\ và\ Y}{tổng\ số\ giao\ dịch}$$

Công thức trên cho biết tần suất xuất hiện của X và Y trong dữ liệu giao dịch. Giá trị Support cao cho thấy luật kết hợp liên quan đến nhiều giao dịch.

- ***Confidence (Độ tin cậy):*** Đo lường xác suất xảy ra Y khi X đã xảy ra. Giá trị Confidence cao cho thấy khi X xuất hiện, Y rất có khả năng xuất hiện.

$$\text{Confidence}(X \rightarrow Y) = \frac{số\ giao\ dịch\ chứa\ cả\ X\ và\ Y}{số\ giao\ dịch\ chứa\ X}$$

- ***Lift:*** Đo lường mức độ mạnh mẽ của mối liên hệ giữa X và Y, so với trường hợp ngẫu nhiên.

$$\text{Lift}(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{Support\ (Y)}$$

➢ **Lift > 1:** Mối quan hệ tích cực giữa X và Y (khi X xảy ra, Y có xu hướng xảy ra cao hơn bình thường).

➢ **Lift = 1:** Không có mối quan hệ (sự xuất hiện của X và Y là ngẫu nhiên).

➢ **Lift < 1:** Mối quan hệ tiêu cực (khi X xảy ra, Y ít có khả năng xảy ra).

2.2. Giới thiệu về bài toán Market Basket

2.2.1 Giới thiệu chung

Phân tích giỏ hàng (Market Basket Analysis) là một phương pháp khai thác dữ liệu giúp khám phá và đánh giá mối liên hệ giữa các sản phẩm thường được mua cùng nhau. Cách tiếp cận này, còn được gọi là khai thác tập hợp mục thường xuyên hoặc phân tích liên kết, giúp nhà bán lẻ nhận diện các mẫu hành vi mua sắm của khách hàng. Bằng cách phân tích dữ liệu giao dịch, nó xác định các sản phẩm có xu hướng đi kèm, từ đó cung cấp thông tin để tối ưu hóa quản lý hàng tồn kho.

Market Basket Analysis có thể được xem là ứng dụng chính của phân tích luật kết hợp, vì nó cho thấy lợi ích kinh doanh cao trong các thuật toán Data Mining được sử dụng. “Market Basket” là tên gọi của Database thu thập dữ liệu giao dịch tại các siêu thị hoặc trung tâm thương mại với tập dữ liệu thường rất lớn. Mỗi giao dịch được ghi lại có thể chứa các thông tin như danh sách sản phẩm được mua, ID khách hàng hoặc mã số giao dịch.

2.2.2 Các loại phân tích giỏ hàng

- ***Phân tích giỏ hàng mô tả (Descriptive Market Basket Analysis)***

Phân tích giỏ hàng mô tả cung cấp những thông tin hữu ích và có thể áp dụng dựa trên dữ liệu lịch sử. Đây là một phương pháp phổ biến, tập trung vào việc đánh giá mối liên hệ giữa các sản phẩm bằng các kỹ thuật thống kê mà không đưa ra dự đoán. Cách tiếp cận này thường được gọi là học máy không giám sát (Unsupervised Learning) do cách nó được xây dựng và vận hành.

- ***Phân tích giỏ hàng dự đoán (Predictive Market Basket Analysis)***

Mặc dù thuật ngữ "Predict" và "Analysis" tạo nên cụm từ "phân tích dự đoán", nhưng thực chất quy trình này diễn ra theo chiều ngược lại: trước tiên phân tích dữ liệu, sau đó dự đoán xu hướng tương lai. Loại phân tích này sử dụng các mô hình học máy có giám sát như hồi quy (Regression) và phân loại (Classification).

Trong phân tích giỏ hàng dự đoán, các mặt hàng được mua theo trình tự sẽ được xem xét để đánh giá cơ hội bán kèm (cross-sell). Chẳng hạn, khi khách hàng mua một chiếc laptop, họ có khả năng cao sẽ mua thêm gói bảo hành mở rộng. Phân tích này giúp nhận diện những sản phẩm liên quan theo trình tự để bán kèm hiệu quả hơn.

- ***Phân tích giỏ hàng phân biệt (Differential Market Basket Analysis)***

Phân tích giỏ hàng phân biệt thường được sử dụng trong phân tích cạnh tranh, giúp xác định lý do tại sao người tiêu dùng lại chọn mua cùng một sản phẩm trên một nền tảng cụ thể, dù giá cả của sản phẩm được niêm yết giống nhau trên cả hai nền tảng.

Bằng cách xem xét các yếu tố tác động đến quyết định của người tiêu dùng, các tổ chức có thể tận dụng phân tích giỏ hàng phân biệt để điều chỉnh các thông số, cải thiện trải nghiệm người dùng và tăng doanh số bán hàng trên nền tảng của mình.

2.2.3 Các thuật ngữ được sử dụng trong phân tích giỏ hàng

Itemset - Tập hợp sản phẩm: Tập hợp sản phẩm là nhóm các mặt hàng được khách hàng mua cùng nhau trong cùng một thời điểm. Trường hợp một tập hợp sản phẩm không chứa mặt hàng nào cụ thể; những tập hợp này thường bị loại bỏ khi phân tích dữ liệu., vì chúng không mang lại giá trị đáng kể trong tập dữ liệu.

Support Count - Tần suất hỗ trợ: là số lần một tập hợp sản phẩm cụ thể xuất hiện trong cơ sở dữ liệu giao dịch và cũng được biểu diễn dưới dạng xác suất.
Ví dụ: Nếu sản phẩm **sữa** có tần suất hỗ trợ là 50 trong tổng số 500 giao dịch, thì xác suất hỗ trợ của nó sẽ là **50/500 = 0,1** (hay 10%).

Confidence - Độ tin cậy: là xác suất có điều kiện biểu thị khả năng các sản phẩm được mua cùng nhau. Chỉ số này thường được áp dụng trong chiến lược bố trí sản phẩm nhằm tăng lợi nhuận.

Antecedent - Tiền đề: là thành phần IF được viết ở phía bên trái của một quy tắc hoặc tập hợp sản phẩm trong dữ liệu. Nó đại diện cho các mặt hàng hoặc điều kiện ban đầu trong quy tắc liên kết.

Consequent - Kết quả: là thành phần **THEN**, biểu thị một sản phẩm hoặc tập hợp sản phẩm được tìm thấy kết hợp với tiền đề (**Antecedent**) trong một quy tắc liên kết.

2.3. Các thuật toán phổ biến

2.3.1. Thuật toán Apriori

a) Tổng quan về thuật toán Apriori

Thuật toán Apriori là một trong những thuật toán phổ biến trong việc khai phá các luật kết hợp từ các tập cơ sở dữ liệu. Thuật toán được đề xuất lần đầu bởi R. Agrawal và R. Srikant vào năm 1994 trong bài nghiên cứu *'Fast Algorithms for Mining Association Rules'*.

Ý tưởng chính của thuật toán Apriori có thể được phát biểu như sau:

- ➢ Tìm tất cả các tập mục phổ biến *(frequent itemsets)* với *minSup* nào đó.
  - *k-itemset* (itemsets gồm *k* items) được dùng để tìm *(k+1)-itemset.*
  - Đầu tiên, tìm L_1 *(1-itemset)* được dùng để tìm L_2 *(2-itemsets).* Tiếp tục, từ L_2 dùng để tìm ra L_3 *(3-itemsets),* và cứ tiếp tục cho đến khi không tìm được *k-itemset* nào.
- ➢ Sử dụng các tập mục phổ biến đã tìm được để phát sinh ra các luật kết hợp mạnh.

b) Mô tả quy trình hoạt động

Quy trình hoạt động của thuật toán Apriori bao gồm các bước chính sau:

Bước 1: Duyệt toàn bộ cơ sở dữ liệu giao dịch *(transaction database)* để xác định tập phổ biến *(frequent itemset)* ban đầu:

- ➢ Duyệt toàn bộ transaction database để tính độ hỗ trợ *(support)* cho từng tập mục *1-itemset.*
- ➢ So sánh giá trị hỗ trợ của mỗi tập mục với độ hỗ trợ tối thiểu *(minSup)* để chọn ra tập mục phổ biến *frequent 1-itemset* L_1.

Bước 2: Sinh ra tập *k-itemset*:

- ➢ Sử dụng tập $(k-1)$-*itemset* phổ biến $L_{(k-1)}$ để kết hợp tạo ra tập ứng viên *k-itemset.*
- ➢ Loại bỏ các *k-itemset* ứng viên mà bất kỳ tập con $(k-1)$-*itemset* nào của nó không phải là tập phổ biến.

Bước 3: Tính toán độ hỗ trợ cho tập ứng viên *k-itemset:*

- ➢ Quét cơ sở dữ liệu giao dịch để tính độ hỗ trợ cho từng tập ứng viên *k-itemset.*
- ➢ So sánh độ hỗ trợ với giá trị *minSup* để xác định tập phổ biến *k-itemset* L_k.

Bước 4: Lặp lại quá trình trên:

- ➢ Tiếp tục từ bước 2, tăng giá trị *k* và lặp lại quy trình cho đến khi không còn tập ứng viên nào.

Bước 5: Phát sinh luật kết hợp *(Association Rules):*

- ➢ Với mỗi tập phổ biến X thu được sinh ra tất cả các tập con khác rỗng $X_i \notin \emptyset$.
- ➢ Với mỗi tập con khác rỗng $X_i \subset X$, sinh ra được các luật kết hợp dưới dạng $X_i \Rightarrow (X - X_i)$, với điều kiện độ tin cậy *conf* của luật lớn hơn hoặc bằng độ tin cậy tối thiểu *minConf.*

c) Ví dụ minh họa:

Cho cơ sở dữ liệu giao dịch như sau, với *[minSup, minConf] = (60%, 80%)*

Tid	Items
1	A, C, T, W

2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

minSup = 60% ⇒ minFreq = 60%. 6 = 4

Áp dụng thuật toán Apriori để khai phá luật kết hợp được mô tả qua các bước sau:

C_1

Items	Count
A	4
C	6
D	4
T	4
W	5

L_1

Items	Count
A	4
C	6
D	4
T	4
W	5

C_2

Items	Count
AC	4
AD	2
AT	2
AW	4
CD	4
CT	4
CW	5
DT	2
DW	3
TW	3

L_2

Items	Count
AC	4
AW	4
CD	4
CT	4
CW	5

C_3

Items	Count
ACW	4

L_3

Items	Count
ACW	4

$C_4 = \varnothing$

Phát sinh luật kết hợp:

1. $AC \rightarrow W$ có độ tin cậy: $conf = \frac{4}{4} = 100\%$
2. $AW \rightarrow C$ có độ tin cậy: $conf = \frac{4}{4} = 100\%$
3. $CW \rightarrow A$ có độ tin cậy: $conf = \frac{4}{5} = 80\%$

2.3.2. Thuật toán FP-Growth

a) Tổng quan về thuật toán FP-Growth

Thuật toán FP-Growth (viết tắt của Frequent Pattern Growth) được đề xuất bởi các tác giả Jiawei Han, Jian Pei, Yiwen Yin, Runying Mao vào năm 2004 trong bài nghiên cứu *'Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach'*.

Khác với thuật toán Apriori, thuật toán FP-Growth khám phá các tập phổ biến *frequent itemsets* mà không cần sinh ra các tập ứng viên *candidate itemsets* và quét cơ sở dữ liệu nhiều lần. Thay vào đó, FP-Growth biểu diễn các cơ sở dữ liệu giao dịch bằng một cấu trúc dữ liệu được gọi là *FP-Tree.*

FP-Tree là cấu trúc dữ liệu được J.Han đề xuất vào năm 2004, nó mô tả lại cơ sở dữ liệu dưới dạng cây mà mỗi nút trong cây sẽ bao gồm các thông tin về *item i* và số lần xuất hiện của *item i* trong cơ sở dữ liệu. Từ nút gốc của cây, đi theo mỗi nhánh, đến nút lá sẽ biểu diễn cho một giao dịch trong cơ sở dữ liệu.

Cụ thể, thuật toán FP-Growth được xây dựng dựa trên ý tưởng sau:

- *Nén dữ liệu vào cây FP-Tree:* Thuật toán tiến hành nén dữ liệu vào *FP-Tree.* Chỉ các tập mục phổ biến *(1-itemset)* được lưu trữ trong cây. Trong đó, các nút của cây sẽ được sắp xếp theo tần suất xuất hiện, từ lớn đến bé. Điều này giúp các items xuất hiện thường xuyên hơn sẽ được chia sẻ hiệu quả hơn với các items ít xuất hiện hơn.
- Tiếp đến, sử dụng cấu trúc *FP-Tree* để khai phá các tập phổ biến.
- Thực hiện tuần tự để tìm kiếm các tập phổ biến.

Nhờ việc nén toàn bộ cơ sở dữ liệu vào *FP-Tree* nên thuật toán FP-Growth chỉ cần duyệt cơ sở dữ liệu đúng hai lần. Từ đó, giúp giảm thiểu đáng kể số lần duyệt và tránh được khối lượng tính toán lớn và phức tạp như trong thuật toán Apriori.

b) Mô tả quy trình hoạt động

Quy trình hoạt động của thuật toán FP-Growth được thực hiện qua các bước sau:

Bước 1: Duyệt cơ sở dữ liệu lần thứ nhất

- Thuật toán tiến hành quét toàn bộ cơ sở dữ liệu giao dịch để tính số lần xuất hiện của từng *item.*
- Loại bỏ các *items* không thỏa mãn ≥ *minSup.*

Bước 2: Sắp xếp danh sách các mục

- ➢ Tạo danh sách *F-List* bao gồm các *items* còn lại, và được sắp xếp theo thứ tự giảm dần của độ hỗ trợ. Thứ tự này đảm bảo rằng các *items* phổ biến hơn sẽ được xuất hiện trước trong *FP-Tree.*

Bước 3: Duyệt cơ sở dữ liệu lần thứ hai và xây dựng *FP-Tree*

- ➢ Với mỗi giao dịch *t* trong cơ sở dữ liệu:
  - ○ Loại bỏ các *items* không thỏa mãn *minSup*
  - ○ Sắp xếp các *items* còn lại trong *t* theo thứ tự giảm dần của độ hỗ trợ như *F-List*
  - ○ Chèn các *items* vào cây *FP-Tree*, với mỗi nhánh trong cây sẽ đại diện cho một giao dịch. Nếu một nhánh đã tồn tại, giá trị đếm *(count)* của nút tương ứng sẽ được tăng lên, nếu không tồn tại, tiến hành tạo nút mới.

Bước 4: Khai phá tập phổ biến từ *FP-Tree*

- ➢ Sau khi hoàn thành xây dựng *FP-Tree,* thực hiện khai phá để tìm các mẫu phổ biến trực tiếp trên cây, mà không cần duyệt lại cơ sở dữ liệu.
- ➢ Kỹ thuật khai phá dựa trên tập mẫu điều kiện *(conditional pattern base)* và cây FP-Tree con *(conditional FP-Tree)* được sinh ra để xác định các tập phổ biến.

2.3.3. Thuật toán ECLAT

Thuật toán ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) được đề xuất bởi M. J. Zaki và cộng sự trong bài nghiên cứu *'New algorithms for fast discovery of association rules'* năm 1997. Thuật toán được nghiên cứu và phát triển nhằm có thể khắc phục được những hạn chế từ thuật toán Apriori bằng phương pháp tối ưu không gian sinh dựa trên khái niệm về lớp tương đương, sử dụng dữ liệu được định dạng theo chiều dọc và chỉ cần duyệt cơ sở dữ liệu duy nhất một lần.

Chi tiết về các khái niệm liên quan và quy trình hoạt động của thuật toán ECLAT sẽ được trình bày cụ thể trong *Chương 3 - Tổng quan về Thuật toán ECLAT*. Chương này sẽ cung cấp cái nhìn sâu hơn về cách thuật toán được triển khai và trình bày ví dụ minh họa để hiểu rõ hơn về thuật toán ECLAT.

2.3.4. So sánh

Nhằm hiểu rõ hơn về sự khác biệt, cũng như ưu và nhược điểm của ba thuật toán *Apriori, FP-Growth* và *ECLAT.* Nhóm tiến hành lập bảng so sánh chi tiết dựa trên các tiêu chí như phương pháp triển khai, số lần duyệt cơ sở dữ liệu, ưu và nhược điểm. *(Srinadh, V. (2022).)*

	Apriori	**FP-Growth**	**ECLAT**
Phương pháp triển khai	- Tìm kiếm theo chiều rộng (Breadth-First Search) - Dựa trên việc sinh các tập ứng viên và quét bộ dữ liệu nhiều lần	- Dựa trên cây *FP-Tree* - Không cần sinh tập ứng viên, khai phá trực tiếp từ FP-Tree	- Tìm kiếm theo chiều sâu (Depth-First Search) - Giao của các TID-List để tạo tập ứng viên
Số lần duyệt cơ sở dữ liệu	Duyệt cơ sở dữ liệu nhiều lần	Duyệt 02 lần: - Một lần để xây dựng FP-Tree - Một lần để khai phá	Duyệt 01 lần: Truy cập trực tiếp vào dữ liệu giao dịch theo chiều dọc
Tốc độ	Chậm, đặc biệt với bộ dữ liệu lớn và nhiều tập ứng viên	Nhanh hơn Apriori nhờ việc giảm số lần duyệt và không tạo tập ứng viên.	Nhanh với dữ liệu lớn, nhưng có thể kém hiệu quả khi có quá nhiều tập con
Định dạng dữ liệu	Dữ liệu dọc (Horizontal layout-based)	Dữ liệu dọc (Horizontal layout-based) và được chuyển thành FP-Tree	Dữ liệu dọc (Vertical layout-based)
Ưu điểm	- Đơn giản, dễ hiểu và dễ triển khai - Được áp dụng rộng rãi trong thực tiễn	- Không cần sinh tập ứng viên - Tìm kiếm các tập phổ biến nhanh chóng và hiệu quả	- Duyệt dữ liệu một lần, giảm thiểu chi phí tính toán
Nhược điểm	- Tốn thời gian, dung lượng bộ nhớ	- Cần nhiều bộ nhớ để xây dựng	- Tốn nhiều bộ nhớ khi có nhiều tập

	với bộ dữ liệu lớn - Số lượng tập ứng viên tăng nhanh	*FP-Tree* - Khó khăn trong việc cập nhật cây *FP-Tree* khi có thay đổi trong dữ liệu	con và lớp tương đương

CHƯƠNG 3 - TỔNG QUAN VỀ THUẬT TOÁN ECLAT

3.1. Tổng quan về thuật toán ECLAT

Thuật toán ECLAT *(Equivalence Class Clustering and bottom-up Lattice Traversal)* được giới thiệu lần đầu bởi Zaki Mohammed J. vào năm 1997, trong bài nghiên cứu có tựa đề *"Scalable Algorithms for Association Mining"* tại hội nghị SIGKDD. Eclat ra đời như một giải pháp thay thế cho thuật toán Apriori vốn tiêu tốn nhiều tài nguyên tính toán khi xử lý tập tin lớn.

Thuật toán ECLAT là phiên bản hiệu quả hơn và có thể mở rộng của thuật toán Apriori. Trong khi thuật toán Apriori hoạt động theo chiều ngang, mô phỏng theo thuật toán tìm kiếm theo chiều rộng của đồ thị, thuật toán Eclat hoạt động theo chiều dọc giống như thuật toán tìm kiếm theo chiều sâu của đồ thị. Cách tiếp cận này giúp ECLAT trở thành thuật toán nhanh hơn thuật toán Apriori.

Thuật toán ECLAT tập trung vào việc tìm frequency itemsets bằng cách sử dụng intersection của tập giao dịch thay vì quét lại toàn bộ dữ liệu nhiều lần. Điều này giúp Eclat tối ưu hơn Apriori trong các trường hợp tập dữ liệu dày đặc.

3.2. Các thuật ngữ được sử dụng trong thuật toán ECLAT

- ➢ *Frequent Itemset:* Tập hợp các items có số giao dịch (support) lớn hơn hoặc bằng minSup.
- ➢ *TID - List (Transaction ID List):* Một danh sách các giao dịch (Transaction IDs) mà một item xuất hiện. Dạng biểu diễn vertical giúp tối ưu hóa việc tính toán tập phổ biến.
- ➢ *Prefix:* Một tập con item được dùng làm cơ sở để mở rộng thêm các items khác.
- ➢ *Intersection of TID - Lists:* Phép giao giữa TID - lists của hai item hoặc tập item, xác định giao dịch chung.
- ➢ *Equivalence Class:* Nhóm các tập item có chung một prefix, giúp chia nhỏ không gian tìm kiếm.

3.3. Cơ chế hoạt động của thuật toán ECLAT

➢ ***Đầu vào:***

- *Dataset D:* danh sách các giao dịch, mỗi giao dịch là một tập hợp các items.
- *minSup:* ngưỡng tần suất tối thiểu để một tập hợp được coi là phổ biến.

➢ ***Đầu* ra:**

- *Tập hợp các tập phổ biến.*

➢ ***Quy trình:***

Bước 1: Chuyển đổi dữ liệu sang vertical representation (TID - List)

- Với mỗi item, lưu trữ danh sách các giao dịch mà item đó xuất hiện.

Bước 2: Bắt đầu với các item riêng lẻ

- Xét từng item đơn lẻ (1-itemset) cùng TID-List của chúng.
- Tính support của mỗi item bằng cách đếm số lượng phần tử trong TID-List.
- Chỉ giữ lại các items có support không nhỏ hơn minSup.

Bước 3: Depth First Search (DFS)

- Sử dụng chiến lược duyệt sâu để khám phá các tập phổ biến lớn hơn từ các tập phổ biến nhỏ hơn (mở rộng các tập prefix).
- Tại mỗi bước thực hiện:
  - Gọi tập prefix hiện tại là {A} và các items khác trong cơ sở dữ liệu là {B, C, D…}.
  - Kết hợp prefix với từng item khác để tạo ra tập itemset mới, ví dụ: {A, B}, {A, C}...
  - Lấy giao của TID-list của {A} với TID-list của từng item khác:

$$TID - list(\{A, B\}) = TID - list(\{A\}) \cap TID - list(\{B\})$$

  - Kiểm tra ngưỡng minSup: đếm số phần tử trong TID-List của tập mở rộng, nếu số lượng này không bé hơn minSup thì tập mở rộng được coi là phổ biến.

Bước 4: Cắt tỉa các itemsets không phổ biến

- Sau mỗi lần mở rộng, nếu TID-List của itemset mở rộng không đủ số lượng giao dịch (bé hơn minSup), thì tập đó và tất cả các tập mở rộng của nó sẽ bị loại bỏ.

- Ví dụ: nếu {A, B} không phổ biến, thì {A, B, C} cũng không phổ biến nên không cần xét đến.

Bước 5: Lặp lại quá trình

- Tiếp tục mở rộng các itemsets phổ biến bằng cách duyệt sâu vào các items khác và tính giao TID-List. Lặp lại khi không thể mở rộng thêm bất kỳ itemset nào.
- Quy trình dừng khi: hoặc không còn item nào để mở rộng hoặc tất cả các tập phổ biến đã được phát hiện.
- Sau khi kết thúc, trả về danh sách các tập phổ biến và độ hỗ trợ của từng tập.

3.4. Ví dụ minh họa

Cho cơ sở dữ liệu giao dịch như hình, với *minSup = 2:*

Transaction ID	Items
T1	A, B, E
T2	B, D
T3	B, C
T4	A, B, D
T5	A, C
T6	B, C
T7	A, C
T8	A, B, C, E
T9	A, B, C

➢ ***k - items = 1***

Items	TID-lists
A	T1, T4, T5, T7, T8, T9
B	T1, T2, T3, T4, T6, T8, T9
C	T3, T5, T6, T7, T8, T9
D	T2, T4
E	T1, T8

➢ ***k - items = 2***

Items	TID-lists
A, B	T1, T4, T8, T9
A, C	T5, T7, T8, T9
A, D	T4
A, E	T1, T8
B, C	T3, T6, T8, T9
B, D	T2, T4
B, E	T1, T8
C, D	Ø
C, E	T8
D, E	Ø

➢ ***k-items = 3***

Items	TID-lists
A, B, C	T8, T9
A, B, D	T4
A, B, E	T1, T8
A, C, D	Ø
A, C, E	T8
B, C, D	Ø
B, C, E	T8
B, D, E	Ø

➢ **Kết luận:** Các itemsets phổ biến {A}, {B}, {C}, {D}, {E}, {A, B}, {A, C}, {A, E}, {B, C}, {B, D}, {B, E}, {A, B, C}, {A, B, E}

CHƯƠNG 4 - XÂY DỰNG THUẬT TOÁN ECLAT TRÊN BỘ DỮ LIỆU

4.1. Mô tả dữ liệu

Bộ dữ liệu Market Basket Analysis 1 lưu trữ thông tin về các sản phẩm trong giỏ hàng của khách hàng khi đi siêu thị. Bao gồm:

- ➢ *7501 dòng:* Mỗi dòng trong bộ dữ liệu tương ứng với một giao dịch.
- ➢ *20 cột:* Mỗi cột đại diện cho một sản phẩm cụ thể trong giỏ hàng.

```
# Quan sát bộ dữ liệu
df.tail()
```

✓ 0.0s

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Hình 4.1. Quan sát bộ dữ liệu Market Basket Analysis 1

Bộ dữ liệu *'Market Basket Analysis 1'* là tập hợp các giao dịch mua sắm, trong đó mỗi dòng tương ứng với một giao dịch và mỗi cột đại diện cho một sản phẩm xuất hiện trong giao dịch đó. Do số lượng sản phẩm trong mỗi giao dịch khác nhau nên các giá trị trống (NaN) không phải lỗi dữ liệu, mà dùng để thể hiện sự khác biệt về số lượng sản phẩm giữa các giao dịch. Để minh họa cho lập luận trên, nhóm lấy ví dụ với giao dịch 7,496 thì chỉ có 3 sản phẩm được mua thì ngoại trừ các cột chứa thông tin sản phẩm các cột còn lại được xem là NaN.

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

```
(7501, 20)
```

Hình 4.2. Kích thước bộ dữ liệu

```
# Quan sát các cột
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7501 entries, 0 to 7500
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       7501 non-null   object
 1   1       5747 non-null   object
 2   2       4389 non-null   object
 3   3       3345 non-null   object
 4   4       2529 non-null   object
 5   5       1864 non-null   object
 6   6       1369 non-null   object
 7   7       981 non-null    object
 8   8       654 non-null    object
 9   9       395 non-null    object
 10  10      256 non-null    object
 11  11      154 non-null    object
 12  12      87 non-null     object
 13  13      47 non-null     object
 14  14      25 non-null     object
 15  15      8 non-null      object
 16  16      4 non-null      object
 17  17      4 non-null      object
 18  18      3 non-null      object
 19  19      1 non-null      object
dtypes: object(20)
memory usage: 1.1+ MB
```

Hình 4.3. Metadata của bộ dữ liệu

Bộ dữ liệu bao gồm 7501 dòng, mỗi dòng đại diện cho 7501 giao dịch với mỗi giao dịch chứa ít nhất một sản phẩm. Các cột trong bộ dữ liệu đại diện cho các sản phẩm xuất hiện trong giao dịch. Dữ liệu cho thấy rằng hầu hết các giao dịch đều có ít nhất một sản phẩm, trong đó giao dịch có số lượng sản phẩm nhiều nhất lên đến 20 sản phẩm.

4.2. Exploratory Data Analysis (EDA)

4.2.1. Xử lý các giao dịch có sản phẩm trùng

Thuật toán *Apriori*, *ECLAT*, và *FP-Growth* đều là các phương pháp phổ biến trong phân tích các luật kết hợp, được sử dụng để khai phá các tập phổ biến trong dữ liệu giao dịch. Tuy nhiên, trong trường hợp nếu một sản phẩm xuất hiện nhiều lần trong cùng một giao dịch, tần suất của sản phẩm đó sẽ được tăng lên, làm ảnh hưởng đến độ chính xác của các mẫu phổ biến được tìm ra. Điều này có thể dẫn đến việc nhận diện không chính xác các mối quan hệ thực sự giữa các sản phẩm trong giỏ hàng. Do đó, việc loại bỏ các sản phẩm trùng lặp trong mỗi giao dịch là một bước quan trọng, không chỉ đảm bảo độ

chính xác của các luật kết hợp mà còn tối ưu hóa hiệu quả tính toán trong quá trình phân tích dữ liệu.

```python
# Hàm tìm các giao dịch có sản phẩm trùng
def find_duplicate_transactions(df, num_columns):
    # Tính số lượng sản phẩm duy nhất trong mỗi giao dịch
    unique_counts = []
    for _, row in df.iterrows():
        unique_items = row.iloc[:num_columns].nunique()
        unique_counts.append(unique_items)
    df['num_uniq'] = unique_counts

    # Tính tổng số sản phẩm (bao gồm trùng lặp) trong mỗi giao dịch
    total_counts = []
    for _, row in df.iterrows():
        total_items = row.iloc[:num_columns].count()
        total_counts.append(total_items)
    df['num_item'] = total_counts

    # Lọc các giao dịch có sản phẩm trùng lặp
    duplicate_transactions = df[df['num_uniq'] != df['num_item']]
    return duplicate_transactions
```

```python
df1=df.copy()
duplicate_items_transactions = find_duplicate_transactions(df1, num_columns=20)
duplicate_items_transactions
```

✓ 1.1s

	0	1	2	3	4	5	6	7	8	9	...	12	13	14	15	16	17	18	19	num_uniq	num_item
4394	burgers	ham	eggs	whole wheat rice	ham	french fries	cookies	green tea	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7	8
4494	ham	eggs	honey	gums	light cream	ham	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	6
4526	ham	milk	chicken	whole wheat rice	ham	eggplant	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	6
6903	ground beef	spaghetti	mineral water	chocolate	salmon	chicken	chocolate	frozen smoothie	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7	8
7109	ham	shrimp	milk	flax seed	salmon	corn	ham	eggplant	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7	8

5 rows × 22 columns

Hình 4.4. Quan sát các giao dịch chứa sản phẩm trùng

Bộ dữ liệu ghi nhận *5 giao dịch* chứa sản phẩm bị trùng lặp, trong đó sản phẩm *'ham'* xuất hiện lặp lại trong *4 giao dịch*, và sản phẩm *'chocolate'* được mua trùng trong giao dịch số 6,903. Sau khi biết được các giao dịch chứa sản phẩm trùng, nhằm đảm bảo mỗi giao dịch chỉ bao gồm các sản phẩm duy nhất, nhóm đã tiến hành lọc bỏ các sản phẩm bị trùng lặp trong các giao dịch này.

```python
# Loại bỏ các sản phẩm trùng trong giao dịch
df = df.replace(np.nan, None)
df_cleaned = []
for _, row in df.iloc[:, 0:20].iterrows():
    temp = []
    for value in row:
        if value is not None and value not in temp:
            temp.append(value)
    df_cleaned.append(temp)
df_cleaned = pd.DataFrame(df_cleaned)
df2 = df_cleaned.copy()
duplicate_items_transactions1 = find_duplicate_transactions(df2, num_columns=20)
print(f'Số giao dịch có sản phẩm trùng là: {len(duplicate_items_transactions1)}')
```

✓ 1.3s

```
Số giao dịch có sản phẩm trùng là: 0
```

```python
# Dữ liệu sau khi lọc
df_cleaned.tail()
```

✓ 0.0s

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7496	butter	light mayo	fresh bread	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	None	None	None	None	None	None	None	None	None	None	None	None	None	None
7498	chicken	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
7499	escalope	green tea	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None

Hình 4.5. Xử lý các giao dịch chứa sản phẩm trùng

Sau khi tiến hành lọc bỏ các sản phẩm trùng lặp, nhóm đã kiểm tra lại và xác nhận không còn giao dịch nào chứa sản phẩm bị trùng. Bộ dữ liệu sau khi xử lý được đặt tên là *df_cleaned,* được sử dụng để tìm kiếm các luật kết hợp, đảm bảo tính chính xác và hiệu quả trong quá trình phân tích.

4.2.2. Biểu diễn trực quan dữ liệu

a) Phân tích sản phẩm

Để có thể biểu diễn trực quan tổng quan các sản phẩm trong tập dữ liệu, nhóm đã chuyển đổi dữ liệu sang dạng *TID-List (Transaction ID List).* Ở dạng này, mỗi sản phẩm được liên kết với danh sách các giao dịch mà sản phẩm đó xuất hiện, giúp dễ dàng quan sát tần xuất xuất hiện của từng sản phẩm, đồng thời hiệu quả hơn trong việc áp dụng các thuật toán khai phá luật kết hợp.

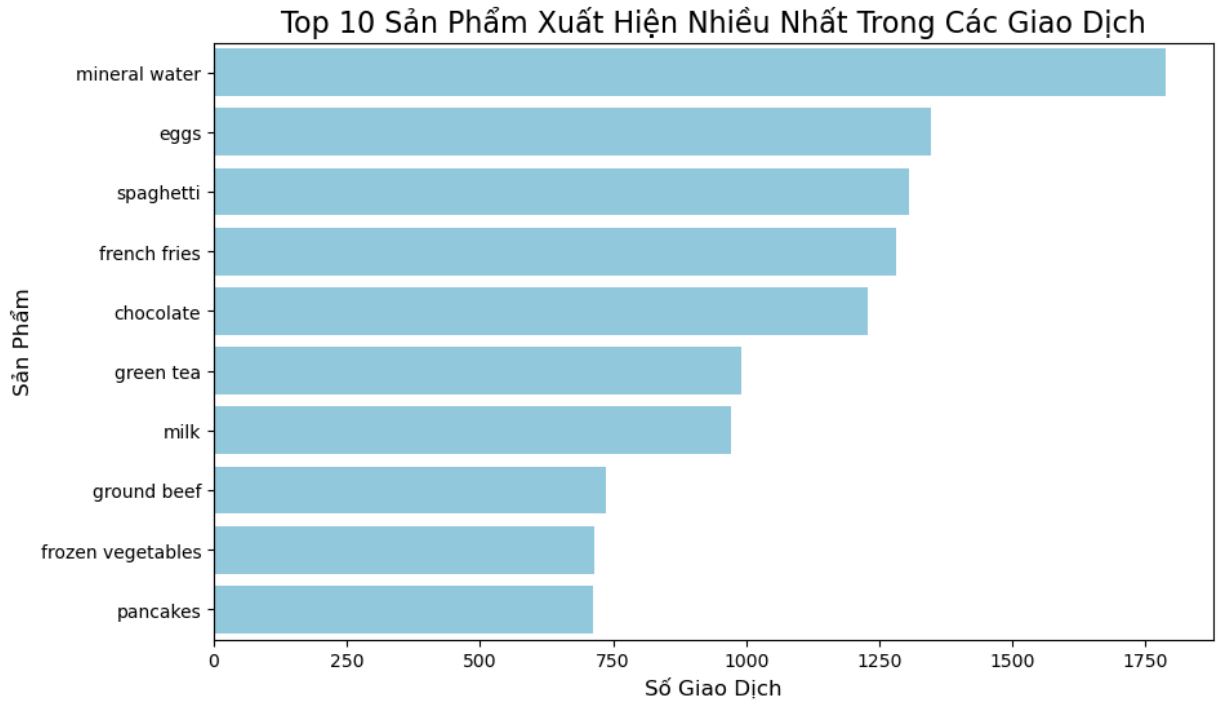
```
df_transposed = df_cleaned.T
# Chuyển đổi từ DataFrame ngang sang dạng dài
df_vertical = df_transposed.reset_index().melt(id_vars='index', value_name='Item', var_name='TransactionID')
# Đặt tên cột
df_vertical.columns = ['TransactionIndex', 'TransactionID', 'Item']
# Kiểm tra và lọc các giá trị trong cột Item
df_filtered = df_vertical[df_vertical['Item'].notna()]
# Loại bỏ các giá trị không phải là tên sản phẩm
df_filtered = df_filtered[df_filtered['Item'].apply(lambda x: isinstance(x, str))]
# Nhóm theo sản phẩm, thu thập các TransactionID có cùng sản phẩm
df_result = df_filtered.groupby('Item')['TransactionID'].apply(list).reset_index()
# Thêm cột 'Total Transactions' là số lượng giao dịch của mỗi sản phẩm
df_result['Total Transactions'] = df_result['TransactionID'].apply(len)
# Sắp xếp theo số lượng giao dịch giảm dần
df_result = df_result.sort_values(by='Total Transactions', ascending=False)
# Kết quả cuối cùng
df_result
```

✓ 0.3s

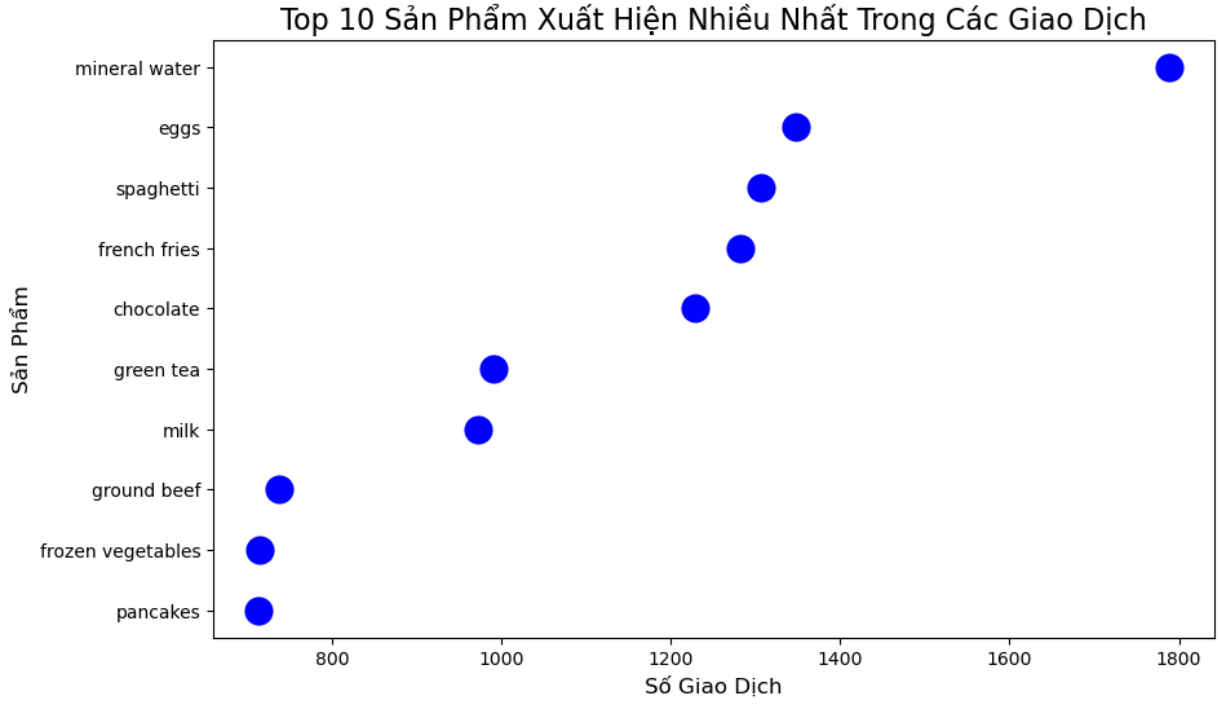
	Item	TransactionID	Total Transactions
72	mineral water	[0, 4, 12, 14, 15, 18, 22, 26, 28, 29, 31, 37,...	1788
37	eggs	[1, 10, 12, 17, 18, 20, 24, 28, 31, 51, 57, 64...	1348
100	spaghetti	[8, 13, 18, 22, 25, 26, 29, 35, 40, 41, 42, 50...	1306
43	french fries	[6, 9, 19, 28, 32, 36, 45, 54, 65, 80, 81, 82,...	1282
25	chocolate	[16, 20, 28, 31, 33, 50, 53, 56, 66, 76, 86, 9...	1229
...	...	...	...
11	bramble	[503, 517, 712, 1485, 1756, 2407, 2627, 3012, ...	14
34	cream	[826, 1306, 2376, 3919, 3952, 4256, 5044]	7
77	napkins	[871, 2961, 3116, 3658, 7177]	5
112	water spray	[149, 390, 1352]	3
0	asparagus	[694]	1

120 rows × 3 columns

Hình 4.6. Chuyển đổi dữ liệu sang dạng TID-List



Biểu đồ 4.1. Biểu đồ cột ngang thể hiện top 10 sản phẩm xuất hiện nhiều nhất trong các giao dịch



Biểu đồ 4.2. Biểu đồ scatter plot hiển thị top 10 sản phẩm xuất hiện nhiều nhất trong các giao dịch

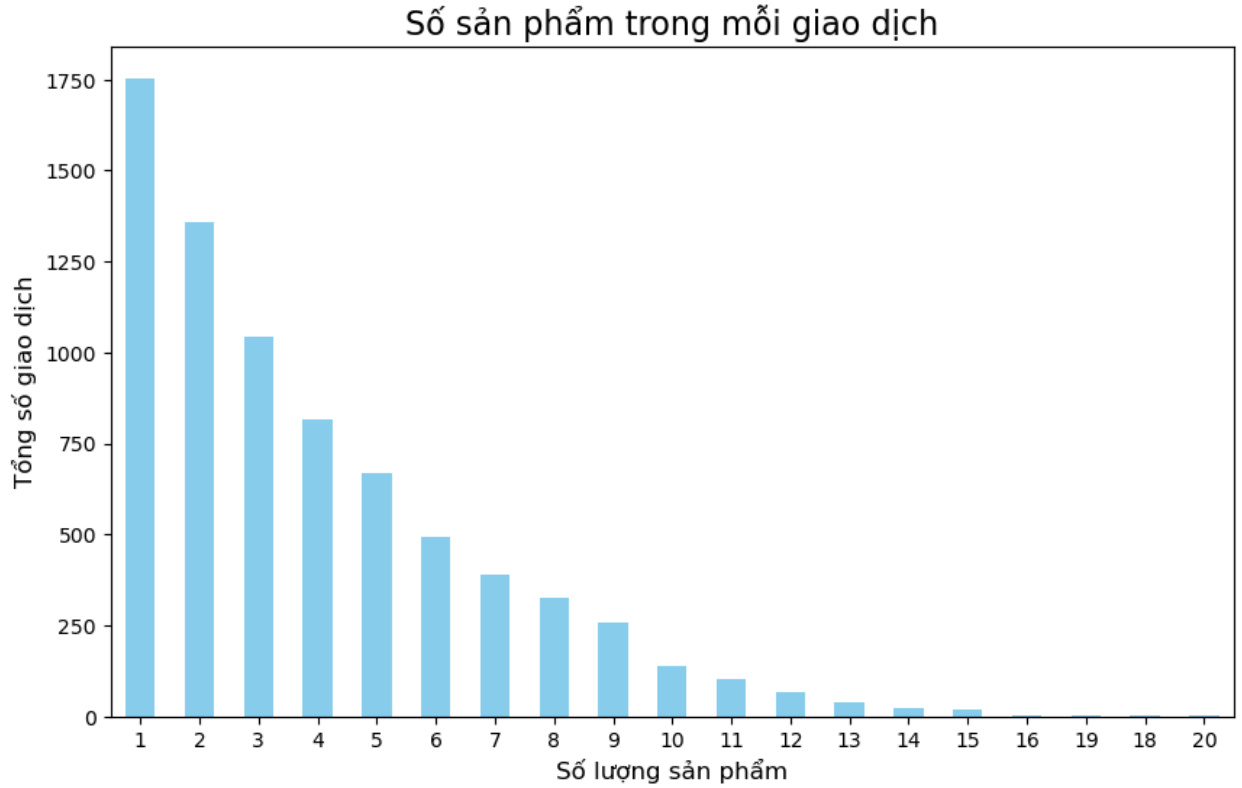
Dựa trên kết quả từ *Biểu đồ 4.1* và *Biểu đồ 4.2,* top 10 sản phẩm phổ biến nhất trong các giao dịch, bao gồm: *mineral water (nước khoáng), eggs (trứng), spaghetti (mì*

sợi), french fries (khoai tây chiên), chocolate (sôcôla), green tea (trà xanh), milk (sữa), ground beef (thịt bò xay), frozen vegetables (rau củ đông lạnh) và pancakes (bánh). Trong đó, *mineral water (nước khoáng)* là sản phẩm được mua nhiều nhất, xuất hiện 1,788 lần trong các giao dịch. Điều này cho thấy nước khoáng là mặt hàng có nhu cầu tiêu thụ rất cao bởi đây là sản phẩm nhu cầu thiết yếu hàng ngày, an toàn vệ sinh và có tính tiện lợi cao.

Các sản phẩm khác như *eggs, spaghetti,* và *french fries* cũng được mua rất thường xuyên bởi đây là các thực phẩm dễ chế biến và phổ biến trong các bữa ăn hàng ngày. Tiếp theo là *chocolate* - một món ăn vặt được yêu thích và thường được mua để làm quà tặng. Bên cạnh đó, *green tea* và *milk* đều là những sản phẩm có giá trị dinh dưỡng và lợi ích sức khỏe cao đáp ứng nhu cầu giải khát lành mạnh, được tiêu thụ ổn định, trở thành lựa chọn quen thuộc trong đời sống hàng ngày.

Cuối cùng, dù xuất hiện với tần suất thấp hơn, nhưng các sản phẩm như *ground beef, frozen vegetables,* và *pancakes* vẫn giữ vị trí trong nhóm các sản phẩm phổ biến. Điều này có thể được lý giải bởi *ground beef* và *frozen vegetables* là các sản phẩm tiện lợi được chế biến sẵn, thường được sử dụng trong các bữa ăn nhanh, còn *pancakes* là sản phẩm phổ biến cho bữa ăn sáng hoặc các bữa ăn nhẹ trong ngày, nhờ tính dễ chế biến và giàu năng lượng.

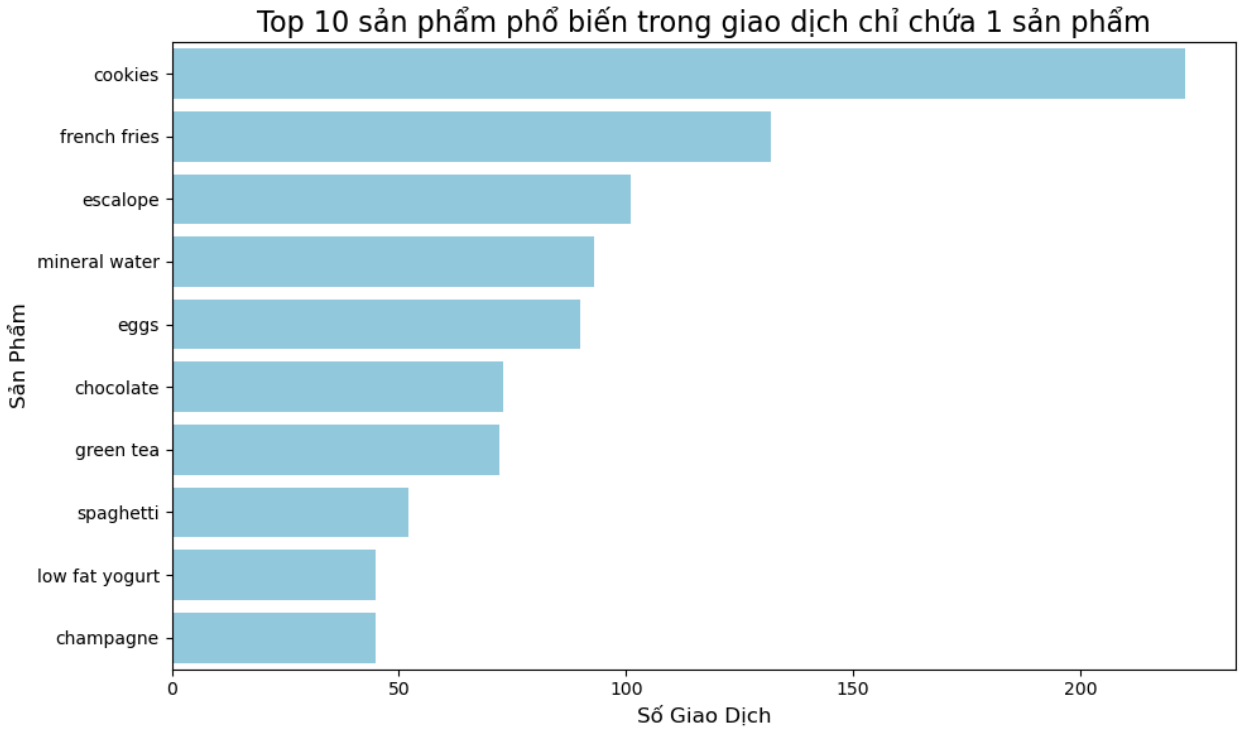
b) Phân tích các giao dịch



Biểu đồ 4.3. Biểu đồ cột thể hiện số lượng sản phẩm có trong các giao dịch

Biểu đồ 4.3 thể hiện số lượng sản phẩm có trong tất cả các giao dịch. Dựa vào biểu đồ, rút ra được các nhận xét như sau:

- Số giao dịch giảm dần khi số sản phẩm trong giao dịch tăng lên, thể hiện rằng hầu hết các giao dịch trong bộ dữ liệu chỉ chứa 01 hoặc 02 sản phẩm:
  - *Các giao dịch chứa 01 sản phẩm* chiếm số lượng lớn nhất, với hơn 1,700 giao dịch. Điều này cho thấy rằng khách hàng có xu hướng chỉ mua 1 đến 2 món khi đi siêu thị, đồng thời, phản ánh thói quen người tiêu dùng thường có xu hướng mua ít sản phẩm thay vì nhiều sản phẩm khi đi siêu thị.
- Số giao dịch chứa 10 sản phẩm trở lên chiếm tỷ lệ rất nhỏ, cho thấy khách hàng hiếm khi mua nhiều sản phẩm trong một lần giao dịch.

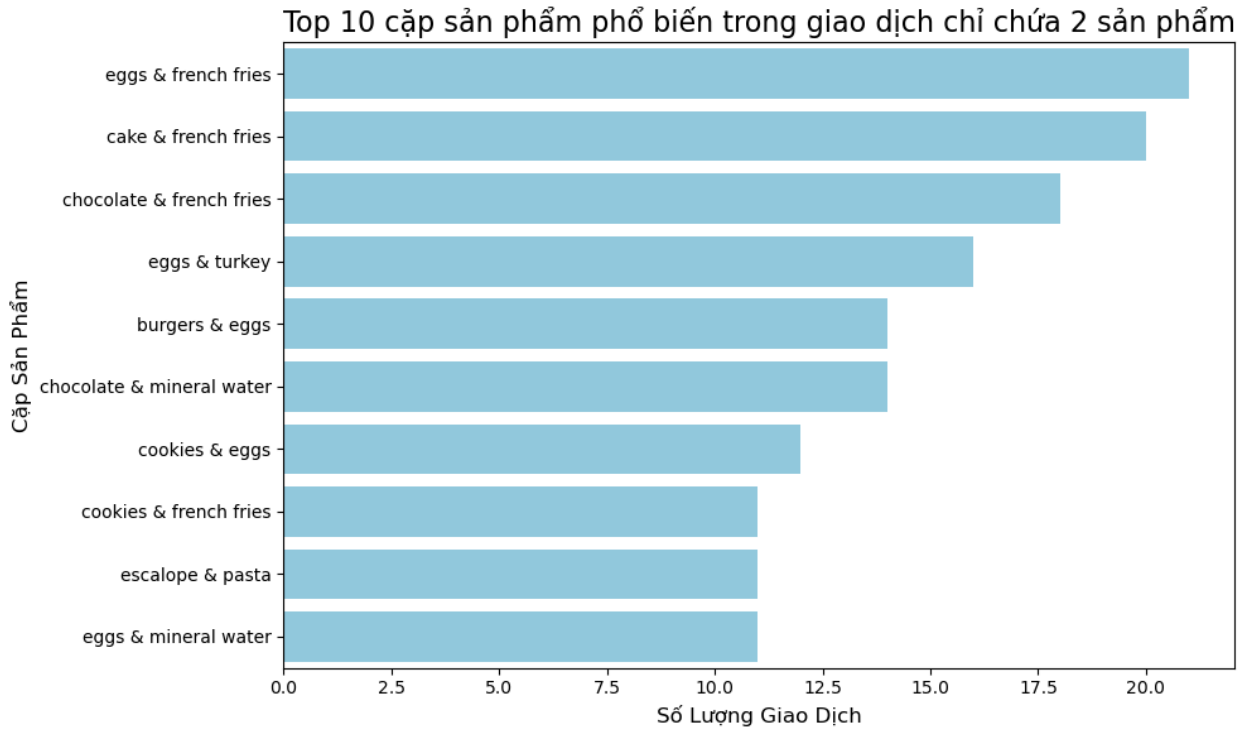


Biểu đồ 4.4. Biểu đồ cột ngang thể hiện Top 10 sản phẩm phổ biến trong giao dịch chỉ chứa một sản phẩm

Biểu đồ 4.4 thể hiện Top 10 sản phẩm phổ biến trong các giao dịch chỉ chứa một sản phẩm, với trục hoành là số lượng giao dịch, trục tung tương với với tên các sản phẩm. Dựa vào biểu đồ, nhóm đưa ra các nhận xét như sau:

- *Cookies* là sản phẩm được mua nhiều nhất, được xuất hiện trong hơn 200 giao dịch, cho thấy đây là mặt hàng được mua riêng lẻ phổ biến nhất.
- Theo sau đó là các mặt hàng như f*rench fries, escalope, mineral water* và *eggs*.
- Những sản phẩm như *chocolate, green tea, spaghetti, low fat yogurt* và *champagne* có độ phổ biến thấp hơn.

Nhìn chung, người tiêu dùng có nhu cầu mua sắm khá đa dạng, họ ưa chuộng tiêu dùng từ các sản phẩm ăn liền như *cookies, french fries* đến sản phẩm thiết yếu như *mineral water, escalope* và *eggs.* Họ chỉ mua một sản phẩm duy nhất trong giao dịch, thể hiện được hành vi mua sắm nhanh gọn, đáp ứng nhu cầu mua hàng nhanh chóng.



Biểu đồ 4.5. Biểu đồ thể hiện Top 10 cặp sản phẩm phổ biến trong giao dịch chỉ chứa hai sản phẩm

Tương tự, *Biểu đồ 4.5* chỉ ra các cặp sản phẩm phổ biến trong các giao dịch chỉ chứa 2 sản phẩm. Kết quả từ biểu đồ cho thấy:

- ➢ *'eggs & french fries'* là cặp sản phẩm phổ biến nhất, thường được mua cùng nhau trong hơn 20 giao dịch.
- ➢ Theo sau là *'cake & french fries'* và *'chocolate & french fries'* Các cặp khác như *'eggs & turkey'* và *'burgers & eggs'* cũng xuất hiện nhiều.
- ➢ Cuối cùng, cặp mặt hàng *'eggs & mineral water'* xuất hiện trong ít giao dịch nhất.

Nhìn chung, các cặp sản phẩm liên quan đến thức ăn nhanh như khoai tây chiên, bánh ngọt và trứng được khách hàng ưa thích và mua thường xuyên.

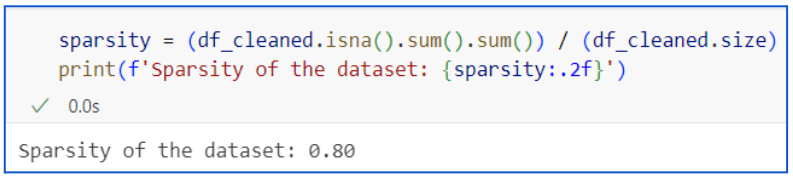
```
rare_transactions = df_cleaned[transaction_lengths <= 3]
print(f'Number of rare transactions: {len(rare_transactions)}')
```
✓ 0.0s

```
Number of rare transactions: 4156
```

Hình 4.7. Kiểm tra số lượng giao dịch hiếm (rare transactions)

Giao dịch hiếm có thể ảnh hưởng đến hiệu suất của thuật toán theo một cách đặc biệt. Những giao dịch này có ít thông tin và không chứa nhiều itemset phổ biến, dẫn đến

việc ECLAT phải kiểm tra rất nhiều giao dịch không cung cấp nhiều giá trị cho việc khai phá các quy tắc kết hợp. Vì ECLAT sử dụng phương pháp set intersection để tìm các itemset phổ biến, việc phải xử lý một lượng lớn giao dịch không có nhiều thông tin (vì số lượng món hàng trong giao dịch ít) làm cho thuật toán mất nhiều thời gian hơn để tìm ra các itemset thực sự có ý nghĩa.



Hình 4.8. Kiểm tra giá trị Sparisity của bộ dữ liệu

Sparsity cao như vậy có thể gây ảnh hưởng đáng kể đến hiệu suất của thuật toán ECLAT. ECLAT là một thuật toán khai phá quy tắc kết hợp, sử dụng các itemset phổ biến để tìm ra các mối quan hệ giữa các món hàng trong giao dịch. Khi dữ liệu có độ thưa cao, tức là hầu hết các giao dịch chỉ chứa một phần nhỏ các món hàng, thuật toán sẽ phải duyệt qua rất nhiều giao dịch và itemset có ít hoặc không có sự xuất hiện chung, điều này làm tăng đáng kể số lần tính toán cần thiết.

4.3. Xây dựng thuật toán ECLAT trên bộ dữ liệu Market Basket Analysis

Mục tiêu bài toán:

- ➢ Áp dụng thuật toán ECLAT để tìm ra luật kết hợp của bộ dữ liệu chứa thông tin sản phẩm trong các giao dịch của khách hàng nhằm xác định mối quan hệ giữa các mặt hàng, giúp hiểu sâu hơn về hành vi và xu hướng mua sắm của khách hàng.
- ➢ Dựa vào các mối quan hệ này, giúp cho doanh nghiệp đưa ra các chiến lược bán hàng thông qua việc bán các sản phẩm theo combo và quản lý hàng hóa trong kho tốt hơn.

Ngưỡng hỗ trợ tối thiểu *(minSupport)* thể hiện tỷ lệ giao dịch tối thiểu mà một tập hợp phải xuất hiện để được xem xét. Nhóm chọn giá trị *minSup = 0.02*, tức chỉ những tập hợp xuất hiện trong ít nhất 2% tổng số giao dịch sẽ được đưa vào quá trình khai phá. Đảm bảo các mẫu được lựa chọn có độ phổ biến đáng kể trong dữ liệu, đồng thời loại bỏ những mẫu hiếm ít mang lại giá trị thực tiễn.

Ngưỡng độ tin cậy tối thiểu *(minConfidence)* là yếu tố quyết định mức độ chắc chắn của các luật kết hợp. Với giá trị *minConf = 0.3*, chỉ những luật có xác suất xảy ra kết quả (confidence) đạt từ 30% trở lên mới được xem xét. Điều này giúp tập trung vào các luật kết hợp có độ tin cậy cao, mang lại thông tin có ý nghĩa trong việc tìm hiểu mối quan hệ giữa các mặt hàng.

Khi sử dụng hàm *association_rules*, nhóm cũng áp dụng thêm điều kiện *num_itemsets = 2*, yêu cầu rằng mỗi luật kết hợp phải có tổng số lượng phần tử trong tập cơ sở *(antecedents)* và tập kết quả *(consequents)* phải bằng 2. Điều kiện này giúp nhóm lọc ra các luật kết hợp không quá phức tạp để mang lại giá trị thực tiễn, dễ gom combo các sản phẩm.

```
df3 =df_cleaned
eclat = ECLAT(data=df3)
eclat.df_bin
```

Hình 4.9. Chuyển đổi df_cleaned sang dạng boolean

Nhóm đã áp dụng phương pháp chuyển đổi dữ liệu thông qua sử dụng hàm ECLAT từ thư viện pyECLAT. Mục tiêu của việc chuyển đổi này là biến đổi *df_cleaned* ban đầu thành một cấu trúc dữ liệu dạng boolean. Cụ thể, bảng dữ liệu được xây dựng với một cấu trúc mới: mỗi hàng thể hiện một giao dịch cụ thể, còn mỗi cột chính là một sản phẩm duy nhất. Với mỗi mặt hàng có mặt trong giao dịch thì sẽ có giá trị là 1, nếu không thì sẽ có giá trị là 0. Cấu trúc mới này giúp cho việc áp dụng thuật toán ECLAT một cách thuận lợi hơn.

```
# Bắt đầu thời gian tính toán
start_time = time.time()
# Đặt các tham số cho thuật toán ECLAT
min_combination = 1
max_combination = 2
# Chạy thuật toán ECLAT để khai phá các tập phổ biến
rule_indices, rule_supports = eclat.fit(
    min_support= 0.02,
    min_combination=min_combination,
    max_combination=max_combination,
    separator=', ',
    verbose=True
)
elapsed_time_eclat = time.time() - start_time
```
✓ 6.2s

Hình 4.10. Tìm các tập phổ biến bằng thuật toán Eclat

Sau khi dữ liệu được chuyển đổi sang dạng boolean, nhóm đã tiến hành huấn luyện mô hình khai phá luật kết hợp bằng thuật toán ECLAT. Thuật toán được áp dụng trên tập dữ liệu đã qua bước làm sạch và chuẩn hóa, nhằm tìm kiếm các tập phổ biến chứa từ 1 đến 2 mặt hàng xuất hiện trong các giao dịch. Kết quả trả về bao gồm danh sách các tập phổ biến cùng với giá trị mức độ hỗ trợ tương ứng. Trong quá trình thực hiện, nhóm đã chọn sẵn giá trị cho các tham số như *min_support = 0.02, min_combination = 1* và *max_combination = 2,* việc khai phá sẽ dựa vào các tham số này để đưa ra các tập phổ biến.

```
# Đo thời gian thực thi
start_time = time.time()
# Chuyển đổi các khóa của tập phổ biến thành frozenset và đưa vào danh sách itemsets
itemsets = [frozenset(i.split(', ')) for i in rule_supports.keys()]
# Tạo DataFrame với các cột 'support' và 'itemsets' từ tập phổ biến
freq_itemsets = pd.DataFrame({
    'support': list(rule_supports.values()),
    'itemsets': itemsets
})
# Khai phá các luật kết hợp từ các tập phổ biến với minimum confidence
min_confidence = 0.3
rules_eclat = association_rules(freq_itemsets, num_itemsets=2, metric="confidence", min_threshold= min_confidence)
rules_eclat = rules_eclat[rules_eclat.lift > 1]
rules_eclat = rules_eclat.reset_index(drop=True)
elapsed_time_eclat = elapsed_time_eclat + time.time() - start_time
rules_eclat
```
✓ 0.0s

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
0	(tomatoes)	(mineral water)	0.068391	0.238368	0.024397	0.356725	1.496530	1.0	0.008095	1.183991	0.356144	0.086402	0.155399	0.229537
1	(tomatoes)	(spaghetti)	0.068391	0.174110	0.020931	0.306043	1.757755	1.0	0.009023	1.190117	0.462740	0.094465	0.159746	0.213129
2	(chicken)	(mineral water)	0.059992	0.238368	0.022797	0.380000	1.594172	1.0	0.008497	1.228438	0.396502	0.082729	0.185958	0.237819
3	(cake)	(mineral water)	0.081056	0.238368	0.027463	0.338816	1.421397	1.0	0.008142	1.151921	0.322617	0.094064	0.131885	0.227014
4	(milk)	(mineral water)	0.129583	0.238368	0.047994	0.370370	1.553774	1.0	0.017105	1.209650	0.409465	0.150000	0.173315	0.285856
5	(low fat yogurt)	(mineral water)	0.076523	0.238368	0.023997	0.313589	1.315565	1.0	0.005756	1.109585	0.259747	0.082493	0.098762	0.207130
6	(olive oil)	(mineral water)	0.065858	0.238368	0.027596	0.419028	1.757904	1.0	0.011898	1.310962	0.461536	0.099759	0.237201	0.267400
7	(olive oil)	(spaghetti)	0.065858	0.174110	0.022930	0.348178	1.999758	1.0	0.011464	1.267048	0.535186	0.105651	0.210764	0.239939
8	(whole wheat rice)	(mineral water)	0.058526	0.238368	0.020131	0.343964	1.442993	1.0	0.006180	1.160960	0.326080	0.072736	0.138644	0.214208
9	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	1.0	0.018223	1.159314	0.369437	0.169312	0.137421	0.296796
10	(shrimp)	(mineral water)	0.071457	0.238368	0.023597	0.330224	1.385352	1.0	0.006564	1.137144	0.299568	0.082441	0.120604	0.214609
11	(frozen smoothie)	(mineral water)	0.063325	0.238368	0.020264	0.320000	1.342461	1.0	0.005169	1.120047	0.272346	0.072004	0.107180	0.202506
12	(pancakes)	(mineral water)	0.095054	0.238368	0.033729	0.354839	1.488616	1.0	0.011071	1.180529	0.362712	0.112544	0.152922	0.248169
13	(ground beef)	(mineral water)	0.098254	0.238368	0.040928	0.416554	1.747522	1.0	0.017507	1.305401	0.474369	0.138413	0.233952	0.294127
14	(frozen vegetables)	(mineral water)	0.095321	0.238368	0.035729	0.374825	1.572463	1.0	0.013007	1.218270	0.402413	0.119911	0.179164	0.262357
15	(soup)	(mineral water)	0.050527	0.238368	0.023064	0.456464	1.914955	1.0	0.011020	1.401255	0.503221	0.086760	0.286354	0.276610
16	(cooking oil)	(mineral water)	0.051060	0.238368	0.020131	0.394256	1.653978	1.0	0.007960	1.257349	0.416672	0.074752	0.204676	0.239354
17	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	1.0	0.013604	1.122357	0.308965	0.150648	0.109018	0.271158
18	(ground beef)	(spaghetti)	0.098254	0.174110	0.039195	0.398915	2.291162	1.0	0.022088	1.373997	0.624943	0.168096	0.272197	0.312015
19	(burgers)	(eggs)	0.087188	0.179709	0.028796	0.330275	1.837830	1.0	0.013128	1.224818	0.499424	0.120941	0.183552	0.245256

Hình 4.11. Các luật phổ biến của tập dữ liệu

Để triển khai các luật kết hợp, nhóm bắt đầu bằng cách chuyển đổi các khóa trong tập phổ biến thành frozenset nhằm sử dụng chúng như những khóa duy nhất. Các frozenset này sau đó được thêm vào danh sách itemsets cùng với giá trị hỗ trợ *(support)* tương ứng.

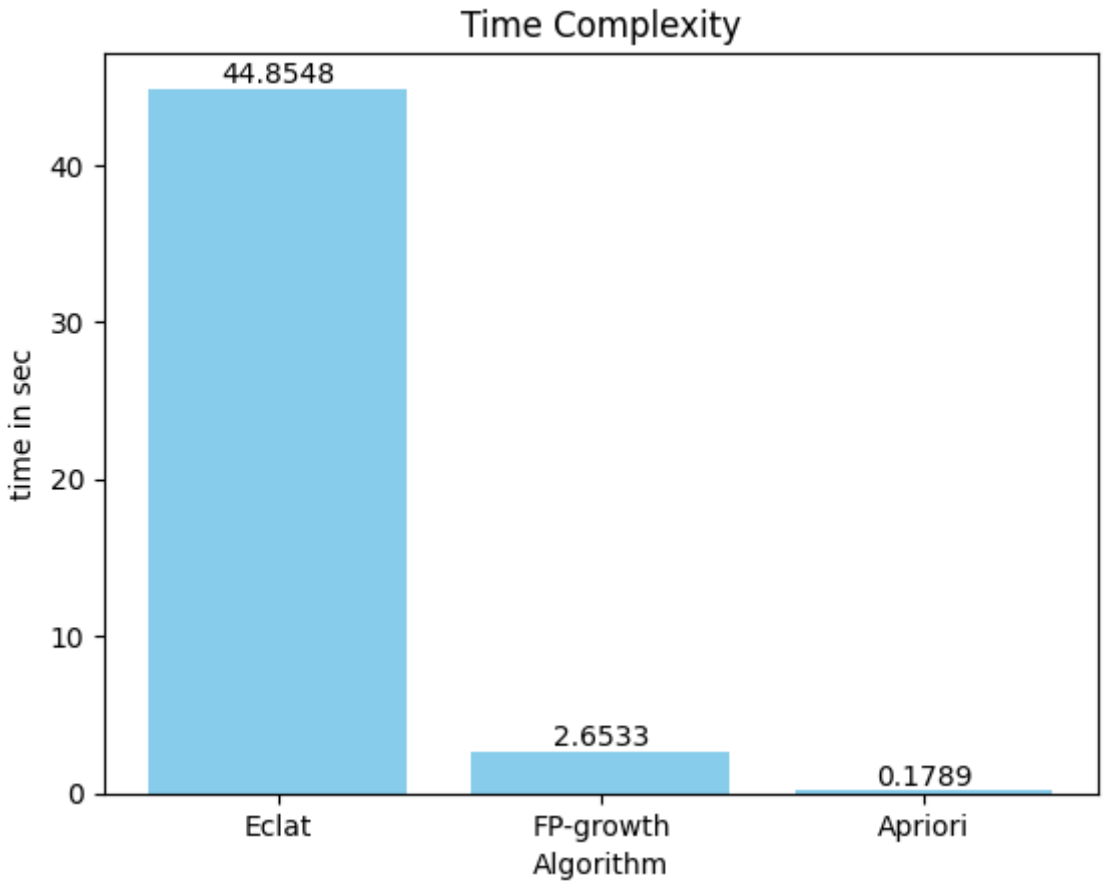
Dựa trên danh sách này, nhóm tạo ra một DataFrame tên là *freq_itemsets*, bao gồm hai cột:

- ➢ *'support'* chứa giá trị hỗ trợ của từng tập phổ biến
- ➢ *'itemsets'* chứa các frozenset đại diện cho các tập phổ biến.

Tiếp theo, nhóm sử dụng hàm *association_rules* từ thư viện *mlxtend* để khai phá các luật kết hợp từ tập phổ biến đã xác định, với *minConf = 0.03, num_itemsets = 2* và *lift > 1*. Các luật kết hợp được khai phá có số phần tử là 2 sẽ được lọc và sắp xếp dựa trên mức độ liên quan và độ tin cậy, sau đó được lưu trữ vào *rules_eclat*.

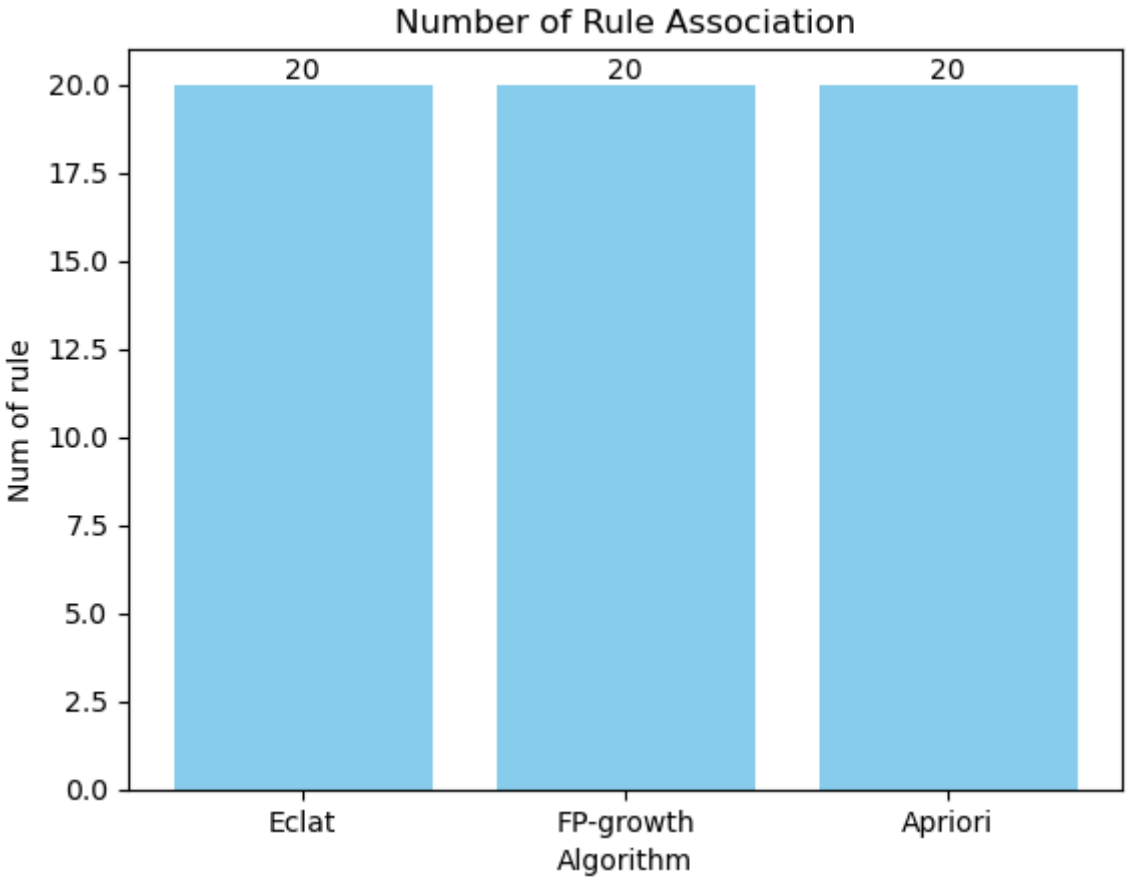
4.4. So sánh thuật toán ECLAT với hai thuật toán Apriori và FP-Growth

Sau khi tìm kiếm luật kết hợp với thuật toán ECLAT, nhóm tiến hành so sánh với hai thuật toán phổ biến khác là Apriori và FP-Growth bằng cách chạy hai thuật toán (Apriori và FP-Growth) từ thư viện *mlxtend* với bộ dữ liệu chứa thông tin các sản phẩm trong các giao dịch của khách hàng. Nhóm tiến hành so sánh thời gian và luật kết hợp của 3 thuật toán với *minSup = 0.02, minConf = 0.3, num_itemsets = 2* và *lift > 1*.



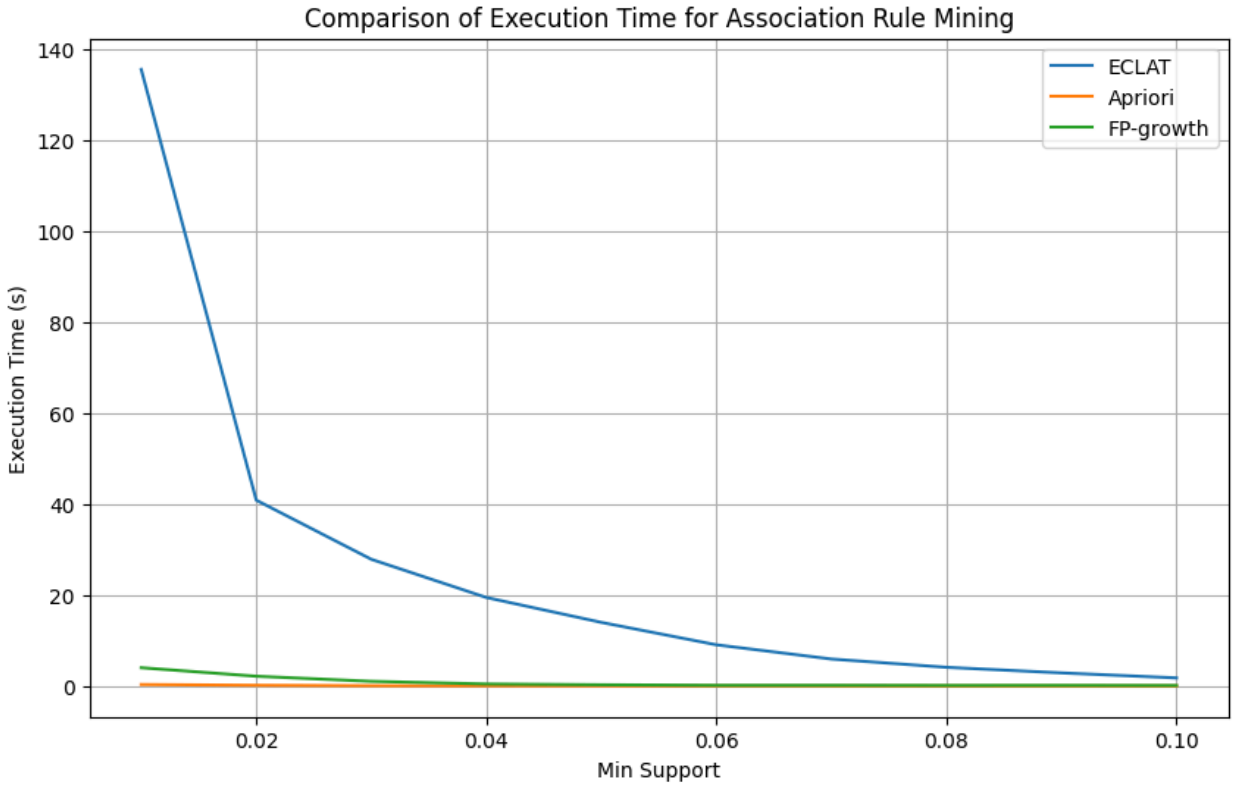
Biểu đồ 4.6. Biểu đồ so sánh thời gian thực hiện của ba thuật toán

Thuật toán ECLAT có thời gian thực hiện lớn nhất, lên đến 44.8548 giây. Trong khi đó, FP-Growth chỉ mất 2.6533 giây, nhanh hơn đáng kể so với Eclat. Thuật toán Apriori vượt trội nhất về thời gian thực hiện với chỉ 0.1789 giây, trở thành thuật toán tối ưu về thời gian với bộ dữ liệu này.



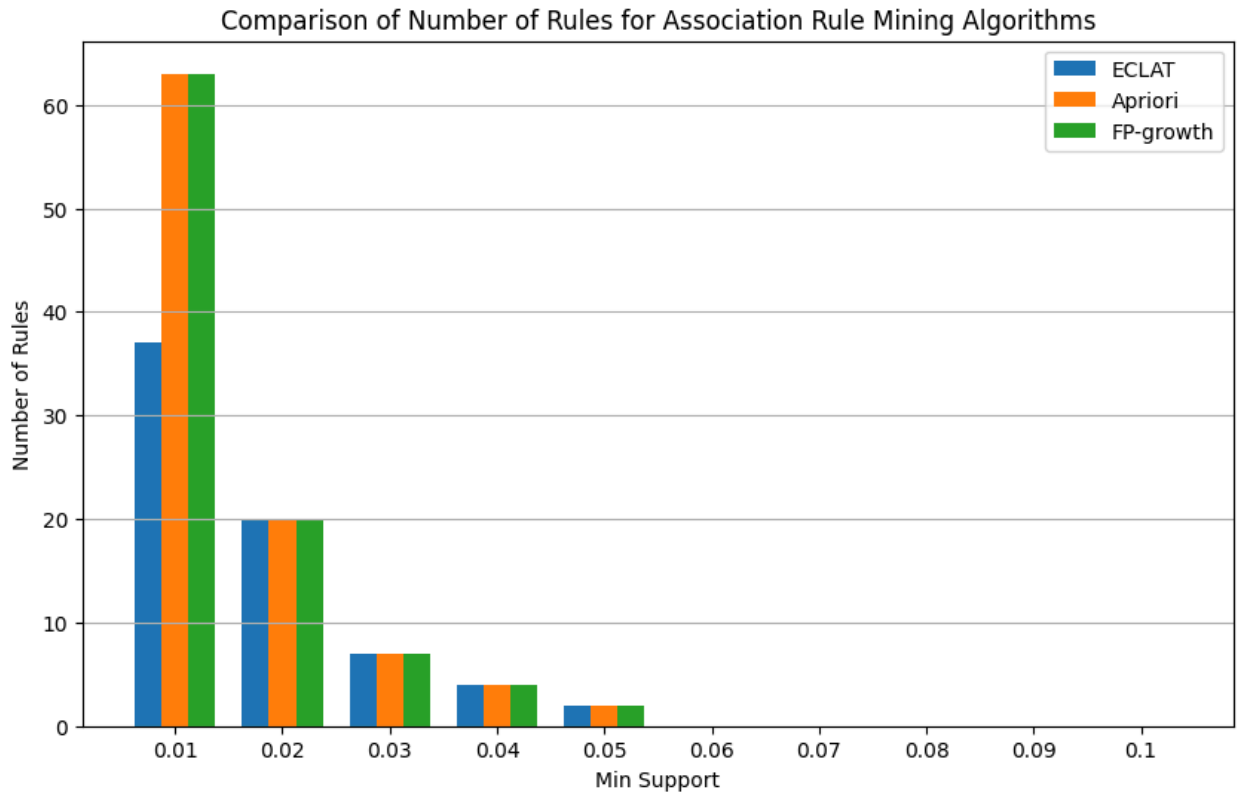
Biểu đồ 4.7. Biểu đồ so sánh số lượng luật kết hợp của ba thuật toán

Cả ba thuật toán ECLAT, FP-Growth, và Apriori đều tạo ra số lượng luật kết hợp giống nhau. Kết quả này cho thấy mặc dù có sự khác biệt về thời gian thực hiện, khả năng khai phá các tập phổ biến và khai thác luật kết hợp của các thuật toán đều đạt được kết quả tương đồng, từ đó thuật toán Apriori được xem như là thuật toán hiệu quả nhất bởi khai phá cùng số luật kết hợp với thời gian ngắn nhất.



Biểu đồ 4.8. Biểu đồ so sánh thời gian thực hiện của ba thuật toán

Biểu đồ so sánh thời gian thực thi của các thuật toán với các giá trị *minConf = 0.3, num_itemsets = 2,* và giá trị *minSup* nằm trong khoảng (0.01, 0.1). Nhận xét chung, khi giá trị *minSup* tăng lên thì thời gian thực hiện của tất cả các thuật toán đều giảm đáng kể. Trong đó, thuật toán Apriori vẫn là thuật toán có thời gian ít nhất, khi *minSup* càng cao thì Apriori và FP-Growth gần như cùng thời gian.



Biểu đồ 4.9. Biểu đồ so sánh số lượng luật của ba thuật toán

Biểu đồ 4.9 so sánh số luật kết hợp được tạo ra của mỗi thuật toán tương ứng với các giá trị *minSup* khác nhau. Kết quả cho thấy:

- Khi giá trị *minSup* tăng, số lượng luật kết hợp được tạo ra bởi cả ba thuật toán đều giảm.
- Khi *minSup = 0.01*, số lượng luật tạo ra bởi thuật toán ECLAT lại ít hơn so với các thuật toán còn lại.

Nguyên nhân có thể đến từ cách hoạt động của thuật toán, ECLAT sử dụng kỹ thuật đếm hỗ trợ và cắt tỉa (pruning) để loại bỏ nhanh chóng các mục không thường xuyên không thỏa mãn giá trị *minSup*. Điều này giúp nó giảm số lượng ứng viên và luật kết hợp, tạo ra ít luật hơn so với Apriori và FP-Growth khi giá trị *minSup* thấp.

CHƯƠNG 5 - KẾT LUẬN

Sau khi thực hiện đồ án nghiên cứu về thuật toán khai phá luật kết hợp ECLAT, nhóm đã hoàn thành việc triển khai và xây dựng thuật toán với bộ dữ liệu *Market Basket Analysis 1*, đồng thời thực hiện việc đánh giá hiệu suất của thuật toán ECLAT so với thuật toán Apriori và FP-Growth. Trong quá trình đánh giá hiệu suất giữa các thuật toán, nhóm đã thực hiện so sánh đầy đủ các yếu tố từ thời gian thực hiện đến số lượng luật kết hợp được tạo ra bởi ba thuật toán này để có một đánh giá trực quan nhất. Dưới đây là đánh giá của nhóm về những ưu điểm và hạn chế của đồ án nghiên cứu:

Ưu điểm của đồ án nghiên cứu:

- ➢ Nhóm đã tìm hiểu và nghiên cứu về cách áp dụng cũng như xây dựng thuật toán ECLAT, cụ thể nhóm đã thực hiện xây dựng thuật toán bằng cách sử dụng thư viện cho bộ dữ liệu Market Basket Analysis 1 để đưa ra thông tin về mối quan hệ của các sản phẩm trong giỏ hàng từ đó làm cơ sở cho việc xây dựng các chương trình khuyến mãi, bố trí sản phẩm trên kệ, quản lý hàng hóa, bán sản phẩm theo combo nhằm cải thiện doanh thu cho cửa hàng.
- ➢ Nhóm cũng đã thực hiện tiền xử lý dữ liệu một cách kỹ càng để có bộ dữ liệu sạch cho việc xây dựng thuật toán, trực quan hóa dữ liệu qua đa dạng các biểu đồ và các phương diện khác nhau để có cái nhìn tổng quan cũng như hiểu rõ hơn về bộ dữ liệu.

Hạn chế của đồ án nghiên cứu:

- ➢ Dù nhóm đã nỗ lực trong việc phân tích thuật toán, tuy nhiên đồ án nghiên cứu về thuật toán vẫn chưa được tối ưu về thời gian xử lý dữ liệu dẫn đến thời gian thực hiện của thuật toán ECLAT còn khá chậm, một phần có thể thuật toán ECLAT phù hợp với bộ dữ liệu dày đặc, nhưng với bộ dữ liệu hiện tại thì phân bố dữ liệu lại khá thưa trong mỗi giao dịch do đó có thể ảnh hưởng đến hiệu suất của thuật toán.
- ➢ Bên cạnh đó, một trong những thiếu sót của nhóm là chưa đưa ra được những hướng cải tiến để cải thiện hiệu suất cho quá trình xây dựng thuật toán ECLAT để khai thác mẫu phổ biến của bộ dữ liệu.

Tổng kết lại, thông qua quá trình thực hiện đồ án, nhóm đã có được những kiến thức nền tảng về thuật toán ECLAT, cũng như hiểu được cách ứng dụng thuật toán này vào bộ dữ liệu thực tế. Tuy nhiên với sự thiếu sót về kinh nghiệm và hạn chế về mặt thời gian nên hiệu suất của thuật toán trong đồ án nghiên cứu vẫn chưa thực sự tối ưu, nhóm sẽ cần phải nghiên cứu và tìm hiểu kỹ hơn về cách thức tổ chức dữ liệu cũng như quá trình thuật toán hoạt động đối với các dạng dữ liệu để có thể cải thiện bài nghiên cứu tốt hơn.

TÀI LIỆU THAM KHẢO

(2024).Humg.edu.vn.https://qlkh.humg.edu.vn/KhoaHocKhac/Download/1825?FileName=PACuong_TIM_HIEU_THUAT_TOAN_APRIORI.docx

Agrawal, R. "Fast Algorithms for Mining Association Rules." VLDB, 1994.

Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data (pp. 207-216).

dqtphu30. (2024). GitHub - dqtphu30/eclat-algorithm: Market Basket Analysis using ECLAT Algorithm. GitHub. https://github.com/dqtphu30/eclat-algorithm/tree/main

Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery, 8, 53-87.

Huấn, P. T. (2017). THUẬT TOÁN HIỆU QUẢ KHAI THÁC TẬP PHỔ BIẾN TỐI ĐẠI TRÊN CƠ SỞ DỮ LIỆU GIAO DỊCH LỚN. PROCEEDING of Publishing House for Science and Technology.

ML | ECLAT Algorithm. (2019, June 11). GeeksforGeeks. https://www.geeksforgeeks.org/ml-eclat-algorithm/

ngocnhcd. (2024, October 26). Thương mại điện tử xuyên biên giới tạo cơ hội cho doanh nghiệp Việt tăng trưởng. Https://Dangcongsan.vn. https://dangcongsan.vn/kinh-te-va-hoi-nhap/thuong-mai-dien-tu-xuyen-bien-gioi-tao-co-hoi-cho-doanh-nghiep-viet-tang-truong-681913.html

Nguyễn, T. V. H. (2023). Tổng quan về khai phá dữ liệu và phương pháp khai phá luật kết hợp trong cơ sở dữ liệu = An overview on data mining and the association rule in data mining.

Richard, J. (n.d.). pyECLAT: A package for association analysis using the ECLAT method. PyPI. https://pypi.org/project/pyECLAT/

Srinadh, V. (2022). Evaluation of Apriori, FP growth and Eclat association rule mining algorithms. International journal of health sciences, (II), 7475-7485.

Zaki, M. J. (2000). Scalable algorithms for association mining. IEEE transactions on knowledge and data engineering, 12(3), 372-390.

Zaki, Mohammed Javeed, et al. "New algorithms for fast discovery of association rules." KDD. Vol. 97. 1997.