UEH UNIVERSITY
ĐẠI HỌC UEH
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

ĐỒ ÁN CUỐI KÌ MÔN HỌC
KHAI PHÁ DỮ LIỆU

ĐỀ TÀI: NGHIÊN CỨU THUẬT TOÁN ECLAT & ỨNG DỤNG PHÂN TÍCH GIỎ HÀNG

GIẢNG VIÊN HƯỚNG DẪN: TS. HUỲNH VĂN ĐỨC

Nhóm sinh viên thực hiện	
Họ và tên	**MSSV**
Mai Hiển Đạt	31211021121
Nguyễn Tố Bình	31211021118
Phạm Minh Toàn	31211021175
Nguyễn Hồng Trâm	31211024053

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG BIỂU

BẢNG ĐÁNH GIÁ CÔNG VIỆC

Sinh viên	Công việc	Tỷ lệ đóng góp
Phạm Minh Toàn	2.3.1. Khái niệm ECLAT 2.3.2. Ý tưởng ECLAT 4. Thực hiện thuật toán 5. Kết luận Nhận xét, hoàn thiện nội dung báo cáo	100%
Nguyễn Tố Bình	1. Giới thiệu đề tài 2.2.1. Giới thiệu luật kết hợp 2.2.2. Các khái niệm liên quan 2.3.5. Minh họa thực hiện thuật toán. 2.3.6. So sánh thuật toán Apriori.	100%
Mai Hiển Đạt	2.1. Khai phá dữ liệu 2.2.3. Ứng dụng luật kết hợp 2.3.4. Quy trình xây dựng thuật toán Hoàn thiện hình thức báo cáo	100%
Nguyễn Hồng Trâm	2.3.3. Cấu trúc dữ liệu 3. Tổng quan bộ dữ liệu 4.2. Tìm tập phổ biến Slide thuyết trình	90%

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.

1.1. Lý do chọn đề tài

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, việc khai thác và phân tích dữ liệu trở thành một nhu cầu thiết yếu đối với các tổ chức và doanh nghiệp. Khai phá dữ liệu (data mining) đóng vai trò quan trọng trong việc tìm kiếm các thông tin có giá trị từ khối lượng dữ liệu khổng lồ. Một trong những nhiệm vụ phổ biến trong khai phá dữ liệu là tìm kiếm các tập mục phổ biến (frequent itemsets) từ cơ sở dữ liệu giao dịch, giúp phát hiện ra các mẫu và mối quan hệ ẩn giấu.

Trong bối cảnh này, phân tích giỏ hàng (market basket analysis) nổi lên như một công cụ mạnh mẽ trong việc phân tích hành vi mua sắm khách hàng. Phân tích giỏ hàng giúp xác định các mặt hàng thường được mua cùng nhau, từ đó hỗ trợ cho các nhà bán lẻ trong việc tối ưu hóa bày trí sản phẩm, xây dựng chiến lược và cải thiện doanh thu.

Thuật toán ECLAT được đề xuất như một giải pháp hiệu quả để giải quyết bài toán này. So với các thuật toán khác như Apriori, ECLAT được biết đến với khả năng xử lý dữ liệu nhanh chóng và hiệu quả hơn trong một số trường hợp, đặc biệt là với dữ liệu dày đặc. Vì vậy, trong bài nghiên cứu về thuật toán ECLAT không chỉ giúp hiểu rõ hơn về thuật toán ECLAT và ứng dụng trong thực tế.

1.2. Mục tiêu nghiên cứu.

- Định nghĩa và ý tưởng cơ bản của thuật toán ECLAT.
- So sánh thuật toán ECLAT với thuật toán Apriori qua các tiêu chí nhất định.
- Thiết kế lớp và quy trình giải thuật cũng như cách thức hoạt động của ECLAT.
- Ứng dụng thực tế của thuật toán trong khai thác dữ liệu mua hàng đưa ra tập mục phổ biến, khai phá luật kết hợp. Việc này giúp tối ưu hóa sắp xếp sản phẩm, cải thiện kho hàng và cá nhân hóa trải nghiệm mua hàng.

1.3. Phương pháp nghiên cứu.

Để đạt được các mục tiêu nghiên cứu đã đề ra, phương pháp nghiên cứu sẽ bao gồm các bước sau:

- Phương pháp thống kê (Statistical Method): Thực hiện các kỹ thuật thống kê để phân tích và đánh giá kết quả của thuật toán ECLAT, như tính toán độ hỗ trợ, độ tin cậy và các chỉ số liên quan khác.

- Phương pháp mô hình hóa (Modeling Method): Áp dụng UML (Unified Modeling Language) để thiết kế và mô tả cấu trúc của thuật toán ECLAT. Mô hình hóa quá trình khai phá tập phổ biến và sinh luật kết hợp trong ECLAT để hiểu rõ hơn về cách thức hoạt động của thuật toán.
- Phương pháp mô phỏng (Simulation Method): Xây dựng ngưỡng các chỉ số quan trọng của thuật toán ECLAT để hỗ trợ khai thác luật kết hợp.

1.4. Tài nguyên sử dụng.

- Ngôn ngữ lập trình: Python.
- Thư viện: pandas, numpy, seaborn, matplotlib, mlxtend.
- Nguồn dữ liệu: Kaggle (Link).

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.

2.1. Khai phá dữ liệu.

2.1.1. Khái niệm

Khai phá dữ liệu (Data Mining) là quá trình khám phá các mẫu hình, mối tương quan và xu hướng có ý nghĩa từ các tập dữ liệu lớn, phức tạp thông qua việc áp dụng các kỹ thuật từ thống kê, máy học và cơ sở dữ liệu mà không thể dễ dàng phát hiện bằng phương pháp thủ công. Mục tiêu chính của khai phá dữ liệu là trích xuất thông tin hữu ích và tri thức mới từ dữ liệu thô, hỗ trợ ra quyết định và cải thiện hiệu suất trong nhiều lĩnh vực. So với truy vấn dữ liệu truyền thống, khai phá dữ liệu khác biệt bởi khả năng khám phá ra các mẫu, xu hướng, và mối quan hệ ẩn chứa bên trong khối dữ liệu. Khai phá dữ liệu được ứng dụng để dự đoán các sự kiện trong tương lai dựa trên dữ liệu lịch sử, phân loại dữ liệu thành các nhóm dựa trên đặc điểm chung, gom nhóm các đối tượng tương đồng, nhận diện các điểm dữ liệu bất thường, hay khai phá luật kết hợp.

2.1.2. Quy trình thực hiện.

- **Thu thập dữ liệu**

Là quá trình thu thập các nguồn dữ liệu phù hợp với mục tiêu nghiên cứu. Các nguồn dữ liệu bao gồm cơ sở dữ liệu, tệp, văn bản, âm thanh, hình ảnh, video,…

- **Tiền xử lý dữ liệu**
  - **Làm sạch dữ liệu**: Xử lý dữ liệu khuyết, dữ liệu nhiễu, dữ liệu bất thường.
  - **Tích hợp dữ liệu**: Kết hợp các nguồn dữ liệu khác nhau vào một tập hợp dữ liệu thống nhất.

- **Chuyển đổi dữ liệu**: Chuyển đổi dữ liệu thô thành dạng bảng phù hợp cho quá trình phân tích
- **Giảm chiều dữ liệu**: Giảm số lượng biến trong dữ liệu mà vẫn giữ được các thông tin cần thiết để giải quyết mục tiêu nghiên cứu

- **Áp dụng các kỹ thuật Data Mining**
  - **Phân cụm (Clustering)**: Nhóm các đối tượng có tính chất tương tự với nhau. Sử dụng các kỹ thuật như: K-means, DBSCAN, hierarchical clustering
  - **Phân loại (Classification)**: Xác định lớp/ nhãn cho các đối tượng mới dựa trên các mẫu đã học. Sử dụng các kỹ thuật như: Cây quyết định, Hồi quy Logistic, SVM, Neural Networks.
  - **Hồi quy (Regression)**: Dự đoán giá trị số liệu liên tục dựa trên các kỹ thuật: Hồi quy tuyến tính (đơn biến, đa biến).
  - **Luật kết hợp (Association Rules)**: Tìm mối tương quan giữa các item dựa trên các kỹ thuật như Apriori, FP-Growth, ECLAT.
  - **Phát hiện bất thường (Anomaly Detection)** : Phát hiện các quan sát khác biệt so với mẫu dữ liệu thông thường dựa trên các kỹ thuật: Isolation Forest, One-Class SVM.
  - **Dự báo (Prediction)**: Dự đoán các sự kiện hoặc giá trị trong tương lai bằng cách sử dụng các kỹ thuật: mô hình chuỗi thời gian (ARIMA), mạng nơ-ron nhân tạo.
- **Đánh giá, mô tả và trực quan hóa kết quả**

Đánh giá hiệu suất các mô hình Data Mining dựa trên các tiêu chí phù hợp với mục tiêu nghiên cứu. Sau đó, mô tả diễn giả các kết quả phân tích. Cuối cùng, trực quan hóa kết quả bằng các biểu đồ, đồ thị dễ hiểu và dễ truyền đạt.

2.1.3. Ứng dụng.

- **Kinh doanh**: Phân tích hành vi khách hàng, quản lý quan hệ khách hàng, dự đoán xu hướng thị trường, phát hiện gian lận, rủi ro tín dụng và tối ưu hóa chuỗi cung ứng.
- **Y tế**: Hỗ trợ chẩn đoán bệnh, phát triển thuốc, khai phá dữ liệu gen, phân tích sinh trắc học và quản lý sức khỏe dựa trên dữ liệu bệnh nhân và hồ sơ y tế.
- **Khoa học kỹ thuật**: Dự báo thời tiết, phân tích khí hậu, Nhận dạng mẫu trong xử lý ảnh, thị giác máy tính, robot.
- **An ninh mạng**: Phát hiện xâm nhập, tấn công mạng, mã độc, botnet.

2.2. Khai phá luật kết hợp.

2.2.1. Giới thiệu.

Khai phá luật kết hợp (Association Rule Mining) là một quy trình phân tích dữ liệu nhằm tìm ra mẫu tương quan và quy tắc kết hợp của các mục trong một tập dữ liệu. Mô hình đầu tiên của bài toán "Khai phá luật kết hợp" là mô hình nhị phân được đề xuất bởi Agrawal et al. (1993), xuất phát từ nhu cầu phân tích dữ liệu của cơ sở dữ liệu giao tác, phát hiện các mối quan hệ giữa các tập mục hàng hóa (Itemsets) đã bán được tại các siêu thị. Mục tiêu của khai phá luật kết hợp là tìm hiểu và khám phá những mô hình, mẫu chuẩn và quy tắc ẩn bên trong dữ liệu để từ đó rút ra thông tin giá trị và hỗ trợ quyết định.

2.2.2. Các khái niệm liên quan.

- Tập mục (itemset) là một tập hợp các items mà chúng ta sẽ xét đồng thời trong một giao dịch hoặc giỏ hàng. Nếu coi một tập hợp các mục items là A = {a1, a2, a3,...an} thì một itemset sẽ là tập con của A. Trong trường hợp A = {a1, a2, a3, a4}, một itemset có thể là {a1, a2}.
- Tập phổ biến (frequent itemset) là một itemset xuất hiện với tần suất tối thiểu (minimum support) trong một tập hợp các giao dịch. Để xác định được tập phổ biến, cần xác định ngưỡng hỗ trợ (support threshold). Biểu diễn công thức:

$$Support(A) = \frac{Số\ lượng\ giao\ dịch\ chứa\ A}{Tổng\ số\ giao\ dịch}$$

Trong đó:

- A: một item hoặc item set.
- Số lượng giao dịch chứa A: số giao dịch trong tập dữ liệu mà A xuất hiện.
- Tổng số giao dịch: tổng số giao dịch trong tập dữ liệu.

Vì vậy Item là tập phổ biến nếu: Support(A) $\geq$ minimum support.

- Luật kết hợp:
  - Khái niệm: Luật kết hợp là một dạng quan hệ "nếu - thì" giữa các itemset trong tập hợp các giao dịch, giúp xác định quan hệ tồn tại giữa các itemset. Một luật kết hợp có thể có dạng B $\Rightarrow$ A, trong đó A, B là các itemset.
  - Các chỉ số để đánh giá luật kết hợp bao gồm Độ tin cậy (Confidence) và Lift.

- Độ tin cậy (Confident): đo lường độ tin cậy vào một luật cụ thể. Dưới đây là công thức tính Confidence của tỉ lệ các giao dịch chứa itemset A cũng chứa itemset B trong luật A → B:

$$\text{Confidence}(\text{A} \rightarrow \text{B}) = \frac{Support(A \cup B)}{Support(A)}$$

Trong đó:

- $Support(A \cup B)$: tần suất xuất hiện giao dịch chứa cả A và B trong tổng số giao dịch
- $Support(A)$: tần suất xuất hiện của A trong tổng số giao dịch.

- Lift: Đo lường mức độ tăng cường của một luật so với tần suất xuất hiện ngẫu nhiên. Công thức dưới đây mô tả tỷ lệ tăng trưởng của xác suất B xuất hiện khi biết A đã xuất hiện, so với khi không biết A:

$$Lift(\text{A} \rightarrow \text{B}) = \frac{Support(A \cup B)}{Support(A) \times Support(B)}$$

Trong đó:

- Lift > 1 mối quan hệ dương, cho thấy sự xuất hiện của A làm tăng khả năng xuất hiện của B
- Lift < 1 cho thấy mối quan hệ tương quan âm, cho thấy sự xuất hiện của A làm giảm khả năng xuất hiện của B
- Lift = 1 cho thấy không mối quan hệ giữa A và B.

2.2.3. Ứng dụng.

Luật kết hợp là một công cụ mạnh mẽ trong lĩnh vực Khai phá dữ liệu, được ứng dụng rộng rãi trong nhiều lĩnh vực, mang lại lợi ích thiết thực cho nhiều ngành nghề.

- **Marketing**. Thông qua việc phân tích giỏ hàng (Market Basket Analysis), Association Rule có thể phát hiện ra các sản phẩm được mua cùng lúc với tần suất cao. Ví dụ, một siêu thị có thể phát hiện ra rằng khách hàng mua sữa thường cũng mua thêm bánh quy. Từ đó, siêu thị có thể đưa ra các chiến lược marketing hiệu quả hơn, chẳng hạn như đặt sữa và bánh quy cạnh nhau trên kệ hàng hoặc đưa ra các khuyến mãi hấp dẫn khi mua cả hai sản phẩm.
- **Y tế**. Luật kết hợp có thể được sử dụng để phân tích dữ liệu bệnh nhân ung thư và xác định các yếu tố liên quan đến nguy cơ mắc bệnh, giúp các bác sĩ đưa ra chẩn đoán chính xác hơn và điều trị hiệu quả hơn.

- **Khoa học máy tính:** Ví dụ, trong lĩnh vực khoa học máy tính, Luật kết hợp có thể được sử dụng để phân error log và tìm ra các lỗi phần mềm tiềm ẩn.
- **Quản lý chuỗi cung ứng:** Khai phá luật kết hợp có thể giúp phân tích dữ liệu về quá trình sản xuất và phân phối để tìm ra các mẫu tương quan giữa các yếu tố như thời gian, vị trí, và quy trình. Điều này giúp cải thiện quy trình vận hành và tối ưu hóa chuỗi cung ứng.

2.3. Thuật toán ECLAT.

2.3.1. Khái niệm.

ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) là một thuật toán khai phá luật kết hợp được giới thiệu bởi Zaki et al. (1997). Mục đích chính của ECLAT là tìm kiếm tất cả các tập mặt hàng phổ biến trong một cơ sở dữ liệu giao dịch với độ hỗ trợ (support) lớn hơn hoặc bằng một ngưỡng cho trước.

ECLAT sử dụng cách tiếp cận dựa trên cơ sở dữ liệu dọc để khai phá các tập phổ biến. Thay vì sử dụng cơ sở dữ liệu giao dịch theo chiều ngang như thuật toán Apriori, ECLAT chuyển đổi cơ sở dữ liệu sang định dạng dọc, trong đó mỗi mặt hàng được liên kết với tập hợp các ID giao dịch (TID) mà mặt hàng đó xuất hiện. Điều này cho phép ECLAT tính toán độ hỗ trợ của các tập mặt hàng bằng cách sử dụng phép giao trên các danh sách TID, thay vì quét toàn bộ cơ sở dữ liệu giao dịch như Apriori.

Do cách thức hoạt động chính của thuật toán ECLAT là lấy giao các TID-set và kích thước của TID-set là một trong những yếu tố chính ảnh hưởng đến thời gian chạy và sử dụng bộ nhớ của ECLAT. TID-set càng lớn thì càng cần nhiều thời gian và bộ nhớ (Bakar et al, 2016).

TID	Items
1	b, d
2	a, c, d
3	a, b
4	c, d
5	a, b, d

Bảng 2-1. Cơ sở dữ liệu ngang.

Item	TID set
A	2, 3, 5
B	1, 3, 5
C	2, 4
D	1, 2, 4, 5

Bảng 2-2. Cơ sở dữ liệu dọc

2.3.2. Ý tưởng.

Ý tưởng của thuật toán ECLAT trong khai phá tập phổ biến, bước quan trọng trước khi khai phá luật kết hợp được mô tả như sau:

- Biểu diễn danh sách TID dưới dạng cơ sở dữ liệu dọc: Mỗi mặt hàng được liên kết với một tập hợp các TID mà mặt hàng đó xuất hiện.
- Thực hiện phép giao trên các danh sách TID: Tiến hành phép giao trên các danh sách TID tương ứng của các mặt hàng trong tập đó để tính toán độ hỗ trợ.
- Gọi hàm đệ quy: Trong mỗi lần gọi đệ quy, bắt đầu từ các tập mặt hàng có độ dài 1 (1-itemsets) và lặp lại quá trình mở rộng các tập mặt hàng bằng cách kết hợp chúng với các tập mặt hàng khác để tạo ra các tập mặt hàng lớn hơn (k-itemsets).

2.3.3. Cấu trúc dữ liệu.

a) *Từ điển (Dictionary).*

Được sử dụng để lưu trữ cơ sở dữ liệu dọc

Khóa (key) của từ điển là tên mặt hàng và giá trị (value) là một tập hợp (set) các TID

Cho phép truy cập nhanh đến các TID của một mặt hàng cụ thể thông qua tên mặt hàng.

b) *Tập hợp (Set).*

Lưu trữ các TID cho mỗi mặt hàng trong cơ sở dữ liệu dọc.

Cho phép truy cập và so sánh nhanh các TID, đồng thời loại bỏ các TID trùng lặp.

c) Danh sách (List).

Lưu trữ các giao dịch ban đầu trước khi chuyển đổi sang cơ sở dữ liệu dọc.

Mỗi giao dịch có thể được biểu diễn bằng một danh sách các mặt hàng.

d) Cây (Tree)

Một cấu trúc cây phổ biến là cây Itemset-TIDset (IT-tree), trong đó mỗi nút đại diện cho một tập mặt hàng và chứa thông tin về tập TID của tập mặt hàng đó.

Cây IT-tree được xây dựng và mở rộng dựa trên quá trình tìm kiếm theo chiều sâu (depth-first search).

Cây *giúp tổ chức và truy xuất hiệu quả các tập phổ biến và TID của chúng*

e) Đệ quy (Recursion)

Cho phép thuật toán duyệt qua không gian tìm kiếm một cách hiệu quả và tạo ra các tập mặt hàng mở rộng từ các tập mặt hàng hiện có.

Mỗi lần gọi đệ quy tương ứng với việc mở rộng một tập mặt hàng và tìm kiếm các tập con phổ biến của nó.

Lưu trữ các trạng thái của thuật toán và thực hiện quá trình quay lui qua ngăn xếp (stack).

2.3.4. Quy trình xây dựng thuật toán.

Dưới đây là một đoạn mã giả minh họa quy trình xây dựng giải thuật ECLAT (Man & Jalil, 2019).

INPUT: a file D consisting of baskets of items, a support threshold , and an item prefix I, such that I ⊆ J.

OUTPUT: A list of itemsets F[I](D,σ) for the specified prefix

METHOD:

F(i) ← {}

for all i ∈ J occurring in D **do**

$F[I] \coloneqq F[I] \cup \{I \cup \{i\}\}$

#Create D_i

$D_i \leftarrow \{\}$

for all j ∈ J occurring in D such that j > i **do**

C ← cover({i}) ∩ cover({j})

if $|C| > \sigma$ **then**

$D_i \leftarrow D_i \cap \{j, C\}$

#Depth-first recursion

Compute $F[I \cap i](D_i, \sigma)$

$F[I] \coloneqq F[I] \cup F[I \cup i]$

Bảng 2-3. Mã giả của thuật toán ECLAT

Đoạn mã giả trên được giải thích tuần tự theo các bước như sau:

1. Khởi tạo tập kết quả $F(i)\{\}$: Trước khi thực hiện vòng lặp bên dưới, khởi tạo tập kết quả $F(i)\{\}$: để lưu trữ các tập con có tiền tố I
2. Lặp qua các mục i trong tập dữ liệu D trong đó i là một phần tử của tập J, trong vòng lặp này, ta sẽ thực hiện các bước sau:
    - Thêm tập hợp I{i} vào tập kết quả $F(I)$.
    - Khởi tạo tập con Di rỗng
    - Lặp qua các mục j trong tập dữ liệu D với điều kiện j > i:
        - Tính độ bao phủ C của tập {i} và {j}
        - Nếu độ bao phủ C > ngưỡng hỗ trợ , thì thêm {j,C} vào tập D_i

3. Đệ quy: Sau khi đã chuẩn bị tập con D_i , gọi đệ quy hàm $F[I \cap i](D_i, \sigma)$, để tính toán các tập con có tiền tố $I \cup i$

4. Kết hợp kết quả: Cuối cùng, ta thêm các tập con được tính toán từ bước đệ quy vào tập kết quả F $F(I)$

Một cách khái quát, ta sẽ tính toán ra tất cả các tập con có tiền tố I dựa trên tập dữ liệu D và mức hỗ trợ đã được xác định ngay ban đầu. Kết quả sẽ lưu được trong tập $F(I)$

2.3.5. Minh họa thực hiện thuật toán.

Dưới đây là minh họa cơ bản về cách thức hoạt động của thuật toán ECLAT:

Giao dịch	Các mặt hàng
1	{A, B, C}
2	{A, C}
3	{A, B, D}
4	{B, E, F}
5	{A, B, C, E}

Bước 1: Chuẩn bị dữ liệu

Thuật toán ECLAT hoạt động trên cơ sở dữ liệu dọc (vertical database). Mỗi mục trong cơ sở dữ liệu sẽ được đại diện bởi một tập các giao dịch chứa mục đó. Ở ví dụ trên, từ cơ sở dữ liệu ngang, ta chuyển thành cơ sở dữ liệu dọc.

Các mặt hàng	Giao dịch
A	{1, 2, 3, 5}
B	{1, 3, 4, 5}
C	{1, 2, 5}
D	{3}
E	{4, 5}
F	{4}

Bước 2: Tìm tập mục phổ biến

Từ cơ sở dữ liệu dọc, chúng ta sẽ tìm tập phổ biến bằng cách sử dụng phép giao của các tập giao dịch. Thuật toán ECLAT sử dụng đệ quy để tìm các tập phổ biến và tạo ra các Equivalent Class với kích thước $K + 1$.

Quá trình này bao gồm các bước sau:

1. Bắt đầu với tập một mục phổ biến: Đếm số lượng giao dịch chứa mỗi mục. Nếu số lượng này lớn hơn hoặc bằng ngưỡng tối thiểu (minimum support), mục đó được xem là mục phổ biến.
2. Sinh tập hai mục phổ biến: Điều này có nghĩa rằng kết hợp hai mục phổ biến để tạo thành một tập hai mục và tính giao của các tập giao dịch tương ứng. Nếu kết quả giao có số lượng phần tử lớn hơn hoặc bằng ngưỡng tối thiểu (minimum support), tập hai mục này là phổ biến.
3. Lặp lại quá trình trên để tìm tập ba mục, bốn mục,... phổ biến cho đến khi không thể tìm thêm tập phổ biến mới.

Chúng ta sẽ bắt đầu với kích thước k = 1. Tất cả các mục (item) đơn lẻ đều là tập phổ biến ban đầu nếu chúng thỏa mãn ngưỡng hỗ trợ (support). Giả sử ngưỡng hỗ trợ (minimum support) là 2, dưới đây là các mặt hàng thỏa mãn ngưỡng hỗ trợ:

Các mặt hàng (Item)	Giao dịch (Transaction)	Hỗ trợ (Support)
A	{1, 2, 3, 5}	4
B	{1, 3, 4, 5}	4
C	{1, 2, 5}	3
E	{4, 5}	2

Tìm các tập phổ biến cấp k = 2:

Các mặt hàng (Item)	Giao dịch (Transaction)	Hỗ trợ (Support)
AB	{1,3,5}	3
AC	{1,2,5}	3
BC	{1,5}	2

BE	{4,5}	2

Tìm các tập phổ biến cấp k = 3:

Các mặt hàng (Item)	Giao dịch (Transaction)	Hỗ trợ (Support)
ABC	{1,5}	2

Chỉ có mặt hàng ABC thỏa mãn ngưỡng tối thiểu. Chúng ta sẽ dừng tìm tập mục phổ biến tại k = 3.

Quy luật trong database với ngưỡng hỗ trợ tối thiểu (minimum support) là 2, chúng ta có thể kết luận các quy luật trong database.

Sản phẩm	Gợi ý sản phẩm
A	B
A	C
B	C
B	E
A và B	C

2.3.6. So sánh thuật toán Apriori

Cả hai thuật toán ECLAT và Apriori đều có mục tiêu chung là tìm ra các tập hợp mục thường xuyên trong cơ sở dữ liệu giao dịch. Tuy nhiên chúng sử dụng các phương pháp và kỹ thuật khác nhau để đạt được mục tiêu này. Srinadh (2022) đã cung cấp đánh giá chi tiết về hiệu suất và đặc điểm của hai thuật toán này:

Tiêu chí	Thuật toán Apriori	Thuật toán Eclat
Cách tiếp cận	Tìm kiếm theo chiều rộng (breadth-first search)	Tìm kiếm theo chiều sâu (depth-first search)
Bố cục cơ sở dữ liệu	Ngang (liệt kê rõ ràng tất cả các giao dịch)	Dọc (lưu trữ mỗi mục cùng với danh sách giao dịch)
Số lần quét cơ sở dữ liệu	Nhiều lần quét	Ít lần quét hơn
Yêu cầu bộ nhớ	Yêu cầu lớn, đặc biệt với số lượng tập hợp mục ứng viên lớn	Yêu cầu ít hơn, nhất là khi các tập hợp mục nhỏ
Tốc độ và hiệu quả	Chậm hơn, đặc biệt với số lượng tập hợp mục lớn	Nhanh hơn, đặc biệt với tập dữ liệu nhỏ và trung bình
Phù hợp với loại dữ liệu	Tập dữ liệu lớn và phức tạp	Tập dữ liệu nhỏ và trung bình
Ứng dụng	Phù hợp cho các trường hợp cần xử lý khối lượng dữ liệu lớn	Phù hợp cho các tình huống yêu cầu nhanh chóng và hiệu quả
Ưu điểm	Khả năng xử lý tốt các cơ sở dữ liệu với nhiều mục và giao dịch	Tốc độ nhanh, hiệu quả với bố cục cơ sở dữ liệu dọc
Nhược điểm	Yêu cầu nhiều không gian bộ nhớ và thời gian xử lý	Hiệu quả giảm khi số lượng tập hợp mục lớn

Bảng 2-4. Bảng so sánh giữa thuật toán ECLAT và Apriori

CHƯƠNG 3. TỔNG QUAN BỘ DỮ LIỆU.

3.1. Mô tả dữ liệu.

- Link của tệp dữ liệu: Link
- Mô tả tệp dữ liệu:

- Tệp dữ liệu chứa các dữ liệu mua hàng của các hội viên thuộc cửa hàng bách hoá từ 21/01/2015 đến năm 30/04/2015
- Tệp dữ liệu bao gồm 38.765 dòng và 3 cột. Chi tiết:

Cột	Loại dữ liệu	Mô tả
Member_number	Int64	Mã khách hàng/hội viên
Date	Object	Ngày mua hàng
ItemDescription	Object	Mô tả những sản phẩm đã mua

- Dữ liệu không có giá trị trống. Chi tiết:

Cột	IsNullValue	Number of Unique Value
Member_number	0	3989
Date	0	728
ItemDescription	0	167

3.2. Tiền xử lý dữ liệu.

3.2.1. Kết hợp Member_number và Date thành một cột duy nhất 'Member_number-Date'

```
[ ]  # Combine 'Member_number' and 'Date' into a single column 'Member_number-Date'
     df['Member_number-Date'] = df['Member_number'].astype(str) + '-' + df['Date'].astype(str)
```

Hình 3-1. Kết hợp Member_number và Date thành cột duy nhất Member_number-Date

- **Mục đích**: Tạo một giá trị định danh duy nhất cho mỗi giao dịch bằng cách kết hợp số thành viên và ngày tháng.
- **Quá trình thực hiện:**

- df['Member_number'].astype(str): Chuyển đổi cột Member_number sang kiểu chuỗi (string).
- df['Date'].astype(str): Chuyển đổi cột Date sang kiểu chuỗi (string).
- + '-' +: Nối hai chuỗi lại với nhau bằng dấu gạch ngang (-) ở giữa

- **Kết quả:** Thêm một cột mới vào df có tên là Member_number-Date, trong đó mỗi giá trị là chuỗi kết hợp giữa Member_number và Date.

3.2.2. Nhóm dữ liệu theo cột mới và gom nhóm cột 'itemDescription'.

```
# Group by the new column and aggregate the 'itemDescription' column
grocery_data = df.groupby('Member_number-Date')['itemDescription'].apply(lambda x: ','.join(x)).reset_index()

# Display the grocery_data
print(grocery_data)
```

Hình 3-2. Nhóm dữ liệu theo cột mới và gom nhóm cột itemDescription.

- **Mục đích**: Nhóm dữ liệu theo cột Member_number-Date mới và kết hợp các mô tả mặt hàng được mua trong mỗi giao dịch thành một chuỗi duy nhất.
- **Quá trình thực hiện**:
  - df.groupby('Member_number-Date'): Nhóm DataFrame theo cột Member_number-Date.
  - ['itemDescription'].apply(lambda x: ','.join(x)): Áp dụng một hàm lambda cho mỗi nhóm. Hàm lambda này sẽ lấy các giá trị của itemDescription trong mỗi nhóm và nối chúng lại thành một chuỗi duy nhất, cách nhau bởi dấu phẩy.
  - .reset_index(): Đặt lại chỉ số của DataFrame kết quả, biến các khóa nhóm thành các cột.
- **Kết quả**: Một DataFrame mới grocery_data mà mỗi hàng đại diện cho một kết hợp Member_number-Date duy nhất, và cột itemDescription chứa danh sách các mặt hàng được mua trong giao dịch đó dưới dạng chuỗi cách nhau bởi dấu phẩy.

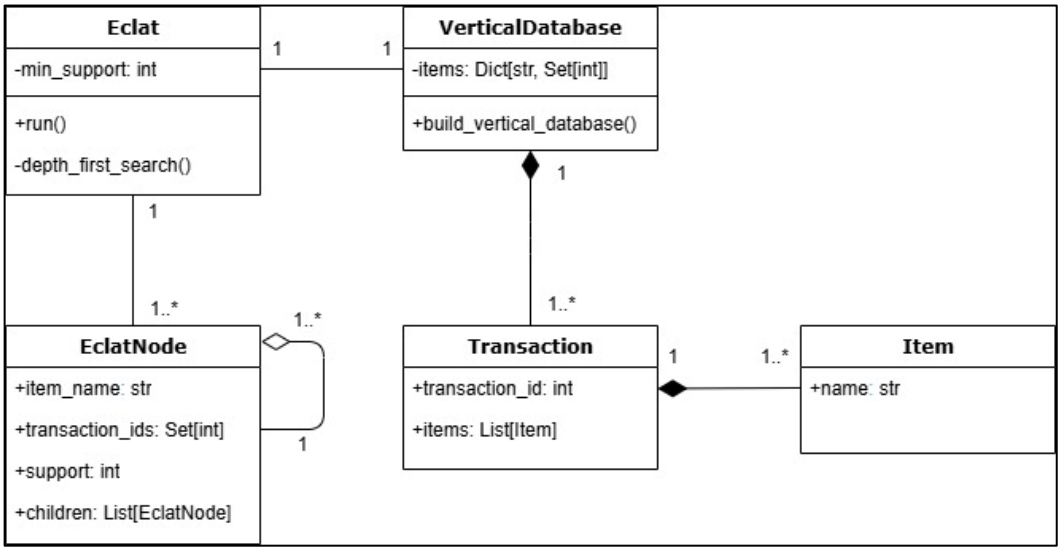
```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` w
  and should_run_async(code)
      Member_number-Date                                itemDescription
0        1000-15-03-2015      sausage,whole milk,semi-finished bread,yogurt
1        1000-24-06-2014                    whole milk,pastry,salty snack
2        1000-24-07-2015                      canned beer,misc. beverages
3        1000-25-11-2015                         sausage,hygiene articles
4        1000-27-05-2015                          soda,pickled vegetables
...                  ...                                             ...
14958    4999-24-01-2015  tropical fruit,berries,other vegetables,yogurt...
14959    4999-26-12-2015                              bottled water,herbs
14960    5000-09-03-2014                    fruit/vegetable juice,onions
14961    5000-10-02-2015          soda,root vegetables,semi-finished bread
14962    5000-16-11-2014                    bottled beer,other vegetables

[14963 rows x 2 columns]
```

Hình 3-3. Hiển thị DataFrame kết quả

Kết quả: In DataFrame ra màn hình console, hiển thị dữ liệu đã được tổng hợp theo dạng "long data", tức là mỗi giao dịch được đại diện bởi một hàng với mô tả các mặt hàng được liệt kê trong cột itemDescription.

CHƯƠNG 4. THỰC HIỆN GIẢI THUẬT.

4.1. Thiết kế lớp.



Hình 4-1. Thiết kế lớp của thuật toán ECLAT

Lớp	Mô tả
Item	Thuộc tính name để lưu trữ tên của mặt hàng.

Transaction	Thuộc tính transaction_id để lưu trữ ID của giao dịch và items để lưu trữ danh sách các mặt hàng trong giao dịch.
VerticalDatabase	Thuộc tính items để lưu trữ ánh xạ giữa tên mặt hàng và tập hợp các ID giao dịch chứa mặt hàng đó, có phương thức build_vertical_database để xây dựng cơ sở dữ liệu dọc từ danh sách các giao dịch.
EclatNode	Các thuộc tính item_name, transaction_ids, support và children để lưu trữ thông tin về tập mặt hàng phổ biến dưới dạng một nút trong cây Eclat.
Eclat	Thuộc tính min_support để lưu trữ ngưỡng hỗ trợ tối thiểu, phương thức run để chạy thuật toán Eclat trên cơ sở dữ liệu dọc và phương thức depth_first_search để thực hiện tìm kiếm theo chiều sâu trên cây Eclat.

Bảng 4-1. Mô tả chi tiết thiết kế lớp của thuật toán ECLAT

4.2. Tìm tập phổ biến.

Bằng việc tiền xử lý dữ liệu thô và chuyển đổi sang cơ sở dữ liệu dọc, mỗi giao dịch được biểu diễn bằng một tập hợp các mục và được ánh xạ vào một cấu trúc dữ liệu dọc, nơi mỗi mục liên kết với một tập hợp các ID giao dịch chứa nó. Ta lựa chọn ngưỡng hỗ trợ tối thiểu 3% sau khi thử nghiệm với các giá trị 1% và 5%. Với ngưỡng hỗ trợ 1%, thuật toán tăng đáng kể thời gian xử lý và phức tạp hóa việc phân tích kết quả do sự gia tăng của các tập mục không thực sự có ý nghĩa. Ngược lại, với ngưỡng hỗ trợ 5%, số lượng tập hợp mục phổ biến giảm đáng kể, dẫn đến việc bỏ sót nhiều tập hợp mục quan trọng có thể cung cấp thông tin giá trị cho phân tích.

Sau khi quyết định ngưỡng hỗ trợ tối thiểu, ta tiến hành tìm tập phổ biến dựa vào thiết kế lớp đã được trình bày phần 4.1.

```python
class Item:
    def __init__(self, name):
        self.name = name

class Transaction:
    def __init__(self, transaction_id, items):
        self.transaction_id = transaction_id
        self.items = items

class VerticalDatabase:
    def __init__(self, transactions):
        self.items = {}
        self.build_vertical_database(transactions)

    def build_vertical_database(self, transactions):
        for transaction in transactions:
            for item in transaction.items:
                if item.name not in self.items:
                    self.items[item.name] = set()
                self.items[item.name].add(transaction.transaction_id)

class EclatNode:
    def __init__(self, item_name, transaction_ids, support):
        self.item_name = item_name
        self.transaction_ids = transaction_ids
        self.support = support
        self.children = []
```

Hình 4-2. Thiết kế lớp Item, Transaction, VerticalDatabase và EclatNode

```python
class Eclat:
    def __init__(self, min_support):
        self.min_support = min_support

    def run(self, vertical_db):
        frequent_itemsets = []
        for item_name, transaction_ids in vertical_db.items.items():
            support = len(transaction_ids)
            if support >= self.min_support:
                frequent_itemsets.append(EclatNode(item_name, transaction_ids, support))
        self.depth_first_search(frequent_itemsets, vertical_db)
        return frequent_itemsets

    def depth_first_search(self, itemsets, vertical_db):
        for i in range(len(itemsets)):
            for j in range(i + 1, len(itemsets)):
                intersection = itemsets[i].transaction_ids & itemsets[j].transaction_ids
                support = len(intersection)
                if support >= self.min_support:
                    new_itemset = EclatNode(
                        itemsets[i].item_name + "," + itemsets[j].item_name,
                        intersection,
                        support
                    )
                    itemsets[i].children.append(new_itemset)
                    self.depth_first_search([new_itemset], vertical_db)
```

Hình 4-3. Thiết kế lớp ECLAT

Trước hết, dữ liệu giao dịch được xử lý để tạo các đối tượng Item và Transaction. Trong bước này, mỗi hàng dữ liệu từ DataFrame grocery_data được duyệt qua, tách các mục và

ngày giao dịch dựa trên số thành viên và ngày. Mỗi mục được gán vào một giao dịch tương ứng, tạo nên danh sách các đối tượng Transaction.

```
# Step 1: Parse the DataFrame and create Item and Transaction objects
transactions = []
for _, row in grocery_data.iterrows():
    item_description = row["itemDescription"]
    member_dates = row["Member_number-Date"].split(',')

    for member_date in member_dates:
        member_number, date = member_date.split('-')[0], member_date.split('-')[1:]
        transaction_id = int(member_number)  # Using member number as transaction ID
        item = Item(item_description)

        # Check if transaction with this ID already exists
        transaction = next((t for t in transactions if t.transaction_id == transaction_id), None)
        if transaction is None:
            transaction = Transaction(transaction_id, [item])
            transactions.append(transaction)
        else:
            transaction.items.append(item)
```

Hình 4-4. Phân tích DataFrame và tạo các đối tượng Item và Transaction

Các đối tượng Transaction được sử dụng để xây dựng cơ sở dữ liệu dọc thông qua lớp VerticalDatabase. Bước này chuyển đổi dữ liệu từ dạng ngang sang dạng dọc, ánh xạ mỗi mục tới các ID giao dịch chứa nó. Thuật toán Eclat được thực hiện trên cơ sở dữ liệu dọc với ngưỡng hỗ trợ tối thiểu 3% đảm bảo rằng các tập mục được phát hiện có đủ tần suất xuất hiện để được coi là phổ biến, nhưng không quá thấp để tránh việc tạo ra quá nhiều tập hợp mục không có ý nghĩa.

```
# Step 2: Build the Vertical Database
vertical_db = VerticalDatabase(transactions)

# Step 3: Run the Eclat Algorithm
total_transactions = len(transactions)
min_support_percentage = 0.03 # 3%
min_support = int(total_transactions * min_support_percentage)
eclat = Eclat(min_support)
frequent_itemsets = eclat.run(vertical_db)
```

Hình 4-5. Xây dựng cơ sở dữ liệu dọc và chạy ECLAT với ngưỡng hỗ trợ 3%

Ta dùng hàm đệ quy extract_frequent_itemsets để trích xuất các tập mục và mức hỗ trợ từ các đối tượng EclatNode.

```python
# Step 4: Retrieve and Process Results
def extract_frequent_itemsets(itemsets, itemset_list=None, support_list=None):
    if itemset_list is None:
        itemset_list = []
        support_list = []

    for itemset in itemsets:
        itemset_list.append(itemset.item_name)
        support_list.append(itemset.support)
        extract_frequent_itemsets(itemset.children, itemset_list, support_list)

    return itemset_list, support_list

itemsets, supports = extract_frequent_itemsets(frequent_itemsets)

# Create a DataFrame from the extracted data
data = {"Itemset": itemsets, "Support": supports}
df = pd.DataFrame(data)

print("Frequent Itemsets (DataFrame):")
print(df)
```

Hình 4-6. Trích xuất và xử lý kết quả

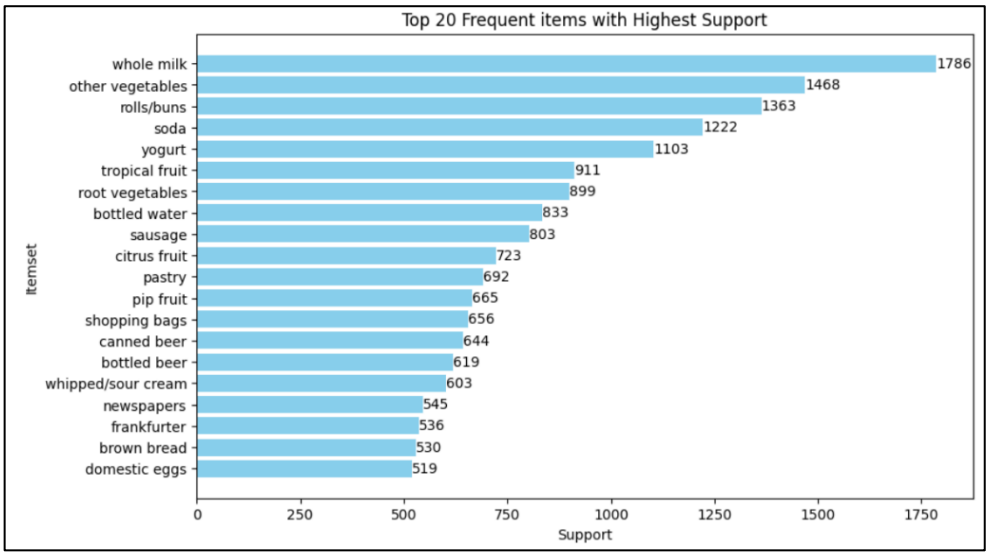
Ta đếm số lượng mục trong mỗi tập mục bằng cách đếm số dấu phẩy trong chuỗi đại diện cho tập hợp mục và cộng thêm 1.

```python
# Count the number of items in each itemset
df['Itemset Count'] = df['Itemset'].str.count(',') + 1
print("DataFrame with Itemset Counts:")
print(df)
```

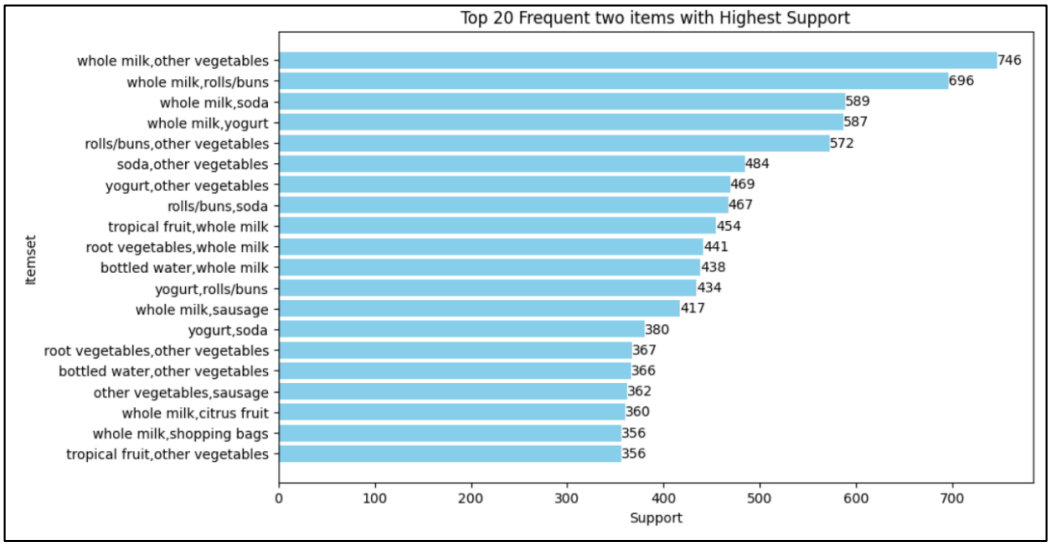
```
DataFrame with Itemset Counts:
                       Itemset  Support  Itemset Count
0                     UHT-milk      306              1
1          UHT-milk,whole milk      158              2
2          UHT-milk,rolls/buns      121              2
3    UHT-milk,other vegetables      152              2
4                bottled water      833              1
..                         ...      ...            ...
327         pickled vegetables      130              1
328                 pot plants      116              1
329             red/blush wine      155              1
330                      pasta      118              1
331                  detergent      127              1

[332 rows x 3 columns]
```

Hình 4-7. Đếm số lượng item trong mỗi itemset

Với min_support 3%, thuật toán trả về kết quả của các k-itemsets (k=1, 2) phổ biến.



Hình 4-8. Top 20 tập mục phổ biến có độ hỗ trợ cao nhất (k=1)



Hình 4-9. Top 20 tập mục phổ biến có độ hỗ trợ cao nhất (k=2)

4.3. Khai thác và đánh giá luật kết hợp.

Để chuẩn bị cho việc sinh luật kết hợp, các tập phổ biến được chuyển đổi thành định dạng frozenset và độ hỗ trợ (support) được tính toán lại dưới dạng số thập phân. Để sinh luật kết hợp, ta dùng hàm association_rules() từ thư viện mlxtend, cho phép tạo ra các luật kết hợp từ một DataFrame chứa các tập phổ biến và độ hỗ trợ. Sau khi thực nghiệm, ta áp dụng ngưỡng độ tin cậy 53% để tạo ra các luật mạnh. Mỗi luật kết hợp bao gồm thông tin về tập mục vế trái (antecedent), tập mục vế phải (consequent), độ hỗ trợ (support), độ tin cậy (confidence) và các chỉ số đánh giá khác như lift và leverage.

```
# Convert itemsets to frozenset
df_one['itemsets'] = df_one['Itemset'].apply(lambda x: frozenset([x]))
df_two['itemsets'] = df_two['Itemset'].apply(lambda x: frozenset(x.split(',')))
# Convert itemsets to frozenset and calculate support as a decimal
df_one['itemsets'] = df_one['Itemset'].apply(lambda x: frozenset([x]))
df_one['support'] = df_one['Support'] / total_transactions
df_two['itemsets'] = df_two['Itemset'].apply(lambda x: frozenset(x.split(',')))
df_two['support'] = df_two['Support'] / total_transactions
# Create a DataFrame with itemsets and support
itemsets_df = pd.concat([df_one[['itemsets', 'support']], df_two[['itemsets', 'support']]], axis=0)
# Generate association rules
rules = association_rules(itemsets_df, metric="confidence", min_threshold=0.53)
```

Hình 4-10 Sinh luật kết quả từ hàm association_rules().

antecedents	Consequents	antecedent support	consequent support	support	Confidence	Lift
(cream cheese)	(whole milk)	0.088507	0.458184	0.047717	0.539130	1.176669
(chocolate)	(whole milk)	0.086455	0.458184	0.047973	0.554896	1.211078
(yogurt)	(whole milk)	0.282966	0.458184	0.150590	0.532185	1.161510
(bottled beer)	(whole milk)	0.158799	0.458184	0.085428	0.537964	1.174124
(frozen vegetables)	(whole milk)	0.102617	0.458184	0.055156	0.537500	1.173110
(white bread)	(whole milk)	0.088763	0.458184	0.047717	0.537572	1.173268
(beef)	(whole milk)	0.119548	0.458184	0.064135	0.536481	1.170886
(shopping bags)	(whole milk)	0.168291	0.458184	0.091329	0.542683	1.184422
(meat)	(whole milk)	0.063622	0.458184	0.034890	0.548387	1.196872
(waffles)	(whole milk)	0.069010	0.458184	0.037968	0.550186	1.200798

(sugar)	(whole milk)	0.065931	0.458184	0.036942	0.560311	1.222897
(ham)	(whole milk)	0.063366	0.458184	0.036942	0.582996	1.272407
(ice cream)	(whole milk)	0.056439	0.458184	0.030272	0.536364	1.170630
(oil)	(whole milk)	0.055670	0.458184	0.030272	0.543779	1.186814
(hamburger meat)	(whole milk)	0.080298	0.458184	0.045408	0.565495	1.234211

Bảng 4-2. Các chỉ số của luật kết hợp.

Ta tiến hành thiết lập ngưỡng cho các chỉ số đánh giá:

Chỉ số	**Ngưỡng**
MinSup	0.03
MinConf	0.53
MinLift	1.16

Bảng 4-3. Ngưỡng đánh giá các chỉ số.

Từ việc khai thác luật kết hợp, ta lọc ra các luật phổ biến theo các chỉ số.

Những luật có độ hỗ trợ cao gồm:

Antecedents	**consequents**	**Support**	**confidence**	**lift**
(yogurt)	(whole milk)	0.150590	0.532185	1.161510
(shopping bags)	(whole milk)	0.091329	0.542683	1.184422
(bottled beer)	(whole milk)	0.085428	0.537964	1.174124
(beef)	(whole milk)	0.064135	0.536481	1.170886
(frozen vegetables)	(whole milk)	0.055156	0.537500	1.173110

Bảng 4-4. Top 5 luật có độ hỗ trợ cao.

Luật (yogurt) => (whole milk) có độ hỗ trợ cao nhất 0.150590, tức xuất hiện trong 15.06% giao dịch. Tuy nhiên luật (yogurt) => (whole milk) có độ tin cậy và lift không lớn, chỉ vừa đạt ngưỡng. Các luật khác có độ hỗ trợ tương đối cao như (shopping bags) => (whole milk) và (bottled beer) => (whole milk) lần lượt là 0.091329 và 0.085428.

Những luật có độ tin cậy cao gồm:

antecedents	**consequents**	**Support**	**confidence**	**lift**
(ham)	(whole milk)	0.036942	0.582996	1.272407
(hamburger meat)	(whole milk)	0.045408	0.565495	1.234211
(sugar)	(whole milk)	0.036942	0.560311	1.222897
(chocolate)	(whole milk)	0.047973	0.554896	1.211078
(waffles)	(whole milk)	0.037968	0.550186	1.200798

Bảng 4-5. Top 5 luật có độ tin cậy và chí số lift cao

Luật (ham) => (whole milk) và (hamburger meat) => (whole milk) có độ confidence cao nhất lần lượt là 0.582996 và 0.565495, nghĩa là khi khách hàng mua ham hay hamburger meat thì có xác suất lần lượt 58.3% hay 56,5% họ cũng sẽ mua whole milk.

Luật (ham) => (whole milk) có chỉ số lift cao nhất 1.272407, tức khi khách hàng mua giăm bông, khả năng họ mua sữa nguyên kem tăng 27.24% so với khi không có thông tin về giăm bông. Các luật khác có lift khá cao như (hamburger meat) => (whole milk) và (sugar) => (whole milk) lần lượt là 1.234211 và 1.222897.

Sau khi đánh giá các chỉ số, ta rút ra một số nhận xét như sau:

- Whole milk là sản phẩm chủ đạo, có mối liên hệ chặt chẽ và đa dạng với nhiều mặt hàng khác.
- Về chỉ số lift, luật (ham) => (whole milk), (hamburger meat) => (whole milk) và (sugar) => (whole milk) có giá trị lần lượt là 1.272407, 1.234211 và 1.222897, cao nhất trong tập luật và vượt xa ngưỡng 1.16. Lift thể hiện mức độ tương quan mạnh mẽ, trong đó sự xuất hiện của ham, hamburger meat hay sugar làm tăng xác suất mua whole milk lên 22% đến 27% so với trường hợp ngẫu nhiên.

CHƯƠNG 5. KẾT LUẬN.

Qua quá trình khai phá và đánh giá các luật kết hợp từ tập dữ liệu giao dịch bán lẻ bằng thuật toán ECLAT, nghiên cứu đã đạt được những kết quả quan trọng, hỗ trợ ra quyết định trong kinh doanh bằng việc áp dụng thành công thuật toán ECLAT với ngưỡng support 0.03, confidence 0.53 và lift 1.16 để khai phá luật kết hợp từ dữ liệu giao dịch bán lẻ. Ba luật kết hợp mạnh gồm (ham) => (whole milk), (hamburger meat) => (whole milk) và (sugar) => (whole milk) đã thể hiện mối quan hệ chặt chẽ và tương quan cao giữa các mặt hàng. Ngoài ra, việc kiểm tra nhiều ngưỡng khác nhau giúp tìm ra bộ thông số tối ưu, phù hợp với đặc trưng của tập dữ liệu.

Từ kết quả trên, các khuyến nghị được đề xuất như xây dựng combo sản phẩm, tái cấu trúc không gian trưng bày và triển khai khuyến mãi chéo hứa hẹn mang lại lợi ích thiết thực, góp phần gia tăng doanh số, nâng cao sự hài lòng của khách hàng và củng cố lợi thế cạnh tranh. Tuy nhiên, nghiên cứu chỉ tập trung vào luật kết hợp 2 mục, có thể bỏ qua mối quan hệ phức tạp và đa chiều giữa sản phẩm. Về hướng phát triển đề tài, nghiên cứu cần khai thác luật kết hợp đa mục và kết hợp với kỹ thuật tiên tiến như phân cụm, chuỗi thời gian, học sâu để đưa ra những khuyến nghị toàn diện và chính xác hơn.

TÀI LIỆU THAM KHẢO

1. Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997, August). New algorithms for fast discovery of association rules. In KDD (Vol. 97, pp. 283-286).
2. Bakar, W. A. B. W. A., Abdullah, Z. B., Saman, M. Y. B. M., Man, M. B., Herawan, T., & Hamdan, A. R. (2016). Incremental-Eclat Model.
3. Man, M., & Jalil, M. A. (2019). Frequent itemset mining: Technique to improve ECLAT based algorithm. International Journal of Electrical and Computer Engineering, 9(6), 5471-5478.
4. Đang, N. H. (2024, May 30). Khai phá luật kết hợp, luật kết hợp hiếm là gì ??? Viblo. https://viblo.asia/p/khai-pha-luat-ket-hop-luat-ket-hop-hiem-la-gi-5pPLkx02VRZ.
5. Srinadh. (2022). Evaluation of Apriori, FP growth, Eclat association rule mining algorithms. *International Journal of Health Sciences*, *6(S2)*(7475-7485).
6. 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).

PHỤ LỤC

Mã nguồn: Link