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

ĐỒ ÁN CUỐI KỲ KHAI PHÁ DỮ LIỆU

ĐỀ TÀI: TÌM HIỂU THUẬT TOÁN ECLAT VÀ ÁP DỤNG GIẢI THUẬT VÀO BỘ DỮ LIỆU GROCERIES 3

SINH VIÊN THỰC HIỆN ĐỒ ÁN:

Nguyễn Trịnh Hiếu Kiên: 31211027199
Đặng Châu Kỳ: 31211027647
Nguyễn Hoàng Hà My: 31211023184
Tất Diệu Ngân: 31211027652
Phan Đình Nhân: 31211027657

NGÀNH: Khoa học dữ liệu – Khóa 47
Giảng viên: TS. Nguyễn An Tế

LỜI CẢM ƠN

Kính gửi thầy Nguyễn An Tế,

Mở đầu bài đồ án này, nhóm em xin được gửi lời cảm ơn đến thầy vì những kiến thức thầy đã truyền đạt cho chúng em trong học phần 'Khai phá dữ liệu'. Các kiến thức này đối với nhóm em là kiến thức quý báu để chúng em có thể hiểu rõ hơn về môn học này và tiếp tục làm nền tảng mở rộng để tiếp cận nhiều hơn với những hiểu biết khác.

Chúng em muốn cám ơn thầy vì đã hướng dẫn cho chúng em tìm hiểu đến những kiến thức mới cũng như củng cố lại những kiến thức cũ. Trải qua những bài giảng và câu hỏi của thầy chúng em được mở rộng vốn tri thức mình có, giúp chúng em suy luận và vận dụng kiến thức liên kết các môn học lại với nhau, từ đó có thể hình dung rõ hơn về ngành 'Khoa học dữ liệu'.

Với đồ án cuối kỳ này, bên cạnh giúp chúng em có khả năng tự tìm hiểu và tư duy về vấn đề mới, đồ án còn tạo cơ hội để nhóm áp dụng những kỹ năng và kiến thức đã được học trong học phần 'Khai phá dữ liệu' để có cái nhìn tổng quan về những vấn đề trong quản lý sản phẩm và phân tích đưa ra quyết định trong kinh doanh.

Cuối cùng, nhóm em xin được chân thành cảm ơn vì những kiến thức quý báu mà thầy đã truyền đạt. Với những nền tảng tri thức này sẽ là bước đệm quan trọng cho quá trình học tập và nghiên cứu của nhóm trong tương lai.

Trân trọng cảm ơn thầy,
Nhóm nghiên cứu

LỜI MỞ ĐẦU

Trong thời đại số hóa ngày nay, dữ liệu đã trở thành nguồn tài nguyên vô cùng quý giá. Việc khai thác và phân tích dữ liệu không chỉ mang lại thông tin hữu ích mà còn giúp chúng ta hiểu rõ hơn về các mô hình, xu hướng, và quy luật ẩn sau những dữ liệu số. Trong đồ án này, chúng tôi tập trung nghiên cứu và áp dụng thuật toán ECLAT - một trong những thuật toán quan trọng trong lĩnh vực khai phá dữ liệu - để tìm hiểu và phân tích bộ dữ liệu 'Groceries 3'.

Bộ dữ liệu 'Groceries 3' chứa thông tin về các sản phẩm được bán trong cửa hàng cùng với thông tin về việc mua hoặc không mua của khách hàng. Sự phong phú và đa dạng của dữ liệu này tạo ra cơ hội để áp dụng các phương pháp khai phá dữ liệu nhằm khám phá các mẫu quy luật mua sắm, mối quan hệ giữa các sản phẩm và thói quen mua hàng của người tiêu dùng.

Nhóm sẽ tiến hành nghiên cứu về thuật toán ECLAT, tìm hiểu cách nó hoạt động và ứng dụng thuật toán này để phân tích bộ dữ liệu 'Groceries 3'. Bằng cách áp dụng ECLAT, nhóm hy vọng có thể xác định được các mẫu quy luật mua hàng phổ biến, chuỗi sản phẩm thường được mua cùng nhau, từ đó hỗ trợ các cửa hàng trong việc tối ưu hóa vị trí sản phẩm, chiến lược bán hàng và cải thiện trải nghiệm mua sắm cho khách hàng.

Qua việc áp dụng ECLAT vào bộ dữ liệu 'Groceries 3', chúng tôi mong rằng đồ án này sẽ cung cấp cái nhìn sâu hơn về khả năng ứng dụng của thuật toán trong việc khai phá dữ liệu thương mại và đóng góp vào việc nâng cao hiệu suất kinh doanh trong ngành bán lẻ.

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1. *Sơ lược đề tài*

Trong đồ án này, nhóm sẽ tìm hiểu về thuật toán ECLAT, một thuật toán phát hiện luật kết hợp phổ biến. Chúng ta sẽ áp dụng thuật toán để phân tích bộ dữ liệu 'Groceries 3', một bộ dữ liệu gồm những thuộc tính là sản phẩm trong cửa hàng và giá trị là mua hoặc không mua.

Bộ dữ liệu Groceries là một bộ dữ liệu mẫu, nhưng nó chứa nhiều thông tin hữu ích về thói quen mua sắm của người tiêu dùng. Bằng cách phân tích bộ dữ liệu này, chúng ta có thể tìm ra những mối quan hệ giữa các sản phẩm để đưa ra những quyết định về hàng tồn kho và cách sắp xếp sản phẩm trong cửa hàng nhằm nâng cao doanh thu.

Ví dụ, chúng ta có thể tìm ra rằng những người mua trái cây cũng có xu hướng mua bánh mì. Hoặc, chúng ta có thể tìm ra rằng những người mua sữa cũng có xu hướng mua bánh ngọt. Những thông tin này có thể được sử dụng để cải thiện hiệu quả kinh doanh của các cửa hàng.

2. *Mục tiêu nghiên cứu*

Mục tiêu của đồ án này là:

- ➢ Tìm hiểu về thuật toán ECLAT
- ➢ Áp dụng thuật toán ECLAT để phân tích bộ dữ liệu Groceries
- ➢ Tìm ra những mối quan hệ giữa các sản phẩm trong bộ dữ liệu 'Groceries 3'

3. *Phương pháp nghiên cứu*

Các phương pháp cụ thể được sử dụng trong đồ án bao gồm:

- ➢ Nghiên cứu tài liệu: Nghiên cứu các tài liệu tham khảo về thuật toán ECLAT và phát hiện luật kết hợp.
- ➢ Quan sát tổng quan bộ dữ liệu
- ➢ Phân tích dữ liệu: Phân tích bộ dữ liệu Groceries bằng thuật toán ECLAT
- ➢ So sánh và cải tiến thuật toán ECLAT với các thuật toán kết hợp khác là Apriori và FP-Growth
- ➢ Rút ra kết luận: Rút ra các kết luận từ phân tích dữ liệu.

4. *Phương pháp thực hiện*

Các bước thực hiện cụ thể như sau:

- ➢ Tìm hiểu về luật kết hợp và các thuật toán liên quan, đặc biệt là thuật toán ECLAT
- ➢ Đọc và quan sát tổng quan dữ liệu: Đọc dữ liệu từ file CSV và trực quan thuộc tính của bộ dữ liệu

- ➢ Xử lý dữ liệu: Xử lý các dữ liệu thiếu hoặc lỗi.
- ➢ Áp dụng giải thuật ECLAT: Tạo cây ECLAT từ bộ dữ liệu đã xử lý.
- ➢ Xuất kết quả: Xuất kết quả phân tích của thuật toán ECLAT.
- ➢ Business Insight: Đưa ra những đánh giá và nhận xét cho bài toán mà nhóm đã đặt ra

CHƯƠNG 2. KHÁI NIỆM TỔNG QUÁT

1. Giới thiệu chung về thuật toán ECLAT:

- Thuật toán ECLAT (Equivalence Class Transformation) được giới thiệu bởi Zaki, Parthasarathy, Ogihara và Li vào năm 1997 được hiểu là Equivalence Class Clustering and Bottom-Up Lattice Traversal được sử dụng trong việc khai thác luật kết hợp (association rules mining).
- Phân tích từng thuật ngữ trong tên thuật toán:
  - + *Lattice (lưới)*: Một cấu trúc giống như cây trong đó mỗi nút đại diện cho một itemset và mỗi cạnh thể hiện mối quan hệ giữa các itemsets. Cấu trúc mạng trong ECLAT đại diện cho tất cả các tập mục có thể có.
  - + *Equivalence Class Clustering (ECC)*: Là một phương pháp được sử dụng để nhóm các tài liệu hoặc các trường hợp tương tự lại với nhau dựa trên sự tương đồng ngữ nghĩa ở đây được sử dụng để nhóm các tập hợp phần tử đơn tương đồng vào các lớp tương đương *(equivalence classes)*. Các lớp tương đương này được tổ chức thành một cấu trúc dữ liệu cây hoặc lưới.
  - + *Bottom-up Lattice Traversal (BULT)*: Là một kỹ thuật được sử dụng trong xử lý ngôn ngữ tự nhiên (NLP) và truy xuất thông tin. Quá trình này giúp tạo ra các frequent itemsets bằng cách kết hợp các tập hợp phần tử đơn từ các lớp tương đương khác nhau. BULT cũng giúp giảm thiểu số lượng phép so sánh cần thực hiện trong quá trình khai phá.
  - + *Deep-first*: Là chiến lược tìm kiếm duyệt qua cấu trúc mạng *(Lattice)* để khám phá các tập phổ biến *(frequent itemset)*. Thay vì tìm kiếm theo chiều rộng *(breadth-first strategy)* duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị. Thuật toán bắt đầu từ một đỉnh gốc (hoặc một đỉnh bất kỳ) và duyệt qua tất cả các đỉnh ở cùng một mức trước khi chuyển sang mức tiếp theo thì ECLAT sử dụng chiến lược tìm kiếm theo chiều sâu *(depth-first strategy)* là một thuật toán duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị. Thuật toán bắt đầu từ một đỉnh gốc (hoặc một đỉnh bất kỳ) và duyệt sâu theo mỗi

nhánh cho đến khi không thể đi xa hơn, sau đó quay lui và duyệt nhánh khác

- ECLAT giúp xác định các mẫu phổ biến *(frequent patterns)* trong tập dữ liệu bằng cách tạo ra các cặp tần suất xuất hiện của các itemsets (tập hợp các mục). Trong cơ sở dữ liệu, nó xác định các nhóm mục thường xuyên *(frequent itemset)* xuất hiện cùng nhau, được gọi là 'tập phổ biến'. Khi đã xác định được các tập phổ biến này, ECLAT sẽ tìm kiếm mối quan hệ giữa chúng, được gọi là '*Quy tắc kết hợp*'.
- Phương pháp này sử dụng kỹ thuật tổ chức cơ sở dữ liệu theo chiều dọc để nhóm các giao dịch liên quan với nhau. ECLAT có thể xử lý hiệu quả các tập dữ liệu lớn. Nó có thể xử lý nhiều loại dữ liệu khác nhau, chẳng hạn như nhị phân, danh nghĩa và số, có thể hữu ích trong các lĩnh vực khác nhau.
- Một số ứng dụng cụ thể của thuật toán ECLAT:
    + Phân tích thông tin về các mẫu mua hàng, quy tắc mua hàng kết hợp (ví dụ: nếu người mua một sản phẩm A thì họ có thể cũng quan tâm đến sản phẩm B), giúp cải thiện chiến lược bán hàng, quảng cáo và quản lý hàng hóa.
    + Phân tích hành vi của người dùng trên website (chẳng hạn như mua sắm, xem trang, thao tác) để đề xuất sản phẩm tương tự hoặc phù hợp với sở thích của họ.
    + Khai thác thông tin từ cơ sở dữ liệu y tế, xác định các mẫu tụ hợp trong dữ liệu bệnh án hoặc dữ liệu thử nghiệm y học để phát hiện mối quan hệ giữa các triệu chứng, căn bệnh và điều trị.
    + Áp dụng trong việc phân tích dữ liệu về các hành vi hoặc tác vụ trên hệ thống để tối ưu hóa hiệu suất và cải thiện trải nghiệm người dùng.

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

- *Items*: Tập hợp những thuộc tính có thể xuất hiện trong cơ sở dữ liệu

$$\mathbf{I = \{I_1, I_2, I_3, I_4, \ldots, I_n\}}$$

- *Itemset*: Tập hợp các mục, ví dụ như các sản phẩm trong siêu thị

$$\mathbf{X \subseteq I}$$

- *Cơ sở dữ liệu giao dịch (transaction database)*: Nhiều tập con của các tập mục đại diện cho một giao dịch

$$\emptyset \neq T \subseteq I$$

- *Database:*

$$D = \{T_i\}$$

- *Mẫu phổ biến (frequent patterns)* là những tập hợp các đối tượng, sự kiện, hoặc hành vi có tần suất xuất hiện cao trong một cơ sở dữ liệu

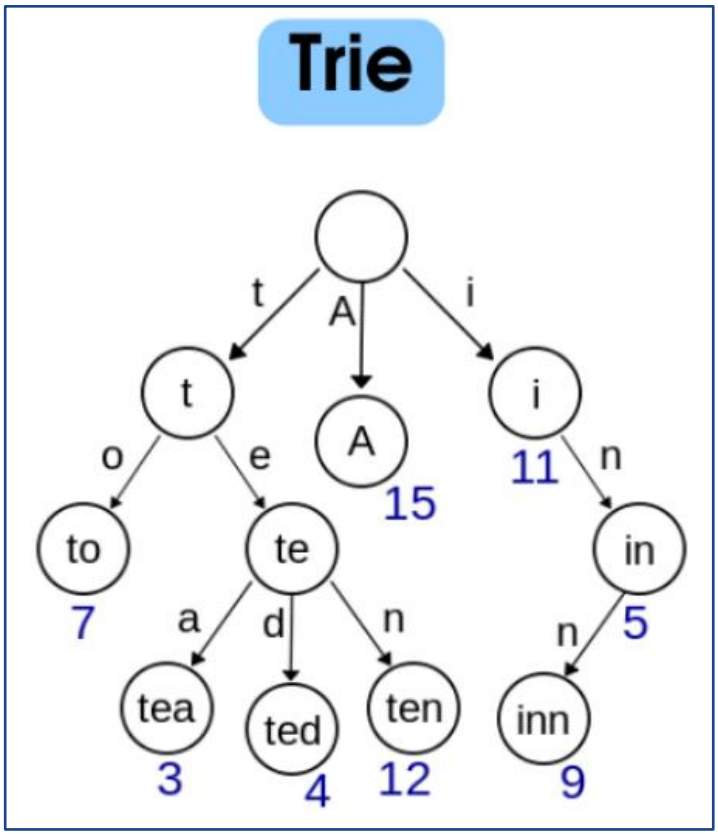
- *Tìm kiếm theo chiều sâu (depth-first search)* là một thuật toán duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị. Thuật toán bắt đầu từ một đỉnh gốc (hoặc một đỉnh bất kỳ) và tiếp tục đi sâu theo mỗi nhánh cho đến khi không thể đi xa hơn, sau đó quay lui và duyệt nhánh khác

- *Ma trận dọc (vertical format)* là một cách bố trí dữ liệu theo chiều dọc, thường dùng trong các bảng biểu, biểu đồ

- *Vertical Data Layout* đề cập đến cách trình bày, trong đó mỗi hàng tương ứng với một mục và mỗi cột tương ứng với một giao dịch. Cách bố trí này trái ngược với cách bố trí dữ liệu theo chiều ngang mỗi hàng tương ứng với một giao dịch và mỗi cột tương ứng với một mục trong tập mục.

- *Prefix* trong thuật toán eclat là một tập con của một tập phổ biến, chứa các đồ vật có thứ tự trước trong tập đó. Ví dụ, nếu tập phổ biến là {a, b, c, d}, thì các prefix *(tiền tố)* của nó là {a}, {a, b}, {a, b, c}. Prefix được dùng để tạo ra các ứng viên (candidates) cho các tập phổ biến lớn hơn, bằng cách kết hợp các prefix có cùng độ dài và có phần tử cuối cùng khác nhau. Ví dụ, nếu có hai prefix là {a, b} và {a, c}, thì có thể tạo ra ứng viên {a, b, c} bằng cách kết hợp chúng

- *Trie (cây tiền tố)* là một cấu trúc dữ liệu dạng cây, dùng để lưu trữ và quản lý một tập hợp các xâu. Mỗi nút của trie biểu diễn một xâu, và mỗi cạnh của trie biểu diễn một ký tự. Trie được sử dụng trong thuật toán ECLAT để lưu trữ các tập phổ biến và các prefix của chúng.



Hình ảnh ví dụ cấu trúc dữ liệu 'Trie'

- *Threshold* trong thuật toán ECLAT là một giá trị xác định tần suất tối thiểu của một tập hợp các mục (itemset) trong cơ sở dữ liệu.
- *Tập mục thường xuyên (frequent itemset):* số lần xuất hiện của X trong các giao dịch của D. Một tập có tần số lớn hơn hoặc bằng ngưỡng minsup

$$freq(X) = |\{T \in D\}|\ X \subseteq T\}$$

VD:

	Apple
0	FALSE
1	FALSE
2	TRUE
3	FALSE
4	TRUE

5	TRUE
6	FALSE
7	TRUE
8	TRUE
9	TRUE
10	TRUE

➔ *freq({Apple, TRUE}) = 7*

- Độ hỗ trợ (Support): Độ phổ biến của luật trong D

sup(X⇒Y) = P(XY)= freq (XY)/D

VD:

	Apple
0	FALSE
1	FALSE
2	TRUE
3	FALSE
4	TRUE
5	TRUE
6	FALSE
7	TRUE
8	TRUE
9	TRUE
10	TRUE

➔ *sup(Apple=>TRUE) = 7/11 ≈ 63.63%*

3. *Cơ chế hoạt động của thuật toán ECLAT*

- Thuật toán ECLAT *(Equivalence Class Clustering and Bottom-Up Lattice Traversal)* hoạt động dựa trên việc sử dụng cơ chế tạo ra các cặp tần suất xuất hiện của các itemsets (tập hợp các mục) và sử dụng các equivalence class để tìm ra các mẫu phổ biến trong dữ liệu.
- Thuật toán ECLAT sử dụng một cấu trúc dữ liệu gọi là *prefix tree (cây tiền tố)* để lưu trữ các tập phổ biến và các prefix của chúng. Mỗi nút của cây tiền tố biểu diễn một tập phổ biến, và mỗi nhánh của cây tiền tố biểu diễn một prefix. Thuật toán eclat duyệt cây tiền tố theo chiều sâu *(depth-first search)* để tìm ra tất cả các tập phổ biến thỏa mãn ngưỡng support tối thiểu
- Để tạo ra các *Equivalence Class* mới từ các *Equivalence Class* đã có, thuật toán eclat sử dụng một quy tắc gọi là *prefix-based intersection (giao dựa trên tiền tố)*. Quy tắc này nói rằng, nếu có hai Equivalence Class có cùng độ dài và có phần tử cuối cùng khác nhau, thì có thể kết hợp chúng bằng cách lấy giao của các tập hợp giao dịch của chúng. Kết quả là một *Equivalence Class* mới có độ dài lớn hơn một và có *prefix* là sự kết hợp của các *prefix* của hai *Equivalence Class* ban đầu. Ví dụ, nếu có hai *Equivalence Class* là {a, b} và {a, c}, thì có thể tạo ra *Equivalence Class* mới là {a, b, c} bằng cách lấy giao của các tập hợp giao dịch của {a, b} và {a, c}

4. *Giải thích cách biểu diễn dữ liệu*

- Thuật toán ECLAT sử dụng một cấu trúc dữ liệu gọi là ma trận dọc (vertical format) để biểu diễn dữ liệu. Trong ma trận dọc, mỗi hàng thể hiện một mục (item), và mỗi cột chứa danh sách các giao dịch (transactions) mà mục đó xuất hiện trong đó. Cấu trúc dữ liệu này giúp ECLAT hiệu quả trong việc tìm kiếm các itemsets phổ biến trong tập dữ liệu.

TID	Items
1	Bread,Butter,Jam
2	Butter,Coke
3	Butter,Milk
4	Bread,Butter,Coke
5	Bread,Milk
6	Butter,Milk
7	Bread,Milk
8	Bread,Butter,Milk,Jam
9	Bread,Butter,Milk

Item Set	TID set
Bread	1,4,5,7,8,9
Butter	1,2,3,4,6,8,9
Milk	3,5,6,7,8,9
Coke	2,4
Jam	1,8

Hình ảnh mô tả sự khác nhau giữa cách trình bày dữ liệu theo chiều ngang của thuật toán Apiori, FP-Growth và cách trình bày dữ liệu theo chiều dọc của thuật toán ECLAT[6]

- Giả sử chúng ta có một tập dữ liệu giao dịch gồm các giao dịch (transactions) và các mục (items) của mỗi giao dịch. Khi đó ma trận dọc được biểu diễn là:

Items	Transactions
A	T1, T3, T4
B	T1, T2, T4
C	T2, T3, T4
D	T1

- Trong ma trận dọc, mỗi hàng tương ứng với một mục (item), và mỗi cột chứa danh sách các giao dịch (transactions) mà mục đó xuất hiện trong đó.
- Dữ liệu được biểu diễn bằng ma trận dọc giúp ECLAT tìm kiếm và xử lý thông tin về tần suất xuất hiện của các itemsets một cách hiệu quả và không cần phải xây dựng cây dữ liệu lớn như trong FP-Growth.

5. *Các chỉ số đánh giá mẫu phổ biến và luật kết hợp có trong đồ án*[1]

Trong bài đồ án này, nhóm sử dụng hai độ đo để đánh giá luật kết hợp của thuật toán ECLAT là Độ tin cây (confidence) và Thang đo Lift

a. Độ tin cậy (Confidence)

$$\boldsymbol{conf}\ (\boldsymbol{X} => \boldsymbol{Y}) = \boldsymbol{P}(\boldsymbol{X}|\boldsymbol{Y}) = \frac{\boldsymbol{freq}(\boldsymbol{X} \cup \boldsymbol{Y})}{\boldsymbol{freq}(\boldsymbol{X})}$$

- Confidence là tỉ lệ giữa số giao dịch chứa cả X và Y trên số giao dịch chỉ chứa X. Độ tin cậy cho biết mức độ chính xác của một quy tắc kết hợp. Một quy tắc kết hợp được gọi là quy tắc kết hợp mạnh nếu độ tin cậy của nó lớn hơn hoặc bằng một ngưỡng minconf do người dùng định nghĩa.

b. Thang đo Lift

- Lift(X⇒Y) là chỉ số lift của quy tắc kết hợp X⇒Y, nghĩa là nếu mua mục X thì cũng mua mục Y:

- $lift\ (X \Rightarrow Y) = \frac{sup(X \Rightarrow Y)}{sup(X).sup(Y)} = \frac{conf(X \Rightarrow Y)}{sup(Y)}$
- $lift\ (X \Rightarrow Y) = \frac{P(X \cap Y)}{P(X).P(Y)} = \frac{P(Y|X)}{P(Y)}$

- Độ đo lift là chỉ số để đánh giá hiệu quả một quy tắc kết hợp. Độ đo lift cho biết mức độ tăng hoặc giảm của xác suất xuất hiện của một sản phẩm khi biết một sản phẩm khác đã xuất hiện
- Giá trị của lift có thể được diễn giải như sau:

➔ $lift\ (X \Rightarrow Y) = 1: P(Y|X) = P(Y) \Rightarrow X$ *và* Y *độc lập*

➔ $lift(X \Rightarrow Y) > 1:$ *Tương quan thuận*

➔ $lift(X \Rightarrow Y) < 1:$ *Tương quan nghịch*

6. *So sánh thuật toán ECLAT với các thuật toán tương tự (Apriori và FP-Growth)[4]*

	Thời gian chạy	Cấu trúc dữ liệu	Phương pháp tìm kiếm	Tần suất tính toán (số lần tính toán độ hỗ trợ của các item)
ECLAT	Xét với bộ dữ liệu lớn, có nhiều mục, giao dịch dài thù ECLAT có thời gian chạy nhanh hơn Apriori	Thuật toán ECLAT sử dụng định dạng dữ liệu dọc (vertical data format), trong đó mỗi mục là một tập hợp các giao dịch chứa nó	ECLAT sử dụng phương pháp tìm kiếm đệ quy	ECLAT tính tần suất xuất hiện của các tập luật kết hợp bằng cách đếm số lần xuất hiện của các sản phẩm trong các giao dịch
Apriori	Phù hợp với các tập dữ liệu nhỏ hoặc có kích thước trung bình nhưng khi áp dụng vào bộ dữ liệu lớn thì thời gian chạy chậm hơn ECLAT và FP-Growth	Apriori sử dụng cấu trúc dữ liệu dạng tập hợp (Horizontal Format). Như vậy, Apriori phải thực hiện nhiều lần lặp và tính toán tần suất xuất hiện nhiều hơn (Ứng với mỗi tập hợp mới)	Apriori sử dụng phương pháp tìm kiếm theo chiều ngang (Level-Wise Search)	Apriori tính tần suất xuất hiện bằng cách so sánh các tập luật kết hợp với các giao dịch
FP-Growth	Thuật toán FP-Growth có thời gian chạy tối ưu và có hiệu suất cao đối với	FP-Growth sử dụng cấu trúc cây FP để lưu trữ và khai thác các tập phổ	FP-Growth sử dụng phương pháp phát triển mẫu, tức là không sinh ra	FP-Growth có tần suất tính toán thấp nhất, vì nó không cần sinh ra các tập ứng

	các tập dữ liệu lớn	biến. Cây FP là một cấu trúc cây có gốc là null, mỗi cạnh chứa một mục, và mỗi nút biểu diễn một tập mục là nối các mục trên đường đi từ gốc đến nút đó	các tập ứng viên mà chỉ sử dụng cấu trúc cây FP để lưu trữ và khai thác các tập phổ biến.	viên mà chỉ sử dụng cấu trúc cây FP để lưu trữ và khai thác các tập phổ biến.

- Nhận xét:
  + Hiệu suất:
    - FP-Growth thường có hiệu suất tốt nhất trên các tập dữ liệu lớn và ít tốn thời gian so với Apriori.
    - ECLAT có thể hiệu quả tương đương hoặc vượt trội so với FP-Growth trong một số trường hợp.
  + Bộ nhớ:
    - ECLAT thường tốn ít bộ nhớ hơn FP-Growth vì không cần xây dựng cây.
    - FP-Growth yêu cầu ít bộ nhớ hơn Apriori vì không cần lưu trữ itemsets candidate.

7. Tác động của mật độ dữ liệu đến các thuật toán Apriori, ECLAT và FP-Growth

- Mật độ tập dữ liệu chỉ ra tỷ lệ phần trăm của các giỏ hàng có chứa các tập mục thường xuyên. Khi mật độ tập mục thường xuyên tăng lên thì thời gian xử lý của Apriori, Eclat và FP-Growth cũng tăng theo như hình sau đây.

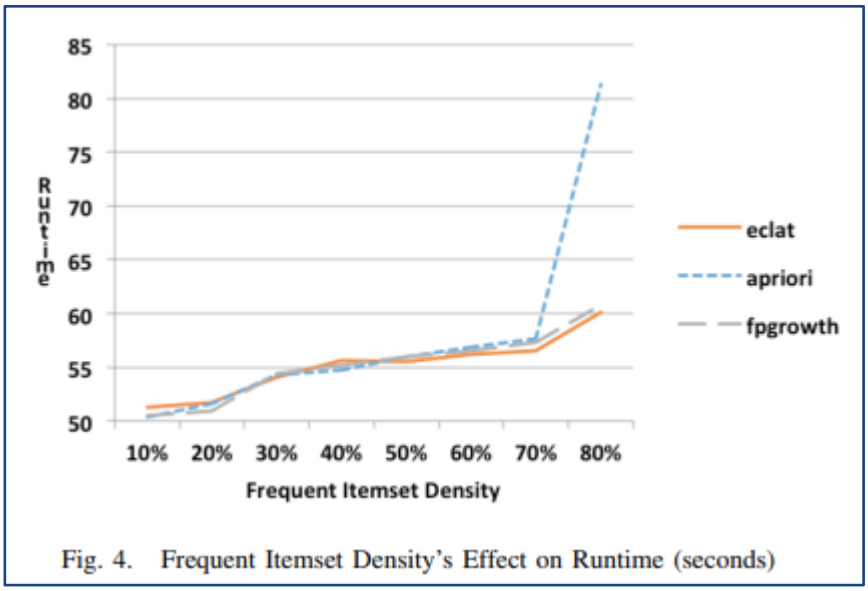


Fig. 4. Frequent Itemset Density's Effect on Runtime (seconds)

Ảnh hưởng của mật độ dữ liệu đến thời gian chạy của các thuật toán [4]

8. *Tác động của số lượng dữ liệu đến các thuật toán Apriori, Eclat và FP-Growth*

- Kích thước giỏ hàng chỉ ra số lượng tối đa các mặt hàng trên mỗi dòng. Kích thước giỏ hàng lớn hơn có nghĩa là các tập mục thường xuyên cũng sẽ lớn hơn. Điều này làm tăng kích thước của các cấu trúc dữ liệu được sử dụng để chứa các tập mục này. Những cấu trúc dữ liệu lớn hơn này yêu cầu nhiều bộ nhớ hơn để lưu trữ và thời gian xử lý lớn hơn để duyệt chúng. Hình minh họa ảnh hưởng của việc tăng kích thước giao dịch đến hiệu năng của ba thuật toán. như sau:

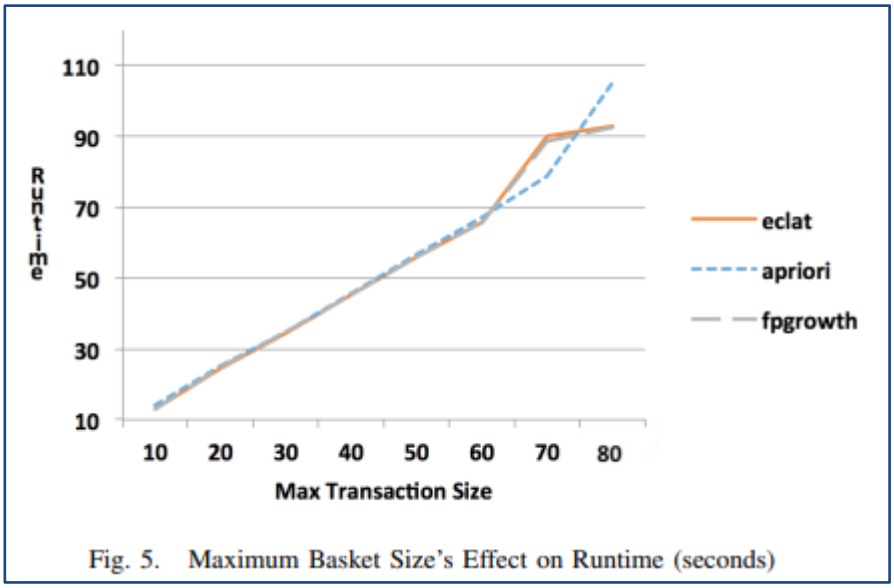


Fig. 5. Maximum Basket Size's Effect on Runtime (seconds)

Ảnh hưởng của kích thước dữ liệu đến thời gian chạy của ba thuật toán [4]

- *Nhận xét:* Biểu đồ này cho thấy kết quả của việc chạy 10 triệu giỏ hàng, với mật độ tập mục thường xuyên là 50% ở các kích thước giỏ hàng tối đa khác nhau. Ba thuật toán cho thấy hiệu năng gần như giống nhau là O(N) cho các kích thước giỏ hàng từ 60 trở xuống. Khi vượt quá 60, Apriori dường như tăng nhanh hơn hai thuật toán còn lại. Điều này có thể là do bộ nhớ sử dụng của Apriori tăng lên. Nhưng Apriori hoạt động tốt nhất trong khoảng từ 60 đến 70 kích thước giao dịch tối đa.

CHƯƠNG 3. CÁC BƯỚC XÂY DỰNG THUẬT TOÁN

1. *Thuật toán ECLAT*

Algorithm 3 Eclat Frequent Itemset Algorithm

INPUT: A file $\mathcal{D}$ consisting of baskets of items, a support threshold σ, and an item prefix I, such that $I \subseteq \mathcal{J}$.
OUTPUT: A list of itemsets $\mathcal{F}[I](\mathcal{D}, \sigma)$ for the specified prefix.
METHOD:
$\mathcal{F}[i] \leftarrow \{\}$
for all $i \in \mathcal{J}$ occurring in $\mathcal{D}$ **do**
    $\mathcal{F}[I] := \mathcal{F}[I] \cup \{I \cup \{i\}\}$
Create $\mathcal{D}_i$
    $\mathcal{D}_i \leftarrow \{\}$
    for all $j \in \mathcal{J}$ occurring in $\mathcal{D}$ such that $j > i$ **do**
        $C \leftarrow \text{cover}(\{i\}) \cap \text{cover}(\{j\})$
        if $|C| \geq \sigma$ **then**
            $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{j, C\}$
Depth-first recursion
    Compute $\mathcal{F}[I \cup i](\mathcal{D}_i, \sigma)$
    $\mathcal{F}[I] := \mathcal{F}[I] \cup \mathcal{F}[I \cup i]$

Thuật toán Eclat trình bày bởi Goethals [4]

2. *Các bước xây dựng thuật toán* [5]

a. Xây dựng ma trận dọc (Vertical Format)

- Từ dữ liệu ban đầu ta tạo danh sách và cách mục (items) và danh sách các giao dịch (transactions). Ta trích 8 dòng và 6 cột từ bộ dữ liệu ‘Groceries 3’ để ví dụ minh họa:

	Apple	Bread	Butter	Cheese	Corn	Dill
1	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE

2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
4	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
5	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
6	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE
7	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
8	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE

➔ *minSup = 25% = 0.25 và threshold = 0.25*8 =2*

- Danh sách các mục là: ['Apple', 'Bread', 'Butter', 'Cheese']. Ta chuyển đổi các giá trị ‘TRUE’ và ‘FALSE’ của bộ dữ liệu thành các giá trị ‘1’ và ‘0’ tương ứng để tiến hành tạo bảng dữ liệu theo chiều dọc cho ECLAT như sau:

	Apple	Bread	Butter	Cheese	Corn	Dill
1	0	1	0	0	1	1
2	0	0	0	0	0	0
3	1	0	1	0	0	1
4	0	0	1	1	0	1
5	1	1	0	0	0	0
6	1	1	1	1	0	1
7	1	0	0	1	0	0
8	1	0	0	0	1	1

- Tiếp theo ta tạo biểu diễn dọc (vertical representation) cho cơ sở dữ liệu:

Sản phẩm	ID
Apple	3, 5, 6, 7, 8
Bread	1, 5, 6
Butter	3, 4, 6
Cheese	4, 6, 7
Corn	1, 8
Dill	1, 3, 4, 6, 8

- Tạo ma trận dọc từ danh sách các mục và giao dịch:
  - Apple: [3, 5, 6, 7, 8]
  - Bread: [1, 5, 6]
  - Butter: [3, 4, 6]
  - Cheese: [4, 6, 7]
  - Corn: [1, 8]
  - Dill: [1, 3, 4, 6, 8]

b. Tạo Equivalence Class đầu tiên

- Tạo danh sách giao dịch đầu tiên cho mỗi mục với K=1

Sản phẩm	ID
Apple	3, 5, 6, 7, 8
Bread	1, 5, 6
Butter	3, 4, 6
Cheese	4, 6, 7

Corn	1, 8
Dill	1, 3, 4, 6, 8

- Quan sát Equivalence Class để loại bỏ các tập mục không thường xuyên với threshold = 2. Và sau đây là kết quả sau khi loại bỏ tập mục không thường xuyên

Sản phẩm	ID
Apple	3, 5, 6, 7, 8
Bread	1, 5, 6
Butter	3, 4, 6
Cheese	4, 6, 7
Corn	1, 8
Dill	1, 3, 4, 6, 8

- ECLAT xuất ra các tập mục thường xuyên có 1 item: {Apple}, {Bread}, {Butter}, {Cheese}, {Corn}, {Dill}

c. Kết hợp các mục có cùng tiền tố (Equivalence Class) vừa xác định để tạo ra các lớp tương đương có kích thước K+1

- Từ bảng tập mục thường xuyên với K = 1 ta kết hợp các tập mục tương đương để tạo ra các lớp tương đương có kích thước K = 2 và có được kết quả như sau

Sản phẩm	ID
Apple, Bread	5, 6

Apple, Butter	3, 6
Apple, Cheese	6, 7
Apple, Corn	8
Apple, Dill	3, 6, 8
Bread, Butter	6
Bread, Cheese	6
Bread, Corn	1
Bread, Dill	1, 6
Butter, Cheese	4, 6
Butter, Dill	3, 4 ,6
Cheese, Dill	4, 6
Corn, Dill	1, 8

- Sau đó áp dụng ngưỡng Threshold = 2 để loại bỏ các tập mục không thường xuyên (infrequent itemsets)

Sản phẩm	ID
Apple, Bread	5, 6
Apple, Butter	3, 6
Apple, Cheese	6, 7
Apple, Dill	3, 6, 8
Bread, Dill	1, 6
Butter, Cheese	4, 6
Butter, Dill	3, 4 ,6

Cheese, Dill	4, 6
Corn, Dill	1, 8

- ECLAT trả về các tập mục thường xuyên là: {Apple, Bread}, {Apple, Butter}, {Apple, Cheese}, {Apple, Corn}, {Apple, Dill}, {Bread, Dill}, {Butter, Cheese}, {Butter, Dill}, {Cheese, Dill}, {Corn, Dill}

d. Tiếp tục xử lý đệ quy để tìm các itemsets phổ biến

- Ta tiếp tục xử lý bảng cơ sở dữ liệu dọc ở trên (bảng K=2) thành bảng có K= 3 như sau

Sản phẩm	ID
Apple, Bread, Butter	6
Apple, Bread, Cheese	6
Apple, Bread, Dill	6
Apple, Butter, Cheese	6
Apple, Butter, Dill	3, 6
Apple, Cheese, Dill	6
Apple, Corn, Dill	8
Butter, Cheese, Dill	4, 6
Apple, Butter, Dill	3, 6
Butter, Cheese, Dill	4, 6

- Áp dụng tần suất tối thiểu (threshold = 2) tìm ra mẫu phổ biến như sau

Sản phẩm	ID
Apple, Butter, Dill	3, 6
Butter, Cheese, Dill	4, 6

e. Kết quả luật kết hợp

- Từ các itemsets phổ biến đã tìm được, thuật toán ECLAT có thể ghép chúng lại để tạo thành các luật kết hợp. Trong ví dụ này các item có độ phổ biến mạnh là {Apple, Butter, Dill} và {Butter, Cheese, Dill} với ngưỡng threshold = 2

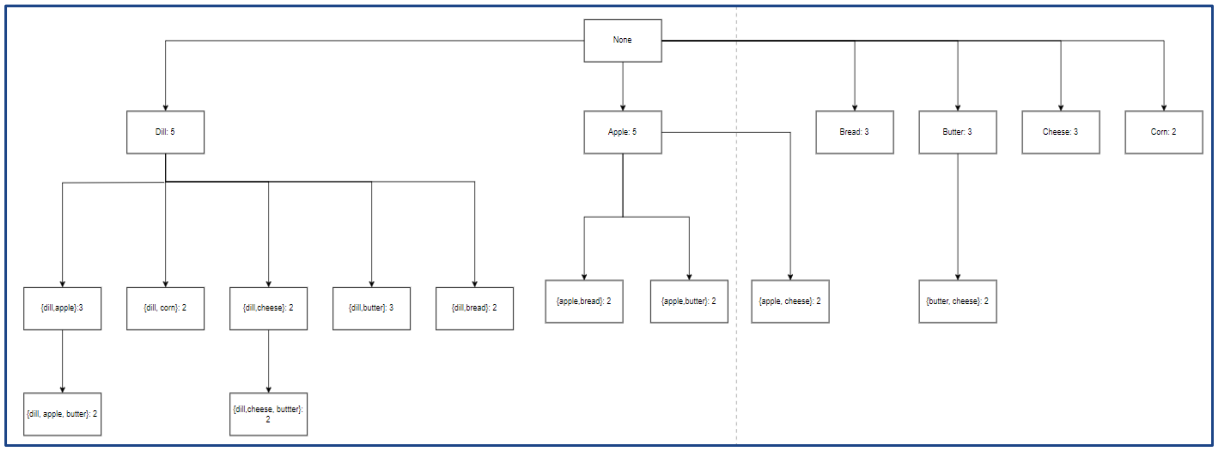


Diagram để minh họa cho ví dụ trên

Nhìn vào Diagram có thể thấy được:

- Ứng với k = 1 (hàng thứ 2) sẽ nhận 6 itemsets
- Ứng với k = 2 (hàng thứ 3) sẽ nhận 9 itemsets
- Ứng với k =3 (hàng cuối cùng) sẽ nhận 2 itemsets

CHƯƠNG 4. TỔNG QUAN BỘ DỮ LIỆU

1. Quan sát bộ dữ liệu

- Dùng hàm read_csv của thư viện Pandas để đọc dữ liệu và trả về một Dataframe và xem một phần nhỏ của DataFrame để kiểm tra dữ liệu, xác nhận rằng nó đã được đọc và lưu trữ đúng cách.

```
df = pd.read_csv('/content/gdrive/MyDrive/Data Mining/Groceries 3.csv')
data = pd.DataFrame(df)
data.head()
```

	Unnamed: 0	Apple	Bread	Butter	Cheese	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Sugar	Unicorn	Yogurt	Chocolate
0	0	False	True	False	False	True	True	False	True	False	False	False	False	True	False	True	True
1	1	False	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False
2	2	True	False	True	False	False	True	False	True	False	True	False	False	False	False	True	True
3	3	False	False	True	True	False	True	False	False	False	True	True	True	False	False	False	False
4	4	True	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False

2. *Kiểm tra missing value*

- Sử dụng phương thức isnull() của DataFrame sẽ trả về một Series chứa các giá trị True cho các giá trị null và False cho các giá trị không null. Sau đó, phương thức sum() sẽ trả về tổng số giá trị True trong mỗi cột

```
groceries.isnull().sum()
```

```
Apple           0
Bread           0
Butter          0
Cheese          0
Corn            0
Dill            0
Eggs            0
Ice cream       0
Kidney Beans    0
Milk            0
Nutmeg          0
Onion           0
Sugar           0
Unicorn         0
Yogurt          0
Chocolate       0
dtype: int64
```

- Sau khi hiển thị kết quả ra màn hình ta có thể nhận thấy rằng các thuộc tính của bộ dữ liệu không có giá trị thiếu

3. *Quan sát tổng quan các cột có trong bộ dữ liệu*

- Để có cái nhìn tổng quan trong lúc phân tích luật kết hợp ta cần biết được tất cả các thuộc tính của bộ dữ liệu

```
groceries.columns
```

```
Index(['Apple', 'Bread', 'Butter', 'Cheese', 'Corn', 'Dill', 'Eggs',
       'Ice cream', 'Kidney Beans', 'Milk', 'Nutmeg', 'Onion', 'Sugar',
       'Unicorn', 'Yogurt', 'Chocolate'],
      dtype='object')
```

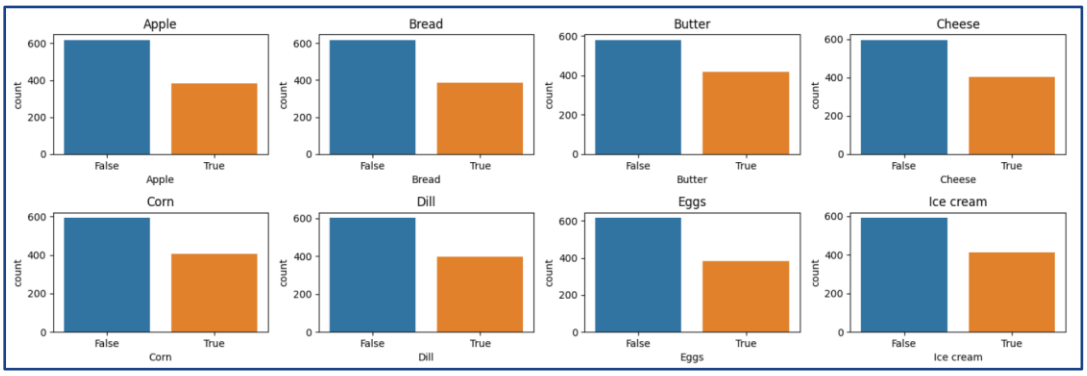
4. *Thống kê và trực quan giá trị của từng cột có trong bộ dữ liệu*

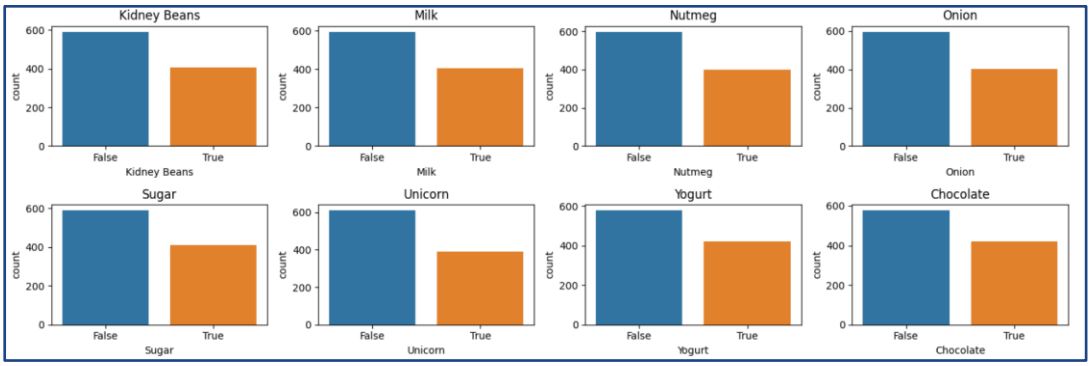
- Để biết chi tiết hơn về các thuộc tính ta thống kê ra những giá trị có trong thuộc tính và biểu diễn trực quan để có cái nhìn rõ hơn về bộ dữ liệu

```
# Lặp qua từng cột trong DataFrame 'df'
for column in groceries.columns:
    # Kiểm tra nếu cột chứa kiểu dữ liệu bool (boolean)
    if groceries[column].dtype == bool:
        # Thống kê số lượng giá trị True và False trong cột
        value_counts = groceries[column].value_counts()
        print(f"Thống kê giá trị của cột '{column}':")
        print(value_counts)
```

```
Thống kê giá trị của cột 'Apple':
False    616
True     383
Name: Apple, dtype: int64
Thống kê giá trị của cột 'Bread':
False    615
True     384
Name: Bread, dtype: int64
Thống kê giá trị của cột 'Butter':
False    579
True     420
```

```
plt.figure(figsize=(15, 10))
for i, col in enumerate(groceries.columns):
    plt.subplot(4, 4, i + 1)
    sns.countplot(x=col, data=df)
    plt.title(col)
plt.tight_layout()
plt.show()
```





Biểu đồ trực quan các giá trị trong bộ dữ liệu

- Nhận xét: Tất cả các thuộc tính chỉ có hai giá trị là 'True' và 'False'. Nhìn chung, giá trị 'False' chiếm giá trị cao hơn 'True' tại tất cả các thuộc tính và các giá trị giữa các thuộc tính có sự tương đồng, không chênh lệch nhiều với nhau. Vì cấu trúc ECLAT dựa trên cấu trúc dữ liệu vertical, nếu giá trị 'True' đồng đều giữa các thuộc tính, nghĩa là mỗi thuộc tính xuất hiện trong khoảng 50% số giao dịch, thì kích thước của các tập giao dịch sẽ lớn, dẫn đến tính giao của các tập giao dịch sẽ tốn nhiều thời gian và bộ nhớ làm giảm hiệu quả của thuật toán ECLAT trong việc khai thác các tập hợp thuộc frequent itemsets

CHƯƠNG 5. ỨNG DỤNG GIẢI THUẬT TRÊN BỘ DỮ LIỆU

Bài toán mà nhóm đặt ra là khi một chủ cửa hàng tạp hóa muốn muốn quản lý tối ưu những sản phẩm trong cửa hàng, phục vụ cho việc tồn kho và muốn biết được một khách hàng khi đến cửa hàng của mình sẽ mua những món đồ gì. Để có thể giải quyết được bài toán này, nhóm sẽ phân tích các dữ liệu về các giao dịch của khách hàng qua bộ dữ liệu 'Groceries 3' để tính toán được tần suất những món đồ mà khách hàng đã mua và những sản phẩm nào thường được mua chung với nhau khi mua sắm tại cửa hàng

1. *Tiền xử lý dữ liệu*

```
df.drop(['Unnamed: 0'], axis = 1, inplace = True)
```

```
transactions = []
for i in range(df.shape[0]):
    transaction = list(df.columns[df.iloc[i] == 1])
    transactions.append(transaction)
```

2. *EDA*

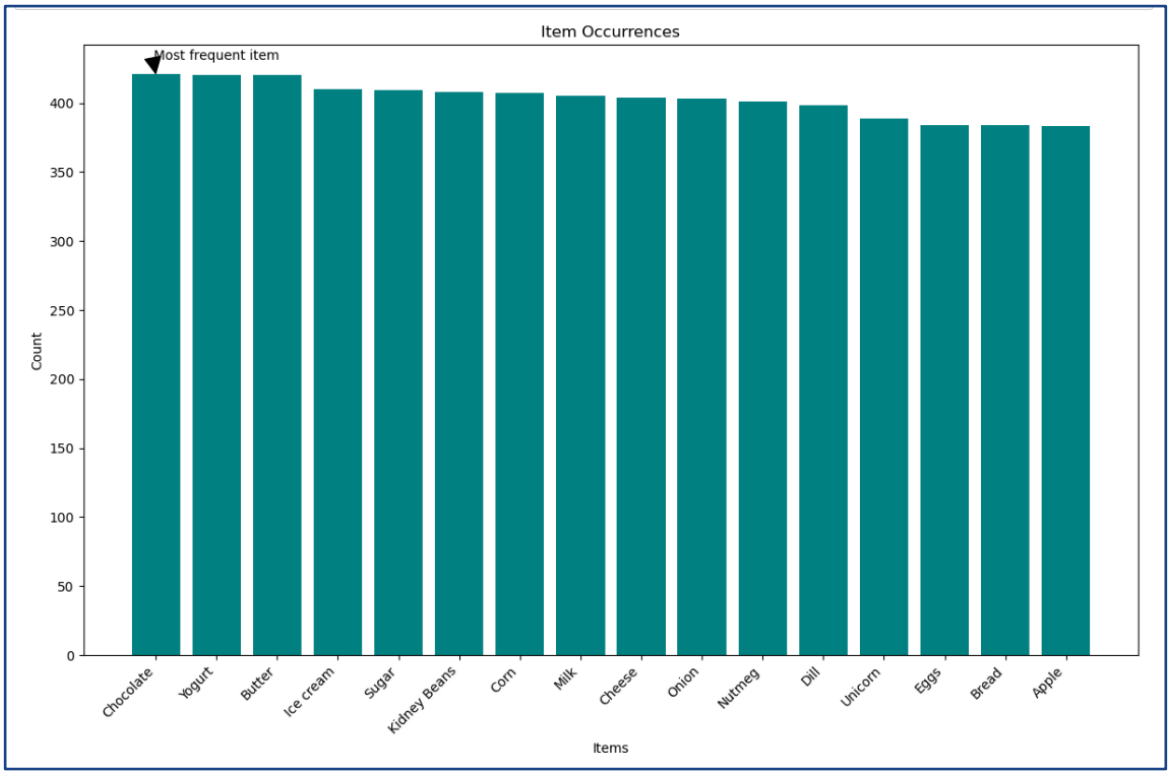
```
#Số lần xuất hiện của các item
item_counts = np.sum(df, axis=0)
items = df.columns.tolist()

#Sắp xếp các mặt hàng theo số lần xuất hiện
sorted_indices = np.argsort(item_counts)[::-1]
sorted_items = [items[i] for i in sorted_indices]
sorted_counts = item_counts[sorted_indices]

plt.figure(figsize=(12, 8))  # Tăng kích thước để cải thiện tính rõ ràng
plt.bar(sorted_items, sorted_counts, color='teal') #màu sắc
plt.title('Item Occurrences')
plt.xlabel('Items')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')

#Thêm chú thích cho mặt hàng phổ biến nhất
plt.annotate('Most frequent item', xy=(0, sorted_counts[0]), xytext=(1, sorted_counts[0] + 10),
             arrowprops=dict(facecolor='black', shrink=0.05),
             ha='center')

plt.tight_layout()
plt.show()
```



Biểu đồ thể hiện tần suất xuất hiện của các item

- Nhận xét: không có sự chênh lệch nhiều giữa các items

- Chúng ta tiến hành đếm số lượng các sản phẩm xuất hiện cùng nhau:

```
# Tạo một danh sách tất cả các mặt hàng duy nhất từ tất cả các giao dịch, sau đó sắp xếp chúng
all_items = sorted(set(item for transaction in transactions for item in transaction))

# Tạo một bộ đếm để giữ số lượng xuất hiện của mỗi cặp sản phẩm
pair_counts = Counter()

# Đếm số lần xuất hiện của mỗi cặp trong các giao dịch
for transaction in transactions:
    # Sử dụng hàm combinations để tìm tất cả các cặp duy nhất trong giao dịch
    pairs = combinations(sorted(transaction), 2)
    pair_counts.update(pairs)

# Tạo một DataFrame trống với các hàng và cột là các mặt hàng
item_matrix = pd.DataFrame(index=all_items, columns=all_items).fillna(0)

# Điền vào DataFrame với số lượng các cặp
for (item1, item2), count in pair_counts.items():
    item_matrix.at[item1, item2] = count
    item_matrix.at[item2, item1] = count  # Cập nhật để ma trận là đối xứng

# Hiển thị DataFrame kết quả
item_matrix.astype(int)  # Chuyển đổi sang kiểu int để hiển thị
```

- Tạo ma trận biểu hiện các item cùng xuất hiện với nhau (sự xuất hiện đồng thời) => khi chạy eclat để suy ra tập phổ biến rồi so sánh có giống nhau hay không

	Apple	Bread	Butter	Cheese	Chocolate	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Sugar	Unicorn	Yogurt
Apple	0	154	188	162	183	186	179	156	172	176	184	172	167	182	166	187
Bread	154	0	180	173	185	174	160	157	181	167	174	171	178	179	168	193
Butter	188	180	0	182	202	191	175	173	207	202	198	198	197	196	182	191
Cheese	162	173	182	0	186	182	177	169	187	200	172	192	185	187	170	181
Chocolate	183	185	202	186	0	192	199	182	202	191	211	186	196	188	186	198
Corn	186	174	191	182	192	0	180	180	192	195	193	181	184	187	177	190
Dill	179	160	175	177	199	180	0	157	185	172	190	173	192	179	168	185
Eggs	156	157	173	169	182	180	157	0	157	169	176	172	174	170	168	186
Ice cream	172	181	207	187	202	192	185	157	0	196	177	187	192	195	185	182
Kidney Beans	176	167	202	200	191	195	172	169	196	0	199	189	170	187	184	194
Milk	184	174	198	172	211	193	190	176	177	199	0	182	182	186	183	190
Nutmeg	172	171	198	192	186	181	173	172	187	189	182	0	195	193	165	192
Onion	167	178	197	185	196	184	192	174	192	170	182	195	0	190	175	192
Sugar	182	179	196	187	188	187	179	170	195	187	186	193	190	0	181	191
Unicorn	166	168	182	170	186	177	168	168	185	184	183	165	175	181	0	184
Yogurt	187	193	191	181	198	190	185	186	182	194	190	192	192	191	184	0

Ma trận thể hiện các item xuất hiện đồng thời với nhau

3. *Áp dụng ECLAT lên bộ dữ liệu*

a. Tìm ngưỡng minSup phù hợp cho bài toán

- Nhóm tiến hành chạy dòng lặp minSup từ 0.1 đến 0.21 để tìm ra độ dài của các tập phổ biến ứng với từng minSup:

```
min_support_values = []
size_association = []
for j in range(10, 22):
    i = j / 100  # Convert to the desired scale
    rule = eclat_setup(transactions, min_support = i)
    df_mode = pd.DataFrame.from_dict(rule, orient='index', columns = ['Support'])
    length = len(df_mode)
    min_support_values.append(i)
    size_association.append(length)
```

- Lưu kết quả vòng lặp vào dataframe "result_df_mode"

```
result_df_mode = pd.DataFrame({
    'minsup': min_support_values,
    'size_association': size_association,
})
result_df_mode
```

✓ 0.0s

- Và nhóm thu được kết quả:

	minsup	size_association
0	0.10	153
1	0.11	122
2	0.12	120
3	0.13	120
4	0.14	120
5	0.15	120
6	0.16	115
7	0.17	104
8	0.18	78
9	0.19	37
10	0.20	6
11	0.21	1

- Nhóm nhận thấy ở giá trị minSup 10% có 153 luật mẫu phổ biến trong khi 11% có ít mẫu phổ biến hơn (122 mẫu). Với việc chọn minsup quá cao gây mất mát thông tin quá thấp sẽ đưa ra mẫu không có ý nghĩa. Vì vậy nhóm quyết định chọn minSup ở mức 11%

b. Sử dụng thư viện

- Áp dụng thuật toán với min support = 0.11, min combination = 2

```
df = pd.DataFrame(transactions )
df
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	Bread	Corn	Dill	Ice cream	Sugar	Yogurt	Chocolate	None	None	None	None	None	None
1	Milk	None	None	None	None	None	None	None	None	None	None	None	None
2	Apple	Butter	Dill	Ice cream	Milk	Yogurt	Chocolate	None	None	None	None	None	None
3	Butter	Cheese	Dill	Milk	Nutmeg	Onion	None	None	None	None	None	None	None
4	Apple	Bread	None	None	None	None	None	None	None	None	None	None	None
...	...	...	...	...	...	...	...	...	...	...	...	...	...
994	Bread	Eggs	Unicorn	Chocolate	None	None	None	None	None	None	None	None	None
995	Apple	Corn	Kidney Beans	Milk	Nutmeg	Yogurt	None	None	None	None	None	None	None
996	Apple	Corn	Dill	Sugar	Chocolate	None	None	None	None	None	None	None	None
997	Butter	Cheese	Corn	Eggs	Ice cream	Kidney Beans	Nutmeg	Sugar	Yogurt	Chocolate	None	None	None
998	Milk	Chocolate	None	None	None	None	None	None	None	None	None	None	None

999 rows × 13 columns

```
eclat = ECLAT(df2, verbose=True)
```
✓ 0.1s

```
100%|██████████| 17/17 [00:00<00:00, 212.01it/s]
100%|██████████| 17/17 [00:00<00:00, 5676.55it/s]
100%|██████████| 17/17 [00:00<00:00, 987.22it/s]
```

```
index, support = eclat.fit(min_support=0.11,min_combination=2)
```
✓ 1.8s

```
Combination 2 by 2
0it [00:00, ?it/s]
120it [00:00, 217.60it/s]
Combination 3 by 3
560it [00:01, 448.77it/s]
```

- Nhóm thu được kết quả tương ứng:

```
df_sup = pd.DataFrame(list(support.items()), columns=['Items', 'Support'])
df_sup = df_sup.sort_values(by='Support', ascending=False).reset_index()
df_sup = df_sup.drop(columns = 'index')
df_sup
```
✓ 0.0s

	Items	Support
0	Milk & Chocolate	0.211211
1	Ice cream & Butter	0.207207
2	Ice cream & Chocolate	0.202202
3	Kidney Beans & Butter	0.202202
4	Butter & Chocolate	0.202202
...	...	...
117	Eggs & Bread	0.157157
118	Apple & Eggs	0.156156
119	Apple & Bread	0.154154
120	Dill & Milk & Chocolate	0.114114
121	Ice cream & Kidney Beans & Butter	0.110110

122 rows × 2 columns

Kết quả của thuật toán ECLAT khi chạy bằng

c. Tự cài đặt

Dựa trên ý tưởng của thuật toán Eclat, ta định nghĩa hàm như sau :

```
def eclat_setup(dataset, min_support, min_combination=2):
    transactions = len(dataset)
    min_support_count = min_support * transactions

    items = {}
    for transaction in dataset:
        for item in transaction:
            items.setdefault(item, 0)
            items[item] += 1

    frequent_items = {item: count for item, count in items.items() if count >= min_support_count}

    frequent_itemsets = {}
    for size in range(min_combination, len(frequent_items) + 1):
        itemsets = [set(itemset) for itemset in combinations(frequent_items.keys(), size)]
        for itemset in itemsets:
            count = sum(all(item in transaction for item in itemset) for transaction in dataset)
            if count >= min_support_count:
                frequent_itemsets[tuple(sorted(itemset))] = count / transactions

    return frequent_itemsets
```

Chọn ngưỡng minsup xấp xỉ 0 để lưu lại hết kết quả của thuật toán

```
min_support = 0.01
result_eclat = eclat_setup(transactions, min_support)
```

Chọn ngưỡng min support sấp xỉ 0 để lưu lại tất cả các kết quả của thuật toán

```
df_eclat = pd.DataFrame.from_dict(result_eclat, orient='index', columns = ['Support'])
df_eclat.sort_values(by=['Support'], ascending=True)
```

	Support
(Apple, Eggs, Kidney Beans, Milk, Onion, Unicorn, Yogurt)	0.010010
(Apple, Butter, Chocolate, Ice cream, Milk, Nutmeg)	0.010010
(Apple, Butter, Chocolate, Ice cream, Milk, Onion)	0.010010
(Apple, Butter, Chocolate, Eggs, Ice cream, Milk)	0.010010
(Apple, Cheese, Chocolate, Ice cream, Milk, Onion)	0.010010
...	...
(Butter, Kidney Beans)	0.202202
(Butter, Chocolate)	0.202202
(Chocolate, Ice cream)	0.202202
(Butter, Ice cream)	0.207207
(Chocolate, Milk)	0.211211

12208 rows × 1 columns

Ta cũng áp dụng thuật toán với min support = 0.11 và min combination = 2 để có thể so sánh kết quả so với dùng thư viện

```
result_eclat = eclat_setup(transactions, 0.11)
df_eclat = pd.DataFrame.from_dict(result_eclat, orient='index', columns = ['Support'])
df_eclat.sort_values(by=['Support'], ascending= False)
```

✓ 48.5s

	Support
(Chocolate, Milk)	0.211211
(Butter, Ice cream)	0.207207
(Butter, Kidney Beans)	0.202202
(Butter, Chocolate)	0.202202
(Chocolate, Ice cream)	0.202202
...	...
(Eggs, Ice cream)	0.157157
(Apple, Eggs)	0.156156
(Apple, Bread)	0.154154
(Chocolate, Dill, Milk)	0.114114
(Butter, Ice cream, Kidney Beans)	0.110110

122 rows × 1 columns

- Nhận xét: Kết quả của việc sử dụng thư viện và nhóm tự cài đặt là giống nhau về mặt kết quả, vì vậy có thể khẳng định được về mặt bản chất nhóm đã cài đặt được đúng ý tưởng của thuật toán.

Nhóm tiến hành lọc ra 10 tập phổ biến nhất để rút ngắn thời gian chạy:

```
# Sắp xếp df_eclat theo 'Support' và lấy ra top N
top_n = 10
top_rules = df_eclat.sort_values('Support', ascending=False).head(top_n)
```

	Support
(Chocolate, Milk)	0.211211
(Butter, Ice cream)	0.207207
(Butter, Kidney Beans)	0.202202
(Butter, Chocolate)	0.202202
(Chocolate, Ice cream)	0.202202
(Cheese, Kidney Beans)	0.200200
(Chocolate, Dill)	0.199199
(Kidney Beans, Milk)	0.199199
(Butter, Milk)	0.198198
(Butter, Nutmeg)	0.198198

Tiếp theo, nhóm tiến hành trực quan hóa top 10 tập phổ biến bằng Network Graph:

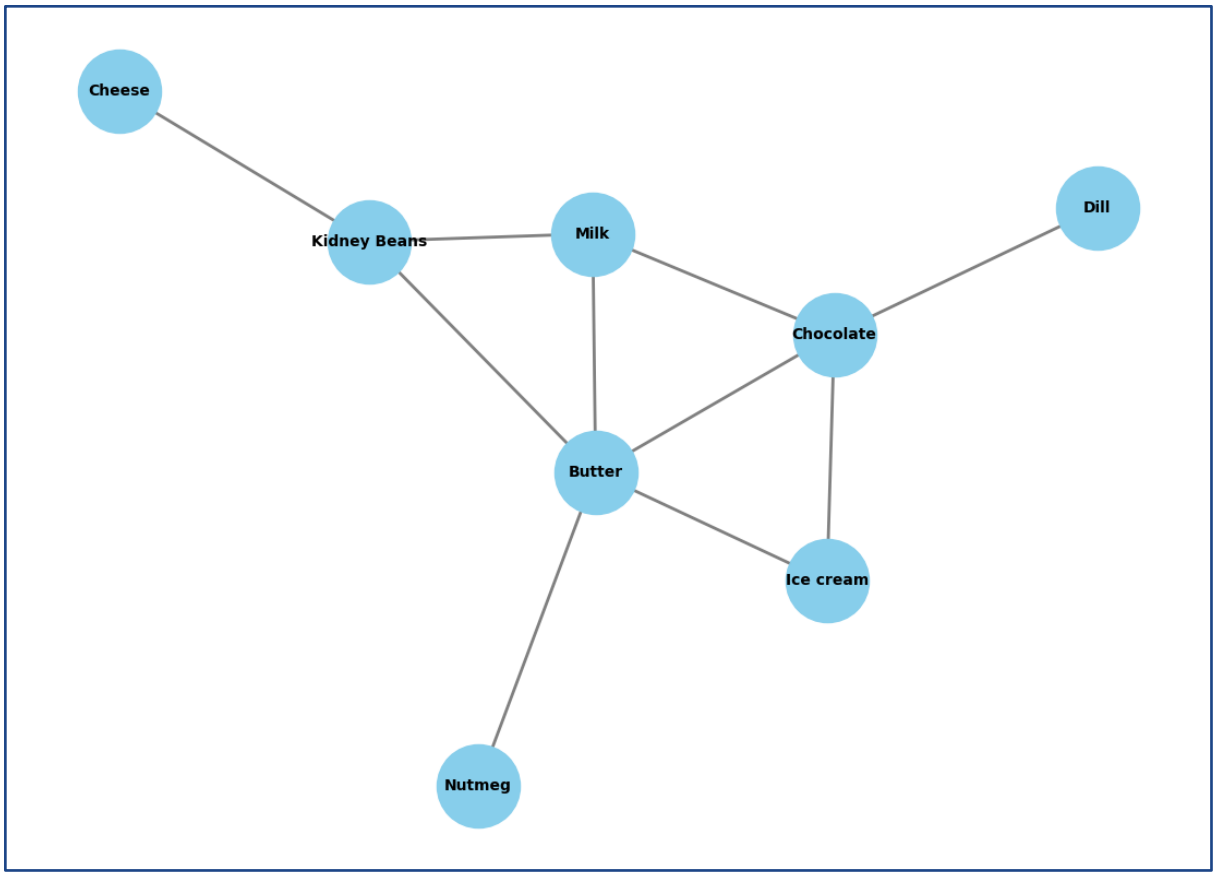
```
#Tạo graph
G = nx.Graph()

# Thêm các nút và cạnh vào graph
for index, row in top_rules.iterrows():
    items = index  # Lấy các items từ index của DataFrame
    for item in items:
        G.add_node(item)  # Thêm nút cho mỗi item
    # Thêm cạnh giữa các items trong cùng một tuple
    # Vì mỗi tuple có 2 items, ta sẽ thêm cạnh giữa cặp items này
    G.add_edge(items[0], items[1])
#Spring layout để bố trí các nút
pos = nx.spring_layout(G, k=0.25, iterations=20)

#Vẽ graph
fig, ax = plt.subplots(figsize=(14, 10))
nx.draw(G, pos, ax=ax, with_labels=True, node_size=3000, node_color='skyblue', font_size=10, font_weight='bold', width=2, edge_color='gray')

#Hiển thị
plt.show()
✓ 0.1s
```

Biểu đồ tương ứng



Nhận xét :

- Có vẻ như 'Butter' là một nút trung tâm, nối với nhiều sản phẩm khác, điều này có thể cho thấy 'Butter' thường được mua cùng với nhiều loại mặt hàng khác. Trong ngữ cảnh kinh doanh, 'Butter' có thể là một sản phẩm mấu chốt, có khả năng kích thích mua hàng chéo.
- 'Milk' và 'Chocolate' cũng có vẻ là các nút quan trọng, cả hai đều được kết nối trực tiếp với 'Butter' và với nhau, phản ánh mối quan hệ mua chung mạnh mẽ giữa ba sản phẩm này.

d. Tìm các luật kết hợp mạnh

Xây dựng hàm tính toán độ chính xác confidence với ngưỡng tin cậy minConf tương ứng cho các luật kết hợp:

```
total_transactions = len(transactions)
```

✓ 0.0s

```
#Số lần xuất hiện của các item
item_counts = np.sum(df, axis=0)
```

- Hàm tính Confidence:

```python
def calculate_confidence(top_rules_df, min_conf, total_transactions):

    # Bây giờ hãy tính độ tin cậy cho từng rule
    confidence_list = []
    for itemset, support in zip(top_rules_df.index, top_rules_df['Support']):
        for item in itemset:
            # Phần còn lại của itemet là consequent
            consequent = tuple(set(itemset) - set([item]))
            # Confience là độ hỗ trợ của toàn bộ tập được chia cho độ hỗ trợ của Antecedent
            confidence = (support * total_transactions) / (item_counts[item])
            if confidence >= min_conf:
                confidence_list.append((item, consequent, support,confidence))

    # Tạo DataFrame
    confidence_df = pd.DataFrame(confidence_list, columns=['Antecedent', 'Consequent', 'Support','Confidence'])

    # Xóa dấu ngoặc đơn khỏi cột Consequent
    confidence_df['Consequent'] = confidence_df['Consequent'].apply(lambda x: str(x).strip("(',')"))

    return confidence_df
```

✓ 0.0s

Nhóm quyết định chọn ngưỡng minConf để tìm ra các luật kết hợp mạnh:

```python
# Tính toán DataFrame cho độ tin cậy confidence
confidence_df = calculate_confidence(top_rules, 0.495, total_transactions)

# Hiển thị kết quả
confidence_df
```

✓ 0.0s

	Antecedent	Consequent	Support	Confidence
0	Chocolate	Milk	0.211211	0.501188
1	Milk	Chocolate	0.211211	0.520988
2	Ice cream	Butter	0.207207	0.504878
3	Kidney Beans	Butter	0.202202	0.495098
4	Cheese	Kidney Beans	0.200200	0.495050
5	Dill	Chocolate	0.199199	0.500000

Nhóm tiến hành xây dựng hàm tính toán độ tương quan lift cho các luật kết hợp:

- Hàm tính toán độ tương quan - LIFT:

```python
def calculate_lift(top_rules_df, total_transactions):

    # tính lift cho từng rule
    lift_list = []
    for antecedent, consequent, confidence in zip(top_rules_df['Antecedent'], top_rules_df['Consequent'], top_rules_df['Confidence']):

        # Tính toán lift
        lift_numerator = confidence
        sup_consequent = item_counts[consequent]/total_transactions
        lift = lift_numerator / sup_consequent

        lift_list.append((antecedent, consequent, lift))

    # Tạo DataFrame
    lift_df = pd.DataFrame(lift_list, columns=['Antecedent', 'Consequent', 'Lift'])

    # Xóa dấu ngoặc đơn khỏi cột Consequent
    lift_df['Consequent'] = lift_df['Consequent'].apply(lambda x: str(x).strip("(',')"))

    return lift_df
```
✓ 0.0s

- Và thu được kết quả của độ đo dựa trên các luật kết hợp mạnh như sau:

```python
# Tính toán DataFrame cho độ tương quan Lift
lift_df = calculate_lift(confidence_df, total_transactions)

# Hiển thị kết quả
lift_df
```
✓ 0.0s

	Antecedent	Consequent	Lift
0	Chocolate	Milk	1.236263
1	Milk	Chocolate	1.236263
2	Ice cream	Butter	1.200889
3	Kidney Beans	Butter	1.177626
4	Cheese	Kidney Beans	1.212143
5	Dill	Chocolate	1.186461

- Nhóm sẽ gộp bảng confidence và bảng lift để thuận tiện cho việc so sánh hơn:

```python
merge_df = pd.merge(confidence_df, lift_df, how='inner')

merge_df
```
✓ 0.0s

	Antecedent	Consequent	Support	Confidence	Lift
0	Chocolate	Milk	0.211211	0.501188	1.236263
1	Milk	Chocolate	0.211211	0.520988	1.236263
2	Ice cream	Butter	0.207207	0.504878	1.200889
3	Kidney Beans	Butter	0.202202	0.495098	1.177626
4	Cheese	Kidney Beans	0.200200	0.495050	1.212143
5	Dill	Chocolate	0.199199	0.500000	1.186461

- Nhận xét:

- Confidence: Đa số các luật kết hợp đều có mức độ confidence xấp xỉ 0.5. Trong đó, quy tắc số 1 (Milk → Chocolate) có confidence cao nhất : 0.520988, nghĩa là khoảng 52% khả năng một giao dịch bao gồm sữa cũng sẽ bao gồm chocolate.
- Lift cao hơn 1: Mọi quy tắc trong bảng đều có giá trị lift lớn hơn 1, điều này cho thấy có mối quan hệ mạnh mẽ giữa các sản phẩm so với việc chúng xuất hiện một cách ngẫu nhiên. Nói cách khác, đây là các luật kết hợp mạnh -> các sản phẩm này thường được mua cùng nhau.
- Mối quan hệ đối ứng: Có các quy tắc đối ứng như Chocolate → Milk (quy tắc 0) và Milk → Chocolate (quy tắc 1), cả hai đều có confidence và lift tương đối cao, cho thấy rằng mối quan hệ này là hai chiều và khá mạnh mẽ.

e. Cải tiến thuật toán

Nhóm tiến hành cải tiến thuật toán bằng cách sử dụng cây tiền tố prefix-tree (cấu trúc Trie Node):

```
class TrieNode:
    def __init__(self, item, count):
        self.item = item
        self.count = count
        self.children = {}

def insert_transaction(root, transaction):
    for itemset in combinations(transaction, 2):
        node = root
        for item in itemset:
            if item is not None:
                if item in node.children:
                    node = node.children[item]
                    node.count += 1
                else:
                    new_node = TrieNode(item, 1)
                    node.children[item] = new_node
                    node = new_node

def find_frequent_itemsets(node, min_support_count, current_itemset, frequent_itemsets, transactions):
    if node.item is not None and node.count >= min_support_count and len(current_itemset) > 1:
        frequent_itemsets[tuple(sorted([item for item in current_itemset if item is not None] + [node.item]))] = node.count / tra

    for _, child_node in node.children.items():
        find_frequent_itemsets(child_node, min_support_count, [node.item] + current_itemset, frequent_itemsets, transactions)

def eclat_trie(dataset, min_support):
    transactions = len(dataset)
    min_support_count = min_support * transactions

    root = TrieNode(None, 0)

    for transaction in dataset:
        insert_transaction(root, transaction)

    frequent_itemsets = {}
    find_frequent_itemsets(root, min_support_count, [], frequent_itemsets, transactions)

    return frequent_itemsets
```

Trong code thứ hai, chúng ta sử dụng Trie để lưu trữ thông tin về các mục itemset và số lần chúng xuất hiện. Thay vì phải duyệt qua toàn bộ dữ liệu mỗi lần để kiểm tra tần suất xuất hiện của mỗi itemset, chúng ta chỉ cần duyệt qua Trie một lần duy nhất để biết được thông tin này.

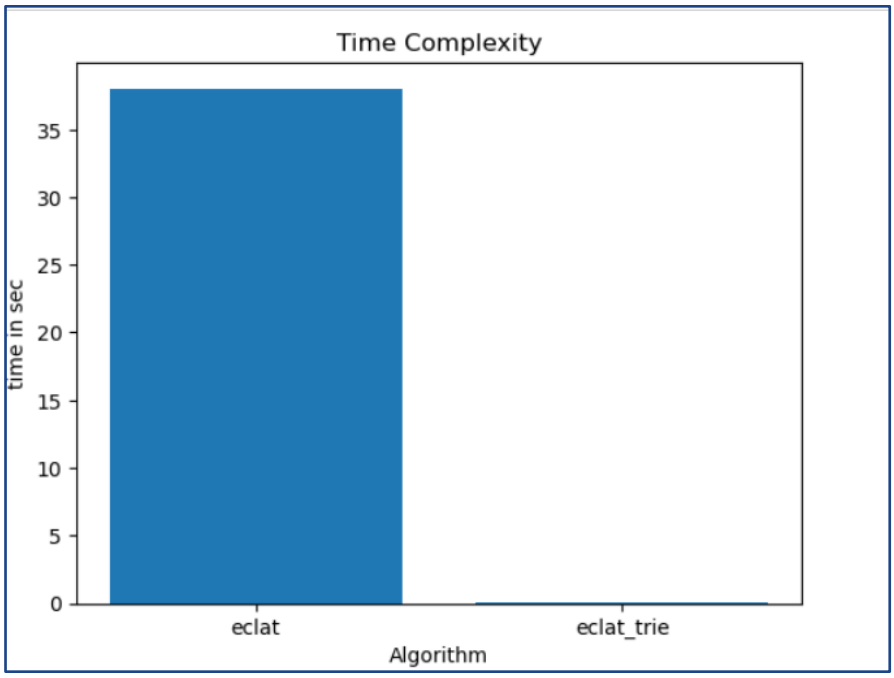
Thuật toán trên sẽ thực hiện các bước như sau:

- Khởi tạo Cây Trie:
    - Mỗi nút chứa thông tin về một mục (item), số lần xuất hiện (count), và một danh sách các nút con (children) tương ứng với các mục khác của itemset.
- Thêm các transaction vào Trie:
    - Duyệt qua từng transaction trong tập dữ liệu.
    - Đối với mỗi transaction , tạo các cặp mục khác nhau và thêm chúng vào cây Trie bằng cách duyệt qua các nút theo đường dẫn tương ứng và cập nhật thông tin.
- Tìm Kiếm Các Itemsets Phổ Biến (find_frequent_itemsets):
    - Gọi hàm 'find_frequent_itemsets' bắt đầu từ nút gốc của cây Trie.
    - Hàm này kiểm tra từng nút để xem những itemset trong tập kết hợp 2 phần tử có thõa mãn điều kiện chung hay không.
    - Nếu điều kiện được đáp ứng, hãy thêm 'itemset' vào một từ điển với số lần xuất hiện và tỷ lệ xuất hiện tương ứng.
    - Tiếp tục đệ quy vào các nút con để kiểm tra các phổ biến tập mục khác.

Kết quả cuối cùng là từ điển 'frequent_itemsets' chứa thông tin về các itemsets phổ biến và tỷ lệ xuất hiện của chúng trong tập dữ liệu.

Điểm mạnh của thuật toán Eclat nằm ở việc chia sẻ các đoạn đầu (prefix) giữa các itemset, giảm độ phức tạp của quá trình tìm kiếm. Điều này sẽ dễ dàng thấy được ở phần so sánh thời gian chạy bên dưới.
Mỗi bước đệ quy đi xuống cây Trie tương ứng với việc thêm một mục mới vào itemset, tận dụng cấu trúc Trie để kiểm tra tính phổ biến của itemset một cách hiệu quả.

Sau khi cải tiến nhóm tiến hành đo thời gian và có biểu đồ tương ứng như sau :



- Nhận xét: Sau khi cải tiến thuật toán, thời gian chạy đã tăng nhanh so với thuật toán eclat ban đầu, tuy nhiên chúng ta cần xem xét kết quả chạy để hiểu rõ hơn sự tăng lên này có ảnh hướng tới kết quả hay không.

Kết quả khi chạy thử với min sup = 0.11:

```
min_support = 0.11

result = eclat_trie(transactions, min_support)
df_test = pd.DataFrame.from_dict(result, orient='index', columns = ['Support'])
df_test
```

✓ 0.0s

	Support
(Bread, Corn)	0.174174
(Bread, Dill)	0.160160
(Bread, Ice cream)	0.181181
(Bread, Sugar)	0.179179
(Bread, Yogurt)	0.193193
...	...
(Kidney Beans, Unicorn)	0.184184
(Onion, Yogurt)	0.192192
(Onion, Sugar)	0.190190
(Onion, Unicorn)	0.175175
(Chocolate, Onion)	0.196196

120 rows × 1 columns

- Nhận xét : Nhìn vào kết quả, nhóm nhận ra rằng kết quả trả về của thuật toán eclat_trie bị thiếu hụt so với thuật toán eclat. Nguyên nhân có thể là do thuật toán chọn lọc ra những itemsets có kích thước nhỏ nhất (ít item) mà vẫn đạt đến ngưỡng minsup. Có thể nó giữ lại những luật có ý nghĩa lớn hơn.

CHƯƠNG 6. ĐÁNH GIÁ

1. *Đánh giá thuật toán ECLAT*

Trong quá trình khai thác tập mục thường xuyên, thách thức chính là tìm hiểu mối liên quan giữa các dữ liệu trong cơ sở dữ liệu giao dịch. Để giải quyết vấn đề này, nhiều thuật toán khác nhau đã được phát triển, trong đó có một thuật toán nổi bật được gọi là Eclat. Eclat không chỉ hữu ích trong ngành bán lẻ, mà còn có thể áp dụng cho nhiều mục đích khác nhau, đặc biệt là khi muốn tìm kiếm sự đồng đều giữa các biến trong một tập dữ liệu cụ thể. Với hiệu suất cao và khả năng khám phá mối quan hệ ẩn trong dữ liệu giao dịch, Eclat trở thành một công cụ mạnh mẽ cho việc phân tích dữ liệu và đưa ra quyết định. Bằng cách hiểu rõ nguyên tắc và cách sử dụng Eclat, ta có thể khám phá thông tin quan trọng từ dữ liệu và tự tin hơn trong quá trình đưa ra quyết định.

2. *Phân tích các sản phẩm theo luật kết hợp*

Rule	Support	Confident	Lift
Chocolate → Milk	0.211	0.502	1.24
Milk → Chocolate	0.211	0.521	1.24
Ice cream → Butter	0.207	0.505	1.201
Kidney Beans → Butter	0.202	0.495	1.178
Cheese → Kidney Beans	0.200	0.495	1.212
Dill → Chocolate	0.199	0.5	1.186

➔ Tất cả các luật trên đầu có lift > 1 cho thấy rằng là những luật trên xảy ra nhiều hơn dự kiến. Lift tăng cao có nghĩa là quy tắc này có ý nghĩa quan trọng, đồng thời có mối liên hệ chặt chẽ giữa antecedent và consequent.

CHƯƠNG 7. KẾT LUẬN

Đầu tiên nhóm em đã thành công trong việc tìm ra các luật kết hợp mạnh và xác định mối quan hệ quan trọng giữa các yếu tố bằng thuật toán eclat, đồng thời hỗ trợ trong việc đưa ra quyết định tối ưu hóa chiến lược như bán hàng hay tồn kho. Ngoài ra nhóm còn tự sự sáng tạo trong việc chạy thuật toán bằng hàm, thay vì chỉ sử dụng thư viện, đã giúp nhóm hiểu rõ hơn về thuật toán Eclat và tránh sự phụ thuộc quá mức vào các thư viện có sẵn khi cần phải thay đổi bộ dữ liệu hay cải tiến thuật toán sau này. Một ưu điểm phụ khác mà nhóm đã làm được trong đồ án là việc sử dụng cây trie để tối ưu thuật toán giảm thời gian chạy thuật toán. Việc này đặc biệt hữu ích khi xử lý các tập dữ liệu lớn.

Một khuyết điểm quan trọng của đồ án nhóm em là chưa đồng thời chọn được minSup và minConf tối ưu cho thuật toán. Điều này có thể ảnh hưởng đến hiệu suất và chất lượng của các luật kết hợp được tìm ra. Và đồng thời nhóm cũng chưa có những cải tiến đáng kể về thuật toán Eclat để tối ưu hiệu suất trong việc khai thác mẫu phổ biến trong bộ dữ liệu. Đây là những thiếu sót mà nhóm chưa khắc phục được.

Tổng kết lại, nhóm em qua đồ án này đã hiểu và sử dụng được thuật toán một cách cơ bản và ứng dụng được trong bộ dữ liệu thực tế. Tuy nhiên, còn nhiều tiềm năng để cải thiện và phát triển thuật toán theo các hướng khác nhau như các nhóm khác đã trình bày mà nhóm em chưa làm được. Dẫu vậy nhóm em nghĩ việc thiếu sót trên sẽ là kinh nghiệm để nhóm có những phân tích thuật toán sâu hơn sau này.

PHỤ LỤC

1. Mã nguồn

Quan sát bộ dữ liệu

Áp dụng và cải tiến thuật toán ECLAT

2. Bảng phân công

Thành viên	Phân công	Đánh giá
Nguyễn Trịnh Hiếu Kiên	Tìm hiểu thuật toán, Tiền xử lý dữ liệu, Tổng quan dữ liệu, Xây dựng thuật toán bằng hàm, Vẽ Network Graph,Tối ưu hóa thuật toán.	100%
Đặng Châu Kỳ	Tìm hiểu thuật toán, Tiền xử lý dữ liệu, Tổng quan dữ liệu, Xây dựng thuật toán bằng hàm, Vẽ Network Graph, Tối ưu hóa thuật toán.	100%
Nguyễn Hoàng Hà My	Tìm hiểu thuật toán, Trình bày khái niệm liên quan, Ví dụ minh họa cho thuật toán,Viết kết luận, Tổng hợp nội dung báo cáo.	100%
Tất Diệu Ngân	Tìm hiểu thuật toán, Tổng quan đề tài, Tổng hợp nội dung báo cáo, Viết kết luận.	80%
Phan Đình Nhân	Tìm hiểu thuật toán, Ví dụ minh họa cho thuật toán, Minh họa bài toán qua các bước, Tổng hợp nội dung báo cáo.	100%

3. Tài liệu tham khảo

[1] Bài giảng Khai phá Dữ liệu, TS. Nguyễn An Tế, Khoa Công nghệ Thông tin Kinh Doanh, Trường Công nghệ và Thiết kế - Đại học UEH, 2023

[2] (N.d.). Algorithmic Features of Eclat. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e5fa87f03af7b4abbbc6acb2085fd8d41ad01db3

[3] (N.d.). New Algorithms for Fast Discovery of Association Rules. Retrieved from https://cdn.aaai.org/KDD/1997/KDD97-060.pdf

[4] (N.d.).Comparing Dataset Characteristics that Favor the Apriori, Eclat, or FP-Growth Frequent Itemset Mining Algorithms. Retrieved from https://arxiv.org/pdf/1701.09042.pdf

[5] (N.d.).The Eclat Algorithm. Retrieved from https://www.philippe-fournier-viger.com/COURSES/Pattern_mining/Eclat.pdf?fbclid=IwAR0RCv07FGBLQNedf1w5FCq9jMOdoisej6pGpXn3UdbSL-6UIqX9XLwe5TI

[6] Eclat algorithm in association rule mining. (2014). Retrieved from https://fr.slideshare.net/deepa15/eclat-37310304

[7] dinhhoainam23894. (n.d.). dinhhoainam23894/Trie. Retrieved from https://github.com/dinhhoainam23894/Trie

[8] “Data Mining and Machine Learning: Fundamental Concepts and Algorithms Second Edition”. (2020). Mohammed J. Zaki and Wagner Meira, Jr. Cambridge University Press.