

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH (UEH)
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ



ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

NGHIÊN CỨU THUẬT TOÁN KHAI PHÁ TẬP PHỔ BIẾN ECLAT

Học phần: Khai Phá Dữ Liệu

Nhóm Sinh Viên:

1. Trần Phạm Hải Nam
2. Lý Minh Nguyên
3. Võ Minh Nguyên
4. Nguyễn Thị Ngọc Nhi
5. Lê Thành Phát

Chuyên Ngành: Khoa Học Dữ Liệu

Khóa: K47

Giảng Viên Hướng Dẫn: TS. Nguyễn An Tế

Thành phố Hồ Chí Minh, tháng 12 năm 2023

LỜI MỞ ĐẦU

Trong lĩnh vực Công nghệ thông tin, khai phá dữ liệu đóng vai trò quan trọng và thu hút sự chú ý của cộng đồng khoa học quốc tế. Một phần quan trọng của khai phá dữ liệu là khai phá tập phổ biến, một kỹ thuật tìm kiếm các tập hạng mục (item) thường xuyên xuất hiện trong cơ sở dữ liệu lớn.

Kỹ thuật này được phát triển từ năm 1993 với mục đích khai thác các luật kết hợp trong cơ sở dữ liệu giao dịch của các siêu thị. Sau đó, nó đã được mở rộng để phát hiện các mối quan hệ khác như phụ thuộc hàm, luật kết hợp đa mức, và nhiều hơn nữa.

Tuy nhiên, khai phá tập phổ biến gặp phải nhiều thách thức khi cơ sở dữ liệu có quy mô lớn và độ phức tạp cao. Mặc dù nhiều nghiên cứu đã được tiến hành để giải quyết vấn đề này, nhưng vẫn còn nhiều vấn đề chưa được khám phá.

Một trong những thuật toán khai phá tập phổ biến nổi bật là ECLAT, được đề xuất bởi Zaki vào năm 2000. ECLAT là một thuật toán tìm kiếm theo chiều sâu, tương tự như thuật toán FP-Growth. ECLAT có ưu điểm là chỉ cần quét cơ sở dữ liệu một lần, và tiết kiệm bộ nhớ hơn bằng cách sinh ra các tập mục ứng viên thông qua phép giao của các tập mục hiện có.

Tuy nhiên, ECLAT cũng có nhược điểm là không hiệu quả với cơ sở dữ liệu có nhiều hạng mục trong mỗi giao dịch. Để hiểu sâu hơn về lĩnh vực khai phá tập phổ biến và thuật toán ECLAT, nhóm chúng tôi đã chọn đề tài “**Nghiên cứu thuật toán khai phá tập phổ biến ECLAT**”.

Từ khóa: Khai phá dữ liệu, Khai phá tập phổ biến, Thuật toán ECLAT.

MỤC LỤC

LỜI MỞ ĐẦU.....	2
CHƯƠNG I. TỔNG QUAN.....	5
1. Sơ lược đề tài.....	5
2. Mục tiêu nghiên cứu.....	5
3. Phương pháp nghiên cứu.....	6
4. Tài nguyên sử dụng.....	6
CHƯƠNG II. TỔNG HỢP KHÁI NIỆM LIÊN QUAN.....	7
1. Một số khái niệm cơ bản.....	7
2. Bài toán khai thác tập phổ biến.....	8
3. Các phương pháp khai thác tập phổ biến.....	9
4. Cấu trúc IT - Tree.....	10
5. Thuật toán ECLAT.....	11
CHƯƠNG III. XÂY DỰNG BÀI TOÁN ĐƠN GIẢN NHẤT.....	13
1. Chuẩn bị dữ liệu.....	13
2. Biểu diễn lại CSDL theo chiều dọc.....	13
3. Loại bỏ các mục hiếm xuất hiện.....	14
4. Kết hợp các mục để tạo Equivalence Class mới.....	15
5. Xử lý đệ quy với mỗi Equivalence Class.....	16
6. Biểu diễn lại bằng IT - Tree ($\text{minSup} = 2$).....	16
7. Kết quả cuối cùng.....	18
CHƯƠNG IV. TỔNG QUAN & TIỀN XỬ LÝ BỘ DỮ LIỆU.....	19
1. Mô tả thuộc tính.....	19
2. Tiền xử lý.....	19
3. Thăm dò dữ liệu (EDA).....	21
4. Chính dạng dữ liệu.....	27
CHƯƠNG V. XÂY DỰNG THUẬT TOÁN ECLAT.....	31
1. Các bước thực hiện.....	31
2. Tìm luật kết hợp mạnh.....	35
CHƯƠNG VII. CẢI THIẾN & ĐÁNH GIÁ.....	41
1. Gom nhóm sản phẩm (Tối ưu độ lớn của dữ liệu).....	41
2. Thay đổi thứ tự các mục (Cải thiện thời gian xử lý).....	48
3. Cải thiện bộ nhớ bằng biến đại diện.....	50
4. So sánh tốc độ của ba thuật toán khai phá mẫu phổ biến.....	56

a. Apriori.....	56
b. ECLAT.....	58
c. FP-Growth.....	60
d. Kết luận.....	60
CHƯƠNG IX. KẾT LUẬN.....	62
PHỤ LỤC.....	64
1. Mã nguồn.....	64
2. Bảng phân công.....	64
3. Tài liệu tham khảo.....	64

CHƯƠNG I. TỔNG QUAN

1. Sơ lược đề tài

Trong thời đại công nghệ thông tin ngày càng tiên tiến, việc khai thác dữ liệu để tìm kiếm thông tin hữu ích đã trở thành một yếu tố quan trọng không thể thiếu. Phân tích mục mua hàng - một phương pháp phân tích các danh mục sản phẩm được mua cùng nhau trong cùng một giao dịch, đã trở thành một công cụ đắc lực để tăng doanh số bán hàng, đặc biệt là trong ngành bán lẻ. Đây là một phương pháp khoa học và hiệu quả để nắm bắt xu hướng tiêu dùng của khách hàng, từ đó giúp cửa hàng có thể tối ưu hóa việc quản lý và tăng cường doanh số bán hàng.

Thuật toán ECLAT, một thuật toán khai thác tập hợp phổ biến hiệu quả, đã được sử dụng rộng rãi trong việc phân tích mục mua hàng. Tuy nhiên, việc áp dụng thuật toán này vào thực tế, đặc biệt là trong ngữ cảnh của cửa hàng tạp hóa, vẫn còn nhiều thách thức. Điều này đòi hỏi một nghiên cứu sâu rộng và cụ thể để tìm hiểu về những thách thức này và đề xuất các giải pháp thích hợp.

Bài nghiên cứu này tập trung vào việc khám phá và đánh giá hiệu quả của việc áp dụng thuật toán ECLAT trong việc phân tích vật phẩm tại cửa hàng tạp hóa. Nhóm nghiên cứu hy vọng rằng, thông qua dự án này, nhóm sẽ cung cấp một cái nhìn sâu sắc về cách thuật toán ECLAT có thể được sử dụng để tối ưu hóa doanh số bán hàng và cung cấp trải nghiệm mua sắm tốt hơn cho khách hàng. Nhóm tin rằng kết quả của bài nghiên cứu này sẽ mở ra những cơ hội mới cho việc tối ưu hóa hoạt động kinh doanh của cửa hàng tạp hóa và cải thiện chất lượng dịch vụ cho khách hàng.

2. Mục tiêu nghiên cứu

Ở bài nghiên cứu này, nhóm sẽ tập trung vào việc khám phá và đánh giá hiệu quả của việc áp dụng thuật toán ECLAT trong việc phân tích vật phẩm tại cửa hàng tạp hóa. Để đạt được mục tiêu này, nhóm đã xác định rõ các mục tiêu cụ thể sau:

Nghiên cứu sâu về thuật toán ECLAT: Nhóm sẽ tìm hiểu kỹ lưỡng về cơ chế hoạt động của thuật toán ECLAT, cũng như cách giải thuật được sử dụng để khai thác tập hợp phổ biến từ dữ liệu.

Áp dụng thuật toán ECLAT vào dữ liệu thực tế: Từ bộ dữ liệu về cửa hàng tạp hóa được cung cấp trước đó, nhóm sẽ thực hiện áp dụng thuật toán ECLAT để phân tích các mục mua hàng. Quá trình này sẽ được thực hiện theo một phương pháp được nghiên cứu cụ thể.

Đánh giá hiệu quả của việc áp dụng thuật toán ECLAT: Kế đến, nhóm sẽ đánh giá hiệu quả của việc sử dụng thuật toán ECLAT thông qua việc so sánh kết quả phân tích

với các phương pháp phân tích khác. Đánh giá này sẽ được thực hiện dựa trên các tiêu chí thống kê cụ thể.

Đề xuất cách cải tiến và tối ưu hóa thuật toán ECLAT: Dựa trên kết quả nghiên cứu, nhóm sẽ đề xuất các cách cải tiến và tối ưu hóa thuật toán ECLAT để tăng cường hiệu quả trong việc phân tích vật phẩm tại cửa hàng tạp hóa.

3. Phương pháp nghiên cứu

Nhóm sử dụng nhiều phương pháp khác nhau linh hoạt theo nội dung nghiên cứu.

- **Chương II:** Tập trung vào việc xây dựng các khái niệm liên quan về thuật toán tìm mẫu phổ biến thông qua luật kết hợp ECLAT. Chương này cũng đề cập tới bài toán khai thác mẫu phổ biến, phương pháp khai thác mẫu biến bằng thuật toán ECLAT sử dụng IT-Tree.
- **Chương III:** Tạo ra một ví dụ minh họa cụ thể để xây dựng các bước khai thác mẫu phổ biến dựa trên thuật toán ECLAT.
- **Chương IV & V:** Thực hiện áp dụng bài toán vào một bộ dữ liệu thực tế với mục tiêu tìm ra luật kết hợp giữa các mặt hàng sản phẩm trong một cửa hàng. Từ đó, tìm ra xu hướng mua sắm hàng hóa của khách hàng. Liệu khi một khách hàng mua loại sản phẩm này thì sẽ có bao nhiêu phần trăm họ sẽ mua loại sản phẩm kia.
- **Chương VII:** Nghiên cứu và đưa ra các phương pháp để cải thiện thuật toán như gom nhóm các sản phẩm có cùng chức năng nhằm tối ưu độ lớn của dữ liệu, thay đổi thứ tự toàn cục nhằm cải thiện thời gian xử lý, cải thiện bộ nhớ bằng biến đại diện sử dụng phương pháp Diffset. Ngoài ra, ở chương này nhóm cũng tiến hành đánh giá so sánh giữa thuật toán ECLAT, Apriori, FP-Growth về độ phức tạp thời gian.

4. Tài nguyên sử dụng

- Ngôn ngữ lập trình: Python
- Bộ dữ liệu (Nguồn - TS. Nguyễn An Tế): [Groceries 1.csv](#)

CHƯƠNG II. TỔNG HỢP KHÁI NIỆM LIÊN QUAN

1. Một số khái niệm cơ bản

Cho cơ sở dữ liệu (CSDL) với tập mục $I = \{A, B, C, D, E\}$, các giao dịch T được mô tả trong bảng sau:

Giao dịch	Mục
T_1	A, B, D, E
T_2	B, C, E
T_3	A, B, D, E
T_4	A, B, C, E
T_5	A, B, C, D, E
T_6	B, C, D

- **Database:** $D = \{T_i\}$ là một bộ gồm hai thành phần: $\langle T, I \rangle$ trong đó:
 - **Transaction:** $T = \{T_1, T_2, \dots, T_m\}$ là tập gồm m giao dịch của CSDL
 - Quy ước: Items trong T được sắp xếp tăng dần
 - **Item:** $I = \{I_1, I_2, \dots, I_n\}$ là tập gồm n mục trong CSDL
 - **Itemset:** $X \subseteq I$ là các tập mục con của I
 - **K - Itemset:** $X_k = \{I_1, I_2, \dots, I_k\}$ với $k \leq n$
 - **TIDset:** $TIDset(X) = \{T_m \mid T_m \in T \wedge X \subseteq T_m\}$ là tập hợp các giao dịch chứa X

Ví dụ:

CSDL trên có sáu giao dịch là $T = \{T_1, T_2, T_3, T_4, T_5, T_6\}$, với các mục là $I = \{A, B, C, D, E\}$, và trong giao dịch T_1 có Itemset là $X_4 = \{A, B, D, E\}$.

- **Độ hỗ trợ (support)** - $support(X)$ là tần suất xuất hiện của tập mục đó trong CSDL, được sử dụng để đánh giá mức độ phổ biến của một mẫu trong tập dữ liệu và được xác định theo công thức như sau:

$$support(X) = \frac{freq(X)}{m} \times 100\%$$

Với m là số giao dịch của CSDL và $freq(X)$ là tần suất xuất hiện của X trong T .

Ví dụ: Dựa vào CSDL trên, ta có $support(AB) = \frac{4}{6} \times 100\% = 66.67\%$ (tập mục AB xuất hiện trong bốn giao dịch T_1, T_3, T_4, T_5).

- **Độ tin cậy (confidence)** - $conf(X \rightarrow Y)$ là tỷ lệ giữa tần suất xuất hiện của tập mục X và Y và tần suất xuất hiện của tập mục X, được sử dụng để đánh giá mức độ chắc chắn của một quy tắc phân lớp, và được xác định theo công thức như sau:

$$conf(X \rightarrow Y) = \frac{freq(X \cup Y)}{freq(X)} \times 100\%$$

- **Tập phổ biến (frequent pattern)** là tập hợp các mục có độ hỗ trợ (*support*) lớn hơn hay bằng ngưỡng phổ biến *minSup* (để khai thác tập phổ biến, người ta đưa vào một ngưỡng, gọi là ngưỡng phổ biến - *minSup*). Như vậy tập các tập mục phổ biến - FI được xác định như sau:

$$FI = \{X \mid X \subseteq I \text{ và } support(X) \geq minSup\}$$

Một số tính chất:

- Mọi tập con của tập phổ biến đều phổ biến, nghĩa là $\forall X \subseteq Y$, nếu $support(Y) \geq minSup$ thì $support(X) \geq minSup$
- Mọi tập cha của tập không phổ biến đều không phổ biến, nghĩa là $\forall Y \supseteq X$, nếu $support(X) < minSup$ thì $support(Y) < minSup$

2. Bài toán khai thác tập phổ biến

Bài toán khai thác tập phổ biến trên CSDL cho trước là bài toán tìm tất cả các tập mục của CSDL có tần số xuất hiện trong các giao dịch thỏa mãn ngưỡng *minSup* do người dùng xác định trước. Tập các tập mục được khai thác theo ngưỡng *minSup* được gọi là *tập phổ biến*.

Tìm tập phổ biến trong khai phá dữ liệu là quy trình quan trọng trong việc khám phá thông tin ẩn, tìm ra các mẫu tương quan và các quy tắc của tập dữ liệu. Mục tiêu của phương pháp này là tìm ra các tập hợp phổ biến mà chúng xuất hiện cùng nhau với một tần suất đáng kể.

Các thuật toán được sử dụng rộng rãi trong phương pháp chọn tập phổ biến có thể được kể đến như là *Apriori*, *FP-Growth* và *ECLAT*.

Ví dụ. Xét CSDL với tập mục $I = \{A, B, C, D, E\}$, các giao dịch T được mô tả trong bảng sau:

Giao dịch	Mục
T ₁	A, B, D, E
T ₂	B, C, E
T ₃	A, B, D, E
T ₄	A, B, C, E

T_5	A, B, C, D, E
T_6	B, C, D

- Cho $\text{minSup} = 0.5$

Các tập phổ biến của CSDL là $\{A, B, C, D, E, AB, AD, AE, BC, BD, BE, DE, ABD, ABE, BDE, ABDE\}$.

- Xét tập mục có 1-itemset:

$$\text{support}(A) = \frac{4}{6} \times 100\% = 66.67\%; \text{support}(B) = \frac{6}{6} \times 100\% = 100\%;$$

$$\text{support}(C) = \frac{4}{6} \times 100\% = 66.67\%; \text{support}(D) = \frac{4}{6} \times 100\% = 66.67\%;$$

$$\text{support}(E) = \frac{5}{6} \times 100\% = 83.33\%$$

- Xét tập mục có 2-itemset:

$$\text{support}(AB) = \frac{4}{6} \times 100\% = 66.67\%; \text{support}(AD) = \frac{3}{6} \times 100\% = 50\%;$$

$$\text{support}(AE) = \frac{4}{6} \times 100\% = 66.67\%; \text{support}(BC) = \frac{4}{6} \times 100\% = 66.67\%;$$

$$\text{support}(BD) = \frac{5}{6} \times 100\% = 83.33\%; \text{support}(BE) = \frac{5}{6} \times 100\% = 83.33\%;$$

$$\text{support}(DE) = \frac{3}{6} \times 100\% = 50\%$$

- Xét tập mục có 3-itemset:

$$\text{support}(ABD) = \frac{3}{6} \times 100\% = 50\%; \text{support}(ABE) = \frac{4}{6} \times 100\% = 66.67\%;$$

$$\text{support}(BDE) = \frac{3}{6} \times 100\% = 50\%$$

- Xét tập mục có 4-itemset:

$$\text{support}(ABDE) = \frac{3}{6} \times 100\% = 50\%$$

3. Các phương pháp khai thác tập phổ biến

- Phương pháp khai thác tập phổ biến trên CSDL ngang

CSDL ngang (*Horizontal Database*) được hiểu là CSDL giao dịch được mô tả như bảng dữ liệu ví dụ minh họa trên. CSDL ngang có thể được mở rộng theo chiều ngang, tức là ta có thể thêm các bản ghi mới vào các hàng của bảng dữ liệu này thay vì mở rộng theo chiều rộng bằng cách thêm các cột mới.

Có hai phương pháp chính khai thác tập phổ biến trên CSDL ngang, đó là thuật toán Apriori và FP-Growth.

- **Phương pháp khai thác tập phổ biến trên CSDL dọc dựa trên IT - Tree**

CSDL dọc (*Vertical Database*) có thể được mở rộng bằng cách thêm các cột mới vào bảng dữ liệu thay vì thêm các hàng mới. Trái ngược với CSDL ngang, CSDL dọc tập trung vào việc mở rộng dữ liệu theo chiều dọc bằng cách thêm các thuộc tính (cột) mới cho các bản ghi có sẵn thay vì thêm các bản ghi mới.

CSDL Ngang			CSDL Dọc	
Giao dịch	Mục		Mục	TIDset
T_1	A, B, D, E		A	$1, 3, 4, 5$
T_2	B, C, E		B	$1, 2, 3, 4, 5, 6$
T_3	A, B, D, E		C	$2, 4, 5, 6$
T_4	A, B, C, E	➔	D	$1, 3, 5, 6$
T_5	A, B, C, D, E		E	$1, 2, 3, 4, 5$
T_6	B, C, D			

Thuật toán ECLAT (*Equivalence Class Clustering and Bottom-Up Lattice Traversal*) được đề xuất bởi Zaki sử dụng cấu trúc IT - Tree (*TIDset Itemset-tree*) để lưu TIDset của các tập mục trên mỗi nút và đưa ra khái niệm lớp tương đương để kết nối các tập mục trong cùng một lớp tương đương để tạo ra tập mục mới. Thuật toán ECLAT với chỉ một lần quét CSDL là một tiếp cận hiện đại, tiết kiệm thời gian xử lý và có thể áp dụng khai thác FI trên nhiều loại CSDL một cách hiệu quả.

Tiếp cận ECLAT sử dụng tính chất Apriori để cắt nhánh các tập mục không thỏa ngưỡng phổ biến. Do vậy các tập mục trên IT - Tree là các FI thỏa mãn ngưỡng minSup.

Nhờ vậy nếu tập mục X không thỏa mãn ngưỡng minSup thì các tập mục là phụ của X cũng không thỏa ngưỡng minSup, do vậy ta không cần xét nhánh do tập mục X tạo ra. Áp dụng tính chất bao đóng giảm trên IT - Tree sẽ cắt nhánh được tất cả các nhánh chắc chắn không chứa FI và các tập mục trên IT - Tree chính là FI cần khai thác.

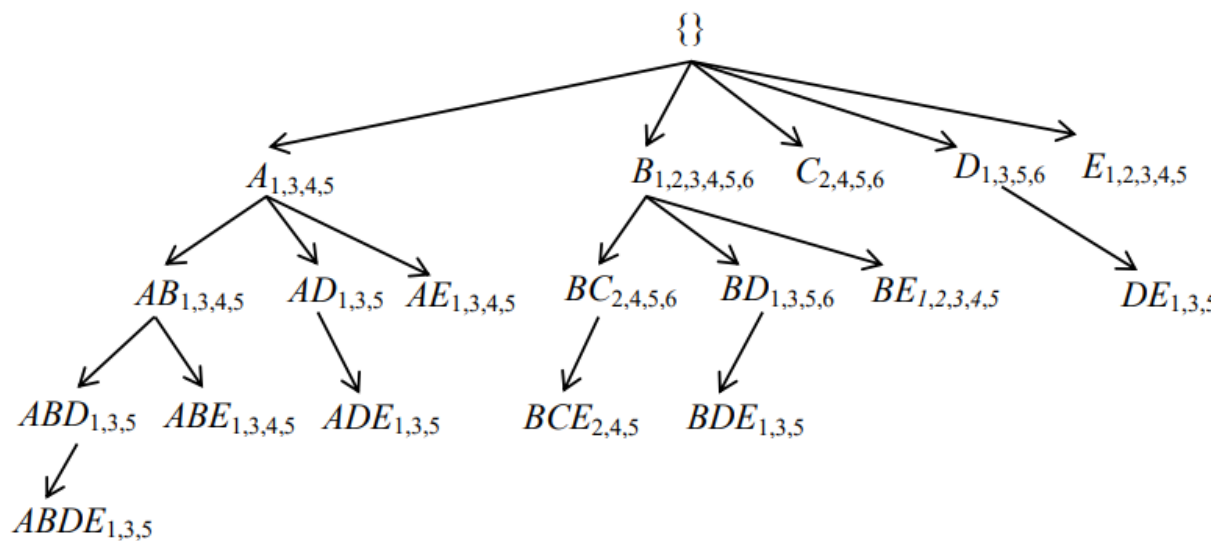
4. Cấu trúc IT - Tree

IT - Tree có cấu trúc gồm nhiều mức, mỗi mức gồm nhiều *lớp tương đương**. Mỗi lớp tương đương gồm các nút có cùng nút cha ở mức trên. Mỗi nút gồm hai thành phần: Tập mục X, TIDset của X.

Mỗi lớp tương đương được tạo ra từ một mục cha ở mức trên kết hợp lần lượt các nút phía sau nó trong cùng một lớp tương đương. Do vậy các nút trong cùng một lớp tương đương có cùng số lượng phần tử chỉ khác nhau 19 phần tử cuối cùng. Một nút

mới được chèn vào IT - Tree chỉ khi nó có độ hỗ trợ (*support*) của nó thỏa mãn ngưỡng phổ biến (*minSup*), do đó các nút trên IT - Tree sau khi xây dựng xong chính là các tập phổ biến cần khai thác.

Ví dụ:



5. Thuật toán ECLAT

- Sơ lược về các bước xây dựng giải thuật (Tham khảo - PGS.TS Võ Đình Bảy):

```

ECLAT ()
    [Ø] = {i ∈ I | σ(i) ≥ minSup}
    ENUMERATE_FREQUENT ([Ø])
ENUMERATE_FREQUENT ([P])
    for all li ∈ [P] do
        [Pi] = Ø
        for all lj ∈ [P] with j > i do
            X = li ∪ lj
            T = t(li) ∩ t(lj)
            if |T| ≥ minSup then
                [Pi] = [Pi] ∪ {X × T}
        ENUMERATE_FREQUENT ([Pi])
    
```

Trong đó $t(X) = \{y \in T \mid X \text{ xuất hiện trong giao dịch } y\}$ được gọi là Tidset của X.

PGS.TS. Võ Đình Bảy

Đầu vào: CSDL và một ngưỡng *minSup*

Đầu ra: Các tập phổ biến thỏa *minSup*

B1. Quét toàn bộ CSDL để xác định tập giao dịch (TIDset) của các mục. Chọn các mục có *support* thỏa ngưỡng *minSup* (1-itemset).

B2. Chèn 1-itemset vào mức 1 của IT - Tree.

B3. Mỗi nút ở mức $k - 1$ kết hợp với các nút có cùng nút cha với nó tạo ra các nút ở mức k nếu *support* của các nút này thỏa ngưỡng *minSup*.

B4. Lặp lại B3 cho đến khi không thể tạo thêm nút mới trên IT - Tree.

B5. Duyệt cây IT - Tree để lấy ra các tập phổ biến (tất cả các nút trên IT - Tree đều là tập phổ biến).

Nhận xét:

- Thuật toán dựa vào phần giao giữa các TIDset để tính nhanh độ hỗ trợ nên chỉ quét CSDL 1 lần.
- Có thể sử dụng Diffset để tính nhanh độ hỗ trợ nhằm làm giảm không gian lưu trữ TIDset.
- Do thuật toán không sinh ứng viên L nên hiệu quả khai thác thường cao hơn so với các họ thuật toán sinh ứng viên.
- Khi số tập phổ biến lớn, thời gian khai thác luật lớn. Do vậy, ta cần tìm phương pháp khai thác hiệu quả hơn.

Thuật toán ECLAT với cấu trúc IT - Tree là hướng tiếp cận tốt nhất được biết đến hiện nay với chỉ một lần quét CSDL. Tuy nhiên, phương pháp này có nhược điểm lớn là tốn bộ nhớ sử dụng để lưu TIDset của các tập mục, do mỗi giao dịch chứa tập mục cần một ô nhớ. Đồng thời bộ nhớ tạm cần thiết trong quá trình tính toán trung gian cũng rất lớn. Các hạn chế này làm cho thời gian tính toán của thuật toán ECLAT chưa được tối ưu.

Để rõ hơn về các bước xây dựng giải thuật, nhóm sẽ thực hiện mô tả từng bước thông qua ví dụ minh họa cụ thể ở chương kế tiếp.

CHƯƠNG III. XÂY DỰNG BÀI TOÁN ĐƠN GIẢN NHẤT

1. Chuẩn bị dữ liệu

Đầu tiên, nhóm nghiên cứu sẽ tiến hành chuẩn bị dữ liệu gốc sao cho phù hợp với quá trình khai thác mẫu. Trong các bước tiếp theo, nhóm sẽ xây dựng giải thuật ECLAT dựa trên bảng dữ liệu minh họa sau đây:

Transaction	Items
T1	Bánh quy, Cam, Nước ngọt, Sữa
T2	Cam ,Sữa
T3	Bánh quy, Cam, Chanh
T4	Cam, Chanh, Sữa

2. Biểu diễn lại CSDL theo chiều dọc

Từ tập dữ liệu ở Bước 1, tiếp theo nhóm sẽ chuyển đổi cơ sở dữ liệu này sang định dạng theo chiều dọc, hay Vertical Format. Trong định dạng này, mỗi sản phẩm sẽ được liên kết với một danh sách, trong đó nó sẽ ghi chép lại các chỉ mục* (số thứ tự) của giao dịch mà sản phẩm đó đã xuất hiện.

** Ứng với mỗi dòng được biểu diễn lại theo chiều dọc được gọi là TID-list (Transaction ID List). Khi đó, mỗi TID là một số (ký tự) duy nhất được gán cho mỗi Transactions trong tập dữ liệu. TID-list chứa các TID của các Transactions mà sản phẩm đó xuất hiện trong CSDL ban đầu.*

Việc biểu diễn lại CSDL theo chiều dọc giúp xử lý và phân tích dữ liệu trở nên thuận tiện hơn, bởi ta có thể nhanh chóng truy xuất thông tin về sự xuất hiện của mỗi sản phẩm trong từng giao dịch.

Ví dụ:

Đọc cơ sở dữ liệu (Database) ban đầu và biểu diễn lại CSDL theo chiều dọc (Vertical Format):

Transaction	Items
T1	Bánh quy, Cam, Nước ngọt, Sữa
T2	Cam ,Sữa

T3	Bánh quy, Cam, Chanh
T4	Cam, Chanh, Sữa



Item	Transactions
Bánh quy	T1, T3
Cam	T1, T2, T3, T4
Chanh	T3, T4
Nước ngọt	T1
Sữa	T1, T2, T4

3. Loại bỏ các mục hiếm xuất hiện

Tiếp tục từ Bước 2, nhóm tiến hành loại bỏ các sản phẩm (Items) hiếm xuất hiện trong tập dữ liệu để xác định các Equivalence Class ban đầu.

Để thực hiện loại bỏ, thuật toán ECLAT cần tính tần suất xuất hiện của mỗi sản phẩm và loại bỏ những sản phẩm có tần suất xuất hiện thấp hơn ngưỡng xuất hiện tối thiểu (minSup) được xác định trước đó.

Bằng cách này, nhóm có thể giảm số lượng các mục ít quan trọng ra khỏi tập dữ liệu, tập trung hơn vào những mục có tần suất xuất hiện cao hơn, giúp tăng hiệu suất và giảm độ phức tạp của thuật toán.

Ví dụ:

Loại bỏ các sản phẩm (Items) hiếm xuất hiện trong tập dữ liệu thu được ở Bước 2. Quá trình loại bỏ các mục hiếm xuất hiện có thể được biểu diễn bằng công thức sau:

$$F = \{\text{frequent itemsets}\} = \{I \mid \text{support}(I) \geq \text{min_support}\}$$

Trong đó:

- **F**: Tập hợp các tập con phổ biến (Frequent Itemsets).
- **I**: Một tập con trong F.

- **Support (I):** Độ hỗ trợ (Support) của tập con I (Tỷ lệ số lượng Trans chứa I trên tổng số Trans)
- **Min Support:** Ngưỡng tối thiểu của độ hỗ trợ, được xác định trước để quyết định mức xuất hiện tối thiểu của một mục để được coi là phổ biến.

Với ngưỡng $\text{minSup} = 2$

Item	Transactions
Bánh quy	T1, T3
Cam	T1, T2, T3, T4
Chanh	T3, T4
Nước ngọt	T1 (Loại bỏ khỏi tập dữ liệu < 2)
Sữa	T1, T2, T4

4. Kết hợp các mục để tạo Equivalence Class mới

Ở bước tiếp theo, nhóm sẽ kết hợp các mục trong các Equivalence Class đã xác định để tạo ra các Equivalence Class mới, mỗi Class có kích thước $K + 1$, với K là kích thước của Class gốc. Mục tiêu là tạo ra các tập hợp phổ biến lớn hơn. Quá trình này tiếp tục cho đến khi không thể kết hợp thêm được nữa.

Ví dụ:

Nhóm sẽ kết hợp các mục của Equivalence Class vừa xác định được ở Bước 3 để tạo ra các Equivalence Class mới có kích thước $K + 1$. Quá trình kết hợp trên có thể được biểu diễn dựa trên công thức sau:

$$EC(k+1) = \{I1 \cup I2 \mid I1 \in EC(k), I2 \in EC(k), |I1 \cup I2| = K + 1, I1 \neq I2\}$$

Trong đó:

- $EC(k+1)$ đại diện cho Equivalence Class mới có kích thước $K + 1$.
- $I1$ và $I2$ là các tập con trong Equivalence Class $EC(k)$.
- $|I1 \cup I2| = K + 1$ đảm bảo rằng tổng số phần tử trong $I1$ và $I2$ là $K + 1$.
- $I1 \neq I2$ đảm bảo rằng $I1$ và $I2$ là các tập con khác nhau.

Các Equivalence Class mới có kích thước $K + 1$ mới được tạo thành:

Bánh quy, Cam	T1, T3
Bánh quy, Chanh	T3 (Loại)
Bánh quy, Sữa	T1 (Loại)

Cam, Chanh	T3, T4
Cam, Sữa	T1, T2, T4

Chanh, Sữa	T4 (Loại)
-----------------------	----------------------

Loại bỏ các Itemsets hiếm xuất hiện trong tập, ta xác định được những Itemset phổ biến như sau: {Bánh quy, Cam}, {Cam, Chanh}, {Cam, Sữa}

5. Xử lý đệ quy với mỗi Equivalence Class

Ở Bước 5, thuật toán ECLAT sẽ thực hiện xử lý đệ quy trên từng Equivalence Class. Đầu tiên, ECLAT thực hiện các bước từ Bước 3 đến Bước 4 cho mỗi Equivalence Class, bao gồm việc kết hợp các mục để tạo ra các Equivalence Class mới có kích thước tăng dần, nhằm tìm kiếm các tập hợp phổ biến. Quá trình đệ quy này tiếp tục cho đến khi không còn Equivalence Class hoặc tập hợp phổ biến mới nào được tìm thấy.

Ví dụ:

Xử lý đệ quy với mỗi Equivalence Class, tiếp tục từ ví dụ trên:

Bánh quy, Cam, Chanh	T3 (Loại)
Bánh quy, Cam, Sữa	T1 (Loại)

Cam, Chanh, Sữa	T4 (Loại)
----------------------------	----------------------

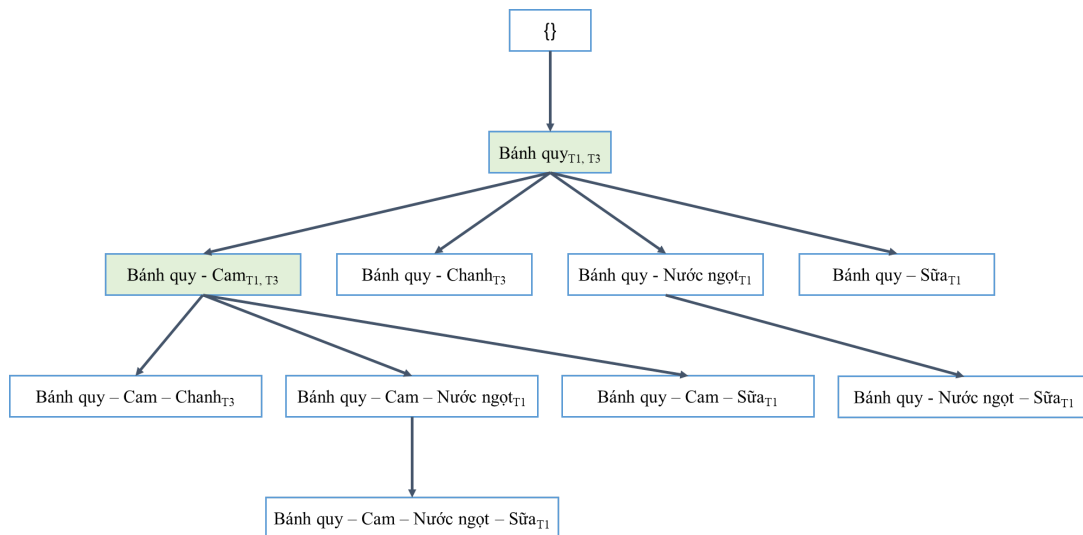
Không có bất kỳ Itemsets nào có 3 phần tử được tạo thành → **Dừng bước 5**. Từ đó, Itemsets phổ biến thu được vẫn giống ở Bước 4, như sau: {Bánh quy, Cam}, {Cam, Chanh}, {Cam, Sữa}.

6. Biểu diễn lại bằng IT - Tree (minSup = 2)

Để thực hiện, nhóm sẽ duyệt IT - Tree để lấy ra các tập phổ biến (tất cả các nút trên IT - Tree đều là tập phổ biến).

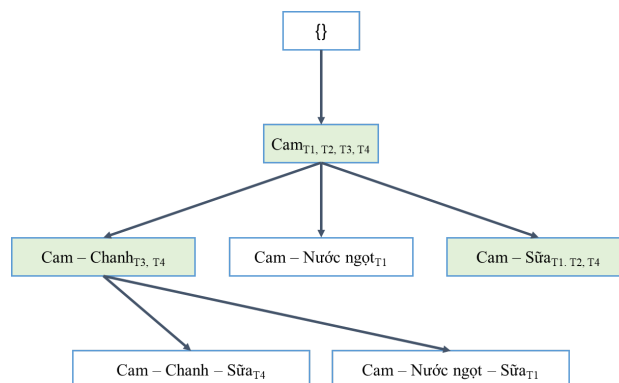
Đầu tiên, tập 1-itemset được đưa vào mức đầu tiên của IT - Tree. Sau đó, các cặp nút trong cùng một lớp tương đương được kết nối với nhau để tạo ra các tập mục mới ở mức tiếp theo nếu tập mục mới thỏa mãn ngưỡng minSup như trên (ứng với những ô màu xanh lục).

- Với EC là *Bánh quy*:



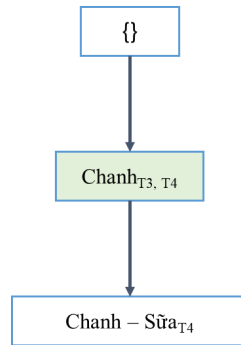
→ Itemsets hiếm xuất hiện trong tập: {Bánh quy}, {Bánh quy, Cam}

- Với EC là *Cam*:



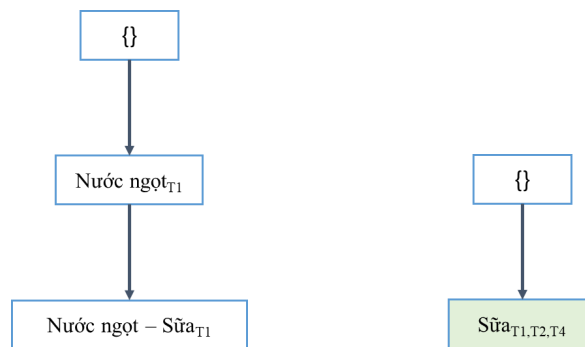
→ Itemsets hiếm xuất hiện trong tập: {Cam}, {Cam, Chanh}, {Cam, Sữa}

- Với EC là *Chanh*:



→ Itemsets hiếm xuất hiện trong tập: {Chanh}

- Với EC là Nước ngọt & Sữa



→ Itemsets hiếm xuất hiện trong tập: {Sữa}

7. Kết quả cuối cùng

Cuối cùng, ECLAT sẽ trả về danh sách các mẫu phổ biến tìm thấy trong cơ sở dữ liệu. Mỗi mẫu là một tập hợp các vật phẩm xuất hiện cùng nhau trong một giao dịch.

Danh sách này cho thấy mối quan hệ giữa các mục, giúp phân tích và hiểu rõ hơn cấu trúc dữ liệu. Từ kết quả này, người dùng có thể nhận biết các mục thường xuất hiện cùng nhau, tập hợp mục phổ biến, và sự tương quan giữa các mục.

Ví dụ: Mẫu phổ biến thu được từ giải thuật ECLAT:

{Bánh quy}	support = 2
{Cam}	support = 4
{Chanh}	support = 2
{Sữa}	support = 3
{Bánh quy, Cam}	support = 2
{Cam, Chanh}	support = 2
{Cam, Sữa}	support = 3

Từ các bước khái quát để xây dựng thuật toán ECLAT vừa cung cấp, nhóm đã có thể xây dựng được thuật toán ECLAT thông qua Python, cũng như ứng dụng để xử lý trên tập tin 'Groceries 1.csv' để tìm ra các kết hợp sản phẩm tiềm năng.

CHƯƠNG IV. TỔNG QUAN & TIỀN XỬ LÝ BỘ DỮ LIỆU

- **Link nguồn:** [1-Preprocessing.ipynb - Colaboratory](#)

1. Mô tả thuộc tính

Bộ dữ liệu **Groceries_1.csv** ghi nhận thông tin về các giao dịch được thực hiện tại một cửa hàng tiện lợi, nhằm mục đích tìm ra các luật kết hợp cho những mặt hàng từ đó có những giải pháp hợp lý nhằm tăng lợi nhuận bán hàng.

Bộ dữ liệu có **38765** quan sát với **3** thuộc tính bao gồm:

Thuộc tính	Mô tả	Kiểu dữ liệu
Member_number	Mã số định danh của từng khách hàng	int64
Date	Ngày thực hiện giao dịch	object
itemDescription	Mặt hàng có trong giao dịch	object

2. Tiền xử lý

Đầu tiên, sẽ tiến hành kiểm tra các giá trị không trùng lặp của từng thuộc tính:

```
properties_chk(df)
```

	Số quan sát	Số giá trị unique	Kiểu dữ liệu
Member_number	38765	3898	int64
Date	38765	728	object
itemDescription	38765	167	object

Kết quả cho thấy có tổng cộng **728** ngày được ghi nhận dữ liệu, trong khoảng thời gian này có đến **3898** khách hàng khác nhau đã thực hiện giao dịch tại cửa hàng với tất cả **167** các loại hàng hóa khác nhau đã được thực hiện giao dịch.

Bước tiếp theo, nhóm nhận thấy kiểu dữ liệu của các thuộc tính vẫn chưa có định dạng chuẩn để thực hiện phân tích mẫu phổ biến. Do đó, nhóm sẽ chỉnh dạng dữ liệu cho từng thuộc tính.

Đối với thuộc tính **Member_number**, nhóm sẽ thực hiện chỉnh dạng từ kiểu số nguyên sang object.

```
df['Member_number'] = df['Member_number'].astype('object')
```

Đối với thuộc tính **Date**, nhóm sẽ chỉnh từ kiểu object về đúng dạng dữ liệu thời gian và kiểm tra liệu có giá trị nào bất thường hay không. Ví dụ: ngày 30 vào tháng 2, ngày 31 vào tháng 4, ...

```
# Chuyển về cùng format
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df.tail()
```

	Member_number	Date	itemDescription
38760	4471	2014-08-10	sliced cheese
38761	2022	2014-02-23	candy
38762	1097	2014-04-16	cake bar
38763	1510	2014-03-12	fruit/vegetable juice
38764	1521	2014-12-26	cat food

```
df['Date'] = set_invalid_date(df['Date'])
invalid_date = df[df['Date'] == False]
print(invalid_date)
```

```
Empty DataFrame
Columns: [Member_number, Date, itemDescription]
Index: []
```

→ Kết quả kiểm tra không cho thấy có bất kỳ giá trị thời gian nào không hợp lệ.

Bước tiếp theo nhóm sẽ kiểm tra các dòng giá trị bị rỗng của bộ dữ liệu.

```
null_chk(df)
```

	Số quan sát	Số giá trị rỗng	Phần trăm rỗng
Member_number	38765	0	0.00%
Date	38765	0	0.00%
itemDescription	38765	0	0.00%

→ Kết quả cho thấy, cả 3 cột thuộc tính đều có đủ **38765** quan sát và không có bất kỳ dòng giá trị nào bị rỗng.

Bước cuối cùng trong phần tiền xử lý, nhóm sẽ kiểm tra các dòng trùng lặp trong bộ dữ liệu.

```
# Các dòng được nhận định là trùng  
duplicated_df = df[df.duplicated(subset=['Member_number', 'Date', 'itemDescription'])]  
duplicated_df
```

759 rows × 3 columns

→ Kết quả cho thấy có đến **759** dòng bị trùng lặp giá trị ở cả ba thuộc tính trong bộ dữ liệu.

Dòng dữ liệu gốc và dòng dữ liệu bị trùng:

	Member_number	Date	itemDescription
4004	2051	2015-11-09	frankfurter
4011	3055	2015-08-18	other vegetables
5015	2051	2015-11-09	frankfurter
5022	3055	2015-08-18	other vegetables
11480	1146	2014-05-23	yogurt
16061	3055	2015-08-18	other vegetables
27628	3834	2014-05-18	salty snack
38722	3834	2014-05-18	salty snack
38723	1146	2014-05-23	yogurt

3. Thăm dò dữ liệu (EDA)

Trong phần này nhóm sẽ tiến hành phân tích dữ liệu theo từng thuộc tính:

- **Thuộc tính itemDescription**

```
# Tổng số lượng loại item
item_num = df['itemDescription'].nunique()
print(f'Tập dữ liệu này có {item_num} unique items')
```

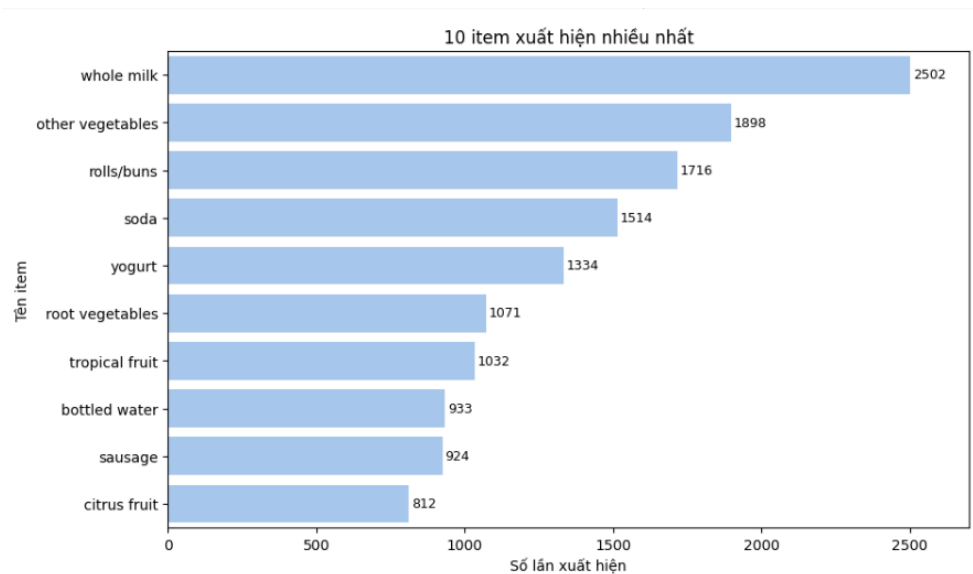
Tập dữ liệu này có 167 unique items

Như đã tìm thấy ở phần trước, thuộc tính này có tổng cộng **167** giá trị khác nhau đồng nghĩa với **167** mặt hàng khác nhau.

whole milk	2502
other vegetables	1898
rolls/buns	1716
soda	1514
yogurt	1334
root vegetables	1071
tropical fruit	1032
bottled water	933
sausage	924
citrus fruit	812

→ **Phân tích bảng thống kê:** Những mặt hàng có tần suất xuất hiện nhiều nhất chính là những tập phổ biến đơn giản nhất với một phần tử duy nhất. Kết quả từ bảng thống kê cho thấy, mặt hàng xuất hiện nhiều nhất trong các giao dịch tại cửa hàng là **whole milk** (các loại sữa) với số lần xuất hiện lên đến 2502 lần, nhiều hơn mặt hàng đứng thứ 2 tận 404 lần. Cả 10 item có số lần xuất hiện nhiều nhất đều là những mặt hàng thực phẩm.

- **Phân tích biểu đồ:**

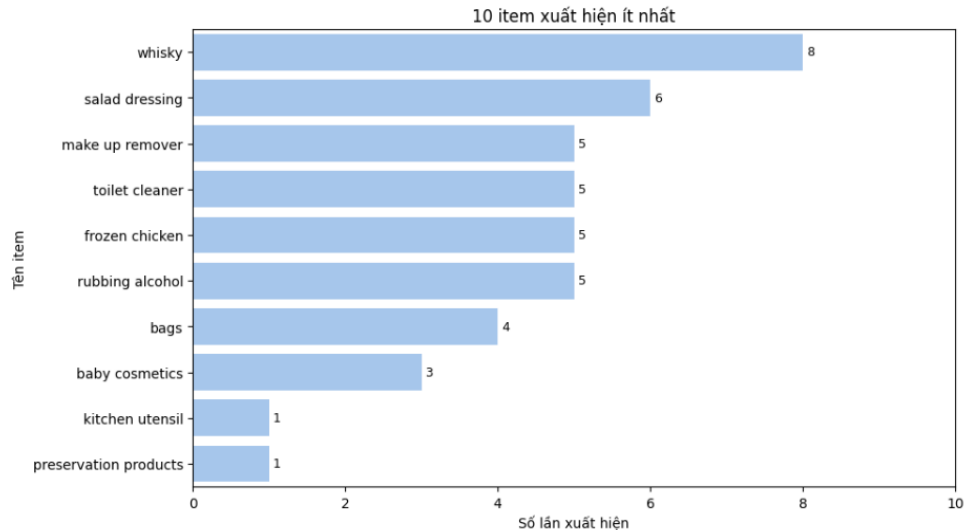


Hình ảnh phân tích biểu đồ cho thấy item đứng đầu là các loại sữa có số lần xuất hiện vượt trội hơn hẳn so với các mặt hàng còn lại. Những Item từ vị trí thứ 2 trở xuống có chênh lệch nhiều nhất với giá trị thấp hơn liền kề nó nhiều nhất không quá 300 lần xuất hiện.

whisky	8
salad dressing	6
make up remover	5
toilet cleaner	5
frozen chicken	5
rubbing alcohol	5
bags	4
baby cosmetics	3
kitchen utensil	1
preservation products	1

→ **Phân tích bảng thống kê:** Ngược lại, các item có số lần xuất hiện ít nhất sẽ là những giá trị bị loại bỏ khi xét đến các tập phổ biến. Mặt hàng có số lần xuất hiện thấp nhất là **kitchen utensil** và **preservation products** với chỉ 1 lần xuất hiện duy nhất.

- **Phân tích biểu đồ:**



Biểu đồ cho thấy có 3 mặt hàng có số lần xuất hiện khác biệt nhất so với các item còn lại là **kitchen utensil**, **preservation products** và **whisky**. Các item còn lại có số lần xuất hiện gần như tương đồng nhau và chỉ hơn kém các giá trị liền kề nhiều nhất chỉ 1 lần quan sát.

- **Thuộc tính *Member_number***

```
# Tổng số lượng khách hàng
customer_num = df['Member_number'].nunique()
# Tổng số lần mua hàng của tất cả khách hàng
purchase_num = df['Member_number'].count()
print(f'Tập dữ liệu này có {customer_num} khách hàng với tổng số {purchase_num} lượt mua hàng')
print(f'Bình quân mỗi khách hàng mua {(purchase_num/customer_num):.2f} lần/ đầu người')
```

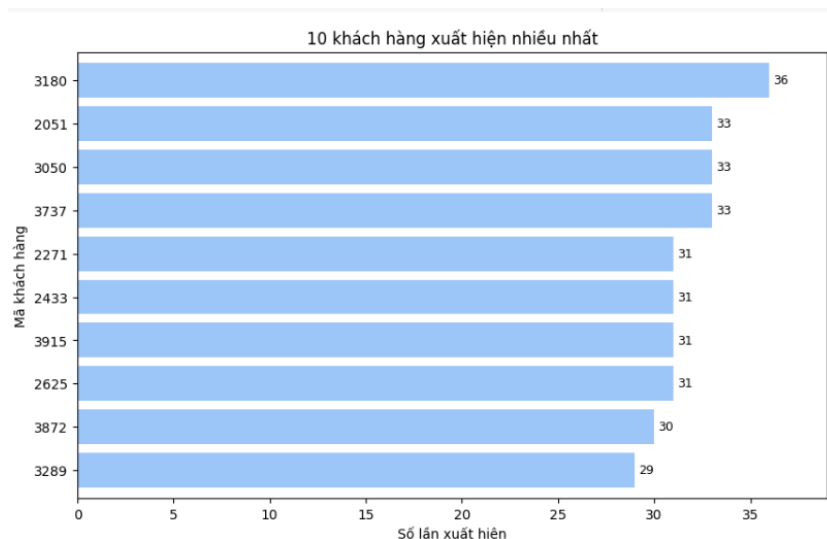
Tập dữ liệu này có 3898 khách hàng với tổng số 38765 lượt mua hàng
 Bình quân mỗi khách hàng mua 9.94 lần/ đầu người

Có tất cả **3898** khách hàng đã từng tham gia giao dịch tại cửa hàng trong số tổng cộng **38765** lượt mua hàng. Trung bình 1 người sẽ thực hiện 9,94 lần giao dịch tại cửa hàng này.

3180	36
3737	33
3050	33
2051	33
2625	31
3915	31
2433	31
2271	31
3872	30
3289	29

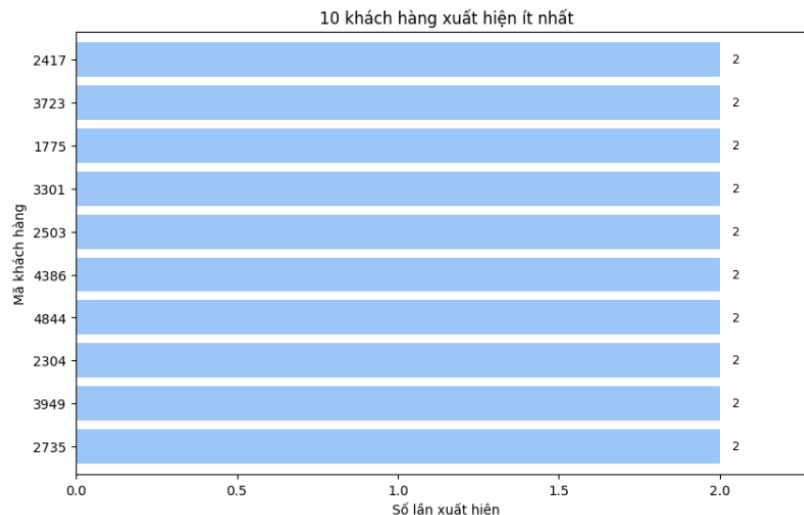
→ **Phân tích bảng thống kê:** Quan sát bảng thống kê top 10 khách hàng từng mua hàng nhiều nhất cho thấy số lần mua hàng nhiều nhất là 36 lần với khách hàng có mã số **3180**. Tính tổng số lần mua hàng trung bình của những khách hàng trong nhóm này sẽ là 31,8 lần, hơn gấp 3 lần trung bình 1 người thực hiện giao dịch tại cửa hàng.

- **Phân tích biểu đồ:**



Hình ảnh trực quan biểu đồ cho thấy số lần xuất hiện của các item không quá chênh lệch, chỉ duy nhất khách hàng đứng đầu có sự chênh lệch nhiều hơn những khách hàng còn lại.

2735	2
3949	2
2304	2
4844	2
4386	2
2503	2
3301	2
1775	2
3723	2
2417	2

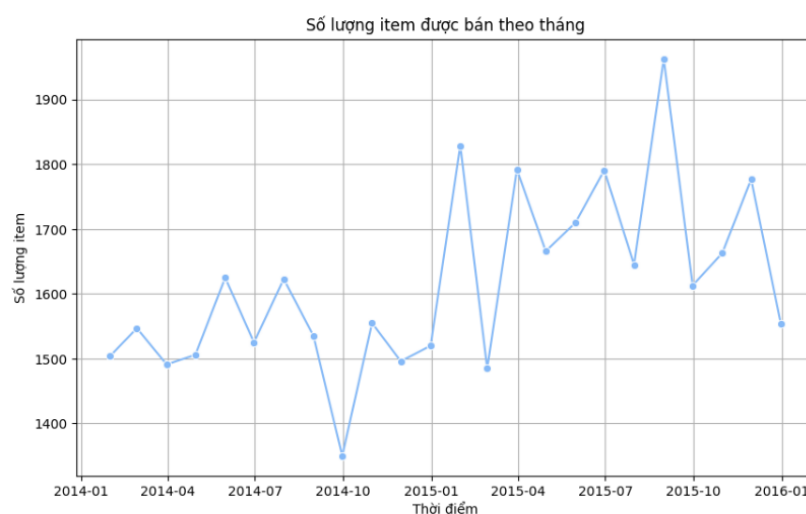


- Kết luận chung

Kết quả bảng thống kê và biểu đồ cho thấy những khách hàng thuộc nhóm ít thực hiện giao dịch tại cửa hàng có chung số lần mua hàng là 2. Đây là số lần mua ít nhất mà một khách hàng được ghi nhận, thấp gần gấp 5 lần lần mua trung bình đầu người.

• Thuộc tính Date:

Khác với hai thuộc tính trước, biến Date có tính chất chu kỳ vì nó ghi nhận dữ liệu thời gian thực hiện giao dịch tại cửa hàng. Do đó, nhóm sẽ thực hiện phân tích theo biểu đồ chuỗi thời gian để tìm ra những thời điểm cửa hàng đạt doanh thu cao nhất thông qua việc đếm số lượng những giao dịch được thực hiện tại các thời điểm này (nhóm sẽ chỉ cho rằng những thời điểm có số lần thực hiện giao dịch nhiều hơn sẽ có doanh thu cao hơn).



- Phân tích biểu đồ:

Xem xét biểu đồ, nhóm nhận thấy sự biến động doanh thu của cửa hàng biến động vô cùng mạnh. Tuy nhiên vẫn có thể nhận ra được những quy luật cụ thể qua từng năm quan sát.

Thứ nhất: Có thể thấy doanh thu vào tháng 10 luôn có sự sụt giảm một cách đột ngột.

Thứ hai: Doanh thu cao nhất của cả 2 năm quan sát đều được ghi nhận vào khoảng tháng 8.

Thứ ba: Doanh thu nếu xét theo từng năm đang có xu hướng tăng mạnh bằng chứng là khi nhóm so sánh các mốc có lượng item được bán ra nhiều và ít nhất của 2 năm 2014 và 2015 cho thấy doanh thu vào năm 2015 lớn hơn khá đáng kể.

Nhận xét chung:

Từ những thông tin thu được từ việc thăm dò, mỗi biến trong tập dữ liệu đều cung cấp thông tin mang ý nghĩa cho việc nghiên cứu. Do đó, việc loại bỏ các dòng dữ liệu trùng lặp theo cách thông thường không phù hợp. Chính vì thế, nhóm sẽ xem xét cách xử lý khác nhằm tối ưu hóa việc sử dụng dữ liệu, chi tiết sẽ được trình bày ở phần sau.

4. Chỉnh dạng dữ liệu

Trong phần tiền xử lý, nhóm đã nhận thấy bộ dữ liệu tồn tại rất nhiều dòng giá trị bị trùng lặp. Điều này làm cho việc khai phá luật kết hợp gặp nhiều trở ngại do các dòng của bộ dữ liệu không phải ở dạng chuẩn. Do đó, nhóm cần phải tiến hành chỉnh dạng dữ liệu về đúng dạng để có thể tiến hành tìm ra được những tập phổ biến bằng thuật toán ECLAT.

• Xử lý trùng lặp

Để giải quyết vấn đề trùng lặp dữ liệu, nhóm đã quyết định thực hiện gom nhóm. Đồng thời, để bộ dữ liệu phù hợp với thuật toán ECLAT, nhóm sẽ chuyển đổi bộ dữ liệu về hai cột chính, có dạng: ID và sản phẩm.

Tuy nhiên, việc gom nhóm gặp phải một số thách thức. Một trong số đó là việc nhiều khách hàng mua sản phẩm trong nhiều thời điểm khác nhau. Nếu chỉ gom nhóm theo mã khách hàng, nhóm sẽ không thể nhận biết rõ về mặt thời gian và xu hướng mua sắm của khách hàng. Ngược lại, nếu chỉ gom nhóm theo thời gian và bỏ đi cột mã khách hàng, nhóm sẽ không thể phân biệt được thói quen mua hàng của từng khách hàng.

Chính vì vậy, để giữ được thông tin của cả hai cột, nhóm đã thực hiện kết hợp hai cột Date và Mã khách hàng (Member_number) lại với nhau (ý nghĩa một giao dịch sẽ được xác định bởi hai yếu tố là ngày thực hiện giao dịch đó và mã số của khách hàng

thực hiện) và đặt tên cho cột kết hợp này là **invoice**. Điều này giúp chúng tôi có thể nắm bắt được cả thông tin về thời gian mua hàng và thói quen mua hàng của từng khách hàng, từ đó giúp chúng tôi tối ưu hóa việc phân tích dữ liệu.

Ở bước đầu tiên, để có thể kết hợp 2 cột lại với nhau, nhóm sẽ chỉnh lại dạng dữ liệu cho cột Date theo đúng định chuỗi tháng năm, điều này giúp cho dữ liệu của biến Date được lưu trữ tinh gọn thuận tiện cho việc phân tích và kết hợp các cột dữ liệu lại với nhau.

```
# Xử lý định dạng ngày tháng thành dạng chuỗi (mmyyyy)
eclat_df['Date'] = pd.to_datetime(eclat_df['Date']).dt.strftime('%m%Y')
eclat_df['Date'].head(5)
```

```
0    072015
1    052015
2    092015
3    122015
4    012015
```

Tiếp theo nhóm sẽ kết hợp 2 cột Date và Member_number lại với nhau để tạo thành cột mới có tên là **invoice**:

```
# Kết hợp cột Member_number và Date thành một cột invoice
eclat_df['invoice'] = eclat_df['Member_number'].astype(str) + '_' + eclat_df['Date']
eclat_df = eclat_df.drop(columns=['Member_number', 'Date'])
eclat_df['invoice'].head(5)
```

```
0    1808_072015
1    2552_052015
2    2300_092015
3    1187_122015
4    3037_012015
```

Sau khi tạo thành công cột dữ liệu mới, nhóm sẽ thực hiện gom nhóm các mặt hàng theo thuộc tính **invoice**, do **invoice** đại diện cho 1 giao dịch được thực hiện cho nên việc gom nhóm mặt hàng trong cột **group** (biến **itemDescription** sau khi được gom nhóm) sẽ giúp nhóm tìm ra được có bao nhiêu mặt hàng xuất hiện trong một giao dịch cụ thể.

```
# Xử lý trùng lặp ở cột invoice bằng cách dồn các giá trị của cột itemDescription thành tập hợp duy nhất
eclat_nongroupitems = eclat_nongroupitems.groupby('invoice')['itemDescription'].apply(set).reset_index()
eclat_nongroupitems.head(10)
```

	invoice	itemDescription
0	1000_032015	{whole milk, semi-finished bread, yogurt, saus...
1	1000_052015	{soda, pickled vegetables}
2	1000_062014	{salty snack, pastry, whole milk}
3	1000_072015	{canned beer, misc. beverages}
4	1000_112015	{hygiene articles, sausage}
5	1001_012015	{frankfurter, whipped/sour cream, soda}
6	1001_022015	{frankfurter, curd}
7	1001_042015	{white bread, beef}
8	1001_072014	{whole milk, rolls/buns, sausage}
9	1001_122014	{soda, whole milk}

Kết quả cho thấy, bảng dữ liệu hiện tại đã về đúng dạng cần thiết cho việc tìm kiếm các tập phổ biến và khai phá luật kết hợp, các dòng của biến invoice đại diện cho mỗi một giao dịch riêng biệt và các dòng trong thuộc tính itemDescription ghi nhận đầy đủ tất cả các mặt hàng có trong giao dịch đó.

Điều quan trọng hiện tại là liệu có các giao dịch nào bị trùng hay không.

```
# Thông tin bộ dữ liệu sau xử lý
eclat_nongroupitems.info()
dup_test = eclat_nongroupitems.copy()

# Kiểm tra xem có dòng dữ liệu nào trùng nhau không
dup_test['itemDescription'] = dup_test['itemDescription'].apply(','.join)
duplicates = dup_test[dup_test.duplicated()]

# Hiển thị các dòng dữ liệu trùng nhau
if not duplicates.empty:
    print("\nCác dòng dữ liệu trùng nhau:")
    print(duplicates)
else:
    print("\nKhông có dòng dữ liệu nào trùng nhau.")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13877 entries, 0 to 13876
Data columns (total 2 columns):
#   Column          Non-Null Count  Dtype
---  -
0   invoice         13877 non-null  object
1   itemDescription 13877 non-null  object
dtypes: object(2)
memory usage: 217.0+ KB
```

Không có dòng dữ liệu nào trùng nhau.

Kết quả cho thấy sau khi rút gọn có tất cả **13877** giao dịch đã được thực hiện tại cửa hàng đồng thời không có bất cứ dòng dữ liệu nào bị trùng lặp sau khi tiến hành chỉnh dạng dữ liệu. Vì vậy bước tiền xử lý và chỉnh dạng dữ liệu coi như là hoàn tất.

CHƯƠNG V. XÂY DỰNG THUẬT TOÁN ECLAT

- **Link nguồn:** [2_AssociationRule.ipynb - Colaboratory](#)

1. Các bước thực hiện

Dựa trên ý tưởng khi xây dựng bài toán ở **Chương III**, nhóm sẽ áp dụng phương pháp IT - Tree vào bài toán cụ thể trong lĩnh vực phân tích hành vi thị trường (Market Basket Analysis) với bộ dữ liệu được chọn.

Đầu tiên, nhóm sẽ thực hiện loại bỏ các dấu ký hiệu không cần thiết để việc xây dựng IT - Tree hiệu quả hơn.

```
# Loại bỏ các ký hiệu {, [, '
GrCsNonGroup_df['itemDescription'] = GrCsNonGroup_df['itemDescription'].astype(str)
GrCsNonGroup_df['itemDescription'] = GrCsNonGroup_df['itemDescription'].apply(lambda x: re.sub(r'[\{\[\']', '', str(x)))
GrCsNonGroup_df.head(5)
```

	invoice	itemDescription
0	1000_032015	whole milk, semi-finished bread, yogurt, sausage
1	1000_052015	soda, pickled vegetables
2	1000_062014	salty snack, pastry, whole milk
3	1000_072015	canned beer, misc. beverages
4	1000_112015	hygiene articles, sausage

Sau đó, nhóm sẽ đặt các ngưỡng chỉ số làm điều kiện ràng buộc cho mẫu cần tìm. Ở đây, nhóm chỉ xét những mặt hàng nào có từ 100 lần mua hàng trở lên, tương đương với mức minFreq là 700 (tương đương với mức minSup là 5%).

```
# Tính toán support và tìm các mẫu phổ biến
total_samples = df_tree.shape[0] # tổng số transactions
min_frequency = 700
min_support = min_frequency / total_samples
min_support
```

Theo các nghiên cứu mà nhóm tìm hiểu, để thực hiện ECLAT, nhóm cần thực hiện chuyển đổi cơ sở dữ liệu dạng nằm ngang thành cơ sở dữ liệu dọc (Vertical Database). Để thuật toán kết hợp có thể so sánh với ngưỡng hỗ trợ tối thiểu thì bảng dữ liệu cần hiện các mặt hàng (Item) đã xuất hiện để tính toán ngưỡng hỗ trợ của từng loại đó dựa trên số lượng mua hàng (Transactions).

invoice			itemDescription			Item	Transactions
0	1000_032015	sausage, whole milk, semi-finished bread, yogurt				0	sausage 1000_032015, 1000_112015, 1001_072014, 1003_10...
1	1000_052015		soda, pickled vegetables			1	whole milk 1000_032015, 1000_062014, 1001_072014, 1001_12...
2	1000_062014		whole milk, pastry, salty snack			2	semi-finished bread 1000_032015, 1074_042014, 1107_072014, 1111_11...
3	1000_072015		canned beer, misc. beverages			3	yogurt 1000_032015, 1008_072015, 1008_072015, 1009_05...
4	1000_112015		sausage, hygiene articles			4	soda 1000_052015, 1001_012015, 1001_122014, 1008_07...
...
13872	4999_092014		semi-finished bread, newspapers			162	frozen chicken 1833_072015, 2845_112014, 3288_042014, 3892_07...
13873	4999_122015		bottled water, herbs			163	salad dressing 2064_062014, 2155_112014, 2580_032014, 2851_05...
13874	5000_092014		fruit/vegetable juice, onions			164	specialty vegetables 2068_022014, 2080_072014, 3081_052014, 3095_07...
13875	5000_102015		soda, root vegetables, semi-finished bread			165	toilet cleaner 2341_032014, 2484_082014, 3256_122014, 3948_08...
13876	5000_112014		bottled beer, other vegetables			166	rubbing alcohol 2437_062014, 2803_042014, 2997_122015, 4495_08...
13877 rows x 2 columns						167 rows x 2 columns	

Với ý tưởng đó, nhóm xem xét tìm tập hợp các loại hàng xuất hiện trong các lần mua hàng và tính toán ngưỡng hỗ trợ của từng loại, nếu như ngưỡng hỗ trợ tính toán được thỏa điều kiện của ngưỡng hỗ trợ tối thiểu 700 lần mua hàng phía trên thì nhóm sẽ xem xét việc kết hợp, còn không thì loại bỏ mặt hàng này ra khỏi bộ dữ liệu để tiết kiệm thời gian tính toán.

```

1 # Tìm tập hợp mặt hàng đã mua
2 items = set()
3 for _, row in df_tree.iterrows():
4     item_set = row['itemDescription'].split(' ')
5     items.update(item_set)
6
7 # tính toán ngưỡng hỗ trợ của từng loại mặt hàng
8 frequent_patterns = []
9 for item in items:
10     support = df_tree[df_tree['itemDescription'].apply(lambda x:
11                                                         item in x.split(' '))].shape[0] / total_samples
12     if support >= min_support:
13         frequent_patterns.append([item], support)
14 frequent_patterns

```

```

([['citrus fruit'], 0.057072854363335014),
(['pastry'], 0.0556316206672912),
(['root vegetables'], 0.07480002882467392),
(['shopping bags'], 0.05116379620955538),
(['rolls/buns'], 0.11695611443395547),
(['other vegetables'], 0.1306478345463717),
(['tropical fruit'], 0.07278230165021259),
(['bottled water'], 0.0648555163219716),
(['soda'], 0.10398501116956115),
(['canned beer'], 0.05051524104633566),
(['sausage'], 0.06456726958276285),
(['whole milk'], 0.16811991064351084),
(['pip fruit'], 0.05260502990559919),
(['yogurt'], 0.0919507098075953)]

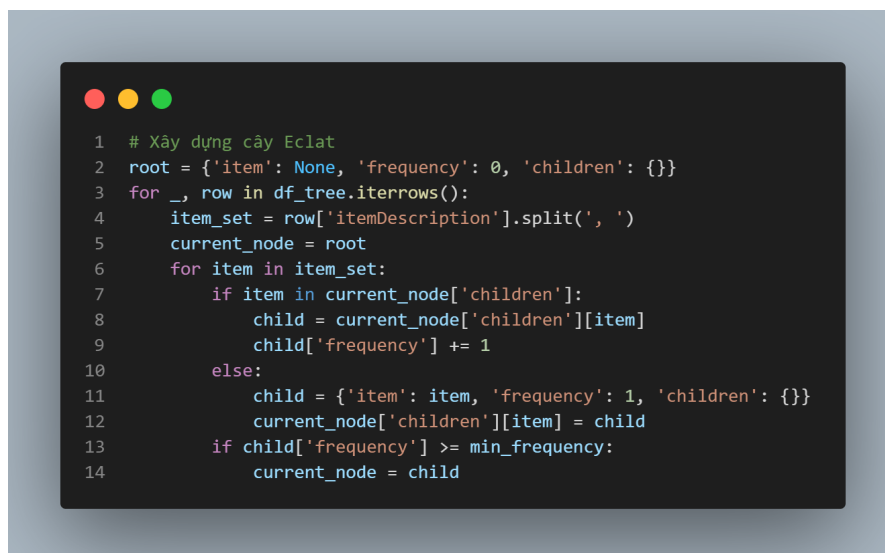
```

Kết quả trả về 14 mặt hàng là có số lần mua hàng từ 700 lần trở lên, cùng với đó

là ngưỡng hỗ trợ (minSup) của nó. Những mặt hàng này với số lần xuất hiện nhiều trong giao dịch sẽ là một cơ sở để nhóm có thể xây dựng nhiều luật kết hợp mạnh.

Tiếp theo, nhóm thực hiện triển khai mô hình cây ECLAT dựa trên lý thuyết về cây tập hợp. Ý tưởng xây dựng mã nguồn này sẽ triển khai một node ban đầu là null, qua từng giao dịch, cây sẽ được cập nhật dựa trên sự xuất hiện của các loại mặt hàng mới. Nếu loại mặt hàng chưa tồn tại, cây sẽ tạo một node mới là tên loại hàng này, nếu mặt hàng đã xuất hiện từ trước, cây sẽ tăng số lần xuất hiện lên và quá trình này sẽ lặp lại cho đến khi hết tất cả các loại hàng.

Lưu ý rằng trong quá trình xây dựng cây, có những giao dịch sẽ xuất hiện nhiều mặt hàng (một khách hàng mua nhiều item), vậy những mặt hàng được mua sau sẽ trở thành node con (children) của node hiện tại.



```
1 # Xây dựng cây Eclat
2 root = {'item': None, 'frequency': 0, 'children': {}}
3 for _, row in df_tree.iterrows():
4     item_set = row['itemDescription'].split(',')
5     current_node = root
6     for item in item_set:
7         if item in current_node['children']:
8             child = current_node['children'][item]
9             child['frequency'] += 1
10        else:
11            child = {'item': item, 'frequency': 1, 'children': {}}
12            current_node['children'][item] = child
13            if child['frequency'] >= min_frequency:
14                current_node = child
```

Ban đầu, IT - Tree gốc sẽ được khởi tạo với node ban đầu là null bởi vì chưa có loại mặt hàng nào được đại diện. Cấu trúc cây nhóm xây dựng bao gồm:

- **Node “item”**: tên mặt hàng đại diện cho node đó.
- **Ngưỡng xuất hiện “frequency”**: số lần xuất hiện của mặt hàng item này trong giao dịch của bộ dữ liệu.
- **“children”**: tương tự với node con của node hiện tại đang xét, mỗi node con này cũng bao gồm tên mặt hàng và ngưỡng số lần xuất hiện tương ứng với mặt hàng đó.

Quá trình xây dựng cây có thể hình dung như sau:

- Sau khi khởi tạo một node gốc, gọi là rễ (root), cây tập hợp sẽ duyệt qua từng số lần mua hàng của bộ dữ liệu. Sau đó, cây sẽ kiểm tra sự tồn tại trong danh sách node con đối với từng mặt hàng trong item_set.

- Nếu như mặt hàng đó đã xuất hiện rồi, cây xem xét tăng số lần xuất hiện thêm 1 lần.
- Nếu như mặt hàng này chưa từng tồn tại trong node con của node đang xét, cây sẽ tạo một node con mới từ node hiện tại và gán thuộc tính cấu trúc của node vào node con.
- Sau khi xem xét node hiện tại dựa trên hai quy tắc trên, cây sẽ kiểm tra xem ngưỡng xuất hiện của node con có thỏa mãn điều kiện đề bài hay không, nếu có, chuyển node hiện tại sang node con về tiếp tục xây dựng dựa trên các quy tắc đã làm bắt đầu từ node con này.
- Sau khi cây duyệt qua tất cả các giao dịch của bộ dữ liệu, thuật toán sẽ kết thúc việc cập nhật và trả về cây tập hợp đã xây dựng được.

```
{'item': None,
 'frequency': 0,
 'children': {'whole milk': {'item': 'whole milk',
 'frequency': 2014,
 'children': {'cookware': {'item': 'cookware',
 'frequency': 2,
 'children': {}},
 'oil': {'item': 'oil', 'frequency': 3, 'children': {}},
 'tropical fruit': {'item': 'tropical fruit',
 'frequency': 76,
 'children': {}},
 'soda': {'item': 'soda', 'frequency': 9, 'children': {}},
 'waffles': {'item': 'waffles', 'frequency': 22, 'children': {}},
 'domestic eggs': {'item': 'domestic eggs',
 'frequency': 30,
 'children': {}},
 'fruit/vegetable juice': {'item': 'fruit/vegetable juice',
 'frequency': 22,
 'children': {}},
 'bottled water': {'item': 'bottled water',
 'frequency': 31,
 'children': {}},
 'other vegetables': {'item': 'other vegetables',
 'frequency': 63,
 'children': {}},
 ...
 'children': {}},
 'toilet cleaner': {'item': 'toilet cleaner', 'frequency': 5, 'children': {}},
 'rubbing alcohol': {'item': 'rubbing alcohol',
 'frequency': 4,
 'children': {}}}}
```

Bộ dữ liệu này bao gồm **13878** giao dịch mỗi khách hàng và **167** loại mặt hàng. Từ cây nhóm thu được một số thông tin cơ bản như node đầu tiên là mặt hàng **“whole milk”** với tổng số lần khách hàng mua mặt hàng này trước các mặt hàng khác là **2014** lần. Trong mặt hàng này, cây duyệt qua tất cả quan sát và tìm được một số node con như **“cookware”**, **“oil”** ... và trong mỗi node con này có thể sẽ bao gồm nhiều node con khác.

Sau khi duyệt hết tất cả node con của mặt hàng **“whole milk”**, cây sẽ tạo một node mới và cứ lặp lại như vậy cho đến hết bộ dữ liệu. Qua trực quan khái quát và kết quả

trong cell, nhóm thấy có một quy luật phổ biến để nhận biến node con của root chính là “**3 lần dấu ngoặc nhọn**”.

Cuối cùng, nhóm xem xét DataFrame mẫu phổ biến vừa tìm được từ phương pháp IT - Tree và lọc theo thứ tự giảm dần của ngưỡng hỗ trợ.

	Pattern	Support
5	[whole milk]	2333
10	[other vegetables]	1813
2	[rolls/buns]	1623
12	[soda]	1443
6	[yogurt]	1276
0	[root vegetables]	1038
3	[tropical fruit]	1010
8	[bottled water]	900
11	[sausage]	896
13	[citrus fruit]	792
7	[pastry]	772
4	[pip fruit]	730
9	[shopping bags]	710
1	[canned beer]	701

Kết quả thu được ở mẫu phổ biến vừa tìm được bao gồm 13 sản phẩm với **whole milk** là sản phẩm có minFreq cao nhất là 2333 (tương đương với ngưỡng minSup = ~17%). Từ bảng kết quả nhóm cũng nhận thấy được một số điểm hạn chế ban đầu của bộ dữ liệu khi thực hiện xây dựng thuật toán ECLAT như sau:

Các mẫu phổ biến tìm được chỉ có duy nhất 1 phần tử trong mẫu, cho thấy sự thiếu đa dạng của các kết hợp sản phẩm, ngoài ra ngưỡng xuất hiện tối thiểu của các sản phẩm cũng đều rất thấp, chỉ dưới 20%.

2. Tìm luật kết hợp mạnh

Nắm được phương pháp thực hiện, nhóm tiếp tục thực hiện tìm những luật kết hợp mạnh của bộ dữ liệu trên.

Vì tính chất bộ dữ liệu có rất nhiều giao dịch và loại sản phẩm, nhóm quyết định sử dụng hàm luật kết hợp trong thư viện **pyECLAT** để tiết kiệm thời gian nghiên cứu.

Đầu tiên, nhóm thực hiện chuyển đổi DataFrame cho phù hợp với yêu cầu đầu vào của thư viện.

```

1 # loại bỏ các ký hiệu {, [, '
2 nongroup_df['itemDescription'] = nongroup_df['itemDescription'].astype(str)
3 nongroup_df['itemDescription'] = nongroup_df['itemDescription'].apply(lambda
4                                 x: re.sub(r'\{|\[|\'', '', str(x)))
5
6 # tách item từ cột itemDescription
7 nongroup_eclat_df = nongroup_df['itemDescription'].str.get_dummies(', ')
8
9 # chuyển đổi format để sử dụng pyECLAT
10 for column in nongroup_eclat_df.columns:
11     nongroup_eclat_df[column] = nongroup_eclat_df[column].apply(lambda
12                             x: column if x else None)
13 nongroup_eclat_df.columns = range(len(nongroup_eclat_df.columns))

```

Sau khi thực hiện xóa những ký tự dư thừa và chuyển đổi DataFrame gốc, bộ dữ liệu của nhóm thể hiện như sau:

	0	1	2	3	4	5	6	7	8	9	...	157	158	159	160	161	162	163	164	165	166
0	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	whole milk	yogurt	None
1	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
2	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	whole milk	None	None
3	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
...
13872	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
13873	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
13874	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
13875	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None
13876	None	None	None	None	None	None	None	None	None	None	...	None	None	None	None	None	None	None	None	None	None

13877 rows x 167 columns

Có thể hiểu rằng ở dạng biểu diễn này, mỗi một cột thuộc tính sẽ đại diện cho một mặt hàng item được mua từ khách hàng, còn mỗi dòng sẽ là một giao dịch. Nhưng với bộ dữ liệu không gom nhóm sản phẩm, mỗi quan sát sẽ tương ứng với mỗi lần mà khách hàng mua những item đó. Nếu mặt hàng đó được mua thì dòng quan sát sẽ hiện tên sản phẩm, nếu không thì mặt hàng sẽ hiện là “None”.

Tiếp theo, nhóm sử dụng ngưỡng hỗ trợ 5% giống với ngưỡng mà nhóm đã thực hiện ở phương pháp sử dụng Cây tập hợp phía trên.

```

nongroup_eclat1 = ECLAT(nongroup_eclat_df, verbose = True)
index1, support1 = nongroup_eclat1.fit(min_support=0.05, separator=', ')

```

	Item	Transactions
0	other vegetables	[13, 19, 22, 39, 47, 60, 61, 83, 103, 105, 128...
1	rolls/buns	[8, 15, 17, 22, 24, 25, 26, 35, 38, 42, 52, 55...
2	root vegetables	[16, 21, 30, 42, 48, 60, 68, 81, 83, 88, 89, 9...
3	canned beer	[3, 18, 51, 71, 72, 85, 88, 95, 124, 237, 256,...

4	soda	[1, 5, 9, 30, 63, 80, 98, 105, 111, 150, 152, ...
5	bottled water	[25, 36, 38, 43, 94, 96, 114, 138, 149, 179, 1...
6	yogurt	[0, 30, 32, 39, 40, 49, 62, 70, 75, 81, 90, 10...
7	citrus fruit	[38, 54, 90, 123, 138, 143, 186, 212, 234, 250...
8	whole milk	[0, 2, 8, 9, 10, 20, 21, 22, 26, 27, 39, 40, 4...
9	shopping bags	[22, 28, 42, 63, 72, 84, 108, 123, 168, 172, 1...
10	tropical fruit	[11, 20, 30, 32, 42, 46, 75, 83, 111, 115, 122...
11	sausage	[0, 4, 8, 17, 49, 78, 82, 90, 110, 117, 126, 1...
12	pastry	[2, 21, 33, 38, 108, 109, 149, 182, 255, 269, ...
13	pip fruit	[20, 34, 35, 56, 59, 113, 119, 126, 127, 152, ...

Kết quả thực nghiệm cho thấy với ngưỡng điều kiện này, có 13 mặt hàng thỏa mãn số lần giao dịch khi tính toán theo hàm kết hợp. Tuy nhiên, nhóm thấy rằng không có bất kỳ luật nào kết hợp từ 2 loại mặt hàng trở lên. Vậy nên nhóm sẽ thực hiện giảm ngưỡng hỗ trợ minSup xuống còn 1%, tuy nhiên điều này có thể đánh đổi bằng việc các luật kết hợp mới khi được sinh ra sẽ không mang nhiều giá trị ý nghĩa khi phân tích.

```
nongroup_eclat2 = ECLAT(nongroup_eclat_df, verbose = False)
index1, support1 = nongroup_eclat2.fit(min_support=0.01, min_combination=1, separator=', ')
nongroup_eclat_df2 = pd.DataFrame(list(index1.items()), columns=['Item', 'Transactions'])
nongroup_eclat_df2
```

	Item	Transactions
0	other vegetables	[13, 19, 22, 39, 47, 60, 61, 83, 103, 105, 128...
1	soft cheese	[56, 178, 472, 496, 517, 527, 596, 620, 663, 7...
2	frankfurter	[5, 6, 26, 36, 39, 40, 59, 109, 124, 157, 174,...
3	rolls/buns	[8, 15, 17, 22, 24, 25, 26, 35, 38, 42, 52, 55...
4	sliced cheese	[37, 73, 141, 193, 231, 273, 400, 498, 508, 56...
...
71	rolls/buns, whole milk	[8, 22, 26, 52, 116, 131, 143, 153, 181, 470, ...
72	soda, whole milk	[9, 153, 177, 216, 317, 373, 374, 449, 478, 48...
73	yogurt, whole milk	[0, 39, 40, 49, 116, 210, 325, 334, 350, 382, ...
74	whole milk, tropical fruit	[20, 136, 203, 207, 216, 224, 266, 317, 325, 3...
75	whole milk, sausage	[0, 8, 49, 126, 311, 319, 508, 529, 531, 699, ...

76 rows × 2 columns

Nhóm thực hiện khởi tạo lại hàm kết hợp ECLAT cho bộ dữ liệu không gom nhóm này, nhưng ở lần này, ngưỡng hỗ trợ ràng buộc đã được chỉnh về mức thấp hơn (0,01), đồng thời với đó, nhóm cũng chỉnh thêm điều kiện số phần tử mặt hàng kết

hợp thấp nhất là 1 để tiết kiệm thời gian chạy thuật toán.

Thời gian chạy lâu hơn bởi vì thuật toán phải tổ chức xem xét thêm những giao dịch có ngưỡng hỗ trợ thấp hơn mức ban đầu. Sau khi hoàn thành, nhóm lọc DataFrame theo thứ tự giảm dần của ngưỡng hỗ trợ và đồng thời khởi tạo thêm cột “Length” là số lượng phần tử mặt hàng trong từng luật.

```
nongroup_frequent_itemsets = pd.DataFrame(list(support1.items()), columns=['Items', 'Support'])
nongroup_frequent_itemsets = nongroup_frequent_itemsets.sort_values(by='Support', ascending=False).reset_index()
nongroup_frequent_itemsets = nongroup_frequent_itemsets.drop(columns = 'index')
nongroup_frequent_itemsets['Length'] = nongroup_frequent_itemsets['Items'].str.count(',') + 1
nongroup_frequent_itemsets
```

	Items	Support	Length
0	whole milk	0.168120	1
1	other vegetables	0.130648	1
2	rolls/buns	0.116956	1
3	soda	0.103985	1
4	yogurt	0.091951	1
...
71	other vegetables, yogurt	0.010737	2
72	flour	0.010521	1
73	whole milk, tropical fruit	0.010449	2
74	semi-finished bread	0.010233	1
75	rolls/buns, soda	0.010089	2

76 rows × 3 columns

Sau đó, nhóm thực hiện lọc ra những luật kết hợp có số lượng phần tử từ 2 mặt hàng trở lên để xem xét tính toán Confidence, từ đó, tìm ra độ mạnh của luật kết hợp đó.

```
# luật kết hợp > 2 item với ngưỡng hỗ trợ trên 1%
nongroup_2items = nongroup_frequent_itemsets[(nongroup_frequent_itemsets['Length'] >= 2)
                                              & (nongroup_frequent_itemsets['Support'] >= 0.01)]
nongroup_2items
```

	Items	Support	Length
43	other vegetables, whole milk	0.018448	2
48	rolls/buns, whole milk	0.017079	2
59	soda, whole milk	0.014484	2
60	yogurt, whole milk	0.013692	2
62	other vegetables, rolls/buns	0.012827	2
65	other vegetables, soda	0.012178	2
68	whole milk, sausage	0.011170	2
71	other vegetables, yogurt	0.010737	2
73	whole milk, tropical fruit	0.010449	2
75	rolls/buns, soda	0.010089	2

Kết quả trả về cho thấy có 10 luật kết hợp có thể đi tính toán độ mạnh với ngưỡng hỗ trợ chỉ lớn hơn 1% không đáng kể. Cuối cùng, nhóm đi tính toán từng khả năng xuất hiện của luật theo công thức ở phần lý thuyết và gán giá trị vào trong DataFrame vừa tìm được.

```
def calculate_nongroup_confidence(row):
    items = row['Items'].split(', ')
    first_item = items[0]
    # tìm support
    support = row['Support']
    first_support = nongroup_frequent_itemsets[nongroup_frequent_itemsets['Items'].str.contains(first_item)]['Support'].values[0]
    # tìm confidence
    confidence = support / first_support
    return confidence

# Thêm cột Confidence vào DataFrame
nongroup_2items['Confidence'] = nongroup_2items.apply(calculate_nongroup_confidence, axis=1)
nongroup_2items
```

	Items	Support	Length	Confidence
43	other vegetables, whole milk	0.018448	2	0.141202
48	rolls/buns, whole milk	0.017079	2	0.146026
59	soda, whole milk	0.014484	2	0.139293
60	yogurt, whole milk	0.013692	2	0.148903
62	other vegetables, rolls/buns	0.012827	2	0.098180
65	other vegetables, soda	0.012178	2	0.093216
68	whole milk, sausage	0.011170	2	0.066438
71	other vegetables, yogurt	0.010737	2	0.082184
73	whole milk, tropical fruit	0.010449	2	0.062152
75	rolls/buns, soda	0.010089	2	0.086260

Qua kết quả thực nghiệm, nhóm thấy được rằng các giá trị Confidence của các

luật sẽ tương đối thấp, hoặc hiểu rằng, khi mua sản phẩm đầu tiên trong luật kết hợp thì chỉ có khoảng 10% là khách hàng sẽ mua tiếp sản phẩm thứ hai trong luật.

Giải thích cho mức ngưỡng hỗ trợ và mức ngưỡng Confidence tương đối thấp này là vì bộ dữ liệu ban đầu có quá nhiều số lượng loại mặt hàng, vậy nên các giao dịch cũng sẽ bị chia ra quá nhiều làm giảm mức hỗ trợ của từng luật, điều này cũng kéo theo làm giảm khả năng mua hàng của món hàng thứ hai khi người mua có thể mua nhiều món hàng khác.

CHƯƠNG VII. CẢI THIẾN & ĐÁNH GIÁ

1. Gom nhóm sản phẩm (Tối ưu độ lớn của dữ liệu)

- Ý tưởng

Với mục tiêu của nhóm là tìm ra các mẫu phổ biến đơn giản và dễ hiểu nhất để cải thiện việc trưng bày sản phẩm trong cửa hàng. Điều này đòi hỏi nhóm xác định các tập hợp cụ thể của các sản phẩm mà nhóm muốn trưng bày lên kệ một cách tốt nhất. Tuy nhiên, khi dữ liệu có quá nhiều thuộc tính với độ phân phối cao, nghĩa là sẽ có số sản phẩm xuất hiện với tần suất thấp, chưa kể đến những sản phẩm có cùng công năng khi xuất hiện quá nhiều sẽ lấn át những sản phẩm có tần suất thấp đó. Điều này đồng nghĩa với việc khi nhóm tìm kiếm mẫu phổ biến, khả năng cao những sản phẩm đó sẽ bị bỏ qua (kết quả ở bộ dữ liệu trên), gây lãng phí thông tin quan trọng mà nhóm đang tìm kiếm.

Để giải quyết vấn đề này, nhóm nghiên cứu đã quyết định gộp các nhóm sản phẩm ban đầu vào từng nhóm mới thích hợp hơn. Điều này giúp nhóm xây dựng một bộ dữ liệu mới có độ tương tự với bộ dữ liệu gốc, nhưng đồng thời đảm bảo rằng các sản phẩm này sẽ gia tăng khả năng những sản phẩm này đều được xuất hiện trong các mẫu phổ biến. Điều này giúp nhóm không bỏ qua thông tin quan trọng và đảm bảo rằng nhóm có thể xác định được những tập hợp sản phẩm sẽ trưng bày tốt nhất trên kệ.

- Các bước thực hiện

Ở bước đầu tiên, nhóm sẽ thực hiện việc gom **167** mặt hàng về thành **21** nhóm hàng hóa để dễ dàng quan sát và đánh giá hơn thông qua các nhóm mặt hàng đại diện.

- *Gom nhóm thực phẩm*: Bao gồm 8 nhóm chính, cụ thể: trái cây, rau củ, các loại hạt, thịt, thực phẩm từ sữa, kẹo ngọt, gia vị, các thực phẩm còn lại

```
# subgroup cho nhóm thực phẩm
'fruit': ['tropical fruit', 'pip fruit', 'citrus fruit', 'canned fruit', 'frozen fruits', 'berries',
          'grapes', 'tidbits'],
'vegetables': ['other vegetables', 'root vegetables', 'pickled vegetables', 'frozen vegetables',
               'canned vegetables', 'specialty vegetables', 'frozen potato products', 'onions'],
'grains': ['brown bread', 'rice', 'pastry', 'pasta', 'long life bakery product', 'white bread', 'waffles',
            'potato products', 'semi-finished bread', 'zwieback', 'roll products', 'cereals'],
'meats': ['frankfurter', 'chicken', 'sausage', 'hamburger meat', 'pork', 'ham', 'turkey', 'frozen meals',
           'fish', 'meat', 'frozen chicken', 'frozen fish', 'canned fish', 'liver loaf', 'organic sausage',
           'meat spreads', 'frozen chicken'],
'dairy': ['butter', 'yogurt', 'curd cheese', 'processed cheese', 'curd', 'hard cheese', 'cream cheese',
           'sliced cheese', 'specialty cheese', 'spread cheese', 'soft cheese'],
'sugary': ['chocolate', 'specialty bar', 'popcorn', 'dessert', 'specialty chocolate', 'ice cream',
            'honey', 'frozen dessert', 'chewing gum', 'jam', 'chocolate marshmallow', 'sweet spreads',
            'cooking chocolate', 'syrup', 'candy', 'cake bar'],
'seasoning': ['sugar', 'herbs', 'salt', 'baking powder', 'vinegar', 'artif. sweetener', 'spices',
               'pudding powder', 'sauces', 'mustard', 'mayonnaise', 'seasonal products', 'ketchup',
               'salad dressing'],
'other foods': ['cream', 'flour', 'margarine', 'salty snack', 'domestic eggs', 'nuts/prunes', 'soups',
                'specialty fat', 'snack products', 'whipped/sour cream', 'nut snack', 'Instant food products',
                'organic products', 'ready soups', 'preservation products', 'flower (seeds)',
                'packaged fruit/vegetables'],
```

- *Gom nhóm đồ uống:* Bao gồm 6 nhóm chính, cụ thể: sữa, nước ép, nước tăng lực, nước khoáng, rượu, các thức uống còn lại.

```
# subgroup cho nhóm đồ uống
'milk': ['UHT-milk', 'condensed milk', 'butter milk', 'whole milk'],
'juice': ['fruit/vegetable juice'],
'energy drinks': ['instant coffee', 'coffee', 'soda', 'soft drinks'],
'water': ['bottled water'],
'alcohol': ['red/blush wine', 'white wine', 'bottled beer', 'liqueur', 'whisky', 'brandy', 'white wine',
            'rum', 'misc. beverages', 'canned beer', 'sparkling wine', 'prosecco', 'liquor (appetizer)',
            'liqueur (appetizer)', 'liquor'],
'other drinks': ['tea', 'cocoa drinks', 'beverages'],
```

- *Gom nhóm các mặt hàng còn lại:* Bao gồm 6 nhóm chính, cụ thể: Thực phẩm thú cưng, sản phẩm hóa học, cặp sách/ balo, dụng cụ vệ sinh nhà cửa, các sản phẩm còn lại.

```
# subgroup cho nhóm còn lại
'pet care': ['dog food', 'pet care', 'cat food'],
'chemistry products': ['detergent', 'decalcifier', 'bathroom cleaner', 'abrasive cleaner', 'rubbing alcohol',
                       'softener', 'toilet cleaner'],
'bags': ['shopping bags', 'cling film/bags', 'bags'],
'cosmetics': ['skin care', 'male cosmetics', 'make up remover', 'hair spray', 'female sanitary products',
              'hygiene articles', 'baby cosmetics', 'dental care'],
'house facilities': ['kitchen utensil', 'cleaner', 'dish cleaner', 'house keeping products', 'cookware'],
'others': ['newspapers', 'finished products', 'pot plants', 'photo/film', 'soap', 'light bulbs', 'napkins',
           'kitchen towels', 'flower soil/fertilizer', 'dishes', 'candles']
```

Sau đó đặt tên cột vừa tạo thành là **group_itemDescription** (biến **itemDescription** sau khi được gom nhóm)

```
eclat_df['group_itemDescription'] = eclat_df['itemDescription'].apply(lambda x: next((key for key,
                                             value in group.items() if x in value), 'other item'))
```

- *Kiểm tra lại*

	Số quan sát	Số giá trị unique	Kiểu dữ liệu
Member_number	38765	3898	int64
Date	38765	728	datetime64[ns]
itemDescription	38765	167	object
group_itemDescription	38765	21	object

	Member_number	Date	itemDescription	group_itemDescription
0	1808	2015-07-21	tropical fruit	fruit
1	2552	2015-05-01	whole milk	milk
2	2300	2015-09-19	pip fruit	fruit
3	1187	2015-12-12	other vegetables	vegetables
4	3037	2015-01-02	whole milk	milk
...
38760	4471	2014-08-10	sliced cheese	dairy
38761	2022	2014-02-23	candy	sugary
38762	1097	2014-04-16	cake bar	sugary
38763	1510	2014-03-12	fruit/vegetable juice	juice
38764	1521	2014-12-26	cat food	pet care

Sau khi hoàn thành, nhóm đã tạo thành công 1 thuộc tính nữa ghi nhận các nhóm lớn của các mặt hàng bao gồm 21 mục hàng hóa khác nhau.

- **Kiểm thử**

- ***Gia tăng độ lớn mẫu phổ biến:***

Kết quả tìm mẫu phổ biến bằng giải thuật ECLAT khi gom nhóm các sản phẩm như sau:

	Pattern	Support
8	[meats]	3595
7	[vegetables]	3585
1	[dairy]	3290
9	[milk]	2909
6	[fruit]	2904
5	[grains]	2485
3	[other foods]	2409
10	[other item]	2365
0	[alcohol]	2138
14	[sugary]	2102
4	[energy drinks]	1922
2	[others]	1411
13	[seasoning]	1077
11	[water]	900
12	[bags]	786

Sau khi gom nhóm các sản phẩm thì số lượng mẫu phổ biến xác định được đã tăng lên 1 (14 mẫu phổ biến), ngoài ra việc thay đổi rõ rệt về minFreq cũng cho thấy hiệu quả việc gom nhóm.

Từ bảng kết quả, nhóm đã nhận thấy rằng trong bộ dữ liệu không gom nhóm, sản phẩm "**whole milk**" (thuộc nhóm "**milk**") đã được xác định có minFreq cao nhất. Tuy nhiên, sau khi thực hiện gom nhóm, sản phẩm thuộc

nhóm "**meats**" đã xuất hiện với minFreq cao nhất, mặc dù trước đây nó chưa xuất hiện trong bảng mẫu phổ biến. Điều này cho thấy rằng sự xuất hiện quá ít của những sản phẩm có chức năng tương tự nhau có thể dẫn đến việc chúng bị bỏ qua, nhưng trong thực tế chúng lại có ảnh hưởng đáng kể, điều này gây nên sự lãng phí thông tin trong quá trình phân tích.

- **Gia tăng số lượng luật kết hợp và độ tin cậy:**

Tương tự như cách đã thực hiện với bộ dữ liệu không gom nhóm ở **Chương VI**, nhóm cũng thực hiện giảm độ hỗ trợ của bộ dữ liệu này xuống 1% để tìm ra các mẫu vật phẩm, trong đó chúng xuất hiện 2 phần tử trong tập hợp. Với kết quả cho thấy, số lượng mẫu phổ biến đã gia tăng đáng kể so với bộ dữ liệu không gom nhóm, cùng với độ hỗ trợ của các cũng gia tăng đáng kể (với mẫu có support cao nhất, tăng từ 16.8% đã gia tăng lên 25.9%). Ngoài ra, đáng kể đến là còn có sự xuất hiện của các mẫu có 3 phần tử.

	Items	Support	Length
0	meats	0.259062	1
1	vegetables	0.258341	1
2	dairy	0.237083	1
3	milk	0.209627	1
4	fruit	0.209267	1
...
106	alcohol, seasoning	0.010737	2
107	grains, water	0.010665	2
108	vegetables, meats, fruit	0.010521	3
109	fruit, bags	0.010089	2
110	seasoning, other item	0.010089	2

111 rows × 3 columns

Tiếp đến, khi xét luật kết hợp mạnh, nhóm cũng nhận thấy sự gia tăng đáng kể của độ chính xác của các luật (với mẫu có conf cao nhất, từ 1.4% đã gia tăng lên đến 5.7%), cũng như độ độ hỗ trợ cũng đều gia tăng, ngoài ra số lượng luật cũng gia tăng lên đáng kể. Cụ thể, xét Top 10 luật mạnh nhất nhóm thu được kết quả như sau:

	Items	Support	Length	Confidence
14	dairy, meats	0.057289	2	0.241641
15	vegetables, meats	0.056840	2	0.219247
17	vegetables, dairy	0.051812	2	0.200558
18	meats, milk	0.047200	2	0.182197
19	vegetables, fruit	0.045887	2	0.176848
20	vegetables, milk	0.045111	2	0.174616
21	meats, fruit	0.043021	2	0.166064
22	dairy, milk	0.042805	2	0.180547
23	dairy, fruit	0.042012	2	0.177204
24	grains, meats	0.041147	2	0.229779

- *Tối ưu hóa thời gian thực hiện của giải thuật:*

Tiếp theo, nhóm sẽ thực hiện so sánh thời gian thực nghiệm của phương pháp này đối với cả hai mẫu dữ liệu và đưa ra nhận định:

```

1  # Bắt đầu đo thời gian
2  start_time = time.time()
3
4  # Tính toán support và tìm các mẫu phổ biến
5  total_samples = GrcsNonGroup_df.shape[0]
6  min_support_percent = 5
7  min_support = min_support_percent * total_samples / 100
8  frequent_patterns = []
9  for item in items_nongroup:
10     support = GrcsNonGroup_df[GrcsNonGroup_df['itemDescription'].apply(lambda
11                                     x: item in x.split(', '))].shape[0]
12     if support >= min_support:
13         frequent_patterns.append([item], support))
14  # Xây dựng cây Eclat
15  root = {'item': None, 'support': 0, 'children': {}}
16  for _, row in GrcsNonGroup_df.iterrows():
17     item_set = row['itemDescription'].split(', ')
18     current_node = root
19     for item in items_nongroup:
20         if item in current_node['children']:
21             child = current_node['children'][item]
22             child['support'] += 1
23         else:
24             child = {'item': item, 'support': 1, 'children': {}}
25             current_node['children'][item] = child
26             if child['support'] >= min_support:
27                 current_node = child
28  # Tìm các mẫu phổ biến từ cây Eclat
29  def find_frequent_patterns(node, prefix):
30     frequent_patterns = []
31     if node['item'] is not None and node['support'] >= min_support:
32         frequent_patterns.append((prefix + [node['item']], node['support']))
33     for child in node['children'].values():
34         frequent_patterns.extend(find_frequent_patterns(child,
35                                                         prefix + [node['item']]))
36     return frequent_patterns
37  if root is None:
38     frequent_patterns = find_frequent_patterns(root, [])
39
40  # Kết thúc đo thời gian
41  end_time = time.time()
42
43  # Tính thời gian thực thi
44  execution_time_ngr = end_time - start_time

```

Nhóm thực hiện đo lường tương tự với mẫu dữ liệu của bộ dữ liệu gom nhóm khách hàng và nhóm sản phẩm đặc trưng.

```
1 # Bắt đầu đo thời gian
2 start_time = time.time()
3
4 # Tính toán support và tìm các mẫu phổ biến
5 total_samples = GrcsGroup_df.shape[0]
6 min_support_percent = 5
7 min_support = min_support_percent * total_samples / 100
8
9 frequent_patterns = []
10 for item in items:
11     support = GrcsGroup_df[GrcsGroup_df['group_itemDescription'].apply(lambda
12                                     x: item in x.split(', '))].shape[0]
13     if support >= min_support:
14         frequent_patterns.append([item], support)
15
16 # Xây dựng cây Eclat
17 root = {'item': None, 'support': 0, 'children': {}}
18 for _, row in GrcsGroup_df.iterrows():
19     item_set = row['group_itemDescription'].split(', ')
20     current_node = root
21     for item in items:
22         if item in current_node['children']:
23             child = current_node['children'][item]
24             child['support'] += 1
25         else:
26             child = {'item': item, 'support': 1, 'children': {}}
27             current_node['children'][item] = child
28             if child['support'] >= min_support:
29                 current_node = child
30
31 # Tìm các mẫu phổ biến từ cây Eclat
32 def find_frequent_patterns(node, prefix):
33     frequent_patterns = []
34     if node['item'] is not None and node['support'] >= min_support:
35         frequent_patterns.append((prefix + [node['item']], node['support']))
36     for child in node['children'].values():
37         frequent_patterns.extend(find_frequent_patterns(child,
38                                                         prefix + [node['item']]))
39     return frequent_patterns
40 if root is None:
41     frequent_patterns = find_frequent_patterns(root, [])
42
43 # Kết thúc đo thời gian
44 end_time = time.time()
45
46 # Tính thời gian thực thi
47 execution_time_gr = end_time - start_time
```

Sau khi thực hiện khởi chạy là cả hai phương pháp xây dựng cây trên từng bộ dữ liệu, nhóm nhận thấy rằng, phương pháp có gom nhóm sản phẩm cho kết quả runtime nhanh hơn so với phương pháp không gom nhóm. Tuy điều này vẫn có nhiều chênh lệch trong nhiều lần thực nghiệm khi yếu tố thời gian này phụ thuộc khá nhiều vào tình trạng ổn định của đường truyền kernel, nhưng nhìn chung, bộ dữ liệu có ít thuộc tính mặt hàng quan sát hơn sẽ cho ra kết quả thời gian chạy tốt hơn so với bộ gốc.

```
1 # In kết quả ra màn hình
2 print("Thời gian thực thi của bộ dữ liệu không gom nhóm sản phẩm: %.2f giây"
3       % execution_time_ngr)
4 print("Thời gian thực thi của bộ dữ liệu có gom nhóm sản phẩm: %.2f giây"
5       % execution_time_gr)
```

Thời gian thực thi của bộ dữ liệu không gom nhóm sản phẩm: 2.94 giây
Thời gian thực thi của bộ dữ liệu có gom nhóm sản phẩm: 0.95 giây

- **Giải thích:**

Kết quả thời gian hiển thị cho thấy bộ dữ liệu có gom nhóm sản phẩm sẽ nhanh hơn so với bộ dữ liệu không gom nhóm đối với cả phương pháp xây dựng Cây tập hợp và phương pháp chạy hàm kết hợp bằng thư viện pyECLAT. Điều này có thể lý giải bởi các nguyên nhân như:

- + Bộ dữ liệu sau khi gom nhóm sẽ luôn có kích thước nhỏ hơn so với bộ dữ liệu không gom nhóm. Việc tối ưu hóa này có thể sẽ trả giá bằng việc mất mát thông tin bởi nó sẽ làm dữ liệu ít đi, tuy nhiên đổi lại, việc thao tác và tính toán trên tập dữ liệu mới này có thể sẽ diễn ra nhanh hơn, giúp giảm độ phức tạp toàn cục.
- + Bằng cách gom nhóm các sản phẩm có đặc điểm giống nhau thành các nhóm cụ thể, bộ dữ liệu mới sẽ giảm sự phân tán của dữ liệu và tạo ra các luật có ý nghĩa và dễ hiểu hơn vì chúng phản ánh sự tương tác giữa các nhóm sản phẩm.
- + Ngoài ra việc gom nhóm các sản phẩm có đặc điểm giống nhau bằng tri thức còn giúp giảm số lượng luật phát sinh. Khi sử dụng sản phẩm riêng lẻ cho tập dữ liệu, số lượng mẫu và luật phát sinh sẽ tăng một cách đáng kể, việc gom nhóm sẽ giúp quản lý kích thước dữ liệu và giúp tập dữ liệu mới có thể trở nên đơn giản hơn và tạo kết quả tổng quát hơn, đặc biệt khi nhóm chỉ quan tâm đến các mô hình và xu hướng chung của khách hàng.

Nếu kết quả thời gian thực hiện giữa hai bộ dữ liệu khác nhau, việc này có thể cung cấp thông tin quan trọng về hiệu suất của thuật toán và cách dữ liệu được tổ chức. Điều này có thể hữu ích để đưa ra quyết định về việc sử dụng hoặc tối ưu hóa phương pháp xử lý dữ liệu trong tình huống cụ thể.

2. Thay đổi thứ tự các mục (Cải thiện thời gian xử lý)

• Ý tưởng

Thứ tự là yếu tố quan trọng tác động đến tốc độ xử lý của thuật toán ECLAT. Bởi, cách hoạt động của giải thuật này là xây dựng một IT - Tree từ dữ liệu đầu vào để tìm các mẫu phổ biến. Trong quá trình xây dựng cây, thứ tự các mục sẽ được xem xét kỹ lưỡng và có ảnh hưởng đáng kể đến hiệu suất của thuật toán.

Khi IT - Tree được xây dựng, thuật toán ECLAT cần kiểm tra tất cả các tập con của các mẫu phổ biến. Nếu thứ tự của các mục không được tối ưu, điều này có thể dẫn đến việc tạo ra nhiều tập con không cần thiết, khiến quá trình xử lý mất nhiều thời gian và tài nguyên tính toán hơn.

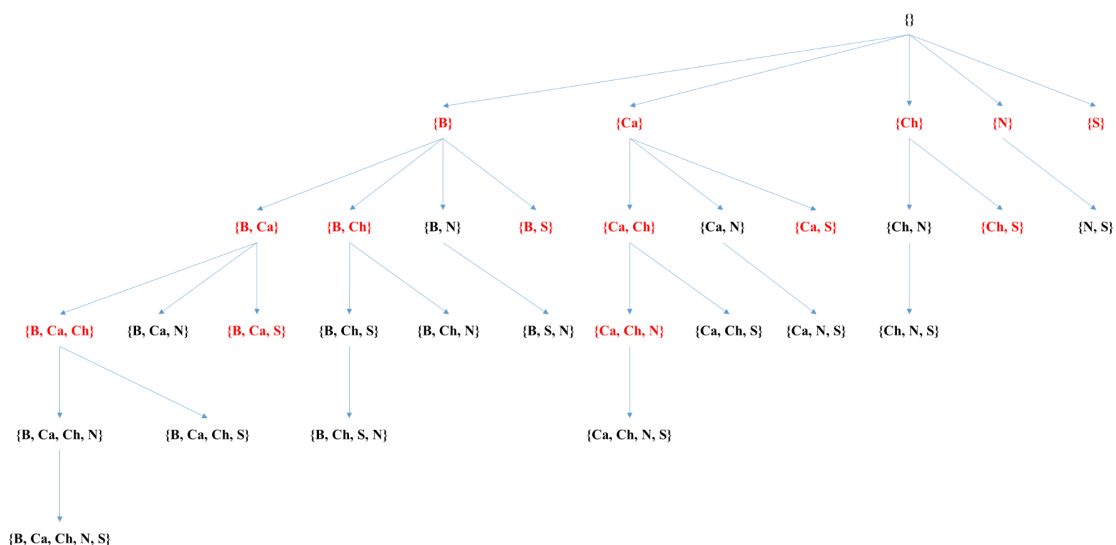
Để minh họa điều này, nhóm sẽ đưa ra tình huống cụ thể như sau:

Nếu nhóm thay đổi thứ tự của bộ dữ liệu từ alphabet thành thứ tự giảm dần của support, thuật toán ECLAT sẽ có thể tìm hiểu và xử lý các mẫu phổ biến có support cao trước sau đó giảm dần. Điều này có nghĩa là những mẫu phổ biến có support thấp hơn sẽ không được xem xét cho đến khi các mẫu có support cao đã được xử lý. Kết quả là, số lượng tập con cần duyệt qua có thể được giảm xuống, giúp tăng tốc độ xử lý của thuật toán.

Ví dụ minh họa: Lấy lại bộ dữ liệu minh họa của **Chương II**

Ở ví dụ trên, các vật phẩm đang sắp xếp tăng dần theo thứ tự alphabet, khi thực hiện xây dựng IT - Tree, nhóm thu được cây có tổng cộng **14** phần tử được tìm thấy:

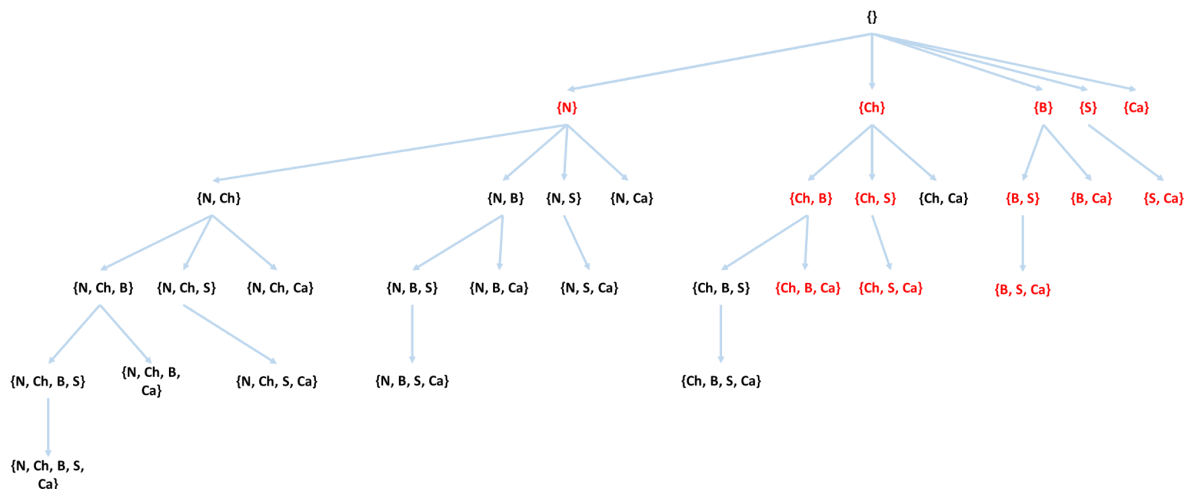
Bánh quy (B) → Cam (Ca) → Chanh (Ch) → Nước ngọt (N) → Sữa (S)



Tuy nhiên, sau khi sắp xếp các vật phẩm tăng dần theo thứ tự support tăng dần,

khi đó IT - Tree mà nhóm thu được chỉ có tổng cộng **13** phần tử được tìm thấy:

Nước ngọt (N) → Chanh (Ch) → Bánh quy (B) → Sữa (S) → Cam (Ca)



Từ ví dụ trên, nhóm thấy được rõ thứ tự của các mục trong thuật toán ECLAT có thể ảnh hưởng đáng kể đến hiệu suất xử lý. Thứ tự tối ưu sẽ giảm thiểu số lượng tập con không cần thiết cần duyệt qua và làm tăng tốc độ của thuật toán.

• Thực hiện & Kiểm thử

Sau khi tìm được mẫu phổ biến, nhóm sẽ thực hiện sắp xếp lại thứ tự các vật phẩm ứng với sự tăng dần của tần suất xuất hiện như sau:

```
# Tính toán support và tìm các mẫu phổ biến
total_samples = GrcsNonGroup_df.shape[0] # Tổng số mẫu trong dữ liệu
min_support_percent = 10 # Đặt minSup dưới dạng phần trăm
min_support = min_support_percent * total_samples / 100

frequent_patterns = []
for item in items_nongroup:
    support = GrcsNonGroup_df[GrcsNonGroup_df['itemDescription'].apply(lambda x: item in x.split(', '))].shape[0]
    if support >= min_support:
        frequent_patterns.append([item], support))

# Sắp xếp các mục theo thứ tự tăng dần về support
frequent_patterns = sorted(frequent_patterns, key=lambda x: x[1])
```

Thời gian thực thi của bộ dữ liệu ban đầu: 1.90 giây

Thời gian khởi chạy thuật toán khi thay đổi thứ tự toàn cục 1.62 giây

Tuy nhiên, sự khác biệt này không quá nhiều (thời gian chênh lệch chỉ **0.28s**), điều này có thể được giải thích bằng việc các mẫu phổ biến có support cao không được phân bố đều trong dữ liệu. Nghĩa là dù nhóm đã sắp xếp lại thứ tự theo support, các mẫu phổ biến với support cao vẫn tập trung ở một số nhóm dữ liệu cụ thể. Trong bộ

dữ liệu này, việc sắp xếp lại thứ tự có thể không đem lại lợi ích đáng kể về tốc độ xử lý, vì các mẫu phổ biến vẫn cần được xem xét lại và kiểm tra trong quá trình xây dựng IT - Tree.

3. Cải thiện bộ nhớ bằng biến đại diện

- Ý tưởng

DIFFSET (*Differential Set*) là một cải tiến được áp dụng trong thuật toán ECLAT để tối ưu hóa việc sử dụng bộ nhớ. ECLAT ban đầu có thể đối mặt với vấn đề về sử dụng bộ nhớ khi xử lý các tập dữ liệu lớn. DIFFSET được thiết kế để giảm bớt lượng bộ nhớ cần thiết để lưu trữ và xử lý các tập hợp dữ liệu tương đương.

Ý tưởng chính của DIFFSET là sử dụng biến đại diện (*Representative Items*) thay vì lưu trữ tất cả các mục trong mỗi cụm tương đương. Biến đại diện là một mục được chọn từ mỗi cụm tương đương để đại diện cho toàn bộ cụm. Bằng cách này, DIFFSET giảm bớt đáng kể lượng dữ liệu cần lưu trữ trong bộ nhớ.

Diffset để tính nhanh độ phổ biến

Diffset của A so với B, ký hiệu $d(AB)$ được định nghĩa như sau:

Gọi PA và PB là 2 nút thuộc lớp tương đương P , ta có:

- $d(PXY) = d(PY) \setminus d(PX)$
- $sup(PXY) = sup(PX) - |d(PXY)|$
- Diffset thường khá nhỏ so với Tidset

→ Chúng ta có thể sử dụng Diffset để thay thế Tidset.

Ví dụ: Cho CSDL ngang trong ví dụ trên với $minSup = 0.5$ như sau:

Giao dịch	Mục
T_1	A, B, D, E
T_2	B, C, E
T_3	A, B, D, E
T_4	A, B, C, E
T_5	A, B, C, D, E
T_6	B, C, D

B1. Quét CSDL chuyển sang CSDL theo chiều dọc như trong bảng sau:

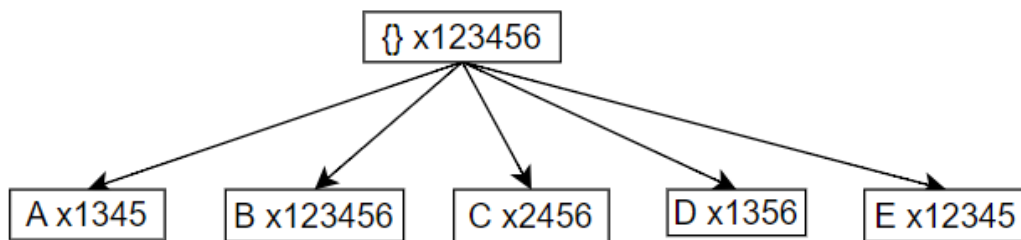
Mục	Tidset
-----	--------

A	1, 3, 4, 5
B	1, 2, 3, 4, 5, 6
C	2, 4, 5, 6
D	1, 3, 5, 6
E	1, 2, 3, 4, 5

B2 + B3. Xây dựng IT - Tree sử dụng Diffset với ngưỡng $minSup = 0.5$:

- **Mức 1 của IT - Tree với $minSup = 3$:**

$T(A) = 1345 \rightarrow sup(A) = 4$
 $T(B) = 123456 \rightarrow sup(B) = 6$
 $T(C) = 2456 \rightarrow sup(C) = 4$
 $T(D) = 1356 \rightarrow sup(D) = 4$
 $T(E) = 123456 \rightarrow sup(E) = 5$



- **Mức 2 của IT - Tree với $minSup = 3$:**

Nút A:

$$d(AB) = T(A) \setminus T(B) = \{1,3,4,5\} \setminus \{1,2,3,4,5,6\} = \emptyset$$

$$d(AD) = T(A) \setminus T(D) = \{1,3,4,5\} \setminus \{1,3,5,6\} = \{4\}$$

$$d(AE) = T(A) \setminus T(E) = \{1,3,4,5\} \setminus \{1,2,3,4,5\} = \emptyset$$

Nút B:

$$d(BC) = T(B) \setminus T(C) = \{1,2,3,4,5,6\} \setminus \{2,4,5,6\} = \{1,3\}$$

$$d(BD) = T(B) \setminus T(D) = \{1,2,3,4,5,6\} \setminus \{1,3,5,6\} = \{2,4\}$$

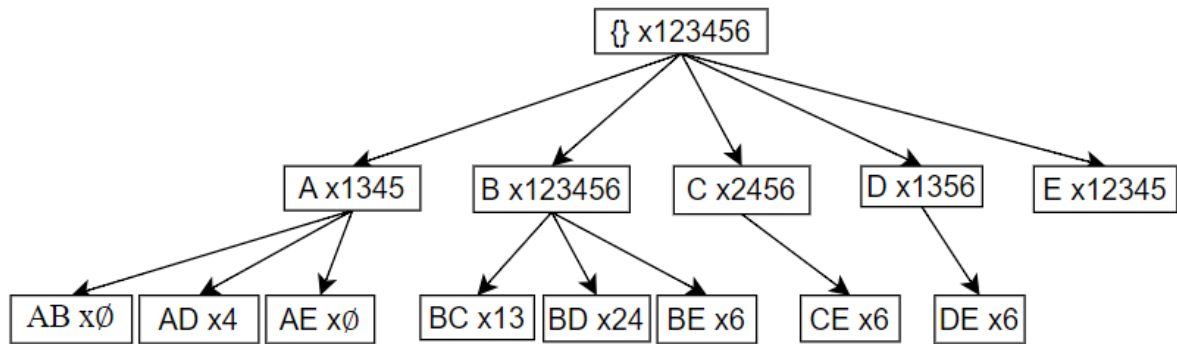
$$d(BE) = T(B) \setminus T(E) = \{1,2,3,4,5,6\} \setminus \{1,2,3,4,5\} = \{6\}$$

Nút C:

$$d(CE) = T(C) \setminus T(E) = \{2,4,5,6\} \setminus \{1,2,3,4,5\} = \{6\}$$

Nút D:

$$d(DE) = T(D) \setminus T(E) = \{1,3,5,6\} \setminus \{1,2,3,4,5\} = \{6\}$$



- **Mức 3 của IT - Tree với minSup = 3:**

Nút A:

$$d(ABD) = d(AD) \setminus d(AB) = \{4\} \setminus \emptyset = 4$$

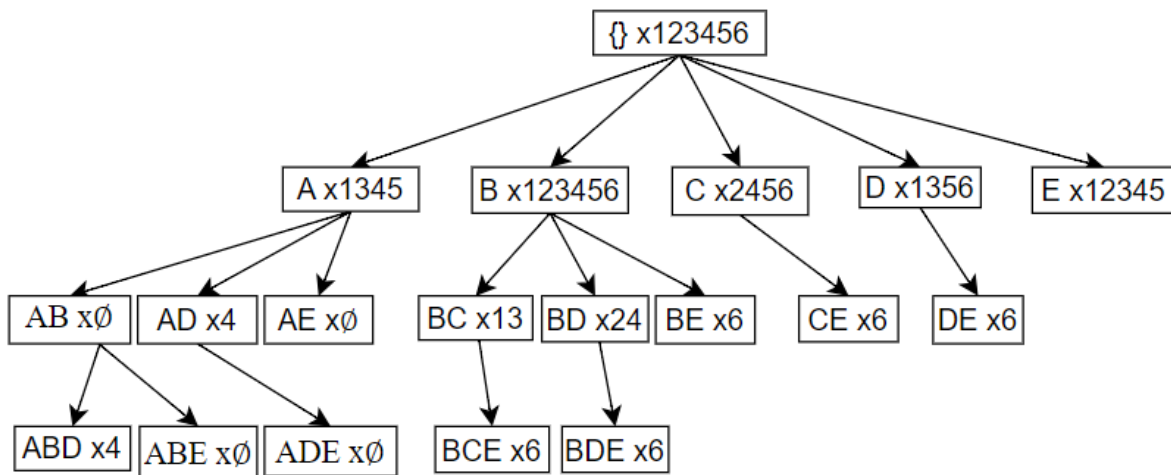
$$d(ABE) = T(AE) \setminus T(AB) = \emptyset \setminus \emptyset = \emptyset$$

$$d(ADE) = T(AE) \setminus T(AD) = \emptyset \setminus \{4\} = \emptyset$$

Nút B:

$$d(BCE) = d(BE) \setminus d(BC) = \{6\} \setminus \{1,3\} = \{6\}$$

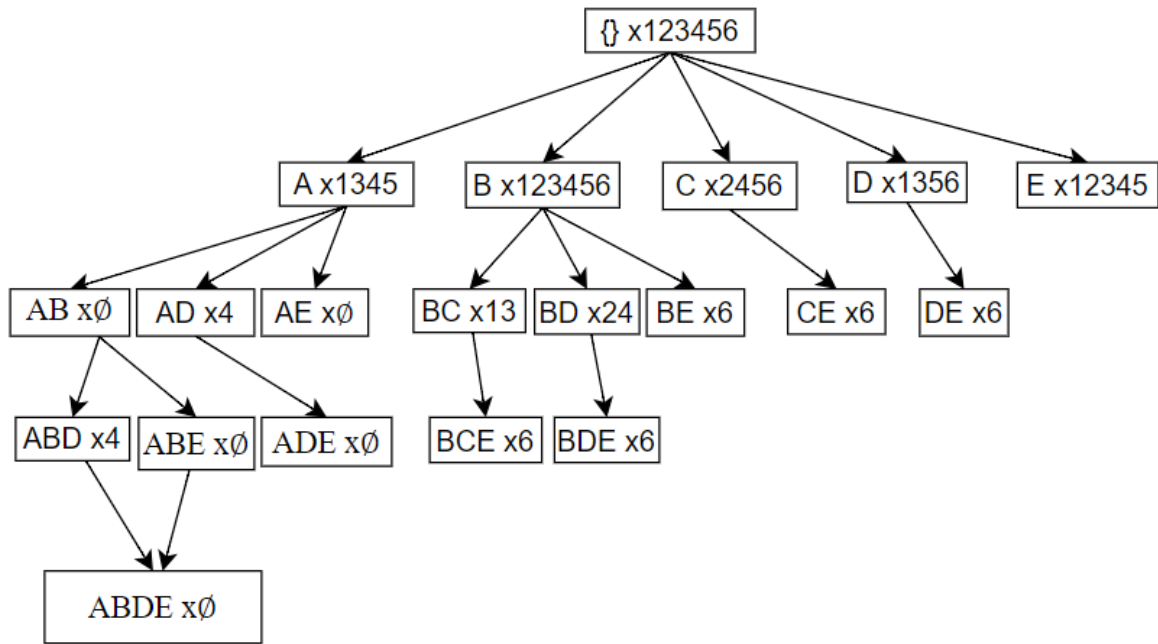
$$d(BDE) = d(BE) \setminus d(BD) = \{6\} \setminus \{2,4\} = \{6\}$$



- **Mức 4 của IT - Tree với minSup = 3:**

Nút A:

$$d(ABDE) = d(ABE) \setminus d(ABD) = \emptyset \setminus \{4\} = \emptyset$$



Vậy các tập phổ biến khai thác được là $\{A, B, C, D, E, AB, AD, AE, BC, BD, BE, DE, ABD, ABE, ADE, BCE, BDE, ABDE\}$ và tập phổ biến lớn nhất là $\{ABDE\}$.

• Thực hiện & Kiểm thử

Sau khi đã tìm hiểu thuật toán ECLAT sử dụng phương pháp Diffset để tìm mẫu phổ biến. Thì nhóm sẽ so sánh thời gian chạy giữa hai phương pháp Tidset và Diffset khi tìm mẫu phổ biến trên bộ dữ liệu *Groceries 1* (đã qua tiền xử lý) và thực hiện đo lường thời gian chạy trên cả hai phương pháp.

Trong quá trình tìm mẫu phổ biến, nhóm đã tính toán giá trị ngưỡng hỗ trợ dựa trên số tỷ lệ phần trăm của tổng số mẫu của bộ dữ liệu. Nhóm đã đặt giá trị cho tỉ lệ này là 0.05 và tính toán được $min_support$ bằng **693,85**.

1 min_support
693.85

Sau đó, nhóm tiến hành đo lường thời gian chạy giữa hai phương pháp *Diffset* và *Tidset*.

```

1 class DiffsetNode:
2     def __init__(self, item, tidlist):
3         self.item = item
4         self.tidlist = tidlist
5
6 class TidsetNode:
7     def __init__(self, item, tidset):
8         self.item = item
9         self.tidset = tidset
10
11 def generate_diffsets(database):
12     diffsets = []
13
14     for item, tidlist in database.items():
15         diffsets.append(DiffsetNode(item, tidlist))
16
17     return diffsets
18
19 def generate_tidsets(database):
20     tidsets = []
21
22     for item, tidset in database.items():
23         tidsets.append(TidsetNode(item, tidset))
24
25     return tidsets
26
27 def find_frequent_items_diffset(diffsets, min_support):
28     frequent_items = set()
29
30     for diffset in diffsets:
31         if len(diffset.tidlist) >= min_support:
32             frequent_items.add(diffset.item)
33
34     return frequent_items
35
36 def find_frequent_items_tidset(tidsets, min_support):
37     frequent_items = set()
38
39     for tidset in tidsets:
40         if len(tidset.tidset) >= min_support:
41             frequent_items.add(tidset.item)
42
43
44 # Đo lường thời gian chạy cho DIFFSET
45 start_time_diffset = time.time()
46 frequent_items_diffset = find_frequent_items_diffset(diffsets, min_support)
47 end_time_diffset = time.time()
48 elapsed_time_diffset = end_time_diffset - start_time_diffset
49
50 # Đo lường thời gian chạy cho TIDSET
51 start_time_tidset = time.time()
52 frequent_items_tidset = find_frequent_items_tidset(tidsets, min_support)
53 end_time_tidset = time.time()
54 elapsed_time_tidset = end_time_tidset - start_time_tidset

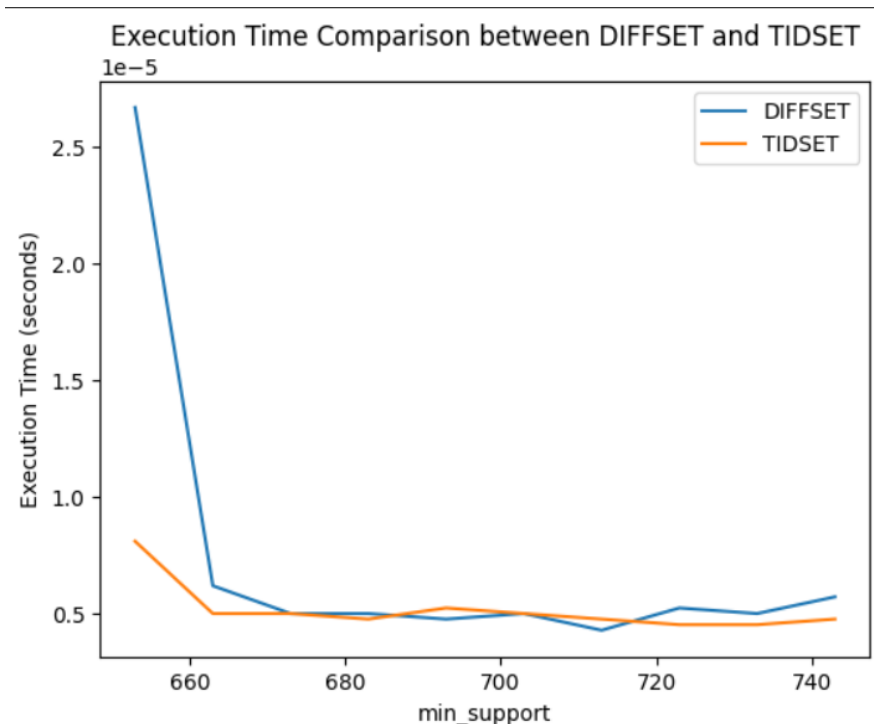
```

So sánh kết quả giữa hai phương pháp:

```
DIFFSET Execution Time: 0.00012493133544921875 seconds
TIDSET Execution Time: 5.53131103515625e-05 seconds
```

Ngoài ra, song song với việc đo lường thời gian chạy, nhóm còn thực hiện *grid_search* để thử nghiệm và đánh giá hiệu suất của mô hình với nhiều giá trị *min_support* khác nhau nhằm tìm ra giá trị *min_support* tối ưu nhất cho bài toán.

Sau khi tiến hành thực hiện đo lường và kiểm thử trên bộ dữ liệu thực tế, ta thu được kết quả như sau:



Nhận xét:

- Về thời gian thực hiện: Cả DIFFSET và TIDSET đều có thời gian thực hiện rất nhỏ, thường chỉ ở mức micro hoặc nanoseconds. Điều này có nghĩa là cả hai phương pháp đều rất hiệu quả và thực hiện nhanh chóng trên tập dữ liệu cụ thể này.
- So sánh giữa *Diffset* và *Tidset*: *Tidset* vẫn thường có thời gian thực hiện nhỏ hơn so với *DIFFSET* trong hầu hết các trường hợp. Điều này có thể phản ánh sự hiệu quả của cách biểu diễn dữ liệu *TIDSET*, giảm lượng thông tin cần xử lý và tăng hiệu suất tính toán.
- Tác động của *minsup* lên thời gian thực hiện: Cả *Diffset* và *Tidset* đều có xu hướng giảm thời gian chạy khi *minsup* tăng lên. Điều này có thể do việc tăng

min_support làm giảm số lượng mẫu phổ biến, từ đó làm giảm độ phức tạp tính toán.

- Sự ổn định của thời gian thực hiện: Thời gian thực hiện của hai phương pháp có vẻ ổn định khi mà *minsup* thay đổi. Các thời gian thực hiện không chênh lệch nhiều giữa các giá trị *minsup* khác nhau.
- Nhìn chung, cả DIFFSET và TIDSET đều cho thấy hiệu suất tốt trên tập dữ liệu này, và quyết định giữa hai phương pháp có thể dựa trên yêu cầu cụ thể của bài toán và đặc tính của dữ liệu.

4. So sánh tốc độ của ba thuật toán khai phá mẫu phổ biến

Để đánh giá hiệu quả của các thuật toán luật kết hợp đã học, nhóm dự định sẽ chạy thực nghiệm ba thuật toán này dựa trên tập dữ liệu, từ đó, nhóm có thể tiến hành so sánh bộ nhớ và thời gian chạy của từng thuật toán với nhau để đưa ra kết luận.

Nhưng có một thách thức là nếu muốn thực hiện những chỉ số đánh giá đó, các thuật toán cần được xây dựng trên cùng một thư viện hoặc cùng một phương pháp để việc so sánh có thể diễn ra công bằng.

Đối với thư viện **mlxtend**, thư viện này chỉ hỗ trợ cho thuật toán Apriori và FP-Growth, còn thư viện **pyECLAT** là một thư viện được xây dựng riêng nhằm đánh giá cho thuật toán ECLAT. Vậy nên phương pháp thực hiện đo lường bằng thời gian thực nghiệm có vẻ sẽ không khả thi.

Chính vì thế, nhóm sẽ tiến hành nghiên cứu độ phức tạp thời gian (time complexity) theo các quy tắc của **Big-O Notation** đã được học trong học phần Cấu trúc Dữ liệu và Giải thuật để đi tính toán cho các giải thuật luật kết hợp, từ đó, đưa ra những nhận định về cả ba thuật toán.

a. Apriori

Đối với Apriori, thuật toán này hoạt động dựa trên cơ sở dữ liệu nằm ngang (horizontal database) và sử dụng quy tắc tìm kiếm theo chiều rộng (BFS - Breadth First Search) để thực hiện. Nhóm tham khảo mã nguồn sau từ trang blog của **Devin Schumacher** về thuật toán Apriori.

```

1 import numpy as np
2 from itertools import combinations
3
4 def generate_frequent_itemsets(transactions, min_support):
5     """
6     Generate frequent itemsets from a list of transactions.
7
8     Args:
9         transactions: A list of transactions where each transaction is a list of items.
10        min_support: The minimum support threshold.
11
12    Returns:
13        A dictionary where the keys are itemsets and the values are the support counts.
14    """
15    item_counts = {}
16    for transaction in transactions:
17        for item in transaction:
18            if item in item_counts:
19                item_counts[item] += 1
20            else:
21                item_counts[item] = 1
22
23    # Filter out infrequent items
24    item_counts = {k: v for k, v in item_counts.items() if v >= min_support}
25
26    # Generate frequent itemsets
27    frequent_itemsets = {}
28    for k in range(2, len(item_counts) + 1):
29        for itemset in combinations(item_counts.keys(), k):
30            support_count = 0
31            for transaction in transactions:
32                if set(itemset).issubset(set(transaction)):
33                    support_count += 1
34            if support_count >= min_support:
35                frequent_itemsets[itemset] = support_count
36
37    return frequent_itemsets

```

```

1 def generate_association_rules(frequent_itemsets, min_confidence):
2     """
3     Generate association rules from frequent itemsets.
4
5     Args:
6         frequent_itemsets: A dictionary of frequent itemsets with their support counts.
7         min_confidence: The minimum confidence threshold.
8
9     Returns:
10        A list of association rules where each rule is a tuple of antecedent, consequent, and confidence.
11    """
12    association_rules = []
13    for itemset, support_count in frequent_itemsets.items():
14        for k in range(1, len(itemset)):
15            for antecedent in combinations(itemset, k):
16                antecedent = set(antecedent)
17                consequent = set(itemset) - antecedent
18                confidence = support_count / frequent_itemsets[tuple(antecedent)]
19                if confidence >= min_confidence:
20                    association_rules.append((antecedent, consequent, confidence))
21
22    return association_rules

```

Từ mã nguồn trên, nhóm thực hiện tính toán độ phức tạp thời gian dựa trên các bước và giai đoạn như sau:

- Giai đoạn 1: Tìm tập phổ biến **generate_frequent_itemsets**.

+ **Dòng 15-21: Tính tần suất xuất hiện của mỗi item.**

Ở bước đầu tiên, nhóm chạy vòng lặp qua từng transaction và các item trong giao dịch để tính số lần xuất hiện frequency của mỗi item. Độ phức tạp thời gian của bước này là $O(n * m)$ với n là số giao dịch transaction và m là số item duy nhất.

+ **Dòng 23-24: Loại bỏ các item không thỏa mãn ngưỡng hỗ trợ tối thiểu của bài toán.**

Bước thứ hai của thuật toán sẽ loại các item có ngưỡng hỗ trợ dưới ngưỡng minSup. Độ phức tạp ở bước này cho trường hợp tệ nhất là $O(m)$.

+ **Dòng 26-35: Tạo các tập phổ biến từ bộ dữ liệu.**

Bước cuối cùng của giai đoạn này, thuật toán thực hiện lặp qua các item để tạo các tập phổ biến mới có kích thước phần tử trong tập từ 2 đến m , độ phức tạp là $O(m * 2^m)$.

- **Giai đoạn 2: Tìm luật kết hợp generate_association_rules.**

+ **Dòng 52-53: Chạy vòng lặp kiểm tra từng tập phổ biến trong frequent_itemsets.**

Bước thứ tư, thuật toán chạy vòng lặp duyệt qua từng tập phổ biến để xem xét các phần tử trong mỗi tập, từ đó, xác định các tập con và tạo ra các luật kết hợp. Vậy thời gian duyệt qua từng phần tử trong mỗi tập là tỷ lệ với số lượng phần tử trong tập đó, nên độ phức tạp thời gian của bước này sẽ là $O(k)$ với k là số lượng tập phổ biến trong frequent_itemsets tìm được từ giai đoạn 1.

+ **Dòng 55-60: Chạy vòng lặp từng tập phổ biến con trong mỗi itemset.**

Bước cuối cùng của thuật toán sẽ duyệt vòng lặp qua tất cả các tập hợp con của từng tập phổ biến. Độ phức tạp thời gian của bước này là $O(2^k)$ với k là số lượng phần tử cần duyệt.

Vậy độ phức tạp thời gian tổng quát của thuật toán Apriori này là $O(n*m + m + m * 2^m + k + 2^k)$ hoặc có thể rút ngắn lại là $O(m * 2^m + k * 2^k)$ với m là số item unique, n là số giao dịch và k là số tập phổ biến tìm được từ giai đoạn 1.

b. ECLAT

ECLAT là thuật toán sử dụng cơ sở dữ liệu nằm dọc (vertical database) để khai thác các tập phổ biến, thuật toán này sử dụng quy tắc tìm kiếm theo chiều sâu (DFS - Depth First Search). Để tìm độ phức tạp thời gian, nhóm thực hiện tính

toán chỉ số dựa theo số vòng lặp của thuật toán. Mã nguồn sau đây được nhóm tham khảo từ trang blog của Devin Schumacher.

```
1 import numpy as np
2 from itertools import combinations
3
4 def eclat(dataset, min_support):
5     # Create a dictionary to store the support count for each item
6     item_support = {}
7     for transaction in dataset:
8         for item in transaction:
9             if item in item_support:
10                 item_support[item] += 1
11             else:
12                 item_support[item] = 1
13
14     # Prune the dictionary to only include items that meet the minimum support threshold
15     item_support = {k:v for k,v in item_support.items() if v >= min_support}
16
17     # Create a list of frequent items
18     frequent_items = list(item_support.keys())
19
20     # Create a list of itemsets
21     itemsets = []
22     for i in range(2, len(frequent_items) + 1):
23         itemsets += list(combinations(frequent_items, i))
24
25     # Create a dictionary to store the support count for each itemset
26     itemset_support = {}
27     for transaction in dataset:
28         for itemset in itemsets:
29             if set(itemset).issubset(set(transaction)):
30                 if itemset in itemset_support:
31                     itemset_support[itemset] += 1
32                 else:
33                     itemset_support[itemset] = 1
34
35     # Prune the dictionary to only include itemsets that meet the minimum support threshold
36     itemset_support = {k:v for k,v in itemset_support.items() if v >= min_support}
37
38     return itemset_support
```

Dựa trên mã nguồn này, nhóm triển khai tính toán độ phức tạp của thuật toán ECLAT theo từng phần của mã nguồn như sau:

- **Dòng 5-12: Tính support cho từng item.**

Bước đầu tiên là bước tính ngưỡng hỗ trợ của từng mặt hàng theo thuật toán. Ở bước này, thuật toán lặp qua từng giao dịch và đếm số lần xuất hiện của từng item. Độ phức tạp thời gian của bước này là $O(n*m)$ với:

- + **n**: số giao dịch transactions của từng item
- + **m**: số lượng item tối đa trong một giao dịch.

- **Dòng 14-15: Loại bỏ item không thỏa mãn ngưỡng minSup.**

Bước thứ hai, thuật toán duyệt vòng lặp từ item_support để loại bỏ các item không đạt ngưỡng hỗ trợ tối thiểu. Với m là số lượng item tối đa cần duyệt, độ phức tạp của bước này là $O(m)$.

- **Dòng 17-23: Tạo danh sách frequent items và itemsets.**

Bước thứ ba, thuật toán tạo danh sách các tập phổ biến từ các item thỏa ngưỡng hỗ trợ trong item_support, sau đó, cho lặp qua kích thước từ hai đến số lượng frequent items và tạo ra danh sách các itemset. Với m là số lượng item, độ phức tạp thời gian $O(2^m)$.

- **Dòng 25-33: Tính ngưỡng hỗ trợ cho từng luật.**

Ở bước thứ tư, thuật toán chạy vòng lặp qua từng transaction và các luật để đếm số lần xuất hiện của các item đó theo transaction. Độ phức tạp thời gian ở bước này sẽ phụ thuộc vào số lượng item m và số giao dịch transaction n: $O(n * 2^m)$.

- **Dòng 35-36: Tiếp tục thực hiện loại bỏ những luật kết hợp không thỏa mãn ngưỡng hỗ trợ tối thiểu.**

Bước cuối cùng, thuật toán chạy vòng lặp để loại bỏ các luật có ngưỡng hỗ trợ thấp hơn mức ngưỡng hỗ trợ của bài toán, độ phức tạp thời gian ở bước cuối cùng này có thể là $O(2^m)$ hoặc ít hơn, tùy thuộc vào số lượng itemset.

Vậy tổng thể, nhóm tìm được độ phức tạp thời gian cho giải thuật luật kết hợp ECLAT là $O(n * m + 2^m)$.

c. FP-Growth

Đối với FP-Growth, thuật toán này dựa trên cây để khai thác các tập phổ biến trong cơ sở dữ liệu, thuật toán này sử dụng phương pháp chia để trị (divide and conquer method) để tìm tập phổ biến mới từ chính cây FP thay vì từ tập phổ biến cũ. Độ phức tạp này sẽ phụ thuộc vào độ sâu của cây và cách tìm kiếm trên cây cho mỗi item trong cơ sở dữ liệu.

Nếu n là số item trong cơ sở dữ liệu thì độ sâu tối đa của cây tối đa cũng sẽ là n cho mỗi cây. Vậy thời gian lâu nhất để thuật toán hoàn thành sẽ được tính toán bằng cách lấy tổng số lượng item nhân với độ sâu tối đa của cây, vì thế nên độ phức tạp thời gian của thuật toán này sẽ là $O(n * n)$ hoặc $O(n^2)$.

d. Kết luận

Vậy qua phân tích, nhóm có thể rút ra được một số kết luận:

- Nhìn chung với cùng một bộ dữ liệu, thời gian chạy của Apriori sẽ lâu hơn so với thuật toán ECLAT, nhanh nhất sẽ là phương pháp duyệt cây của thuật toán FP-Growth.

- Apriori có độ phức tạp cao, đặc biệt là trong giai đoạn 2 với việc phải xử lý số lượng lớn các tập phổ biến. Ngoài ra, ECLAT cũng có độ phức tạp không nhỏ ở bước tạo danh sách frequent_items và itemsets. Trong khi đó, FP-Growth có độ phức tạp thấp hơn so với Apriori và ECLAT. Chi tiết hơn:
 - + Việc thực thi thuật toán Apriori sẽ mất nhiều thời gian hơn nhiều, điều này có thể là do tính chất của luật kết hợp này phải duyệt lại cơ sở dữ liệu mỗi khi tạo một tập phổ biến itemset mới.
 - + Thuật toán ECLAT có thời gian chạy nhanh hơn so với Apriori, nguyên nhân có thể là do tính chất của thuật toán này là tính toán frequency và support thông qua giao của các tập, vậy nên chỉ cần duyệt cơ sở dữ liệu với số ít lần.
 - + Trong ba thuật toán, thời gian chạy của FP-Growth sẽ là ngắn nhất. Lý do là vì thuật toán này chỉ cần duyệt cơ sở dữ liệu một lần để xây dựng cây FP-Tree, tính chất này có thể là lợi thế lớn do không cần phải tạo các tập ứng viên.

CHƯƠNG IX. KẾT LUẬN

Sau quá trình thực hiện bài đồ án nghiên cứu về thuật toán khai phá luật kết hợp ECLAT, nhóm nghiên cứu đã hoàn thành việc xây dựng và đánh giá hiệu quả của thuật toán. Đồng thời, nhóm cũng đã đề xuất một số phương pháp cải tiến nhằm tối ưu hóa khả năng giải thuật của thuật toán. Dưới đây là những đánh giá của nhóm nghiên cứu:

Ưu điểm của đồ án nghiên cứu:

- Nhóm nghiên cứu đã tiếp cận thuật toán ECLAT từ nhiều góc độ khác nhau, nhằm tìm ra phương pháp hiệu quả nhất để xây dựng thuật toán một cách tối ưu.
- Nhóm đã thực hiện tốt việc chuẩn bị dữ liệu thông qua nhiều phương pháp tiền xử lý hiệu quả, giúp việc xây dựng thuật toán trở nên dễ dàng hơn.
- Việc đánh giá và nghiên cứu các phương pháp cải thiện, cùng với việc so sánh các thuật toán khai phá khác thông qua các yếu tố thời gian thực hiện và độ phức tạp, đã giúp nhóm có cái nhìn tổng quan hơn về hiệu suất hoạt động của thuật toán.

Nhược điểm và hướng cải thiện:

- Mặc dù đã nỗ lực, nhóm nghiên cứu vẫn chưa thể xây dựng thuật toán một cách tối ưu nhất do thiếu kinh nghiệm và kiến thức, đặc biệt là khi đây là lần đầu tiên nhóm nghiên cứu tìm hiểu về thuật toán ECLAT.
- Nhóm cũng chưa thể đưa ra nhiều phương pháp cải thiện thuật toán do thời gian thực hiện nghiên cứu có hạn.

Để nâng cao chất lượng của bài đồ án, nhóm cần phải nghiên cứu sâu hơn về cách thức hoạt động và tổ chức dữ liệu của thuật toán. Nhóm sẽ tiếp tục cố gắng và không ngừng nỗ lực để hoàn thiện hơn trong tương lai.

PHỤ LỤC

1. Mã nguồn

- Link Github: [Neyung/NAT_DM_GROUP6](#)
- Link nguồn:

Tiền xử lý dữ liệu: [1-Preprocessing.ipynb - Colaboratory](#)

Xây dựng & Cải thiện & Đánh giá: [2_AssociationRule.ipynb - Colaboratory](#)

2. Bảng phân công

THÀNH VIÊN	PHÂN CÔNG	ĐÁNH GIÁ
Trần Phạm Hải Nam	Tìm hiểu thuật toán, Xây dựng thuật toán, Cải thiện thuật toán bằng cách gom nhóm sản phẩm, Đánh giá độ phức tạp thời gian.	100%
Lý Minh Nguyên	Tìm hiểu thuật toán, Tiền xử lý dữ liệu, Tổng quan dữ liệu, Viết kết luận.	100%
Võ Minh Nguyên	Tìm hiểu thuật toán, Minh họa bài toán qua các bước, Xây dựng thuật toán, Tối ưu hóa thuật toán bằng thứ tự các mục, Tổng hợp nội dung báo cáo.	100%
Nguyễn Thị Ngọc Nhi	Tìm hiểu thuật toán, Trình bày khái niệm liên quan, Ví dụ minh họa cho thuật toán, Tối ưu hóa thuật toán bằng biến đại diện.	100%
Lê Thành Phát	Tìm hiểu thuật toán, Tổng quan đề tài.	80%

3. Tài liệu tham khảo

[1] Nguyen An Te, “Data Mining.2023.Ch4 - Luật kết hợp”, UEH, 2023

[2] Nguyen Duy Ham, “Phát triển một số thuật toán hiệu quả khai thác tập mục trên cơ sở dữ liệu số lượng có sự phân cấp các mục”, 2016

Link: https://repository.vnu.edu.vn/bitstream/VNU_123/74893/1/01050003160.pdf

[3] PGS.TS. Vo Dinh Bay, “Khai thác luật kết hợp”

Link: <https://www.fit.uet.vnu.edu.vn/dmss2016/files/lecture/VoDinhBay-Lecture.pdf>

[4] [Apriori \(serp.ai\)](#)

[5] [ECLAT \(serp.ai\)](#)

[6] [Frequent Pattern Mining Algorithms for Finding Associated Frequent Patterns for Data Streams: A Survey - ScienceDirect](#)

[7] [How to Analyze the Complexity of Pattern Mining Algorithms ? | The Data Blog \(philippe-fournier-viger.com\)](#)

[8] [\(PDF\) Comparing the Performance of Frequent Pattern Mining Algorithms \(researchgate.net\)](#)

[9] [5. Time Complexity Of A Priori And Evolutionary Algorithm For Numerical Association Rule Mining Optimization.pdf \(unmul.ac.id\)](#)

[10] [ch6.pdf \(umn.edu\)](#)

[11] [pyECLAT/pyECLAT/pyECLAT.py at master · jeffrichardchemistry/pyECLAT \(github.com\)](#)