BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ TP. HỒ CHÍ MINH

ĐỒ ÁN MÔN HỌC

KHAI PHÁ DỮ LIỆU

ÁP DỤNG THUẬT TOÁN ECLAT CHO BỘ DỮ LIỆU GROCERIES 2

Họ và tên	MSSV	Lớp
Vũ Nguyễn Thảo Vi	31211027686	DS001
Nguyễn Quốc Việt	31211027687	DS001
Bùi Quốc Việt	35221020290	DS001
Nguyễn Thanh Vy	31211027689	DS002
Nguyễn Nhật Thảo Vy	31211025542	DS001

GVHD: TS. Nguyễn An Tế

TP. Hồ Chí Minh, Ngày 15 tháng 12 năm 2023

Mục lục

LỜI MỞ ĐẦU

Trong thời đại kỹ thuật số hiện nay, con người không chỉ sản sinh ra mà còn thu thập một lượng lớn dữ liệu chưa từng có tiền lệ. Dữ liệu này mở ra cơ hội để hiểu biết sâu sắc hơn về hành vi và nhu cầu của người tiêu dùng, tạo điều kiện cho các doanh nghiệp phát triển chiến lược kinh doanh mới. Việc chuyển hóa dữ liệu thành thông tin và sau đó là tri thức giúp củng cố nền tảng quyết định cho doanh nghiệp. Ngày nay, các doanh nghiệp đang tích cực sử dụng thông tin và hiểu biết từ dữ liệu để liên tục cải thiện chiến lược bán hàng và tiếp thị của mình.

Trong bối cảnh đó, thuật toán ECLAT trong Khai phá Dữ liệu trở nên rất quan trọng, phục vụ như một công cụ mạnh mẽ giúp chúng ta khai thác dữ liệu một cách hiệu quả. ECLAT là một kỹ thuật phổ biến trong khám phá quy tắc kết hợp và tập trung vào việc tìm kiếm mối liên hệ giữa các mặt hàng trong cơ sở dữ liệu lớn, giúp doanh nghiệp hiểu rõ hơn về thói quen và hành vi mua hàng của khách hàng.

Theo đó, trong đề tài KTHP môn học Khai Phá Dữ Liệu, nhóm của chúng em đã áp dụng thuật toán ECLAT vào dự án phân tích dữ liệu khách hàng từ bộ dữ liệu Groceries. Trong quá trình làm đồ án, chúng tôi đã gặp phải một số thách thức và hạn chế, và rất mong nhận được phản hồi từ thầy để cải thiện hơn nữa.

Dưới sự hướng dẫn của Thầy Nguyễn An Tế, chúng em đã học hỏi được nhiều điều giá trị về Khai phá Dữ liệu. Thầy đã truyền đạt kiến thức một cách dễ hiểu và gần gũi, giúp chúng em tiếp cận với những khái niệm phức tạp một cách dễ hiểu. Chúng em xin gửi lời cảm ơn chân thành đến Thầy vì đã không ngừng hỗ trợ và dẫn dắt chúng em trên con đường học thuật này.

1 CHƯƠNG 1: TỔNG QUAN

1.1 Vai trò và ý nghĩa của Khai phá dữ liệu

Ngày nay, dữ liệu là thành phần quan trọng trong chuyển đổi số cũng như các ngành nghề khoa học, kỹ thuật. Khai thác dữ liệu từ đó trở thành một phần quan trọng trong quá trình quản lý thông tin và đưa ra quyết định trong nhiều lĩnh vực. Lấy ví dụ trong thương mại điện tử nói chung và quản lý hàng hóa nói riêng, việc hiểu rõ mối quan hệ giữa các sản phẩm có thể mang lại nhiều lợi ích về chiến lược chiến dịch và tối ưu hóa dịch vụ cũng như quy trình kinh doanh.

1.2 Giới thiệu đề tài

Mục tiêu của đề tài là tìm hiểu về cơ chế hoạt động và cấu trúc của thuật toán ECLAT và áp dụng nó vào một bộ dữ liệu Groceries 2. Đề tài cũng thực hiện so sánh đánh giá với thuật toán Apriori - một trong những thuật toán khai thác luật kết hợp rất phổ biến để cái nhìn rõ ràng hơn về những ưu và nhược điểm của ECLAT so với Apriori.

2 CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Luật kết hợp

Luật kết hợp *(Association rule)* là mối quan hệ kết hợp giữa các tập thuộc tính trong cơ sở dữ liệu và là công cụ hữu ích để khám phá các mối liên kết trong dữ liệu. Một luật kết hợp là một mệnh đề kéo theo có dạng $X \to Y$, trong đó $X, Y \subseteq I$, đồng thời thỏa mãn điều kiện $X \cap Y = \varnothing$.

Một vài định nghĩa:

Items: $I = \{I_1, I_2, \ldots, I_n\}$

Itemset: $X \subseteq I$

k-itemset: $X_k = \{I_1, I_2, \ldots, I_k\} \quad k \leq n$

Transaction: $\varnothing \neq T \subseteq I$

Quy ước: items trong T được sắp xếp tăng dần

Database: $D = \{T_i\}$

Association rule: $X \Rightarrow Y \quad \varnothing \neq X \subset I \quad \varnothing \neq Y \subset I \quad X \cap Y = \varnothing$

Tần số *(frequency)* của itemset X là số lần xuất hiện của X trong các giao dịch của D và được định nghĩa như sau:

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

Độ hỗ trợ *(support)* của $X \Rightarrow Y$ được định nghĩa như sau:

$$\sup(X \Rightarrow Y) = P(X \cup Y) = \frac{\text{freq}(X \cup Y)}{|D|}$$

Độ tin cậy *(confidence)* của $X \Rightarrow Y$ là độ chính xác của luật và được định nghĩa như sau:

$$\text{conf}(X \Rightarrow Y) = P(Y \mid X) = \frac{\text{freq}(X \cup Y)}{\text{freq}(X)}$$

2.2 Cơ sở lý thuyết thuật toán Apriori

Thuật toán Apriori Agrawal and Srikant (1994) là một phương pháp cổ điển trong khai phá dữ liệu, được sử dụng để phát hiện các tập mục *(itemsets)* phổ biến và luật kết hợp trong một cơ sở dữ liệu chứa các giao dịch.

Quá trình này diễn ra qua hai giai đoạn chính. Đầu tiên là khai phá các tập phổ biến, trong đó thuật toán bắt đầu bằng việc xác định các tập mục phổ biến *(frequent itemsets)* trong cơ sở dữ liệu. Điều này được thực hiện bằng cách sử dụng phương pháp tìm kiếm từng cấp *(level-wise search)*. Trong bước này, từ các tập mục đơn (1-itemsets) phổ biến, thuật toán mở rộng dần để tìm các tập mục 2-itemsets, 3-itemsets, v.v., cho đến khi không còn tìm thấy tập mục phổ biến mới nào. Mỗi tập mục phổ biến được xác định dựa trên ngưỡng tần suất tối thiểu (minSup).

Sau khi đã xác định được các tập mục phổ biến, thuật toán tiến hành tạo các luật kết hợp từ những tập này. Một luật kết hợp được xem là mạnh nếu nó đáp ứng điều kiện về độ tin cậy tối thiểu *(minConf)*. Để tạo ra một luật kết hợp, thuật toán xem xét tất cả các tập con không rỗng của một tập mục phổ biến, và từ mỗi tập con này, tạo ra một luật kết hợp với phần còn lại của tập mục. Phương pháp tạo các tập mục phổ biến (L_k) từ tập mục phổ biến cấp trước đó (L_{k-1}) bao gồm hai bước chính. Đầu tiên là bước nối kết (join step), trong đó ta sẽ tạo ra các tập ứng viên cho tập mục k-itemset từ các tập mục $(k-1)$-itemset phổ biến bằng cách kết hợp chúng với nhau. Tiếp theo là bước cắt tỉa *(prune step)*. Ở bước này, ta sẽ loại bỏ các tập ứng viên không phổ biến, áp dụng tính chất Apriori để giảm kích thước tìm kiếm và tối ưu hóa hiệu suất.

Tóm lại, thuật toán Apriori là một quy trình hai bước, bao gồm việc xác định các tập mục phổ biến và từ đó tạo ra các luật kết hợp mạnh, dựa trên các tiêu chí về tần suất và độ tin cậy.

Thuật toán Apriori xử lý dữ liệu được biểu diễn theo chiều ngang (horizontal data layout), có nghĩa là danh sách các mặt hàng trong giao dịch được liệt kê theo thứ tự từ trái sang phải trên cùng một dòng giao dịch. Vì thế, trong mỗi lần chạy, thuật toán phải duyệt lại toàn bộ dữ liệu xem mặt hàng đó có trong những giao dịch nào. Dữ liệu biểu diễn theo chiều ngang được mô tả như bảng sau:

Tid	**Items**
T1	A,C,D
T2	B,C,E
T3	A,B,C,E
T4	B,E

2.3 Cơ sở lý thuyết thuật toán ECLAT

ECLAT (viết tắt của **E**quivalence **CLA**ss **T**ransformation) là một thuật toán để xác định luật kết hợp trong cơ sở dữ liệu, được phát triển bởi Zaki (2000). Khác với thuật toán Apriori, ECLAT xử lý dữ liệu được biểu diễn theo chiều dọc (vertical data layout): item - transactions, có nghĩa là ứng với từng đồ vật sẽ là các giao dịch chứa đồ vật đó. Điều này giúp thuật toán ECLAT không cần xét toàn bộ dữ liệu gốc cho mỗi lần tính toán, chỉ cần kết hợp các giao dịch để tạo ra một lớp dữ liệu mới. Dữ liệu biểu diễn theo chiều dọc được mô tả như bảng sau:

Item	**Tid**
A	T1, T2
B	T2, T3, T4
C	T1, T2, T3
D	T1
E	T2, T3, T4

Việc này giúp thuật toán hoạt động nhanh hơn Apriori. Cụ thể, do số lượng itemset ứng viên *(candidate)* được sinh ra từ thuật toán Apriori tăng theo cấp số nhân, vì vậy, với dữ liệu lớn, Apriori trở nên kém hiệu quả khi phải quét đi quét lại toàn bộ cơ sở dữ liệu để đếm số lần xuất hiện, khiến cho chi phí đọc ghi và chi phí lưu trữ các itemset tạm thời trở nên tốn kém. Thuật toán ECLAT là một trong những thuật toán khai phá luật kết hợp, được cải tiến từ thuật toán Apriori. Thay vì duyệt nhiều lần cơ sở dữ liệu, thuật toán ECLAT chỉ duyệt cơ sở dữ liệu một lần.

Ý tưởng của thuật toán ECLAT là tìm các *itemsets* có *Transaction Id Sets (tidsets)* giao nhau phải thỏa điều kiện độ hỗ trợ *(support)* lớn hơn hoặc bằng ngưỡng *minSup* cho

trước. Độ hỗ trợ được tính bằng cách lấy tỷ lệ giữa số lần xuất hiện đồng thời của X và Y trên tổng số giao dịch.

Các bước thực hiện thuật toán ECLAT

Cho cơ sở dữ liệu sau, với $minSup = 40\%$ => $minFreq = 2$:

Tid	Items
T1	A,C,D
T2	B,C,E
T3	A,B,C,E
T4	B,E

Bước 1: Trong lần duyệt đầu tiên, thuật toán sẽ duyệt qua toàn bộ cơ sở dữ liệu để tìm tidsets của từng item riêng lẻ. Đồng thời, chuyển định dạng dữ liệu từ chiều ngang sang chiều dọc.

Itemset	Tidset
{A}	{T1, T2}
{B}	{T2, T3, T4}
{C}	{T1, T2, T3}
{D}	{T1}
{E}	{T2, T3, T4}

Bước 2: Sau đó, thuật toán sẽ tiếp tục tìm các 2-itemsets gồm các cặp item có *tidsets* giao nhau khác rỗng. 2-itemsets phải được kết hợp từ 2 item có $minFreq \geq 2$

Itemset	Tidset
{A,B}	{T2}
{A,C}	{T1, T2}
{A,E}	{T2}
{B, C}	{T2, T3}
{B, E}	{T2, T3, T4}
{C, E}	{T2, T3}

Bước 3: Tìm 3-itemsets dựa trên việc kết hợp hai 2-itemsets có $minFreq \geq 2$ và có item đầu tiên giống nhau. Mỗi 3-itemset gồm 3 items, có *tidsets* được giao nhau từ hai 2-itemsets. Giữ lại các 3-itemsets có tidsets giao nhau khác rỗng.

Itemset	Tidset
{B, C, E}	{T2, T3}

Bước 4: Kết thúc tại 3-itemset vì không thể tiếp tục kết hợp. Vì vậy, tập phổ biến gồm:

Frequent Itemset
{A, C}
{B, C}
{B, E}
{C, E}
{B, C, E}

Bước 5: Tạo các luật kết hợp mạnh từ các tập phổ biến, với $[minSup, minConf] = (40\%, 70\%)$

Rule	[Sup, Conf]
A $\rightarrow$ C	[50%,100%]
C $\rightarrow$ A	[50%,66.67%]
B $\rightarrow$ C	[50%,66.67%]
C $\rightarrow$ B	[50%,66.67%]
B $\rightarrow$ E	[66.67%,100%]
E $\rightarrow$ B	[66.67%,100%]
C $\rightarrow$ E	[50%,66.67%]
E $\rightarrow$ C	[50%,66.67%]
BC $\rightarrow$ E	[50%,100%]
BE $\rightarrow$ C	[50%,66.67%]
CE $\rightarrow$ B	[50%,100%]
B $\rightarrow$ CE	[50%,66.67%]
C $\rightarrow$ BE	[50%,66.67%]
E $\rightarrow$ BC	[50%,66.67%]

Vậy, ta sẽ có 5 luật kết hợp mạnh là:

- A $\rightarrow$ C
- B $\rightarrow$ E
- E $\rightarrow$ B

- BC $\rightarrow$ E
- CE $\rightarrow$ B

2.4 So sánh thuật toán ECLAT và Apriori

2.4.1 Sự giống nhau giữa thuật toán ECLAT và Apriori

Trong lĩnh vực Khai phá dữ liệu, cả hai thuật toán ECLAT và Apriori đều được sử dụng để tìm ra mối quan hệ kết hợp giữa các mục (items) trong dữ liệu. Thêm vào đó, cả hai thuật toán đều không xem xét thứ tự của các mặt hàng trong mỗi giao dịch, nghĩa là thứ tự xuất hiện của các mặt hàng trong mỗi giao dịch không quan trọng trong quá trình phân tích dữ liệu.

Ngoài ra, cả hai thuật toán này đều tạo ra các tập ứng viên (candidate) trong quá trình thực hiện. Trong đó, hai thuật toán sẽ duyệt qua các nhóm item khác nhau và đánh giá dựa trên các chỉ số cho trước như minSup. Chỉ những itemset thoả mãn điều kiện minSup mới được xem là tập phổ biến. Ngoài ra, một điểm giống nhau khác của hai thuật toán là bước cắt tỉa (pruning). Theo đó, một tính chất được tận dụng là việc nếu một tập cha là phổ biến thì tập con cũng là một tập phổ biến và ngược lại. Nhờ vào tính chất này, lượng tính toán sẽ được giảm bớt đi một cách đáng kể.

2.4.2 Sự khác nhau giữa thuật toán ECLAT và Apriori

Bảng 1 So sánh các yếu tố bao gồm nguyên lý, độ phức tạp, kích thước bộ nhớ và database của thuật toán Apriori và ECLAT.

	Apriori	**ECLAT**
Độ phức tạp	$O(2^n \times m)$	$O(m \times n \log n)$
Thời gian thực thi	Thời gian thực thi không tối ưu vì liên tục lặp lại việc duyệt qua toàn bộ CSDL	Về lý thuyết, thời gian thực thi nhanh hơn vì chỉ cần duyệt qua CSDL 1 lần
Memory cost	Tốn bộ nhớ để lưu các tập itemset trong mỗi lần duyệt CSDL	Tốn bộ nhớ hơn để lưu trữ nếu số lượng giao dịch lớn
Data Layout	Theo cấu trục bảng ngang	Theo cấu trục bảng dọc

3 CHƯƠNG 3: ÁP DỤNG THUẬT TOÁN

3.1 Áp dụng thuật toán ECLAT cho bộ dữ liệu Groceries 2

Đầu tiên ta sẽ đọc bộ dữ liệu như sau:

```
with open("Groceries 2.csv", 'r') as temp_f:
    col_count = [ len(l.split(",")) for l in temp_f.readlines() ]
```

```
column_names = [i for i in range(0, max(col_count))]
df = pd.read_csv("Groceries 2.csv", header=None, delimiter=",",
    names=column_names)
```

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	29	30	31
0	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Sử dụng thư viện pyECLAT để tìm ra luật kết hợp từ bộ dữ liệu bằng thuật toán ECLAT. Sau đó chuyển đổi dataframe về dạng nhị phân. Trong đó, mỗi hàng là một giao dịch, mỗi cột đại diện cho 1 item. Giá trị của ô bằng 1 cho biết item tương ứng có mặt trong giao dịch và ngược lại, giá trị 0 có nghĩa là item không có trong giao dịch. Việc chuyển đổi sang dạng nhị phân giúp các bước xử lý, phân tích và tính toán ở phần sau đơn giản hơn và cải thiện tốc độ tính toán cho thuật toán.

```
!pip install pyECLAT
```

```
from pyECLAT import ECLAT
eclat = ECLAT(data=df)

# Chuyển về binary dataframe
binary_df = eclat.df_bin
print("Binary DataFrame:")
print(binary_df[:5])
```

```
Binary DataFrame:
   dish cleaner  bathroom cleaner  brandy  jam  UHT-milk  liquor  pork  \
0             0                 0       0    0         0       0     0
1             0                 0       0    0         0       0     0
2             0                 0       0    0         0       0     0
3             0                 0       0    0         0       0     0
4             0                 0       0    0         0       0     0

   nut snack  kitchen utensil  zwieback  ...  instant coffee  flower (seeds)  \
0          0                0         0  ...               0               0
1          0                0         0  ...               0               0
2          0                0         0  ...               0               0
3          0                0         0  ...               0               0
4          0                0         0  ...               0               0

   herbs  baby food  pudding powder  misc. beverages  specialty chocolate  \
0      0          0               0                0                    0
1      0          0               0                0                    0
2      0          0               0                0                    0
3      0          0               0                0                    0
4      0          0               0                0                    0

   soda  canned fruit  sugar
0     0             0      0
1     0             0      0
2     0             0      0
3     0             0      0
4     0             0      0

[5 rows x 169 columns]
```

Sau đó xem qua các số lượng bán ra của các mặt hàng trong tất cả các giao dịch, và số lượng các mặt hàng trong mỗi giao dịch.

```
#Xem số lượng bán ra của các món hàng
items_total = eclat.df_bin.astype(int).sum(axis=0)
items_total
```

```
salt                106
dessert             365
brandy               41
cooking chocolate    25
rice                 75
                   ...
cleaner              50
hamburger meat      327
coffee              571
hair spray           11
zwieback             68
Length: 169, dtype: int64
```

```
#Xem số lượng món hàng được mua bởi mỗi transaction
items_per_transaction = eclat.df_bin.astype(int).sum(axis=1)
items_per_transaction
```

```
0        4
1        3
2        1
3        4
4        4
       ..
9830    17
9831     1
9832    10
9833     4
9834     5
Length: 9835, dtype: int64
```

Tiếp theo, áp dụng thuật toán ECLAT với ngưỡng giá trị minSup = 0.03 và min_combination = 2. Min_combination = 2 vì một luật kết hợp có ý nghĩa nên có tối thiểu 2 items. Ngưỡng minSup nhóm chỉ định cho thuật toán tương đối nhỏ, vì số lượng giao dịch của bộ dữ liệu tương đối lớn (hơn 9800 giao dịch), nếu chọn ngưỡng minSup lớn hơn sẽ không thể tìm được itemset thỏa điều kiện. Với minSub = 0.03, các luật phổ biến có tần suất xuất hiện lên đến gần 300 giao dịch, do đó các luật này vẫn sẽ mang đến thông tin hữu ích.

```
#Áp dụng thuật toán ECLAT tìm frequent itemsets với minSup = 0.03
get_ECLAT_indexes, get_ECLAT_supports = eclat.fit_all(min_support=0.03,
                                                      min_combination=1,
                                                      ↪ separator=', ')
#Áp dụng thuật toán ECLAT tìm frequent itemsets với minSup = 0.03
get_ECLAT_indexes, get_ECLAT_supports = eclat.fit_all(min_support=0.03,
                                                      min_combination=1,
                                                      ↪ separator=', ')
```

Thời gian chạy thuật toán:

```
time: 1min 53s (started: 2023-12-15 16:51:20 +00:00)
```

Sau đó xem các tập phổ biến (frequent itemset) được xác định bởi thuật toán. Các tập phổ biến là tập luật có support $\geq$ minSup.

```
# Tạo dataframe chứa frequent itemsets
frequent_itemsets_eclat = pd.DataFrame({'support':
    get_ECLAT_supports.values(),'itemsets': get_ECLAT_supports.keys()})
frequent_itemsets_eclat = frequent_itemsets_eclat.sort_values(by='support',
    ascending=False)
frequent_itemsets_eclat
```

	support	itemsets
12	0.074835	other vegetables, whole milk
8	0.056634	rolls/buns, whole milk
14	0.056024	yogurt, whole milk
4	0.048907	root vegetables, whole milk
3	0.047382	root vegetables, other vegetables
11	0.043416	other vegetables, yogurt
5	0.042603	rolls/buns, other vegetables
2	0.042298	tropical fruit, whole milk
18	0.040061	whole milk, soda
9	0.038332	rolls/buns, soda
1	0.035892	tropical fruit, other vegetables
6	0.034367	rolls/buns, yogurt
15	0.034367	bottled water, whole milk
17	0.033249	whole milk, pastry
13	0.032740	other vegetables, soda
16	0.032232	whole milk, whipped/sour cream
7	0.030605	rolls/buns, sausage
10	0.030503	citrus fruit, whole milk
0	0.030097	pip fruit, whole milk

Thời gian hiển thị tập phổ biến:

```
time: 8.07 ms (started: 2023-12-15 16:53:13 +00:00)
```

Tiếp theo chuyển các itemsets sang frozenset(), kết quả trả về là một tập hợp (Set) không thể thay đổi. Vì các itemset không nên thay đổi sau khi đã được tạo, việc sử dụng frozenset giúp đảm bảo tính toàn vẹn của dữ liệu.

```
#Chuyển itemsets về kiểu frozenset để phù hợp với format của hàm
↪ association_rule()
frequent_itemsets_eclat['itemsets'] =
↪ frequent_itemsets_eclat['itemsets'].apply(lambda x: frozenset(x.split(',
↪ ')))
frequent_itemsets_eclat['itemsets'] =
↪ tuple(frequent_itemsets_eclat['itemsets'])
frequent_itemsets_eclat['itemsets']
↪ =frequent_itemsets_eclat['itemsets'].apply(frozenset)
frequent_itemsets_eclat
```

Tạo bảng kết quả gồm các thành phần của các luật phổ biến cùng với các chỉ số support, confidence, lift, conviction của các tập luật. Tuy nhiên, không giống như Apriori, phương pháp ECLAT không dựa trên tính toán độ đo confidence và lift, do đó thư viện chỉ hỗ trợ hàm tính support, các chỉ số còn lại cần được tính toán thủ công. Do đó ở bảng này một số cột bị trống.

```
from mlxtend.frequent_patterns import association_rules


# Tạo bảng luật kết hợp
rules_eclat = association_rules(frequent_itemsets_eclat, metric = "support",
↪ min_threshold = 0.03,support_only=True)
rules_eclat = rules_eclat.sort_values(['support'], ascending = [False])


# Chỉ in các chỉ số cần thiết
rules_eclat[['antecedents', 'consequents','antecedent support',
       'consequent support', 'support', 'confidence', 'lift','conviction']]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	conviction
0	(other vegetables)	(whole milk)	NaN	NaN	0.074835	NaN	NaN	NaN
1	(whole milk)	(other vegetables)	NaN	NaN	0.074835	NaN	NaN	NaN
2	(rolls/buns)	(whole milk)	NaN	NaN	0.056634	NaN	NaN	NaN
3	(whole milk)	(rolls/buns)	NaN	NaN	0.056634	NaN	NaN	NaN
4	(whole milk)	(yogurt)	NaN	NaN	0.056024	NaN	NaN	NaN
5	(yogurt)	(whole milk)	NaN	NaN	0.056024	NaN	NaN	NaN
6	(whole milk)	(root vegetables)	NaN	NaN	0.048907	NaN	NaN	NaN
7	(root vegetables)	(whole milk)	NaN	NaN	0.048907	NaN	NaN	NaN
9	(root vegetables)	(other vegetables)	NaN	NaN	0.047382	NaN	NaN	NaN
8	(other vegetables)	(root vegetables)	NaN	NaN	0.047382	NaN	NaN	NaN
10	(other vegetables)	(yogurt)	NaN	NaN	0.043416	NaN	NaN	NaN

Tiếp theo nhóm xác định các luật kết hợp mạnh. Luật kết hợp mạnh là những luật có

support $\geq$ minSup đồng thời confidence $\geq$ minConf. Để tìm ra các luật kết hợp mạnh, cần tính độ đo confidence. Sau đó chọn ra các luật có confidence $\geq$ minConf được chỉ định.

```
#Tính chỉ số confidence cho từng luật
conf_list = []
for i, row in rules_eclat.iterrows():
  conf = row['support']*len(df)/int(binary_df[row['antecedents']].sum()[0])
  conf_list.append(conf)
rules_eclat['confidence'] = conf_list


#In những luật kết hợp mạnh, minConf = 0.4
rules_eclat=rules_eclat[rules_eclat['confidence']>=0.4][['antecedents',
↪  'consequents','support', 'confidence']]
rules_eclat
```

	antecedents	consequents	support	confidence
5	(yogurt)	(whole milk)	0.056024	0.401603
7	(root vegetables)	(whole milk)	0.048907	0.448694
9	(root vegetables)	(other vegetables)	0.047382	0.434701
15	(tropical fruit)	(whole milk)	0.042298	0.403101
31	(whipped/sour cream)	(whole milk)	0.032232	0.449645

Vậy bằng thuật toán ECLAT, các tập luật kết hợp mạnh được xác định gồm:

- (yogurt whole milk): có khoảng 5,6% các giao dịch có cả yogurt và whole milk, trong đó có khoảng 40% nếu mua yogurt thì sẽ mua whole milk.
- (root vegetables whole milk): có khoảng 4,8% các giao dịch cùng có mặt root vegetables và whole milk, trong đó có khoảng 44% các giao dịch nếu mua vegetables thì sẽ mua whole milk.
- (root vegetable other vegetables): khoảng 4,7% tổng các giao dịch xuất hiện đồng thời root vegetable và other vegetables, trong đó khoảng 43% mua root vegetable thì sẽ mua other vegetable.
- (tropical fruit whole milk): khoảng 4.2% các giao dịch có cả tropical fruit và whole milk, trong đó khoảng 40% mua tropical fruit sẽ mua whole milk.

- (whipped/sour cream whole milk): khoảng 3.2% các giao dịch cùng có whipped/sour cream và whole milk, trong đó gần 45% nếu mua whipped/sour cream thì sẽ mua whole milk.

3.2 Áp dụng thuật toán Apriori cho bộ dữ liệu Groceries 2

Đối với tập dữ liệu Groceries, phương pháp tìm luật kết hợp phổ biến bằng thuật toán Apriori được tiến hành như sau:

Đầu tiên ta cần import thư viện có hỗ trợ thuật toán Apriori:

```
!pip install -q mlxtend
```

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
```

Chuyển đổi danh sách các tập hợp thành một ma trận nhị phân, trong đó mỗi cột đại diện cho một mục và mỗi hàng đại diện cho một giao dịch:

```
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
transformed_data = pd.DataFrame(te_ary, columns=te.columns_)
```

Sử dụng hàm `apriori` từ thư viện `mlxtend.frequent_patterns` để tìm các tập hợp mục phổ biến trong dữ liệu (transformed_data). Chọn $min\_support = 0.03$: ngưỡng hỗ trợ tối thiểu, tức là chỉ các tập hợp mục xuất hiện trong ít nhất 3% số giao dịch sẽ được coi là phổ biến.

```
frequent_itemsets_apriori = apriori(transformed_data, min_support=0.03,
 ↪  use_colnames=True)
frequent_itemsets_apriori[frequent_itemsets_apriori['itemsets'].apply(lambda x:
 ↪  len(x) >= 2)].sort_values(['support'], ascending = [False])
```

Có tổng tất cả 19 itemsets phổ biến thoả giá trị $minSup = 0.03$

	support	itemsets
50	0.074835	(whole milk, other vegetables)
56	0.056634	(whole milk, rolls/buns)
62	0.056024	(whole milk, yogurt)
58	0.048907	(whole milk, root vegetables)
47	0.047382	(other vegetables, root vegetables)
51	0.043416	(other vegetables, yogurt)
46	0.042603	(other vegetables, rolls/buns)
60	0.042298	(whole milk, tropical fruit)
59	0.040061	(whole milk, soda)
55	0.038332	(rolls/buns, soda)
49	0.035892	(other vegetables, tropical fruit)
57	0.034367	(rolls/buns, yogurt)
44	0.034367	(whole milk, bottled water)
52	0.033249	(pastry, whole milk)
48	0.032740	(other vegetables, soda)
61	0.032232	(whole milk, whipped/sour cream)
54	0.030605	(sausage, rolls/buns)
45	0.030503	(whole milk, citrus fruit)
53	0.030097	(whole milk, pip fruit)

Thời gian chạy thuật toán và hiển thị tập phổ biến:

```
time: 70.3 ms (started: 2023-12-15 16:51:09 +00:00)
```

Ta tiếp tục tìm ra các itemset có luật kết hợp mạnh thỏa mãn giá trị $minConf = 0.4$:

```
rules_apriori = association_rules(frequent_itemsets_apriori, metric =
↪  "confidence", min_threshold = 0.4)
rules_apriori = rules_apriori.sort_values(['support'], ascending = [False])
```

Sau khi áp dụng thuật toán Apriori để tìm ra luật kết hợp mạnh từ các itemset phổ biến trong tập dữ liệu Groceries thoả giá trị $minSup = 0.03, minConf = 0.4$, nhóm thu được kết quả có tổng cộng 5 luật kết hợp mạnh như sau:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	conviction
4	(yogurt)	(whole milk)	0.139502	0.255516	0.056024	0.401603	1.571735	1.244132
1	(root vegetables)	(whole milk)	0.108998	0.255516	0.048907	0.448694	1.756031	1.350401
0	(root vegetables)	(other vegetables)	0.108998	0.193493	0.047382	0.434701	2.246605	1.426693
2	(tropical fruit)	(whole milk)	0.104931	0.255516	0.042298	0.403101	1.577595	1.247252
3	(whipped/sour cream)	(whole milk)	0.071683	0.255516	0.032232	0.449645	1.759754	1.352735

4 CHƯƠNG 4: SO SÁNH KẾT QUẢ THUẬT TOÁN ECLAT VÀ APRIORI

Qua kết quả trên, ta có có thể thấy thuật toán ECLAT thực thi chậm hơn rất nhiều so với thuật toán Apriori. Điều này gây ra sự mâu thuẫn trong nhận định về thời gian chạy đã đề cập ở Chương 2. Lý do có thể đến từ hai thư viện hỗ trợ cho hai thuật toán là khác nhau. Thuật toán Apriori được hỗ trợ trong thư viện mlxtend, nhưng để áp dụng được thuật toán ECLAT, ta cần một thư viện khác đó là pyECLAT. Điều này cũng khiến cho thuật toán ECLAT không tính được các độ đo khác ngoài support khi chạy hàm association_rules() của thư viện mlxtend. Tuy nhiên, thuật toán ECLAT và Apriori đều cho kết quả là các tập phổ biến và luật kết hợp giống nhau. Vì vậy, thuật toán ECLAT đã được nhóm triển khai đúng.

5 CHƯƠNG 5 : ĐÁNH GIÁ LUẬT KẾT HỢP

5.1 Độ đo tương quan Chi-square

Trong đánh giá kết quả của luật kết hợp, độ đo tương quan Chi-square đo lường mức độ khác biệt giữa tần suất quan sát và kỳ vọng thông qua việc thực hiện các phép tính trên bảng tương quan (contingency table). Bảng tương quan là một bảng mô tả phân phối tần suất của hai biến. Trong luật kết hợp X => Y, các hàng trong bảng tương quan lần lượt thể hiện sự có xuất hiện và không xuất hiện của X; các cột lần lượt thể hiện sự có xuất hiện và không xuất hiện của Y. Bảng tương quan được thể hiện như sau:

	$\mathbf{Y}$	$\bar{\mathbf{Y}}$	
$\mathbf{X}$	$O_1(E_1)$	$O_2(E_2)$	O_1+O_2
$\bar{\mathbf{X}}$	$O_3(E_3)$	$O_4(E_4)$	O_1+O_2
	O_1+O_3	O_2+O_4	

Trong đó:

- O_1 là số lần xuất hiện đồng thời của X và Y.
- O_2 là số lần X xuất hiện nhưng Y không xuất hiện.

- O_3 là số lần Y xuất hiện nhưng X không xuất hiện.
- O_4 là số lần cả X và Y đều không xuất hiện.
- E_{ij} là giá trị kỳ vọng, được tính bằng cách lấy tích của tổng giá trị dòng i và tổng giá trị cột j rồi chia cho tổng số quan sát. Trong bảng, E_{11} được ký hiệu là E_1 thể hiện giá trị kỳ vọng của O_1, tương tự với E_2, E_3, E_4.

Khi đã xác định được các giá trị quan sát và kỳ vọng, độ đo Chi-square được tính như sau:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Để thực hiện tính độ đo tương quan Chi-square cho bộ dữ liệu Groceries, ta có thể cài đặt code như sau:

```
from scipy.stats import chi2_contingency

chi2_list = []
expected_freq = []
freq = []
p_value = []

for i, row in rules_eclat.iterrows():
  # số lần xuất hiện đồng thời của antecedents và consequents
  ob1 = row['support']*len(df)
  # số lần antecedents xuất hiện nhưng consequents không xuất hiện
  ob2 = int(binary_df[row['antecedents']].sum()[0]) - ob1
  # số lần consequents xuất hiện nhưng antecedents không xuất hiện
  ob3 = int(binary_df[row['consequents']].sum()[0]) - ob1
  # số lần cả 2 đều không xuất hiện
  ob4 = (len(df)-int(binary_df[row['antecedents']].sum()[0])) - ob3

  # tạo contingency table
  table = np.array([[ob1,ob2],[ob3,ob4]])
  contingency_table = chi2_contingency(table)

  chi2 = contingency_table.statistic
  chi2_list.append(round(chi2))
  p_value.append(contingency_table.pvalue)
  expected_freq.append(round(contingency_table.expected_freq[0,0])) # lấy giá
  ↪  trị kỳ vọng cả 2 cùng xuất hiện
  freq.append(round(ob1))

```

```
rules_eclat['Chi2'] = chi2_list
rules_eclat['pvalue'] = p_value
rules_eclat['freq'] = freq
rules_eclat['expected_freq'] = expected_freq
rules_eclat[['antecedents',
↪  'consequents','Chi2','pvalue','freq','expected_freq']]
```

	antecedents	consequents	Chi2	pvalue	freq	expected_freq
21	(yogurt)	(whole milk)	178	1.333683e-40	551	351
13	(root vegetables)	(whole milk)	235	5.124682e-53	481	274
15	(root vegetables)	(other vegetables)	447	3.519862e-99	466	207
23	(tropical fruit)	(whole milk)	131	2.292893e-30	416	264
25	(whipped/sour cream)	(whole milk)	149	2.397312e-34	317	180

Cả 5 luật kết hợp mạnh đều có giá trị Chi-square lớn hơn 1, cho thấy có sự phụ thuộc giữa *antecedents* và consequents. Cụ thể hơn, cả 5 luật đều có tần suất quan sát (freq) lớn hơn khá nhiều so với giá trị kỳ vọng (expected_freq) cho thấy sự liên kết giữa *antecedents* và consequents là đáng kể và có thể có ảnh hưởng lớn đến sự xuất hiện của *consequents* khi *antecedents* xuất hiện. Ví dụ, khách hàng mua yogurt cũng sẽ có xu hướng mua whole milk. Đồng thời giá trị p rất thấp có thể là biểu hiện cho một quy luật mạnh và có ý nghĩa thống kê.

5.2 Độ đo tương quan Lift

Độ đo tương quan lift trong luật kết hợp được sử dụng để đánh giá mức độ tương quan giữa hai sự kiện được xác định bởi công thức:

$$\text{lift}(X \Rightarrow Y) = \frac{\sup(X \Rightarrow Y)}{\sup(X) \cdot \sup(Y)} = \frac{\text{conf}(X \Rightarrow Y)}{\sup(Y)}$$

hoặc

$$\text{lift}\ (X \Rightarrow Y) = \frac{P(X \cap Y)}{P(X) \cdot P(Y)} = \frac{P(Y \mid X)}{P(Y)}$$

- Nếu Lift(X, Y) = 1: X và Y độc lập nhau, không có tương quan.
- Nếu Lift(X, Y) < 1: X tương quan nghịch với Y, tức là sự có mặt của một item có khả năng dẫn đến sự vắng mặt của item khác.

- Nếu Lift(X, Y) > 1: X tương quan thuận với Y, tức là sự có mặt của một item có khả năng dẫn đến sự có mặt của item khác.

Độ đo lift được tính toán thủ công như sau:

```
lift_list = []
for i, row in rules_eclat.iterrows():
  lift =
   ↪  row['confidence']/(int(binary_df[row['consequents']].sum()[0])/len(df))
  lift_list.append(lift)


rules_eclat['lift'] = lift_list
```

```
#X, Y độc lập
rules_eclat[rules_eclat.lift == 1][['antecedents', 'consequents','support',
 ↪  'confidence', 'lift']]
```

```
#X, Y tương quan nghịch
rules_eclat[rules_eclat.lift < 1][['antecedents', 'consequents','support',
 ↪  'confidence', 'lift']]
```

Cả 2 đoạn chương trình đều cho ra kết quả trống như sau:

antecedents	consequents	support	confidence	lift

```
#X, Y tương quan thuận
rules_eclat[rules_eclat.lift > 1][['antecedents', 'consequents','support',
 ↪  'confidence', 'lift']]
```

	antecedents	consequents	support	confidence	lift
5	(yogurt)	(whole milk)	0.056024	0.401603	1.571735
7	(root vegetables)	(whole milk)	0.048907	0.448694	1.756031
9	(root vegetables)	(other vegetables)	0.047382	0.434701	2.246605
15	(tropical fruit)	(whole milk)	0.042298	0.403101	1.577595
31	(whipped/sour cream)	(whole milk)	0.032232	0.449645	1.759754

Kết luận: không có luật nào mà các thành phần của chúng là độc lập hoặc tương quan nghịch với nhau. Vậy, tất cả các luật kết hợp mạnh này đề có mối tương quan thuận, sự có mặt của món hàng này có khả năng dẫn đến sự có mặt của món còn lại trong luật.

6 CHƯƠNG 5: KẾT LUẬN

Với đề tài Áp dụng thuật toán ECLAT cho bộ dữ liệu Groceries 2, nhóm đã tìm hiểu về thuật toán ECLAT đồng thời áp dụng thuật toán này vào bộ dữ liệu được cung cấp để tiến hành tìm ra các luật kết hợp. Để đánh giá thuật toán, các độ đo bao gồm Chi-square và LIFT đã đươc sử dụng để có đánh giá phù hợp với mục tiêu khai thác các tập luật kết hợp có ý nghĩa và có thể áp dụng vào thực tế trong việc gợi ý sản phẩm cũng như giúp đưa ra các kế hoạch kinh doanh tốt. Bên cạnh xây dựng thuật toán ECLAT, đồ án còn áp dụng thuật toán Apriori và so sánh với thuật toán ECLAT nhằm đưa ra góc nhìn chi tiết hơn về hai phương pháp phổ biến trong khai phá dữ liệu này.

Tuy nhiên, đồ án vẫn tồn tại một số hạn chế. Trong quá trình phát triển đề tài, nhóm chưa có đánh giá khách quan nhất về hiệu suất của thuật toán do lượng dữ liệu chưa đáp ứng được độ lớn và đa dạng (số dòng dữ liệu ít). Nhìn chung, hiệu suất của thuật toán ECLAT trong việc khai thác luật kết hợp là rất tốt và là thuật toán mạnh mẽ cho việc khám phá các mối quan hệ tương quan giữa các items trong bộ dữ liệu.

BẢNG PHÂN CÔNG

Thành viên	Phân công	Đánh giá
Vũ Nguyễn Thảo Vi	Tìm hiểu lý thuyết ECLAT Thực hiện thuật toán ECLAT Đánh giá luật kết hợp So sánh thuật toán	100%
Nguyễn Quốc Việt	Tìm hiểu lý thuyết ECLAT Thực hiện thuật toán Apriori Tổng hợp nội dung So sánh thuật toán	100%
Bùi Quốc Việt	Tìm hiểu lý thuyết ECLAT Đọc dữ liệu Đánh giá luật kết hợp Mở đầu, Kết luận	100%
Nguyễn Thanh Vy	Tìm hiểu lý thuyết ECLAT Thực hiện thuật toán Apriori Đánh giá luật kết hợp So sánh thuật toán	100%
Nguyễn Nhật Thảo Vy	Tìm hiểu lý thuyết ECLAT Thực hiện thuật toán ECLAT Đánh giá luật kết hợp So sánh thuật toán	100%

Tài liệu

(n.d.). Association rule mining. Retrieved December 15, 2023.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, page 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Borgelt, C. (2003). Efficient implementations of apriori and eclat. *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations; 19 November 2003*.

Han, J., Kamber, M., and Pei, J. (2012). 6 - mining frequent patterns, associations, and correlations: Basic concepts and methods. In Han, J., Kamber, M., and Pei, J., editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 243–278. Morgan Kaufmann, Boston, third edition edition.

Richard, J. (2020). pyeclat. Retrieved December 15, 2023.

Zaki, M. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390.