TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA TOÁN - ỨNG DỤNG

BÁO CÁO THỰC HÀNH

TẬP PHỔ BIẾN & LUẬT KẾT HỢP

SVTH: NGUYỄN ĐĂNG TIẾN

MSSV: 3123580050

GVHD: TS. ĐỖ NHƯ TÀI

Năm học: 2025 - 2026

Mục lục

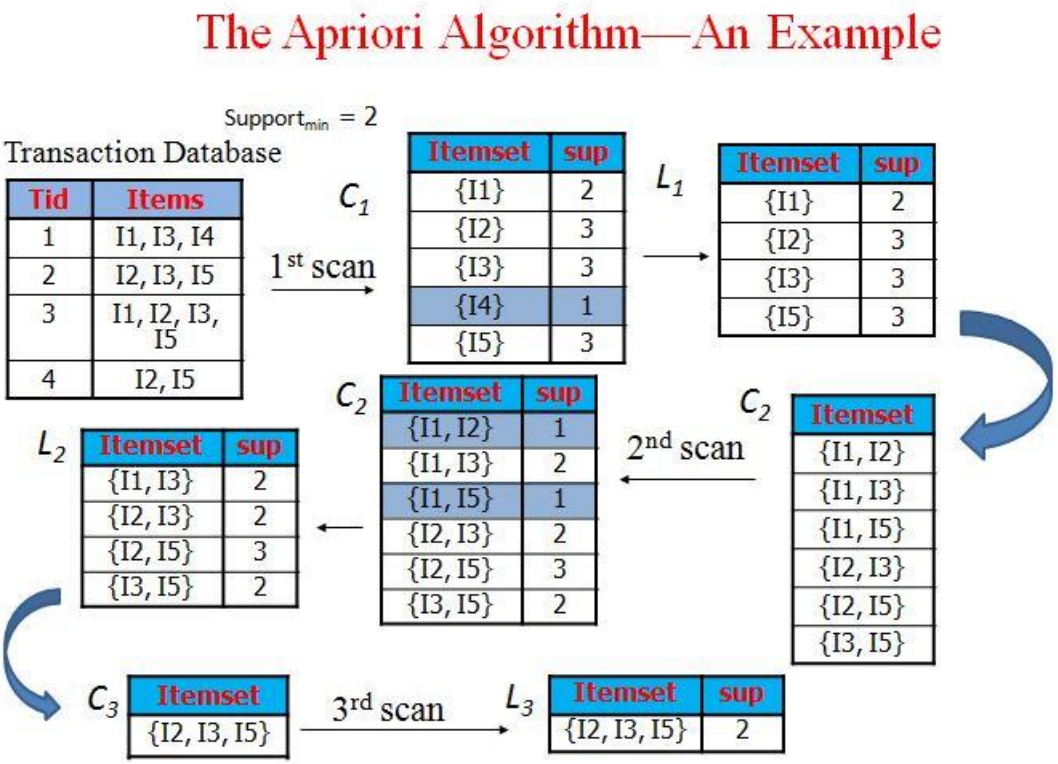
Mục tiêu chung

- Hướng dẫn sinh viên triển khai thực hiện việc tìm tập phổ biến bằng giải thuật Apriori.
- Hướng dẫn sinh viên trích xuất luật kết hợp từ tập phổ biến và cách thức trình bày một luật
- kết hợp
- Hướng dẫn sinh viên ôn tập lý thuyết về tập phổ biến và luật kết hợp
- Hướng dẫn sinh viên cách tiếp cận để tìm kiếm các đoạn mã nguồn bằng Python hỗ trợ cho
- việc xây dựng các giải thuật

CHƯƠNG I: XÁC ĐỊNH TẬP PHỔ BIẾN VỚI GIẢI THUẬT APRIORI

1.1. Ôn tập lý thuyết

Giải thuật Apriori hoạt động như thế nào? Hãy giải thích các bước chính trong quá trình tìm tập phổ biến?



Apriori là thuật toán kinh điển dùng để tìm **các tập mục (itemsets) phổ biến** trong cơ sở dữ liệu giao dịch.

Một tập mục được coi là phổ biến nếu **độ hỗ trợ (support)** của nó ≥ một ngưỡng mô tả trước (min_sup).

Các bước chính của thuật toán:

- **B1: Tạo tập ứng viên C_1**
  - Liệt kê tất cả các mục đơn lẻ xuất hiện trong dữ liệu
  - Đếm số giao dịch chứa từng mục
- **B2: Sinh tập phổ biến L_1 từ C_1**
  - Với mỗi item, tính **support** = số giao dịch chứa item / tổng giao dịch.
  - Giữ lại những item có support ≥ min_sup → gọi là L_1.

- **B3: Sinh tiếp tập ứng viên từ tập phổ biến L_{k-1}**
  - **<u>Tạo ứng viên:</u>** Ghép đôi các tập mục phổ biến L_{k-1} để tạo tập mục mới kích thước k.
  - **<u>Cắt tỉa từng ứng viên:</u>** Loại bỏ những ứng viên mà bất kỳ tập con (k−1) của nó không nằm trong L_{k-1}, dựa trên Apriori property.
- **B4: Quét cơ sở dữ liệu dể tìm tập phổ biến cuối cùng**
  - Đếm support cho các ứng viên C_k bằng cách duyệt toàn bộ dữ liệu.
  - Giữ lại các tập có support ≥ min_sup → gọi là L_k.
  - Nếu L_k rỗng → dừng.
  - Ngược lại → quay lại bước 3 để sinh tập lớn hơn.

Các tham số như support, confidence, và lift có ý nghĩa gì trong khai thác luật kết hợp? Chúng được sử dụng như thế nào trong Apriori?

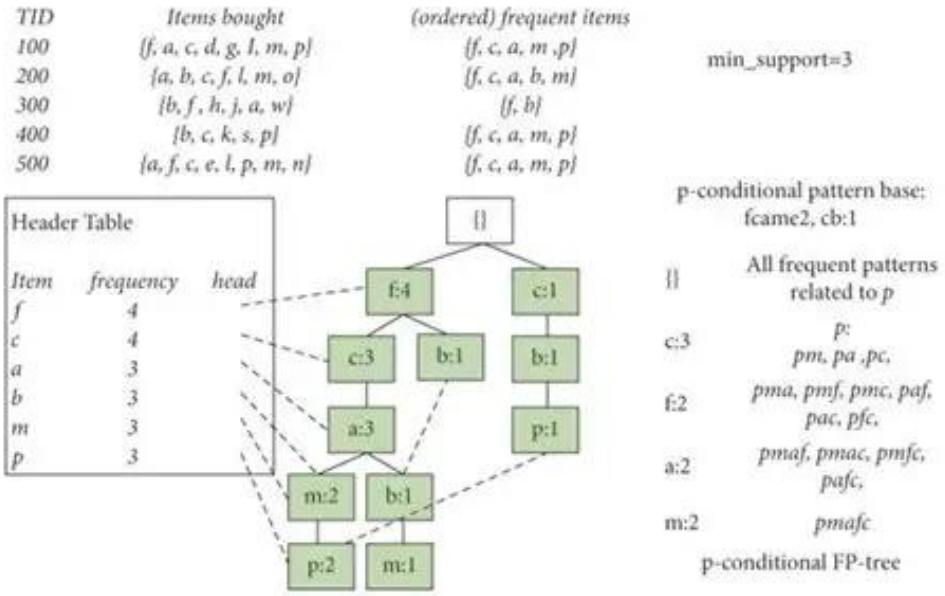
1. **Support:** Đo mức độ phổ biến của một tập item trong toàn bộ cơ sở dữ liệu giao dịch (Tỉ lệ số giao dịch chứu X trên tổng số giao dịch = **Support(X)**)
   - **Vai trò:** Xác định xem tập mục có phổ biến hay khồn, lọc các item không dủ phổ biến
2. **Confidence:** Đo mức độ chính xác của luật, tức trong số các giao dịch chứa X, có bao nhiêu giao dịch đồng thời chứa Y (**Confidence(X->Y) = Support(X hợp Y) / Support(X)**) giá trị này càng cao thì luật càng đáng tin
   - **Vai trò:** Dùng sau khi tìm được các tập phổ biến, kiểm tra chất lượng của luật kết hợp sinh ra từ các itemset phổ biến
3. **Lift: Mức độ tương quan giữa X và Y (Lift(X→Y) = support(X ∪ Y) / (support(X) × support(Y))) Nếu:**
   - Lift > 1: X và Y có quan hệ dương tức là X mua thì có thể mua thêm Y
   - Lift = 1: X và Y độc lập hoặc ngẫu nhiên
   - Lift < 1: X và Y có quan hệ âm tức là X mua thì có khả năng cao là không mua Y
   - **Vai trò:** Dùng sau khi tìm được các tập phổ biến, kiểm tra mối quan hệ giữa X và Y xác định xem luật có ý nghĩa hay không?

- **Support** lọc tập phổ biến (trong quá trình chạy Apriori).
- **Confidence** lọc các luật tốt (sau khi có tập phổ biến).
- **Lift** đánh giá mức độ tương quan thật sự để chọn luật thật sự có ý nghĩa.

Sự khác biệt giữa giải thuật Apriori và các thuật toán khác như FP-Growth trong việc tìm tập phổ biến là gì?

Apriori: Dựa trên việc sinh nhiều ứng việc tìm tập phổ biến bằng cách sinh các tập ứng viên lẻ rồi quét toàn bộ cơ sở dữ liệu để kiểm tra (Phải sinh ứng viên và quét dữ liệu nhiều lần)

FP-Growth: Sử dụng cấu trúc FP-Tree, nén toàn bộ dữ liệu vào một cây, duyệt FP-Tree theo từng điều kiện để sinh các tập phổ biến (Không cần quét dữ liệu nhiều lần)



Khác biệt lớn nhất chính là: Về cách quét và tổ chức dữ liệu để tìm tập kết hợp nhanh hơn để có thể sinh ra luật

*Viết đoạn code mẫu bằng Python (sử dụng thư viện như mlxtend hoặc apyori) để triển khai giải thuật Apriori không? Hãy mô tả các bước thực hiện***?**

```
# ----------------------------

# 1. Import thư viện cần thiết

# ----------------------------

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori,
association_rules


# ----------------------------

# 2. Chuẩn bị dữ liệu giao dịch
```

```
    #    (mỗi giao dịch là 1 list item)
    # ----------------------------
    transactions = [
        ['Milk', 'Bread', 'Butter'],
        ['Beer', 'Bread'],
        ['Milk', 'Bread', 'Beer', 'Eggs'],
        ['Milk', 'Bread', 'Butter'],
        ['Bread', 'Butter']
    ]

    # ------------------------------------------------
    # 3. Mã hóa dữ liệu giao dịch thành dạng one-hot
    #    (mỗi cột là 1 item, giá trị True/False)
    # ------------------------------------------------
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)

    print("=== Dữ liệu sau khi mã hóa one-hot ===")
    print(df)

    # ------------------------------------------------
    # 4. Chạy Apriori để tìm các tập mục phổ biến
    #    min_support: ngưỡng độ hỗ trợ tối thiểu
    # ------------------------------------------------
    frequent_itemsets = apriori(df, min_support=0.4,
use_colnames=True)
```

```
print("\n=== Các tập mục phổ biến ===")
print(frequent_itemsets)


# ------------------------------------------------
# 5. Sinh các luật kết hợp từ tập phổ biến
#    metric: dùng 'confidence' để lọc
#    min_threshold: ngưỡng confidence tối thiểu
# ------------------------------------------------
rules = association_rules(frequent_itemsets,
                          metric="confidence",
                          min_threshold=0.6)


print("\n=== Các luật kết hợp tìm được ===")
print(rules[['antecedents', 'consequents',
             'support', 'confidence', 'lift']])
```

Làm thế nào để tiền xử lý dữ liệu giao dịch (transactional data) trong Python trước khi áp dụng Apriori?

- Với thư viện `mixtend` cần ở dạng môi hàng là một giao dịch và mỗi cột là 1 item giá trị True/False
- Với thư viện `apyori` cần list giao dịch mỗi giao dịch là một list item
- Cụ thể có thể bỏ khoảng trắng hai đầu, chuyển về chữ thường, bỏ các item NaN, bỏ các itrem quá hiếm, không dóng góp nhiều vào luật, lọc giao dịch quá ngắn, xử lý file dạng hóa đơn thì chuyển về one-hot hoặc list item

Hàm nào trong Python để tính toán tập phổ biến và luật kết hợp từ giải thuật Apriori? Hãy chia sẻ một đoạn code mẫu

Hàm `mlxtend.frequent_patterns.apriori` → **tính tập phổ biến (frequent itemsets)**

Hàm `mlxtend.frequent_patterns.association_rules` → **sinh luật kết hợp (association rules)** từ các tập phổ biến đó.

```
import pandas as pd
```

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


    # 1. Dữ liệu giao dịch (mỗi phần tử là 1 giao
dịch)
    transactions = [
        ['milk', 'bread', 'butter'],
        ['beer', 'bread'],
        ['milk', 'bread', 'beer', 'eggs'],
        ['milk', 'bread', 'butter'],
        ['bread', 'butter']
    ]


    # 2. Mã hoá one-hot cho dữ liệu giao dịch
    te = TransactionEncoder()
    te_ary =
te.fit(transactions).transform(transactions)
    df_onehot = pd.DataFrame(te_ary,
columns=te.columns_)


    print("=== DataFrame one-hot ===")
    print(df_onehot)


    # 3. Tính các tập mục phổ biến bằng Apriori
    frequent_itemsets = apriori(
        df_onehot,
```

```
        min_support=0.4,   # chỉ giữ các itemset xuất
hiện ≥ 40% giao dịch
        use_colnames=True  # dùng tên item thay vì
index cột
    )

    print("\n=== Các tập mục phổ biến ===")
    print(frequent_itemsets)

    # 4. Sinh luật kết hợp từ tập phổ biến
    rules = association_rules(
        frequent_itemsets,
        metric="confidence",  # lọc theo độ tin cậy
        min_threshold=0.6     # chỉ giữ luật có
confidence ≥ 60%
    )

    print("\n=== Các luật kết hợp ===")
    print(rules[['antecedents', 'consequents',
                 'support', 'confidence', 'lift']])
```

Điều chỉnh ngưỡng support và confidence trong Apriori để đạt được kết quả tốt hơn không? Hãy mô tả cách thực hiện

Điều chỉnh ngưỡng support và confidence có khả năng đạt được kết quả tốt hơn

- **support quá cao** → chỉ thu được vài tập phổ biến (hoặc không có luật nào).
- **support quá thấp** → rất nhiều tập phổ biến → thuật toán chạy **rất chậm**, dễ sinh **luật vô nghĩa**.

Tương tự:

- **confidence cao** → luật thu được ít nhưng rất đáng tin.
- **confidence thấp** → nhiều luật nhưng có thể nhiễu.

Nguyên tắc điều chỉnh support và confidence

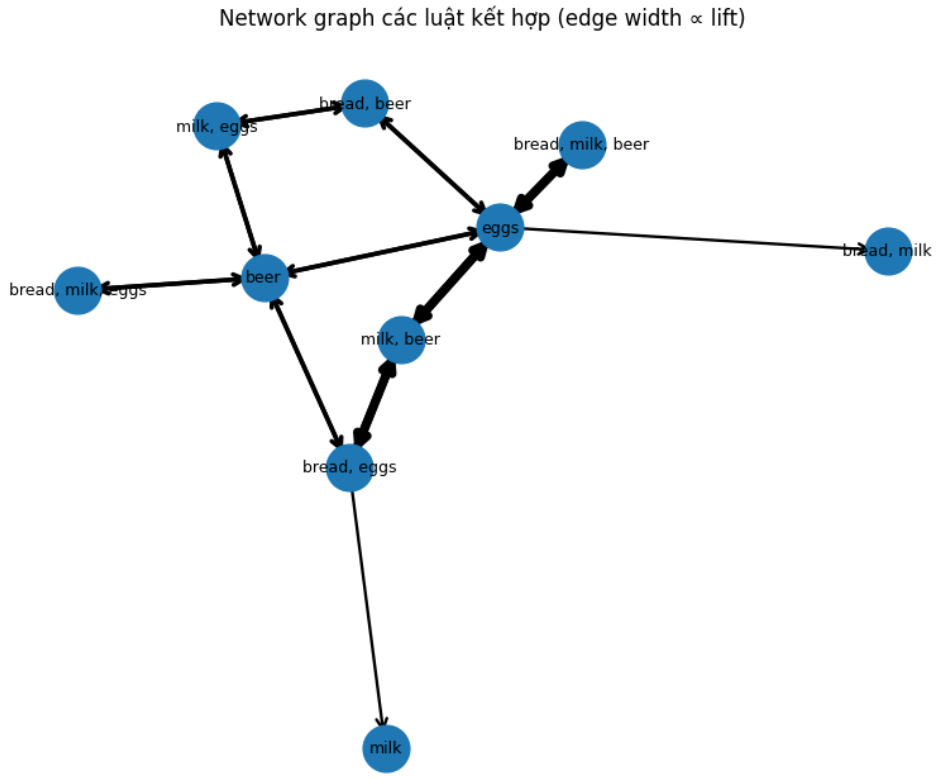
1. Điều chỉnh support trước
   - Số lượng tập phổ biến và thời gian chạy để sinh ứng viên rất lâu
   - Với tập dữ liệu dưới **1000** giao dịch support nên là **0.1 – 0.3**
   - Với tập dữ liệu từ **1000 - 100000** giao dịch support nên là **0.01 – 0.1**
   - Với tập dữ liệu trên **100000** giao dịch support nên là **0.001 – 0.01**
2. Điều chỉnh confidence sau
   - Khi có số lượng tập phổ biến
   - Mục tiêu nếu muốn có nhiều luật để khám phá confidence **0.3 – 0.5**
   - Mục tiêu nếu muốn có các luật gợi ý hành vi có ý nghĩa confidence **0.6 – 0.6**
   - Muốn các luật rất chất lượng confidence **0.8+**

Cách thực hiện điều chỉnh luật

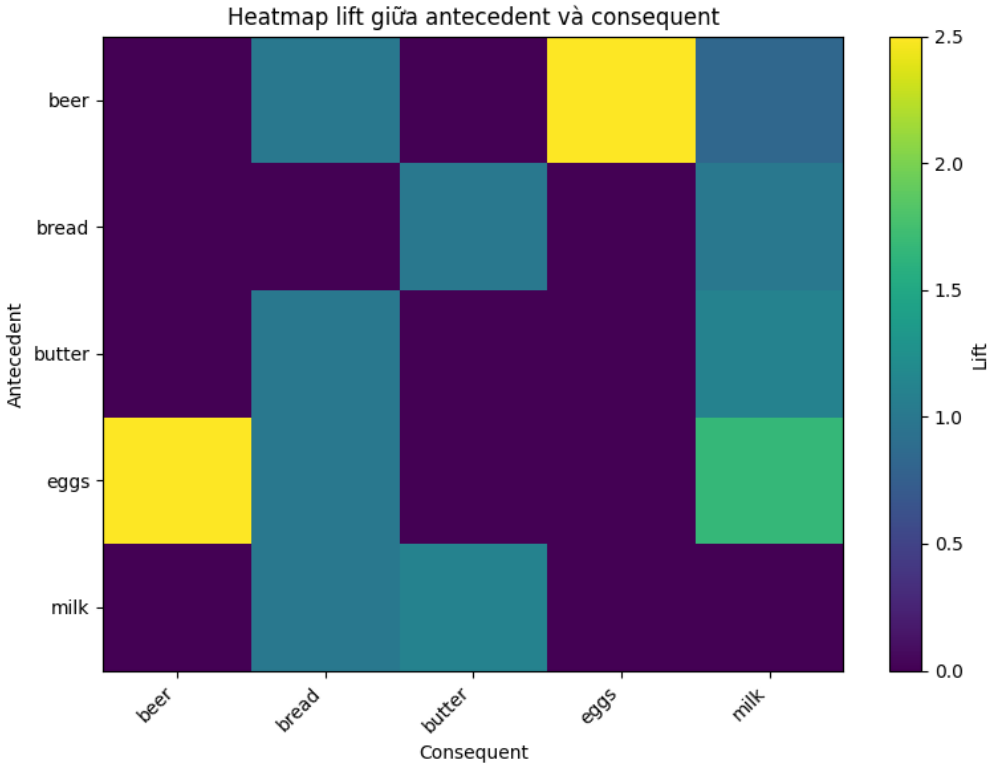
- Chạy thử nghiệm
- Nếu ít luật giảm support ngược lại thì tang
- Nếu muốn giữ lại các luạt mạch đáng tin tang confidence
- Nếu ít luật quá thì giảm confidence

Cách trực quan hóa các luật kết hợp (association rules) dưới dạng biểu đồ mạng (network graph) hoặc heatmap trong Python

Biểu đồ mạng (network graph): node là item, cạnh là luật X → Y



Heatmap: trục X/Y là item/law, màu thể hiện *confidence* hoặc *lift*



- Không vẽ tất cả các luật, Lấy các luật đơn giản khi vẽ heatmap, Kết hợp cả 2

CHƯƠNG II: XÁC ĐỊNH LUẬT KẾT HỢP TỪ TẬP PHỔ BIẾN

2.1. Ôn tập lý thuyết

Luật kết hợp (association rules) là gì? Hãy giải thích các khái niệm support, confidence, và lift trong luật kết hợp

Luật kết hợp là những quy tắc dạng X kéo theo Y (Nếu một giao dịch chứa X có khả năng nó sẽ chứa Y)

Các khái niệm

1. **Support:** Đo mức độ phổ biến của một tập mục trong toàn bộ dataset
2. **Confidence:** Đo xác suất Y xảy ra khi X xảy ra
3. **Lift:** Xác định mối quan hệ giữa X và Y

Quá trình tìm luật kết hợp từ tập phổ biến bao gồm những bước nào? Làm thế nào để chuyển từ tập phổ biến sang luật kết hợp?

Quy trình bao gồm: Tìm tập mục phổ biến (Theo support) – Sinh luật kết hợp từ các tập phồ biến (Theo confidence)

1. Lấy từng cặp phổ biến có kích thước lớn hơn bằng 2
2. Sinh tất carc các luật ứng viên từ các tập mục phổ biến
3. Tính confidence cho từng luật
4. Tính lift để đánh giá luật tốt hay không
5. Giữ lại các luật thỏa yêu cầu

Yêu cầu:

- ***Support đủ lớn***
- ***Confidence cao***
- ***Lift > 1 hoặc < 1 tùy bài toán***

Khi nào một luật kết hợp được coi là "mạnh" (strong rule)? Các chỉ số như lift hoặc confidence ảnh hưởng như thế nào đến đánh giá này?

Một luật kết hợp được coi là “mạnh” khi:

- **Support cao:** Luật xuất hiện đủ nhiều trong dữ lieuj -> Thống ke có ỹ nghĩa
- **Confidence cao:** Khi có X thì khả năng cao xảy ra Y
- **Lift > 1:** X và Y thực sự có quan hệ không phải xuất hiện ngãu nhiên

- **Confidence:** Đóng vai trò làm tiêu chí chính để học luật, là tiền đề để dự đoán Y

- **Lift:** Đo xem nếu có X thì khả năng sẽ mụa Y hay không là điều kiện quan trọng để đánh giá xem luật có ý nghĩa

Viết đoạn code mẫu bằng Python (sử dụng thư viện như mlxtend hoặc apyori) để tìm luật kết hợp từ tập phổ biến không? Hãy mô tả các bước thực hiện?

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules


# 1) Dữ liệu giao dịch mẫu
transactions = [
    ["milk", "bread", "eggs"],
    ["milk", "bread"],
    ["milk", "diaper", "beer", "bread"],
    ["bread", "diaper", "beer"],
    ["milk", "diaper", "bread", "cola"],
]


# 2) One-hot encode: transaction -> dataframe (True/False)
te = TransactionEncoder()
te_array = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_array, columns=te.columns_)


# 3) Tìm tập phổ biến (frequent itemsets)
frequent_itemsets = apriori(df, min_support=0.4, use_colnames=True)
```

```
    frequent_itemsets =
frequent_itemsets.sort_values("support",
ascending=False)


    print("=== Frequent itemsets ===")
    print(frequent_itemsets)


    # 4) Sinh luật kết hợp từ frequent itemsets
    rules = association_rules(
        frequent_itemsets,
        metric="confidence",
        min_threshold=0.6
    )


    # 5) Lọc / sắp xếp (ví dụ lọc lift > 1)
    rules = rules[rules["lift"] >
1].sort_values(["confidence", "lift"],
ascending=False)


    print("\n=== Association rules (filtered) ===")
    print(rules[["antecedents", "consequents",
"support", "confidence", "lift"]])
```