CASE STUDY: PHÂN TÍCH GIỎ HÀNG

ThS. Lê Thị Thùy Trang

2025-12-13

1 Bán lẻ dựa trên dữ liệu (Data-Driven Retail)

1.1 Giới thiệu dự án

Dự án này tiếp nối dự án mà sinh viên đã thực hiện trong bài lab 1. Tương tự như việc áp dụng thuật toán Apriori, quy trình thuật toán tuân theo các bước sau:

1. Khám phá dữ liệu: Hiểu cấu trúc và phân bố của tập dữ liệu.
2. Thu thập và làm sạch dữ liệu giao dịch thô.
3. Chuyển đổi dữ liệu về dạng phù hợp cho thuật toán khai phá luật (danh sách các sản phẩm trong từng giỏ hàng).
4. Áp dụng FP-Growth: Tìm các tập phổ biến và luật với các ngưỡng tối ưu.
5. Phân tích kết quả: So sánh hiệu suất của thuật toán Apriori với thuật toán FP-Growth. Trực quan hoá các rules và đưa ra đề xuất kinh doanh.

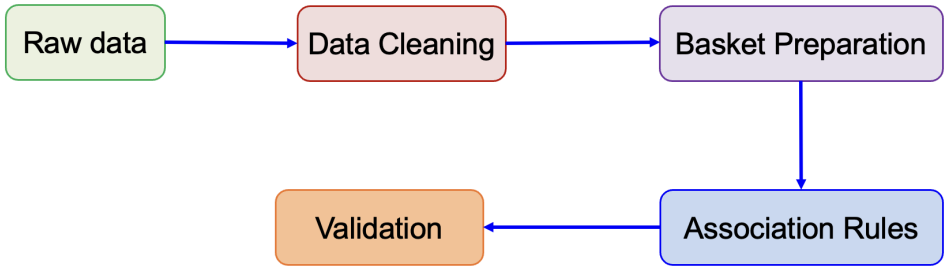

Figure 1: Pipeline project

1.2 Danh sách các module trong project

Dự án được tổ chức với các lớp (module) tương tự Lab 1 để đảm bảo tính rõ ràng và tái sử dụng.

Chúng ta tái sử dụng các lớp cũ cho những bước chung và bổ sung lớp mới cho FP-Growth. Cụ thể: `DataCleaner` (làm sạch dữ liệu) và `BasketPreparer` (chuẩn bị dữ liệu basket) giữ nguyên như trước.

Thay đổi chính nằm ở module khai phá luật: thay vì chỉ sử dụng `AssociationRulesMiner` dành cho Apriori, ta triển khai lớp mới `FPGrowthMiner`, thực hiện việc tìm tập phổ biến bằng thuật toán FP-Growth. Lớp này có thể có phương thức `run(min_support)` để trả về DataFrame các tập phổ biến kèm support. Sau đó, `FPGrowthMiner` có thể gọi tiếp hàm `association_rules` (tương tự Apriori) để sinh luật và tính các chỉ số cho mỗi luật. Lớp `DataVisualizer`

vẫn được dùng để trực quan hóa, không cần thay đổi vì đầu vào (danh sách luật kèm chỉ số) có định dạng giống nhau từ cả Apriori và FP-Growth.

Ngoài ra, pipeline mới sẽ có một notebook dành riêng cho FP-Growth (`fp_growth_modelling.ipynb`) tương tự `apriori_modelling.ipynb` trước đây, và notebook `compare_apriori_fpgrowth.ipynb` để so sánh hai thuật toán dựa trên các tiêu chí như: thời gian chạy, số lượng tập phổ biến, số lượng luật kết hợp sinh ra, độ dài trung bình của itemset. Đồng thời, script `run_papermill.py` có thể được điều chỉnh để tùy chọn chạy pipeline với thuật toán mong muốn (Apriori hoặc FP-Growth).

Thành phần	**Ý nghĩa và chức năng**
`src/apriori_library.py`	Thư viện trung tâm của dự án, chứa các lớp (class) xử lý chính: `DataCleaner` (làm sạch dữ liệu), `BasketPreparer` (tạo danh sách giỏ hàng), `AssociationRulesMiner` (thực hiện thuật toán Apriori, sinh luật kết hợp) và `DataVisualizer` (trực quan hoá).
`notebooks/preprocessing_and_eda.ipynb`	Notebook bước 1: Thực hiện tải dữ liệu thô, xử lý các giá trị thiếu, lọc dữ liệu rác (đơn hàng bị huỷ) và thực hiện phân tích khám phá dữ liệu (EDA) để hiểu tổng quan về tập dữ liệu.
`notebooks/basket_preparation.ipynb`	Notebook chuẩn bị dữ liệu giỏ hàng (basket) từ dữ liệu giao dịch gốc: lọc các đơn hàng hợp lệ, nhóm các sản phẩm theo từng đơn hàng và lưu ra định dạng phù hợp cho bước áp dụng thuật toán Apriori.
`notebooks/apriori_modelling.ipynb`	Notebook bước 3: Notebook này sử dụng ma trận `basket_bool` (được chuẩn bị ở bước 2) để: khai thác tập mục phổ biến bằng thuật toán Apriori; sinh luật kết hợp với các chỉ số `support, confidence, lift`.
`notebooks/fp_growth_modelling.ipynb`	Notebook này sử dụng ma trận `basket_bool` (được chuẩn bị ở bước 2) để: khai thác tập mục phổ biến bằng thuật toán FP-Growth.
`notebooks/compare_apriori_fpgrowth.ipynb`	Notebook này so sánh hai thuật toán trên cùng một `basket_bool` (được chuẩn bị ở bước 2).
`run_papermill.py`	Script điều phối toàn bộ pipeline: sử dụng thư viện `papermill` để tự động chạy lần lượt các notebook `preprocessing_and_eda.ipynb`, `basket_preparation.ipynb`, `apriori_modelling.ipynb` với các tham số cấu hình; lưu lại các notebook sau khi thực thi, ghi nhận thời gian chạy (execution time) và giúp bạn có thể tái chạy toàn bộ quy trình chỉ bằng một lệnh từ dòng lệnh.
`data/`	Thư mục chứa dữ liệu đầu vào (`online_retail.csv`) và lưu trữ các dữ liệu trung gian đã qua xử lý (processed data) để sử dụng giữa các bước phân tích.

Table 1: Bảng thành phần Project

1.3 Chuẩn bị môi trường thực hành

Kích hoạt môi trường ảo

```
conda activate shopping_env
```

Toàn bộ các thư viện cần thiết cho dự án được liệt kê trong file `requirements.txt`. Vì vậy, sau khi kích hoạt môi trường, cài đặt các thư viện cần thiết bằng câu lệnh:

```
pip install requirements.txt
```

2 FP-Growth vs Apriori

Về bản chất, FP-Growth và Apriori đều nhằm mục tiêu tìm tập sản phẩm hay xuất hiện cùng nhau vượt ngưỡng hỗ trợ, nhưng phương pháp tiếp cận khác nhau dẫn đến hiệu năng khác biệt.

Apriori duyệt qua không gian tập hợp bằng cách mở rộng dần các tập phổ biến (chiến lược bottom-up), phải sinh các tập ứng viên và quét dữ liệu nhiều lần. Mỗi `k-itemset` phổ biến được dùng để tạo ứng viên `(k+1)-itemset`, và bất kỳ ứng viên nào có tập con không phổ biến sẽ bị loại ngay (dựa trên nguyên lý Apriori). Cơ chế này đơn giản nhưng dễ bị bùng nổ tổ hợp: số lượng ứng viên tăng rất nhanh khi dữ liệu có nhiều mục phổ biến, khiến Apriori đòi hỏi bộ nhớ lớn và thời gian chạy cao.

Ngược lại, FP-Growth áp dụng phương pháp tree-based và chia để trị. Thuật toán chỉ quét cơ sở dữ liệu hai lần : lần 1 xác định các item phổ biến 1 phần tử và loại bỏ item hiếm, lần 2 xây dựng cây FP-Tree nén toàn bộ giao dịch. Cây FP-Tree lợi dụng tính chất: các giao dịch chia sẻ item chung sẽ dùng chung nhánh trên cây, do đó dữ liệu được nén lại đáng kể. Sau đó, FP-Growth khai thác FP-tree: mỗi nhánh của cây (tương ứng với một đường đi các sản phẩm thường cùng xuất hiện) được phân tích độc lập bằng cách lập cơ sở mẫu điều kiện và cây con điều kiện để tìm các tập phổ biến chứa một sản phẩm cụ thể. Cách làm này tránh tạo tập ứng viên trên toàn bộ không gian, giúp giảm mạnh chi phí. Nhờ cấu trúc cây nhỏ gọn, FP-Growth tiêu tốn ít bộ nhớ hơn Apriori dù dữ liệu lớn. Cũng vì không mất thời gian cho việc sinh và kiểm tra ứng viên mỗi mức, FP-Growth thường nhanh hơn Apriori rõ rệt về thời gian thực thi. Thậm chí, nghiên cứu ban đầu cho thấy FP-Growth có thể nhanh hơn Apriori một bậc độ lớn (order of magnitude) trên các bộ dữ liệu lớn và ngưỡng support thấp. Tuy nhiên, FP-Growth chỉ phát huy tối đa khi dữ liệu có cấu trúc phù hợp (nhiều mẫu lặp để nén cây). Trong trường hợp xấu nhất (ví dụ mỗi giao dịch hoàn toàn khác nhau, ít phần chung), cây FP có thể phình to gần bằng dữ liệu gốc và khi đó lợi thế sẽ giảm.

Tổng kết sự khác biệt: Apriori thuộc loại thuật toán sinh-tập-ứng-viên (join-based), trong khi FP-Growth thuộc loại thuật toán dựa trên cây (tree-based). Apriori dùng chiến lược tìm kiếm theo bề rộng tầng (mở rộng dần kích thước itemset), còn FP-Growth dùng chiến lược chia để trị theo chiều sâu trên cây FP. Apriori cần nhiều lần I/O quét dữ liệu, FP-Growth chỉ cần hai lần. Bộ nhớ Apriori tốn cho ứng viên trung gian, FP-Growth tiết kiệm bộ nhớ hơn nhờ cấu trúc cây nén. Nhờ vậy, FP-Growth thường vượt trội Apriori về tốc độ trên nhiều loại dữ liệu. Dẫu vậy, Apriori có ưu điểm là cài đặt tương đối đơn giản và trong một số trường hợp với ngưỡng rất cao (rất ít mẫu phổ biến) thì hiệu năng cũng có thể chấp nhận được. Ngược lại, FP-Growth với cấu trúc phức tạp có chi phí khởi động (xây dựng cây) nhưng sẽ lợi thế hơn khi bài toán yêu cầu độ sâu phân tích lớn (nhiều mẫu phổ biến kích thước lớn hoặc support thấp).

3 So sánh kết quả

Sau khi áp dụng FP-Growth với ngưỡng min_support tương tự Apriori (ví dụ 1%), ta thu được danh sách các itemset phổ biến cùng các luật kết hợp thỏa mãn `min_confidence` và `min_lift`đặt ra. Về mặt lý thuyết, FP-Growth và Apriori sẽ tìm ra cùng một tập phổ biến nếu dùng chung tham số, bởi cả hai đều đảm bảo trích xuất đầy đủ các mẫu thỏa ngưỡng. Thật vậy, trong thử nghiệm trên dữ liệu này, các luật kết hợp hàng đầu do FP-Growth sinh ra tương đồng với Apriori.

Sự khác biệt nằm ở hiệu suất: FP-Growth cho thời gian chạy nhanh hơn đáng kể. Với `min_support = 0.01`, Apriori mất khoảng vài chục giây để quét và sinh luật, trong khi FP-Growth chỉ mất vài giây. Khi hạ thấp ngưỡng hỗ trợ (ví dụ 0.5%), Apriori nhanh chóng trở nên chậm hoặc không khả thi do phải xét quá nhiều kết hợp, còn FP-Growth vẫn xử lý được nhờ cách nén dữ liệu hiệu quả. Điều này minh họa ưu điểm thực tế của FP-Growth đối với các bài toán cần đào sâu các "mẫu đuôi dài" (long-tail patterns) – những kết hợp hiếm nhưng giá trị.

4 Thực hành tại lớp

Q1: Sinh viên xem lại cấu trúc dự án của Lab 1 sau đó đọc hiểu và phân tích cấu trúc dự án với FP-Growth.

Q2: So sánh độ nhạy của tham số.

Một chủ đề thú vị là khảo sát độ nhạy của hai thuật toán đối với các tham số như `min_support` và `min_confidence`. Sinh viên giảm dần `min_support` và quan sát số lượng luật tìm được cũng như thời gian chạy của Apriori vs FP-Growth. Kỳ vọng: Apriori sẽ suy giảm hiệu năng mạnh khi min_support giảm (số luật tăng vọt nhưng nhiều luật nhiễu), trong khi FP-Growth có thể mở rộng tốt hơn. Điều này giúp hiểu rõ giới hạn của từng thuật toán và cách chọn tham số phù hợp. Tương tự, thay đổi `min_confidence` hay `min_lift` cũng ảnh hưởng đến số luật cuối cùng – so sánh kết quả giúp đánh giá thuật toán nào linh hoạt hơn trong việc tinh chỉnh đầu ra.

5 Hướng dẫn thực hiện hoạt động FIT-DNU CONQUER

5.1 Mục tiêu của hoạt động

Hoạt động FIT-DNU CONQUER được thiết kế nhằm giúp sinh viên:

- Hiểu sâu bản chất của khai phá luật kết hợp (Apriori) thông qua việc ứng dụng trên dữ liệu thực.
- Tập diễn giải kết quả bằng ngôn ngữ đơn giản (Feynman style), tránh chỉ mô tả code hoặc trích xuất số liệu.
- Rèn luyện khả năng trình bày, giải thích và thuyết phục thông qua hoạt động chia sẻ kết quả.
- Phát triển tư duy phân tích theo góc nhìn kinh doanh và đưa ra đề xuất hành động.

Sinh viên cần xem xét dự án như một nhiệm vụ Data Scientist thực thụ: không chỉ "chạy đúng thuật toán", mà phải chuyển dữ liệu thành tri thức hữu ích.

5.2 Yêu cầu

Đối với mỗi nhóm (3-4 sinh viên), yêu cầu thực hiện như sau:

1. **Đọc hiểu cấu trúc dự án và pipeline (Q1):**
   - Ôn lại pipeline của Lab 1 (Apriori): `preprocessing_and_eda.ipynb` → `basket_preparation.ipynb` → `apriori_modelling.ipynb`.
   - Đọc và phân tích cấu trúc dự án mở rộng với FP-Growth:
     - Các lớp: `DataCleaner`, `BasketPreparer`, `AssociationRulesMiner`, `FPGrowthMiner`, `DataVisualizer`.
     - Các notebook mới: `fp_growth_modelling.ipynb`, `compare_apriori_fpgrowth.ipynb`.
     - Vai trò của `run_papermill.py` trong việc điều phối pipeline tự động.
2. **Thực nghiệm Apriori và FP-Growth, so sánh độ nhạy tham số (Q2):**
   - Chạy Apriori và FP-Growth trên cùng một `basket_bool` với các giá trị `min_support`, `min_confidence`, `min_lift` khác nhau.
   - Quan sát và so sánh:
     - Số lượng tập phổ biến và số lượng luật sinh ra.
     - Thời gian chạy của Apriori vs FP-Growth khi giảm dần `min_support`.
     - Độ dài trung bình của itemset và chất lượng luật (support, confidence, lift).
   - Rút ra nhận xét về độ nhạy tham số và giới hạn của từng thuật toán.
3. **Trực quan hóa kết quả khai phá luật:**
   - Xây dựng *tối thiểu 02* biểu đồ để minh hoạ kết quả, ví dụ:
     - Bar chart phân bố `support`/`lift` của các luật nổi bật.

  - Scatter plot (trục X: `support`, trục Y: `confidence`) để so sánh Apriori và FP-Growth.
  - Network graph biểu diễn quan hệ giữa các sản phẩm trong luật.
- Với mỗi biểu đồ, cần có phần mô tả ngắn:
  - Biểu đồ thể hiện điều gì về hành vi mua sắm.
  - Điểm khác biệt (nếu có) giữa Apriori và FP-Growth.

4. **Insight kinh doanh từ luật kết hợp:**
   - Trích xuất và diễn giải *tối thiểu 05 insight* có ý nghĩa, dựa trên các luật mạnh (support, confidence, lift phù hợp).
   - Mỗi insight cần trả lời rõ câu hỏi:

     "Nếu là quản lý cửa hàng, từ luật này tôi sẽ làm gì?"
   - Ví dụ: gợi ý trưng bày chung, combo khuyến mãi, gợi ý mua thêm (cross-selling), điều chỉnh tồn kho theo mùa vụ, v.v.
5. **Báo cáo dạng blog/report:**
   - Viết báo cáo dưới dạng blog (Notion/GitHub) với mạch trình bày rõ ràng, súc tích:
     - Bài toán và dữ liệu.
     - Pipeline Apriori & FP-Growth.
     - Kết quả chính, trực quan hóa và insight.
     - So sánh và kết luận.
   - Tránh trình bày dưới dạng *dump code* hoặc copy toàn bộ bảng luật:
     - Chỉ trích dẫn những đoạn code/luật thật sự cần để minh họa.
     - Tập trung vào câu chuyện dữ liệu và ý nghĩa kinh doanh.
   - Ngôn ngữ đơn giản, hướng tới người đọc không chuyên về data mining.
6. **Trình bày và chia sẻ kết quả tại lớp (FIT-DNU CONQUER):**
   - Thời lượng trình bày: **5–7 phút/nhóm**.
   - Không đọc code, không đọc báo cáo; tập trung:
     - Giới thiệu bài toán và mục tiêu nhóm.
     - Tóm tắt pipeline và kết quả nổi bật.
     - Diễn giải các biểu đồ và insight kinh doanh.
     - Nêu rõ so sánh Apriori vs FP-Growth và bài học rút ra.
   - Áp dụng **Feynman style**: giải thích như đang dạy cho một người bạn chưa biết về khai phá luật kết hợp.

5.3 Phát triển dự án

5.3.1 Luật kết hợp có trọng số

Trong Lab 1 và Lab 2, các luật kết hợp đều được xây dựng dựa trên tần suất xuất hiện của sản phẩm trong giỏ hàng:

- *Support* đo xem một tập sản phẩm xuất hiện trong bao nhiêu phần trăm giao dịch.
- *Confidence*, *Lift* cũng đều dựa trên số lần xuất hiện.

Tuy nhiên, trong bối cảnh bán lẻ thực tế, không phải giao dịch nào cũng "quan trọng như nhau":

- Một hoá đơn trị giá 5 USD và một hoá đơn trị giá 500 USD nếu đếm như nhau sẽ làm mất ý nghĩa kinh doanh.
- Một sản phẩm rẻ tiền (ví dụ: thiệp, túi giấy) có thể xuất hiện rất thường xuyên nhưng đóng góp doanh thu thấp.
- Một bộ quà tặng đắt tiền có thể xuất hiện rất ít, nhưng mỗi lần bán ra mang lại lợi nhuận lớn.

Luật kết hợp có trọng số (Weighted Association Rules) ra đời để giải quyết vấn đề đó bằng cách gán cho mỗi giao dịch hoặc mỗi sản phẩm một *trọng số* phản ánh tầm quan trọng kinh doanh, ví dụ:

- Trọng số theo doanh thu (tổng tiền hoá đơn).
- Trọng số theo lợi nhuận (doanh thu trừ chi phí).
- Trọng số theo mức độ ưu tiên kinh doanh (sản phẩm chiến lược, sản phẩm mới cần đẩy mạnh, …).

Ý tưởng trực giác:

> "Một luật xuất hiện ít lần nhưng chủ yếu trong các hoá đơn giá trị cao có thể quan trọng hơn nhiều so với luật xuất hiện rất thường xuyên nhưng chỉ trong các giỏ hàng nhỏ."

5.3.1.1 Các khái niệm cơ bản

Giả sử ta gán cho mỗi giao dịch T một trọng số $w(T)$ (ví dụ: tổng doanh thu của hoá đơn đó). Khi đó:

- Weighted support của một tập sản phẩm X được định nghĩa xấp xỉ:

$$ws(X) = \frac{\sum\limits_{T \supseteq X} w(T)}{\sum\limits_{T} w(T)}$$

- Tương tự, ta có thể định nghĩa weighted confidence và weighted lift dựa trên weighted support.

Như vậy, thay vì "đếm số lần xuất hiện", ta cộng tổng trọng số của các giao dịch chứa tập X. Tập luật nào xuất hiện trong các hoá đơn giá trị cao sẽ có weighted support lớn, dù support thường có thể không cao.

5.3.1.2 Chiến lược áp dụng trong project

Trong phạm vi project này, sinh viên không nhất thiết phải cài đặt một thuật toán hoàn toàn mới. Ta có thể tận dụng pipeline hiện có (Apriori / FP-Growth) và bổ sung bước tính toán trọng số:

1. Bước 1: Gán trọng số cho giao dịch
   - Từ dữ liệu thô, tính thêm một cột:

     $$\texttt{InvoiceValue} = \sum(\texttt{Quantity} \times \texttt{UnitPrice})$$

     cho mỗi hoá đơn.
   - Xem `InvoiceValue` như trọng số $w(T)$ của giao dịch T.
2. Bước 2: Khai phá luật như bình thường
   - Dùng Apriori hoặc FP-Growth để sinh tập phổ biến và luật với support thường.
   - Kết quả thu được là một `DataFrame` các luật kèm support, confidence, lift (như trong Lab 1, Lab 2).
3. Bước 3: Bổ sung các chỉ số có trọng số
   - Với mỗi luật $X \Rightarrow Y$, tìm tất cả các hoá đơn chứa $X \cup Y$ và cộng tổng `InvoiceValue`.
   - Từ đó tính:
     - `weighted_support` (tỷ lệ giá trị hoá đơn chứa luật trên tổng giá trị toàn bộ hoá đơn).
     - `weighted_lift` (so sánh so với kỳ vọng nếu các sản phẩm độc lập, nhưng theo giá trị).

- Thêm các cột này vào bảng luật để so sánh trực tiếp:
  - support vs weighted_support,
  - lift vs weighted_lift.

Gợi ý thuật toán

Tuỳ mức độ tham vọng, sinh viên có thể chọn một trong các hướng sau:

5.3.1.3 Dễ như ăn kẹo thì điểm vừa tầm – *Post-processing có trọng số*

- Giữ nguyên thuật toán tìm tập phổ biến:
  - Apriori (Lab 1),
  - hoặc FP-Growth (Lab 2).
- Tập trung vào bước hậu xử lý: viết thêm hàm trong module `AssociationRulesMiner` (hoặc một lớp mới) để:
  - Đọc bảng luật đã sinh,
  - Ghép lại với bảng hoá đơn có trọng số,
  - Tính thêm các chỉ số weighted_support, weighted_lift, …
- Đây là cách dễ triển khai, giúp sinh viên tập trung vào diễn giải sự khác biệt giữa luật thường và luật có trọng số.

5.3.1.4 Đỡ dễ hơn thì điểm cao hơn – *Weighted Apriori / Weighted FP-Growth đơn giản*

- Mở rộng lớp `FPGrowthMiner` hoặc `AssociationRulesMiner`:
  - Khi tính support cho một itemset, thay vì tăng đếm 1 mỗi khi itemset xuất hiện, thì cộng thêm $w(T)$.
  - Chỉ giữ lại các itemset có weighted_support $\geq$ một ngưỡng do nhóm lựa chọn.
- Hướng này phù hợp với các nhóm muốn thử thách bản thân trong khuôn khổ FIT-DNU CONQUER:
  - Có thể bắt đầu từ phiên bản đơn giản (chỉ áp dụng cho một vài tập phổ biến đã biết).
  - Không cần tối ưu quá nhiều về hiệu năng, nhưng phải **giải thích rõ ý tưởng và giới hạn** của cách làm.

5.3.1.5 Cho nhóm tham vọng lấy 10 – *High-utility itemset mining*

- Dành cho nhóm có nền tảng tốt, có thể tham khảo thêm các thuật toán *high-utility itemset mining* (tập phổ biến giá trị cao).
- Ý tưởng chung: thay vì tối ưu theo số lần xuất hiện, tối ưu theo tổng "utility" (doanh thu/lợi nhuận).
- Nếu nhóm chọn hướng này, nên:
  - Bắt đầu từ một thư viện hoặc hiện thực đơn giản,
  - Tập trung viết blog giải thích sự khác biệt về tư duy giữa "frequent" và "high-utility".

5.3.2 Chủ đề gợi ý

Mỗi nhóm chọn một chủ đề dưới đây và so sánh/đối chiếu giữa:

- Luật kết hợp thông thường (dựa trên số lần xuất hiện).
- Luật kết hợp có trọng số (dựa trên giá trị đơn hàng, doanh thu, lợi nhuận, …).

5.3.2.1 Chủ đề 1: Phân tích luật theo thời gian kết hợp trọng số (Temporal + Weighted Association)

- Chia dữ liệu theo tháng/quý hoặc các giai đoạn kinh doanh quan trọng.
- Với mỗi giai đoạn, so sánh:
  - Số lượng luật và độ mạnh của luật *thông thường*.
  - Số lượng luật và độ mạnh của luật *có trọng số* (weighted support, weighted lift).
- Nhận định "mùa vụ" không chỉ theo tần suất mua, mà còn theo giá trị mang lại (doanh thu/lợi nhuận) của các combo sản phẩm.
- Thảo luận: khi dùng trọng số, các mùa vụ/đỉnh doanh thu có thay đổi so với khi chỉ nhìn support thường hay không.

5.3.2.2 Chủ đề 2: Tìm sản phẩm trung tâm theo giá trị (Value-based Product Hub)

- Từ tập luật *không trọng số*, xác định các sản phẩm xuất hiện nhiều nhất (hub theo tần suất).
- Từ tập luật *có trọng số*, xác định các sản phẩm đóng góp tổng doanh thu/lợi nhuận lớn nhất trong các luật.
- So sánh:
  - Hub theo tần suất vs hub theo giá trị.
  - Sản phẩm nào nên được xem là "ngôi sao doanh thu" dù tần suất xuất hiện không nhiều.
- Đề xuất chiến lược bố trí hàng hoá, gợi ý mua thêm (cross-selling) ưu tiên theo hub giá trị thay vì chỉ theo tần suất.

5.3.2.3 Chủ đề 3: Đánh giá luật theo Lift, trọng số và giá trị kinh doanh

- Tạo hai bảng xếp hạng luật:
  - Theo lift và support *thông thường*.
  - Theo `weighted lift`, `weighted support` (gắn trọng số theo doanh thu/giá trị đơn hàng).
- Phân loại luật thành các nhóm:
  1. Support cao, Lift cao, giá trị (doanh thu) cũng cao.
  2. Support không quá cao nhưng weighted support/lợi nhuận cao (luật "đắt tiền").
  3. Confidence cao nhưng Lift ≈ 1 hoặc giá trị mang lại thấp (ít giá trị thực tế).
- Đề xuất chiến lược marketing tương ứng cho từng nhóm, nhấn mạnh:
  - Luật nào nên dùng cho đại đa số khách hàng.
  - Luật nào nên dùng cho phân khúc khách hàng cao cấp/đơn hàng lớn.

5.3.2.4 Chủ đề 4: Phân tích độ nhạy tham số với và không có trọng số (Parameter Sensitivity)

- Thực nghiệm nhiều giá trị `min_support`, `min_confidence`, `min_lift` cho:
  - Luật thường.
  - Luật có trọng số (ví dụ: `min_weighted_support`, `min_weighted_lift`).
- Quan sát sự thay đổi về:
  1. Số lượng luật.
  2. Cấu trúc/cụm sản phẩm chính.

3. Sự xuất hiện biến mất của các luật có giá trị kinh doanh cao.

- Rút ra "ngưỡng hợp lý" cho cả hai trường hợp:
  - Khi mục tiêu là khai thác hành vi mua phổ biến.
  - Khi mục tiêu là tối đa hoá giá trị/doanh thu.

5.3.2.5 Chủ đề 5: Phân tích bằng Network Graph có trọng số

- Xây dựng network graph cho luật thông thường và luật có trọng số:
  - Node: sản phẩm.
  - Edge: luật kết hợp, độ đậm/dày của cạnh thể hiện support hoặc weighted support.
- So sánh hai đồ thị:
  - Cụm sản phẩm theo tần suất vs cụm sản phẩm theo giá trị.
  - Sản phẩm/cụm nào trông "bình thường" ở graph thường nhưng trở nên rất quan trọng khi xét trọng số.
- Nhận diện nhóm sản phẩm nên trưng bày cạnh nhau hoặc tạo combo **ưu tiên theo giá trị**, không chỉ theo mức độ xuất hiện chung.

5.3.2.6 Chủ đề 6: Nhóm sản phẩm và trọng số theo danh mục (Category-level Weighted Insight)

- Gán danh mục (category) cho sản phẩm (quà tặng, trang trí, gia dụng, …).
- Đối với từng cặp danh mục, so sánh:
  - Mức độ kết hợp theo số lượng luật.
  - Mức độ kết hợp theo tổng doanh thu/lợi nhuận của các luật liên quan.
- Nhận định:
  - Nhóm danh mục nào "ồn ào" (nhiều luật nhưng giá trị thấp).
  - Nhóm danh mục nào "trầm nhưng chất" (ít luật nhưng giá trị cao).
- Đề xuất các chiến dịch marketing theo danh mục ưu tiên dựa trên trọng số.

5.3.2.7 Chủ đề 7: Luật "niche' ' có trọng số (Hiếm nhưng giá trị cao)

- Tìm các luật có support thấp nhưng:
  - Weighted support cao, hoặc
  - Weighted lift rất cao.
- Phân tích lý do các luật này đáng chú ý:
  - Nhắm tới phân khúc khách hàng đặc biệt (VIP, đơn hàng lớn).
  - Liên quan tới sản phẩm giá cao, bộ sưu tập theo mùa, set quà tặng premium, …
- Đề xuất chiến lược bán hàng chuyên sâu theo phân khúc (niche marketing):
  - Gợi ý gói combo cao cấp.
  - Chương trình chăm sóc khách hàng riêng dựa trên các luật niche.

5.4 Kết quả kỳ vọng

Mỗi nhóm cần hoàn thành:

- Blog/Report (link Notion/GitHub).
- Slide trình bày.

5.5 Gợi ý cho phần trình bày tại lớp

1. Giới thiệu bài toán và mục tiêu nhóm.
2. Trình bày kết quả trọng tâm (không kể lể, không giải thích code).
3. Diễn giải các biểu đồ.
4. Kết luận và đề xuất hành động.