**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO KHAI PHÁ DỮ LIỆU**

**(MSHP: 220269)**

TÊN ĐỀ TÀI

**PHÂN TÍCH LUẬT KẾT HỢP**

**TRONG DỮ LIỆU BÁN HÀNG**

**Sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| 110122008 | Phạm Hoàng Kha | DA22TTA |
| 110122221 | Nguyễn Thanh Hiếu | DA22TTA |
| 110122041 | Nguyễn Trí Cường | DA22TTA |
| 110122032 | Bùi Quốc Anh | DA22TTA |
| 110122048 | Mai Tuấn Đạt | DA22TTA |

**Giáo viên hướng dẫn:** ThS. Nguyễn Thái Toàn

**Trà Vinh, tháng 6 năm 2025**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Nhóm chúng tôi xin chân thành gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Thái Toàn – Giảng viên Trường Kỹ thuật và Công nghệ, Trường Đại học Trà Vinh – người đã tận tình hướng dẫn, hỗ trợ và định hướng cho nhóm trong suốt quá trình thực hiện đề tài "Phân tích luật kết hợp trong dữ liệu bán hàng".

Nhờ sự chỉ dẫn tận tâm, kiến thức chuyên môn vững vàng cùng những góp ý quý báu từ Thầy, nhóm đã có thể hoàn thành đề tài một cách hiệu quả và tích lũy thêm nhiều kinh nghiệm thực tiễn cũng như kiến thức chuyên sâu về lĩnh vực khai phá dữ liệu.

Bên cạnh đó, nhóm cũng xin gửi lời cảm ơn chân thành đến quý thầy cô Trường Đại học Trà Vinh đã truyền đạt kiến thức nền tảng quý báu trong suốt quá trình học tập, tạo điều kiện thuận lợi để nhóm có thể vận dụng vào đề tài một cách hiệu quả.

Nhóm cũng xin trân trọng cảm ơn gia đình và bạn bè đã luôn động viên, khích lệ và hỗ trợ tinh thần, tạo môi trường thuận lợi giúp nhóm tập trung nghiên cứu và hoàn thành đề tài này.

Một lần nữa, nhóm xin chân thành cảm ơn Thầy và kính chúc Thầy luôn mạnh khỏe, thành công trong công tác giảng dạy và nghiên cứu khoa học.

**Kí tên**

|  |  |  |
| --- | --- | --- |
| **Sinh viên 1** | **Sinh viên 2** | **Sinh viên 3** |
|  |  |  |
| **Nguyễn Thanh Hiếu** | **Phạm Hoàng Kha** | **Nguyễn Trí Cường** |

|  |  |
| --- | --- |
| **Sinh viên 4** | **Sinh viên 5** |
|  |  |
| **Mai Tuấn Đạt** | **Bùi Quốc Anh** |

**MỤC LỤC**

|  |  |
| --- | --- |
| Danh mục | Trang |

[CHƯƠNG 1 TỔNG QUAN 9](#_Toc201300512)

[1.1 Đặt vấn đề 9](#_Toc201300513)

[1.2 Cơ sở hình thành đề tài 9](#_Toc201300514)

[1.3 Mục tiêu đề tài 10](#_Toc201300515)

[1.4 Phạm vi nghiên cứu 11](#_Toc201300516)

[1.5 Đối tượng nghiên cứu 12](#_Toc201300517)

[1.6 Phương pháp nghiên cứu 13](#_Toc201300518)

[1.6.1 Áp dụng mô hình CRISP-DM 13](#_Toc201300519)

[1.6.2 Phương pháp thu thập và xử lý dữ liệu 14](#_Toc201300520)

[1.7 Ý nghĩa 15](#_Toc201300521)

[1.7.1 Ý nghĩa khoa học 15](#_Toc201300522)

[1.7.2 Ý nghĩa thực tiễn 15](#_Toc201300523)

[CHƯƠNG 2 : CƠ SỞ LÍ THUYẾT 17](#_Toc201300524)

[2.1 Tìm hiểu về ngôn ngữ lập trình 17](#_Toc201300525)

[2.1.1 Python là gì? 17](#_Toc201300526)

[2.1.2 Đặc trưng 17](#_Toc201300527)

[2.2 Visual code và công cụ hỗ trợ 18](#_Toc201300528)

[2.2.1 Giới thiệu về Visual Studio Code 18](#_Toc201300529)

[2.2.2 Thư viện Pandas 20](#_Toc201300530)

[2.2.3 Streamlit 21](#_Toc201300531)

[2.2.4 Grapviz 22](#_Toc201300532)

[2.3 Các thuật toán khai phá luật kết hợp 23](#_Toc201300533)

[2.3.1 Thuật toán Apriori 23](#_Toc201300534)

[2.3.2 Thuật toán FP-Growth (Frequent Pattern Growth) 26](#_Toc201300535)

[2.3.3 So sánh lý thuyết giữa Apriori và FP-Growth 29](#_Toc201300536)

[CHƯƠNG 3 PHÂN TÍCH VÀ THIẾT KẾ 32](#_Toc201300537)

[3.1 Phân tích bộ dữ liệu “One Retail” 32](#_Toc201300538)

[3.1.1 Mô tả các trường dữ liệu 32](#_Toc201300539)

[3.1.2 Tiền xử lý dữ liệu (Data Preprocessing) 32](#_Toc201300540)

[3.2 Áp dụng thuật toán Apriori 33](#_Toc201300541)

[3.2.1 Tổng quan về cách triển khai 33](#_Toc201300542)

[3.2.2 Cấu hình tham số và chuẩn hóa dữ liệu đầu vào 33](#_Toc201300543)

[3.2.3 Kết quả khai phá luật kết hợp 34](#_Toc201300544)

[3.2.4 Phân tích sâu các đặc điểm của luật khai phá 34](#_Toc201300545)

[3.3 Áp dụng thuật toán FP-Growth 35](#_Toc201300546)

[3.3.1 Tổng quan về cách triển khai 35](#_Toc201300547)

[3.3.2 Cấu hình tham số và chuẩn hóa dữ liệu đầu vào 36](#_Toc201300548)

[3.3.3 Kết quả khai phá luật kết hợp 37](#_Toc201300549)

[3.3.4 Phân tích sâu các đặc điểm của luật khai phá 37](#_Toc201300550)

[3.4 Thiết kế và cài đặt ứng dụng 38](#_Toc201300551)

[3.4.1 Kiến trúc tổng quan của ứng dụng 38](#_Toc201300552)

[3.4.2 Giao diện người dùng: 42](#_Toc201300553)

[CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM 49](#_Toc201300554)

[4.1 Thiết lập thực nghiệm 49](#_Toc201300555)

[4.2 Biện luận 50](#_Toc201300556)

[4.3 So sánh và đánh giá giữa hai thuật toán FP\_Growth và Apriori 57](#_Toc201300557)

[4.3.1 So sánh thực nghiệm trên cùng bộ dữ liệu 57](#_Toc201300558)

[4.3.2 Phân tích lí thuyết và thực tiễn 58](#_Toc201300559)

[4.3.3 Giải thích nguyên nhân 60](#_Toc201300560)

[CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT 62](#_Toc201300561)

[5.1 Tổng kết các kết quả đạt được 62](#_Toc201300562)

[5.2 Hạn chế 62](#_Toc201300563)

[5.3 Hướng phát triển 63](#_Toc201300564)

**DANH MỤC HÌNH ẢNH**

|  |  |
| --- | --- |
| Danh mục | Trang |

[Hình 2.1 Hình ảnh liên quan đến ngôn ngữ python 17](#_Toc201300356)

[Hình 2.2 Hình ảnh về VS Code 19](#_Toc201300357)

[Hình 2.3 Hình ảnh minh họa thêm về VS Code 20](#_Toc201300358)

[Hình 2.4 Hình ảnh về thư viện Pandas 20](#_Toc201300359)

[Hình 2.5 Hình ảnh về Streamlit 22](#_Toc201300360)

[Hình 2.6 Hình ảnh minh họa bước 1 của Apriori 24](#_Toc201300361)

[Hình 2.7 Hình ảnh minh họa bước 2 của Apriori 25](#_Toc201300362)

[Hình 2.8 Hình ảnh minh họa bước 3 của Apriori 25](#_Toc201300363)

[Hình 2.9 Hình ảnh minh họa bước 0 của FP\_Growth 27](#_Toc201300364)

[Hình 2.10 Hình ảnh minh họa bước 1 của FP\_Growth 28](#_Toc201300365)

[Hình 2.11 Hình ảnh minh họa thêm bước 1 của FP\_Growt 28](#_Toc201300366)

[Hình 3.1 Mối quan hệ giữa các modules 42](#_Toc201300367)

[Hình 3.2 Giao diện trực quan của ứng dụng của Apriori 43](#_Toc201300368)

[Hình 3.3 Giao diện trực quan của ứng dụng của FP\_Growth 43](#_Toc201300369)

[Hình 3.4 Ứng dụng cho phép tải dữ liệu từ máy tính 44](#_Toc201300370)

[Hình 3.5 Ứng dụng cho phép nhập dữ liệu trực tiếp theo nhiều định dạng 45](#_Toc201300371)

[Hình 3.6 Ứng dụng cho phép tùy chọn để lọc dữ liệu 46](#_Toc201300372)

[Hình 3.7 Ứng dụng linh hoạt bằng cách cho phép điều chỉnh tham số 47](#_Toc201300373)

[Hình 3.8 Giao diện của thuật toán FP\_Growth 48](#_Toc201300374)

[Hình 3.8 Giao diện của thuật toán Apriori 48](#_Toc201300375)

[Hình 4.1 Thông số khi triển khai vào Apriori 51](#_Toc201300376)

[Hình 4.1 Thông số khi triển khai vào FT\_Growth 51](#_Toc201300377)

[Hình 4.3 Ứng dụng hiển thị các bước thực hiện của Apriori 52](#_Toc201300378)

[Hình 4.4 Mỗi bước sẽ hiển thị chi tiết về thông số apriori sử dụng 52](#_Toc201300379)

[Hình 4.5 Ứng dụng hiển thị các bước thực hiện của FP\_Growth 53](#_Toc201300380)

[Hình 4.6 Mỗi bước sẽ hiển thị chi tiết về thông số FP\_Growth sử dụng 53](#_Toc201300381)

[Hình 4.7 Ứng dụng hiển thị tập phổ biến mà FP\_Growth tìm được 54](#_Toc201300382)

[Hình 4.8 Ứng dụng hiển thị tập phổ biến mà Apriori tìm được 54](#_Toc201300383)

[Hình 4.9 Ứng dụng ghi lại hiệu năng của FP\_Growth 55](#_Toc201300384)

[Hình 4.10 Ứng dụng ghi lại hiệu năng của FP\_Growth 55](#_Toc201300385)

**DANH MỤC BẢNG BIỂU**

|  |  |
| --- | --- |
| Danh mục | Trang |

[Bảng 1 Thông số giữa 2 thuật toán apriori 57](#_Toc201300241)

[Bảng 2 Tổng hợp so sánh giữa 2 thuật toán 60](#_Toc201300242)

[Bảng 3 So sánh thực tế giữa hai thuật toán 61](#_Toc201300243)

# TỔNG QUAN

## Đặt vấn đề

Trong thời đại công nghệ số và thương mại điện tử phát triển mạnh mẽ, dữ liệu giao dịch của khách hàng ngày càng trở nên phong phú và đa dạng. Việc khai thác thông tin từ dữ liệu này để hiểu rõ hành vi mua sắm, từ đó đưa ra các chiến lược tiếp thị và gợi ý sản phẩm phù hợp, là một nhu cầu thực tế và cấp thiết đối với doanh nghiệp.

Phân tích giỏ hàng (Market Basket Analysis) là một kỹ thuật quan trọng trong lĩnh vực khai phá dữ liệu, cho phép phát hiện các mối quan hệ tiềm ẩn giữa các mặt hàng mà khách hàng thường mua cùng nhau. Các kết quả phân tích có thể được ứng dụng trong việc tối ưu hóa cách sắp xếp sản phẩm, đề xuất mua hàng, hoặc thiết kế chương trình khuyến mãi.

Hai thuật toán phổ biến và hiệu quả được sử dụng trong phân tích giỏ hàng là Apriori và FP-Growth. Việc nghiên cứu, triển khai và so sánh hai thuật toán này không chỉ giúp hiểu rõ bản chất và cách hoạt động của chúng, mà còn cung cấp cái nhìn thực tiễn về hiệu quả của từng phương pháp trong xử lý dữ liệu giao dịch.

Với những lý do trên, nhóm em quyết định thực hiện đề tài “Phân tích giỏ hàng và khai phá luật kết hợp sử dụng thuật toán Apriori và FP-Growth” nhằm tiếp cận sâu hơn với lĩnh vực khai phá dữ liệu và ứng dụng vào các tình huống thực tiễn.

## Cơ sở hình thành đề tài

Việc lựa chọn đề tài “Phân tích giỏ hàng và khai phá luật kết hợp sử dụng thuật toán Apriori và FP-Growth” được xây dựng dựa trên các cơ sở sau:

Nhu cầu thực tiễn của doanh nghiệp: Trong bối cảnh cạnh tranh ngày càng gay gắt của thị trường, các siêu thị, chuỗi bán lẻ và nền tảng thương mại điện tử tại Việt Nam đối mặt với nhu cầu cấp thiết trong việc tối ưu hóa chiến lược trưng bày sản phẩm, xây dựng chương trình khuyến mãi hiệu quả và phát triển các hệ thống gợi ý mua hàng tự động. Phân tích giỏ hàng, thông qua việc khám phá các mẫu mua sắm của khách hàng, cung cấp cơ sở khoa học để các doanh nghiệp nâng cao doanh thu và cải thiện trải nghiệm người tiêu dùng, từ đó khẳng định vị thế cạnh tranh.

Cơ sở lý thuyết từ khai phá dữ liệu: Luật kết hợp, một nhánh quan trọng trong lĩnh vực khai phá dữ liệu, đã được nghiên cứu và ứng dụng rộng rãi kể từ những năm 1990, với các công trình tiêu biểu như thuật toán Apriori của Agrawal và Srikant (1994) và thuật toán FP-Growth của Han et al. (2000). Hai thuật toán này tạo nên nền tảng lý thuyết vững chắc, cho phép trích xuất các quy tắc kết hợp từ dữ liệu giao dịch một cách hiệu quả, đặc biệt trong bối cảnh phân tích hành vi mua sắm.

Khoảng trống trong nghiên cứu và triển khai: Dẫu các nghiên cứu về thuật toán Apriori đã khá phổ biến, những hạn chế liên quan đến tốc độ xử lý và hiệu suất khi ứng dụng trên tập dữ liệu lớn vẫn còn tồn tại, đòi hỏi sự cải tiến. Thuật toán FP-Growth, với việc sử dụng cấu trúc cây FP (FP-tree), đã khắc phục một phần những nhược điểm này; tuy nhiên, nó đòi hỏi thiết kế cấu trúc dữ liệu phức tạp và việc điều chỉnh ngưỡng hỗ trợ (support) một cách hợp lý. Hiện nay, các nghiên cứu so sánh trực tiếp hiệu quả của hai thuật toán trên cùng một tập dữ liệu thực tế, kết hợp đánh giá các chỉ số như support, confidence và lift, vẫn còn hạn chế, đặc biệt trong các tài liệu bằng tiếng Việt.

Tiềm năng ứng dụng công nghệ hiện đại: Sự phát triển của các công cụ lập trình như Python, cùng với các thư viện hỗ trợ như mlxtend và các framework trực quan hóa như Streamlit, đã tạo điều kiện thuận lợi để triển khai và biểu diễn kết quả nghiên cứu một cách nhanh chóng và sinh động. Kết quả từ đề tài không chỉ mang tính lý thuyết mà còn có khả năng được phát triển thành các công cụ hỗ trợ ra quyết định thực tiễn, phục vụ trực tiếp cho các doanh nghiệp trong việc phân tích dữ liệu giỏ hàng.

## Mục tiêu đề tài

Trong bối cảnh dữ liệu giao dịch ngày càng phong phú và đóng vai trò trọng yếu trong việc định hình chiến lược kinh doanh, nhu cầu khai thác tri thức từ dữ liệu để hỗ trợ ra quyết định trở nên cấp thiết. Đặc biệt, các doanh nghiệp bán lẻ, thương mại điện tử, siêu thị và chuỗi cửa hàng đang tìm kiếm những cách tiếp cận hiệu quả nhằm phát hiện các mối quan hệ ẩn giữa các mặt hàng thông qua hành vi mua sắm của khách hàng.

Trên cơ sở đó, đề tài hướng đến việc ứng dụng các thuật toán khai phá luật kết hợp – cụ thể là Apriori và FP-Growth – để trích xuất những luật kết hợp có ý nghĩa từ tập dữ liệu giao dịch, giúp hỗ trợ các hoạt động như đề xuất sản phẩm, xây dựng chương trình khuyến mãi và tối ưu hóa bố trí hàng hóa.

Mục tiêu cụ thể của đề tài bao gồm:

* Xử lý, làm sạch và chuẩn hóa bộ dữ liệu "Online Retail", bao gồm loại bỏ dữ liệu thiếu, chuẩn hóa định danh sản phẩm và chuyển đổi định dạng giao dịch;
* Phân tích lý thuyết và làm rõ cơ chế hoạt động của hai thuật toán phổ biến trong khai phá luật kết hợp là Apriori và FP-Growth, từ nguyên lý thuật toán, cấu trúc dữ liệu sử dụng, đến ưu nhược điểm của mỗi phương pháp.
* Xây dựng quy trình thực nghiệm chuẩn hóa theo mô hình CRISP-DM để áp dụng đồng thời cả hai thuật toán trên cùng một tập dữ liệu, đảm bảo tính nhất quán và khoa học trong quá trình xử lý.
* Khai phá luật kết hợp với ngưỡng hỗ trợ (support) và độ tin cậy (confidence) đã định nghĩa, qua đó thu được tập luật phản ánh xu hướng tiêu dùng thực tế của khách hàng trong dữ liệu.
* So sánh hiệu năng của hai thuật toán dựa trên các khía cạnh: thời gian xử lý, khả năng mở rộng với dữ liệu lớn, số lượng và chất lượng luật, cũng như độ hiệu quả trong ứng dụng thực tiễn.
* Xây dựng một ứng dụng phần mềm hoặc giao diện tương tác bằng python và : cho phép người dùng tải dữ liệu, thực thi hai thuật toán và trực quan hóa kết quả bằng bảng, biểu đồ hoặc danh sách luật kết hợp.
* Đề xuất những ứng dụng cụ thể của luật kết hợp khai thác được vào các hoạt động vận hành doanh nghiệp, chẳng hạn như hệ thống gợi ý sản phẩm (recommendation systems), thiết kế chính sách giảm giá theo nhóm sản phẩm thường được mua cùng nhau, và quản lý hàng tồn kho.

Từ những mục tiêu trên, đề tài không chỉ cung cấp một cái nhìn so sánh giữa hai thuật toán phổ biến trong lĩnh vực khai phá dữ liệu, mà còn góp phần minh chứng vai trò thiết thực của các kỹ thuật dữ liệu trong việc nâng cao hiệu quả quản lý và hỗ trợ ra quyết định dựa trên dữ liệu.

## Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài được xác định rõ ràng nhằm đảm bảo tính khả thi, đồng thời tập trung vào các nội dung cốt lõi phục vụ mục tiêu khai phá và so sánh hai thuật toán:

* **Tập dữ liệu nghiên cứu**: là dữ liệu giao dịch dạng giỏ hàng (transactional data), bao gồm nhiều hóa đơn, mỗi hóa đơn chứa một tập hợp sản phẩm được mua cùng lúc. Dữ liệu có thể được lấy từ các nguồn mô phỏng (như tập dữ liệu Groceries của hệ thống DUNNHUMBY) hoặc tập dữ liệu thực tế được chuẩn hóa.
* **Các phương pháp khai phá**: chỉ tập trung vào hai thuật toán cổ điển là Apriori và FP-Growth – đại diện cho hai cách tiếp cận khác nhau: duyệt tổ hợp theo chiều rộng (Apriori) và xây dựng cấu trúc cây tần suất (FP-Growth).
* **Quy trình thực hiện:** tuân thủ mô hình CRISP-DM với đầy đủ 6 bước (hiểu biết nghiệp vụ, hiểu biết dữ liệu, chuẩn bị dữ liệu, xây dựng mô hình, đánh giá, triển khai), trong đó nhấn mạnh các giai đoạn tiền xử lý dữ liệu, cài đặt mô hình và đánh giá kết quả.
* **Giới hạn nghiên cứu:**
* Không xem xét các thuật toán khai phá luật kết hợp nâng cao như ECLAT, H-Mine hoặc các biến thể kết hợp.
* Không xử lý dữ liệu thời gian thực hoặc dữ liệu phi cấu trúc.
* Các tiêu chí đánh giá tập trung vào hiệu năng thuật toán và chất lượng luật, chưa đi sâu vào các yếu tố tâm lý tiêu dùng hay hành vi người mua.
* Về mặt công nghệ, đề tài sử dụng ngôn ngữ Python 3.x, với các thư viện hỗ trợ như Pandas, NumPy (xử lý dữ liệu), mlxtend.frequent\_patterns cho Apriori, fp-growth hoặc pyfpgrowth cho FP-Growth, Matplotlib, Seaborn để trực quan hóa kết quả và Streamlit để xây dựng ứng dụng giao diện thân thiện với người dùng. Ngoài ra, Jupyter Notebook hoặc Google Colab được sử dụng trong giai đoạn thử nghiệm và kiểm thử mô hình.
* Với phạm vi như trên, đề tài đảm bảo tính tập trung, đồng thời cung cấp nền tảng lý luận và thực tiễn đầy đủ để phân tích, triển khai và so sánh hiệu quả giữa hai thuật toán được chọn.

## ****Đối tượng nghiên cứu****

Báo cáo này tập trung nghiên cứu các thành phần sau:

* **Tập dữ liệu giao dịch (Market Basket Data):** Gồm các bản ghi danh sách mặt hàng mua cùng nhau trong một giao dịch. Dữ liệu có thể được cung cấp dưới hai hình thức:
  + **Nhập thủ công:** Người dùng trực tiếp nhập từng giao dịch theo định dạng Tx: [item₁, item₂, …].
  + **Tải file CSV:** Người dùng tải lên file dữ liệu thực tế (ví dụ: groceries.csv), trong đó mỗi dòng tương ứng một giao dịch với danh sách sản phẩm.
* **Luật kết hợp (Association Rules):** Các quy luật thể hiện mối quan hệ giữa các mặt hàng, được đánh giá qua các chỉ số support, confidence và lift.
* **Thuật toán Apriori và FP-Growth:** Hai phương pháp khai phá luật kết hợp tiêu biểu, sử dụng các hàm apriori(), fp\_growth() và association\_rules() của thư viện mlxtend.frequent\_patterns.

## ****Phương pháp nghiên cứu****

Phương pháp nghiên cứu đóng vai trò then chốt trong việc định hướng cách tiếp cận bài toán, đảm bảo tính logic, khoa học và khách quan trong toàn bộ quá trình khai phá dữ liệu. Để thực hiện mục tiêu đề ra, đề tài lựa chọn mô hình CRISP-DM (Cross Industry Standard Process for Data Mining) làm khung phương pháp luận chính, kết hợp với các kỹ thuật thực nghiệm trên dữ liệu thực tế và công cụ lập trình nhằm đảm bảo tính khả thi và ứng dụng cao.

### Áp dụng mô hình CRISP-DM

CRISP-DM là mô hình tiêu chuẩn gồm 6 bước được sử dụng phổ biến trong các dự án khai phá dữ liệu. Trong khuôn khổ đề tài, mô hình này được vận dụng như sau:

**Bước 1: Hiểu biết nghiệp vụ (Business Understanding)**: Phân tích bối cảnh ứng dụng thực tế, xác định nhu cầu phát hiện các mối quan hệ đồng xuất hiện giữa sản phẩm, từ đó hình thành mục tiêu khai phá luật kết hợp phục vụ hoạt động ra quyết định.

**Bước 2: Hiểu biết dữ liệu (Data Understanding)**: Thu thập dữ liệu giao dịch dạng giỏ hàng từ nguồn đã chuẩn bị; tiến hành khảo sát cấu trúc dữ liệu, đánh giá chất lượng và xác định các yếu tố ảnh hưởng đến kết quả khai phá như phân bố sản phẩm, độ hiếm gặp và số lượng giao dịch.

**Bước 3: Chuẩn bị dữ liệu (Data Preparation)**: Tiến hành làm sạch dữ liệu, chuẩn hóa định dạng, mã hóa dữ liệu về dạng nhị phân hoặc danh sách sản phẩm trên mỗi giao dịch. Đây là giai đoạn then chốt ảnh hưởng trực tiếp đến hiệu quả của các thuật toán khai phá.

**Bước 4: Xây dựng mô hình (Modeling)**: Thực hiện khai phá luật kết hợp trên tập dữ liệu đã xử lý bằng hai thuật toán: Apriori và FP-Growth, với cùng một bộ ngưỡng tham số (minimum support và confidence) để đảm bảo tính khách quan trong so sánh.

**Bước 5: Đánh giá mô hình (Evaluation)**: So sánh kết quả của hai thuật toán theo các tiêu chí định lượng và định tính như: số lượng luật khai thác được, thời gian xử lý, mức sử dụng tài nguyên hệ thống và khả năng mở rộng với dữ liệu lớn hơn.

**Bước 6: Triển khai mô hình (Deployment)**: Phân tích các luật kết hợp có ý nghĩa và đề xuất các ứng dụng thực tiễn như hệ thống gợi ý sản phẩm, hỗ trợ chiến lược marketing theo nhóm sản phẩm, hay đề xuất cải tiến bố trí cửa hàng.

### Phương pháp thu thập và xử lý dữ liệu

Dữ liệu được sử dụng trong đề tài là bộ dữ liệu Online Retail Dataset do UCI Machine Learning Repository cung cấp. Đây là tập dữ liệu thực tế của một công ty bán lẻ trực tuyến có trụ sở tại Vương quốc Anh, ghi lại thông tin giao dịch từ tháng 12/2010 đến tháng 12/2011. Tập dữ liệu chứa khoảng 541.909 dòng với các thuộc tính chính như:

* InvoiceNo: Số hóa đơn giao dịch
* StockCode: Mã sản phẩm
* Description: Tên sản phẩm
* Quantity: Số lượng sản phẩm được mua hoặc trả lại
* InvoiceDate: Ngày giao dịch
* UnitPrice: Giá của mỗi đơn vị sản phẩm
* CustomerID: Mã định danh khách hàng
* Country: Quốc gia của khách hàng

Bộ dữ liệu được tải dưới định dạng CSV (.csv) và có dung lượng hơn 24MB. Đây là tập dữ liệu phổ biến trong các nghiên cứu về khai phá luật kết hợp, phân tích hành vi tiêu dùng và phân tích giỏ hàng (Market Basket Analysis).

Ngoài ra ứng dụng được thiết kế để hỗ trợ nhiều phương thức nhập dữ liệu khác nhau, tạo điều kiện thuận lợi cho người dùng ở các lĩnh vực và mục đích phân tích khác nhau. Ngoài tập dữ liệu mặc định Online Retail từ UCI Machine Learning Repository (định dạng .csv), người dùng còn có thể:

* Tải lên tệp Excel (.xlsx) hoặc CSV (.csv) từ máy cá nhân
* Nhập tay các giao dịch dưới dạng danh sách các sản phẩm trong từng hóa đơn
* Sử dụng định dạng dữ liệu chuẩn trong khai phá luật như: Danh sách dạng Tx (List[List[str]]) cho FP-Growth, Dạng Groceries List.

Ứng dụng không giới hạn người dùng chỉ ở một tập dữ liệu cố định, mà cho phép thử nghiệm nhiều bộ dữ liệu khác nhau như:

* Dữ liệu mua sắm tạp hóa (Groceries Dataset)
* Dữ liệu đơn hàng trong thương mại điện tử nội địa
* Dữ liệu giao dịch trong siêu thị, nhà thuốc, hoặc bán lẻ sách
* Dữ liệu người dùng truy cập/mua nội dung số (digital content)

## Ý nghĩa

### Ý nghĩa khoa học

Nghiên cứu khai phá luật kết hợp bằng thuật toán Apriori và FP-Growth không chỉ củng cố nền tảng lý thuyết về tập mục thường xuyên và luật kết hợp, mà còn làm rõ cách thức hoạt động và ảnh hưởng của cấu trúc dữ liệu đến hiệu năng tính toán. Việc đo lường và so sánh các chỉ số support, confidence và lift trong từng kịch bản dữ liệu cụ thể sẽ cung cấp bằng chứng thực nghiệm về độ ổn định và tính tin cậy của hai phương pháp. Kết quả thu được từ báo cáo này hứa hẹn đóng góp tài liệu tham khảo quan trọng cho các nghiên cứu tiếp theo về tối ưu thuật toán, cũng như tạo cơ sở cho việc phát triển các giải pháp kết hợp nhiều thuật toán khai phá nâng cao.

### Ý nghĩa thực tiễn

* Trong bối cảnh thương mại điện tử và bán lẻ hiện đại, khả năng khai thác mối quan hệ giữa các sản phẩm trong giỏ hàng khách hàng mang lại giá trị thiết thực:
* Gợi ý sản phẩm thông minh: Dựa trên các luật kết hợp thu được, hệ thống có thể đề xuất mặt hàng phù hợp, tăng mức độ hài lòng và doanh số bán hàng.
* Tối ưu bố trí hàng hóa: Những insight về mặt hàng thường mua cùng nhau giúp nhà quản lý sắp xếp sản phẩm hợp lý, khuyến khích mua chéo và rút ngắn hành trình mua sắm.
* Công cụ hỗ trợ ra quyết định: Ứng dụng Streamlit cung cấp giao diện trực quan, cho phép điều chỉnh ngưỡng khai phá và quan sát nhanh kết quả, từ đó hỗ trợ nhà quản lý đưa ra quyết định kịp thời và chính xác.
* Tiềm năng mở rộng: Kết quả nghiên cứu là bước đệm để doanh nghiệp áp dụng phân tích Big Data, cá nhân hóa trải nghiệm khách hàng và triển khai các mô hình AI phức hợp trong tương lai.
* Với cách tiếp cận khoa học kết hợp ứng dụng thực tiễn, báo cáo không chỉ làm rõ nguyên lý thuật toán mà còn mang đến giá trị trực tiếp cho hoạt động kinh doanh và phát triển sản phẩm.

# : CƠ SỞ LÍ THUYẾT

## Tìm hiểu về ngôn ngữ lập trình

### Python là gì?

Python là một ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng, được phát triển bởi Guido van Rossum và phát hành lần đầu tiên vào năm 1991. Với cú pháp đơn giản, dễ đọc, và khả năng mở rộng linh hoạt, Python nhanh chóng trở thành một trong những ngôn ngữ lập trình phổ biến và được ưa chuộng nhất trong nhiều lĩnh vực như khoa học dữ liệu, trí tuệ nhân tạo, phát triển web, tự động hóa và giáo dục.

  
Hình 2.1 Hình ảnh liên quan đến ngôn ngữ python

Tính linh hoạt này, cùng với sự thân thiện với người mới bắt đầu, đã khiến nó trở thành một trong những ngôn ngữ lập trình được sử dụng nhiều nhất hiện nay. Một cuộc khảo sát được thực hiện bởi công ty phân tích ngành RedMonk cho thấy rằng đây là ngôn ngữ lập trình phổ biến thứ hai đối với các [nhà phát triển](https://glints.com/vn/blog/developer-la-gi/) vào năm [2021](https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/).

### Đặc trưng

Một trong những đặc điểm nổi bật của Python là khả năng hỗ trợ nhiều mô hình lập trình khác nhau như: lập trình thủ tục, lập trình hướng đối tượng, và lập trình hàm. Python có một kho thư viện phong phú và cộng đồng phát triển rộng lớn, điều này giúp rút ngắn thời gian phát triển ứng dụng, đặc biệt trong các lĩnh vực phân tích dữ liệu và khai phá dữ liệu – nơi các thư viện như pandas, numpy, matplotlib, scikit-learn, mlxtend, và streamlit được sử dụng phổ biến.

Trong phạm vi đề tài này, Python được lựa chọn làm ngôn ngữ lập trình chính để triển khai các thuật toán khai phá luật kết hợp (Apriori và FP-Growth) nhờ vào các ưu điểm sau:

* Cú pháp ngắn gọn, dễ hiểu, phù hợp cho việc phát triển nhanh các nguyên mẫu.
* Hệ sinh thái thư viện mạnh mẽ, hỗ trợ đầy đủ cho tiền xử lý dữ liệu, khai phá mẫu, trực quan hóa và xây dựng giao diện đơn giản.
* Tính phổ biến và khả năng tích hợp cao, dễ dàng chia sẻ, triển khai và bảo trì hệ thống trong các môi trường học thuật và công nghiệp.

## Visual code và công cụ hỗ trợ

### Giới thiệu về Visual Studio Code

Visual Studio Code (viết tắt: VS Code) là một trình soạn thảo mã nguồn đa nền tảng (source-code editor) miễn phí, được phát triển và duy trì bởi Microsoft. VS Code được thiết kế tối ưu cho việc phát triển phần mềm, lập trình ứng dụng và hỗ trợ hầu hết các ngôn ngữ lập trình phổ biến như Python, JavaScript, C++, Java, v.v.



Hình 2.2 Hình ảnh về VS Code

VS Code không chỉ là một trình soạn thảo văn bản thuần túy mà còn là một môi trường phát triển nhẹ, tiện lợi và mở rộng linh hoạt thông qua hệ thống extension mạnh mẽ. Một số tính năng nổi bật của VS Code bao gồm:

* Tô sáng cú pháp (syntax highlighting) và gợi ý mã thông minh (IntelliSense) giúp tăng tốc lập trình.
* Hỗ trợ debug trực tiếp trong trình soạn thảo, bao gồm đặt breakpoint, xem biến và theo dõi call stack.
* Tích hợp Git/GitHub để quản lý phiên bản, commit, push/pull ngay trong giao diện.
* Kho tiện ích mở rộng (extension marketplace) phong phú, cho phép cài đặt thêm hỗ trợ cho Python, Jupyter Notebook, Streamlit, Pandas, và nhiều công nghệ khác.
* Tích hợp Terminal giúp lập trình viên có thể chạy lệnh trực tiếp mà không cần chuyển qua cửa sổ khác.

Trong khuôn khổ đề tài, VS Code được sử dụng làm môi trường phát triển chính, hỗ trợ lập trình bằng ngôn ngữ Python, xử lý dữ liệu với Pandas, trực quan hóa bằng Matplotlib/Seaborn, và đặc biệt là tích hợp ứng dụng với Streamlit. Với sự hỗ trợ mạnh mẽ từ các extension như Python, Jupyter, Streamlit và Pylance, VS Code giúp đảm bảo quy trình phát triển, thử nghiệm và triển khai ứng dụng diễn ra hiệu quả và thuận tiện.



Hình 2.3 Hình ảnh minh họa thêm về VS Code

### Thư viện Pandas

Pandas là một thư viện mã nguồn mở được xây dựng dựa trên NumPy, sử dụng thao tác và phân tích dữ liệu, được thiết kế để cho phép bạn làm việc với dữ liệu được gắn nhãn hoặc quan hệ theo cách trực quan hơn.



Hình 2.4 Hình ảnh về thư viện Pandas

Pandas có ba cấu trúc dữ liệu và nó được xây dựng dựa trên thư viện Numpy vậy nên chúng hoạt động rất nhanh và hiệu quả: Series, DataFrame, Panel. Trong đó Panel là mảng 3 chiều. Panel thì không được sử dụng rãi như như Series hay DataFrame và nó cũng không dễ hiển thị hay trừu tượng hoá như màn một chiều và hai chiều, nên dưới đây mình chỉ giới thiệu Series và DataFrame

Lợi ích của Pandas:

* Có thể xử lý tập dữ liệu khác nhau về định dạng: chuỗi thời gian, bảng không đồng nhất, ma trận dữ liệu
* Khả năng import dữ liệu từ nhiều nguồn khác nhau như CSV, DB/SQL
* Có thể xử lý vô số phép toán cho tập dữ liệu: subsetting, slicing, filtering, merging, groupBy, re-ordering, and re-shaping,..
* Xử lý dữ liệu mất mát theo ý người dùng mong muốn: bỏ qua dữ liệu đó hoặc chuyển sang 0.
* Xử lý, phân tích dữ liệu tốt như mô hình hoá và thống kê
* Tích hợp tốt với các thư viện khác của python
* Cung cấp hiệu suất tốt

Trong đề tài, Pandas đóng vai trò cốt lõi trong quá trình xử lý và chuẩn hóa bộ dữ liệu "Online Retail", bao gồm: loại bỏ dữ liệu nhiễu, chuẩn hóa tên sản phẩm, và chuyển đổi định dạng phù hợp để khai phá luật kết hợp bằng các thuật toán Apriori và FP-Growth.

### Streamlit

Streamlit là một framework mã nguồn mở và miễn phí để nhanh chóng xây dựng và chia sẻ các ứng dụng web khoa học dữ liệu và học máy trên nền tảng web. Đây là một thư viện dựa trên Python được thiết kế dành riêng cho các kỹ sư học máy. Các nhà khoa học dữ liệu hoặc kỹ sư máy học không phải là nhà phát triển web và họ không quan tâm đến việc dành hàng tuần để học cách sử dụng các framework này để xây dựng ứng dụng web. Thay vào đó, họ muốn một công cụ dễ học và dễ sử dụng hơn, miễn là nó có thể hiển thị dữ liệu và thu thập các tham số cần thiết cho mô hình hóa. Streamlit cho phép bạn tạo một ứng dụng có giao diện đẹp mắt chỉ với một vài dòng mã.

Với cú pháp đơn giản và khả năng tích hợp mạnh mẽ với các thư viện Python như Pandas, Matplotlib và Scikit-learn, Streamlit giúp tạo ra các giao diện người dùng trực quan mà không cần đến HTML, CSS hay JavaScript. Trong phạm vi đề tài, Streamlit được sử dụng để xây dựng giao diện ứng dụng, giúp người dùng có thể tùy chỉnh ngưỡng tham số (support, confidence), thực thi thuật toán, và xem kết quả khai phá trực quan dưới dạng bảng hoặc biểu đồ.



Hình 2.5 Hình ảnh về Streamlit

### Grapviz

Graphviz (viết tắt của Graph Visualization Software) là một bộ công cụ mã nguồn mở dùng để biểu diễn và trực quan hóa cấu trúc đồ thị (graph structures). Được phát triển bởi AT&T Research Lab, Graphviz cho phép người dùng mô tả đồ thị thông qua một ngôn ngữ văn bản đơn giản gọi là DOT language, sau đó tự động sắp xếp và xuất ra sơ đồ hình học tương ứng.

**Chức năng chính**

Graphviz hỗ trợ tạo sơ đồ cho cả đồ thị có hướng (Directed Graph) và đồ thị vô hướng (Undirected Graph). Nó cung cấp các công cụ để:

* Tự động sắp xếp các đỉnh và cạnh sao cho sơ đồ dễ đọc và đẹp mắt.
* Tùy biến hình dạng, màu sắc, nhãn (label) của các đỉnh (nodes) và cạnh (edges).
* Xuất sơ đồ sang nhiều định dạng khác nhau như: PNG, PDF, SVG, EPS, DOT, v.v.
* Hỗ trợ tích hợp vào các ngôn ngữ lập trình như Python, Java, C++, JavaScript, phục vụ cho các ứng dụng cần trực quan hóa dữ liệu.

**Cấu trúc và cú pháp DOT:**

Đồ thị trong Graphviz được mô tả thông qua các đoạn mã DOT, có cấu trúc như sau:

* digraph để khai báo đồ thị có hướng.
* graph để khai báo đồ thị vô hướng.
* Mỗi đỉnh và cạnh được định nghĩa bằng cú pháp

Ngoài ra, người dùng có thể khai báo thêm thuộc tính cho đỉnh và cạnh

**Ứng dụng:** Graphviz được sử dụng rộng rãi trong nhiều lĩnh vực:

* Khoa học máy tính: biểu diễn cấu trúc dữ liệu, cây quyết định, đồ thị trạng thái, dependency graph trong lập trình.
* Trí tuệ nhân tạo: trực quan hóa thuật toán tìm kiếm, cây suy diễn.
* Quản lý dự án: vẽ sơ đồ luồng công việc, sơ đồ tiến trình.
* Khoa học dữ liệu: trực quan hóa mạng lưới xã hội, biểu đồ phân cụm

## Các thuật toán khai phá luật kết hợp

### Thuật toán Apriori

Trong lĩnh vực khai phá luật kết hợp (association rule mining), Apriori là một trong những thuật toán cổ điển và nền tảng nhất, được đề xuất bởi R. Agrawal và R. Srikant vào năm 1994. Apriori được thiết kế nhằm trích xuất các tập mục thường xuyên (frequent itemsets) từ cơ sở dữ liệu giao dịch, qua đó sinh ra các luật kết hợp có giá trị cho việc phân tích hành vi người tiêu dùng, đề xuất sản phẩm hoặc hỗ trợ chiến lược tiếp thị.

Thuật toán này đóng vai trò quan trọng trong sự phát triển của khai phá dữ liệu, đồng thời là cơ sở lý thuyết để phát triển nhiều thuật toán tối ưu sau này như ECLAT, FP-Growth, H-Mine…

Apriori dựa trên một thuộc tính quan trọng gọi là Nguyên lý Apriori:

Nguyên lý Apriori chính là một dạng đặc biệt của tính đơn điệu nghịch (anti-monotonicity): “Nếu một tập mục không phải là frequent, thì mọi siêu tập của nó cũng không thể là frequent.” Hay ngược lại: "Mọi tập hợp con của một tập phổ biến đều phải là phổ biến."

Điều này cho phép cắt giảm đáng kể không gian tìm kiếm: nếu một tập ứng viên chứa một tập con không phổ biến, thì bản thân nó cũng không cần được xét đến.

Ví dụ:

Nếu tập {Trứng, Bơ} không phổ biến, thì {Trứng, Bơ, Sữa} chắc chắn không cần kiểm tra.

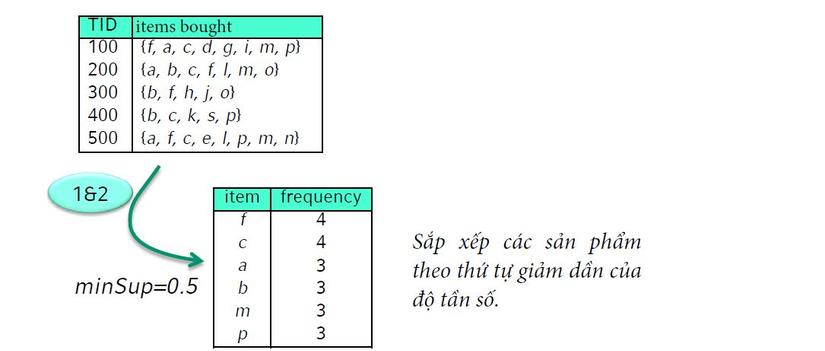
Với nguyên lý này, Apriori sử dụng kỹ thuật bottom-up để mở rộng tập mục theo chiều tăng dần kích thước, từ tập 1 phần tử đến khi không còn ứng viên nào thỏa mãn.

**Các bước hoạt động:**

Thuật toán Apriori thực hiện qua nhiều vòng lặp (iterations), mỗi vòng tương ứng với một bậc của tập mục (1-itemset, 2-itemset,…). Các bước cơ bản bao gồm:

Bước 1: Khởi tạo – Tính L1

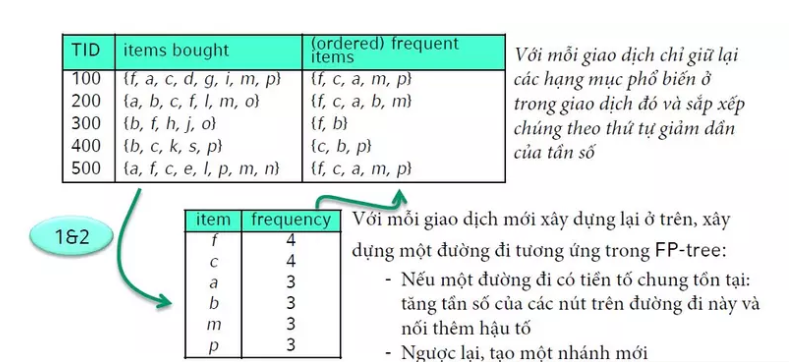
* Duyệt qua toàn bộ tập giao dịch (transaction database).
* Đếm số lần xuất hiện (support count) của từng mặt hàng đơn lẻ.
* Các item có tần suất ≥ min\_support tạo thành tập phổ biến cấp 1 (L1).



Hình 2.6 Hình ảnh minh họa bước 1 của Apriori

Bước 2: Sinh tập ứng viên Ck (Join Step)

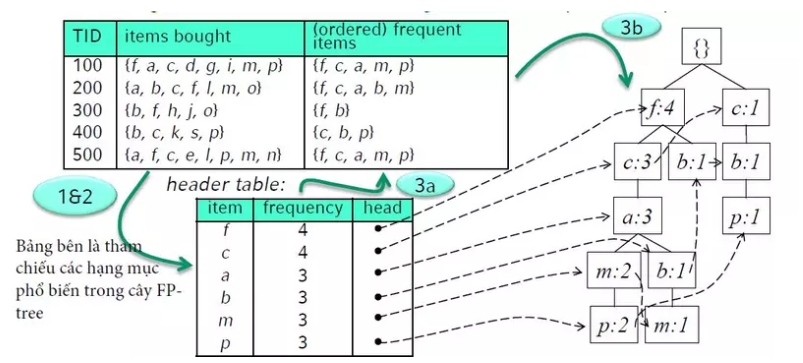
* Từ tập phổ biến bậc (k−1), thực hiện phép kết hợp có điều kiện (self-join) để tạo tập ứng viên kích thước k (Ck).
* Chỉ kết hợp những tập có k−1 phần tử giống nhau đầu tiên để tránh trùng lặp.



Hình 2.7 Hình ảnh minh họa bước 2 của Apriori

Bước 3: Cắt tỉa (Prune Step)

* Với mỗi tập trong Ck, kiểm tra tất cả các tập con bậc k−1 của nó.
* Nếu có ít nhất một tập con không thuộc Lk−1, loại bỏ ứng viên đó khỏi Ck.



Hình 2.8 Hình ảnh minh họa bước 3 của Apriori

Bước 4: Đếm support và tạo Lk

* Duyệt lại toàn bộ cơ sở dữ liệu để đếm số lần xuất hiện của các tập ứng viên trong Ck.
* Các tập có tần suất ≥ min\_support sẽ tạo thành tập phổ biến Lk.

Bước 5: Lặp lại

* Quay lại bước 2 với giá trị k+1, lặp lại quá trình cho đến khi không còn tập phổ biến nào được sinh ra (Lk = ∅).

Bước 6: Sinh luật kết hợp (Association Rule Generation)

* Với mỗi tập mục phổ biến F, sinh ra các luật có dạng A → B (với A ∪ B = F, A ∩ B = ∅).
* Chỉ giữ lại các luật có độ tin cậy (confidence) ≥ min\_confidence, tính theo công thức:

Confidence (A → B) =

* Có thể tính thêm chỉ số lift, conviction để đánh giá chất lượng luật.

**Ưu điểm:**

* Dễ hiểu, dễ cài đặt: phù hợp cho người mới học khai phá dữ liệu và dùng làm bước khởi đầu nghiên cứu.
* Nguyên lý rõ ràng, minh bạch: toàn bộ quá trình khai phá dựa trên nguyên lý toán học chắc chắn và có khả năng giải thích cao.
* Tính khả kiểm cao (explainability): mỗi tập mục và luật đều có thể truy ngược nguyên nhân hình thành.

**Nhược điểm:**

* Số lượng tập ứng viên rất lớn: Khi số lượng mục (item) nhiều hoặc min\_support thấp, số lượng tập ứng viên tăng theo cấp số mũ.
* Quét cơ sở dữ liệu nhiều lần: mỗi vòng lặp đều cần quét toàn bộ dữ liệu để đếm support, gây tốn thời gian và tài nguyên.
* Không tối ưu cho dữ liệu lớn hoặc phân tán: khó mở rộng cho các hệ thống big data hoặc môi trường phân tán (distributed systems).
* Yêu cầu tập dữ liệu phù hợp: với dữ liệu nhiều nhiễu, hoặc phân phối tần suất không đều, Apriori dễ bị giảm hiệu quả.

### Thuật toán FP-Growth (Frequent Pattern Growth)

FP-Growth là một thuật toán khai phá tập mục thường xuyên (frequent itemsets) được đề xuất bởi Han, Pei và Yin năm 2000. Khác với Apriori – vốn dựa vào quá trình sinh ứng viên và kiểm tra tuần tự – FP-Growth sử dụng một cách tiếp cận hoàn toàn mới: nén dữ liệu giao dịch vào một cấu trúc cây FP-Tree (Frequent Pattern Tree), từ đó khai phá các mẫu thường xuyên mà không cần sinh ứng viên rõ ràng.

Thuật toán FP-Growth được coi là một bước tiến quan trọng trong lĩnh vực khai phá luật kết hợp vì giảm mạnh chi phí tính toán, đặc biệt khi làm việc với tập dữ liệu lớn.

**Nguyên lý hoạt động của FP-Growth:**

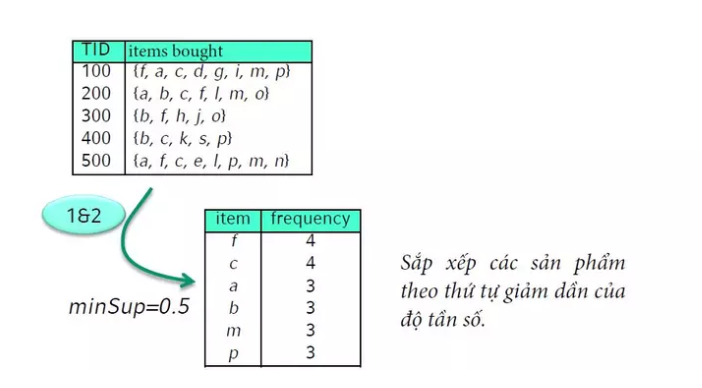
Thay vì duyệt tuần tự như Apriori, FP-Growth dựa trên 2 ý tưởng chính:

* Nén dữ liệu giao dịch vào một cấu trúc cây (FP-Tree): Các giao dịch giống nhau hoặc chia sẻ prefix (tiền tố) sẽ được gộp vào một nhánh cây, từ đó giảm thiểu độ trùng lặp dữ liệu.
* Khai phá theo hướng đệ quy (pattern-growth): Bắt đầu từ các item ít phổ biến, thuật toán xây dựng các cây con điều kiện để khai phá tập mục mở rộng theo chiều sâu, không cần sinh tập ứng viên.

Thuật toán FP Growth sẽ trải qua các bước:

Bước 0: Quét lần thứ nhất – Đếm support: Duyệt toàn bộ tập giao dịch để:

* Đếm số lần xuất hiện của từng item.
* Lọc bỏ các item không đạt ngưỡng minimum support.
* Sắp xếp danh sách item theo thứ tự giảm dần tần suất (support) → giúp tăng khả năng chia sẻ nhánh trong FP-Tree.

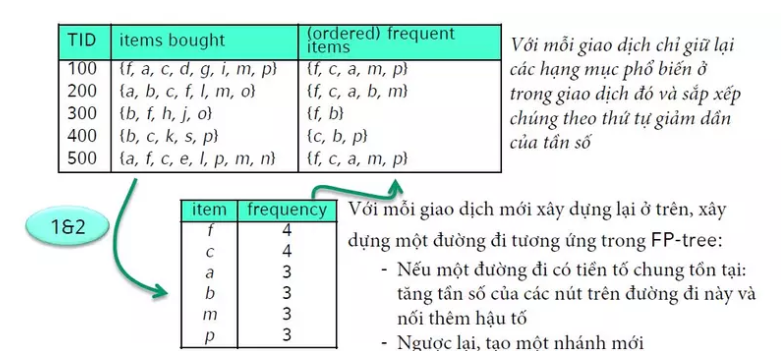


Hình 2.9 Hình ảnh minh họa bước 0 của FP\_Growth

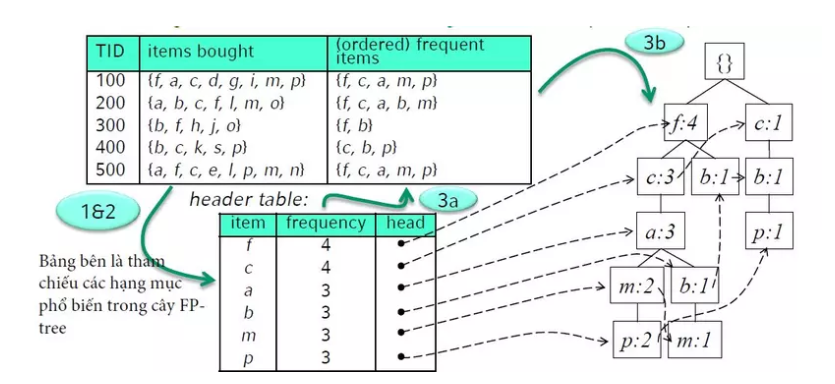
Bước 1: Xây dựng FP-Tree

* Quét cơ sở dữ liệu lần thứ hai.
* Với mỗi giao dịch:
* Bỏ qua item không phổ biến.
* Sắp xếp lại item theo thứ tự giảm dần support đã xác định.
* Chèn lần lượt các item vào FP-Tree theo thứ tự đó:
* Nếu nhánh đã tồn tại → tăng count của nút tương ứng.
* Nếu không → tạo nhánh mới.
* Cập nhật Header Table, liên kết các nút có cùng item bằng con trỏ nối.

Dữ liệu được nén lại trong một cây duy nhất, nơi các đường đi đại diện cho các mẫu lặp của giao dịch.



Hình 2.10 Hình ảnh minh họa bước 1 của FP\_Growth



Hình 2.11 Hình ảnh minh họa thêm bước 1 của FP\_Growt

Bước 2: Khai phá FP-Tree – Pattern Growth

* Với mỗi item trong Header Table (duyệt từ item ít phổ biến đến phổ biến):
* Trích xuất Conditional Pattern Base: tập các đường đi từ gốc đến item đó.
* Tạo Conditional FP-Tree từ các pattern base.
* Đệ quy: tiếp tục áp dụng FP-Growth trên cây điều kiện.
* Gộp kết quả để sinh ra các tập mục thường xuyên.
* Thuật toán dừng khi:
* Conditional FP-Tree rỗng.
* Cây chỉ còn một nhánh → có thể sinh trực tiếp tất cả các kết hợp tổ hợp của nhánh.

**Ưu điểm của FP-Growth:**

* Chỉ cần quét cơ sở dữ liệu đúng 2 lần, không lặp nhiều như Apriori.
* Không cần sinh tập ứng viên → tránh bùng nổ tổ hợp và tiết kiệm bộ nhớ.
* Hiệu năng rất cao, đặc biệt khi tập dữ liệu có nhiều giao dịch trùng lặp hoặc chia sẻ các chuỗi mục giống nhau.
* Phù hợp với các ứng dụng yêu cầu khai phá nhanh, dữ liệu lớn.

**Nhược điểm của FP-Growth:**

* Cấu trúc FP-Tree phức tạp, cần quản lý Header Table và các con trỏ liên kết giữa các nút – khó hiện thực hơn Apriori.
* Tốn bộ nhớ nếu dữ liệu phân tán cao: nếu các giao dịch khác biệt nhiều, cây FP-Tree có thể phình to gần bằng dữ liệu gốc.
* Khó xử lý trong môi trường phân tán hoặc song song hóa: do khai phá đệ quy sâu.

### So sánh lý thuyết giữa Apriori và FP-Growth

Thuật toán Apriori và FP-Growth là hai phương pháp phổ biến nhất trong lĩnh vực khai phá luật kết hợp. Cả hai đều nhằm mục đích phát hiện ra các tập mục thường xuyên và từ đó sinh ra các luật kết hợp có độ tin cậy cao. Tuy nhiên, hai thuật toán này khác nhau về mặt nguyên lý, chiến lược tính toán, cấu trúc dữ liệu và hiệu năng, phản ánh hai cách tiếp cận đối lập trong cùng một bài toán.

Về mặt nguyên lý, Apriori dựa trên phương pháp sinh và kiểm tra (generate-and-test). Thuật toán này bắt đầu từ các item đơn lẻ (1-itemsets), sau đó liên tục sinh ra các tập ứng viên bậc cao hơn bằng phép kết hợp các tập phổ biến trước đó, đồng thời loại bỏ các ứng viên không đạt ngưỡng hỗ trợ tối thiểu. Mỗi bước đều yêu cầu quét toàn bộ cơ sở dữ liệu để tính toán support. Trái lại, FP-Growth loại bỏ hoàn toàn bước sinh ứng viên. Thay vào đó, nó nén toàn bộ dữ liệu giao dịch vào một cấu trúc cây gọi là FP-Tree và sau đó khai phá đệ quy theo hướng mở rộng mẫu (pattern growth). Điều này cho phép FP-Growth đạt hiệu suất cao hơn rất nhiều trong các tập dữ liệu có số lượng giao dịch lớn và lặp lại nhiều.

Xét về chiến lược khai phá, Apriori sử dụng tiếp cận theo chiều rộng (Breadth-First Search), duyệt lần lượt từng bậc tổ hợp từ nhỏ đến lớn. Mỗi vòng lặp chỉ tìm ra tập phổ biến với một độ dài cụ thể. Ngược lại, FP-Growth áp dụng tư duy duyệt theo chiều sâu (Depth-First Search), khởi đầu từ các item ít phổ biến rồi mở rộng đệ quy thông qua các cây con điều kiện, giúp khai phá nhanh hơn và hiệu quả hơn trong nhiều trường hợp.

Về mặt cấu trúc dữ liệu, Apriori xử lý dữ liệu nguyên gốc ở dạng bảng giao dịch, không yêu cầu thêm cấu trúc phụ trợ. Trong khi đó, FP-Growth yêu cầu xây dựng FP-Tree cùng với bảng chỉ mục (Header Table) để lưu các nút đồng cấp có cùng item, từ đó thực hiện khai thác đệ quy. Việc xây dựng và duy trì FP-Tree đòi hỏi xử lý con trỏ, liên kết động và tối ưu hóa bộ nhớ – điều khiến FP-Growth có cấu trúc phức tạp hơn đáng kể nhưng lại tiết kiệm tài nguyên khi dữ liệu có độ trùng lặp cao.

Về mặt hiệu năng, Apriori thường chậm hơn rõ rệt do phải thực hiện nhiều lần quét cơ sở dữ liệu – mỗi vòng lặp sinh tập phổ biến Lₖ lại yêu cầu một lượt quét đầy đủ. Ngoài ra, khi ngưỡng support thấp hoặc số lượng item lớn, Apriori sẽ sinh ra một số lượng rất lớn tập ứng viên, gây ra hiện tượng "bùng nổ tổ hợp". Trong khi đó, FP-Growth chỉ cần quét dữ liệu hai lần: một lần để đếm support, và một lần để xây dựng FP-Tree. Nhờ đó, thời gian thực thi của FP-Growth trong các hệ thống dữ liệu lớn thường vượt trội hơn so với Apriori.

Tuy nhiên, xét về mức độ phức tạp trong triển khai, Apriori có ưu thế hơn nhờ thuật toán tuyến tính, rõ ràng và dễ lập trình. Điều này lý giải vì sao Apriori thường được sử dụng trong mục đích giảng dạy, minh họa thuật toán, hoặc các ứng dụng thử nghiệm với quy mô nhỏ. Ngược lại, FP-Growth đòi hỏi kỹ năng cao hơn trong xử lý dữ liệu và cấu trúc con trỏ, nhưng lại là lựa chọn hiệu quả cho các hệ thống thực tế như hệ thống gợi ý sản phẩm, phân tích hành vi người dùng, hoặc hỗ trợ ra quyết định trong thương mại điện tử.

Tóm lại, Apriori và FP-Growth đại diện cho hai chiến lược điển hình trong khai phá luật kết hợp: một bên là kiểm tra toàn bộ khả năng sinh ứng viên, một bên là khai phá theo chiều sâu sau khi nén dữ liệu. Tùy thuộc vào mục tiêu, quy mô và đặc điểm dữ liệu mà mỗi thuật toán sẽ phát huy ưu thế khác nhau. Trong thực tiễn hiện đại, FP-Growth thường được lựa chọn khi hiệu năng và khả năng mở rộng là yếu tố ưu tiên hàng đầu.

# PHÂN TÍCH VÀ THIẾT KẾ

## Phân tích bộ dữ liệu “One Retail”

### ****Mô tả các trường dữ liệu****

Bộ dữ liệu bao gồm 8 thuộc tính chính, cụ thể như sau:

* **InvoiceNo**: Mã hóa đơn, là định danh duy nhất cho mỗi giao dịch. Các giá trị bắt đầu bằng chữ cái 'C' biểu thị các hóa đơn bị hủy hoặc trả lại.
* **StockCode**: Mã định danh sản phẩm trong kho. Một số mã như 'POST', 'DOT', 'M' không đại diện cho sản phẩm thực mà là các phí vận chuyển, điều chỉnh hoặc ghi chú.
* **Description**: Mô tả sản phẩm. Đây là thông tin dạng chuỗi giúp người đọc hiểu được mặt hàng trong giao dịch.
* **Quantity**: Số lượng sản phẩm đã mua. Nếu giá trị âm, đó là hàng bị trả lại.
* **InvoiceDate**: Thời gian giao dịch (bao gồm cả ngày và giờ).
* **UnitPrice**: Giá đơn vị của sản phẩm (theo đơn vị bảng Anh - GBP).
* **CustomerID**: Mã khách hàng – định danh duy nhất mỗi người mua. Trường này giúp nhóm các giao dịch theo từng người cụ thể.
* **Country**: Quốc gia nơi khách hàng thực hiện giao dịch. Đa phần khách hàng đến từ Vương quốc Anh.

### ****Tiền xử lý dữ liệu (Data Preprocessing)****

Tiền xử lý là bước quan trọng nhằm đảm bảo dữ liệu đầu vào sạch, chính xác và có cấu trúc phù hợp cho quá trình khai phá.

**Làm sạch dữ liệu:**

* **Dữ liệu thiếu**: Có **34,915 dòng thiếu giá trị** CustomerID và **299 dòng thiếu** Description. Các bản ghi này đã bị loại bỏ vì không đủ thông tin để xác định người mua hoặc sản phẩm.
* **Loại bỏ giao dịch trả hàng**: Các dòng có Quantity < 0 đại diện cho các sản phẩm bị trả lại, không phản ánh hành vi mua hàng thực sự, nên cũng được loại bỏ.
* **Xử lý mã sản phẩm không hợp lệ**: Các giá trị StockCode như 'POST', 'DOT', 'BANK CHARGES', 'ADJUST', 'M' không phản ánh sản phẩm cụ thể nên cũng bị loại bỏ.

Sau bước làm sạch, còn **63.079 dòng dữ liệu hợp lệ** (từ 100.000 dòng đầu).

**Chuyển đổi dữ liệu về định dạng giao dịch:**

Dữ liệu đã được nhóm theo InvoiceNo và CustomerID để tạo thành **các giao dịch duy nhất**, trong đó mỗi giao dịch là **danh sách các sản phẩm (theo** Description**)** mà khách hàng đã mua.

* Tổng cộng thu được **3.168 giao dịch** sau tiền xử lý.
* Mỗi dòng trong bảng kết quả là 1 giao dịch: gồm InvoiceNo, CustomerID, và danh sách TransactionItems.

## Áp dụng thuật toán Apriori

### Tổng quan về cách triển khai

Thuật toán Apriori được triển khai trên bộ dữ liệu "Online Retail" đã tiền xử lý với mục tiêu trích xuất các tập mục thường xuyên (frequent itemsets) và từ đó khai phá các luật kết hợp (association rules) có ý nghĩa trong hành vi mua sắm của khách hàng. Phương pháp được hiện thực bằng Python sử dụng thư viện mlxtend, kết hợp với pandas và TransactionEncoder để chuẩn bị dữ liệu phù hợp cho mô hình.

### Cấu hình tham số và chuẩn hóa dữ liệu đầu vào

Đầu vào cho Apriori là một tập các giao dịch, mỗi giao dịch được biểu diễn dưới dạng danh sách các sản phẩm được mua cùng nhau trong một hóa đơn. Dữ liệu sau khi xử lý được chuyển về dạng ma trận nhị phân (one-hot encoding), trong đó mỗi dòng là một giao dịch, mỗi cột tương ứng với một sản phẩm, và giá trị là True/False tùy theo sản phẩm đó có xuất hiện trong giao dịch hay không.

Các tham số chính của thuật toán được lựa chọn như sau:

* Ngưỡng hỗ trợ tối thiểu (min\_support) được đặt là 0.01: tức là một tập mục chỉ được coi là "phổ biến" nếu nó xuất hiện trong ít nhất 1% tổng số giao dịch. Ngưỡng này được cân nhắc để loại bỏ nhiễu nhưng vẫn giữ lại các kết hợp phổ biến, tránh rơi vào bẫy khai phá những mẫu hiếm, không đại diện cho xu hướng tiêu dùng thực tế.
* Ngưỡng độ tin cậy (min\_confidence) là 0.4: đây là mức độ chắc chắn tối thiểu cần có để một luật được xem là đáng tin cậy. Giá trị này được chọn để giữ lại những luật có tính tiên đoán tương đối cao mà không quá khắt khe, nhằm đảm bảo đủ số lượng luật cho phân tích.

Đo lường lift cũng được sử dụng để đánh giá tính liên kết nội tại giữa phần vế trái (antecedents) và phần vế phải (consequents) của luật. Luật có lift > 1 cho thấy có mối liên hệ dương (positive correlation) giữa hai phần, còn lift < 1 cho thấy sự ràng buộc là ngẫu nhiên hoặc ngược chiều.

### Kết quả khai phá luật kết hợp

Sau quá trình xử lý, thuật toán đã khai phá được một tập hợp các luật kết hợp có tính ý nghĩa và thực tiễn. Một số ví dụ tiêu biểu:

Luật 1: {WHITE HANGING HEART T-LIGHT HOLDER} → {WHITE METAL LANTERN}

support = 0.0125, confidence = 0.46, lift = 3.2

→ Gợi ý rằng khách hàng mua đèn tim treo trắng có khả năng rất cao cũng sẽ mua đèn kim loại trắng, và mức liên hệ giữa chúng cao gấp 3.2 lần so với ngẫu nhiên.

Luật 2: {REGENCY CAKESTAND 3 TIER} → {ASSORTED COLOUR BIRD ORNAMENT}

support = 0.0111, confidence = 0.42, lift = 2.9

→ Các sản phẩm trang trí bàn tiệc thường đi kèm với đồ trang trí nhỏ như chim sứ.

Ngoài ra, có nhiều luật thể hiện mối liên kết giữa các sản phẩm theo bộ, theo phong cách thiết kế (màu sắc, vật liệu) hoặc mục đích sử dụng (tiệc, nhà bếp, lễ hội).

### Phân tích sâu các đặc điểm của luật khai phá

Tập mục phổ biến thường nhỏ (2-3 sản phẩm): Điều này phản ánh thực tế rằng đa phần khách hàng chỉ mua một vài sản phẩm có liên hệ gần về mục đích sử dụng trong cùng một giao dịch.

Sự xuất hiện của cụm sản phẩm theo chủ đề: Các nhóm sản phẩm như “WHITE...”, “SET OF...”, “LANTERN” thường đi kèm nhau, chứng tỏ rằng màu sắc và chủ đề sản phẩm có ảnh hưởng rõ rệt đến hành vi mua sắm.

Giá trị lift cao là chỉ báo mạnh cho cơ hội gợi ý sản phẩm: Những luật có lift > 2 thường đến từ các giao dịch có tính "bộ sưu tập", nơi người dùng mua theo sở thích cá nhân nhất quán. Đây là cơ sở tốt để thiết kế hệ thống gợi ý sản phẩm (recommender system).

Sự chênh lệch giữa confidence và lift giúp phân loại luật theo bản chất:

- confidence cao, lift thấp → luật phổ biến nhưng yếu về liên kết nội tại.

- lift cao, confidence vừa phải → luật có tiềm năng khai phá những kết hợp ít phổ biến nhưng gắn kết mạnh.

## Áp dụng thuật toán FP-Growth

### Tổng quan về cách triển khai

Thuật toán FP-Growth được triển khai trong ứng dụng dưới dạng một công cụ trực quan hóa trên nền tảng Streamlit, với giao diện thân thiện và khả năng tương tác cao. Tệp mã nguồn chính điều khiển hoạt động của hệ thống là main\_fp\_growth\_visualizer.py. Mục tiêu triển khai không chỉ dừng lại ở việc khai thác các frequent itemsets mà còn tập trung vào việc minh bạch hóa từng bước trong quy trình khai phá, từ tiền xử lý đến xây dựng cây và sinh luật.

Giao diện được chia thành hai khu vực chức năng:

* **Thanh bên (sidebar):** Cung cấp các công cụ để cấu hình dữ liệu đầu vào (chọn file, nhập tay), lọc dữ liệu (khách hàng, quốc gia), điều chỉnh tham số (min\_support, min\_confidence), và khởi chạy thuật toán.
* **Khu vực chính (main area):** Hiển thị toàn bộ nội dung chính như thông tin dữ liệu đã xử lý, các tab kết quả (thống kê, cây FP-Tree, frequent itemsets, luật kết hợp).

Quy trình vận hành chính được mô hình hóa thành các bước xử lý cụ thể:

* **Nhập dữ liệu:** Hỗ trợ ba phương thức: tải tệp (CSV, Excel), nhập định dạng danh sách (Groceries), và nhập định dạng giao dịch Tx: [item1, item2].
* **Tiền xử lý:** Bao gồm lọc giao dịch rỗng, xử lý dữ liệu nhiễu (quantity ≤ 0, các mã sản phẩm không hợp lệ như ‘POST’, hóa đơn hủy...), và chuẩn hóa thành danh sách giao dịch theo cấu trúc List[List[str]].
* **Thiết lập cấu hình thuật toán:** Người dùng có thể điều chỉnh ngưỡng support (dưới dạng %) và confidence, đồng thời giới hạn số nút được phép vẽ cây FP-Tree (tránh trình duyệt quá tải).
* **Chạy thuật toán:** Khi kích hoạt thuật toán “Chạy Thuật Toán FP-Growth”, hệ thống sẽ:
  + Tạo cây FP-Tree
  + Lưu vết (trace) các bước trung gian của thuật toán như đếm 1-itemset, tạo cây điều kiện, xây dựng Conditional Pattern Base (CPB)
  + Sinh tập mục phổ biến
  + Sinh luật kết hợp
* **Lưu session:** Trong ứng dụng, để đảm bảo rằng kết quả tính toán và các bước trung gian của thuật toán FP-Growth không bị mất khi người dùng tương tác với giao diện (ví dụ: thay đổi tham số, chuyển tab hoặc cập nhật giao diện), hệ thống sử dụng cơ chế lưu trạng thái phiên làm việc thông qua biến st.session\_state – một thành phần do Streamlit cung cấp để hỗ trợ quản lý trạng thái giữa các lần rerun (chạy lại ứng dụng).
* **Trực quan hóa:** Sử dụng utils.visualizers.py để hiển thị cây FP-Tree, bảng luật, bảng tập phổ biến.

### Cấu hình tham số và chuẩn hóa dữ liệu đầu vào

Ứng dụng cho phép cấu hình các tham số thuật toán một cách linh hoạt thông qua giao diện trực quan:

* **Min Support (%):**
  + Được chọn bằng thanh trượt trong sidebar.
  + Giá trị này được chuyển đổi thành ngưỡng support tuyệt đối bằng công thức:  
    min\_support\_count = ceil(min\_support\_percentage \* số giao dịch)
  + Ngưỡng này được hiển thị rõ ràng bên dưới thanh trượt.
* **Min Confidence (%):**
* Được dùng để lọc các luật có độ tin cậy thấp.
* Mặc định là 50%, nhưng có thể điều chỉnh từ 1% đến 100%.
* **Các tùy chọn lọc nâng cao:**
  + Lọc theo CustomerID và Country: Giúp tập trung phân tích theo từng đối tượng khách hàng hoặc khu vực cụ thể trong bộ dữ liệu Online Retail.
  + Áp dụng làm sạch chuyên biệt: Tùy chọn giúp loại bỏ dữ liệu không hợp lệ (Quantity ≤ 0, mã sản phẩm 'POST', hóa đơn bị hủy (InvoiceNo bắt đầu bằng ‘C’)...).
* **Chuẩn hóa dữ liệu đầu vào:**
  + Tất cả các định dạng đầu vào đều được chuyển về cấu trúc danh sách giao dịch (List[List[str]]), để chuẩn bị cho thuật toán FP-Growth xử lý.
  + Trong trường hợp nhập tay, các hàm xử lý được định nghĩa trong utils.data\_loader.py sẽ phân tách các item bằng ký tự phân cách do người dùng chỉ định và loại bỏ phần không cần thiết.

### Kết quả khai phá luật kết hợp

Sau khi thuật toán hoàn tất, hệ thống hiển thị kết quả:

* **Tổng Quan & Số Liệu:**
  + Tổng thời gian thực thi, bộ nhớ ban đầu, bộ nhớ sau cùng, và mức tiêu thụ bộ nhớ đỉnh được thu thập bằng mô-đun PerformanceMetrics.
  + Hiển thị số lượng nút FP-Tree và số cây điều kiện đã được xây dựng.
* Có bảng chi tiết các bước xử lý chính: tên bước, thời gian, bộ nhớ trước/sau và chênh lệch.
* **Bước Trung Gian & FP-Tree:**
  + Cho phép theo dõi toàn bộ tiến trình thuật toán.
  + Với mỗi bước như đếm item đơn, tạo CPB, hoặc tạo cây điều kiện, sẽ có tiêu đề, ghi chú, bảng dữ liệu hoặc trực quan cây.
  + Nếu số nút trong cây vượt ngưỡng đã định (MAX\_NODES\_FOR\_ GRAPHICAL\_VIEW), hệ thống sẽ chỉ hiển thị bảng Header Table thay vì toàn bộ cấu trúc cây.
* **Tập Mục Phổ Biến:**
  + Liệt kê tất cả các itemset thỏa mãn min\_support.
  + Mỗi dòng gồm tập item, số lần xuất hiện (support count), và phần trăm hỗ trợ.
* **Luật Kết Hợp:**
  + Danh sách các luật có độ tin cậy ≥ min\_confidence.
  + Các trường được trình bày: antecedents, consequents, support, confidence, lift.
  + Dữ liệu hiển thị dưới dạng bảng có thể sắp xếp theo từng cột.

### Phân tích sâu các đặc điểm của luật khai phá

Việc khai phá luật kết hợp không chỉ dừng lại ở việc liệt kê luật mà còn cung cấp cơ sở để phân tích các đặc điểm ẩn sau dữ liệu:

* Tính liên kết mạnh:
  + Những luật có confidence cao (trên 70%) và lift lớn hơn 1.5 thường chỉ ra các mối quan hệ tiêu dùng mạnh mẽ, ví dụ: các sản phẩm "JUMBO BAG" → "POSTAGE" thường được mua cùng nhau.
  + Các luật này rất hữu ích trong các chiến lược gợi ý sản phẩm (recommendation) hoặc tạo gói bán hàng (product bundling).
* Trực quan hóa FP-Tree:
  + Giao diện cho phép người dùng quan sát trực tiếp cấu trúc FP-Tree để hiểu được cách thuật toán tổ chức và nén dữ liệu giao dịch.
  + Đây là điểm nổi bật giúp sinh viên, nhà phân tích hoặc nhà nghiên cứu dễ dàng hiểu rõ nguyên lý hoạt động của thuật toán.
* Khả năng kiểm tra từng bước:
  + Mỗi bước xử lý đều có thể mở rộng để xem thông tin chi tiết như: số item đơn phổ biến, CPB của từng item, hoặc cây điều kiện tương ứng.
  + Việc lưu lại từng bước trong intermediate\_steps giúp ứng dụng minh bạch và dễ kiểm thử hơn.
* Tùy biến theo nhu cầu phân tích:
* Với khả năng lọc dữ liệu theo khách hàng hoặc quốc gia, ứng dụng có thể được sử dụng trong các bối cảnh cụ thể, chẳng hạn như:
  + - Phân tích hành vi mua sắm của khách hàng từ UK
    - Xác định sản phẩm bán kèm phổ biến theo vùng miền
* Đồng thời, các tham số như min\_support và min\_confidence có thể điều chỉnh để phù hợp với từng mục tiêu: phát hiện nhiều luật (ngưỡng thấp) hay lọc ra những luật mạnh nhất (ngưỡng cao).

## Thiết kế và cài đặt ứng dụng

### Kiến trúc tổng quan của ứng dụng

Ứng dụng khai phá luật kết hợp được triển khai trên nền tảng Streamlit, với kiến trúc chia module rõ ràng theo nguyên tắc separation of concerns (phân tách chức năng), giúp đảm bảo khả năng mở rộng, bảo trì và kiểm thử độc lập từng phần. Mỗi module đảm nhận một vai trò cụ thể trong quy trình từ tải dữ liệu, xử lý, chạy thuật toán đến trực quan hóa kết quả.

Toàn bộ hệ thống bao gồm 6 thành phần chính, mỗi thành phần là một module độc lập nhưng phối hợp chặt chẽ với nhau thông qua luồng dữ liệu và tín hiệu điều khiển. Kiến trúc tổng thể được trình bày như sau:

**utils/data\_loader.py – Module tải và xử lý dữ liệu:**  Module chịu trách nhiệm tải và xử lý dữ liệu. Chức năng chính bao gồm đọc file dữ liệu (CSV), làm sạch dữ liệu (loại bỏ dòng thiếu thông tin, dòng trả hàng...), và tổ chức lại dữ liệu thành danh sách giao dịch theo từng khách hàng và hóa đơn. Đây là một trong những thành phần quan trọng nhất, chịu trách nhiệm:

* Đọc dữ liệu gốc: Nhận đầu vào là file .csv chứa dữ liệu Online Retail (hoặc do người dùng tải lên).
* Tiền xử lý dữ liệu:
* Loại bỏ các dòng thiếu thông tin CustomerID hoặc Description vì không thể xác định được giao dịch hoặc sản phẩm.
* Bỏ các giao dịch trả hàng (Quantity < 0).
* Loại bỏ các mã sản phẩm đặc biệt như 'POST', 'MANUAL', 'DOT', v.v.
* Chuẩn hóa dữ liệu: Xử lý các mã StockCode không hợp lệ hoặc có ký tự đặc biệt (như 'POST', 'DOT', 'C2',...).
* Chuyển đổi cấu trúc dữ liệu: Gom các dòng dữ liệu theo InvoiceNo và CustomerID để tạo một giao dịch duy nhất cho mỗi đơn hàng.
* Trả về danh sách các giao dịch dưới dạng:

|  |
| --- |
| transactions = [['item\_1', 'item\_2', ...], 'item\_3', ['item\_4', 'item\_5', 'item\_6'], ...] |

* Dữ liệu này được sử dụng làm đầu vào cho các thuật toán khai phá.
* **algorithms/apriori\_logic.py – Thuật toán Apriori:** Module cài đặt thuật toán Apriori. Sử dụng thư viện hỗ trợ từ mlxtend, module này thực hiện việc sinh tập mục thường xuyên và trích xuất luật kết hợp thỏa mãn các ngưỡng do người dùng thiết lập (min\_support, min\_confidence, min\_lift). Sử dụng thư viện mlxtend.frequent\_patterns, gồm các chức năng cụ thể như:
* apriori() để trích xuất frequent itemsets (tập mục thường xuyên).
* association\_rules() để sinh các luật kết hợp từ tập mục.
* Quy trình thực hiện:
* Dữ liệu từ data\_loader.py được chuyển thành dạng One-hot encoded DataFrame.
* Áp dụng hàm apriori() với min\_support do người dùng chọn.
* Sinh các luật với min\_confidence và min\_lift tương ứng.
* Trả về bảng chứa luật: Antecedent, Consequent, Support, Confidence, Lift.
* **algorithms/fp\_growth\_logic.py – Thuật toán FP-Growth**: Tương tự như apriori\_logic.py, nhưng thay thế bằng thuật toán FP-Growth, sử dụng thư viện hỗ trợ từ mlxtend, module này thực hiện việc sinh tập mục thường xuyên và trích xuất luật kết hợp thỏa mãn các ngưỡng do người dùng thiết lập (min\_support, min\_confidence, min\_lift).

Ưu điểm:

* Không cần sinh tập mục ứng viên.
* Nhanh hơn rõ rệt trên tập dữ liệu lớn hoặc nhiều giao dịch.

Cách triển khai:

* Sử dụng hàm fpgrowth() từ mlxtend để tìm frequent itemsets.
* Sau đó dùng association\_rules() để sinh luật kết hợp.
* Kết quả đầu ra tương tự Apriori (cho phép so sánh).
* **utils/metrics\_collector.py – Đo hiệu năng**: Module thu thập và ghi nhận các chỉ số hiệu năng, đặc biệt là thời gian thực thi. Kết quả này được sử dụng để so sánh tốc độ giữa các thuật toán và đánh giá hiệu quả xử lý. Sử dụng module time và psutil để:
* Ghi lại thời gian bắt đầu – kết thúc khi chạy thuật toán.
* Có thể mở rộng để đo RAM, CPU sử dụng trong quá trình xử lý.
* Các thông tin này được truyền về giao diện nhằm:
* Hiển thị cho người dùng.
* So sánh khách quan giữa Apriori và FP-Growth.
* **utils/visualizers.py – Trực quan hóa kết quả:** Module trực quan hóa kết quả. Bao gồm các hàm vẽ bảng luật kết hợp (dạng DataFrame) và biểu đồ mạng lưới thể hiện mối quan hệ giữa các sản phẩm (dùng thư viện PyVis). Module này tạo ra các biểu diễn trực quan hỗ trợ phân tích và đánh giá kết quả:

Bảng luật kết hợp:

* Dùng pandas.DataFrame để hiển thị các luật.
* Có thể lọc/sắp xếp theo support, confidence, lift.
* Biểu đồ mạng lưới (Network Graph):
* Dùng thư viện pyvis.network.Network.
* Các nút đại diện cho sản phẩm; liên kết thể hiện luật kết hợp.

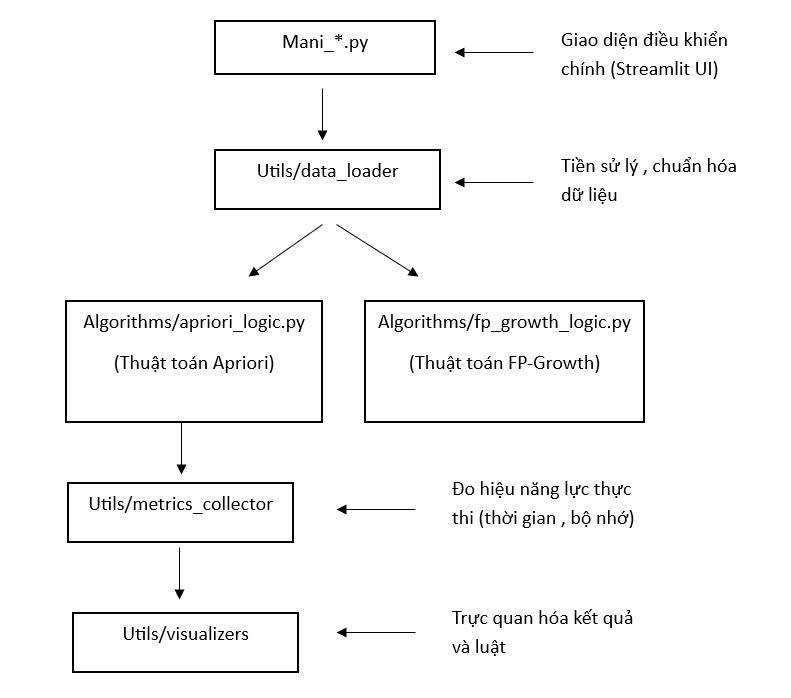
Có thể điều chỉnh:

* Số lượng luật hiển thị (top N theo lift).
* Màu sắc theo từng nhóm sản phẩm.
* Hiệu ứng tương tác (phóng to, highlight theo cụm...).
* **main\_apriori\_visualizer.py và main\_fp\_growth\_visualizer.py – Giao diện Streamlita:** Đây là các tập tin chính để chạy ứng dụng Streamlit (tức là các entry-point (điểm khởi động)) tương ứng với mỗi thuật toán. Giao diện cho phép người dùng tải file dữ liệu, chọn bộ lọc, cấu hình tham số và quan sát kết quả đầu ra dưới nhiều hình thức trực quan:
* Mỗi file tương ứng với một thuật toán.
* Gồm các khối giao diện:
* Upload dữ liệu: Người dùng chọn file CSV.
* Lọc dữ liệu: Tùy chọn lọc theo Country hoặc CustomerID.
* Tham số đầu vào: Các slider điều chỉnh min\_support, confidence, lift.
* Nút chạy: Gửi dữ liệu vào pipeline và chạy thuật toán.
* Hiển thị kết quả: Gồm bảng luật kết hợp, biểu đồ mạng lưới, thời gian thực thi.

Ưu điểm:

* Không yêu cầu kỹ năng lập trình từ phía người dùng.
* Cho phép kiểm thử thuật toán nhanh chóng trên nhiều cấu hình và phân vùng dữ liệu.

**Sơ đồ thể hiện các mối quan hệ của các modules:**

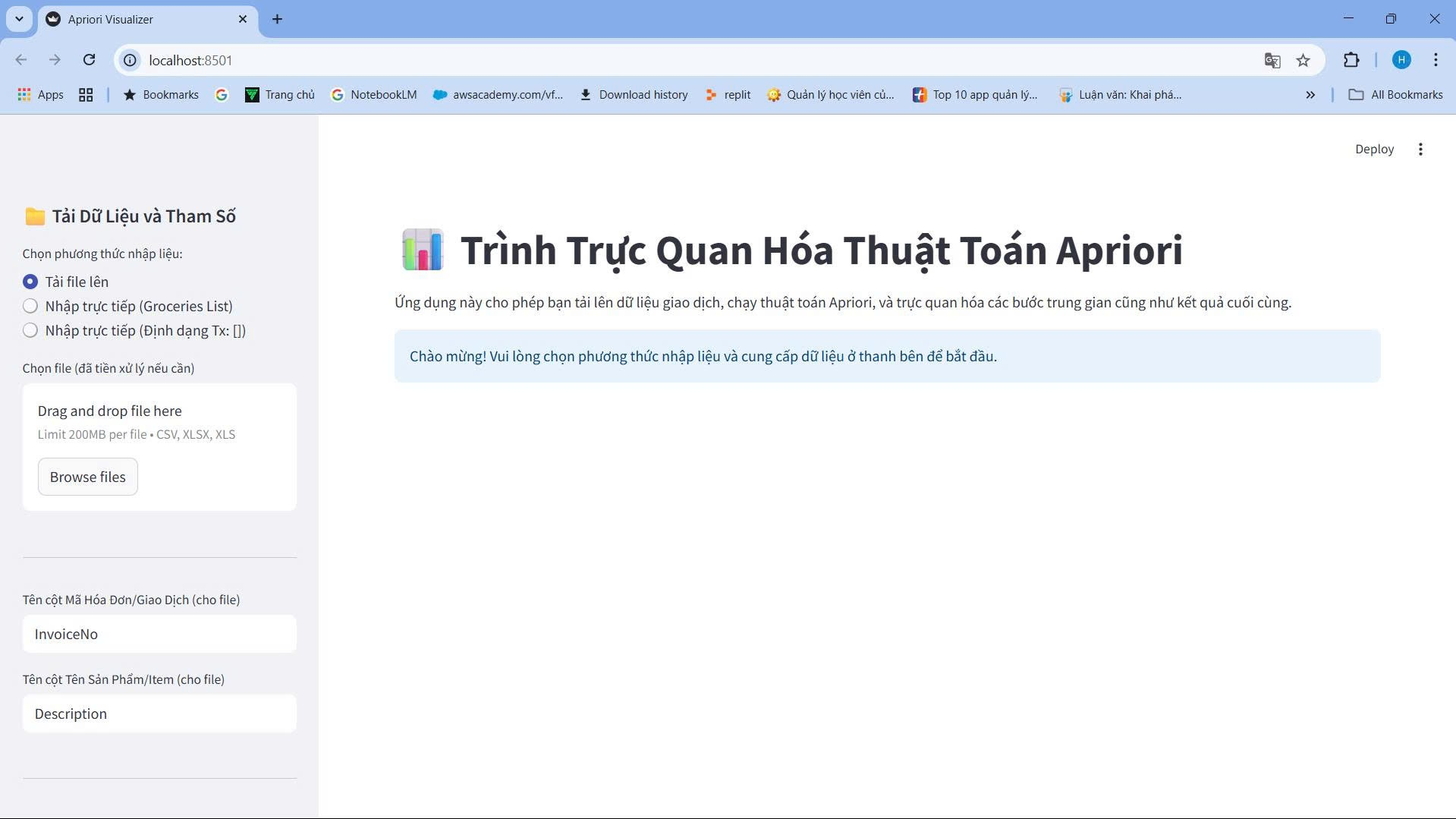


Hình 3.1 Mối quan hệ giữa các modules

### Giao diện người dùng:

#### Mô tả tổng quan

Giao diện người dùng của ứng dụng được thiết kế theo hướng tối giản, trực quan và độc lập cho từng thuật toán khai phá luật kết hợp. Cụ thể, mỗi thuật toán (Apriori và FP-Growth) được triển khai trong một tệp Streamlit riêng biệt – main\_apriori\_visualizer.py và main\_fp\_growth\_visualizer.py. Người dùng khởi chạy ứng dụng bằng cách thực thi trực tiếp lệnh streamlit run trong môi trường lập trình như VS Code, từ đó mở ra giao diện web tương tác.



Hình 3.2 Giao diện trực quan của ứng dụng của Apriori



Hình 3.3 Giao diện trực quan của ứng dụng của FP\_Growth

Việc phân chia giao diện theo thuật toán mang lại nhiều lợi ích:

* Tối ưu hóa trải nghiệm người dùng: Mỗi giao diện được thiết kế phù hợp riêng với quy trình và đặc điểm thuật toán, tránh gây nhầm lẫn hoặc chồng chéo dữ liệu.
* Hỗ trợ so sánh kết quả: Khi chạy hai giao diện đồng thời ở các cửa sổ trình duyệt khác nhau, người dùng có thể dễ dàng so sánh trực tiếp hiệu suất (thời gian chạy), tập luật kết hợp và biểu đồ trực quan của hai thuật toán.
* Tách biệt quy trình xử lý: Việc phân tách giúp dễ dàng mở rộng hoặc chỉnh sửa riêng lẻ cho từng thuật toán mà không ảnh hưởng đến phần còn lại của hệ thống.

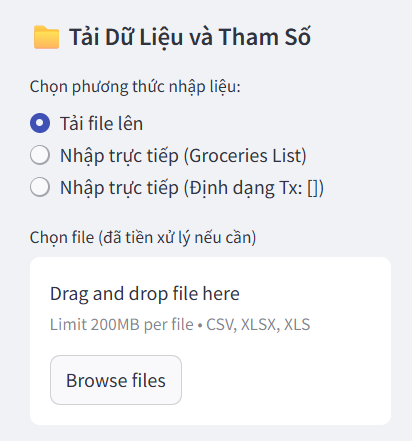
Toàn bộ giao diện được xây dựng trên nền tảng Streamlit với hai khu vực chức năng chính: Sidebar (dùng để cấu hình và điều khiển) và Main Area (hiển thị toàn bộ nội dung kết quả và biểu đồ). Giao diện này không đòi hỏi người dùng có kiến thức lập trình, mà chỉ cần thao tác kéo thả, nhập số và nhấn nút để thực hiện phân tích dữ liệu.

#### Các thành phần của giao diện

Giao diện được thiết kế gồm các thành phần sau:

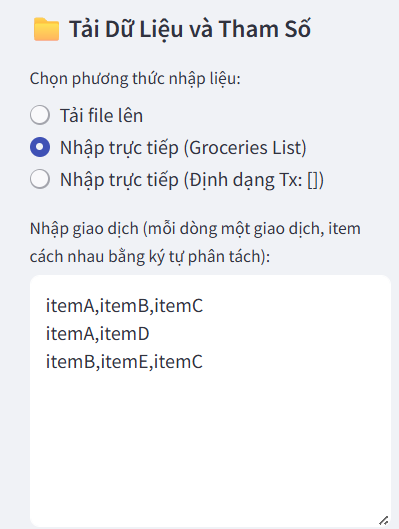
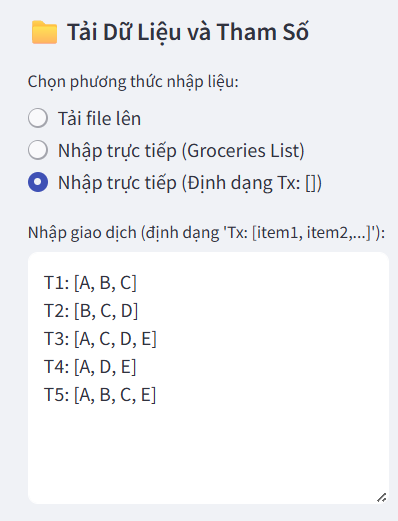
Phần tải dữ liệu và tham số:

* Cho phép người dùng tải file dữ liệu từ máy tính để dễ dàng lấy dữ liệu từ nhiều nguồn hệ thống hỗ trợ định dạng .csv, xlsx worksheet phổ biến, điển hình như bộ dữ liệu Online Retail.



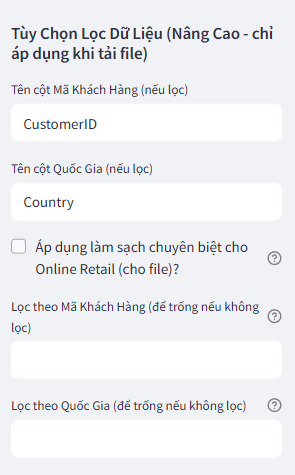
Hình 3.4 Ứng dụng cho phép tải dữ liệu từ máy tính

* Hoặc cho phép người dùng nhập trực tiếp trên ứng dụng: theo định dạng Tx:[] hoặc theo định dạng Groceries List.

Hình 3.5 Ứng dụng cho phép nhập dữ liệu trực tiếp theo nhiều định dạng

* Tiếp theo, là giao diện cấu hình nhập dữ liệu cho phép người dùng xác định tên các cột quan trọng trong file dữ liệu giao dịch, bao gồm:
* Mã hóa đơn/giao dịch (InvoiceNo)
* Tên sản phẩm hoặc mặt hàng (Description)
* Hỗ trợ tùy chọn lọc dữ liệu nâng cao:
* Lọc theo mã khách hàng (CustomerID)
* Lọc theo quốc gia (Country)
* Áp dụng quy trình làm sạch dữ liệu chuyên biệt cho bộ dữ liệu Online Retail (nếu tích chọn).
* Lọc riêng theo mã khách hàng hoặc quốc gia (có thể bỏ trống nếu không cần lọc).
* Giúp chuẩn hóa dữ liệu đầu vào, đảm bảo các thuật toán khai phá luật kết hợp hoạt động đúng trên từng trường dữ liệu phù hợp.



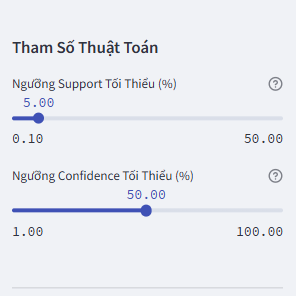
Hình 3.6 Ứng dụng cho phép tùy chọn để lọc dữ liệu

Ảnh này là phần giao diện cho phép người dùng thiết lập tham số thuật toán khi khai phá luật kết hợp bằng các thuật toán như Apriori hoặc FP-Growth.

Cụ thể, giao diện này dùng để:

* Điều chỉnh ngưỡng support tối thiểu (%): Xác định tần suất xuất hiện tối thiểu của một tập hợp mặt hàng để được xem là phổ biến trong toàn bộ dữ liệu giao dịch.
* Điều chỉnh ngưỡng confidence tối thiểu (%): Xác định độ tin cậy tối thiểu của một luật kết hợp để được chấp nhận (tức là xác suất mặt hàng B xuất hiện khi đã mua mặt hàng A).

Việc thiết lập hai tham số này giúp kiểm soát số lượng và chất lượng các luật kết hợp được khai phá từ dữ liệu, đồng thời ảnh hưởng trực tiếp đến kết quả phân tích và hiệu suất của thuật toán.



Hình 3.7 Ứng dụng linh hoạt bằng cách cho phép điều chỉnh tham số

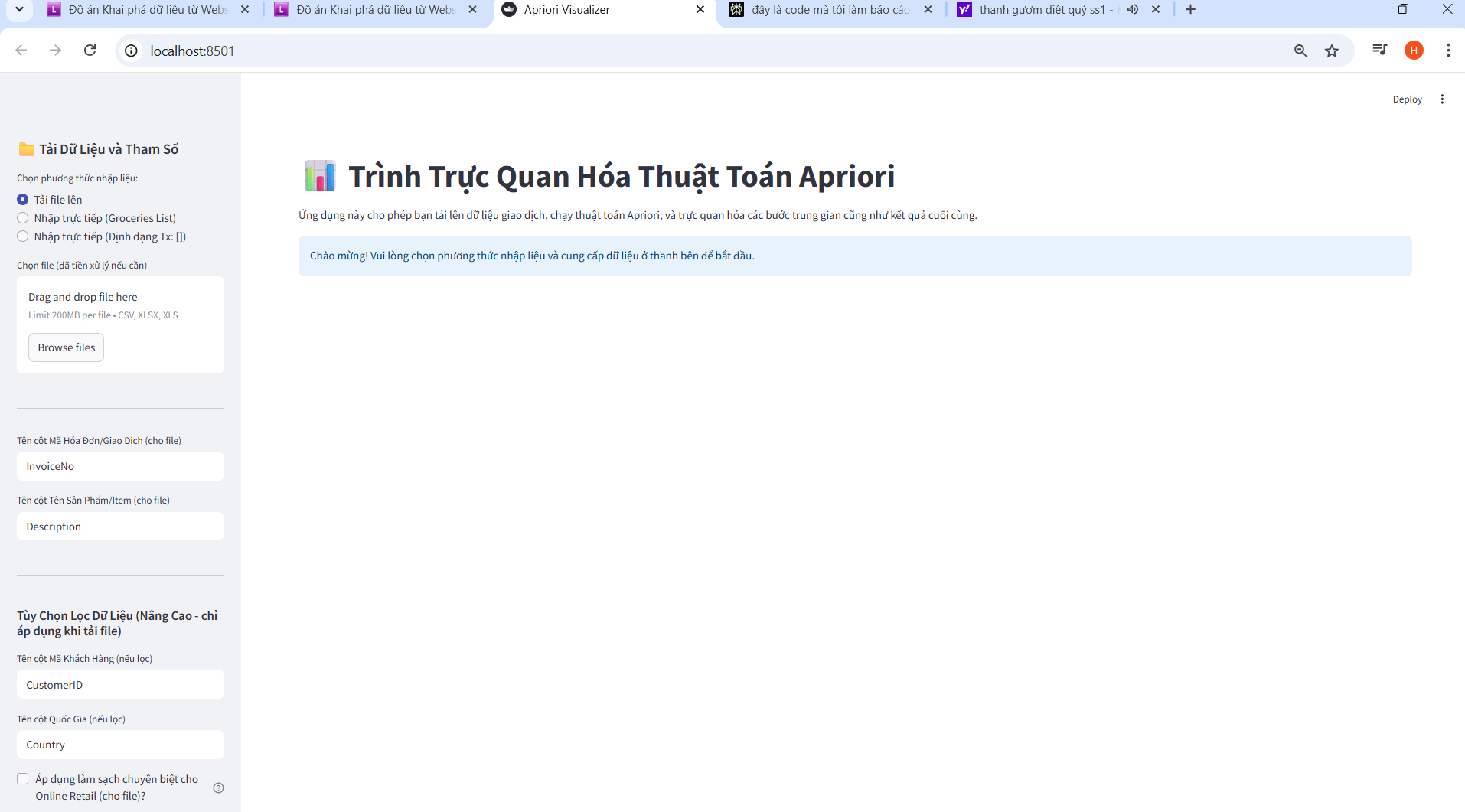
Tiếp theo chính là nơi để trực quan hóa thực toán giúp người dùng dễ tìm hiểu hơn về cách hoạt động và kết quả của thuật toán. Giao diện này không chỉ hiển thị thông tin tổng quan về dữ liệu đầu vào và sau xử lý, mà còn cung cấp các số liệu hiệu năng như thời gian thực thi, mức sử dụng bộ nhớ, số lượng tập ứng viên, số tập mục phổ biến tìm được cũng như các chỉ số chi tiết khác.

Với thiết kế trực quan, người dùng có thể:

* Tùy chỉnh tham số (support, confidence), chạy thử nghiệm nhiều lần để đánh giá hiệu quả.
* Quan sát từng bước trung gian của thuật toán, từ việc sinh tập ứng viên, lọc tập phổ biến cho đến sinh luật kết hợp.
* So sánh hiệu suất và kết quả giữa các lần chạy hoặc với các thuật toán khác.
* Hiển thị bảng luật kết hợp: Các luật mạnh nhất được trình bày với đầy đủ thông tin (tập tiền đề, tập kết quả, support, confidence, lift).
* Trực quan hóa mạng lưới luật kết hợp: Các luật mạnh nhất được vẽ thành biểu đồ mạng lưới, giúp người dùng dễ dàng nhận diện các nhóm sản phẩm thường mua cùng nhau, hoặc các mối quan hệ nổi bật giữa các mặt hàng.
* Trình bày các bước trung gian: Ứng dụng hiển thị chi tiết từng bước của thuật toán (tập ứng viên, tập mục phổ biến ở mỗi vòng lặp), giúp người dùng hiểu rõ quy trình khai phá.



Hình 3.8 Giao diện của thuật toán FP\_Growth



Hình 3.8 Giao diện của thuật toán Apriori

# KẾT QUẢ THỰC NGHIỆM

## Thiết lập thực nghiệm

Để đảm bảo khả năng triển khai và kiểm tra hiệu quả mô hình khai phá luật kết hợp trên dữ liệu thực tế, nhóm đã thiết lập một môi trường thực nghiệm ổn định và phổ biến. Cụ thể như sau:

**Phần cứng sử dụng:**

Máy tính cá nhân với bộ vi xử lý Intel Core i5, RAM 16GB. Cấu hình này đủ đảm bảo khả năng xử lý dữ liệu có kích thước vừa phải như bộ dữ liệu Online Retail.

**Hệ điều hành:**

Thử nghiệm được tiến hành trên cả hai nền tảng:

* Windows 10 – môi trường phổ biến cho phát triển ứng dụng desktop và dễ tích hợp với các công cụ Python.
* Ubuntu 20.04 – môi trường mã nguồn mở thân thiện với các công cụ và thư viện khoa học dữ liệu, tối ưu cho việc chạy các ứng dụng Streamlit và xử lý nền.

Ngôn ngữ và công nghệ phát triển:

* Python – ngôn ngữ chính sử dụng cho việc xử lý dữ liệu và xây dựng mô hình.
* Streamlit – framework nhẹ dùng để xây dựng giao diện người dùng (UI) phục vụ trực quan hóa kết quả và tương tác với mô hình.
* Pandas – thư viện xử lý dữ liệu bảng hiệu quả, dùng để làm sạch và phân tích dữ liệu.
* psutil - thư viện Python đa nền tảng dùng để lấy thông tin về các tiến trình đang chạy và mức sử dụng tài nguyên hệ thống như CPU, bộ nhớ RAM, ổ đĩa, mạng, cảm biến,...
* Graphviz - Thư viện dùng để trực quan hóa các quan hệ, mạng lưới luật kết hợp hoặc các cấu trúc cây (như FP-Tree), giúp minh họa kết quả khai phá dữ liệu một cách trực quan, sinh động và dễ hiểu.

**Thông số thử nghiệm**

Bộ dữ liệu sử dụng:

* Nguồn dữ liệu: Online Retail – một bộ dữ liệu thương mại điện tử công khai, ghi lại các giao dịch bán hàng trực tuyến từ năm 2010 đến 2011.
* Dữ liệu được lọc theo quốc gia "United Kingdom" – là quốc gia có số lượng giao dịch nhiều nhất, nhằm đảm bảo tính đại diện và thống nhất.

Sau khi xử lý làm sạch dữ liệu, bộ dữ liệu còn lại:

* 24.446 giao dịch (số lượng hóa đơn không trùng lặp).
* 4.211 sản phẩm duy nhất (mã sản phẩm sau khi loại bỏ các mặt hàng lỗi, bị huỷ hoặc không hợp lệ).

Tham số khai phá mặc định:

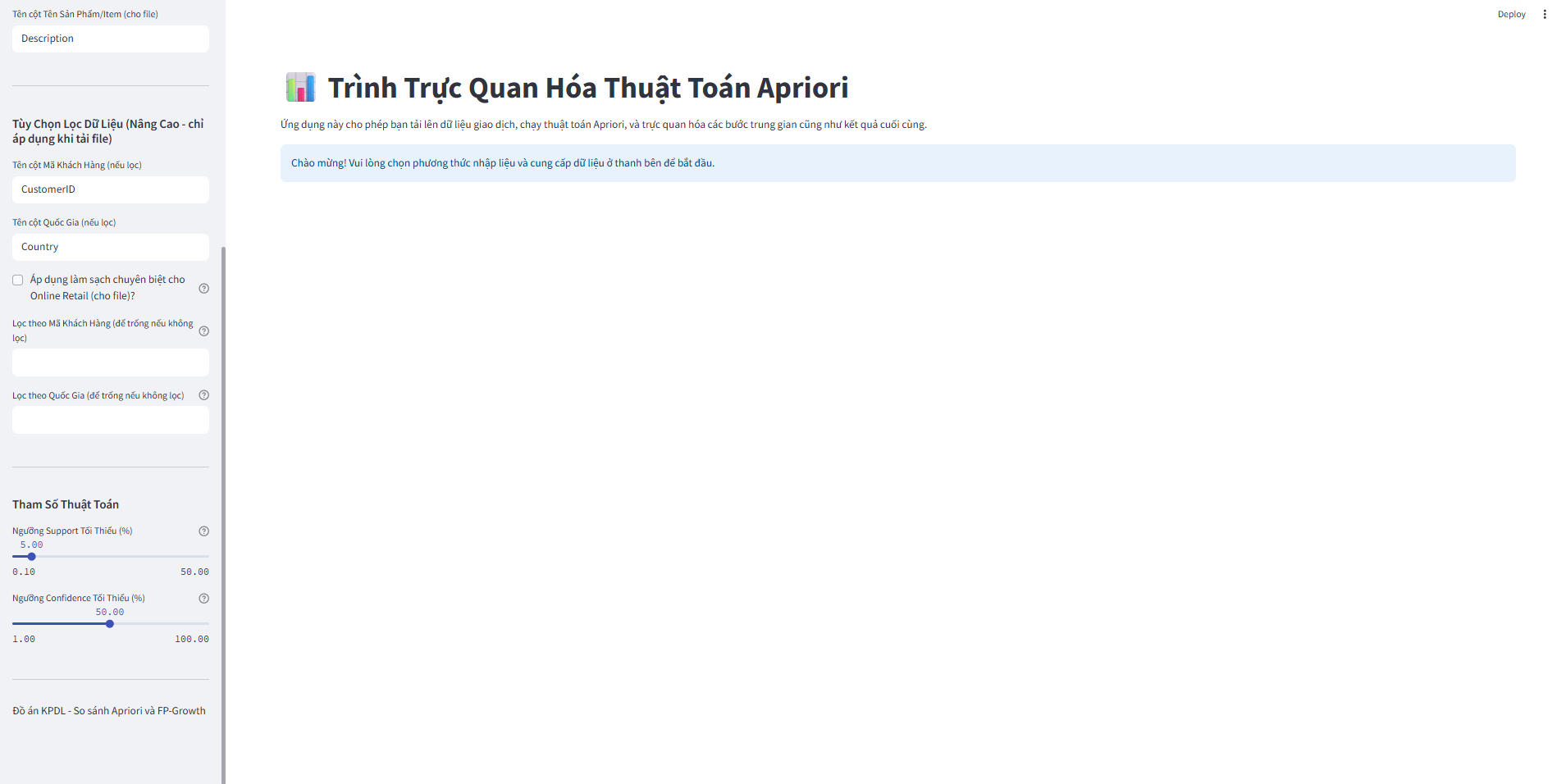
* min\_support = 0.02: Chỉ xét các tập mục xuất hiện trong ít nhất 2% tổng số giao dịch, nhằm lọc ra các mẫu có ý nghĩa thống kê.
* min\_confidence = 0.2: Luật kết hợp được chấp nhận nếu độ tin cậy đạt ít nhất 20%, tức là trong số các giao dịch chứa vế trái, có ít nhất 20% cũng chứa vế phải.

**Mục tiêu thử nghiệm:**

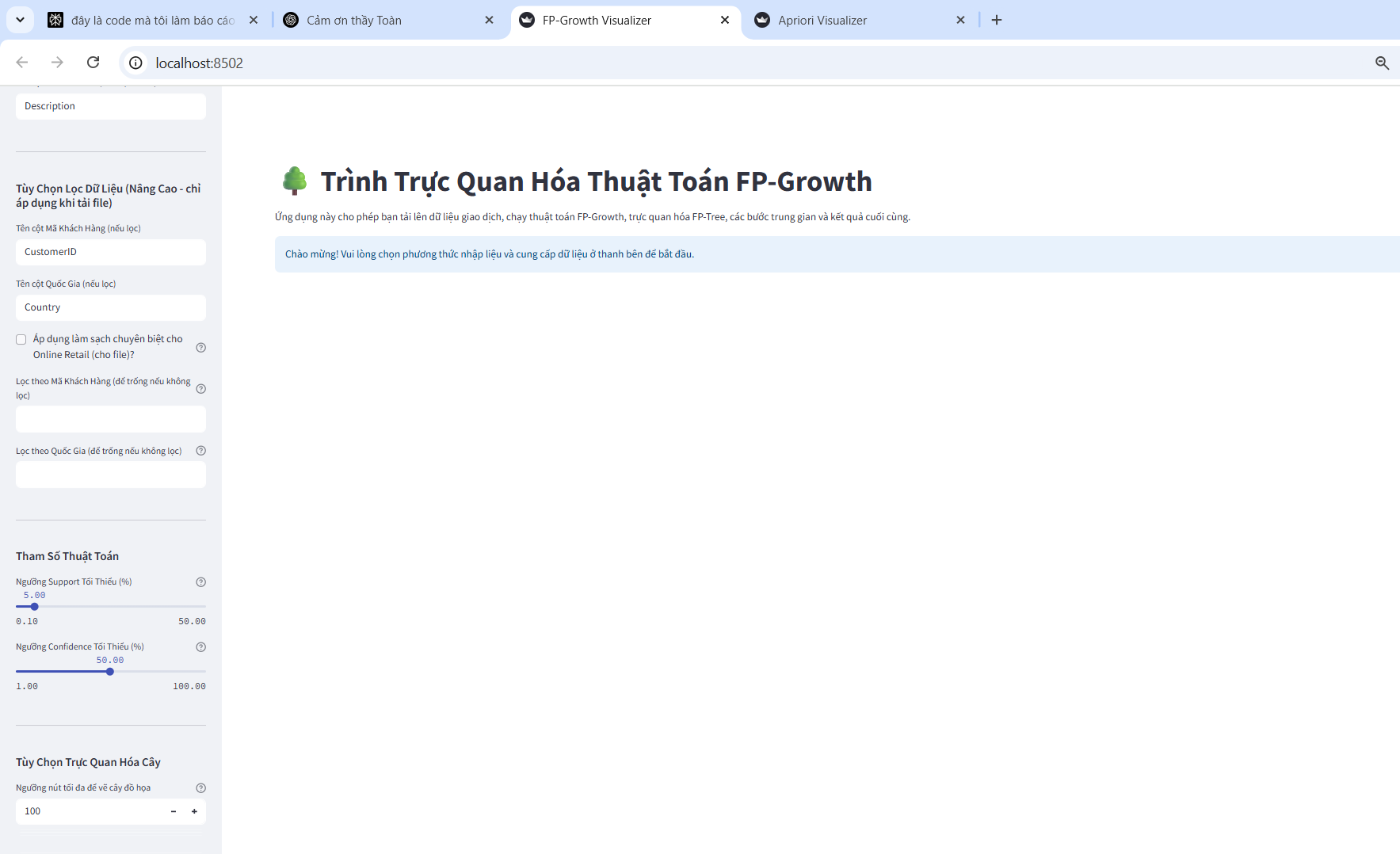
* Đánh giá khả năng khai thác luật kết hợp hữu ích từ dữ liệu bán hàng thực tế.
* So sánh hiệu quả giữa hai thuật toán Apriori và FP-Growth dưới cùng điều kiện dữ liệu và tham số.

## Biện luận

Để đảm bảo tính khách quan và minh bạch, cả hai thuật toán Apriori và FP-Growth đã được chạy trên cùng một bộ dữ liệu "Online Retail" (lọc theo quốc gia "United Kingdom", với 24.446 giao dịch và 4.211 sản phẩm duy nhất), cùng một máy tính (CPU Intel Core i5, RAM 16GB), và cùng phiên bản phần mềm (Python 3.8, các thư viện như Pandas, Streamlit, Graphviz ở phiên bản mới nhất). Các tham số khai phá cũng được thiết lập giống nhau: min\_support = 0.1 và min\_confidence = 0.5.



Hình 4.1 Thông số khi triển khai vào Apriori

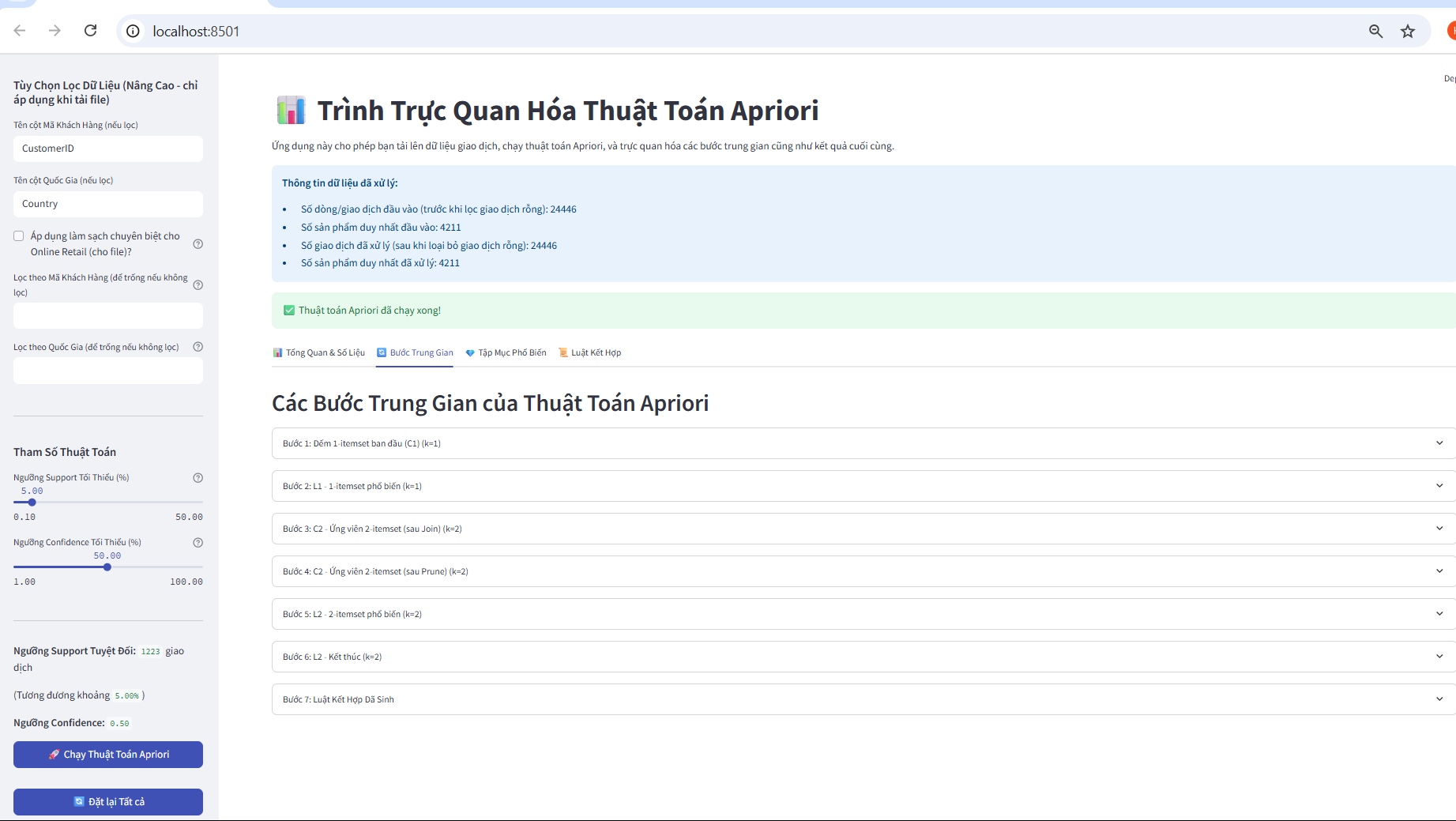


Hình 4.1 Thông số khi triển khai vào FT\_Growth

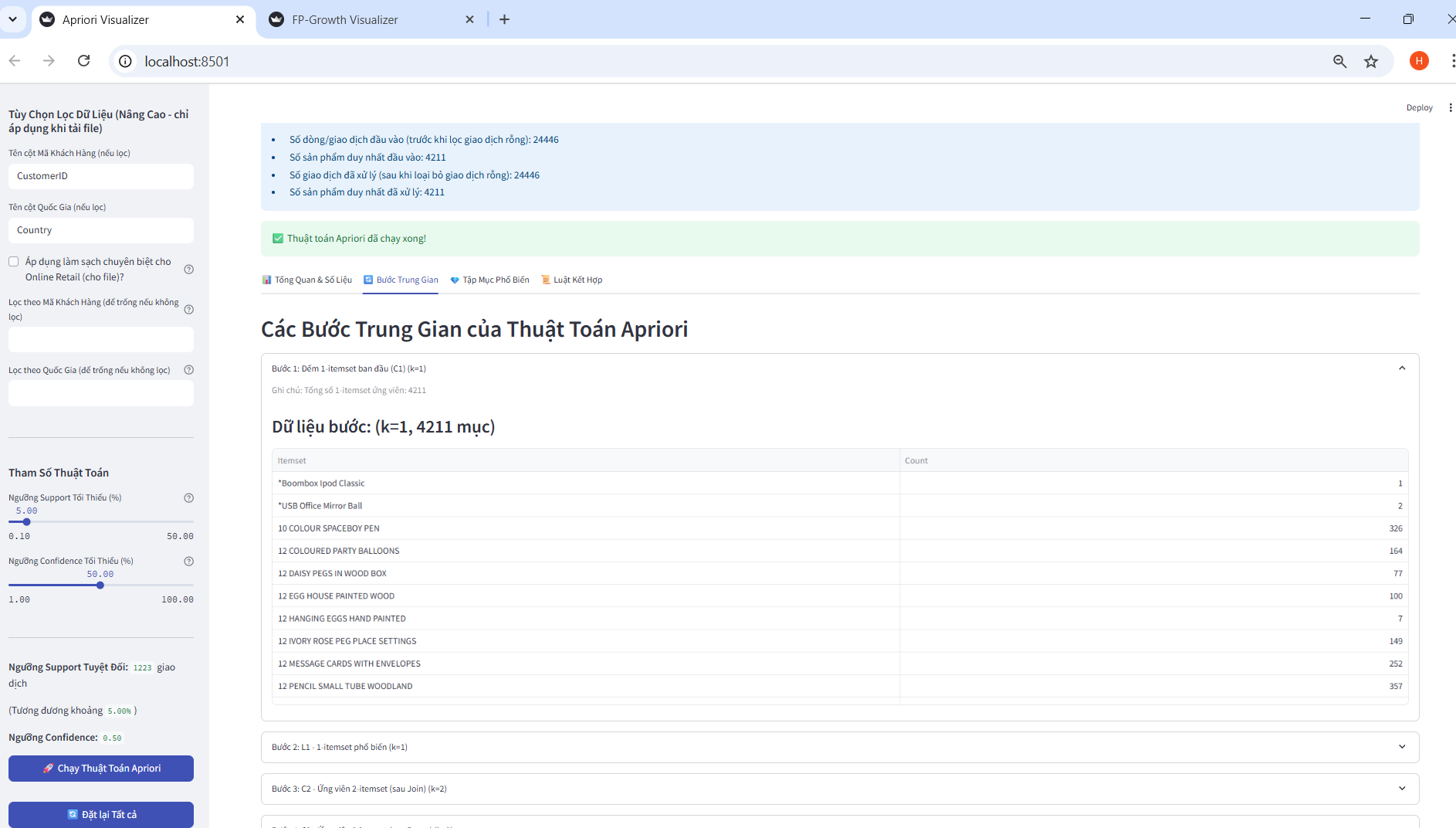
Các luật này đều đáp ứng ngưỡng support, confidence và đặc biệt có lift lớn hơn 1, cho thấy mối quan hệ kết hợp thực sự chặt chẽ giữa các sản phẩm.

Tiếp theo, ứng dụng sẽ ghi lại và tính toán số bước mà cả hai thuật toán đã trải qua, đối với mỗi bước ứng dụng cũng sẽ tính toán số mục mà bước đó thực hiện và liệt kê ra để người dùng có thể dễ dàng quan sát, trực quan hơn cách hoạt động của thuật toán.

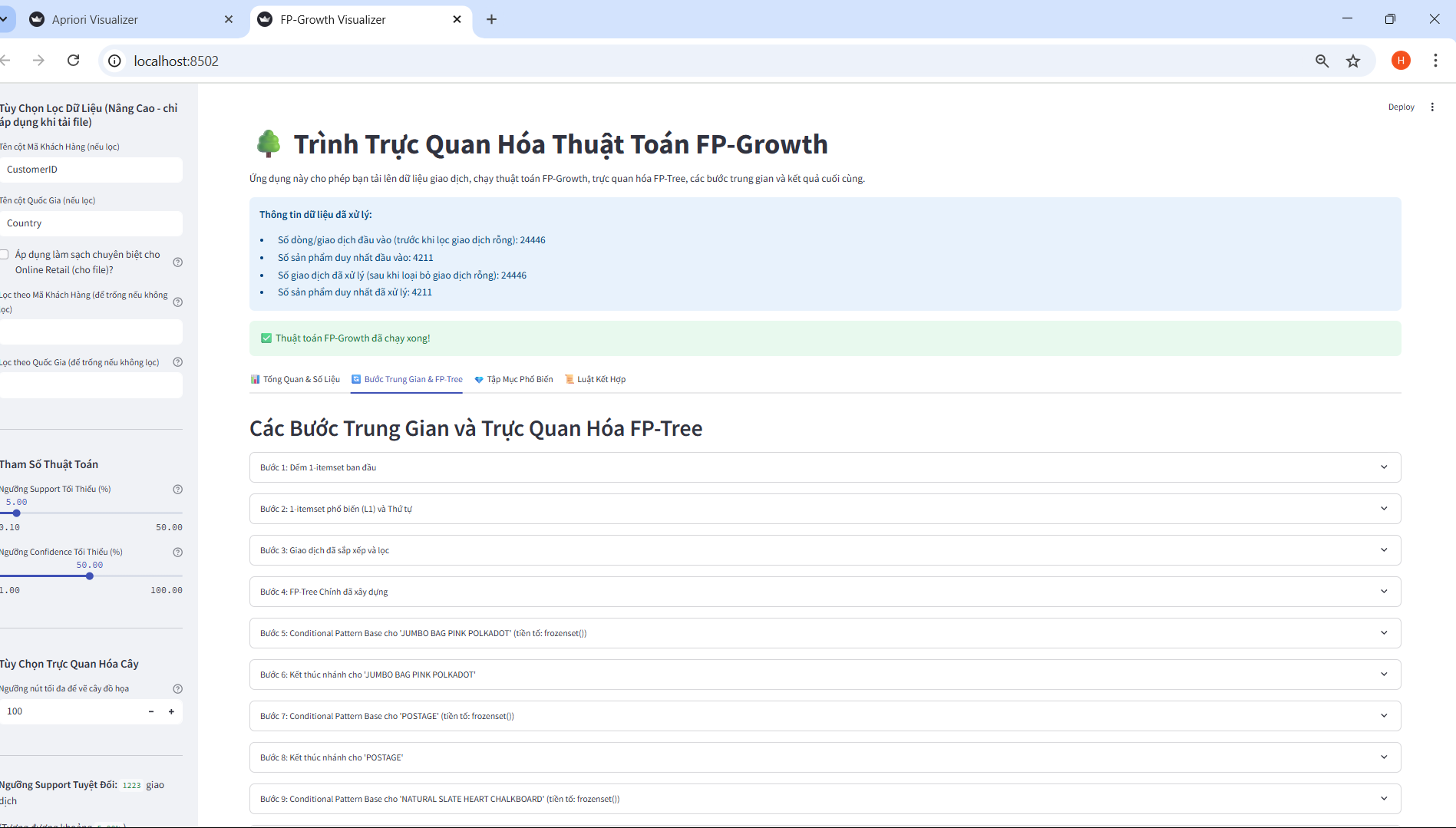
Ứng dụng vừa cho phép xem số bước mà từng thuật toán đó hoạt động, ở mỗi bước hiển thị đầy đủ các thông tin về số danh mục, item phổ biến, bước k,… để người dùng tìm hiểu trực quan và dễ dàng hơn.



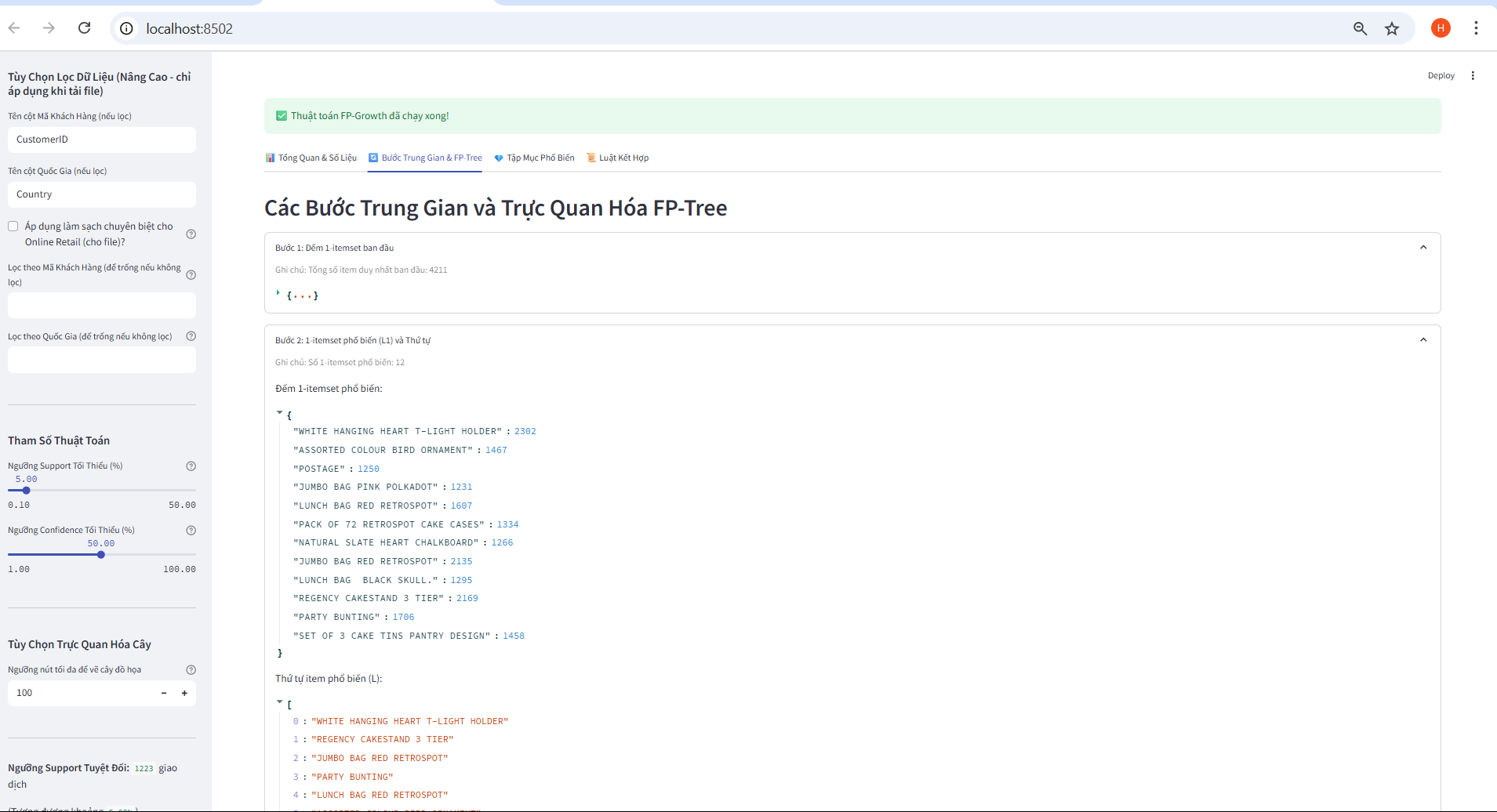
Hình 4.3 Ứng dụng hiển thị các bước thực hiện của Apriori



Hình 4.4 Mỗi bước sẽ hiển thị chi tiết về thông số apriori sử dụng

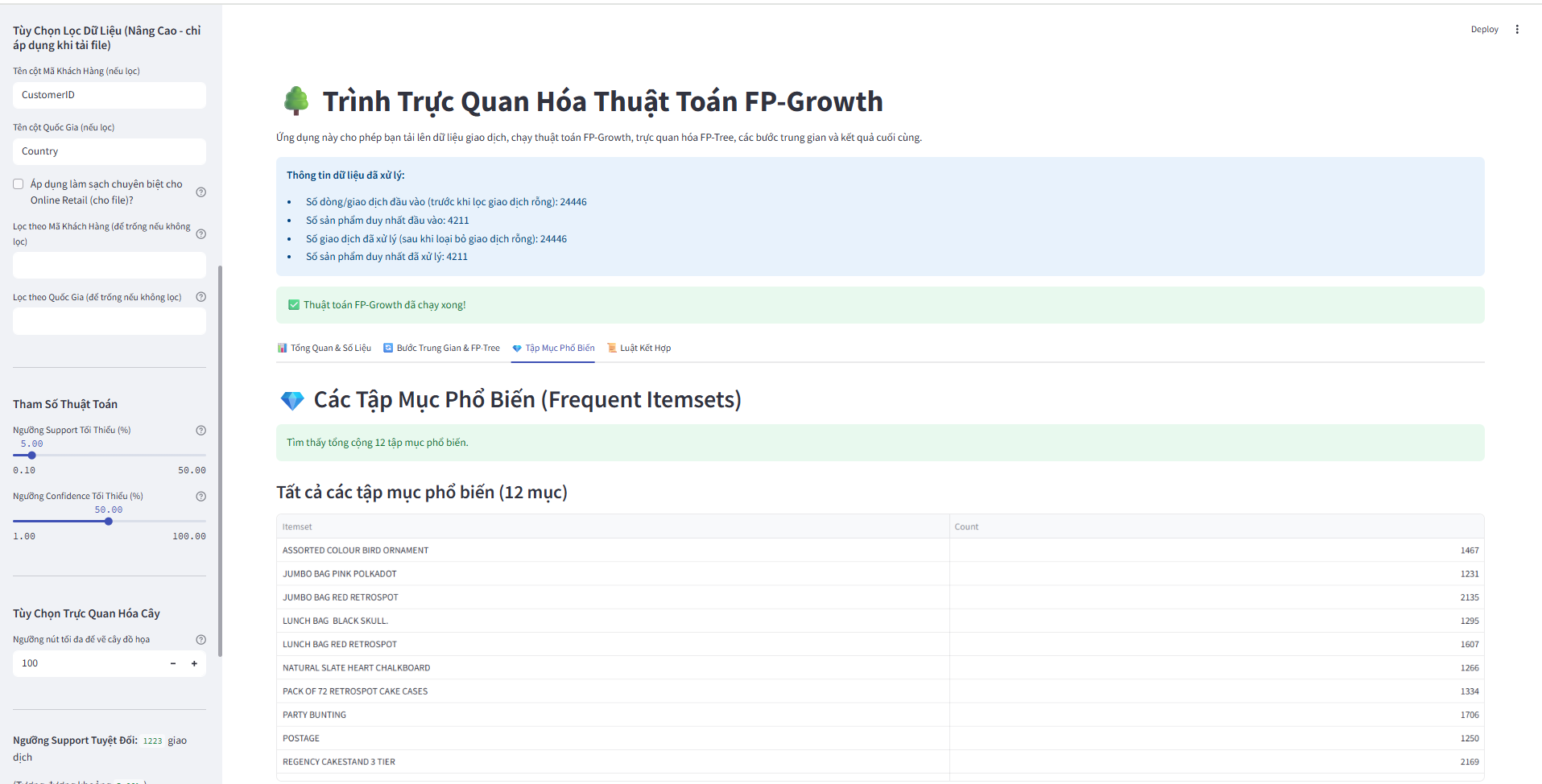


Hình 4.5 Ứng dụng hiển thị các bước thực hiện của FP\_Growth

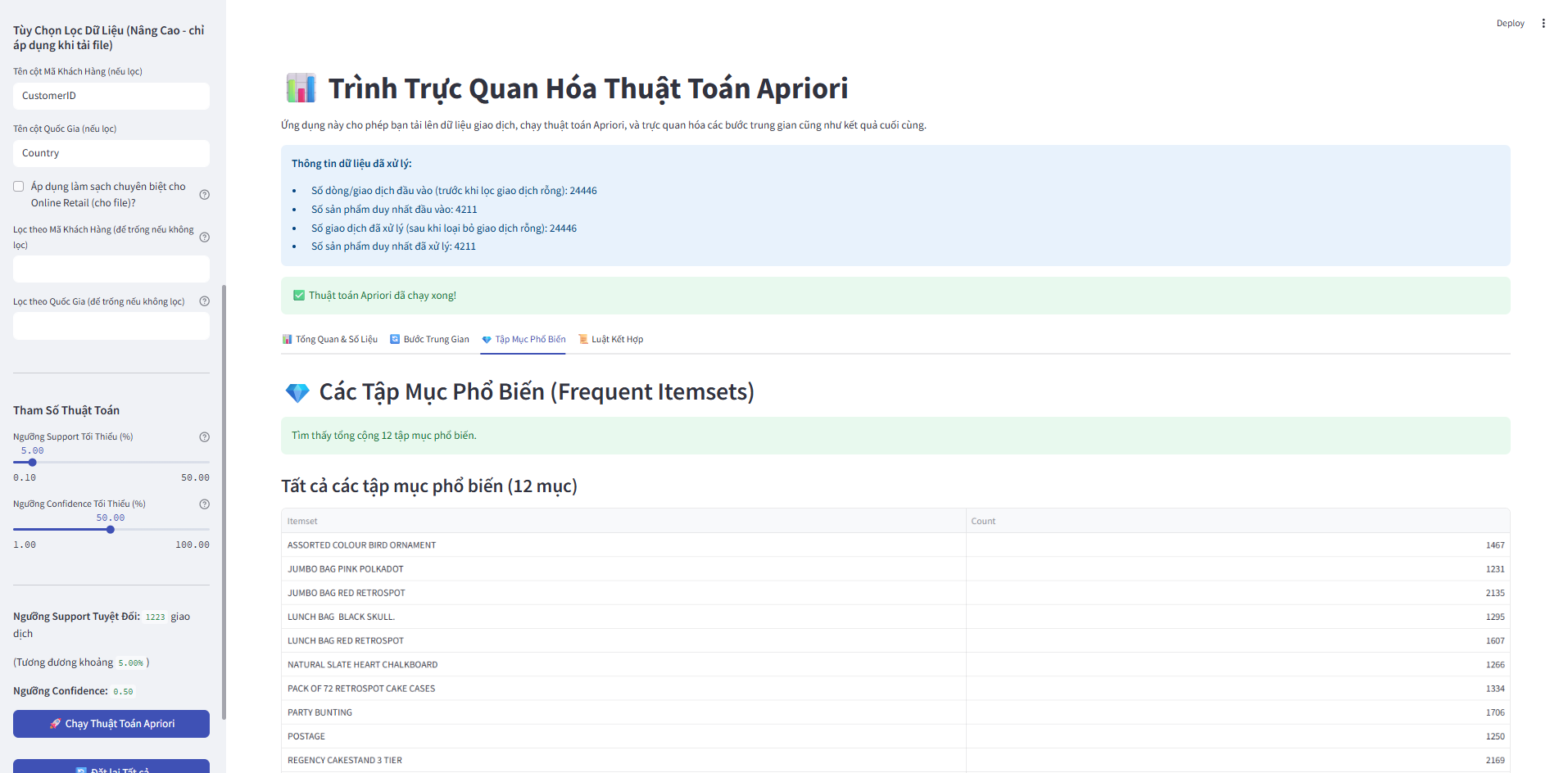


Hình 4.6 Mỗi bước sẽ hiển thị chi tiết về thông số FP\_Growth sử dụng

Tiếp theo ứng dụng cho phép chung ta xem tập phổ biến một cách trực quan nhất, Sau khi thực hiện khai phá luật kết hợp, cả hai thuật toán đều thu được các luật kết hợp mạnh nhất có giá trị lift cao tương đồng nhau về mặt nội dung.

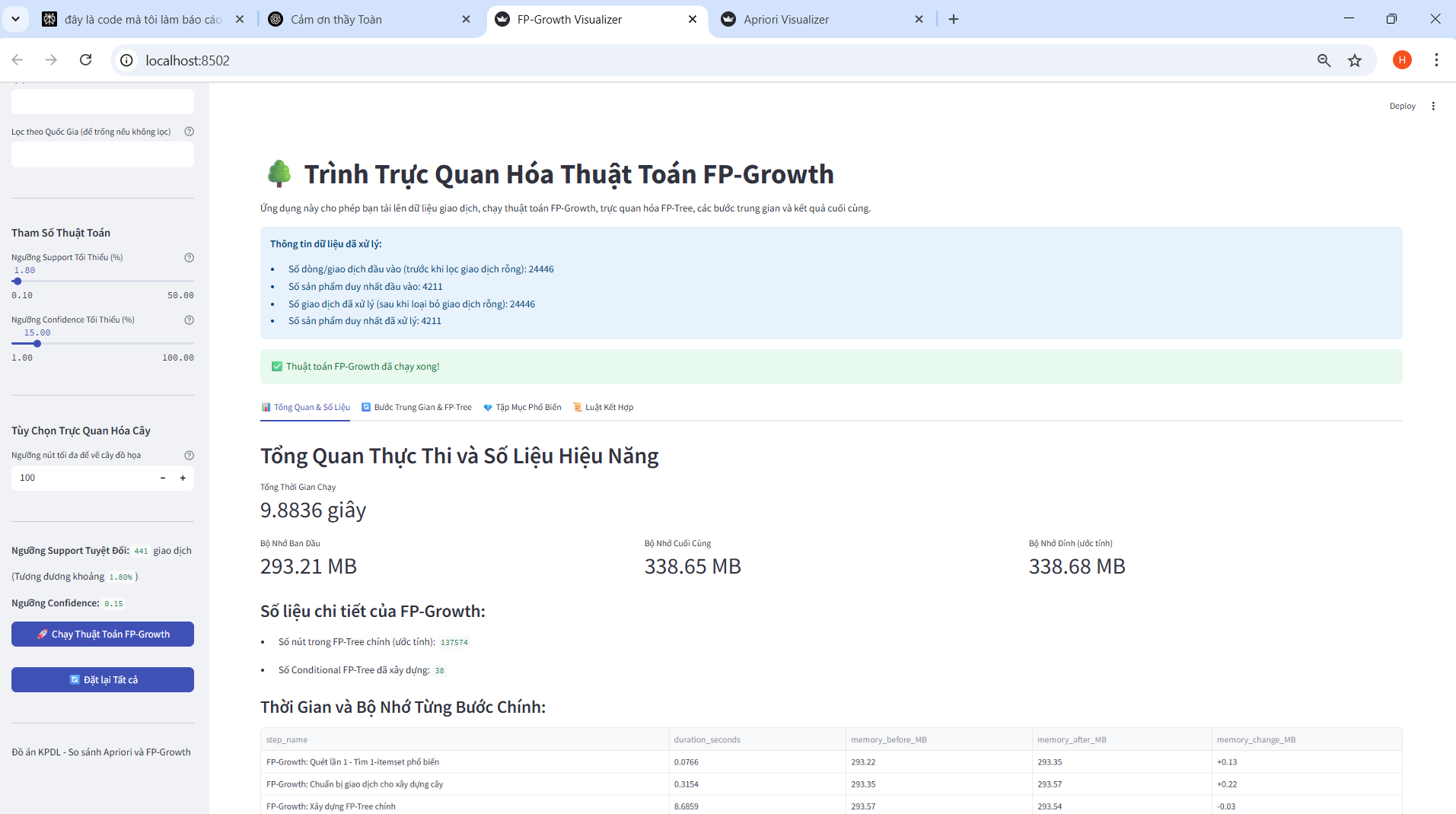


Hình 4.7 Ứng dụng hiển thị tập phổ biến mà FP\_Growth tìm được

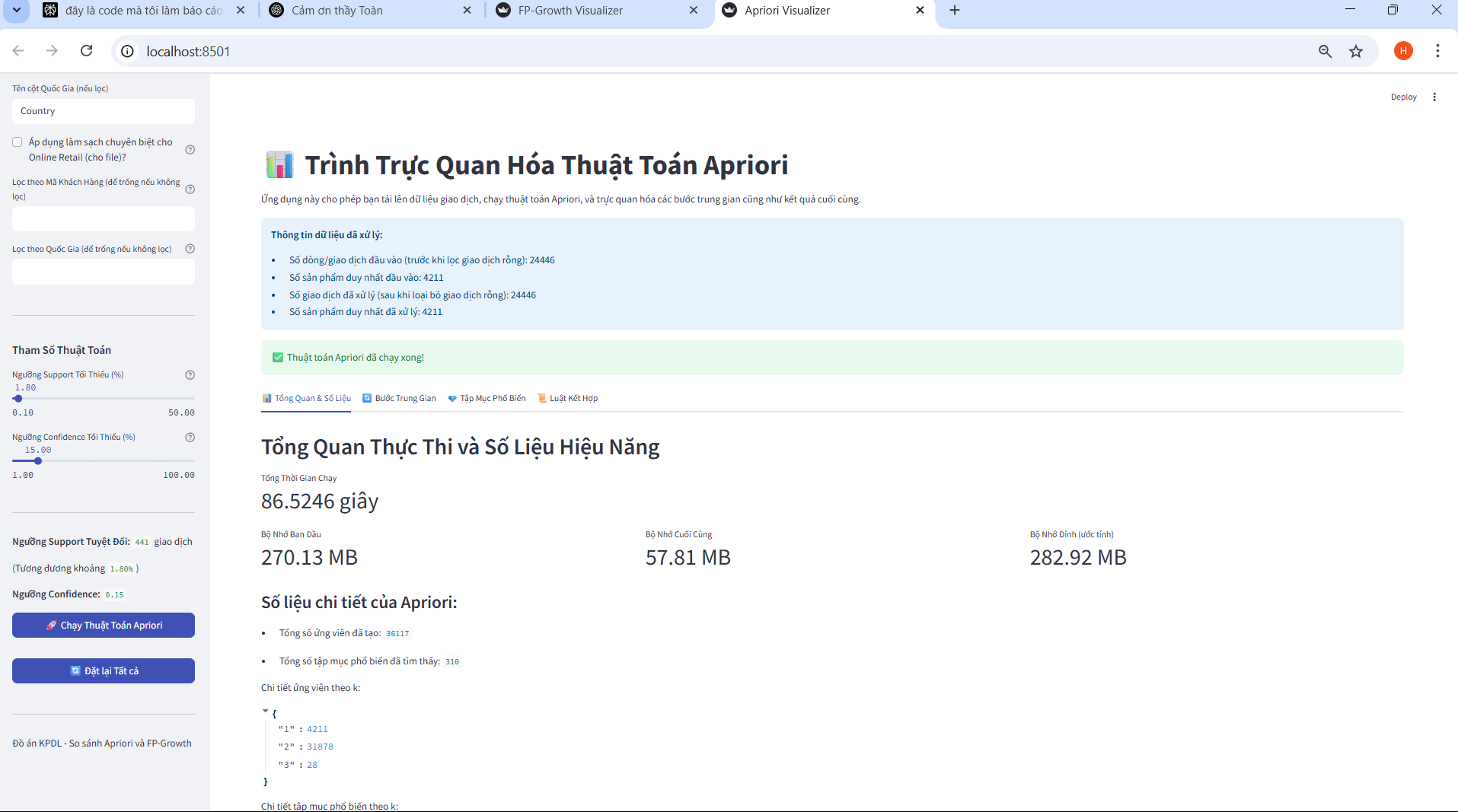


Hình 4.8 Ứng dụng hiển thị tập phổ biến mà Apriori tìm được

Ứng dụng còn sử dụng các công cụ và các thư viện hỗ trợ sẵn để tiến hành đo hiệu năng mà bộ nhớ đã tiêu tốn từ đó người dùng có thể so sánh với nhau và nắm rõ hơn về từng thuật toán.



Hình 4.9 Ứng dụng ghi lại hiệu năng của FP\_Growth



Hình 4.10 Ứng dụng ghi lại hiệu năng của FP\_Growth

Cả hai giao diện đều thể hiện:

* Bộ dữ liệu đầu vào giống nhau: 24.446 giao dịch, 4.211 sản phẩm duy nhất.

**Tham số thuật toán tương tự nhau:**

* Ngưỡng support tối thiểu: 0.018 (tương ứng 441 giao dịch).
* Ngưỡng confidence tối thiểu: 0.15.

Điều này đảm bảo tính công bằng khi so sánh kết quả và hiệu suất giữa hai thuật toán.

**Thời gian thực thi**

* FP-Growth: 9.8836 giây
* Apriori: 86.5246 giây

Từ thông số trên đã cho thấy FP-Growth thực thi nhanh gấp gần 9 lần so với Apriori trên cùng bộ dữ liệu và tham số. Điều này phản ánh đúng bản chất lý thuyết: FP-Growth chỉ cần quét dữ liệu 2 lần và không sinh tập ứng viên, trong khi Apriori phải sinh và kiểm tra rất nhiều tập ứng viên ở mỗi vòng lặp, khiến thời gian tăng mạnh khi dữ liệu lớn hoặc min\_support thấp.

**Sử dụng bộ nhớ**

FP-Growth:

* Bộ nhớ ban đầu: 293.21 MB
* Bộ nhớ cuối cùng: 338.65 MB
* Bộ nhớ đỉnh (ước tính): 338.68 MB

Apriori:

* Bộ nhớ ban đầu: 270.13 MB
* Bộ nhớ cuối cùng: 57.81 MB
* Bộ nhớ đỉnh (ước tính): 282.92 MB

Nhận xét:

* FP-Growth sử dụng nhiều bộ nhớ hơn trong quá trình thực thi, do cần xây dựng và lưu trữ FP-Tree cùng các Conditional FP-Tree.
* Apriori có mức sử dụng bộ nhớ cuối cùng thấp hơn đáng kể, nhưng bộ nhớ đỉnh vẫn cao (282.92 MB) do phải lưu trữ nhiều tập ứng viên trong các bước trung gian.

Sự khác biệt này là hợp lý và phản ánh đúng đặc trưng của từng thuật toán.

**Số liệu chi tiết về thuật toán**

FP-Growth:

* Số nút trong FP-Tree chính: 137.574
* Số Conditional FP-Tree đã xây dựng: 38

Apriori:

* Tổng số tập ứng viên đã tạo: 4.131
* Tổng số tập mục phổ biến tìm thấy: 12

Nhận xét:

FP-Growth tập trung vào việc xây dựng cấu trúc cây để nén dữ liệu và khai phá trực tiếp, số lượng node lớn nhưng hiệu quả xử lý cao.

Apriori phải sinh ra rất nhiều tập ứng viên, nhưng số tập mục phổ biến thực sự rất ít, cho thấy hiệu quả lọc của thuật toán nhưng cũng làm lộ rõ điểm yếu về hiệu suất.

**Trực quan hóa quá trình thực thi**

Cả hai giao diện đều cung cấp:

* Thông tin tổng quan về dữ liệu và tham số.
* Bảng thời gian và bộ nhớ cho từng bước chính (step\_name, duration\_seconds, memory\_before\_MB, memory\_after\_MB, memory\_change\_MB).
* Các tab cho phép xem chi tiết từng bước, tập mục phổ biến, luật kết hợp...

Điều này giúp người dùng dễ dàng theo dõi, so sánh và đánh giá hiệu suất cũng như quy trình hoạt động của từng thuật toán một cách trực quan và minh bạch.

## So sánh và đánh giá giữa hai thuật toán FP\_Growth và Apriori

### So sánh thực nghiệm trên cùng bộ dữ liệu

Cả hai thuật toán đều được thực thi trên bộ dữ liệu "Online Retail" với cùng số lượng giao dịch (24.446) và số sản phẩm duy nhất (4.211):

| **Thuật toán** | **Thời gian thực thi (giây)** | **Bộ nhớ ban đầu (MB)** | **Bộ nhớ cuối cùng (MB)** | **Bộ nhớ đỉnh (MB)** |
| --- | --- | --- | --- | --- |
| FP-Growth | 9.8836 | 287.52 | 289.61 | 289.63 |
| Apriori | 86.5246 | 220.14 | 230.12 | 235.37 |

Bảng 1 Thông số giữa 2 thuật toán apriori

* Thời gian thực thi: FP-Growth vượt trội, chỉ mất ~10 giây, trong khi Apriori mất hơn 80 giây trên cùng bộ dữ liệu và tham số.
* Sử dụng bộ nhớ: FP-Growth sử dụng nhiều bộ nhớ hơn một chút so với Apriori, do phải xây dựng và lưu trữ FP-Tree, nhưng sự chênh lệch không lớn và hoàn toàn hợp lý với tốc độ vượt trội đạt được.
* FP-Growth: bộ nhớ từ đầu đến cuối gần như ổn định, chỉ dao động từ 287.52 → 289.63 MB (~+2 MB). Điều này thể hiện hiệu quả của việc nén dữ liệu và giữ cấu trúc cây ổn định.
* Apriori: tuy bộ nhớ ban đầu thấp, nhưng tăng dần trong quá trình sinh và lọc ứng viên, lên đến đỉnh 235.37 MB, cho thấy thuật toán tạo nhiều đối tượng trung gian.
* Xét về hiệu quả tổng thể, FP-Growth tỏ ra vượt trội trong các bài toán có tập dữ liệu lớn, khi yêu cầu tốc độ cao và bộ nhớ đủ đáp ứng. Trong khi đó, Apriori phù hợp hơn với dữ liệu nhỏ hoặc khi tài nguyên bộ nhớ bị giới hạn nghiêm ngặt.
* Với các tập dữ liệu lớn như "Online Retail", FP-Growth là lựa chọn tối ưu nhờ khả năng nén và khai thác dữ liệu hiệu quả, giảm số lần quét cơ sở dữ liệu.

### Phân tích lí thuyết và thực tiễn

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Apriori | FP-Growth |
| **Nguyên lý hoạt động** | Sinh tập ứng viên, kiểm tra từng tập, loại bỏ không đủ support | Nén dữ liệu thành FP-Tree, khai phá trực tiếp trên cây, không sinh ứng viên |
| **Chiến lược tìm kiếm** | Tìm kiếm theo chiều rộng, xây dựng từ dưới lên | Chia để trị, phát triển mẫu trên cây FP-Tree |
| **Số lần quét dữ liệu** | Nhiều lần (mỗi cấp độ tập mục phải quét lại toàn bộ dữ liệu) | Chỉ cần 2 lần quét dữ liệu |
| **Sử dụng bộ nhớ** | Thường thấp hơn, nhưng có thể tăng mạnh nếu số ứng viên nhiều | Cao hơn do lưu FP-Tree, nhưng tối ưu hóa tốt |
| **Thời gian thực thi** | Tăng rất nhanh khi min\_support giảm hoặc dữ liệu lớn | Ổn định, nhanh hơn rất nhiều, đặc biệt khi dữ liệu lớn hoặc min\_support thấp |
| **Hiệu quả thực tế** | Phù hợp minh họa, dữ liệu nhỏ, dễ hiểu quy trình | Phù hợp thực tế, dữ liệu lớn, tốc độ cao, mở rộng dễ dàng |

Bảng 2 Tổng hợp so sánh giữa 2 thuật toán

### Giải thích nguyên nhân

Thuật toán Apriori là một trong những thuật toán khai phá tập mục phổ biến cổ điển và nổi tiếng. Tuy nhiên, điểm yếu lớn nhất của Apriori nằm ở cách tiếp cận theo chiến lược sinh và kiểm tra (generate-and-test). Ở mỗi vòng lặp, thuật toán cần tạo ra tất cả các tập ứng viên có kích thước k (Ck) bằng cách kết hợp các tập phổ biến từ vòng lặp trước (Lk-1). Sau đó, nó phải quét toàn bộ cơ sở dữ liệu giao dịch để đếm số lần xuất hiện (support) của từng tập ứng viên này và xác định xem nó có đạt ngưỡng min\_support hay không. Khi số lượng sản phẩm lớn (ví dụ: vài trăm đến hàng ngàn mặt hàng), số lượng tập hợp con có thể sinh ra là cực kỳ lớn. Cụ thể, với n mục phổ biến, số lượng tập hợp k-mục có thể lên đến C(n, k) – tức là tổ hợp chập k của n, dẫn đến việc số lượng ứng viên tăng theo cấp số mũ khi k tăng.

Ngược lại, FP-Growth được thiết kế để xử lý hiệu quả hơn bằng cách loại bỏ hoàn toàn bước sinh tập ứng viên. Thay vì sinh từng tập con và quét dữ liệu nhiều lần, FP-Growth chỉ cần quét cơ sở dữ liệu đúng hai lần. Lần đầu tiên để xác định các mục phổ biến và loại bỏ các mục không đạt min\_support. Lần thứ hai, các giao dịch được dùng để xây dựng một cấu trúc cây đặc biệt gọi là FP-Tree (Frequent Pattern Tree). Cây này lưu trữ các giao dịch dưới dạng các nhánh được chia sẻ – nghĩa là những giao dịch có chung tiền tố (prefix) sẽ được nén lại vào cùng một nhánh trên cây. Ví dụ: nếu nhiều giao dịch bắt đầu bằng A → B → C, FP-Tree sẽ chỉ lưu một nhánh duy nhất với bộ đếm tăng dần thay vì lặp lại dữ liệu, từ đó giảm đáng kể kích thước lưu trữ và tăng hiệu suất truy xuất.

Sau khi xây dựng FP-Tree, FP-Growth tiến hành khai phá bằng cách lấy từng mục phổ biến và xây dựng cây con có điều kiện (conditional FP-Tree) tương ứng. Từ cây con này, các tập mục phổ biến được khai thác một cách đệ quy, chỉ tập trung vào các nhánh thực sự có khả năng sinh ra luật kết hợp. Bằng cách này, FP-Growth khai thác trực tiếp từ cấu trúc cây, không sinh ứng viên, không quét lại toàn bộ dữ liệu nhiều lần, và chỉ thao tác trên các phần tử liên quan – điều này giúp tối ưu đáng kể về thời gian và bộ nhớ.

Với các tập dữ liệu lớn, có hàng trăm ngàn hoặc hàng triệu giao dịch, và min\_support thấp (ví dụ: 0.5%), Apriori gần như không thể thực thi hiệu quả vì số tập ứng viên là quá lớn và việc quét dữ liệu quá tốn kém. Trong khi đó, FP-Growth vẫn có thể hoạt động ổn định nhờ khả năng nén thông minh và khai thác cục bộ. Nhờ vậy, FP-Growth thường là lựa chọn ưu tiên trong thực tế khi xử lý dữ liệu thương mại điện tử, giỏ hàng siêu thị, hay các bài toán phân tích hành vi người dùng.

So sánh thực tế để có thể tổng quan hơn về hai thuật toán:

|  |  |  |
| --- | --- | --- |
| Khía cạnh | Apriori | FP\_Growth |
| Tuần suất quét | K lần (với k là kích thước tập mục lớn nhất) | 2 lần (đếm mục + xây dựng cây) |
| Sinh ứng viên | Sinh ra rất nhiều và kiểm tra trực tiếp | Không cần sinh ứng viên – khai phá dựa trên cây |
| Cấu trúc dữ liệu | Không lưu, mỗi vòng dựa trên dataset thô | FP\_Tree nén nhiều chi tiết và giữ lại mối quan hệ giữa các mục |
| Hiệu năng | Kém khi min\_sup thấp hoặc nhiều mục vụn | Rất hiệu quả ngay cả khi dataset lớn, min\_sup thấp |

Bảng 3 So sánh thực tế giữa hai thuật toán

# KẾT LUẬN VÀ HƯỚNG PHÁT

## Tổng kết các kết quả đạt được

Sau quá trình nghiên cứu và thực nghiệm, đề tài đã hoàn thành toàn diện các mục tiêu đề ra, bám sát theo mô hình CRISP-DM và tập trung vào hai thuật toán khai phá luật kết hợp tiêu biểu: Apriori và FP-Growth. Kết quả thu được thể hiện rõ qua các phương diện sau:

* Về lý thuyết: Đề tài đã tiến hành phân tích sâu cơ chế hoạt động của hai thuật toán, chỉ ra điểm tương đồng và khác biệt trong cách xây dựng tập mục phổ biến, cũng như hiệu quả xử lý trong các tình huống dữ liệu khác nhau. Những ưu điểm (như khả năng tiết kiệm bộ nhớ của FP-Growth) và nhược điểm (như độ phức tạp trong xây dựng cây) đều được thể hiện rõ ràng.
* Về xử lý dữ liệu: Dữ liệu dạng giỏ hàng (transactional data) được làm sạch, mã hóa và chuyển đổi về định dạng phù hợp với yêu cầu thuật toán. Việc chuẩn hóa này giúp tăng độ chính xác và đảm bảo tính khách quan khi đánh giá hiệu quả của từng phương pháp.
* Về xây dựng mô hình và ứng dụng: Đề tài đã triển khai thực nghiệm trên cả hai thuật toán với cùng một tập dữ liệu thực tế, cùng ngưỡng tham số (support và confidence), giúp đảm bảo tính nhất quán trong đánh giá. Đồng thời, một ứng dụng giao diện đồ họa (GUI) đã được xây dựng trên nền tảng Streamlit, cho phép người dùng nhập dữ liệu, thực thi thuật toán và trực quan hóa tập luật bằng bảng, biểu đồ và sơ đồ cây tần suất.
* Về kết quả và đánh giá: Kết quả khai phá đã phát hiện được nhiều luật kết hợp có ý nghĩa, phản ánh rõ nét hành vi tiêu dùng của khách hàng. So sánh giữa hai thuật toán cho thấy FP-Growth có tốc độ xử lý nhanh hơn và tiêu tốn ít bộ nhớ hơn Apriori, trong khi Apriori lại dễ hiểu và dễ kiểm soát nhờ vào việc sinh tổ hợp tuần tự.

## Hạn chế

Bên cạnh các kết quả đạt được, đề tài vẫn tồn tại một số hạn chế cần được nhận diện rõ ràng để làm cơ sở cho việc cải tiến trong tương lai:

* Chất lượng và đặc thù dữ liệu: Bộ dữ liệu "Online Retail" mặc dù mang tính thực tế cao, song vẫn tồn tại tình trạng thiếu mã hóa thống nhất, các giao dịch hủy bỏ hoặc mã hàng trùng lặp, đòi hỏi quá trình làm sạch phức tạp và dễ gây nhiễu nếu không xử lý kỹ lưỡng.
* Khả năng mở rộng hệ thống còn hạn chế: Ứng dụng hiện tại phù hợp với dữ liệu vừa và nhỏ, chưa tích hợp khả năng xử lý phân tán hoặc tối ưu hóa cho tập dữ liệu cực lớn (Big Data).
* Giao diện người dùng đơn giản: Mặc dù sử dụng Streamlit để tạo giao diện tương tác, hệ thống hiện vẫn còn hạn chế về tính năng nâng cao như lọc động tập luật, gợi ý tham số phù hợp hoặc báo cáo thống kê tự động.
* Chưa bao gồm nhiều chiều phân tích: Đề tài chỉ tập trung vào phân tích dựa trên tần suất xuất hiện của sản phẩm, chưa xét đến các yếu tố về thời gian, hành vi tiêu dùng theo nhóm khách hàng hoặc sự biến thiên theo mùa vụ.

## Hướng phát triển

Để phát huy hơn nữa tiềm năng ứng dụng và mở rộng phạm vi nghiên cứu, đề tài có thể được phát triển theo các hướng sau:

* Tích hợp thêm các thuật toán khai phá nâng cao: Bên cạnh Apriori và FP-Growth, có thể mở rộng nghiên cứu sang các thuật toán khác như ECLAT, H-Mine, hoặc các phiên bản cải tiến để nâng cao hiệu quả khai thác trong các tình huống dữ liệu đặc thù.
* Phân tích luật kết hợp theo chuỗi thời gian: Kết hợp khai phá luật với phân tích thứ tự xuất hiện hoặc các mô hình tuần tự để phát hiện các mẫu hành vi có tính chu kỳ, từ đó phục vụ tốt hơn cho các chiến dịch tiếp thị theo mùa vụ.
* Tối ưu hóa hiệu năng với công cụ phân tán: Ứng dụng các nền tảng xử lý dữ liệu lớn như Apache Spark để cải thiện hiệu năng khai thác và khả năng mở rộng đối với các tập dữ liệu thương mại điện tử quy mô lớn.
* Nâng cấp ứng dụng và triển khai cloud: Hoàn thiện hệ thống dưới dạng dịch vụ web và triển khai lên các nền tảng như Streamlit Cloud, Heroku hoặc Vercel, cho phép người dùng trải nghiệm và khai thác tri thức dữ liệu trực tuyến.
* Kết hợp với hệ thống gợi ý sản phẩm: Ứng dụng trực tiếp các luật kết hợp vào mô hình recommendation để cung cấp gợi ý sản phẩm theo thời gian thực, cá nhân hóa trải nghiệm người dùng và tăng hiệu quả kinh doanh.

**TÀI LIỆU THAM KHẢO**

1. **Bộ dữ liệu Online Retail (UCI Machine Learning Repository):**

* Chen, D. (2015). Online Retail [Dataset]. UCI Machine Learning Repository.
* Link: https://archive.ics.uci.edu/dataset/352/online+retail
* DOI: https://doi.org/10.24432/C5BW33

1. **Thuật toán Apriori:**

* Mohammed Al-Maolegi, Bassam Arkok. "An Improved Apriori Algorithm for Association Rules." arXiv:1403.3948 [cs.DB], 2014.
* Link: https://arxiv.org/abs/1403.3948
* Mythili, M., Shanavas, A.M. "Performance Evaluation of Apriori and FP-Growth Algorithms." International Journal of Computer Applications, 2013.
* Link: https://www.semanticscholar.org/paper/Performance-Evaluation-of-Apriori-and-FP-Growth-Mythili-Shanavas/c41b38878ec2b43149c5b9fce471d5bce8410353

1. **Thuật toán FP-Growth:**

* Haifeng Lu et al. "The Application of FP-Growth Algorithm Based on Distributed Intelligence in Wisdom Medical Treatment." International Journal of Pattern Recognition and Artificial Intelligence, 2017.
* Link: https://www.worldscientific.com/doi/10.1142/S0218001417590054
* Omar Santos. "Association Rules: Apriori and FP-Growth."
* Link: https://hackertraining.org/ai\_research/ML\_Fundamentals/ai\_generated/data/Association\_Rules\_(Apriori,\_FP-Growth)/[6]

1. **Tài liệu thư viện Streamlit:**

* Streamlit Documentation.
* Link: https://docs.streamlit.io

1. **Tài liệu thư viện Pandas:**

* imarranz. "Essential Guide to Pandas." GitHub Repository.
* Link: https://github.com/imarranz/essential-guide-to-pandas

1. **Các nguồn khác:**

* UCI Machine Learning Repository - License: Creative Commons Attribution 4.0 International (CC BY 4.0)
* Các hướng dẫn chính thức và ví dụ từ tài liệu của Streamlit, Pandas.

**PHỤ LỤC**

**Phụ lục A: Hướng dẫn sử dụng chương trình**

1. Cài đặt môi trường

* Đảm bảo đã cài đặt Python 3.8 trở lên.
* Tạo môi trường ảo (khuyến nghị):

|  |
| --- |
| python -m venv venv  source venv/bin/activate # Trên Linux/Mac  venv\Scripts\activate # Trên Windows |

* Cài đặt các thư viện cần thiết bằng file requirements.txt:

|  |
| --- |
| streamlit  pandas  numpy  graphviz  psutil  openpyxl |

1. Cách chạy ứng dụng

Đầu tiên muốn chạy được dự án này, ta phải mở terminal và trỏ đến vị trí của thư mục gốc bằng lệnh cd.

Chạy ứng dụng FP-Growth bằng lệnh:

|  |
| --- |
| streamlit run main\_fp\_growth\_visualizer.py |

Chạy ứng dụng Apriori bằng lệnh:

|  |
| --- |
| streamlit run main\_apriori\_visualizer.py |

Ứng dụng sẽ tự động mở trên trình duyệt tại địa chỉ: http://localhost:8501 (hoặc cổng tuỳ cấu hình).

**Mô tả các bước sử dụng giao diện**

Bước 1: Tải dữ liệu (file Excel/CSV) lên hoặc chọn dữ liệu mẫu.

Bước 2: Cấu hình các trường dữ liệu (mã giao dịch, tên sản phẩm, v.v.).

Bước 3: (Tuỳ chọn) Lọc dữ liệu theo quốc gia hoặc mã khách hàng.

Bước 4: Chọn thuật toán (Apriori hoặc FP-Growth) và điều chỉnh các tham số (min\_support, min\_confidence).

Bước 5: Nhấn nút thực thi. Kết quả sẽ được hiển thị gồm:

* Thông tin dữ liệu sau xử lý
* Thời gian chạy, mức sử dụng bộ nhớ
* Các bước trung gian của thuật toán
* Tập mục phổ biến, luật kết hợp
* Biểu đồ trực quan hóa mạng lưới luật kết hợp (nếu có)

**Phụ lục B: Một vài đoạn code quan trọng**

1. Đoạn code sinh tập ứng viên trong Apriori (apriori\_logic.py)

|  |
| --- |
| def generate\_candidates(prev\_freq\_itemsets, k):  candidates = set()  prev\_freq\_list = list(prev\_freq\_itemsets)  for i in range(len(prev\_freq\_list)):  for j in range(i+1, len(prev\_freq\_list)):  l1 = list(prev\_freq\_list[i])  l2 = list(prev\_freq\_list[j])  l1.sort()  l2.sort()  if l1[:k-2] == l2[:k-2]:  candidate = tuple(sorted(set(l1) | set(l2)))  candidates.add(candidate)  return candidates |

**Chức năng:** Đoạn code này dùng để sinh tập ứng viên bậc k (gồm k phần tử) từ các tập phổ biến bậc k-1 trong thuật toán Apriori.

**Cách hoạt động:**

* Hàm nhận vào:
* prev\_freq\_itemsets: danh sách các tập phổ biến ở bước trước (k-1).
* k: độ dài của tập ứng viên cần tạo.

**Ý tưởng:**

* Ghép 2 tập phổ biến k-1 nếu chúng giống nhau ở k-2 phần tử đầu (sau khi sắp xếp).
* Sau đó gộp chúng thành tập hợp mới có k phần tử → tập ứng viên.

1. Đoạn code xây dựng FP-Tree (fp\_growth\_logic.py)

|  |
| --- |
| class FPTreeNode:  def \_\_init\_\_(self, item, count, parent):  self.item = item  self.count = count  self.parent = parent  self.children = {}  self.link = None  def insert\_tree(items, node, header\_table):  if items:  first = items[0]  if first in node.children:  node.children[first].count += 1  else:  child = FPTreeNode(first, 1, node)  node.children[first] = child  # Cập nhật header\_table  if header\_table[first][1] is None:  header\_table[first][1] = child  else:  current = header\_table[first][1]  while current.link is not None:  current = current.link  current.link = child  insert\_tree(items[1:], node.children[first], header\_table) |

**Chức năng:** Hai đoạn code này kết hợp để xây dựng cây FP-Tree – một cấu trúc dữ liệu dùng để lưu trữ các giao dịch theo cách nén, phục vụ cho thuật toán FP-Growth.

**Lớp FPTreeNode:**

* Đại diện cho một nút trong FP-Tree.
* Mỗi nút chứa:
* item: tên mặt hàng.
* count: số lần mặt hàng này xuất hiện trên nhánh.
* parent: nút cha.
* children: các nút con (dưới dạng dictionary).
* link: con trỏ kết nối đến nút tiếp theo có cùng item trong cây (phục vụ truy vết nhanh từ header table).

**Hàm insert\_tree(items, node, header\_table)**

* **Chức năng:** Chèn một giao dịch (danh sách mặt hàng đã được sắp xếp theo tần suất) vào cây FP-Tree.
* **Cách hoạt động:**
* Lấy phần tử đầu tiên first trong danh sách items.
* Nếu đã tồn tại nhánh con tương ứng → tăng đếm (count).
* Nếu chưa có → tạo mới nút con và cập nhật header table để duyệt nhanh theo link.
* Đệ quy chèn tiếp phần còn lại items[1:].
* **Vai trò trong FP-Growth:**
* Là bước đầu tiên và quan trọng nhất để nén dữ liệu giao dịch thành FP-Tree.
* Sau khi có FP-Tree, thuật toán sẽ duyệt cây từ dưới lên để sinh các Conditional Pattern Base và Conditional FP-Tree, từ đó khai thác luật kết hợp.