BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ THÀNH PHỐ HỒ CHÍ MINH (UEH)

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

ỨNG DỤNG PHƯƠNG PHÁP PHÂN LỚP DỰA TRÊN MẪU PHỔ BIẾN ĐỂ DỰ ĐOÁN KHẢ NĂNG HỦY ĐẶT PHÒNG CỦA KHÁCH HÀNG

Học phần: Máy Học

Nhóm Sinh Viên:

1. Võ Minh Nguyên
2. Nguyễn Phúc Bảo Nhân
3. Phan Đình Nhân
4. Nguyễn Quang Nhật
5. Nguyễn Thị Ngọc Nhi

Chuyên Ngành: Khoa Học Dữ Liệu

Khóa: K47

Giảng Viên: TS. Nguyễn An Tế

Thành phố Hồ Chí Minh , tháng 12 năm 2023

LỜI MỞ ĐẦU

Trong thời đại hội nhập toàn cầu, các trang web đặt phòng và mạng xã hội đã trở thành những cầu nối không thể thiếu, kết nối khách du lịch và nhà hàng - khách sạn trên phạm vi toàn cầu. Những nền tảng này không chỉ là nơi cho các doanh nghiệp và cá nhân trưng bày dịch vụ của mình, mà còn là công cụ đắc lực để khai thác trí tuệ nhân tạo (AI) trong việc tạo ra những đề xuất dịch vụ cá nhân hóa dựa trên sở thích và lịch trình của khách hàng.

Giờ đây, các thuật toán và phương pháp máy học được triển khai một cách tinh vi để phân tích và phân loại từng tệp khách hàng cụ thể dựa trên hành vi của họ, từ đó đưa ra những đề xuất dịch vụ tương tự hoặc liên quan mà khách hàng tiềm năng có thể quan tâm. Điều này không chỉ nâng cao trải nghiệm đặt phòng trực tuyến của khách hàng, mà còn giúp các doanh nghiệp tối ưu hóa chiến lược kinh doanh của mình.

Bên cạnh đó, việc nắm bắt các yêu cầu về đặt phòng hoặc dịch vụ đi kèm của khách hàng là rất quan trọng. Điều này giúp các doanh nghiệp nhà hàng - khách sạn hiểu rõ hơn về mong muốn và hành vi của khách hàng, từ đó xác định hoặc phân loại khách hàng tiềm năng. Điều này không chỉ giúp tăng doanh số đặt phòng, mà còn giúp các doanh nghiệp xây dựng một mối quan hệ lâu dài và bền vững với khách hàng của mình.

Từ khóa: Nhà hàng - Khách sạn, Yêu cầu đặt phòng, Dịch vụ đi kèm, Phân loại khách hàng, Phân lớp dựa trên mẫu phổ biến (Classification Using Frequent Patterns).

MỤC LỤC

CHƯƠNG I. TỔNG QUAN ĐỀ TÀI

1. Sơ lược đề tài

Trong bối cảnh số hóa toàn cầu, ngành dịch vụ nhà hàng và khách sạn đang phải đối mặt với nhiều thách thức trong việc tiếp nhận công nghệ mới, thậm chí còn nhiều hơn so với các ngành khác trong cùng lĩnh vực. Việc đánh giá chất lượng dịch vụ yêu cầu sự am hiểu về các quy tắc phức tạp và sự khác biệt cho mỗi loại dịch vụ. Tuy nhiên, trong một môi trường thiếu thông tin về khách hàng (NON - KYC), cùng với hệ phức tạp, việc áp dụng quy trình đánh giá chất lượng truyền thống có thể gây ra xung đột với mục tiêu lợi nhuận kinh doanh.

Trong ngành này, việc hủy phòng của khách hàng tiềm ẩn rủi ro cho doanh nghiệp. Hủy phòng đột ngột có thể gây ra thiệt hại về doanh thu và lợi nhuận, đặc biệt khi doanh nghiệp đã chuẩn bị và phục vụ dựa trên đặt phòng đó. Hơn nữa, việc hủy phòng thường xuyên từ phía khách hàng có thể làm tổn hại đến uy tín và danh tiếng của doanh nghiệp, làm mất lòng tin của khách hàng và ảnh hưởng đến việc thu hút khách hàng mới trong tương lai.

Vì vậy, việc xác định và quản lý khả năng hủy phòng của khách hàng là rất quan trọng để giảm thiểu rủi ro và tối ưu hóa lợi nhuận cho doanh nghiệp. Thông qua bộ dữ liệu gốc, nhóm nghiên cứu có thể xác định khả năng hủy phòng của khách hàng dựa trên yêu cầu đặt phòng và dịch vụ đi kèm. Điều này giúp doanh nghiệp tránh được rủi ro về lợi nhuận và đóng góp vào việc đánh giá tiềm năng và chất lượng dịch vụ nhà hàng - khách sạn, từ đó mang lại hiểu biết sâu hơn về tương tác giữa các thuộc tính của khách hàng đặt phòng và ảnh hưởng của chúng đến chi phí dịch vụ của doanh nghiệp.

2. Mục tiêu nghiên cứu

Mục tiêu chính của nghiên cứu này là tìm hiểu và phân tích các yếu tố ảnh hưởng đến khả năng hủy phòng của khách hàng trong ngành nhà hàng - khách sạn. Điều này bao gồm việc xác định các thuộc tính của khách hàng và yêu cầu đặt phòng có liên quan đến việc hủy phòng.

Nghiên cứu cũng nhằm mục đích tìm hiểu về mối quan hệ giữa các yếu tố này và hoạt động dịch vụ hiện tại của doanh nghiệp. Điều này sẽ giúp doanh nghiệp tối ưu hóa quy trình đánh giá chất lượng dịch vụ, giảm thiểu rủi ro về lợi nhuận và tăng cường mối quan hệ với khách hàng.

Ngoài ra, thông qua việc phân tích và hiểu rõ hơn về các yếu tố này, doanh nghiệp có thể đưa ra các chiến lược và quyết định kinh doanh hiệu quả hơn, từ đó nâng cao chất lượng dịch vụ và tăng cường sự hài lòng của khách hàng. Điều này không chỉ

giúp tăng cường mối quan hệ với khách hàng hiện tại mà còn thu hút khách hàng mới, góp phần vào sự phát triển bền vững của doanh nghiệp.

3. Phương pháp nghiên cứu

Nhóm sử dụng nhiều phương pháp khác nhau linh hoạt theo nội dung nghiên cứu. Trong phương pháp phân lớp dựa trên mẫu phổ biến, nhóm chia bài toán lớn thành hai bài toán nhỏ: tìm mẫu phổ biến và học có giám sát:

- **Chương II**: Tập trung vào việc xây dựng khái niệm về thuật toán tìm mẫu phổ biến thông qua luật kết hợp như Apriori và FP-Growth. Ngoài ra, chương này cũng giới thiệu về mô hình học máy SVM được sử dụng để phân lớp trong bài toán.

- **Chương III**: Tạo ra một ví dụ minh họa cụ thể để xây dựng các bước phân lớp khách hàng dựa trên mẫu phổ biến.

  Theo như một số nghiên cứu mà nhóm tìm hiểu được trong quá trình nghiên cứu thuật toán [3], nhóm nhận ra rằng các đặc trưng có độ hỗ trợ thấp hoặc ít xuất hiện trong bộ dữ liệu không đóng góp nhiều vào việc phân lớp. Thay vào đó, việc kết hợp các đặc trưng lại với nhau có thể mang lại ngữ nghĩa tốt hơn so với việc sử dụng các đặc trưng đơn lẻ. Tuy nhiên, việc này tạo ra một lượng lớn các trường hợp, cụ thể là cấp số nhân của số lượng các đặc trưng đơn lẻ trong bộ dữ liệu.

  Để giải quyết vấn đề này, trong bài báo nghiên cứu của Hong Cheng và các cộng sự, tác giả có đề đến việc sử dụng mẫu phổ biến để lọc lại những đặc trưng và những kết hợp tốt nhất cho việc phân lớp. Quá trình này bao gồm ba bước chính:

  1. **Khai thác mẫu phổ biến**: Đây là bước đầu tiên, nhóm phải tìm kiếm các mẫu phổ biến trong bộ dữ liệu.
  2. **Chọn lọc các đặc trưng**: Sử dụng các mẫu phổ biến đã tìm được, nhóm sẽ thực hiện lọc lại các đặc trưng có trong bộ dữ liệu bằng mẫu phổ biến vừa tìm được.
  3. **Xây dựng mô hình máy học**: Cuối cùng, nhóm mới thực hiện xây dựng mô hình máy học dựa trên bộ dữ liệu đã được chọn lọc.

  Như vậy, thông qua việc áp dụng mẫu phổ biến, nhóm có thể tối ưu hóa quá trình phân lớp và cải thiện hiệu suất của mô hình máy học.

- **Chương IV**: Thực hiện áp dụng bài toán vào một bộ dữ liệu thực tế và đánh giá độ chính xác của việc dự đoán các tình huống có thể xảy ra với khách

hàng, như việc hủy phòng hoặc không hủy phòng. Kết quả này sau đó được so sánh với thực tế để đánh giá hiệu quả của phương pháp.

4. Tài nguyên sử dụng

- Ngôn ngữ lập trình: Python
- Bộ dữ liệu Hotel Reservations Dataset từ Kaggle: Hotel Reservations Dataset

CHƯƠNG II. KHÁI NIỆM TỔNG QUÁT

1. Lựa chọn mẫu phổ biến

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

Hai thuật toán được sử dụng rộng rãi trong phương pháp chọn mẫu phổ biến là: Apriori và FP-Growth

- **Apriori:**

  Được xây dựng bởi Agrawal và các cộng sự. Thuật toán Apriori đã tồn tại gần như cùng thời gian với khái niệm về tập hợp phần tử thường xuyên và rất phổ biến. Vấn đề chính của thuật toán Apriori là việc thuật toán này phải quét cơ sở dữ liệu ở mọi cấp độ. Nên với cơ sở dữ liệu lớn và phức tạp thì thuật toán này chưa tối ưu do tốn kém về bộ nhớ và thời gian chạy lâu. Để khắc phục những điều đó, những thuật toán được phát triển dựa trên Apriori đã được ra đời là ECLAT và FP-Growth.

- **FP-Growth:**

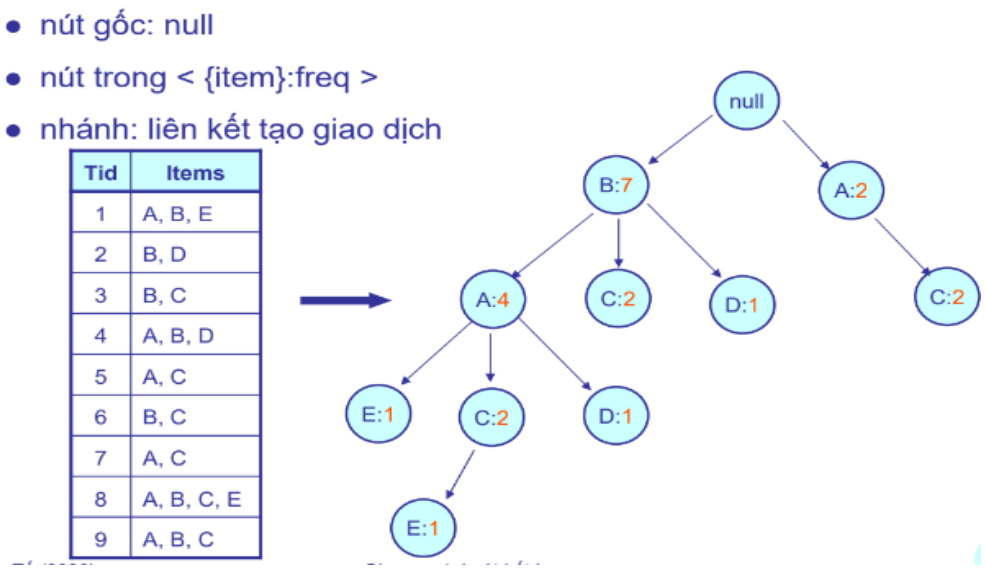
  Để khắc phục những điểm yếu của thuật toán Apriori, thuật toán FP-Growth đã áp dụng cấu trúc cây FP điều kiện và đệ quy nhằm giảm việc quét dữ liệu nhiều lần cũng như là tránh sinh ra những tập con không cần thiết.

Khi thực hiện phân lớp dựa trên mẫu phổ biến, sự lựa chọn giữa Apriori và FP-Growth có ý nghĩa khá quan trọng. Tuy nhiên, trong hầu hết các trường hợp, FP-Growth là phương pháp được ưu tiên hơn do nhiều lợi ích mà nó mang lại.

2. FP-Growth

a. Cấu trúc dữ liệu

Thuật toán FP-Growth, được Han và đồng nghiệp giới thiệu vào năm 2000, áp dụng cấu trúc dữ liệu dọc và ngang để lưu trữ cơ sở dữ liệu trong bộ nhớ chính. Khác với việc lưu trữ bìa cho từng mục, FP-Growth lưu trữ các giao dịch thực tế từ cơ sở dữ liệu theo cấu trúc cây điều kiện. Mỗi mục trong cây có một danh sách liên kết, đi qua tất cả các giao dịch chứa mục đó. Cấu trúc dữ liệu này được biểu diễn dưới dạng FP-Tree. Cụ thể như sau *(Tham khảo - TS. Nguyễn An Tế)*:



b. Các bước thực hiện

FP-Growth là thuật toán với phương thức tìm kiếm theo chiều sâu được mô tả khái quát các bước như sau [2]:

- **Bước 1: Xây dựng F-list từ L_1.**
- **Bước 2: Xây dựng FP-Tree và Item Header Table.**
- **Bước 3: Duyệt "ngược" các item $I_k \in$ F-list để tạo các tập phổ biến**:
  - Tạo cơ sở mẫu điều kiện CPB từ prefix của các paths đến I_k.
  - Dùng CPB như CSDL để tạo FP-Tree điều kiện.
  - Tạo Item Header Table chỉ bao gồm các items phổ biến (cộng dồn freq(item) trong các paths).
  - Phát sinh tập phổ biến từ MỖI path trong FP-Tree điều kiện.

3. Các khái niệm quan trọng trong mẫu phổ biến

a. Item

Trong phân tích dữ liệu, một item đề cập đến một đơn vị dữ liệu đơn lẻ, một thuộc tính cụ thể hoặc một đối tượng trong tập dữ liệu.

Ví dụ:
Gọi I = {I1, I2, I3, ...} là tập hợp các sản phẩm (Items) được bán trong cửa hàng.
I = {Sữa, Bánh quy, Nước ngọt, Cam, Chanh}

b. Itemset

Một itemset là một tập hợp các item được tổ chức lại cùng nhau. Điều này có thể là một tập hợp con của các item trong dữ liệu hoặc một tập hợp các item hoàn chỉnh. Itemset có thể có kích thước từ 1 (item đơn lẻ) đến n (tập hợp tất cả các item có thể).

Ví dụ:
Gọi X là tập hợp mục tiêu (Itemset), nhằm xác định các sản phẩm xuất hiện cùng nhau trong một tập dữ liệu, với $X \subseteq I$.
{Sữa, Bánh quy}

c. K - Itemset

Kích thước tập (K - Itemset) được xác định dựa trên K số lượng các sản phẩm khác nhau có mặt trong cùng một tập hợp mục tiêu (với $K \leq n$).

Ví dụ:
{Sữa, Bánh quy} $\rightarrow$ *Kích thước tập (Size) = 2*

d. Support

Support (độ hỗ trợ) được sử dụng để đo lường tần suất xuất hiện của một itemset trong dữ liệu. Support được tính bằng tỷ lệ giữa số lần xuất hiện của itemset và tổng số mẫu trong dữ liệu.

Ví dụ:
{Sữa, Bánh quy} cùng xuất hiện 3 lần trong tổng số 5 hóa đơn:
Mẫu {Sữa, Bánh quy} với support = 3/5 = 0.6 (60%)

e. Ngưỡng hỗ trợ tối thiểu (minSup)

MinSup là ngưỡng hỗ trợ tối thiểu được đặt ra từ ban đầu để xác định các mẫu phổ biến. Với chỉ số này, cần phải cân nhắc trước khi lựa chọn, vì khi đặt ngưỡng minSup quá cao hoặc quá thấp trong thuật toán khai phá luật kết hợp có thể xảy ra những hiện tượng sau:

- ***Ngưỡng minSup quá cao:***

  Khi minSup được đặt quá cao, số lượng itemsets thỏa mãn ngưỡng này sẽ giảm đáng kể. Điều này có thể dẫn đến việc bỏ qua nhiều mẫu phổ biến quan trọng hoặc hiển thị một số luật kết hợp không có ý nghĩa. Tác động của việc đặt minSup quá cao là mất mát thông tin quan trọng và giảm khả năng tìm ra các mẫu dữ liệu ý nghĩa.

- ***Ngưỡng minSup quá thấp***

Khi minSup được đặt quá thấp, số lượng itemsets thỏa mãn ngưỡng này sẽ tăng lên đáng kể. Điều này có thể dẫn đến một số vấn đề như:

- **Sự gia tăng về số lượng luật kết hợp**: Một minSup quá thấp có thể tạo ra quá nhiều luật kết hợp, trong đó có nhiều luật không có ý nghĩa hoặc không thực tế. Điều này làm tăng khối lượng dữ liệu được trả về và khó khăn khi phân tích và hiểu các luật quan trọng.
- **Overfitting**: Trong trường hợp MinSup quá thấp, mô hình có thể học quá nhiều quy tắc hoặc mẫu hiếm gặp trong dữ liệu huấn luyện, bao gồm cả các quy tắc không có ý nghĩa hoặc nhiễu. Điều này có thể dẫn đến việc mô hình bị khớp quá mức với dữ liệu huấn luyện, nhưng không thể áp dụng tốt cho dữ liệu mới.

➔ Vì vậy, việc lựa chọn giá trị MinSup phù hợp là cần thiết để đảm bảo rằng các mẫu phổ biến tìm thấy có ý nghĩa và hữu ích trong quá trình khai phá dữ liệu, đồng thời tránh những tác động tiêu cực như Overfitting và tăng hao phí tính toán.

f. Frequent Itemset

Frequent itemset (tập hợp mục phổ biến) là một tập hợp các item mà cùng xuất hiện với tần suất đáng kể trong dữ liệu. Frequent itemset thường được xác định bằng cách đếm số lần xuất hiện của từng tập hợp item trong dữ liệu và so sánh với ngưỡng tần suất.

Ví dụ: Có tổng cộng 5 hóa đơn, với ngưỡng tần suất tối thiểu là 50%.
Mẫu {Sữa, Bánh quy} có support 0.6 nên được xem là frequent itemset (>50%).

4. Phương pháp phân lớp (SVM)

Phương pháp phân lớp SVM (Support Vector Machine) là một thuật toán học máy được sử dụng rộng rãi để phân loại các điểm dữ liệu vào các lớp khác nhau trong các bài toán phân loại. SVM là một phương pháp dựa trên việc tìm một siêu phẳng (hyperplane) trong không gian đặc trưng sao cho tối đa hóa khoảng cách giữa các điểm dữ liệu gần nhất của các lớp.

Ý tưởng cơ bản của SVM là chọn một siêu phẳng sao cho khoảng cách từ siêu phẳng đến điểm gần nhất của các lớp là lớn nhất. Điểm gần nhất của mỗi lớp được gọi là vector hỗ trợ (support vector), vì chúng "hỗ trợ" định nghĩa siêu phẳng phân lớp. Quá trình tìm siêu phẳng tối ưu này có thể được thực hiện thông qua một bài toán tối ưu hóa.

Ứng dụng SVM trong bài toán phân lớp dựa trên mẫu phổ biến:

Trong bài toán phân loại, việc ổn định mô hình và tránh overfitting là vô cùng quan trọng. SVM (Support Vector Machine) là một phương pháp phân lớp mà có thể giải quyết vấn đề này một cách hiệu quả. SVM có khả năng chịu được nhiễu và điều chỉnh overfitting thông qua các thông số như C (tham số điều chỉnh độ lỗi cho phân lớp sai) và các hàm kernel.

Ngoài ra, SVM tập trung vào tìm siêu phẳng tối ưu để phân chia các lớp dữ liệu, đặc biệt chú trọng đến các mẫu gần biên giới. Điều này làm cho SVM trở nên hữu ích trong trường hợp các mẫu phổ biến, tức là các mẫu dữ liệu quan trọng và thường xuyên xuất hiện. Các mẫu phổ biến thường nằm gần biên giới và có ảnh hưởng lớn đến quá trình phân loại. Bằng cách tập trung vào những mẫu này, SVM có khả năng định rõ siêu phẳng tối ưu và đưa ra quyết định phân lớp chính xác.

CHƯƠNG III. CÁC BƯỚC XÂY DỰNG THUẬT TOÁN

1. Chuẩn bị dữ liệu

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

Khách hàng	Sản phẩm	Loại Khách hàng
KH1	Áo phông, Giày thể thao, Quần Jean	Mục tiêu
KH2	Áo phông, Mũ lưỡi trai, Quần Kaki	Không là mục tiêu
KH3	Áo khoác, Áo phông, Quần Kaki, Túi xách	Mục tiêu
KH4	Mũ lưỡi trai, Quần Jean, Túi xách	Mục tiêu
KH5	Áo phông, Mũ lưỡi trai, Quần Jean, Quần Kaki, Túi xách	Không là mục tiêu

2. Khai thác mẫu phổ biến

Nhóm sẽ sử dụng các thuật toán khai thác mẫu phổ biến, như: Apriori, FP-Growth, ECLAT, …. Nhờ đó tìm ra các mẫu phổ biến từ dữ liệu đã chuẩn bị ban đầu.

Ví dụ: Thông qua thuật toán FP-Growth, nhóm sẽ tìm ra các mẫu phổ biến từ bảng dữ liệu minh họa. Ở bước này, nhóm chỉ lấy cột **Sản phẩm** để thực hiện tính toán.

- Xây dựng F-List: Duyệt cơ sở dữ liệu ban đầu để xác định L_1, sau đó nhóm sẽ sắp xếp lại L_1 theo thứ tự giảm dần của tần suất:

Sản phẩm	L1
Áo khoác	1
Áo phông	4
Giày thể thao	1
Mũ lưỡi trai	3

→

Sản phẩm	F-List
Áo phông	4
Quần Jean	3
Quần Kaki	3
Mũ lưỡi trai	3

Quần Jean	3
Quần Kaki	3
Túi xách	3

Túi xách	3
Áo khoác	1
Giày thể thao	1

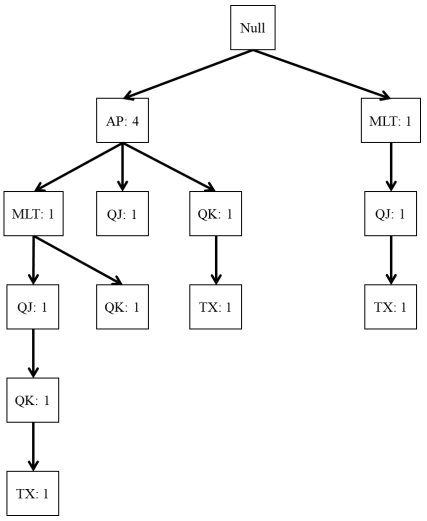
- Lựa chọn vật phẩm dựa trên ngưỡng tần suất xuất hiện tối thiểu (minSup) là 40%: Có 5 giao dịch, nên để thỏa mãn minSup = 0.4, nhóm sẽ xem xét giữ lại các mẫu xuất hiện ít nhất 2 lần (0.4 × 5). Sau đó, thực hiện loại bỏ các vật phẩm không đủ điều kiện ra khỏi bảng ban đầu:

Sản phẩm	**F-List**
Áo phông	4
Mũ lưỡi trai	3
Quần Jean	3
Quần Kaki	3
Túi xách	3

Loại: Áo khoác, Giày thể thao (Do có 1 lần xuất hiện)

Khách hàng	**Sản phẩm**
KH1	Áo phông, Quần Jean
KH2	Áo phông, Mũ lưỡi trai, Quần Kaki
KH3	Áo phông, Quần Kaki, Túi xách
KH4	Mũ lưỡi trai, Quần Jean, Túi xách
KH5	Áo phông, Mũ lưỡi trai, Quần Jean, Quần Kaki, Túi xách.

- Từ bảng các vật phẩm mới được xác định lại, nhóm thực hiện xây dựng FP-Tree. Với quy ước các vật phẩm như sau: Áo phông - AP, Mũ lưỡi trai - MLT, Quần Jean - QJ, Quần Kaki - QK, Túi xách - TX:



- Sau khi hoàn thành việc xây dựng cây, nhóm tiến hành tạo ra cơ sở mẫu điều kiện (Conditional Pattern Base - CPB). Mục đích là để khởi tạo FP-Tree điều kiện, dựa trên ngưỡng xuất hiện tối thiểu (minSup) trước đó là 2. Tiếp theo, tìm ra các tập phổ biến từ mỗi nhánh (Path) của cây điều kiện vừa được khởi tạo. Với thứ tự duyệt lần lần lượt là: TX → QK → QJ → MLT → AP:

Vật phẩm	**CPB**	**FP-Tree điều kiện**	**Mẫu phổ biến**
TX	AP-MLT-QJ-QK: 1 AP-QK: 1 MLT-QJ: 1	AP: 2 MLT: 2 QJ: 2 QK: 2	AP-TX MTL-TX QJ-TX QK-TX
			AP-MLT-TX AP-QJ-TX AP-QK-TX MTL-QJ-TX MLT-QK-TX QJ-QK-TK
			AP-MLT-QJ-TX AP-MLT-QK-TX AP-QJ-QK-TX MLT-QJ-QK-TX
			AP-MLT-QJ-QK-TX
MLT	AP: 1	Ø	Ø
QK	AP-MLT-QJ: 1 AP-MLT: 1 AP: 1	AP: 3 MLT: 2	AP-QK MLT-QK AP-MTL-QK
QJ	AP-MLT: 1 AP: 1 MLT: 1	AP: 2 MLT: 2	AP-QJ MLT-QJ AP-MLT-QJ
AP	Ø	Ø	Ø

- Từ kết quả trên, nhóm thu được lần lượt các mẫu phổ biến như sau:

Mẫu 1-Itemset	**Mẫu 2-Itemset**	**Mẫu 3-Itemset**

{Áo phông} {Mũ lưỡi trai} {Quần Jean} {Quần Kaki} {Túi xách}	{Áo phông, Túi xách} {Mũ lưỡi trai, Túi xách} {Quần Jean, Túi xách} {Quần Kaki, Túi xách} {Áo phông, Quần Kaki} {Mũ lưỡi trai, Quần Kaki} {Áo phông, Quần Jean} (Mũ lưỡi trai, Quần Jean}	{Áo phông, Mũ lưỡi trai, Túi xách} {Áo phông, Quần Jean, Túi xách} {Áo phông, Quần Kaki, Túi xách} {Mũ lưỡi trai, Quần Jean, Túi xách} {Mũ lưỡi trai, Quần Kaki, Túi xách} {Quần Jean,Quần Kaki, Túi xách} {Áo phông, Mũ lưỡi trai, Quần Kaki} {Áo phông, Mũ lưỡi trai, Quần Jean}
Mẫu 4-Itemset		**Mẫu 5-Itemset**
{Áo phông, Mũ lưỡi trai, Quần Jean, Túi xách} {Áo phông, Mũ lưỡi trai, Quần Kaki, Túi xách} {Áo phông, Quần Jean, Quần Kaki, Túi xách} {Mũ lưỡi trai, Quần Jean, Quần Kaki, Túi xách}		{Áo phông, Mũ lưỡi trai, Quần Jean, Quần Kaki, Túi xách}

3. Chuẩn bị dữ liệu

Sau khi đã chọn lọc mẫu phổ biến quan trọng, nhóm cần chuyển đổi những mẫu này thành dạng đặc trưng phù hợp để có thể đưa vào mô hình học máy ở bước 4. Trong trường hợp này, nhóm sẽ sử dụng One-hot Encoding để chuyển đổi mẫu phổ biến thành đặc trưng.

Ví dụ: Tiếp tục từ ví dụ trên.

Như thông tin được đề cập ở **Chương 1.3: Phương pháp nghiên cứu**, việc kết hợp các đặc trưng lại với nhau có thể mang lại ngữ nghĩa tốt hơn so với việc sử dụng các đặc trưng đơn lẻ. Tuy nhiên, việc này tạo ra một lượng lớn các trường hợp, cụ thể là cấp số nhân của số lượng các đặc trưng đơn lẻ trong bộ dữ liệu.

Chính vì vậy, nhóm quyết định chỉ lựa chọn mẫu phổ biến lớn nhất {Áo phông, Mũ lưỡi trai, Quần Jean, Quần Kaki, Túi xách} để thêm vào bộ dữ liệu ban đầu. Điều này giúp tập trung vào những thông tin quan trọng nhất, đồng thời loại bỏ những thông tin không cần thiết.

Sau khi lựa chọn mẫu phổ biến, nhóm tiếp tục biểu diễn lại các vật phẩm đã được chọn lọc. Bao gồm: bảng dữ liệu ban đầu, bổ sung thêm cột mới là kết hợp của itemset lớn nhất. Giá trị của cột mới được xác định bằng phép toán logic XOR.

Cuối cùng, để phục vụ cho việc phân lớp ở bước tiếp theo, nhóm sẽ sử dụng phương pháp One-hot Encoding để biểu diễn các vật phẩm đã được chọn lọc. Phương pháp này giúp nhóm chuyển đổi dữ liệu ở dạng định danh thành dạng nhị phân, giúp máy học dễ dàng hơn trong việc xử lý và phân lớp dữ liệu.

Cụ thể, dữ liệu sẽ được biểu diễn như sau:

STT	AP	MLT	QJ	QK	TX	AP×MLT×QJ×QK×TX
1	1	0	1	0	0	0
2	1	1	0	1	0	1
3	1	0	0	1	1	1
4	0	1	1	0	1	1
5	1	1	1	1	1	1

Cuối cùng, để có thể thực hiện phân lớp, nhóm thực hiện ghép lại cột **Loại khách hàng** mà nhóm đã tách ra từ ban đầu để thực hiện tìm mẫu phổ biến. Đây là bước quan trọng để nhóm có thể chuyển từ bài toán tìm mẫu phổ biến sang bài toán học có giám sát tiếp theo. Cụ thể như sau:

STT	AP	MLT	QJ	QK	TX	AP×MLT×QJ×QK×TX	Loại khách hàng
1	1	0	1	0	0	0	Mục tiêu
2	1	1	0	1	0	1	Không là mục tiêu
3	1	0	0	1	1	1	Mục tiêu
4	0	1	1	0	1	1	Mục tiêu
5	1	1	1	1	1	1	Không là mục tiêu

4. Xây dựng mô hình phân lớp

Ở bước này, nhóm sẽ tiến hành sử dụng các mẫu phổ biến đã được chọn lọc trước đó để xây dựng mô hình phân lớp. Điều này giúp nhóm có thể tập trung vào những thông tin quan trọng nhất, đồng thời loại bỏ những thông tin không cần thiết.

Để thực hiện việc huấn luyện dữ liệu, nhóm có thể áp dụng rất nhiều thuật toán học máy khác nhau. Các thuật toán này bao gồm: Naive Bayes, Decision Tree, Random Forest, SVM, và Neural Network, …. Mỗi thuật toán có những ưu điểm và nhược điểm riêng, và sẽ được chọn lựa dựa trên đặc điểm của bộ dữ liệu và yêu cầu của bài toán.

Mục tiêu cuối cùng của nhóm là xác định được những kết hợp sản phẩm nào có thể hỗ trợ việc xác định tệp khách hàng thông qua mô hình phân lớp. Việc này không chỉ giúp nhóm hiểu rõ hơn về hành vi mua sắm của khách hàng, mà còn hỗ trợ doanh nghiệp đưa ra được những chiến lược kinh doanh và marketing tối ưu.

5. Sử dụng mô hình

Sau cùng, mô hình phân lớp có thể được sử dụng để phân lớp dữ liệu mới chưa được gán nhãn. Các mẫu phổ biến trong dữ liệu mới sẽ được khai thác và sử dụng làm đặc trưng để dự đoán nhãn của dữ liệu.

Ví dụ: Nếu có một khách hàng mới và biết được hành vi mua hàng của khách hàng đó là {Áo phông, Quần jean, Mũ lưỡi trai}, chúng ta có thể sử dụng mô hình đã xây dựng để dự đoán nhãn cho khách hàng đó.

CHƯƠNG IV. TỔNG QUAN BỘ DỮ LIỆU

1. Sơ lược về bộ dữ liệu

Hotel Reservations Dataset, được đăng tải trên Kaggle bởi Ahsan Raza vào ngày 4 tháng 1 năm 2023, nhằm mục đích phân tích và dự báo khả năng hủy đặt phòng của khách hàng.

Bộ dữ liệu cung cấp thông tin chi tiết về các đơn đặt phòng tại các khách sạn trong khoảng thời gian từ tháng 7 năm 2017 đến tháng 12 năm 2018. Bao gồm 19 thuộc tính, với tổng cộng 36,275 quan sát, và không có giá trị thiếu. Các thuộc tính bao gồm ngày nhận phòng, ngày trả phòng, ngày tạo đặt phòng, cùng với nhiều thông tin khác. Mỗi quan sát đại diện cho một khách hàng đã đặt phòng tại khách sạn.

Bằng cách phân tích và hiểu rõ hơn về các thuộc tính này, nhóm nghiên cứu có thể nắm bắt được hành vi và xu hướng của khách hàng. Điều này giúp nhóm có thể dự đoán chính xác hơn về khả năng hủy đặt phòng thông qua phương pháp phân lớp dựa trên mẫu phổ biến mà nhóm đã trình bày trước đó.

2. Mô tả thuộc tính của bộ dữ liệu

TÊN THUỘC TÍNH	MÔ TẢ	KIỂU DỮ LIỆU
Booking_ID	Mã đặt phòng	Text
no_of_adults	Số người lớn	Numeric
no_of_children	Số trẻ em	Numeric
no_of_weekend_nights	Số đêm khách đặt vào cuối tuần (Thứ 7, Chủ Nhật)	Numeric
no_of_week_nights	Số đêm khách đặt trong tuần (Thứ 2 - Thứ 5)	Numeric
type_of_meal_plan	Loại bữa ăn khách hàng chọn	Categorical
required_car_parking_space	Nếu khách hàng yêu cầu có bãi đỗ xe thì là ‘1’; nếu không thì là ‘0’	Categorical
room_type_reserved	Loại phòng	Categorical
lead_time	Số ngày giữa ngày đặt phòng và ngày đến	Numeric

arrival_year	Năm đến	Numeric
arrival_month	Tháng đến	Numeric
arrival_date	Ngày đến	Numeric
market_segment_type	Hình thức tiếp cận của khách hàng	Categorical
repeated_guest	Nếu là khách hàng cũ thì là '1'; nếu là khách hàng mới thì là '0'	Categorical
no_of_previous _cancellations	Số lần khách hàng hủy đặt phòng trước đây	Numeric
no_of_previous _bookings_not_canceled	Số lần khách hàng không hủy đặt phòng trước đây	Numeric
avg_price_per_room	Giá phòng bình quân theo ngày	Numeric
no_of_special_requests	Tổng số yêu cầu đặc biệt khác (ở tầng cao, tầm nhìn đẹp)	Numeric
booking_status	Đơn đặt phòng bị hủy hoặc không	Categorical

3. Tiền xử lý dữ liệu

a. Thăm dò dữ liệu

Đầu tiên, nhóm sẽ tiến hành thăm dò dữ liệu (data exploration) để nắm được những thông tin cơ bản như số lượng thuộc tính (cột), bản ghi (dòng) được ghi nhận trong bộ dữ liệu. Qua đó, giúp nhóm có thể xác định được quy mô cũng như các thông tin quan trọng của bộ dữ liệu mà nhóm sẽ cần quan tâm đến.

```
Kích thước của bộ dữ liệu gồm 36275 dòng và 19 cột.
```

```
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Booking_ID                            36275 non-null  object
 1   no_of_adults                          36275 non-null  int64
 2   no_of_children                        36275 non-null  int64
 3   no_of_weekend_nights                  36275 non-null  int64
 4   no_of_week_nights                     36275 non-null  int64
 5   type_of_meal_plan                     36275 non-null  object
 6   required_car_parking_space            36275 non-null  int64
 7   room_type_reserved                    36275 non-null  object
 8   lead_time                             36275 non-null  int64
 9   arrival_year                          36275 non-null  int64
 10  arrival_month                         36275 non-null  int64
 11  arrival_date                          36275 non-null  int64
 12  market_segment_type                   36275 non-null  object
 13  repeated_guest                        36275 non-null  int64
 14  no_of_previous_cancellations          36275 non-null  int64
 15  no_of_previous_bookings_not_canceled  36275 non-null  int64
 16  avg_price_per_room                    36275 non-null  float64
 17  no_of_special_requests                36275 non-null  int64
 18  booking_status                        36275 non-null  object
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
```

Qua việc tìm hiểu sơ lược bộ dữ liệu, nhóm có thể thu được các thông tin cơ bản như sau: bộ dữ liệu gồm 19 cột và 36.275 dòng.

Ngoài ra, để có thể nắm bắt thêm các thông tin cụ thể hơn về bộ dữ liệu, nhóm sẽ tiến hành quan sát số giá trị duy nhất của mỗi thuộc tính.

```
Booking_ID                            36275
no_of_adults                              5
no_of_children                            6
no_of_weekend_nights                      8
no_of_week_nights                        18
type_of_meal_plan                         4
required_car_parking_space                2
room_type_reserved                        7
lead_time                               352
arrival_year                              2
arrival_month                            12
arrival_date                             31
market_segment_type                       5
repeated_guest                            2
no_of_previous_cancellations              9
no_of_previous_bookings_not_canceled     59
avg_price_per_room                     3930
no_of_special_requests                    6
booking_status                            2
dtype: int64
```

Bằng việc tìm hiểu về các giá trị duy nhất của mỗi thuộc tính, nhóm đã có thể đưa ra nhận xét rằng: Đa số các biến định danh đều mang số giá trị riêng biệt tương đối ít. Ngược lại với các biến định lượng là '**lead_time**' và '**avg_price_per_room**' có rất nhiều giá trị. Điều này sẽ có thể gây khó khăn cho nhóm trong quá trình phân tích dữ liệu về sau.

Sau khi đã có hiểu biết cơ bản về bộ dữ liệu, giờ đây nhóm sẽ thực hiện một vài thao tác như loại bỏ các thuộc tính không cần thiết, và chỉnh dạng dữ liệu của một số thuộc tính để giúp bộ dữ liệu trở nên phù hợp hơn cho việc phân tích và xây dựng các mô hình phân lớp.

Ở đây, nhóm sẽ loại bỏ cột '**Booking_ID**' khỏi bộ dữ liệu vì đây chỉ là thuộc tính chỉ mã đặt phòng nên ta không cần quan tâm đến.

```
# Loại bỏ cột Booking_ID
df = df.drop(columns = ['Booking_ID'])
```

Tiếp theo đó, nhóm cũng sẽ chuyển kiểu dữ liệu hai biến '**repeated_guest**' và '**required_car_parking_space**' mang giá trị {0, 1}, từ biến định lượng sang biến định danh để thuận tiện cho quá trình phân lớp bộ dữ liệu.

```
# Chuyển kiểu cột dữ liệu
df['repeated_guest'] = df['repeated_guest'].astype(object)
df['required_car_parking_space'] = df['required_car_parking_space'].astype(object)
```

Mục tiêu của bài toán là dự đoán khả năng hủy đặt phòng của khách hàng dựa trên việc kết hợp các thuộc tính như '**no_of_adults**' (số người lớn), '**room_type_reserved**' (loại phòng), '**avg_price_per_room**' (giá phòng bình quân theo ngày) từ đó khai thác mẫu phổ biến để chọn lựa các đặc trưng để xây dựng mô hình phân lớp. Do vậy, ở đây ta sẽ tiến hành đặt biến mục tiêu là '**booking_status**' (đơn đặt phòng bị hủy hoặc không).

```
# Đặt cột booking_status là target
target = df['booking_status']
df = df.drop(columns = 'booking_status')
```

b. Làm sạch dữ liệu

Đây được xem là một trong những bước quan trọng nhất trong quá trình tiền xử lý dữ liệu. Bước này giúp nhóm có thể phát hiện các giá trị bị thiếu (missing values) và các giá trị bất thường/ nhiễu (noisy values) và xử lý chúng. Bước này giúp bộ dữ liệu đảm bảo tính nhất quán và chắc chắn rằng không có sự thiếu hụt về thông tin. Từ đó,

nhóm có thể sử dụng bộ dữ liệu này và tiến hành phân tích ở các bước tiếp theo với độ tin cậy cao.

- ***Kiểm tra giá trị bị thiếu***

```
no_of_adults                            0
no_of_children                          0
no_of_weekend_nights                    0
no_of_week_nights                       0
type_of_meal_plan                       0
required_car_parking_space              0
room_type_reserved                      0
lead_time                               0
arrival_year                            0
arrival_month                           0
arrival_date                            0
market_segment_type                     0
repeated_guest                          0
no_of_previous_cancellations            0
no_of_previous_bookings_not_canceled    0
avg_price_per_room                      0
no_of_special_requests                  0
dtype: int64
```

Sau khi tiến hành kiểm tra các giá trị bị thiếu, nhóm nhận thấy rằng trong các cột không có giá trị nào bị thiếu nên không cần phải xử lý bước này.

- ***Kiểm tra giá trị nhiễu***

Đầu tiên, nhóm sẽ tiến hành kiểm tra nhiễu ở tất cả các biến định lượng. Ở phần này, nhóm sẽ sử dụng quy tắc 3-sigma để tìm ra các outlier của từng cột một. Outlier được xem là những giá trị nằm ngoài khoảng ba độ lệch chuẩn. Do vậy, nhóm sẽ tính toán các giới hạn trên/ dưới so với giá trị trung bình cho từng biến.

```python
# Sử dụng hàm để kiểm tra outlier cho tất cả các cột trong DataFrame
def check_outliers_using_3_sigma(df):
    has_outliers = True
    while has_outliers:
        has_outliers = False
        for col in df.columns:
            if df[col].dtype in ['int64', 'float64']:
                lech3sigma_tren = round(df[col].mean() + 3 * df[col].std(ddof=1))
                lech3sigma_duoi = round(df[col].mean() - 3 * df[col].std(ddof=1))
                df_filter = df[(df[col] > lech3sigma_tren) | (df[col] < lech3sigma_duoi)]

                if df_filter[col].count() > 0:
                    print(f'Cột {col} có {df_filter[col].count()} giá trị nhiễu là các giá trị ngoài khoảng ({lech3sigma_duoi}, {lech3sigma_tren})')
                    # Thay thế nhiễu bằng trung vị
                    trungvi = df[col].median()
                    df.loc[(df[col] > lech3sigma_tren) | (df[col] < lech3sigma_duoi), col] = trungvi
                    has_outliers = True
                else:
                    print(f'Cột {col}: Không có outlier.')

# Gọi hàm trên DataFrame df
check_outliers_using_3_sigma(df)
```

Sau khi đã tính toán xong các giá trị trên, nhóm sẽ tiến hành so sánh giá trị của mỗi phần tử trong cột với giới hạn trên/ dưới tương ứng. Nếu giá trị nằm ngoài khoảng này, nó sẽ được xem là ngoại lệ và được in ra theo danh sách như sau:

```
Cột no_of_adults có 16 giá trị nhiễu là các giá trị ngoài khoảng (0, 3)
Cột no_of_children có 1080 giá trị nhiễu là các giá trị ngoài khoảng (-1, 1)
Cột no_of_weekend_nights có 184 giá trị nhiễu là các giá trị ngoài khoảng (-2, 3)
Cột no_of_week_nights có 324 giá trị nhiễu là các giá trị ngoài khoảng (-2, 6)
Cột lead_time có 376 giá trị nhiễu là các giá trị ngoài khoảng (-173, 343)
Cột arrival_year: Không có outlier.
Cột arrival_month: Không có outlier.
Cột arrival_date: Không có outlier.
Cột no_of_previous_cancellations có 140 giá trị nhiễu là các giá trị ngoài khoảng (-1, 1)
Cột no_of_previous_bookings_not_canceled có 267 giá trị nhiễu là các giá trị ngoài khoảng (-5, 5)
Cột avg_price_per_room có 353 giá trị nhiễu là các giá trị ngoài khoảng (-2, 209)
Cột no_of_special_requests có 86 giá trị nhiễu là các giá trị ngoài khoảng (-2, 3)
```

Sau khi tiến hành kiểm tra các giá trị bất thường của các biến có kiểu dữ liệu số, nhóm nhận thấy rằng hầu hết các biến điều có các giá trị bất thường. Do vậy, nhóm sẽ tiến hành xử lý những giá trị này bằng cách thay thế chúng bằng trung vị thay vì xóa chúng đi vì muốn đảm bảo sự đầy đủ và tránh làm mất đi thông tin ban đầu của bộ dữ liệu.

```
Cột no_of_adults: Không có outlier.
Cột no_of_children: Không có outlier.
Cột no_of_weekend_nights: Không có outlier.
Cột no_of_week_nights: Không có outlier.
Cột lead_time: Không có outlier.
Cột arrival_year: Không có outlier.
Cột arrival_month: Không có outlier.
Cột arrival_date: Không có outlier.
Cột no_of_previous_cancellations: Không có outlier.
Cột no_of_previous_bookings_not_canceled: Không có outlier.
Cột avg_price_per_room: Không có outlier.
Cột no_of_special_requests: Không có outlier.
```

Sau khi xử lý, tất cả các biến đều đã không còn giá bất thường nào.

c. Rời rạc hóa dữ liệu

Như đã tìm hiểu thông tin về bộ dữ liệu ở trên, nhóm đã biết được rằng bộ dữ liệu có chứa vài thuộc tính có kiểu dữ liệu là số (liên tục) như '**lead_time**' và '**avg_price_per_room**', và một vài thuộc tính khác cũng mang kiểu dữ liệu số (rời rạc) như '**no_of_adults**', '**no_of_children**', '**no_of_special_requests**'....

Do đó, để chuẩn bị cho quá trình phân lớp dữ liệu ở chương sau, ở phần này nhóm sẽ tiến hành rời rạc hóa các biến này.

```
# Rời rạc các biến lead_time và avg_price_per_room
df['lead_time_group'] = pd.qcut(df['lead_time'], 4)
df['avg_price_per_room_group'] = pd.qcut(df['avg_price_per_room'], 4)
df = df.drop(columns = ['lead_time', 'avg_price_per_room'])
```

Đầu tiên, nhóm sẽ xử lý các biến liên tục trước bằng cách tạo ra một cột mới có tên là '**lead_time_group**', trong đó giá trị của mỗi dòng được xác định dựa trên phân vị (quantile) của cột '**lead_time**' thành 4 khoảng (quartiles). Cụ thể, giá trị của cột mới sẽ là nhóm của giá trị trong cột 'lead_time' sao cho mỗi nhóm có số lượng gần bằng nhau. Tương tự như trên, tạo ra một cột mới có tên là '**avg_price_per_room_group**', trong đó giá trị của mỗi dòng được xác định dựa trên phân vị của cột '**avg_price_per_room**' thành 4 khoảng.

Sau khi đã xử lý các biến liên tục, nhóm sẽ tiến hành chuyển kiểu dữ liệu của tất cả các thuộc tính trong bộ dữ liệu thành biến định danh. Lý do là vì nhóm cần mã hóa tất cả các cột dữ liệu bằng phương pháp OneHotEncoding để tiện cho quá trình phân lớp dữ liệu.

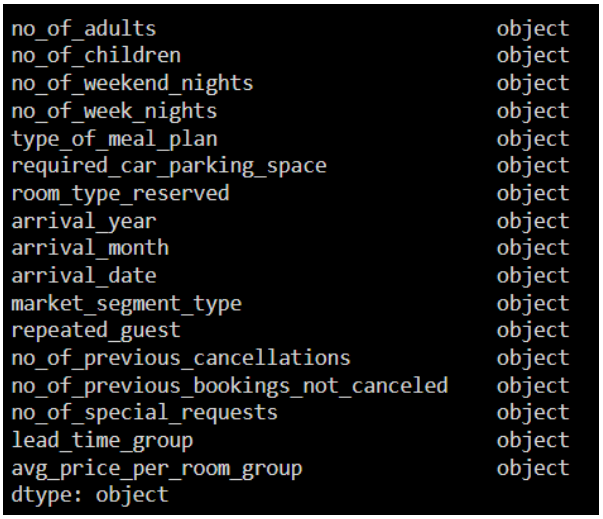
```
1 df = df.astype(object)
```

Đồng thời nhóm cũng tiếp tục xử lý các biến rời rạc. Đối với các biến này nhóm sẽ không áp dụng như cách trên. Mà thay vào đó, nhóm sẽ thay đổi giá trị ở các biến này thành dạng '**tên biến + giá trị ô tương ứng**'.

```
columns_to_exclude = ['type_of_meal_plan', 'room_type_reserved', 'market_segment_type']

# Chuyển giá trị ở các ô thành dạng "tên cột (giá trị ở ô tương ứng)" cho các cột trừ các cột có giá trị số rời rạc và nhị phân
df = df.apply(lambda x: x.astype(str).map(lambda y: f'{x.name} ({y})') if x.name not in columns_to_exclude else x)
```

Sau quá trình tiền xử lý, bộ dữ liệu còn 17 cột và 36.275 dòng, không có thay đổi gì về số lượng thuộc tính và bản ghi so với ban đầu. Tuy nhiên, về bản chất các thuộc tính đã có một số thay đổi về chỉnh dạng dữ liệu như sau:



Qua kết quả trên, nhóm có thể đưa ra nhận xét rằng, tất cả các thuộc tính đều có kiểu dữ liệu là object (biến định danh). Điều này sẽ giúp cho quá trình phân lớp trở nên dễ dàng hơn và mang lại kết quả phân lớp chính xác hơn.

CHƯƠNG V. ỨNG DỤNG GIẢI THUẬT TRÊN BỘ DỮ LIỆU

- **Mã nguồn**: Classification_using_Frequent_Patterns.ipynb

1. Chuẩn bị dữ liệu

Đầu tiên, nhóm nghiên cứu sẽ tiến hành biến đổi từ bộ dữ liệu ban đầu sang dạng dữ liệu để có thể khai thác mẫu phổ biến bằng thuật toán FP-Growth bằng cách sử dụng phương pháp OneHotEncoding. Tên của các cột mới trong dataframe này chính là giá trị của các cột ban đầu. Sau khi đã loại bỏ các cột ban đầu và thay thế bằng các cột mới được mã hóa, nhóm sẽ có được một bảng dữ liệu có dạng như sau:

	no_of_adults (2)	no_of_adults (1)	no_of_adults (3)	no_of_adults (0)	no_of_children (0)	no_of_children (1)	no_of_weekend_nights (1)	no_of_weekend_nights (2)	no_of_weekend_nights (0)	no_of_weekend_nights (3)	...
0	False	False	True	False	True	False	False	True	False	False	...
1	False	False	True	False	True	False	False	False	True	False	...
2	False	True	False	False	True	False	False	False	True	False	...
3	False	False	True	False	True	False	True	False	False	False	...
4	False	False	True	False	True	False	False	True	False	False	...
...	...	...	...	...	...	...	...	...	...	...	...
36270	False	False	False	True	True	False	False	False	True	False	...
36271	False	False	True	False	True	False	False	True	False	False	...
36272	False	False	True	False	True	False	False	False	True	False	...
36273	False	False	True	False	True	False	True	False	False	False	...
36274	False	False	True	False	True	False	False	True	False	False	...

36275 rows × 96 columns

Bộ dữ liệu ban đầu:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space
0	no_of_adults (2)	no_of_children (0)	no_of_weekend_nights (1)	no_of_week_nights (2)	Meal Plan 1	required_car_parking_space (0)
1	no_of_adults (2)	no_of_children (0)	no_of_weekend_nights (2)	no_of_week_nights (3)	Not Selected	required_car_parking_space (0)
2	no_of_adults (1)	no_of_children (0)	no_of_weekend_nights (2)	no_of_week_nights (1)	Meal Plan 1	required_car_parking_space (0)
3	no_of_adults (2)	no_of_children (0)	no_of_weekend_nights (0)	no_of_week_nights (2)	Meal Plan 1	required_car_parking_space (0)
4	no_of_adults (2)	no_of_children (0)	no_of_weekend_nights (1)	no_of_week_nights (1)	Not Selected	required_car_parking_space (0)

2. Khai thác mẫu phổ biến và chọn lọc mẫu

Để tìm ra các mẫu phổ biến từ dữ liệu đã chuẩn bị ban đầu, ở đây nhóm sẽ sử dụng thuật toán khai thác mẫu phổ biến FP-Growth. Tuy nhiên, đối với việc lựa chọn ngưỡng hỗ trợ tối thiểu (minSup) như đã đề cập ở **Chương II**, nhóm cần phải đưa ra được chiến lược lựa chọn phù hợp để tránh tình trạng mất mát độ chính xác hoặc Overfitting từ việc lựa chọn ngưỡng này quá cao hoặc quá thấp gây nên.

Chính vì thế nhóm thực hiện sử dụng **Grid Search** để tìm ra giá trị minSup có độ chính xác (Accuracy) cao nhất cho mô hình phân lớp mà nhóm lựa chọn:

```
Giá trị accuracy của minSup=0.3 là:0.735354927636113
Giá trị accuracy của minSup=0.4 là:0.7390764989662302
Giá trị accuracy của minSup=0.5 là:0.7389386629910406
Giá trị accuracy của minSup=0.6 là:0.6763611302549966
Giá trị accuracy của minSup=0.7 là:0.6763611302549966
Giá trị accuracy của minSup=0.8 là:0.6763611302549966
```

Dựa trên kết quả đánh giá, nhóm nhận thấy việc lựa chọn minSup ở mức 0.4, 0.5 hoặc thậm chí là 0.3 đều mang lại độ chính xác tương tự, xấp xỉ 0.74. Tuy nhiên, để tránh tình trạng mô hình học quá nhiều các kết hợp không mang ý nghĩa, dẫn đến hiện tượng Overfitting, từ việc lựa chọn minSup thấp hơn mức trung bình làm xuất hiện quá nhiều mẫu phổ biến. Chính vì thế, nhóm đã quyết định lựa chọn **minSup = 0.5** để thực hiện khai thác mẫu phổ biến thông qua thuật toán FP-Growth.

Đầu tiên, nhóm sẽ áp dụng thuật toán FP-Growth để tìm ra các mẫu phổ biến trong dataframe. Các mẫu phổ biến này sẽ được lưu trữ trong '**frequent_patterns**'. Sau đó, nhóm sẽ tính toán số lần xuất hiện của mỗi item và sắp xếp chúng theo thứ tự giảm dần rồi so sánh với min_sup. Nếu các mẫu có chỉ số support lớn hơn so với min_sup, mẫu đó sẽ được đưa vào danh sách mẫu phổ biến.

```
frequent_patterns = fpgrowth(df, min_support = 0.5, use_colnames=True)
frequent_patterns['itemsets'] = frequent_patterns['itemsets'].apply(lambda x: ', '.join(x))

# Sắp xếp chỉ số dựa trên số lượng từ
word_counts = frequent_patterns['itemsets'].str.split(",").apply(len)
sorted_indices = np.argsort(word_counts)

# Sắp xếp lại cột "Itemsets" dựa trên chỉ số đã sắp xếp
frequent_patterns['itemsets'] = frequent_patterns['itemsets'].iloc[sorted_indices].values

# Lưu các tập phổ biến lớn nhất
frequent_patterns_filter = frequent_patterns.copy()
frequent_patterns_reverse = frequent_patterns.iloc[::-1]
for valuehigh in frequent_patterns_reverse['itemsets']:
  for valuelow in frequent_patterns['itemsets']:
    dem = 0
    if len(valuelow.split(', ')) == len(valuehigh.split(', ')):
      break
    for l in valuelow.split(', '):
      if l in valuehigh.split(', '):
        dem = dem + 1
    if dem == len(valuelow.split(', ')):
      frequent_patterns_filter = frequent_patterns_filter[frequent_patterns['itemsets'] != valuelow]
```

Sau đây là bảng dữ liệu chứa các mẫu phổ biến mà nhóm đã tìm được:

	support	itemsets
0	1.000000	no_of_previous_bookings_not_canceled (0)
1	1.000000	Complementary
2	0.974363	arrival_year (2018)
3	0.969014	no_of_adults (3)
4	0.955396	Meal Plan 1
...	...	...
422	0.502329	required_car_parking_space (0), Meal Plan 1, n...
423	0.504121	required_car_parking_space (0), no_of_previous...
424	0.502329	required_car_parking_space (0), no_of_previous...
425	0.502329	required_car_parking_space (0), no_of_previous...
426	0.502329	required_car_parking_space (0), no_of_previous...

427 rows × 2 columns

Để có thể chọn lọc ra mẫu phổ biến lớn nhất từ các mẫu phổ biến đã tìm được, nhóm sẽ dùng vòng lặp để so sánh số lượng các tập itemsets và lọc bỏ các mẫu phổ biến mà có số tập itemsets nhỏ hơn.

	support	itemsets
410	0.537092	required_car_parking_space (0), no_of_previous...
411	0.537092	required_car_parking_space (0), Meal Plan 1, n...
417	0.520276	Meal Plan 1, no_of_previous_bookings_not_cance...
418	0.520276	required_car_parking_space (0), Meal Plan 1, n...
421	0.504121	required_car_parking_space (0), Meal Plan 1, n...
422	0.502329	required_car_parking_space (0), Meal Plan 1, n...
423	0.504121	required_car_parking_space (0), no_of_previous...
424	0.502329	required_car_parking_space (0), no_of_previous...
425	0.502329	required_car_parking_space (0), no_of_previous...
426	0.502329	required_car_parking_space (0), no_of_previous...

3. Chọn lọc bộ dữ liệu từ mẫu phổ biến

Nhóm sẽ chọn lọc lại các đặc trưng chỉ giữ lại các đặc trưng là mẫu phổ biến đồng thời tạo ra các đặc trưng bằng toán tử **XOR** là các mẫu phổ biến lớn nhất từ các đặc trưng đơn lẻ.

```
# Lấy các cột là mẫu phổ biến
df_filtered = df[df.columns[df.columns.isin(frequent_patterns.itemsets)]]

## XOR của các tập phổ biến
for item in frequent_patterns_filter['itemsets']:
  row = []
  value = []
  for index in item.split(', '):
    row.append(index)
  name = ' x '.join(row)

  for i in row:
    value.append(df_filtered[i])

  result = value[0]
  for i in range(1, len(value)):
    result = [x ^ y for x, y in zip(result, value[i])]

  df_filtered[name] = result

# Xây dựng đặc trưng
features = df.astype(int).values
```

Sau khi hoàn tất việc hoàn tất bước này, bài toán mà nhóm đang giải quyết đã chuyển thành một bài toán học có giám sát thông thường.

	no_of_adults (3)	no_of_children (0)	Meal Plan 1	required_car_parking_space (0)	Room_Type 1	arrival_year (2018)	Complementary	repeated_guest (0)	no_of_previous_cancellations (0)
0	True	True	True	True	True	False	False	True	True
1	True	True	False	True	True	True	True	True	True
2	False	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True
4	True	True	False	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...
36270	False	True	True	True	False	True	True	True	True
36271	True	True	True	True	True	True	True	True	True
36272	True	True	True	True	True	True	True	True	True
36273	True	True	False	True	True	True	True	True	True
36274	True	True	True	True	True	True	False	True	True

required_car_parking_space (0) x no_of_previous_bookings_not_canceled (0) x repeated_guest (0) x arrival_year (2018) x no_of_adults (3) x no_of_previous_cancellations (0) x no_of_children (0)	required_car_parking_space (0) x no_of_previous_bookings_not_canceled (0) x Room_Type 1 x repeated_guest (0) x no_of_adults (3) x no_of_previous_cancellations (0) x no_of_children (0)	required_car_parking_space (0) x no_of_previous_bookings_not_canceled (0) x Room_Type 1 x repeated_guest (0) x arrival_year (2018) x no_of_previous_cancellations (0) x no_of_children (0)	required_car_parking_space (0) x no_of_previous_bookings_not_canceled (0) x repeated_guest (0) x arrival_year (2018) x no_of_previous_cancellations (0) x Complementary x no_of_children (0)	booking_status
False	True	False	True	Not_Canceled
True	True	True	True	Not_Canceled
False	False	True	True	Canceled
True	True	True	True	Canceled
True	True	True	True	Canceled
...	...	...	...	...
False	True	False	True	Not_Canceled
True	True	True	True	Canceled
True	True	True	True	Not_Canceled
True	True	True	True	Canceled
True	True	True	False	Not_Canceled

4. Xây dựng mô hình phân lớp

Để xây dựng mô hình phân lớp, nhóm sử dụng thuật toán SVM (Support Vector Machine), vì SVM làm việc hiệu quả với bộ dữ liệu có số lượng đầu vào cao. Hơn thế nữa, SVM còn có khả năng kiểm soát hiệu quả và hạn chế tình trạng quá khớp (Overfitting).

Ngoài ra, nhóm còn quyết định huấn luyện mô hình dựa trên K-Fold Cross-Validation (CV) - là một phương pháp chia dữ liệu thành các tập con để đánh giá hiệu suất của các mô hình máy học hoặc học sâu. Quá trình này giúp đánh giá khả năng tổng quát hóa của mô hình đối với dữ liệu mới, không được sử dụng trong quá trình huấn luyện.

Khi sử dụng phương pháp này, nhóm có thể đảm bảo tất cả dữ liệu đều được sử dụng để đánh giá mô hình, giúp tăng cường độ chính xác và đảm bảo tính đại diện của mẫu. Cross-Validation có thể giúp phát hiện các vấn đề như Overfitting (quá khớp) bằng cách đánh giá hiệu suất trên các tập kiểm thử một cách độc lập.

Để thực hiện, nhóm thực hiện tuần tự các bước sau:

- Đầu tiên ta sẽ gán giá trị cần dự đoán là biến target '**booking_status**'.

```
y = train['booking_status']
X = train.drop(columns = 'booking_status')
```

- Tiếp sau đó, để huấn luyện mô hình ta cần tiến hành chia dữ liệu thành 5 phần để thực hiện 5-Fold Cross-Validation.

- Cuối cùng, ta sẽ dùng mô hình SVM để huấn luyện dựa trên các Fold huấn luyện và đánh giá nó trên Fold kiểm thử:

```python
# Số fold
k = 5

# Tạo đối tượng KFold
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Xây dựng mô hình
model = svm.SVC(kernel = 'linear')

# Biến để lưu trữ các kết quả đánh giá
accuracy_scores = []

# Chia dữ liệu và thực hiện cross-validation
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Huấn luyện mô hình
    model.fit(X_train, y_train)
```

- Sau khi đã huấn luyện mô hình thành công, ta sẽ dùng mô hình này để dự đoán kết quả, và so sánh các giá trị dự đoán so với thực tế.

```python
# Dự đoán nhãn lớp cho tập kiểm tra
y_pred = model.predict(X_test)
```

	Actual	Predicted
1	Not_Canceled	Not_Canceled
9	Not_Canceled	Not_Canceled
16	Not_Canceled	Not_Canceled
18	Canceled	Not_Canceled
53	Not_Canceled	Not_Canceled
...	...	...
36241	Canceled	Not_Canceled
36242	Not_Canceled	Not_Canceled
36258	Not_Canceled	Not_Canceled
36260	Canceled	Not_Canceled
36263	Not_Canceled	Canceled

7255 rows × 2 columns

CHƯƠNG VI. ĐÁNH GIÁ

1. Đánh giá mô hình

Trong các bài toán phân loại, accuracy (độ chính xác) thường được sử dụng để đánh giá hiệu suất của mô hình. Độ chính xác được tính bằng tỉ lệ giữa số lượng dự đoán đúng và tổng số mẫu trong tập kiểm tra. Độ chính xác càng cao đồng nghĩa với việc mô hình có khả năng dự đoán đúng cao trên tập kiểm tra. Dựa trên K-Fold Cross-Validation (CV), nhóm thu được độ chính xác của các Fold và độ chính xác của thuật toán như sau:

```
# Đánh giá mô hình bằng độ chính xác
accuracy = accuracy_score(y_test, y_pred)
accuracy_scores.append(accuracy)
```

```
Fold 1: Accuracy = 0.729841488628532
Fold 2: Accuracy = 0.7422467263955892
Fold 3: Accuracy = 0.7399035148173674
Fold 4: Accuracy = 0.7346657477601654
Fold 5: Accuracy = 0.7436250861474845

Average Accuracy across all folds: 0.7380565127498278
```

Từ kết quả trên, nhóm có thể đưa ra một số nhận xét như sau:

- Mỗi chỉ số trên là đại diện cho độ chính xác của mô hình trên một Fold cụ thể trong quá trình Cross-Validation.
- Với độ chính xác trung bình của mô hình khi được đánh giá trên tất cả các Fold là 0.738, thì nó có thể được coi là một chỉ số tương đối tốt và đáng tin cậy. Cho thấy rằng, đây là mô hình có khả năng phân lớp đúng với một phần dữ liệu lớn.

2. So sánh độ chính xác của mô hình phân lớp giữa tập dữ liệu sử dụng mẫu phổ biến và không sử dụng mẫu phổ biến

```
Fold 1: Accuracy = 0.6669882839421089
Fold 2: Accuracy = 0.6758097863542385
Fold 3: Accuracy = 0.6665747760165404
Fold 4: Accuracy = 0.6719503790489317
Fold 5: Accuracy = 0.6804962095106822

Average Accuracy across all folds: 0.6723638869745003
```

Bảng kết quả trên là độ chính xác của mô hình trên mỗi Fold cụ thể trong quá trình Cross-Validation của bộ dữ liệu được phân lớp bằng phương pháp SVM nhưng chỉ xử lý như thông thường mà không sử dụng các mẫu phổ biến.

Nhóm có thể nhận thấy rõ các thông tin sau: Hầu hết các độ chính xác qua mỗi Fold của bộ dữ liệu được phân lớp không sử dụng mẫu phổ biến thì tương đối thấp hơn là khi sử dụng mẫu phổ biến.

Từ đó nhóm có thể đưa ra một số kết luận như sau:

- Phân loại dựa trên tập phổ biến có thể tích hợp tri thức từ dữ liệu thông qua việc sử dụng quy tắc kết hợp. Điều này giúp mô hình hiểu biểu diễn mối quan hệ phức tạp giữa các đặc trưng và lớp mục tiêu.
- Khi có một lượng lớn dữ liệu, phân loại dựa trên tập phổ biến có thể giúp tìm ra các quy tắc phổ biến mà có thể áp dụng rộng rãi cho toàn bộ tập dữ liệu mà không cần phải xem xét mỗi đặc trưng độc lập.
- Việc sử dụng các quy tắc phổ biến giúp khám phá mối liên kết và quan hệ giữa các đặc trưng, cung cấp hiểu biết sâu sắc hơn về dữ liệu.
- Thay vì sử dụng toàn bộ tập đặc trưng, phân loại dựa trên tập phổ biến có thể giúp giảm chiều dữ liệu bằng cách tập trung vào những quy tắc quan trọng nhất và loại bỏ những đặc trưng ít quan trọng.

Tuy nhiên, phương pháp phân loại dựa trên mẫu phổ biến và phương pháp phân loại thông thường là hai phương pháp có những yêu cầu về bộ dữ liệu khác nhau. Lựa chọn giữa phân loại thông thường và phân loại dựa trên tập phổ biến phụ thuộc vào bối cảnh cụ thể của nhiệm vụ và đặc điểm của dữ liệu.

Nếu dữ liệu có tính chất phổ biến và quan trọng là tìm hiểu mẫu xuất hiện thường xuyên, phân loại dựa trên mẫu phổ biến có thể được xem xét. Đối với dữ liệu phức tạp và đa dạng, phân loại thông thường thường là lựa chọn phù hợp. Phân loại thông thường thường đạt hiệu suất tốt trên nhiều loại bài toán. Phân loại dựa trên mẫu phổ biến có thể hiệu quả trong các tình huống cụ thể.

Chính vì vậy mà việc lựa chọn giữa phân loại dựa trên mẫu phổ biến và phân loại thông thường phụ thuộc nhiều vào đặc tính cụ thể của dữ liệu và mục tiêu của bài toán phân loại.

3. Bảng đánh giá các phương pháp phân lớp

Có thể nói, việc lựa chọn mô hình phân lớp tốt nhất cho bài toán phân loại dựa trên mẫu thường xuyên phụ thuộc vào nhiều yếu tố, bao gồm kích thước dữ liệu, đặc trưng của dữ liệu, và yêu cầu cụ thể của bài toán. Tuy nhiên, có một số mô hình phổ biến và hiệu quả được sử dụng rộng rãi trong các tình huống khác nhau mà điển hình chúng ta có thể kể đến như Support Vector Machine (SVM), Decision Tree, k-Nearest Neighbors (kNN), Naive Bayes.

Nhằm giúp chúng ta có thêm cái nhìn tổng quan hơn về bốn phương pháp phân lớp này. Dưới đây là một số so sánh giữa bốn mô hình phân lớp SVM, Decision Tree, Naive Bayes, và kNN khi áp dụng cho bài toán phân loại dựa trên mẫu thường xuyên:

- **Support Vector Machines (SVM):**

  ***Ưu điểm*:**

  - Hiệu suất tốt trong không gian đặc trưng lớn.
  - Khả năng xử lý tốt với dữ liệu không gian cao.
  - Có thể sử dụng kernel trick để xử lý dữ liệu phi tuyến tính.

  ***Nhược điểm*:**

  - Đòi hỏi thời gian và tài nguyên tính toán lớn đối với dữ liệu lớn.

- **Decision Tree:**

  ***Ưu điểm*:**

  - Dễ giải thích và hiểu.
  - Không đòi hỏi nhiều tiền xử lý cho dữ liệu.
  - Có thể xử lý cả dữ liệu số và dữ liệu rời rạc.

  ***Nhược điểm*:**

  - Dễ bị quá mức phức tạp và dễ bị overfitting.
  - Không xử lý tốt với dữ liệu có tính tuyến tính cao.

- **Naive Bayes:**

  ***Ưu điểm*:**

  - Dễ triển khai và nhanh chóng.
  - Hoạt động tốt với dữ liệu lớn và có nhiều đặc trưng.
  - Độ chính xác tốt đối với các vấn đề đơn giản.

  ***Nhược điểm*:**

  - Giả định về độc lập giữa các đặc trưng có thể không phù hợp với thực tế.
  - Độ chính xác giảm khi các đặc trưng không độc lập.

- **k-Nearest Neighbors (KNN):**

***Ưu điểm*:**

- Dễ triển khai và không yêu cầu giả định về phân phối của dữ liệu.
- Hoạt động tốt với dữ liệu không gian thấp và có tính tuyến tính.

***Nhược điểm*:**

- Đòi hỏi nhiều tài nguyên tính toán khi k lớn hoặc khi dữ liệu lớn.
- Nhạy cảm với nhiễu và các giá trị ngoại lệ.

Để có thể lựa chọn được mô hình tốt nhất, thường thì việc thử nghiệm và đánh giá trên tập dữ liệu kiểm thử là cách tốt nhất để xác định mô hình nào phù hợp nhất cho bài toán này. Do vậy, với bộ dữ liệu đã được chuẩn bị để tiến hành phân lớp, chúng ta sẽ đồng thời xây dựng bốn mô hình phân lớp này trên cùng một bộ dữ liệu để so sánh độ chính xác giữa các mô hình này với nhau. Từ đó, chọn ra mô hình phân lớp tốt nhất cho bài toán này.

SVM:

```
y_pred = svmmodel.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy của mô hình SVM: {accuracy:6f}')
```

```
Accuracy của mô hình SVM: 0.738939
```

Decision Tree:

```
y_pred = DCTmodel.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy của mô hình cây quyết định: {accuracy:6f}')
```

```
Accuracy của mô hình cây quyết định: 0.749552
```

kNN:

```
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy của mô hình knn: {accuracy:6f}')
```

```
Accuracy của mô hình knn: 0.738387
```

Naive Bayes:

```
y_pred = NBmodel.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy của mô hình Naive Bayes: {accuracy:6f}')
```

```
Accuracy của mô hình Naive Bayes: 0.491661
```

Sau đây là bảng so sánh về độ chính xác của 4 mô hình phân lớp:

Mô hình	*SVM*	*Decision Tree*	*kNN*	*Naive Bayes*
Độ chính xác	73.89%	**74.96%**	73.84%	49.17%

Nhận xét:

Dựa trên kết quả của bảng so sánh trên, có thể thấy rằng Decision Tree có độ chính xác cao nhất (74.96%), tiếp theo là SVM (73.89%), kNN (73.84%), và Naive Bayes có độ chính xác thấp nhất (49.17%).

Vậy ta có thể dễ dàng đưa ra kết luận rằng, đối với bài toán phân loại dữ liệu sử dụng mẫu thường xuyên áp dụng trên bộ dữ liệu Hotel Reservations thì **Decision Tree là mô hình tốt nhất** trong số bốn mô hình phân loại đã được thử nghiệm.

CHƯƠNG VII. KẾT LUẬN

Sau khi thực hiện dự án nghiên cứu về: "**Ứng dụng phương pháp phân lớp dựa trên mẫu phổ biến để dự đoán khả năng hủy đặt phòng của khách hàng**", nhóm nghiên cứu đã tiến hành phân tích và triển khai phương pháp lấy mẫu phổ biến để làm cơ sở thực hiện phân loại khách hàng. Mục tiêu là dự đoán khả năng hủy đặt phòng của họ, từ đó mang lại chiến lược chi tiết giúp doanh nghiệp hiểu sâu hơn về những tình huống cụ thể dẫn đến quyết định này từ phía khách hàng. Nhờ vào quá trình này, không chỉ có thể cải thiện trải nghiệm của khách hàng mà còn tăng cường sự tương tác trong quá trình họ sử dụng dịch vụ hoặc liên lạc với doanh nghiệp.

Đặc điểm nổi bật:

Trong bài nghiên cứu, nhóm đã áp dụng đa các phương pháp, bao gồm: lựa chọn mẫu phổ biến thích hợp, sau đó mới thực hiện phân lớp để đạt được hiệu quả cao nhất. Thông qua phương pháp này, nhóm có thể xác định các tập mục quan trọng trong dữ liệu khách hàng, điều này giúp nhóm hiểu rõ hơn về xu hướng có thể xảy ra, cũng như mối quan hệ giữa các mục trong dữ liệu, từ đó cung cấp thông tin quan trọng để phân loại khách hàng. Hơn nữa, việc lựa chọn và áp dụng các phương pháp phân lớp phù hợp giúp nhóm đạt được kết quả phân loại chính xác và hiệu quả. Điều này giúp tối ưu hóa quy trình tính toán và sử dụng các kỹ thuật song song để gia tăng tốc độ xử lý.

Về mặt thuận lợi:

- **Xác định mẫu phổ biến**: Phương pháp này giúp phân tích và nhận biết các mẫu phổ biến trong dữ liệu khách hàng, giúp hiểu sâu hơn về hành vi lựa chọn của họ, hỗ trợ đưa ra dự đoán về khả năng xảy ra rủi ro của doanh nghiệp đối với việc hủy đặt phòng.

- **Tính linh hoạt và đa dạng**: Giải thuật này có khả năng xử lý đa dạng và linh hoạt các thuộc tính và đặc điểm của khách hàng. Nó có thể được áp dụng cho nhiều lĩnh vực nhà hàng - khách sạn khác nhau, từ đó có thể điều chỉnh mô hình để phù hợp với yêu cầu cụ thể của từng đối tượng khách hàng.

- **Độ chính xác cao**: Phương pháp này dự đoán nhãn sản phẩm với độ chính xác cao. Bằng cách sử dụng các mẫu phổ biến có sẵn trong dữ liệu, mô hình tìm ra sự tương đồng và liên quan giữa hành vi của khách hàng để đưa ra dự đoán chính xác về khả năng đặt phòng của họ.

Về mặt hạn chế:

- **Lựa chọn mẫu phổ biến**: Nhóm nhận thấy số lượng các mẫu phổ biến xuất hiện rất lớn, nhưng nhóm không thể xác định rõ số lượng mẫu nào sẽ cần thiết cho mô hình phân lớp. Điều này vẫn chưa được nhóm trình bày rõ trong báo cáo.

- **Thiếu kinh nghiệm thực tế**: Vì nhóm hiện đang là sinh viên, do đó chưa có nhiều kinh nghiệm trong lĩnh vực phân tích dữ liệu trong ngành nhà hàng - khách sạn. Trong quá trình thực hiện đề tài, nhóm đã tham khảo nhiều tài liệu từ các nguồn bên ngoài để giải quyết bài toán. Do đó, chúng em không thể hoàn toàn đảm bảo tính chính xác của kết quả nghiên cứu.

PHỤ LỤC

1. Mã nguồn

- Link Github: Neyung/NAT_ML_GROUP6 (github.com)
- Ứng dụng vào bộ dữ liệu: Classification_using_Frequent_Patterns.ipynb

2. Bảng phân công

THÀNH VIÊN	PHÂN CÔNG	ĐÁNH GIÁ
Võ Minh Nguyên	Tìm hiểu thuật toán, Viết các bước xây dựng thuật toán, Tổng hợp nội dung báo cáo	100%
Nguyễn Phúc Bảo Nhân	Tìm hiểu thuật toán, Sơ lược đề tài, Viết kết luận	100%
Phan Đình Nhân	Tìm hiểu thuật toán, Viết khái niệm tổng quát, Đánh giá minSup	100%
Nguyễn Quang Nhật	Tìm hiểu thuật toán, Tiền xử lý, Xây dựng thuật toán	100%
Nguyễn Thị Ngọc Nhi	Tìm hiểu thuật toán, Xây dựng thuật toán, Đánh giá	100%

3. Tài liệu tham khảo

[1] B. Goethals, "Survey on frequent pattern mining," University of Helsinki, 2003.
[2] Nguyen An Te, "Data Mining.2023.Ch4 - Luat ket hop", UEH, 2023.
[3] Hong Cheng el at, "Discriminative Frequent Pattern Analysis for Effective Classification", University of Illinois, 2007. Link: icde07_discriminative.pdf (ucsb.edu)