

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC NGOẠI NGỮ-TIN HỌC TP. HỒ CHÍ MINH**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC**  
**KHAI KHOÁNG DỮ LIỆU**

**ĐỀ TÀI**

**Dự đoán doanh số bán hàng dựa trên thuật toán Probabilistic  
Model-Based Clustering (Thuật Toán EM)**

**SVTH:**

Nguyễn Bảo Khang – 21DH110785

Lê Đăng Tùng – 21DH114280

Lê Quốc Đạt – 21DH113563

Nguyễn Dương Khoa – 21DH113776

**GVHD: ThS. Nguyễn Thị Phương Trang**

**Thành phố Hồ Chí Minh, ngày 8 tháng 7 năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC NGOẠI NGỮ-TIN HỌC TP. HỒ CHÍ MINH**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC**  
**KHAI KHOÁNG DỮ LIỆU**

**ĐỀ TÀI**

**Dự đoán doanh số bán hàng dựa trên thuật toán Probabilistic  
Model-Based Clustering (Thuật Toán EM)**

**SVTH:**

Nguyễn Bảo Khang – 21DH110785

Lê Đăng Tùng – 21DH114280

Lê Quốc Đạt – 21DH113563

Nguyễn Dương Khoa – 21DH113776

**GVHD: ThS. Nguyễn Thị Phương Trang**

**Thành phố Hồ Chí Minh, ngày 8 tháng 7 năm 2024**

## BẢNG ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

Nhiệm vụ	Thành viên thực hiện	Kết quả
Tìm data	Lê Đăng Tùng, Nguyễn Dương Khoa	Hoàn thành
Tiền xử lí dữ liệu	Lê Quốc Đạt, Nguyễn Dương Khoa	Hoàn thành
Xử lí luật kết hợp	Lê Quốc Đạt, Lê Đăng Tùng	Hoàn thành
Xây dựng, huấn luyện, đánh giá mô hình	Nguyễn Bảo Khang	Hoàn thành
Viết báo cáo	Nguyễn Bảo Khang	Hoàn thành

*Bảng 1. Đánh giá mức độ hoàn thành*

## MỤC LỤC

<b>BẢNG ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH.....</b>	<b>1</b>
<b>MỤC LỤC .....</b>	<b>2</b>
<b>DANH MỤC HÌNH ẢNH .....</b>	<b>4</b>
<b>DANH MỤC BẢNG BIỂU .....</b>	<b>5</b>
<b>CHƯƠNG 1: GIỚI THIỆU .....</b>	<b>6</b>
1.1 Giới thiệu đề tài .....	6
1.2 Nội dung thực hiện .....	6
1.3 Giới hạn đề tài.....	7
1.4 Bố cục báo cáo .....	7
<b>CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....</b>	<b>9</b>
2.1 Các khái niệm cơ bản.....	9
2.1.1 Tứ phân vị xử lý giá trị ngoại biên.....	9
2.1.2 Các tập mục thường xuyên và luật kết hợp.....	10
2.1.3 Thuật toán Apriori .....	11
2.1.4 Thuật toán EM.....	13
2.1.5 Hồi qui tuyến tính (Linear Regression) .....	16
2.1.6 Các thang đo.....	17
2.1.7 Gaussian Mixture Model.....	18
2.1.8 Label Encoder .....	19
2.1.9 Standard Scaler .....	19
2.2 Các công trình liên quan .....	19
<b>CHƯƠNG 3: DỰ ĐOÁN DOANH SỐ BÁN HÀNG DỰA TRÊN THUẬT TOÁN PROBABILISTIC MODEL-BASED CLUSTERING (THUẬT TOÁN EM) .....</b>	<b>22</b>
3.1 Tổng quan phương pháp hiện thực .....	22
3.2 Công cụ hiện thực .....	22
3.3 Tập dữ liệu .....	23
3.4 Tiền xử lý dữ liệu.....	27
3.5 Triển khai khai thác luật kết hợp .....	33
3.6 Quá trình phân cụm, dự đoán.....	35
3.7 Kết quả.....	40

<b>CHƯƠNG 4: KẾT QUẢ - KẾT LUẬN .....</b>	<b>41</b>
4.1 Nhận xét kết quả .....	41
4.2 Ưu nhược điểm .....	41
4.3 Hướng phát triển .....	41
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>42</b>

## DANH MỤC HÌNH ẢNH

Hình 1. Quá trình lập đi lập lại của EM để cải thiện độ phù hợp của mô hình.....	13
Hình 2. Cách thức hoạt động của thuật toán Expectation-Maximization (EM).....	15
Hình 3. Đọc dữ liệu .....	23
Hình 4. Hiển thị 5 dòng đầu .....	24
Hình 5. Xem thông tin của tập dữ liệu .....	24
Hình 6. Hiển thị mô tả tập train.....	26
Hình 7. Hiển thị mô tả tập test.....	26
Hình 8. Hiển thị mô tả tập sample.....	26
Hình 9. Kiểm tra giá trị duy nhất.....	27
Hình 10. Kiểm tra giá trị thiếu .....	28
Hình 11. Xử lý giá trị thiếu .....	29
Hình 12. Kiểm tra lại giá trị thiếu .....	30
Hình 13. Xử lý giá trị không nhất quán.....	30
Hình 14. Kiểm tra giá trị ngoại biên.....	31
Hình 15. Kiểm tra giá trị ngoại biên sau xử lý.....	32
Hình 16. Xóa cột định danh.....	32
Hình 17. Copy file train mới .....	33
Hình 18. Rời rạc hóa các biến dạng số.....	33
Hình 19. Xem thông tin.....	33
Hình 20. Đưa về dạng nhị phân.....	34
Hình 21. Dùng Apriori để tiến hành tìm tập phổ biến.....	34
Hình 22. Các luật kết hợp.....	35
Hình 23. Mã hóa các biến dạng phân loại .....	35
Hình 24. Xem ma trận tương quan.....	36
Hình 25. Chia tập dữ liệu .....	36
Hình 26. Huấn luyện tập dữ liệu .....	37
Hình 27. Chuẩn hóa dữ liệu.....	38
Hình 28. Phân cụm .....	38
Hình 29. Xây dựng lại mô hình Linear regression kết hợp với kết quả gom cụm .....	39
Hình 30. Kết quả mô hình dự đoán khi kết hợp .....	39
Hình 31. So sánh kết quả 2 phương pháp.....	40

## **DANH MỤC BẢNG BIỂU**

Bảng 1. Đánh giá mức độ hoàn thành .....	1
Bảng 2. Bảng mô tả thuộc tính .....	25

## **CHƯƠNG 1: GIỚI THIỆU**

### **1.1 Giới thiệu đề tài**

Doanh số bán hàng là một chỉ số quan trọng đối với bất kỳ doanh nghiệp nào. Nó giúp doanh nghiệp đánh giá hiệu quả hoạt động, đưa ra chiến lược kinh doanh phù hợp và dự báo nhu cầu thị trường trong tương lai. Tuy nhiên, việc dự đoán doanh số bán hàng một cách chính xác là một bài toán phức tạp, do nhiều yếu tố ảnh hưởng như xu hướng thị trường, chiến lược marketing, điều kiện kinh tế, v.v. Trong bối cảnh thị trường kinh doanh ngày càng cạnh tranh khốc liệt, việc dự đoán chính xác doanh số bán hàng trở thành một yếu tố then chốt giúp các doanh nghiệp đưa ra quyết định chiến lược, tối ưu hóa hoạt động kinh doanh và tăng cường lợi thế cạnh tranh. Đề tài "Dự đoán doanh số bán hàng dựa trên thuật toán Probabilistic Model-Based Clustering (Thuật Toán EM)" nhằm mục tiêu ứng dụng một phương pháp phân cụm tiên tiến trong học máy để phân tích và dự đoán doanh số bán hàng.

Thuật toán Expectation-Maximization (EM) là một phương pháp mạnh mẽ và hiệu quả trong việc tìm kiếm các tham số tối ưu cho mô hình xác suất, đặc biệt là trong các bài toán phân cụm dữ liệu phức tạp. Bằng cách áp dụng thuật toán này, thuật toán này không chỉ giúp nhận diện các mô hình tiềm ẩn trong dữ liệu bán hàng mà còn dự đoán chính xác xu hướng doanh số trong tương lai dựa trên các nhóm khách hàng và sản phẩm tương tự. Kết quả cung cấp các gợi ý thực tiễn quan trọng cho các doanh nghiệp trong việc quản lý và phát triển kinh doanh.

### **1.2 Nội dung thực hiện**

Khám phá bộ dữ liệu Big Mart Sales Prediction Datasets.

Xử lý các vấn đề tìm ẩn trong bộ dữ liệu ảnh hưởng đến quá trình phân tích cũng như dự đoán.

Tìm ra các luật kết hợp/ mẫu trong tập dữ liệu.

Tìm hiểu thuật toán Probabilistic Model-Based Clustering(EM) và Linear Regression.

Tìm hiểu thang đo đánh giá kết quả thuật toán hợp lý.

Cài đặt thuật toán Linear Regression để dự đoán và kiểm tra kết quả.



Cài đặt thuật toán Probabilistic Model-Based Clustering (EM) để cải thiện độ chính xác của mô hình.

Đánh giá kết quả thực hiện.

### 1.3 Giới hạn đề tài

Đề tài được giới hạn áp dụng trên tập dữ liệu Big Mart Sales Prediction Datasets. Thuật toán được dùng là Probabilistic Model-Based Clustering (Thuật Toán EM) kết hợp với Linear Regression nhằm tối ưu kết quả dự đoán của mô hình và được triển khai bằng ngôn ngữ Python.

Mục tiêu đề tài này là áp dụng thuật toán Probabilistic Model-Based Clustering (EM) để phân nhóm các yếu tố ảnh hưởng đến doanh số bán hàng. Và xây dựng mô hình dự đoán doanh số bán hàng dựa trên các nhóm đã được phân loại. Mô hình này dựa trên giả định về phân phối xác suất của tập dữ liệu, có thể không hoàn toàn chính xác trong mọi tình huống.

Việc áp dụng mô hình vào thực tế có thể gặp nhiều khó khăn do các yếu tố như mô hình dựa trên giả định về phân phối xác suất của dữ liệu, có thể không hoàn toàn chính xác trong mọi tình huống. Thuật toán EM có thể trở nên phức tạp và khó xử lý khi dữ liệu lớn hoặc có nhiều biến số.

### 1.4 Bố cục báo cáo

Bảng đánh giá mức độ hoàn thành: Nêu ra công việc của từng thành viên trong nhóm cũng như kết quả thực hiện.

CHƯƠNG 1 GIỚI THIỆU: Giới thiệu về đề tài và nội dung thực hiện.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT: Giới thiệu về các cơ sở lý thuyết các khái niệm cơ bản liên quan đến các phương pháp, mô hình, cũng như thuật toán có liên quan trong đề tài. Giới thiệu về một số công trình liên quan đến đề tài dự đoán doanh số bán hàng dựa trên thuật toán EM.

CHƯƠNG 3 DỰ ĐOÁN DOANH SỐ BÁN HÀNG DỰA TRÊN THUẬT TOÁN PROBABILISTIC MODEL-BASED CLUSTERING (THUẬT TOÁN EM): Giới thiệu về tổng quan phương pháp hiện thuật, công cụ để hiện thực, tập dữ liệu, quá trình xử lý dữ liệu cũng như triển khai đề tài.

CHƯƠNG 4 KẾT QUẢ - KẾT LUẬN: Nhận xét kết quả thực hiện nêu ra ưu cũng như nhược điểm và hướng phát triển trong tương lai.

TÀI LIỆU THAM KHẢO: Giới thiệu về các tài liệu mà nhóm đã tham khảo trong quá trình thực hiện đề tài.

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1 Các khái niệm cơ bản

#### 2.1.1 Tứ phân vị xử lý giá trị ngoại biên

Trong phân tích dữ liệu, ngoại lệ (outliers) là các giá trị dữ liệu nằm xa phần lớn các giá trị khác trong tập dữ liệu. Các ngoại lệ này có thể gây ra ảnh hưởng tiêu cực đến kết quả phân tích, đặc biệt là trong các mô hình thống kê và học máy. Do đó, việc xử lý ngoại lệ là một bước quan trọng trong quy trình tiền xử lý dữ liệu. Một trong những phương pháp phổ biến để xác định và xử lý ngoại lệ là sử dụng tứ phân vị.

Tứ phân vị chia tập dữ liệu thành bốn phần bằng nhau. Các tứ phân vị chính bao gồm:

- Q1 (Quartile 1): Giá trị ở vị trí 25% của dữ liệu, hay còn gọi là phân vị thứ nhất.
- Q2 (Quartile 2): Giá trị ở vị trí 50% của dữ liệu, tương đương với trung vị (median).
- Q3 (Quartile 3): Giá trị ở vị trí 75% của dữ liệu, hay còn gọi là phân vị thứ ba.
- IQR (Interquartile Range): Khoảng tứ phân vị, được tính bằng  $Q3 - Q1$ , là phạm vi chứa 50% giá trị giữa Q1 và Q3.

Ngoại lệ thường được xác định dựa trên khoảng tứ phân vị (IQR). Các bước để xác định ngoại lệ như sau:

- Tính Q1 và Q3 của tập dữ liệu.
- Tính IQR:  $IQR = Q3 - Q1$ .
- Xác định giới hạn dưới và giới hạn trên:
  - o Giới hạn dưới:  $Q1 - 1.5 * IQR$ .
  - o Giới hạn trên:  $Q3 + 1.5 * IQR$ .
- Các giá trị nằm ngoài khoảng giới hạn trên và dưới này được coi là ngoại lệ.

Xử lý ngoại lệ có thể áp dụng các cách như: Loại bỏ, thay thế, Biến đổi dữ liệu,...

### 2.1.2 Các tập mục thường xuyên và luật kết hợp

Các tập mục thường xuyên, còn được gọi là luật kết hợp, là một khái niệm cơ bản trong khai phá luật kết hợp, một kỹ thuật trong khai thác dữ liệu nhằm khám phá mối quan hệ giữa các mục trong một tập dữ liệu. Mục tiêu của khai phá luật kết hợp là xác định các mối quan hệ giữa các mục trong một tập dữ liệu mà thường xuyên xuất hiện cùng nhau.

Một tập mục thường xuyên là một tập hợp các mục thường xuyên xuất hiện cùng nhau trong một tập dữ liệu. Tần suất của một tập mục được đo bằng số lượng hỗ trợ (support count), đó là số lượng giao dịch hoặc bản ghi trong tập dữ liệu chứa tập mục đó. Ví dụ, nếu một tập dữ liệu chứa 100 giao dịch và tập mục {sữa, bánh mì} xuất hiện trong 20 giao dịch, thì số lượng hỗ trợ cho {sữa, bánh mì} là 20.

Các thuật toán khai phá luật kết hợp, chẳng hạn như Apriori hoặc FP-Growth, được sử dụng để tìm các tập mục thường xuyên và tạo ra các luật kết hợp. Các thuật toán này hoạt động bằng cách lặp lại việc tạo ra các tập mục ứng viên và loại bỏ những tập không đáp ứng ngưỡng hỗ trợ tối thiểu. Khi các tập mục thường xuyên đã được tìm thấy, các luật kết hợp có thể được tạo ra bằng cách sử dụng khái niệm độ tin cậy (confidence), là tỷ lệ giữa số lượng giao dịch chứa tập mục đó và số lượng giao dịch chứa điều kiện (bên trái) của luật.

Các tập mục thường xuyên và luật kết hợp có thể được sử dụng cho nhiều nhiệm vụ khác nhau như phân tích giỏ hàng, bán chéo và hệ thống gợi ý. Tuy nhiên, cần lưu ý rằng khai phá luật kết hợp có thể tạo ra một số lượng lớn các luật, nhiều trong số đó có thể không liên quan hoặc không thú vị. Do đó, việc sử dụng các biện pháp thích hợp như lift và conviction để đánh giá mức độ thú vị của các luật được tạo ra là rất quan trọng.

Khai phá luật kết hợp tìm kiếm các mục thường xuyên trong tập dữ liệu. Trong khai phá thường xuyên, thường tìm thấy các mối liên kết và tương quan thú vị giữa các tập mục trong các cơ sở dữ liệu giao dịch và quan hệ. Nói ngắn gọn, khai phá thường xuyên cho thấy những mục nào xuất hiện cùng nhau trong một giao dịch hoặc mối quan hệ.

### 2.1.3 Thuật toán Apriori

Trong lĩnh vực khai phá dữ liệu, thuật toán Apriori xây dựng nền tảng dựa trên một số khái niệm cơ bản, quan trọng để hiểu rõ cách thức hoạt động và ứng dụng của nó:

- Tập hợp item: Một nhóm các mục hoặc sản phẩm trong cơ sở dữ liệu. Ví dụ, trong cơ sở dữ liệu bán lẻ, một tập hợp item có thể bao gồm các sản phẩm như bánh mì, sữa, và bơ.
- Tập hợp phổ biến (Frequent Itemset): Tập hợp các item xuất hiện cùng nhau trong cơ sở dữ liệu với tần suất vượt qua một ngưỡng nhất định (độ hỗ trợ - support). Tập hợp này giúp xác định các mẫu mua sắm hoặc xu hướng sử dụng sản phẩm chung giữa các giao dịch.
- Luật kết hợp (Association Rule): Một quy tắc thể hiện mối quan hệ giữa hai sự kiện, sản phẩm hoặc tập hợp item, dưới dạng  $A \Rightarrow B$ , nơi A và B là các tập hợp item. Luật này chỉ ra rằng khi A xuất hiện, B cũng có khả năng xuất hiện.
- Độ hỗ trợ (Support): Tỷ lệ phần trăm của tất cả giao dịch chứa một tập hợp item nhất định so với tổng số giao dịch, giúp xác định mức độ phổ biến của một tập hợp item.
- Độ tin cậy (Confidence): Tỷ lệ phần trăm các giao dịch chứa tập hợp item A cũng chứa item B, đo lường mức độ đáng tin cậy của một luật kết hợp.

Nguyên lý hoạt động của thuật toán Apriori dựa trên việc tìm ra tất cả các tập hợp item phổ biến trong cơ sở dữ liệu giao dịch. Thuật toán bắt đầu bằng việc tính toán độ hỗ trợ của từng item riêng lẻ và loại bỏ những item có độ hỗ trợ dưới ngưỡng đã định. Sau đó, nó kết hợp các item còn lại thành tập hợp 2 item, tính toán độ hỗ trợ của từng tập hợp và loại bỏ các tập hợp không đạt ngưỡng. Quá trình này lặp lại với các tập hợp lớn hơn cho đến khi không còn tìm được tập hợp item phổ biến mới. Từ các tập hợp item phổ biến, thuật toán sinh ra các luật kết hợp dựa trên độ tin cậy, giúp tiết lộ các mối quan hệ giữa các item trong cơ sở dữ liệu.

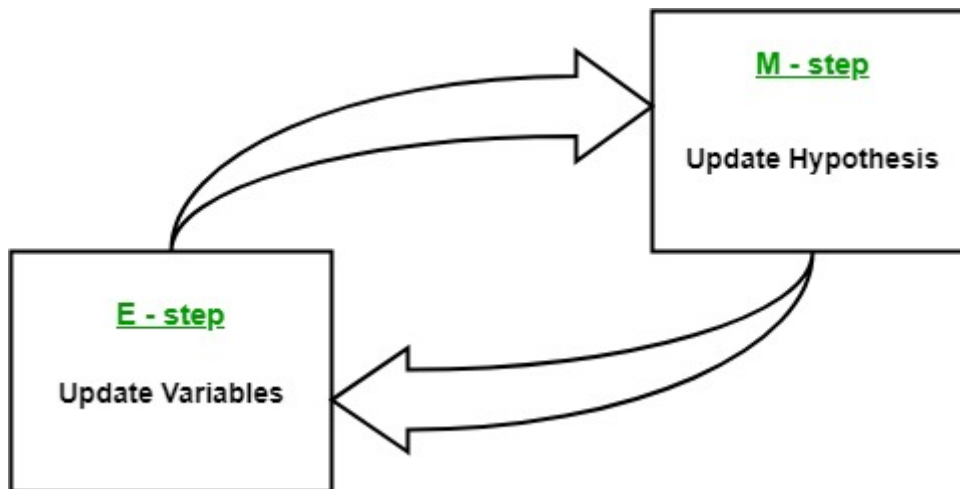
Các bước cụ thể của thuật toán:

- Bước 1: Xác định Tập Hợp Item Phổ Biến: Tính độ hỗ trợ cho tất cả các item riêng lẻ trong cơ sở dữ liệu và loại bỏ các item có độ hỗ trợ dưới ngưỡng  $\text{min\_support}$ . Lặp lại việc kết hợp các item còn lại thành tập hợp item có kích thước lớn hơn và tính độ hỗ trợ cho từng tập hợp. Loại bỏ những tập hợp không đạt ngưỡng  $\text{min\_support}$ . Quá trình này tiếp tục cho đến khi không thể tạo ra thêm tập hợp item phổ biến mới.
- Bước 2: Sinh Ra Các Luật Kết Hợp: Từ các tập hợp item phổ biến đã tìm được, sinh ra các luật kết hợp có dạng  $A \Rightarrow B$ , nơi A và B là các tập hợp item không giao nhau. Tính độ tin cậy (confidence) cho mỗi luật. Nếu độ tin cậy của luật dưới ngưỡng  $\text{min\_confidence}$ , luật đó sẽ bị loại bỏ. Lựa chọn các luật kết hợp đạt ngưỡng độ tin cậy để đưa ra kết quả.

### 2.1.4 Thuật toán EM

Thuật toán Expectation-Maximization (EM) là một phương pháp tối ưu hóa lặp đi lặp lại, kết hợp các thuật toán học máy không giám sát khác nhau để tìm các ước lượng hợp lý cực đại hoặc hậu nghiệm cực đại của các tham số trong các mô hình thống kê có liên quan đến các biến ẩn không quan sát được. Thuật toán EM thường được sử dụng cho các mô hình biến ẩn và có thể xử lý dữ liệu bị thiếu. Nó bao gồm một bước ước lượng (E-step) và một bước tối đa hóa (M-step), hình thành một quá trình lặp đi lặp lại để cải thiện độ phù hợp của mô hình. EM + mô hình phân loại:

- Bước E (Expectation Step): Trong bước này, thuật toán tính toán các biến ẩn, tức là kỳ vọng của log-likelihood bằng cách sử dụng các ước lượng tham số hiện tại.
- Bước M (Maximization Step): Trong bước này, thuật toán xác định các tham số tối đa hóa log-likelihood kỳ vọng thu được ở bước E, và các tham số mô hình tương ứng được cập nhật dựa trên các biến ẩn ước lượng.



Hình 1. Quá trình lặp đi lặp lại của EM để cải thiện độ phù hợp của mô hình

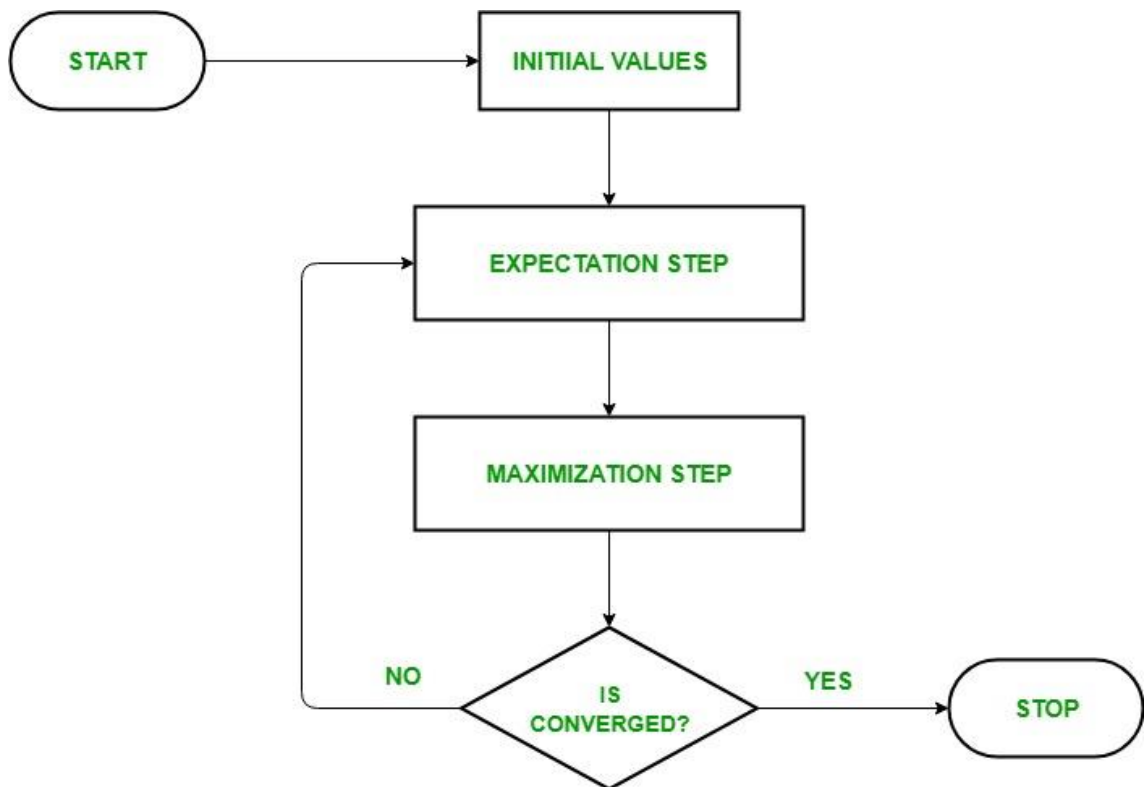
Bằng cách lặp lại các bước này, thuật toán EM tìm cách tối đa hóa khả năng của dữ liệu quan sát được. Nó thường được sử dụng cho các nhiệm vụ học không giám sát, như phân cụm, nơi các biến ẩn được suy luận và có ứng dụng trong nhiều lĩnh vực khác nhau, bao gồm học máy, thị giác máy tính, và xử lý ngôn ngữ tự nhiên.

Các thuật ngữ chính trong thuật toán Expectation-Maximization (EM):

- **Biến Ẩn (Latent Variables):** Biến ẩn là các biến không quan sát được trong các mô hình thống kê mà chỉ có thể được suy luận gián tiếp thông qua các tác động của chúng lên các biến quan sát được. Chúng không thể đo lường trực tiếp nhưng có thể được phát hiện qua tác động của chúng lên các biến quan sát.
- **Khả Năng (Likelihood):** Đây là xác suất của việc quan sát được dữ liệu đã cho dựa trên các tham số của mô hình. Trong thuật toán EM, mục tiêu là tìm các tham số tối đa hóa khả năng.
- **Log-Likelihood:** Đây là logarit của hàm likelihood, đo lường độ phù hợp giữa dữ liệu quan sát và mô hình. Thuật toán EM tìm cách tối đa hóa log-likelihood.
- **Ước Lượng Hợp Lý Cực Đại (Maximum Likelihood Estimation - MLE):** MLE là một phương pháp ước lượng các tham số của một mô hình thống kê bằng cách tìm các giá trị tham số tối đa hóa hàm likelihood, đo lường mức độ giải thích dữ liệu quan sát của mô hình.
- **Xác Suất Hậu Nghiệm (Posterior Probability):** Trong bối cảnh suy luận Bayesian, thuật toán EM có thể được mở rộng để ước lượng các ước lượng hậu nghiệm cực đại (MAP), trong đó xác suất hậu nghiệm của các tham số được tính toán dựa trên phân phối tiên nghiệm và hàm likelihood.
- **Hội Tụ (Convergence):** Hội tụ đề cập đến điều kiện khi thuật toán EM đã đạt được một giải pháp ổn định. Nó thường được xác định bằng cách kiểm tra nếu sự thay đổi trong log-likelihood hoặc các ước lượng tham số giảm xuống dưới một ngưỡng định trước.

Cách thức hoạt động của thuật toán Expectation-Maximization (EM): Bản chất của thuật toán Expectation-Maximization là sử dụng dữ liệu quan sát có sẵn của tập dữ liệu để ước tính dữ liệu bị thiếu và sau đó sử dụng dữ liệu đó để cập nhật giá trị của các tham số.





Hình 2. Cách thức hoạt động của thuật toán Expectation-Maximization (EM)

- Khởi Tạo (Initialization): Ban đầu, một tập giá trị khởi tạo của các tham số được xem xét. Một tập dữ liệu quan sát không đầy đủ được đưa vào hệ thống với giả định rằng dữ liệu quan sát đến từ một mô hình cụ thể.
- Bước E (Expectation Step): Trong bước này, chúng ta sử dụng dữ liệu quan sát để ước lượng hoặc đoán giá trị của các dữ liệu thiếu hoặc không hoàn chỉnh. Nó chủ yếu được sử dụng để cập nhật các biến. Tính toán xác suất hậu nghiệm hoặc trách nhiệm của mỗi biến ẩn dựa trên dữ liệu quan sát và ước lượng tham số hiện tại. Ước lượng giá trị của các dữ liệu thiếu hoặc không hoàn chỉnh bằng cách sử dụng các ước lượng tham số hiện tại. Tính toán log-likelihood của dữ liệu quan sát dựa trên các ước lượng tham số hiện tại và dữ liệu thiếu ước lượng.
- Bước M (Maximization Step): Trong bước này, chúng ta sử dụng dữ liệu hoàn chỉnh được tạo ra trong bước "Expectation" trước đó để cập nhật giá trị của các tham số. Nó chủ yếu được sử dụng để cập nhật giả thuyết. Cập nhật các tham số của mô hình bằng cách tối đa hóa log-likelihood của dữ liệu hoàn chỉnh thu được từ bước E. Điều này thường liên quan đến việc giải quyết các bài toán tối

ưu hóa để tìm các giá trị tham số tối đa hóa log-likelihood. Kỹ thuật tối ưu hóa cụ thể được sử dụng phụ thuộc vào tính chất của vấn đề và mô hình được sử dụng.

- Hội Tụ (Convergence): Trong bước này, kiểm tra xem các giá trị có hội tụ hay không, nếu có, thì dừng lại, nếu không, lặp lại bước E và bước M cho đến khi hội tụ xảy ra. Kiểm tra hội tụ bằng cách so sánh sự thay đổi trong log-likelihood hoặc các giá trị tham số giữa các lần lặp. Nếu sự thay đổi dưới ngưỡng định trước, dừng lại và xem xét thuật toán đã hội tụ. Nếu không, quay lại bước E và lặp lại quá trình cho đến khi đạt được hội tụ.

### 2.1.5 Hồi quy tuyến tính (Linear Regression)

Hồi quy tuyến tính là một loại thuật toán học máy có giám sát, tính toán mối quan hệ tuyến tính giữa biến phụ thuộc và một hoặc nhiều đặc trưng độc lập bằng cách khớp một phương trình tuyến tính với dữ liệu quan sát.

Khi chỉ có một đặc trưng độc lập, nó được gọi là Hồi quy tuyến tính đơn, và khi có nhiều hơn một đặc trưng, nó được gọi là Hồi quy tuyến tính đa biến.

Tương tự, khi chỉ có một biến phụ thuộc, nó được coi là Hồi quy tuyến tính đơn biến, trong khi khi có nhiều hơn một biến phụ thuộc, nó được gọi là Hồi quy tuyến tính đa biến.

Tính giải thích của hồi quy tuyến tính là một điểm mạnh đáng chú ý. Phương trình của mô hình cung cấp các hệ số rõ ràng, giúp làm sáng tỏ tác động của mỗi biến độc lập lên biến phụ thuộc, tạo điều kiện hiểu sâu hơn về các động lực cơ bản. Đơn giản của nó là một ưu điểm, vì hồi quy tuyến tính minh bạch, dễ triển khai và phục vụ như một khái niệm nền tảng cho các thuật toán phức tạp hơn.

Hồi quy tuyến tính không chỉ là một công cụ dự đoán; nó là cơ sở cho nhiều mô hình tiên tiến. Các kỹ thuật như chuẩn hóa và máy vector hỗ trợ (SVM) lấy cảm hứng từ hồi quy tuyến tính, mở rộng tính hữu dụng của nó. Ngoài ra, hồi quy tuyến tính là một nền tảng trong kiểm tra giả định, cho phép các nhà nghiên cứu xác thực các giả định chính về dữ liệu.

### 2.1.6 Các thang đo

**MSE (Mean Squared Error):** Đây là một chỉ số được sử dụng rộng rãi trong thống kê và học máy để đo lường độ chính xác của một mô hình dự đoán. Nó tính toán trung bình của bình phương các sai số giữa giá trị thực tế và giá trị dự đoán của mô hình. MSE càng nhỏ, mô hình dự đoán càng chính xác. Tuy nhiên, vì MSE tính theo bình phương sai số, nó đặc biệt nhạy cảm với các giá trị ngoại biên.

**MAE (Mean Absolute Error):** Đây là một chỉ số đo lường mức độ chính xác của một mô hình dự đoán, tương tự như MSE, nhưng thay vì sử dụng bình phương sai số, MAE tính toán trung bình của các giá trị tuyệt đối của sai số. MAE cung cấp một cách đo lường trực quan hơn về độ chính xác của mô hình, vì nó không bị ảnh hưởng nhiều bởi các giá trị ngoại lai như MSE. Mức độ sai số trung bình được biểu diễn bằng cùng đơn vị với dữ liệu gốc, điều này giúp dễ dàng hơn trong việc hiểu và diễn giải kết quả.

**RMSE (Root Mean Squared Error):** Đây là một chỉ số đo lường độ chính xác của mô hình dự đoán bằng cách tính căn bậc hai của MSE. RMSE cung cấp một thước đo độ lệch trung bình của dự đoán so với giá trị thực tế, có cùng đơn vị với dữ liệu gốc. RMSE là một chỉ số rất hữu ích vì nó phóng đại các lỗi lớn hơn do bình phương các sai số. Điều này có nghĩa là RMSE sẽ đặc biệt nhạy cảm với các giá trị ngoại biên, và do đó, nó cung cấp một bức tranh rõ ràng hơn về mức độ ảnh hưởng của các dự đoán sai lệch lớn đến độ chính xác tổng thể của mô hình. RMSE càng nhỏ, mô hình dự đoán càng chính xác.

**R<sup>2</sup> Score (R-squared):** Đây là một chỉ số thống kê đo lường mức độ phù hợp của mô hình dự đoán tuyến tính với dữ liệu thực tế. Nó cho biết tỷ lệ phần trăm biến động của biến phụ thuộc được giải thích bởi các biến độc lập trong mô hình.

$R^2$  nằm trong khoảng từ 0 đến 1. Nếu  $R^2 = 1$  thì mô hình dự đoán hoàn toàn chính xác các giá trị thực tế. Nếu  $R^2 = 0$  thì mô hình không giải thích được bất kỳ biến động nào của biến phụ thuộc so với giá trị trung bình. Nếu  $R^2 < 0$  mô hình dự đoán tồi hơn so với việc chỉ sử dụng giá trị trung bình của dữ liệu thực tế.

$R^2$  giúp đánh giá độ phù hợp của mô hình tuyến tính, nhưng không thể hiện rõ ràng mức độ chính xác của các dự đoán cá nhân. Do đó, nó thường được sử dụng cùng

với các chỉ số khác như MSE, MAE hoặc RMSE để có cái nhìn toàn diện hơn về hiệu suất của mô hình.

### 2.1.7 Gaussian Mixture Model

Gaussian Mixture Model là một mô hình phân cụm thuộc lớp bài toán học không giám sát mà phân phối xác suất của mỗi một cụm được giả định là phân phối Gaussian đa chiều. Sở dĩ mô hình được gọi là Mixture là vì xác suất của mỗi điểm dữ liệu không chỉ phụ thuộc vào một phân phối Gaussian duy nhất mà là kết hợp từ nhiều phân phối Gaussian khác nhau từ mỗi cụm.

Mục tiêu của mô hình GMM là ước lượng tham số phù hợp nhất cho  $k$  cụm thông qua phương pháp ước lượng hợp lý tối đa. Một số giả định của mô hình GMM:

- Có  $k$  cụm cần phân chia mà mỗi cụm tuân theo phân phối Gaussian đa chiều với tập tham số đặc trưng  $\{(\mu_i, \Sigma_i)\}_{i=1}^k$ .
- $z_k$  được giả định là một biến ngẫu nhiên nhận giá trị 1 nếu như quan sát  $x$  rơi vào cụm thứ  $k$  các trường hợp còn lại nhận giá trị 0.
- $z_k$  được coi như là một biến ẩn (latent variable hoặc hidden variable) mà ta chưa biết giá trị của nó. Xác suất xảy ra của  $p(z_k = 1|x)$  giúp chúng ta xác định tham số phân phối của Gaussian Mixture.

Tập hợp các giá trị của  $z_k$  đối với các cụm sẽ tạo thành một phân phối xác suất sẽ tạo thành một phân phối xác suất  $(\pi_1, \pi_2, \dots, \pi_k)$  trong đó  $\pi_k = p(z_k = 1|x)$ .

Một xác suất hỗn hợp tại một điểm dữ liệu  $x$  sẽ được tính theo công thức Bayes như sau:

$$\begin{aligned} p(x) &= \sum_{c=1}^k p(z_c)p(x|z_c) \\ &= \sum_{c=1}^k p(z_c = 1)p(x|\mu_c, \Sigma_c) \\ &= \sum_{c=1}^k \pi_c p(x|\mu_c, \Sigma_c) \\ &= \sum_{c=1}^k \pi_c N(x|\mu_c, \Sigma_c) \end{aligned}$$

Thành phần xác suất  $p(x|\mu_i, \Sigma_i)$  được tính từ phân phối Gaussian đa chiều và chúng đồng thời là mục tiêu mà chúng ta cần tham số hoá.

### 2.1.8 Label Encoder

Label Encoder là một công cụ trong học máy dùng để chuyển đổi các nhãn phân loại thành các số nguyên. Là một quá trình quan trọng trong việc tiền xử lý dữ liệu để chuẩn bị cho các mô hình học máy yêu cầu dữ liệu số. Label Encoder dễ triển khai và hiểu hữu ích cho việc xử lý các biến phân loại trước khi đưa vào các mô hình học máy. Label Encoder không bảo toàn thứ tự của các nhãn phân loại và không phù hợp với các thuật toán nhạy cảm với thứ tự của dữ liệu. Nếu mô hình sử dụng khoảng cách giữa các giá trị số như KNN, hồi quy tuyến tính, việc sử dụng Label Encoder có thể không thích hợp vì nó tạo ra ấn tượng sai lệch về thứ tự và khoảng cách giữa các nhãn.

### 2.1.9 Standard Scaler

Standard Scaler là một công cụ trong tiền xử lý dữ liệu, sử dụng để chuẩn hóa các đặc trưng bằng cách loại bỏ trung bình và chia tỷ lệ theo độ lệch chuẩn.

Standard Scaler đảm bảo dữ liệu có cùng thang đo giúp các mô hình học máy hội tụ nhanh hơn và cải thiện độ chính xác. Không bị ảnh hưởng bởi đơn vị của dữ liệu giúp so sánh các đặc trưng có đơn vị khác nhau. Giảm ảnh hưởng của các đặc trưng có giá trị lớn đặc biệt hữu ích khi làm việc với mô hình nhạy cảm với tỷ lệ, như hồi quy tuyến tính hoặc SVM.

Standard Scaler không bảo toàn thông tin gốc: Việc chuẩn hóa có thể làm mất thông tin về biên độ của đặc trưng và không phù hợp với dữ liệu có phân phối lệch. Đối với dữ liệu có phân phối không chuẩn, cần cân nhắc trước khi sử dụng.

## 2.2 Các công trình liên quan

### A Two-Level Statistical Model for Big Mart Sales Prediction

Bài báo này, dự đoán doanh số bán sản phẩm từ một cửa hàng cụ thể được thực hiện thông qua phương pháp tiếp cận hai cấp độ mang lại hiệu suất dự đoán tốt hơn so với bất kỳ thuật toán học dự đoán mô hình đơn phổ biến nào.

Data exploration, data transformation and feature engineering đóng vai trò quan trọng trong việc dự đoán kết quả chính xác. Kết quả đã chứng minh rằng phương pháp Two-Level Statistical hoạt động tốt hơn phương pháp tiếp cận mô hình đơn lẻ vì phương pháp này cung cấp nhiều thông tin hơn dẫn đến dự đoán tốt hơn.

## A Comparative Study of Big Mart Sales Prediction

Trong bài báo này, tác giả đề xuất một mô hình dự đoán sử dụng kỹ thuật Xgboost để dự đoán doanh số bán hàng của một công ty như Big Mart và nhận thấy rằng mô hình này mang lại hiệu suất tốt hơn so với các mô hình hiện có.

Phân tích so sánh mô hình với các mô hình khác về mặt số liệu hiệu suất cũng được giải thích chi tiết.

## Grid Search Optimization (GSO) Based Future Sales Prediction for Big Mart

Trong bài báo này, tác giả đề xuất kỹ thuật Grid Search Optimization (GSO) để tối ưu hóa các tham số và chọn các siêu tham số điều chỉnh tốt nhất, đồng thời kết hợp thêm với các kỹ thuật Xgboost để dự báo doanh số bán hàng trong tương lai của một công ty bán lẻ như Big Mart và họ nhận thấy mô hình của họ tạo ra kết quả tốt hơn.

## Sales Data Analysis and Prediction System for Big Mart using Deep Recurrent Reinforcement Principles

Trong bài báo này, hệ thống dự đoán doanh số bán hàng của siêu thị lớn được đề xuất phát triển Multivariate Poisson Distribution Scheme, Deep RL Engines and Recurrent Multi-Level Generative Adversarial Network để cung cấp kết quả chính xác hơn.

Các thử nghiệm và kết quả cho thấy hiệu suất của mô hình dự đoán bán hàng dựa trên mô hình deep learning được đề xuất so với các hệ thống khác. Nó cho thấy phương pháp deep learning được đề xuất hoạt động tốt hơn từ 5% đến 10% so với các mô hình dự đoán doanh số hiện có khác.

## Big Mart Sales Prediction Using Machine Learning

Bài báo này đề xuất nhiều vấn đề để dự đoán doanh thu của Big Mart và cách dự đoán bằng cách sử dụng năm loại thuật toán hồi quy khác nhau để dự báo nó, bao gồm XGBoost regressor, linear regression, decision trees, random forest regressor, và grid search CV.

Các thuật toán này được sử dụng để xây dựng các mô hình, so sánh độ chính xác của các mô hình đó và chuẩn bị cho chúng phù hợp với các giả định của hội đồng quản trị để có thể thực hiện các hành động thận trọng nhằm hoàn thành mục đích của liên kết.

Trong nghiên cứu này, có rất nhiều thách thức trong việc triển khai các mô hình thực tế. Nhiều mô hình khác nhau được tạo và sau khi so sánh R2 score của chúng, R2 score tốt nhất là random forest grid search với 0,5588858290914282. Bằng cách nào đó, R2 score thấp, nhưng đó là do tính biến đổi và thiếu dữ liệu nên nó đủ tốt để dự đoán đầu ra.

## **CHƯƠNG 3: DỰ ĐOÁN DOANH SỐ BÁN HÀNG DỰA TRÊN THUẬT TOÁN PROBABILISTIC MODEL-BASED CLUSTERING (THUẬT TOÁN EM)**

### **3.1 Tổng quan phương pháp hiện thực**

Phương pháp hiện thực đề tài này của nhóm chúng em được thực hiện lần lượt theo các bước sau:

- Tìm tập dữ liệu phù hợp với việc dự đoán doanh số bán hàng là một tập dữ liệu bán hàng nào đó của trung tâm thương mại hay siêu thị,...
- Sau khi tìm được tập dữ liệu tiến hành xử lý các vấn đề của tập dữ liệu.
- Sử dụng các luật kết hợp để tìm ra các tập phổ biến và hiểu thêm về tập dữ liệu chúng em đang thực hiện.
- Tiến hành tìm hiểu về thuật toán Probabilistic Model-Based Clustering (Thuật Toán EM) cũng như các thuật toán liên quan sau đó là cài đặt.
- Tìm hiểu các thang đo để đánh giá kết quả hiện thực.
- Tìm hiểu các phương pháp để tiến hành cải thiện hiệu suất.
- Đánh giá kết quả cũng như quá trình thực hiện toàn bộ bài làm.

### **3.2 Công cụ hiện thực**

Đề tài này được thực hiện trên ứng dụng google colab thông qua một trình duyệt web bất kì được dùng trên laptop. Ngôn ngữ lập trình được dùng trong đề tài này là python và các thuật toán được hiện thực trong bài làm đa phần đều sử dụng thư viện.



### 3.3 Tập dữ liệu

#### 3.3.1 Thông tin tập dữ liệu

- Tên: Big Mart Sales Prediction Datasets.
- Nguồn: kaggle.
- Người đăng tải: SHIVAN KUMAR
- Link data: <https://www.kaggle.com/datasets/shivan118/big-mart-sales-prediction-datasets>
- File dữ liệu (gồm 3 file) : train.csv, test.csv, sample\_submission.csv.
- Kích thước:
- train.csv gồm 12 cột và 8,523 dòng.
- test.csv gồm 11 cột và 5,681 dòng.
- sample\_submission.csv gồm 3 cột và 5,681 dòng.

#### 3.3.2 Chi tiết bộ dữ liệu

- Chúng em kiểm tra bộ dữ liệu bằng python với thư viện numpy.
- Đọc data

```
[3] # Đọc các file dataset
train = pd.read_csv("/content/drive/MyDrive/DataMining/data/train.csv")
test = pd.read_csv("/content/drive/MyDrive/DataMining/data/test.csv")
sample = pd.read_csv("/content/drive/MyDrive/DataMining/data/sample_submission.csv")
```

*Hình 3. Đọc dữ liệu*

- Hiện thị thông tin mẫu các tập giúp ta quan sát bao quát dữ liệu

train.head(5)

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDM15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

test.head(5)

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	Tier 2	Supermarket Type1
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	Tier 3	Grocery Store
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	Tier 2	Supermarket Type1
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	Tier 3	Supermarket Type3

[4] sample.head(5)

	Item_Identifier	Outlet_Identifier	Item_Outlet_Sales
0	FDW58	OUT049	1000
1	FDW14	OUT017	1000
2	NCN55	OUT010	1000
3	FDQ58	OUT017	1000
4	FDY38	OUT027	1000

Hình 4. Hiển thị 5 dòng đầu

## - Xem thông tin của tập dữ liệu

train.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Item_Identifier        8523 non-null   object
1   Item_Weight            7060 non-null   float64
2   Item_Fat_Content       8523 non-null   object
3   Item_Visibility        8523 non-null   float64
4   Item_Type              8523 non-null   object
5   Item_MRP               8523 non-null   float64
6   Outlet_Identifier       8523 non-null   object
7   Outlet_Establishment_Year 8523 non-null   int64
8   Outlet_Size            6113 non-null   object
9   Outlet_Location_Type   8523 non-null   object
10  Outlet_Type            8523 non-null   object
11  Item_Outlet_Sales      8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB

```

test.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5681 entries, 0 to 5680
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Item_Identifier        5681 non-null   object
1   Item_Weight            4705 non-null   float64
2   Item_Fat_Content       5681 non-null   object
3   Item_Visibility        5681 non-null   float64
4   Item_Type              5681 non-null   object
5   Item_MRP               5681 non-null   float64
6   Outlet_Identifier       5681 non-null   object
7   Outlet_Establishment_Year 5681 non-null   int64
8   Outlet_Size            4075 non-null   object
9   Outlet_Location_Type   5681 non-null   object
10  Outlet_Type            5681 non-null   object
dtypes: float64(3), int64(1), object(7)
memory usage: 488.3+ KB

```

[5] sample.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5681 entries, 0 to 5680
Data columns (total 3 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Item_Identifier        5681 non-null   object
1   Outlet_Identifier       5681 non-null   object
2   Item_Outlet_Sales      5681 non-null   int64
dtypes: int64(1), object(2)

```


Hình 5. Xem thông tin của tập dữ liệu

- Dựa vào thông tin trên ta có thể rút ra được thông tin chi tiết các thuộc tính trong tập dữ liệu. Và 2 tập dữ liệu train và test này đều có 2 cột chứa dữ liệu null là Item\_weight và Outlet\_Size. Còn lại tập sample\_submission không chứa giá trị null.

Stt	Thuộc tính	KDL	Ý nghĩa	Ví dụ thể hiện
1	Item_Identifier	object	Mã định danh của sản phẩm.	FDA15, DRC01 , ...
2	Item_weight	float	Trọng lượng của sản phẩm.	9.3, 5.92,...
3	Item_Fat_Content	object	Hàm lượng chất béo của sản phẩm.	'Low Fat', 'Regular', 'low fat', 'LF', 'reg'
4	Item_Visibility	float	Độ hiển thị của sản phẩm.	0.016047, 0.000000, ...
5	Item_Type	object	Loại sản phẩm.	'Dairy', 'Soft Drinks', 'Meat', ...
6	Item_MRP	float	Giá bán lẻ tối đa của sản phẩm.	249.8092, 48.2692, ...
7	Outlet_Identifier	object	Mã định danh của cửa hàng.	'OUT049', 'OUT018', ...
8	Outlet_Establishment_Year	int	Năm thành lập của cửa hàng	1999,2009, ...
9	Outlet_Size	object	Quy mô cửa hàng.	Medium, High
10	Outlet_Location_Type	object	Loại địa điểm của cửa hàng.	Tier 1, Tier 2, ..
11	Outlet_Type	object	Loại cửa hàng.	'Supermarket Type1', 'Supermarket Type2', ...
12	Item_Outlet_Sales	float	Doanh số bán hàng của sản phẩm tại cửa hàng.	3735.138, 443.4228,...


*Bảng 2. Bảng mô tả thuộc tính*

- Tiếp theo ta tiến hành thống kê tập dữ liệu dựa vào thống kê mô tả bên dưới cho ta thấy các giá trị như số lượng, trung bình, độ lệch chuẩn, min, max, và các nguồn phân bố của các giá trị. Đặt biệt ở tập sample 5681 giá trị của Item\_Olet\_Sales dựa vào các chỉ số trên điều là 1000 có thể thấy toàn bộ giá trị của cột Item\_Olet\_Sales này điều là 1000 và ta không thể sử dụng tập dữ liệu này để kiểm tra tập test.

 `train.describe()`


	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
<b>count</b>	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
<b>mean</b>	12.857645	0.066132	140.992782	1997.831867	2181.288914
<b>std</b>	4.643456	0.051598	62.275067	8.371760	1706.499616
<b>min</b>	4.555000	0.000000	31.290000	1985.000000	33.290000
<b>25%</b>	8.773750	0.026989	93.826500	1987.000000	834.247400
<b>50%</b>	12.600000	0.053931	143.012800	1999.000000	1794.331000
<b>75%</b>	16.850000	0.094585	185.643700	2004.000000	3101.296400
<b>max</b>	21.350000	0.328391	266.888400	2009.000000	13086.964800

Hình 6. Hiển thị mô tả tập train

 `test.describe()`

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year
<b>count</b>	4705.000000	5681.000000	5681.000000	5681.000000
<b>mean</b>	12.695633	0.065684	141.023273	1997.828903
<b>std</b>	4.664849	0.051252	61.809091	8.372256
<b>min</b>	4.555000	0.000000	31.990000	1985.000000
<b>25%</b>	8.645000	0.027047	94.412000	1987.000000
<b>50%</b>	12.500000	0.054154	141.415400	1999.000000
<b>75%</b>	16.700000	0.093463	186.026600	2004.000000
<b>max</b>	21.350000	0.323637	266.588400	2009.000000

Hình 7. Hiển thị mô tả tập test

 `[6] sample.describe()`

	Item_Outlet_Sales
<b>count</b>	5681.0
<b>mean</b>	1000.0
<b>std</b>	0.0
<b>min</b>	1000.0
<b>25%</b>	1000.0
<b>50%</b>	1000.0
<b>75%</b>	1000.0
<b>max</b>	1000.0

Hình 8. Hiển thị mô tả tập sample

- Tiếp tục ta kiểm tra giá trị duy nhất của tập dữ liệu và phát hiện cột Item\_Fat\_Content có các giá trị trùng nghĩa và không thống nhất một

định dạng. Ví dụ: "low fat", "LF", "Low Fat" đều có nghĩa là ít béo và “reg”, “Regular” đều có nghĩa là thông thường. Điều này xuất hiện ở cả 2 tập train và test chúng em sẽ xử lý nó ở phần tiền xử lý dữ liệu. Ngoài phát hiện trên ta còn phát hiện cột Item\_Identifier chứa giá trị định danh nên ta sẽ xóa nó ở phần tiền xử lý dữ liệu

```
#Kiểm tra giá trị duy nhất
c_train = train.select_dtypes(include=['object'])
for c in c_train.columns:
    unique_values = c_train[c].unique()
    print(f"Cột: {c}")
    print(f"Giá trị duy nhất: {unique_values}")
    print("\n")
```

Cột: Item\_Identifier  
Giá trị duy nhất: ['FDA15' 'DRC01' 'FDN15' ... 'NCF55' 'NCW30' 'NCW05']

Cột: Item\_Fat\_Content  
Giá trị duy nhất: ['Low Fat' 'Regular' 'low fat' 'LF' 'reg']

Cột: Item\_Type  
Giá trị duy nhất: ['Dairy' 'Soft Drinks' 'Meat' 'Fruits and Vegetables' 'Household' 'Baking Goods' 'Snack Foods' 'Frozen Foods' 'Breakfast' 'Health and Hygiene' 'Hard Drinks' 'Canned' 'Breads' 'Starchy Foods' 'Others' 'Seafood']

Cột: Outlet\_Identifier  
Giá trị duy nhất: ['OUT049' 'OUT018' 'OUT010' 'OUT013' 'OUT027' 'OUT045' 'OUT017' 'OUT046' 'OUT035' 'OUT019']

Cột: Outlet\_Size  
Giá trị duy nhất: ['Medium' nan 'High' 'Small']

Cột: Outlet\_Location\_Type  
Giá trị duy nhất: ['Tier 1' 'Tier 3' 'Tier 2']

Cột: Outlet\_Type  
Giá trị duy nhất: ['Supermarket Type1' 'Supermarket Type2' 'Grocery Store' 'Supermarket Type3']

*Hình 9. Kiểm tra giá trị duy nhất*

### 3.4 Tiền xử lý dữ liệu

- Kiểm tra giá trị thiếu (Missing Values) và phát hiện xảy ra ở cột Item\_Weight và Outlet\_Size của cả 2 tập dữ liệu.

```
print(train.isnull().sum())
```

```
Item_Identifier      0
Item_Weight         1463
Item_Fat_Content      0
Item_Visibility      0
Item_Type            0
Item_MRP             0
Outlet_Identifier    0
Outlet_Establishment_Year  0
Outlet_Size         2410
Outlet_Location_Type  0
Outlet_Type          0
Item_Outlet_Sales    0
dtype: int64
```

```
print(test.isnull().sum())
```

```
Item_Identifier      0
Item_Weight          976
Item_Fat_Content      0
Item_Visibility      0
Item_Type            0
Item_MRP             0
Outlet_Identifier    0
Outlet_Establishment_Year  0
Outlet_Size         1606
Outlet_Location_Type  0
Outlet_Type          0
dtype: int64
```

*Hình 10. Kiểm tra giá trị thiếu*

- Xử lý giá trị null đối với cột Item\_Weight ta lấy trung bình của các giá trị của cột sau đó dùng giá trị đó để lấp đầy các khoản null. Đối với cột Outlet\_Size là dạng phân loại ta dùng cách lấy giá trị xuất hiện nhiều nhất của cột và dùng giá trị đó lấp đầy chỗ thiếu các xử lý đối với 2 tập điều như nhau. Sau khi xử lý xong tiến hành kiểm tra lại.

```

▶ # Xử lý giá trị null cột Item Weight
mean_item_weight=train['Item_Weight'].mean()
train['Item_Weight'].fillna(mean_item_weight, inplace=True)

mean_item_weight=test['Item_Weight'].mean()
test['Item_Weight'].fillna(mean_item_weight, inplace=True)

[ ] train['Outlet_Size'].unique()

⇒ array(['Medium', nan, 'High', 'Small'], dtype=object)

[ ] test['Outlet_Size'].unique()

⇒ array(['Medium', nan, 'Small', 'High'], dtype=object)

[ ] #In giá trị xh nhiều nhất
print('train:',train['Outlet_Size'].mode()[0])
print('test:',test['Outlet_Size'].mode()[0])

⇒ train: Medium
test: Medium

[ ] train['Outlet_Size'].fillna(train['Outlet_Size'].mode()[0], inplace=True)
test['Outlet_Size'].fillna(test['Outlet_Size'].mode()[0], inplace=True)

```

*Hình 11. Xử lý giá trị thiếu*

```

✓ 0 giây
▶ print(train.isnull().sum())

⇒ Item_Identifier      0
   Item_Weight         0
   Item_Fat_Content     0
   Item_Visibility     0
   Item_Type           0
   Item_MRP            0
   Outlet_Identifier    0
   Outlet_Establishment_Year  0
   Outlet_Size         0
   Outlet_Location_Type  0
   Outlet_Type         0
   Item_Outlet_Sales    0
dtype: int64

✓ 0 giây
[17] ▶ print(test.isnull().sum())

⇒ Item_Identifier      0
   Item_Weight         0
   Item_Fat_Content     0
   Item_Visibility     0
   Item_Type           0
   Item_MRP            0
   Outlet_Identifier    0
   Outlet_Establishment_Year  0
   Outlet_Size         0
   Outlet_Location_Type  0
   Outlet_Type         0
dtype: int64

```

Hình 12. Kiểm tra lại giá trị thiếu

- Tiếp tục ta tiến hành xử lý dữ liệu không nhất quán ta đã phát hiện trước đó bằng cách thống nhất về cùng một giá trị chuẩn là Low Fat và Regular bằng cách dùng replace().

```

[ ] print("train",train['Item_Fat_Content'].unique())
    print("test",train['Item_Fat_Content'].unique())

⇒ train ['Low Fat' 'Regular' 'low fat' 'LF' 'reg']
   test ['Low Fat' 'Regular' 'low fat' 'LF' 'reg']

[ ] #Tiến hành xử lý dữ liệu không nhất quán
    train['Item_Fat_Content'] = train['Item_Fat_Content'].replace({'LF': 'Low Fat', 'low fat': 'Low Fat', 'reg': 'Regular'})
    test['Item_Fat_Content'] = test['Item_Fat_Content'].replace({'LF': 'Low Fat', 'low fat': 'Low Fat', 'reg': 'Regular'})

▶ #Kiểm tra lại
    print("train",train['Item_Fat_Content'].value_counts())
    print("test",test['Item_Fat_Content'].value_counts())

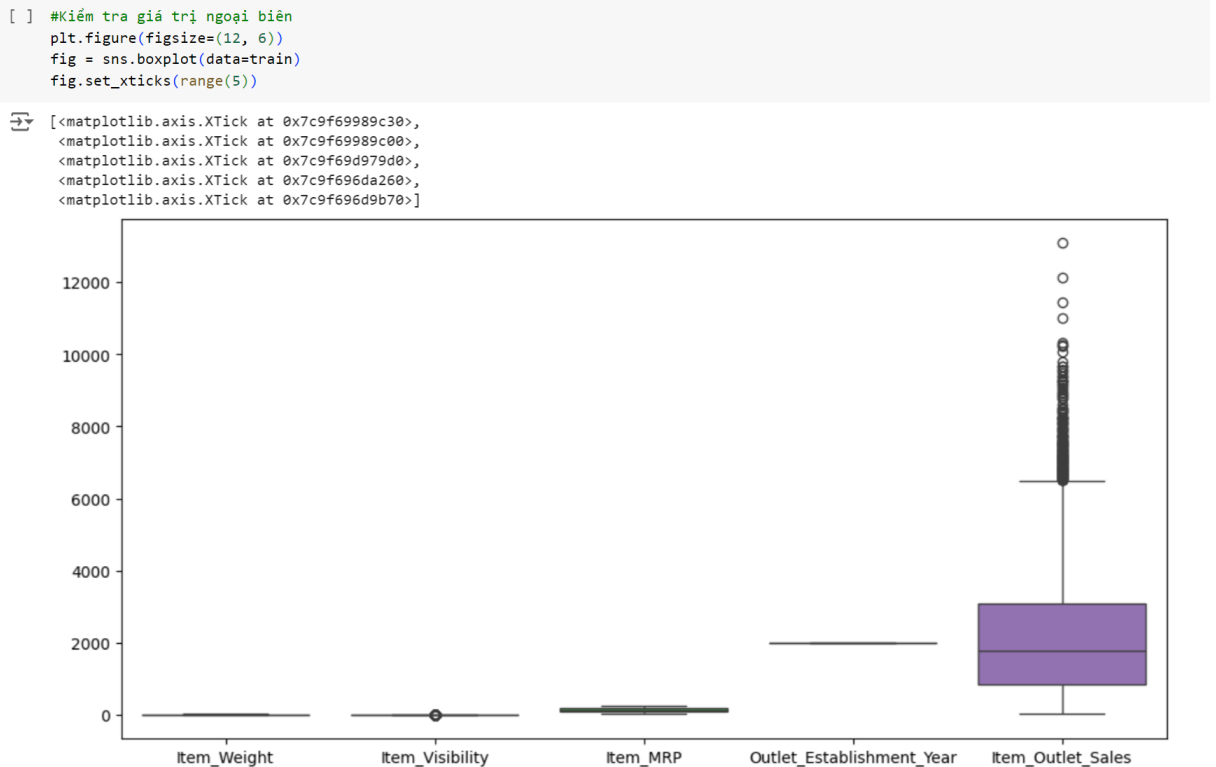
⇒ train Item_Fat_Content
   Low Fat    5517
   Regular   3006
   Name: count, dtype: int64
   test Item_Fat_Content
   Low Fat    3668
   Regular    2013
   Name: count, dtype: int64

```

Hình 13. Xử lý giá trị không nhất quán



- Dùng boxplot kiểm tra giá trị ngoại biên và phát hiện ngoại biên ở 2 cột Item\_Visibility và Item\_Outlet\_Sales ta tiến hành loại bỏ ngoại biên bằng phương pháp tứ phân vị.

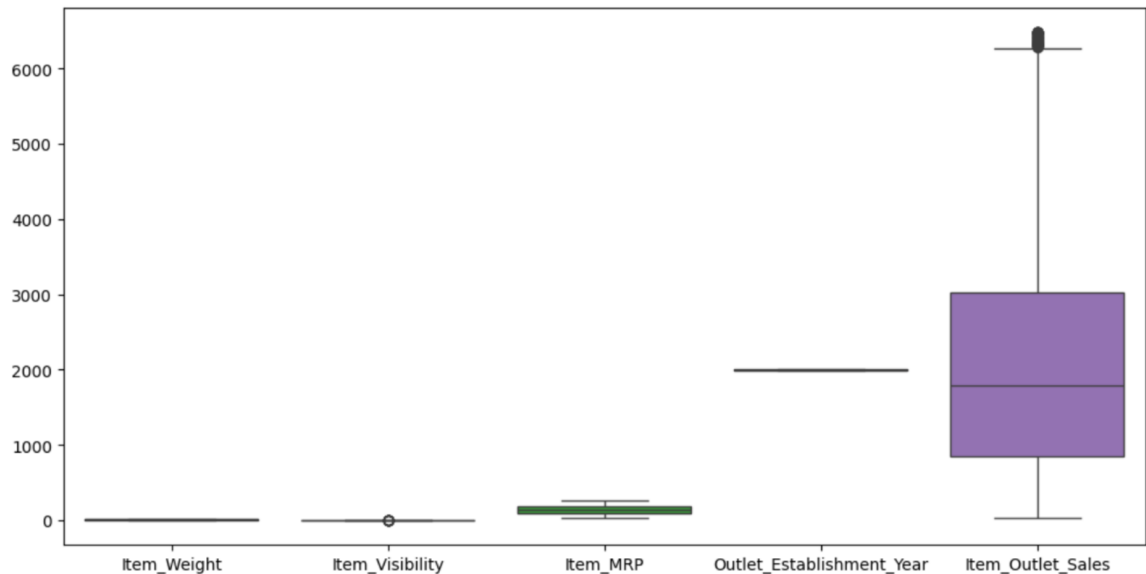


Hình 14. Kiểm tra giá trị ngoại biên

- Chi tiết quá trình thực hiện như sau: Đầu tiên lấy các cột dạng số, sau đó tính các giá trị phân vị thứ 1 và thứ 3, dùng q1 và q3 tìm được tính khoảng tứ phân vị là iqr. Sau đó xác định khoảng giới hạn dưới( $q1 - 1.5 * iqr$ ) và giới hạn trên ( $q3 + 1.5 * iqr$ ) và dùng điều kiện để loại bỏ các giá trị nằm ngoài giới hạn này. Sau quá trình trên ta kiểm tra lại có thể thấy được các giá trị ngoại biên đã được loại bỏ phần lớn và tập train sau khi loại giá trị ngoại biên còn 8,193 dòng.

```
#Kiểm tra giá trị ngoại biên
plt.figure(figsize=(12, 6))
fig = sns.boxplot(data=train)
fig.set_xticks(range(5))
```

```
[<matplotlib.axis.XTick at 0x7a847fcc0b50>,
 <matplotlib.axis.XTick at 0x7a847fcc0be0>,
 <matplotlib.axis.XTick at 0x7a847fb2e020>,
 <matplotlib.axis.XTick at 0x7a847fe13550>,
 <matplotlib.axis.XTick at 0x7a847fe11ba0>]
```



ập đúp (hoặc nhấn Enter) để chỉnh sửa

```
| train.shape
```

```
(8193, 12)
```

Hình 15. Kiểm tra giá trị ngoại biên sau xử lý

- Tiến hành xóa cột định danh và kiểm tra kết quả

```
[26] train=train.drop(['Item_Identifier'],axis=1)
      test=test.drop(['Item_Identifier'],axis=1)
```

```
print(train.columns)
print(test.columns)
```

```
Index(['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_Type',
       'Item_MRP', 'Outlet_Identifier', 'Outlet_Establishment_Year',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type',
       'Item_Outlet_Sales'],
      dtype='object')
Index(['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_Type',
       'Item_MRP', 'Outlet_Identifier', 'Outlet_Establishment_Year',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type'],
      dtype='object')
```

Hình 16. Xóa cột định danh

### 3.5 Triển khai khai thác luật kết hợp

- Ở phần này ta sẽ tiến hành khai thác luật kết hợp trên tập train và áp dụng độ hỗ trợ thấp nhất là 40% và độ tin cậy trên 50%.

```
[ ] data=train.copy()
```

Hình 17. Copy file train mới

- Khởi đầu ta sẽ rời rạc hóa các biến dạng số. Sau khi rời rạc ta kiểm tra lại kết quả

Rời rạc hóa

```
[38] #Outlet_Establishment_Year
bins = [0, 2000, float('inf')]
labels = ['Long standing', 'Recent']
data['Outlet_Establishment_Year'] = pd.cut(data['Outlet_Establishment_Year'], bins=bins, labels=labels)

#Item_Weight
bins = [0, 5, 10, float('inf')]
labels = ['Light', 'Medium', 'Heavy']
data['Item_Weight'] = pd.cut(data['Item_Weight'], bins=bins, labels=labels)

#Item_Visibility
bins = [-1, 0.05, 0.12, float('inf')]
labels = ['Low', 'Medium', 'high']
data['Item_Visibility'] = pd.cut(data['Item_Visibility'], bins=bins, labels=labels)

#Item_MRP
bins = [0, 115, 160, float('inf')]
labels = ['Low', 'Medium', 'high']
data['Item_MRP'] = pd.cut(data['Item_MRP'], bins=bins, labels=labels)

#Item_Outlet_Sales
bins = [-1, 475, 1900, 2800, 4500, float('inf')]
labels = ["Very low", 'Low', 'Medium', 'high', "Very high"]
data['Item_Outlet_Sales'] = pd.cut(data['Item_Outlet_Sales'], bins=bins, labels=labels)
```

Hình 18. Rời rạc hóa các biến dạng số

data.head()

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	Medium	Low Fat	Low	Dairy	high	OUT049	Long standing	Medium	Tier 1	Supermarket Type1	high
1	Medium	Regular	Low	Soft Drinks	Low	OUT018	Recent	Medium	Tier 3	Supermarket Type2	Very low
2	Heavy	Low Fat	Low	Meat	Medium	OUT049	Long standing	Medium	Tier 1	Supermarket Type1	Medium
3	Heavy	Regular	Low	Fruits and Vegetables	high	OUT010	Long standing	Medium	Tier 3	Grocery Store	Low
4	Medium	Low Fat	Low	Household	Low	OUT013	Long standing	High	Tier 3	Supermarket Type1	Low

Hình 19. Xem thông tin

- Dùng get\_dummies đưa dữ liệu về dạng nhị phân thuật tiện cho quá trình khai thác luật.

```
#chuyển về one hot
data = pd.get_dummies(data, columns=['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_Type',
                                     'Item_MRP', 'Outlet_Identifier', 'Outlet_Establishment_Year',
                                     'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type',
                                     'Item_Outlet_Sales'])
data.head()
```

	Item_Weight_Light	Item_Weight_Medium	Item_Weight_Heavy	Item_Fat_Content_Low Fat	Item_Fat_Content_Regular	Item_Visibility_Low	Item_Visibility_Medium	Item_Visibility_high	Item_Type_...
0	False	True	False	True	False	True	False	False	
1	False	True	False	False	True	True	True	False	
2	False	False	True	True	False	True	False	False	
3	False	False	True	False	True	True	False	False	
4	False	True	False	True	False	True	False	False	

5 rows x 54 columns

Hình 20. Đưa về dạng nhị phân

- Sau khi đưa dữ liệu về dạng nhị phân ta tiến hành dùng Apriori để tiến hành tìm tập phổ biến với độ hỗ trợ là 40%. Sau đó dùng len() kết hợp với hàm lamda tìm ra số lượng item của tập phổ biến.

```
#Tìm tập phổ biến bằng Apriori với min support là 40%
frequent_itemsets = apriori(data, min_support=0.4, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

	support	itemsets	length
0	0.711388	(Item_Weight_Heavy)	1
1	0.647915	(Item_Fat_Content_Low Fat)	1
2	0.487741	(Item_Visibility_Low)	1
3	0.550342	(Outlet_Establishment_Year_Long standing)	1
4	0.449658	(Outlet_Establishment_Year_Recent)	1
5	0.608339	(Outlet_Size_Medium)	1
6	0.673678	(Outlet_Type_Supermarket Type1)	1
7	0.401493	(Item_Outlet_Sales_Low)	1
8	0.461232	(Item_Fat_Content_Low Fat, Item_Weight_Heavy)	2
9	0.417797	(Outlet_Establishment_Year_Long standing, Item...	2
10	0.433727	(Item_Weight_Heavy, Outlet_Size_Medium)	2
11	0.444182	(Outlet_Type_Supermarket Type1, Item_Weight_He...	2
12	0.435719	(Item_Fat_Content_Low Fat, Outlet_Type_Superma...	2

Hình 21. Dùng Apriori để tiến hành tìm tập phổ biến

- Sau khi có được tập phổ biến có min support là 40% ta tiến hành khai thác luật với associantion\_rules() và độ tinh cậý được đặt ở mức 50% ta thu được 9 rules sau.

```
[34] #Khai thác luật kết hợp
rules = association_rules(frequent_itemsets, min_threshold=0.5)
rules[['antecedents', 'consequents', 'support', 'confidence']]
```

	antecedents	consequents	support	confidence
0	(Item_Fat_Content_Low Fat)	(Item_Weight_Heavy)	0.461232	0.711871
1	(Item_Weight_Heavy)	(Item_Fat_Content_Low Fat)	0.461232	0.648355
2	(Outlet_Establishment_Year_Long standing)	(Item_Weight_Heavy)	0.417797	0.759159
3	(Item_Weight_Heavy) (Outlet_Establishment_Year_Long standing)		0.417797	0.587299
4	(Item_Weight_Heavy)	(Outlet_Size_Medium)	0.433727	0.609692
5	(Outlet_Size_Medium)	(Item_Weight_Heavy)	0.433727	0.712971
6	(Outlet_Type_Supermarket Type1)	(Item_Weight_Heavy)	0.444182	0.659339
7	(Item_Weight_Heavy)	(Outlet_Type_Supermarket Type1)	0.444182	0.624388
8	(Item_Fat_Content_Low Fat)	(Outlet_Type_Supermarket Type1)	0.435719	0.672493
9	(Outlet_Type_Supermarket Type1)	(Item_Fat_Content_Low Fat)	0.435719	0.646776

Hình 22. Các luật kết hợp

### 3.6 Quá trình phân cụm, dự đoán

- Bước đầu tiên trong quá trình này ta tiến hành sử dụng label\_encoder để mã hóa các biến dạng phân loại của 2 tập train và test về dạng số.

```
[35] #Mã hóa về dạng số
for col in ['Item_Fat_Content', 'Item_Type', 'Outlet_Identifier', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type']:
    label_encoder = LabelEncoder()
    train[col] = label_encoder.fit_transform(train[col])
    test[col] = label_encoder.transform(test[col])

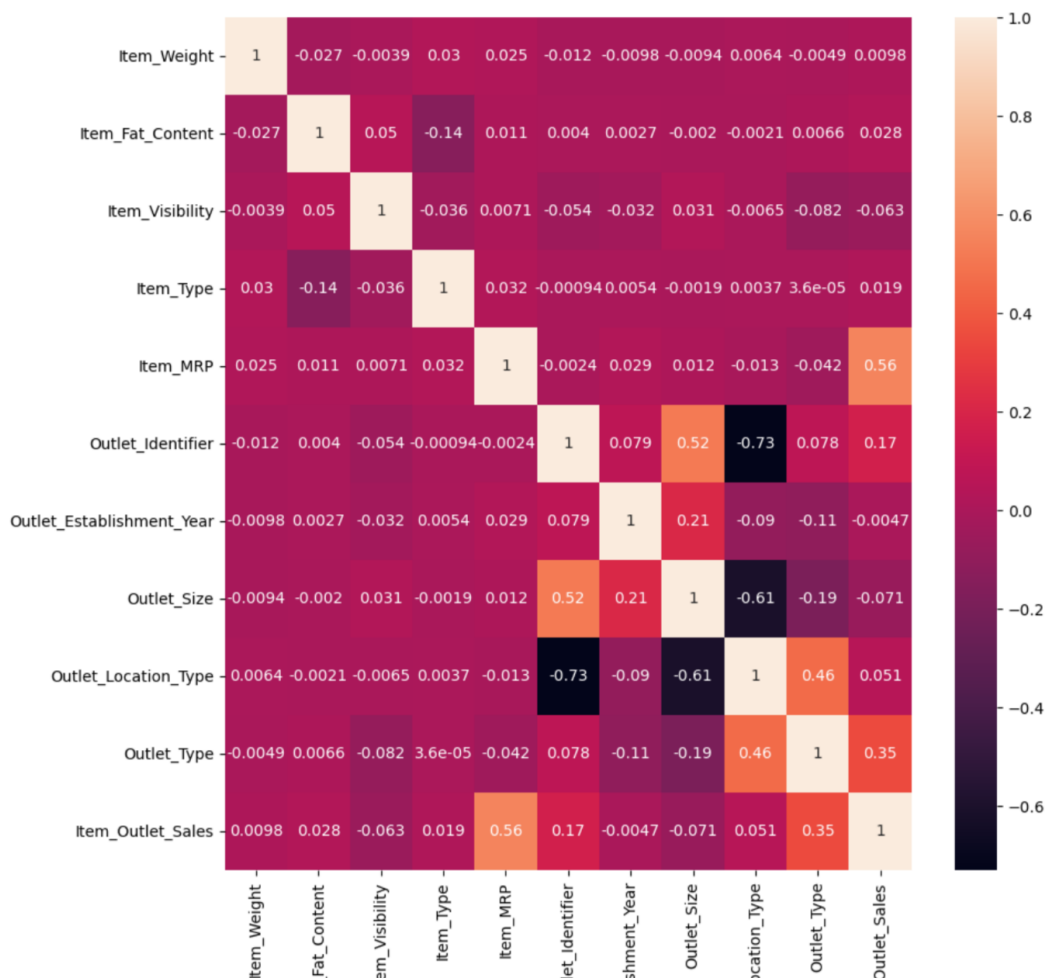
[52] train.head(3)
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	clus
0	9.30	0	0.016047	4	249.8092	9	1999	1	0	1	3735.1380	
1	5.92	1	0.019278	14	48.2692	3	2009	1	2	2	443.4228	
2	17.50	0	0.016760	10	141.6180	9	1999	1	0	1	2097.2700	

Hình 23. Mã hóa các biến dạng phân loại

- Dùng .corr() để tính độ tương quan ma trận của tập train và sau đó dùng sns.heatmap() để biểu diễn ma trận tương quan. Biểu đồ này giúp dễ dàng nhận ra các mối quan hệ tuyến tính mạnh hoặc yếu giữa các biến số. Có thể liệt kê một số biến có tương quan cao như: Item\_MRP và Outlet\_type nhưng nhìn chung các biến này sẽ không gây ra quá nhiều ảnh hưởng đa cộng tuyến.

```
plt.figure(figsize=(10, 10))
sns.heatmap(train.corr(),annot=True)
plt.show()
```



Hình 24. Xem ma trận tương quan

- Tiếp tục sử dụng `train_test_split` để chia mô hình ra thành tập train và test dùng cho huấn luyện và đánh giá. Ở phần này em chia dữ liệu thành tỉ lệ 75% train và 25% test với `random_state` là 11.

```
from sklearn.model_selection import train_test_split
x=train.drop(['Item_Outlet_Sales'],axis=1)
y=train['Item_Outlet_Sales']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =0.25, random_state = 11)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

((6026, 11), (6026,), (2009, 11), (2009,))

Hình 25. Chia tập dữ liệu

- Xây dựng mô hình Linear regression để tiến hành đánh giá lần 1 các thang đo được sử dụng cho quá trình đánh giá là MSE, MAE, RMSE và  $R^2$ . Có thể thấy được MSE và RMSE rất cao, MAE cho thấy sai số trung bình với kết quả dự

đoán của mô hình là khoản 803 cũng ở mức rất cao. Giá trị  $R^2$  là 0.49 cho thấy khoản 49% phương sai của biến mục tiêu được giải thích bởi các biến độc lập trong mô hình. Nhưng còn cho thấy rằng 51% phương sai trong mô hình không được giải thích. Chung quy kết quả đánh giá mô hình vẫn chưa hoàn toàn khả quan và cần cải tiến thêm để cho ra độ chính xác tốt hơn.

```
#Khởi tạo và huấn luyện
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x_train, y_train)
y_pred = lin_reg.predict(x_test)
#Đánh giá
mse_no_clusters = mean_squared_error(y_test, y_pred)
mae_no_clusters = mean_absolute_error(y_test, y_pred)
rmse_no_clusters = np.sqrt(mse_no_clusters)
r2_no_clusters = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse_no_clusters}")
print(f"Mean Absolute Error: {mae_no_clusters}")
print(f"Root Mean Squared Error: {rmse_no_clusters}")
print(f"R-squared: {r2_no_clusters}")
```

⇒ Mean Squared Error: 1137820.7439694453  
Mean Absolute Error: 823.6689762029561  
Root Mean Squared Error: 1066.6868068788726  
R-squared: 0.4930648619365956

Hình 26. Huấn luyện tập dữ liệu

- Chuẩn hóa dữ liệu bằng StandardScaler() giúp thuật toán hoạt động hiệu quả hơn bằng cách đưa tất cả các đặc trưng về cùng một thang đo, tránh hiện tượng một số đặc trưng chi phối quá trình học của mô hình.

```

▶ # Chuẩn hóa dữ liệu
scaler = StandardScaler()
train_features = train.drop(columns=['Item_Outlet_Sales'])
train_features = scaler.fit_transform(train_features)
test_features = scaler.transform(test)
train_features

⇒ array([[ -0.83433673, -0.73716446, -1.02468908, ..., -1.3690801 ,
          -0.26158209, -1.62204817],
        [ -1.62443564,  1.35654938, -0.95283609, ...,  1.10685026,
          1.04470261, -1.31469864],
        [  1.08247126, -0.73716446, -1.00883755, ..., -1.3690801 ,
          -0.26158209,  0.22204904],
        ...,
        [ -0.53045254, -0.73716446, -0.59905355, ..., -0.13111492,
          -0.26158209, -0.69999956],
        [ -1.32288901,  1.35654938,  1.84802381, ...,  1.10685026,
          1.04470261, -1.31469864],
        [  0.45132717, -0.73716446, -0.38351094, ..., -1.3690801 ,
          -0.26158209, -0.69999956]])

```

Hình 27. Chuẩn hóa dữ liệu

- Sau đó dùng dữ liệu chuẩn hóa tiến hành phân cụm và đưa vào tập dữ liệu dùng để đánh giá mô hình Linear regression khi kết hợp với phân cụm bằng GMM.

```

✓ [154] gmm = GaussianMixture(n_components=12, random_state=42)
iây   train['clusters'] = gmm.fit_predict(train_features)
      test['clusters'] = gmm.fit_predict(test_features)

```

```

✓ [155] train['clusters']
0 iây
⇒
   0      0
   1      1
   2      6
   3     11
   4      4
   ..
8518    4
8519     8
8520     3
8521     1
8522     3
Name: clusters, Length: 8035, dtype: int64

```

Hình 28. Phân cụm

- Tiến hành xây dựng lại mô hình Linear regression kết hợp với kết quả gom cụm.



Xây dựng mô hình Linear Regression với kết quả gom cụm

```
[157] #Model Making
      from sklearn.model_selection import train_test_split
      x=train.drop(['Item_Outlet_Sales'],axis=1)
      y=train['Item_Outlet_Sales']
      x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =0.25, random_state = 11)

      x_train.shape, y_train.shape, x_test.shape, y_test.shape

      ((6026, 11), (6026,), (2009, 11), (2009,))

[159] #Khởi tạo và huấn luyện
      from sklearn.linear_model import LinearRegression
      lin_reg = LinearRegression()
      lin_reg.fit(x_train, y_train)
      y_pred = lin_reg.predict(x_test)
```

Hình 29. Xây dựng lại mô hình Linear regression kết hợp với kết quả gom cụm

- Tiến hành đánh giá mô hình Linear regression kết hợp với kết quả gom cụm cho thấy các giá trị của thang đo được cải tiến nhưng chưa nhiều.

```
#Đánh giá
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

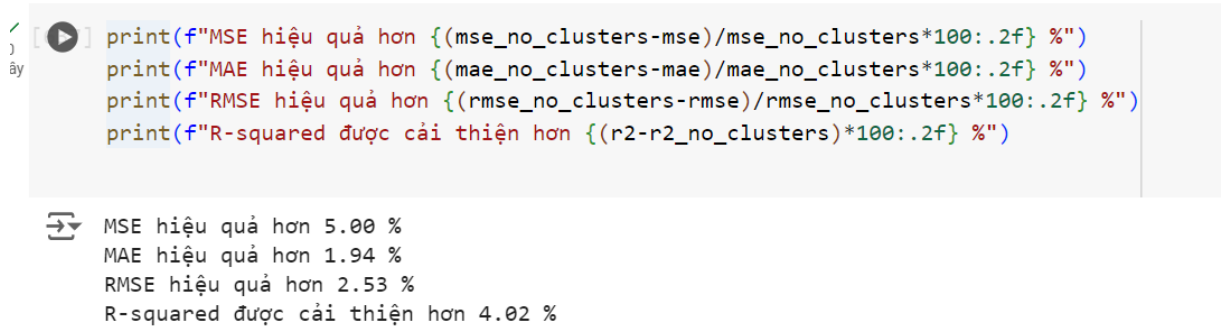
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")

Mean Squared Error: 1080974.9920954038
Mean Absolute Error: 807.6764819192066
Root Mean Squared Error: 1039.6994720088126
R-squared: 0.5332555278367164
```

Hình 30. Kết quả mô hình dự đoán khi kết hợp

### 3.7 Kết quả

Dựa vào quá trình thực hiện trên ta có thể thấy được độ hiệu quả của Linear regression được cải thiện nhưng không quá nhiều. Nhưng vẫn là một kết quả đáng mong đợi hiệu quả có thể sẽ tăng khi ta tối ưu các mô hình.



```
print(f"MSE hiệu quả hơn {(mse_no_clusters-mse)/mse_no_clusters*100:.2f} %")
print(f"MAE hiệu quả hơn {(mae_no_clusters-mae)/mae_no_clusters*100:.2f} %")
print(f"RMSE hiệu quả hơn {(rmse_no_clusters-rmse)/rmse_no_clusters*100:.2f} %")
print(f"R-squared được cải thiện hơn {(r2-r2_no_clusters)*100:.2f} %")
```

MSE hiệu quả hơn 5.00 %  
MAE hiệu quả hơn 1.94 %  
RMSE hiệu quả hơn 2.53 %  
R-squared được cải thiện hơn 4.02 %

*Hình 31. So sánh kết quả 2 phương pháp*

## **CHƯƠNG 4: KẾT QUẢ - KẾT LUẬN**

### **4.1 Nhận xét kết quả**

Trong quá trình thực hiện áp dụng thuật toán EM vào bài toán dự đoán doanh số bán hàng tuy kết quả cải thiện hiệu suất không đạt được như nhóm em mong đợi nhưng có thể thấy được phương pháp áp dụng thuật toán EM vào mô hình dự đoán là khả quan. Kết quả của quá trình nghiên cứu và phát triển mô hình dự đoán doanh số bán hàng cho thấy rằng việc sử dụng các thuật toán học máy và các phương pháp tiền xử lý dữ liệu đã mang lại những cải thiện đáng kể về hiệu suất. Tuy nhiên, vẫn còn nhiều cơ hội để tối ưu hóa và nâng cao độ chính xác của mô hình. Việc tiếp tục nghiên cứu và áp dụng các phương pháp mới sẽ giúp đạt được kết quả tốt hơn trong tương lai.

### **4.2 Ưu nhược điểm**

- Về ưu điểm của việc áp dụng thuật toán EM vào trong bài toán dự đoán doanh số giúp mô hình dự đoán trở nên ổn định hơn, cho thấy tiềm năng của thuật toán khi kết hợp với các mô hình dự đoán khác.
- Nhược điểm có thể thấy được là mức độ cải thiện vẫn chưa đạt mức kì vọng cho thấy mô hình dự đoán và mô hình phân cụm cần được tối ưu thêm.

### **4.3 Hướng phát triển**

- Chọn lọc lại các đặc trưng ảnh hưởng đến doanh số bán hàng.
- Áp dụng phương pháp tương tự với các tập dữ liệu khác để kiểm tra hiệu suất của mô hình cũng như độ tương thích.
- Áp dụng thử với nhiều mô hình dự đoán khác.
- Tinh chỉnh các siêu tham của mô hình để tối ưu hiệu suất.

## TÀI LIỆU THAM KHẢO

- [1] raman\_257, 1 8 2023. [Trực tuyến]. Available: <https://www.geeksforgeeks.org/ml-expectation-maximization-algorithm/>.
- [2] tufan\_gupta2000, 10 6 2023. [Online]. Available: <https://www.geeksforgeeks.org/gaussian-mixture-model/>.
- [3] m. g. :), 20 3 2024. [Trực tuyến]. Available: <https://www.geeksforgeeks.org/ml-linear-regression/>.
- [4] T. Vu, 2021. [Trực tuyến]. Available: [https://machinelearningcoban.com/tabml\\_book/ch\\_data\\_processing/process\\_outliers.html](https://machinelearningcoban.com/tabml_book/ch_data_processing/process_outliers.html).
- [5] aakarshachug, 18 4 2023. [Trực tuyến]. Available: <https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/>.