

BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI
BỘ MÔN CÔNG NGHỆ THÔNG TIN



BÁO CÁO HỌC PHẦN
KHAI PHÁ DỮ LIỆU
ĐỀ TÀI

3A SUPER STORE (MARKET ORDERS DATA – CRM)

GIẢNG VIÊN HƯỚNG DẪN TS. VŨ THỊ HẠNH
Nhóm sinh viên thực hiện:
Trịnh Á Châu – 2251068179
Ngô Nhật Minh - 2251068209

Thành phố Hồ Chí Minh , ngày 28 tháng 10 năm 2025

PHIẾU CHẤM ĐIỂM

Sinh viên thực hiện:

Họ và tên	Công việc thực hiện
Trịnh Á Châu (Dự đoán chi tiêu khách hàng)	<ul style="list-style-type: none">- Tổng hợp nội dung word, nội dung nội dung notebook- Tiền xử lý dữ liệu và làm sạch dữ liệu- Huấn luyện mô hình BG-NBG và Gamma Gamma- Thực hiện Elbow Method cho Kmean để phân loại khách hàng- Lưu mô hình và đánh giá- Tổng hợp báo cáo, trình bày kết quả và đề xuất hướng phát triển
Ngô Nhật Minh (Dự đoán doanh thu / lợi nhuận của chi nhánh trong 1 tháng tiếp theo)	<ul style="list-style-type: none">- Viết nội dung word và notebook- Tiền xử lý dữ liệu- Huấn luyện mô hình Prophet và XGBoost- Lưu mô hình và đánh giá- Đề xuất hướng phát triển

MỤC LỤC

LỜI NÓI ĐẦU	1
CHƯƠNG I: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU	2
1.1 Giới thiệu Tổng quan về Khai phá Dữ liệu	2
1.2 Các kỹ thuật khai phá dữ liệu.....	2
1.2.1 Kỹ thuật Khai phá Luật Kết hợp (Association Rule Mining).....	2
1.2.2 Kỹ thuật Phân lớp (Classification).....	2
1.2.3 Kỹ thuật phân cụm (Clustering)	3
CHƯƠNG II: DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU (DATA AND PREPROCESSING).....	5
2.1 Giới thiệu về tập dữ liệu.....	5
2.2 Mục tiêu và phạm vi dự án	5
2.3 Tiền xử lý dữ liệu	7
2.3.1 Tầm quan trọng của Tiền xử lý dữ liệu	7
2.3.2 Quy trình tiền xử lý dữ liệu	8
2.3.3 Chi tiết Tiền xử lý dữ liệu	9
2.3.4 Chi tiết tiền xử lý dữ liệu cho dự đoán doanh thu theo chi nhánh	14
Chương III: PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU VÀ XÂY DỰNG MÔ HÌNH DỰ ĐOÁN.....	24
3.1 Phương pháp khai phá dữ liệu	24
3.1.1 Khái niệm về khai phá dữ liệu.....	24
3.1.2 Các phương pháp khai phá dữ liệu.....	24
3.2 Xây dựng mô hình dự đoán	24
3.2.1. Dự đoán chi tiêu của khách hàng.....	25
3.2.2. Dự đoán doanh thu của từng chi nhánh	35
Chương IV: KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH	39
4.1 Dự đoán chi tiêu của khách hàng.....	39
4.1.1 Mô hình BG-NBD	39
4.1.2 Mô hình Gamma - Gamma.....	41
4.2 Dự đoán doanh thu của từng chi nhánh	42

CHƯƠNG V: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	48
5.1 Dự đoán chỉ tiêu của khách hàng.....	48
5.2 Dự đoán doanh thu của từng chi nhánh.....	49
TÀI LIỆU THAM KHẢO.....	50

DANH MỤC HÌNH ẢNH

Hình 1 Load dữ liệu.....	9
Hình 2 Kiểm tra dữ liệu.....	9
Hình 3 Thông tin về dữ liệu	10
Hình 4 Chuyển đổi dạng dữ liệu của ngày giao dịch sang dạng date_time	11
Hình 5 Xử lý cột TOTALBASKET	11
Hình 6 Kiểm tra lại dữ liệu sau khi đã xử lý	12
Hình 7 Kiểm tra sự phân bố của dữ liệu	12
Hình 8 Kiểm tra giá trị ngoại lệ.....	13
Hình 9 Xử lý giá trị ngoại lệ.....	13
Hình 10 Chuyển định dạng số	14
Hình 11 Giảm số cột không cần thiết	14
Hình 12 Điều chỉnh định dạng ngày giờ.....	15
Hình 13 Tập dữ liệu sau khi điều chỉnh định dạng ngày.....	15
Hình 14 Chuyển dấu phẩy sang dấu chấm hoặc bỏ luôn.....	16
Hình 15 Chuyển định dạng từ object sang float	16
Hình 16 Chuyển dữ liệu dạng Object sang chữ thường	16
Hình 17 Kiểm tra dữ liệu rỗng	17
Hình 18 : Biểu đồ Boxtrot thể hiện dữ liệu ngoại lệ	17
Hình 19 Mô tả lợi nhuận với các dữ liệu nhất định.....	18
Hình 20 Kiểm tra giá trị trong phân vị nhất định	18
Hình 21 Kết quả ra được	19
Hình 22 Biểu đồ thể hiện giá trị tại khoảng phân vị nhất định.....	19
Hình 23 Gọt tập dữ liệu	20
Hình 24 Biểu đồ Boxtrot thể hiện dữ liệu ngoại lệ sau tiền xử lí.....	20
Hình 25 Các hàm để xuất ra biểu đồ	21
Hình 26 Biểu đồ thể hiện giá trị doanh thu tại khoảng phân vị nhất định.....	21
Hình 27 Kiểu dữ liệu	22
Hình 28 Các thông số của TOTALBASKET	22
Hình 29 Chuyển đổi mô hình RFM.....	25
Hình 30 Đặt tên cho các cột của RFM	26
Hình 31 Tính giá trị chi trả trung bình của khách hàng	28
Hình 32 Chuyển đổi Recency và T sang đơn vị tuần	28
Hình 33 Train mô hình BG-NBD	29
Hình 34 Dự đoán số lần mua của khách hàng trong tương lai	30
Hình 35 Fitting dữ liệu cho mô hình Gamma Gamma.....	31
Hình 36 Dự đoán CLV cho khách hàng trong 3 tháng kế tiếp.....	31
Hình 37 Sử dụng Elbow Method để tìm chỉ số k cho mô hình K-Means	32
Hình 38 Mô hình K-Means phân loại khách hàng	33

Hình 39 Kiểm tra thông tin các cụm	34
Hình 40 Phân loại khách hàng	34
Hình 41 Số lượng khách hàng mỗi phân khúc	35
Hình 42 Nhập mô hình Prophet	35
Hình 43 Huấn luyện mô hình Prophet	36
Hình 44 Nhập mô hình XGBoost	37
Hình 45 Hàm tạo đặc điểm	37
Hình 46 Tách dữ liệu theo ngày	37
Hình 47 Biểu đồ thể hiện tổng doanh thu theo thời gian	38
Hình 48 Tách tập dữ liệu	38
Hình 49 Huấn luyện mô hình	38
Hình 50 Bảng Summary của BG-NBD	39
Hình 51 Bảng Summary của mô hình Gamma-Gamma	41
Hình 52 Mô tả dự đoán doanh thu	43
Hình 53 Biểu đồ dự đoán doanh thu	43
Hình 54 Nhập 3 mô hình điểm số	44
Hình 55 Vòng lặp để lấy điểm MAE, RMSE, và R2	44
Hình 56 Xuất ba điểm số trên	45
Hình 57 Kết quả trả về	45
Hình 58 Biểu đồ thể hiện độ quan trọng của biến phụ thuộc	46
Hình 59 Ba điểm số đánh giá mô hình	46
Hình 60 Biểu đồ thể hiện tổng doanh thu theo thời gian	47

LỜI NÓI ĐẦU

Trong thời kỳ chuyển đổi số mạnh mẽ, dữ liệu trở thành nguồn tài nguyên chiến lược giúp doanh nghiệp bán lẻ nắm bắt xu hướng thị trường và hành vi người tiêu dùng. Khả năng dự báo chính xác doanh thu và mức chi tiêu của khách hàng không chỉ hỗ trợ quản lý tài chính hiệu quả mà còn giúp tối ưu hóa chiến lược kinh doanh và gia tăng lợi nhuận.

Dự án này hướng đến việc ứng dụng các phương pháp học máy trong dự báo doanh thu chi nhánh và phân tích hành vi chi tiêu của khách hàng dựa trên dữ liệu giao dịch thực tế. Thông qua việc khai thác các mô hình thống kê và thuật toán dự đoán, dự án mong muốn xây dựng hệ thống hỗ trợ ra quyết định có khả năng thích ứng với biến động của thị trường bán lẻ.

Báo cáo trình bày toàn bộ quy trình xử lý dữ liệu theo hướng khoa học, bao gồm khám phá dữ liệu, tiền xử lý, xây dựng đặc trưng và huấn luyện mô hình dự báo. Kết quả không chỉ đánh giá độ chính xác của mô hình mà còn rút ra những yếu tố tác động lớn đến doanh thu và hành vi mua hàng, qua đó mang lại giá trị thực tiễn cho quản trị và hoạch định chiến lược kinh doanh.

CHƯƠNG I: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU

1.1 Giới thiệu Tổng quan về Khai phá Dữ liệu

Khai phá dữ liệu (Data Mining) là một lĩnh vực thuộc khoa học dữ liệu, tập trung vào việc khai thác tri thức tiềm ẩn từ các tập dữ liệu lớn thông qua các thuật toán và phương pháp tính toán. Mục tiêu của Data Mining là phát hiện những mẫu, quy luật và mối quan hệ có ý nghĩa trong dữ liệu mà con người khó nhận thấy bằng quan sát trực tiếp.

Khác với các quy trình xử lý dữ liệu thông thường, khai phá dữ liệu không chỉ dừng lại ở việc thống kê hay mô tả mà hướng đến việc tự động hoặc bán tự động rút ra tri thức mới, hỗ trợ cho việc ra quyết định. Quá trình này bao gồm nhiều bước như lựa chọn dữ liệu, tiền xử lý, biến đổi, áp dụng thuật toán khai phá và đánh giá kết quả.

Data Mining vẫn cần sự tham gia của con người trong việc hiểu rõ ngữ cảnh dữ liệu, chọn phương pháp phù hợp và diễn giải kết quả. Ngoài ra, chất lượng dữ liệu đóng vai trò quan trọng — dữ liệu phải được làm sạch, loại bỏ nhiễu và sai lệch trước khi đưa vào phân tích. Các mô hình khai phá dữ liệu yêu cầu tập dữ liệu đủ lớn, đa dạng và đáng tin cậy để có thể trích xuất được tri thức hữu ích và đảm bảo tính chính xác của kết quả.

1.2 Các kỹ thuật khai phá dữ liệu

1.2.1 Kỹ thuật Khai phá Luật Kết hợp (Association Rule Mining)

Trong khai phá dữ liệu, mục đích của luật kết hợp là tìm ra các mối quan hệ giữa các đối tượng trong khối lượng lớn dữ liệu. Để khai phá luật kết hợp có rất nhiều thuật toán, nhưng dùng phổ biến nhất là thuật toán Apriori. Đây là thuật toán khai phá tập phổ biến trong dữ liệu giao dịch để phát hiện các luật kết hợp dạng khẳng định nhị phân và được sử dụng để xác định, tìm ra các luật kết hợp trong dữ liệu giao dịch. Ngoài ra, còn có các thuật toán FP-growth, thuật toán Partition,...

1.2.2 Kỹ thuật Phân lớp (Classification)

Kỹ thuật phân lớp (Classification) là một nhánh quan trọng trong khai phá dữ liệu, được sử dụng để dự đoán nhãn hoặc nhóm của một đối tượng dựa trên các thuộc

tính đã biết. Mục tiêu là xây dựng một mô hình học từ dữ liệu huấn luyện, sau đó áp dụng mô hình này để phân loại các đối tượng mới chưa được gán nhãn.

Một số thuật toán phân lớp phổ biến gồm:

- **Phân lớp bằng cây quyết định (Decision Tree) :**

Phân lớp dựa trên cấu trúc dạng cây, trong đó mỗi nút đại diện cho một thuộc tính và mỗi nhánh thể hiện điều kiện phân tách. Cây quyết định giúp trực quan hóa quá trình ra quyết định và dễ dàng diễn giải kết quả.

- **Phân lớp dựa trên xác suất (Naïve Bayes):**

Dựa trên định lý Bayes, giả định rằng các thuộc tính của dữ liệu là độc lập với nhau. Thuật toán này có ưu điểm đơn giản, tốc độ xử lý nhanh và hiệu quả với dữ liệu có kích thước lớn.

- **Phân lớp dựa trên khoảng cách (K-Nearest Neighbors – KNN):**

Dựa trên khoảng cách giữa các điểm dữ liệu. Một mẫu mới sẽ được gán vào lớp của K điểm gần nhất trong không gian đặc trưng. Phương pháp này không cần huấn luyện mô hình, nhưng hiệu năng phụ thuộc vào cách chọn K và phép đo khoảng cách.

- **Phân lớp bằng SVM (Support Vector Machine – SVM):**

Tìm một siêu phẳng tối ưu để phân tách các lớp dữ liệu trong không gian nhiều chiều. SVM có khả năng xử lý tốt các bài toán phân lớp phức tạp và dữ liệu không tuyến tính thông qua việc sử dụng các hàm kernel.

1.2.3 Kỹ thuật phân cụm (Clustering)

Phân cụm dữ liệu là quá trình nhóm các đối tượng có đặc điểm tương đồng vào cùng một cụm sao cho các phần tử trong cùng cụm giống nhau nhiều hơn so với các phần tử ở cụm khác. Đây là kỹ thuật phổ biến trong khai phá dữ liệu nhằm phát hiện cấu trúc ẩn và mối quan hệ tự nhiên trong dữ liệu mà không cần nhãn sẵn có.

Một số phương pháp cơ bản:

- **K-means:** Chia dữ liệu thành k cụm bằng cách xác định tâm cụm sao cho khoảng cách giữa các điểm dữ liệu và tâm cụm là nhỏ nhất.

- Phân cụm dựa trên đồ thị hoặc mật độ: Nhóm dữ liệu dựa vào mối liên kết hoặc mật độ điểm trong không gian đặc trưng (ví dụ: DBSCAN).

Kỹ thuật phân cụm được ứng dụng trong nhiều lĩnh vực như phân nhóm khách hàng, phát hiện bất thường, phân tích hành vi và khai thác mẫu dữ liệu tiềm ẩn.

CHƯƠNG II: DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU (DATA AND PREPROCESSING)

2.1 Giới thiệu về tập dữ liệu

Link : <https://www.kaggle.com/datasets/cemeraan/3a-superstore/>

Tập dữ liệu: Bộ dữ liệu được sử dụng trong dự án là 3A Superstore Dataset, mô phỏng hệ thống bán lẻ đa chi nhánh tại Thổ Nhĩ Kỳ. Dữ liệu phản ánh toàn bộ hoạt động kinh doanh của một chuỗi siêu thị, bao gồm thông tin về : Branches, Categories, Customers, Order_Detail, và Orders.

Tổng quan bộ dữ liệu:

Số lượng bảng: 5 bảng dữ liệu có liên kết với nhau

- Branches: Thông tin về các chi nhánh bán lẻ
- Categories: Danh mục và phân loại sản phẩm
- Customers: Hồ sơ và thông tin nhận dạng khách hàng
- Order_Detail: Chi tiết từng mặt hàng trong đơn hàng
- Orders: Dữ liệu tổng hợp giao dịch

Thống kê tổng quát:

- **161** chi nhánh siêu thị
- **27.000** sản phẩm được phân loại
- **99.998** khách hàng duy nhất (kèm địa chỉ)
- **10.235.193** đơn hàng được ghi nhận
- **51.185.032** dòng dữ liệu chi tiết đơn hàng
- **230.323.422** sản phẩm được bán ra

2.2 Mục tiêu và phạm vi dự án

Mục tiêu chính:

- Xây dựng hai mô hình dự báo độc lập:
 1. Dự đoán chỉ tiêu khách hàng trong tương lai (Customer Lifetime Value – CLV) bằng cách kết hợp hai mô hình thống kê BG/NBD và Gamma-Gamma.
 2. Dự đoán doanh thu của từng chi nhánh dựa trên dữ liệu lịch sử đơn hàng, xu hướng thời gian và hành vi mua của khách hàng.
- Áp dụng K-Means để phân cụm khách hàng theo giá trị CLV và tần suất mua hàng, nhằm xác định nhóm khách hàng tiềm năng, trung thành hoặc có nguy cơ rời bỏ.
- Phân tích và đánh giá các yếu tố ảnh hưởng đến hiệu quả kinh doanh của từng chi nhánh để hỗ trợ ra quyết định chiến lược.
- Xây dựng quy trình khai phá dữ liệu toàn diện, có thể tái sử dụng cho các bài toán dự báo khác trong lĩnh vực bán lẻ.

Mục tiêu cụ thể:

- Thực hiện EDA (Exploratory Data Analysis) để hiểu rõ đặc điểm, xu hướng và mối quan hệ trong dữ liệu giao dịch.
- Tiền xử lý hơn 10 triệu bản ghi đơn hàng, gồm làm sạch, tổng hợp và chuyển đổi dữ liệu sang dạng RFM (Recency, Frequency, Monetary) cho mô hình CLV.
- Áp dụng mô hình BG/NBD để dự đoán khả năng khách hàng quay lại mua hàng, và mô hình Gamma-Gamma để ước lượng giá trị chỉ tiêu trung bình.
- Tính toán CLV (Customer Lifetime Value) cho từng khách hàng, sau đó sử dụng K-Means để phân cụm khách hàng dựa trên hành vi và giá trị chỉ tiêu.
- Xây dựng mô hình dự báo doanh thu chi nhánh bằng các thuật toán học máy (như Random Forest, XGBoost, Linear Regression).
- Trực quan hóa kết quả dự báo và phân cụm để rút ra insight phục vụ chiến lược kinh doanh và tối ưu vận hành.

Phạm vi phân tích:

- Tập trung vào hành vi chi tiêu của khách hàng và hiệu suất doanh thu của các chi nhánh.
- Dữ liệu được tổng hợp và xử lý để phục vụ cho hai bài toán chính: dự đoán CLV và dự báo doanh thu chi nhánh.

Ứng dụng thực tế:

- **Dự đoán CLV:** Giúp xác định nhóm khách hàng giá trị cao để ưu tiên chăm sóc và giữ chân.
- **Phân cụm khách hàng:** Hỗ trợ cá nhân hóa chiến dịch marketing và tối ưu nguồn lực.
- **Dự báo doanh thu chi nhánh:** Phục vụ lập kế hoạch bán hàng, quản lý tồn kho, nhân sự và chiến lược mở rộng.

Phân tích hiệu quả kinh doanh: Cung cấp thông tin định lượng giúp doanh nghiệp ra quyết định dựa trên dữ liệu thay vì cảm tính

2.3 Tiền xử lý dữ liệu

2.3.1 Tầm quan trọng của Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là bước nền tảng và chiếm nhiều công sức nhất trong quy trình khoa học dữ liệu. Theo thống kê của các chuyên gia, giai đoạn này thường chiếm 60–80% tổng thời gian thực hiện dự án, và chất lượng của nó ảnh hưởng trực tiếp đến hiệu suất và độ chính xác của mô hình.

Các nhiệm vụ chính cần xử lý trong giai đoạn này:

- Dữ liệu thiếu (Missing Values) và không đầy đủ (Incomplete Data)
- Giá trị ngoại lệ (Outliers) và dữ liệu bất thường (Anomalous Data)
- Chuẩn hóa dữ liệu (Feature Scaling & Normalization)
- Lựa chọn đặc trưng và giảm chiều (Feature Selection & Dimensionality Reduction)

2.3.2 Quy trình tiền xử lý dữ liệu

Quy trình tiền xử lý dữ liệu (Data Preprocessing) được thiết kế nhằm chuyển đổi dữ liệu thô từ bộ 3A Superstore (đặc biệt là bảng Orders) thành dạng sạch, nhất quán và sẵn sàng cho mô hình hóa.

Với dữ liệu lớn hơn 10 triệu dòng, quy trình này được thực hiện tuần tự theo các bước sau:

Bước 1. Kiểm tra và đánh giá dữ liệu đầu vào (Data Inspection)

- Kiểm tra kích thước bộ dữ liệu, loại dữ liệu và tính toàn vẹn của các đặc trưng của dữ liệu
- Phát hiện dữ liệu thiếu, dữ liệu trùng lặp, dữ liệu không khớp (giá trị âm) và giá trị ngoại lệ
- Kiểm tra định dạng thời gian và sự nhất quán của các mã khách hàng, mã chi nhánh

Bước 2: Làm sạch dữ liệu (Data Cleaning)

- Xử lý Missing Values: Với giá trị thiếu nhỏ, có thể thay thế bằng trung vị (median) hoặc giá trị trung bình (mean) nếu phù hợp.
- Xử lý dữ liệu trùng lặp: Xóa các bản ghi trùng để tránh nhân bản dữ liệu
- Phát hiện và xử lý Outliers: Sử dụng phương pháp IQR (Interquartile Range) hoặc Z-score để loại bỏ các giá trị bất thường, quá thấp hoặc quá cao so với phân phối chung.

Bước 3: Chuyển đổi dữ liệu (Data Transformation)

- Chuyển đổi định dạng thời gian: Biến DATE_ được chuyển sang kiểu datetime, sau đó trích xuất thêm các đặc trưng như ngày, tháng, quý, năm phục vụ mô hình doanh thu chi nhánh.
- Tổng hợp dữ liệu (Aggregation): Gộp dữ liệu theo USERID hoặc BRANCH_ID để phù hợp với mô hình cần huấn luyện

Bước 4: Tạo đặc trưng (Feature Engineering)

- Tính toán các đặc trưng riêng cần tạo thêm cho mô hình huấn luyện. Ví dụ tạo các đặc trưng RFM cho mô hình dự đoán chi tiêu của khách hàng.
- Tạo các biến về thời gian như ngày, tháng, năm cho mô hình hồi quy tuyến tính.

Còn rất nhiều quy trình cũng như cách thức để tiền xử lý dữ liệu nhưng dựa trên phạm vi đề tài thì chỉ cần tiền xử lý dữ liệu đến đây là có thể thực hiện train model.

2.3.3 Chi tiết Tiền xử lý dữ liệu

Load dữ liệu: Tiến hành tải bộ dữ liệu từ trên Kaggle về. Nhận thấy bộ dữ liệu gồm 5 bảng: Branches, Categories, Customer, Orders, Order_Details. Nhưng trong phạm vi dự án này thì chỉ cần dùng bộ dữ liệu Orders là đủ.

```
4]: df = pd.read_csv("/kaggle/input/3a-superstore/Orders.csv", sep = ",")
df.head(10)
```

```
4]:
```

	ORDERID	BRANCH_ID	DATE	USERID	NAMESURNAME	TOTALBASKET
0	7905270	320-DE1	2022-08-22 00:00:00	72946	Ali İlhan	2637,5499999999997
1	8131447	56-AN4	2022-06-05 00:00:00	58126	Aysun Dinç	2262,06
2	10176430	348-MU1	2023-01-02 00:00:00	41317	Taner Yavuz	2195,54
3	8445704	39-AY3	2021-01-28 00:00:00	39303	Esra Lara Keleş	446,86
4	8616360	777-YA1	2022-10-24 00:00:00	64870	Ela Çakır	430,18
5	7369024	716-BU3	2021-05-21 00:00:00	89153	Selin İlhan	996,81
6	7656284	56-AN1	2022-12-25 00:00:00	55529	Yelda Erdoğan	784,26
7	7468769	146-KA3	2023-04-29 00:00:00	23596	Hatice Çoban	402,68
8	8398865	734-IS1	2021-12-31 00:00:00	99068	Berk Koçak	1651,96
9	9203327	734-IS1	2021-12-27 00:00:00	67055	Kadir Avdın	109,32

Hình 1 Load dữ liệu

Load dữ liệu bằng pandas và nhận thấy dữ liệu gồm 6 đặc trưng: ORDERID, BRANCH_ID, DATE_, USERID, NAMESURNAME, TOTALBASKET.

Kiểm tra dữ liệu bị thiếu, trùng lặp:

```
[5]: df['ORDERID'].duplicated().sum()

[5]: 0
```

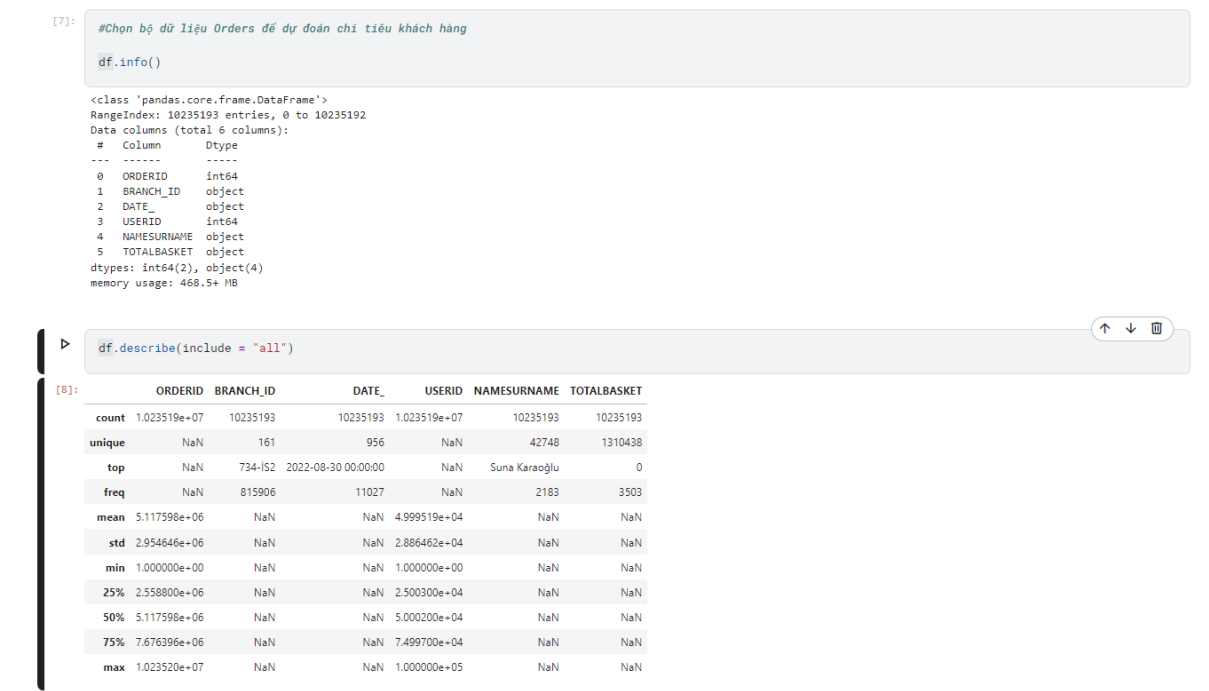
```
[6]: #Kiểm tra giá trị bị thiếu
df.isna().sum()

[6]: ORDERID      0
BRANCH_ID      0
DATE_          0
USERID         0
NAMESURNAME    0
```

Hình 2 Kiểm tra dữ liệu

Dữ liệu không có đơn hàng nào bị trùng lặp và không có giá trị bị thiếu. Có vẻ dữ liệu đã được làm khá sạch nên công đoạn tiền xử lý này cũng sẽ đơn giản và hơn.

Kiểm tra thông tin dữ liệu:



The screenshot shows a Jupyter Notebook cell with the following code and output:

```
[7]: #Chọn bộ dữ liệu Orders để dự đoán chi tiêu khách hàng
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10235193 entries, 0 to 10235192
Data columns (total 6 columns):
#   Column      Dtype
---  ---
0    ORDERID    int64
1   BRANCH_ID  object
2    DATE_      object
3    USERID    int64
4  NAMESURNAME object
5   TOTALBASKET object
dtypes: int64(2), object(4)
memory usage: 468.5+ MB
```

Below the code cell, the output of `df.describe(include = "all")` is displayed as a table:

	ORDERID	BRANCH_ID	DATE_	USERID	NAMESURNAME	TOTALBASKET
count	1.023519e+07	10235193	10235193	1.023519e+07	10235193	10235193
unique	NaN	161	956	NaN	42748	1310438
top	NaN	734-IS2	2022-08-30 00:00:00	NaN	Suna Karaoglu	0
freq	NaN	815906	11027	NaN	2183	3503
mean	5.117598e+06	NaN	NaN	4.999519e+04	NaN	NaN
std	2.954646e+06	NaN	NaN	2.886462e+04	NaN	NaN
min	1.000000e+00	NaN	NaN	1.000000e+00	NaN	NaN
25%	2.558800e+06	NaN	NaN	2.500300e+04	NaN	NaN
50%	5.117598e+06	NaN	NaN	5.000200e+04	NaN	NaN
75%	7.676396e+06	NaN	NaN	7.499700e+04	NaN	NaN
max	1.023520e+07	NaN	NaN	1.000000e+05	NaN	NaN

Hình 3 Thông tin về dữ liệu

Dữ liệu gồm có 6 cột mô tả cho hóa đơn thanh toán của một khách hàng gồm: Mã hóa đơn (ORDERID), Mã chi nhánh (BRANCH_ID), ngày giao dịch (DATE_), mã khách hàng (USERID), tên của khách hàng (NAMESURNAME), tổng tiền của hóa đơn (TOTALBASKET). Nhận thấy trong dữ liệu cột TOTALBASKET là tổng tiền nhưng lại là dạng object và DATE_ cũng chưa đúng định dạng ngày tháng nên ta tiến hành xử lý để chuyển đổi về dạng dữ liệu chuẩn.


```
In [7]: #Chuyển đổi kiểu dữ liệu sang datetime
df['DATE_'] = pd.to_datetime(df['DATE_'])
```

```
In [8]: df.head(10)
```

Out[8]:

	ORDERID	BRANCH_ID	DATE_	USERID	NAMESURNAME	TOTALBASKET
0	7905270	320-DE1	2022-08-22	72946	Ali İlhan	2637,5499999999997
1	8131447	56-AN4	2022-06-05	58126	Aysun Dinç	2262,06
2	10176430	348-MU1	2023-01-02	41317	Taner Yavuz	2195,54
3	8445704	39-AY3	2021-01-28	39303	Esra Lara Keleş	446,86
4	8616360	777-YA1	2022-10-24	64870	Ela Çakır	430,18
5	7369024	716-BU3	2021-05-21	89153	Selin İlhan	996,81
6	7656284	56-AN1	2022-12-25	55529	Yelda Erdoğan	784,26
7	7468769	146-KA3	2023-04-29	23596	Hatice Çoban	402,68
8	8398865	734-IS1	2021-12-31	99068	Berk Koçak	1651,96
9	9203327	734-IS1	2021-12-27	67055	Kadir Aydın	109,32

Hình 4 Chuyển đổi dạng dữ liệu của ngày giao dịch sang dạng date_time

```
In [10]: # 2. Kiểm tra giá trị không phải số
df[~df['TOTALBASKET'].str.isnumeric() == False]
```

Out[10]:

	ORDERID	BRANCH_ID	DATE_	USERID	NAMESURNAME	TOTALBASKET
33	5766176	427-GA3	2021-02-13	63550	Atilla Deniz Yaman	42
246	87944	427-GA3	2022-06-09	31372	Can Güven	1511
286	1556360	131-HA1	2022-11-30	28437	Figen Özge İlhan	1404
827	4653329	542-KO1	2022-07-25	37971	Deniz Nevzat İlhan	674
946	7646666	655-SA3	2021-12-21	10678	Özgür Güney	95
...
10234398	7599510	734-IS2	2022-09-05	80580	Feride Çelik	336
10234402	3918914	542-KO2	2021-05-30	30419	Atilla Hasan Eren	440
10234662	10168477	734-IS2	2021-03-17	92533	Gül Hande Başar	278
10234911	5487429	345-MA3	2022-08-02	67552	Suna Nurgül Erdem	94
10235139	6537061	734-IS1	2021-01-19	84028	Ercan Burak Sönmez	396

124643 rows × 6 columns

```
In [11]: #Các giá trị là giá trị Int nhưng trong đó các giá trị còn lại của TotalBasket là Float nên bị hiểu
là một Object.
#Thực hiện chuyển giá trị của cột này thành giá trị float64
#Chuyển dấu , trong số thành dấu . để có thể chuyển thành float64

df["TOTALBASKET"] = df["TOTALBASKET"].str.replace(',','.')
df["TOTALBASKET"] = df["TOTALBASKET"].astype('float64')
```

Hình 5 Xử lý cột TOTALBASKET

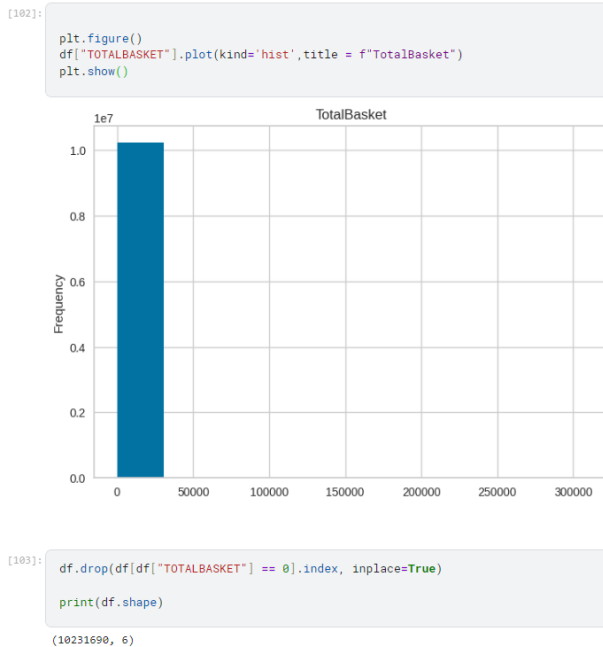
```
[14]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10235193 entries, 0 to 10235192
Data columns (total 6 columns):
#   Column      Dtype
---  -
0   ORDERID     int64
1   BRANCH_ID   object
2   DATE_       datetime64[ns]
3   USERID      int64
4   NAMESURNAME object
5   TOTALBASKET float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(2)
memory usage: 468.5+ MB

+ Code      + Markdown
```

Hình 6 Kiểm tra lại dữ liệu sau khi đã xử lý

Dữ liệu hiện tại đã khá sạch. Hãy kiểm tra thử khoảng dữ liệu của TOTALBASKET xem có đơn hàng nào là giá trị âm không.



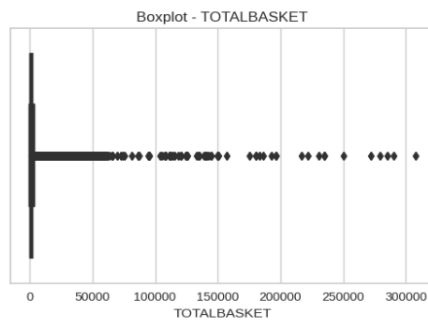
Hình 7 Kiểm tra sự phân bố của dữ liệu

Các đơn hàng không có giá trị âm nhưng có những đơn hàng 0 đồng. Dữ liệu đó không cần thiết nên sẽ xóa những đơn có TOTALBASKET là 0.

Kiểm tra giá trị ngoại lệ của TOTALBASKET

```
[109]: #Kiểm tra các outlier của TOTALBASKET bằng boxplot
```

```
plt.figure(figsize=(6,4))
sns.boxplot(x=df['TOTALBASKET'])
plt.title('Boxplot - TOTALBASKET')
plt.show()
```



Hình 8 Kiểm tra giá trị ngoại lệ

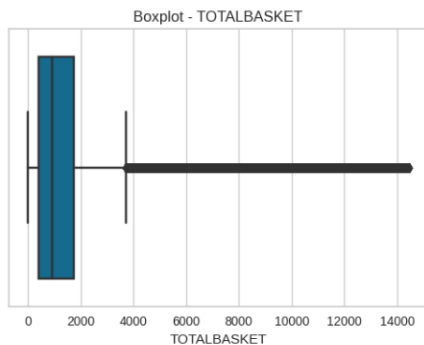
Với bộ dữ liệu hơn 10 triệu hóa đơn thì các giá trị ngoại lệ có vẻ dữ liệu đang bị lệch quá nhiều vào bên phải khi các giá trị ngoại lệ rất cao so với giá trị phổ biến. Điều này có thể làm cho mô hình bị nhiễu dự đoán không tốt nên cần phải chuẩn hóa lại dữ liệu bằng IQR (Interquartile Range) để giảm bớt outlier

```
# Xác định các mốc Q1 và Q3 với giá trị tại 1% của Q1 và 99% của Q3
Q1 = df['TOTALBASKET'].quantile(0.01)
Q3 = df['TOTALBASKET'].quantile(0.99)
IQR = Q3 - Q1
```

```
upper = Q3 + 1.5 * IQR
# Thay giá trị vượt ngưỡng trên (Không dùng lower vì không có giá trị âm)
df = df[df['TOTALBASKET'] <= upper]
```

+ Code + Markdown

```
[25]: plt.figure(figsize=(6,4))
sns.boxplot(x=df['TOTALBASKET'])
plt.title('Boxplot - TOTALBASKET')
plt.show()
```



Hình 9 Xử lý giá trị ngoại lệ

Sau khi đã xử lý ngoại lệ dữ liệu tuy vẫn còn ngoại lệ nhưng giá trị không còn bị lệch quá nhiều như lúc đầu nên có thể chấp nhận được.

2.3.4 Chi tiết tiền xử lý dữ liệu cho dự đoán doanh thu theo chi nhánh

- Quá trình đọc và kiểm tra thông tin dữ liệu tương tự với quá trình tiền xử lý dữ liệu của dự đoán chi tiêu khách hàng, vậy nên không cần nhắc lại tại đây.
- Điểm khác biệt ở đây là việc dự đoán này yêu cầu việc tiền xử lý khác hơn nhiều so với phần tiền xử lý của dự đoán chi tiêu trên.

```
pd.options.display.float_format = '{:.2f}'.format
```

Hình 10 Chuyển định dạng số

- Chuyển định dạng số từ dạng chứa “1e+n”, tức 10^n (với n là số tự nhiên bất kì), thành dạng thuần số (không chứa chữ nào, chỉ được chứa một dấu chấm thập nhân nếu có) được làm tròn đến 2 chữ số thập phân.

```
df_orders = df_orders.drop(['NAMESURNAME', 'USERID', 'ORDERID'], axis=1)
```

Hình 11 Giảm số cột không cần thiết

- Giảm đi số cột không cần thiết, tức không liên quan đến việc phân tích doanh thu theo chi nhánh như đã nêu trên, nhằm tiết kiệm chi phí tài nguyên trong quá trình phân tích, nhất là khi bộ nhớ RAM với tài nguyên có giới hạn là 12.7 GB (nếu dùng CPU để tăng tốc phần cứng) hoặc 30 GB (nếu chạy trên Kaggle).
- Khi này ta chỉ còn 3 cột `BRANCH_ID`, `DATE_`, và `TOTALBASKET`:
 - + `BRANCH_ID`: Mã khóa (ID) của nhánh tại vị trí nhất định trong tập dữ liệu `Branches.csv`. Vì mỗi mã khóa trên cũng như địa chỉ của nhánh liên quan đều có tính độc đáo riêng của mỗi đối tượng, nên mỗi dòng trên trên cột này đều sẽ được coi như địa chỉ của nhánh, thay cho việc phải nhập và gộp dữ liệu từ các cột trên `Branches.csv`, với lý do tiết kiệm tài nguyên được nêu trên;
 - + `DATE_`: Ngày (và giờ) mà hóa đơn được xuất ra tại thời điểm đó, theo thời gian thực;
 - + `TOTALBASKET`: Lợi nhuận, hoặc doanh thu. Nguồn của tập dữ liệu `Orders` không giải thích rõ ràng định nghĩa của cột trên, vậy nên từ thời điểm này cột này có thể được hiểu theo một trong hai nghĩa trên.

```
df_orders['DATE_'] = pd.to_datetime(df_orders['DATE_'])
df_orders['DATE_'] = df_orders['DATE_'].dt.strftime('%m/%d/%Y')
```

Hình 12 Điều chỉnh định dạng ngày giờ

- Chuyển định dạng ngày trong cột DATE_ từ kiểu Object (String) sang datetime, rồi chuyển định dạng ngày-tháng-năm sang “%m/%d/%y”, tức “tháng/ngày/năm”. Lí do ta chỉnh định dạng như thế là để phục vụ cho việc đưa tập dữ liệu (sau tiền xử lí) vào mô hình Prophet (sẽ được giải thích sau này), vốn chỉ chấp nhận định dạng “%m/%d/%y” kể trên.

df_orders.head()

	BRANCH_ID	DATE_	TOTALBASKET
0	320-DE1	08/22/2022	26,375,499,999,999,900
1	56-AN4	06/05/2022	2262,06
2	348-MU1	01/02/2023	2195,54
3	39-AY3	01/28/2021	446,86
4	777-YA1	10/24/2022	430,18

Hình 13 Tập dữ liệu sau khi điều chỉnh định dạng ngày

- Từ giờ, nếu báo cáo không nhắc gì thêm, thì dấu chấm thập phân trong Colab được coi như dấu phẩy thập phân của nước ta.
- Trong cột “TOTALBASKET”, dấu phẩy được sử dụng vừa để tách số bên trái dấu chấm thập phân theo định dạng 3-3-3, vừa để được sử dụng như dấu chấm/phẩy thập phân. Rất khó để Colab biết được nếu “26,375,499,999,999,900” nghĩa là “2637,5499999999997” (hai ngàn sáu trăm ba bảy phẩy ...) hoặc “26 375 499 999 999 900” (hai sáu triệu ba trăm bảy lăm ngàn bốn trăm chín chín tỷ...) khi tập tin .csv (được đọc bởi Excel) đọc con số như thế khác với cách số trên được đọc trên Kaggle.
- Vậy nên, ta chạy vòng lặp for để biến dấu phẩy thành dấu chấm thập phân, hoặc xóa luôn dấu phẩy trên:
 - + Nếu có đúng một dấu phẩy, ta chuyển nó thành dấu chấm thập phân;
 - + Nếu có bằng hoặc trên hai dấu phẩy, ta xóa hết dấu phẩy trong số đó đi.

```
def change_comma(basket):
    if str(basket).count(',') == 1:
        return str(basket).replace(',', '.')
    else:
        return str(basket).replace(',', '')

df_orders['TOTALBASKET'] = df_orders['TOTALBASKET'].apply(change_comma)
```

Hình 14 Chuyển dấu phẩy sang dấu chấm hoặc bỏ luôn

- Kết quả ra được khi đó có thể là một con số hàng ngàn, hoặc lên tới hàng triệu tỉ. Ta coi các đối tượng hóa đơn chứa lợi nhuận lên tới hàng triệu tỉ sau xử lý trên như dữ liệu ngoại lệ (outliner data) mà ta sẽ xóa sau trong quá trình tiền xử lý này.
- Ngay sau khi ta chuyển dấu phẩy xong, tất cả con số trong cột này được chuyển định dạng từ Object (String) sang float (có hỗ trợ số thập phân) cho việc phân tích sau này.

```
df_orders['TOTALBASKET'] = df_orders['TOTALBASKET'].astype(float)
```

Hình 15 Chuyển định dạng từ object sang float

- Chuyển dữ liệu chữ của các cột trong tập dữ liệu từ chữ hoa lẫn chữ thường sang chữ thuần thường.
- Python phân biệt chữ hoa chữ thường, tức coi một tên (kiểu Object/String) với hai kiểu viết khác nhau là hai đối tượng hoàn toàn khác nhau. Ví dụ: Python coi “Hà Nội” và “hà nội” như hai danh từ riêng khác nhau tuyệt đối, dù chỉ khác nhau chỗ viết in hoa và in thường, và đều chung một danh từ riêng nêu trên.
- Vậy nên, các chữ trong hai tập dữ liệu trên, bao gồm cả mã nhánh (BRANCH_ID), được chuyển sang in hoa hết nhằm đồng bộ hóa tên các địa danh cho việc phân tích dữ liệu sau này.

```
for col in df_branches.select_dtypes(include='object').columns:
    df_branches[col] = df_branches[col].str.lower().str.strip()

for col in df_orders.select_dtypes(include='object').columns:
    df_orders[col] = df_orders[col].str.lower().str.strip()
```

Hình 16 Chuyển dữ liệu dạng Object sang chữ thường

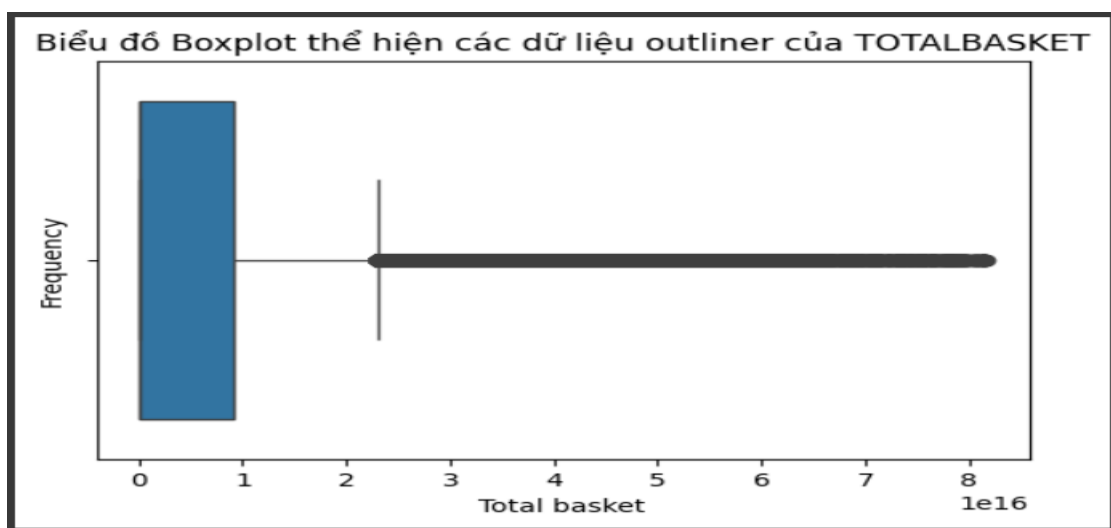
- Ngay sau đó, ta tiến hành kiểm tra tập dữ liệu nếu tồn tại các dòng chứa dữ liệu rỗng hay không.

```
df_orders.isna().sum()

0
BRANCH_ID 0
DATE_      0
TOTALBASKET 0
dtype: int64
```

Hình 17 Kiểm tra dữ liệu rỗng

- Như những gì ta thấy trên hình, thì không tồn tại dòng dữ liệu rỗng nào trên tập dữ liệu.
- Ta tiến hành kiểm tra dữ liệu ngoại lệ của hóa đơn từ biểu đồ Boxplot và số lượng doanh thu với lợi nhuận trong một khoảng nhất định. Việc này rất quan trọng trong việc huấn luyện mô hình, nhằm tránh việc kết quả ra được thiên vị về phía con số lớn (như ta đã thấy), và làm hao tốn tài nguyên có hạn của hệ thống trong quá trình máy huấn luyện mô hình.



Hình 18 : Biểu đồ Buxtrot thể hiện dữ liệu ngoại lệ

- Biểu đồ Boxplot thể hiện dữ liệu tập trung ở vị trí nào một cách trực quan, dựa trên tần suất xuất hiện của dữ liệu trong một khoảng phân vị nhất định (Trong trường hợp này là doanh thu/lợi nhuận trong cột TOTALBASKET).
- Ta chỉ thấy được rõ vạch tại phân vị 75% và 100%, trong khi các vạch còn lại bị gom vào một chỗ gần giá trị 0 trên biểu đồ.

```
df_orders['TOTALBASKET'].describe()
```

TOTALBASKET	
count	1048575.00
mean	6248292769792183.00
std	11327590205430676.00
min	0.00
25%	579.86
50%	1611.67
75%	9221449999999996.00
max	8172230000000000.00

```
dtype: float64
```

Hình 19 Mô tả lợi nhuận với các dữ liệu nhất định

- Hiện thị các giá trị nhất định liên quan đến cột TOTALBASKET trước khi loại bỏ dữ liệu ngoại lệ:

- + Count: Số lượng dòng trên cột (1048575 dòng);
- + Mean: Trung bình cộng giá trị của tất cả các dòng trên cột (6248292769792183);
- + STD: Độ lệch chuẩn của giá trị các dòng trên cột xung quanh giá trị trung bình cộng nêu trên;
- + Min và Max: Lần lượt là giá trị nhỏ nhất (0) và lớn nhất (8172230000000000), tương ứng với phân vị 0% và 100%;
- + 25%, 50%, 75%: Giá trị của dòng tại vị trí phân vị 25% (579.86), 50%/trung vị/median (1611.67), và 75% (9221449999999996); tương ứng với các khoảng Q1, Q2, và Q3 (quartile thứ nhất, hai, và ba).

- Như ta đã thấy, có sự khác biệt rõ ràng giữa hai giá trị tại trung vị 50% và phân vị 75%. Vậy nên, ta tiến hành chạy vòng lặp "for" để kiểm tra giá trị tại từng phân vị trong khoảng phân vị đấy.

```
percentile_50to75 = pd.DataFrame(columns=['PERCENTILE', 'VALUE'])

for i in range(50, 76):
    new_row = pd.DataFrame({
        'PERCENTILE': [i],
        'VALUE': [np.percentile(df_orders['TOTALBASKET'], i, interpolation='midpoint')]
    })
    percentile_50to75 = pd.concat([percentile_50to75, new_row], ignore_index=True)
    print("Value of total basket at the " + str(i) + "% percentile: ")
    print(np.percentile(df_orders['TOTALBASKET'], i, interpolation='midpoint'))
    print("\n")

print(percentile_50to75)
```

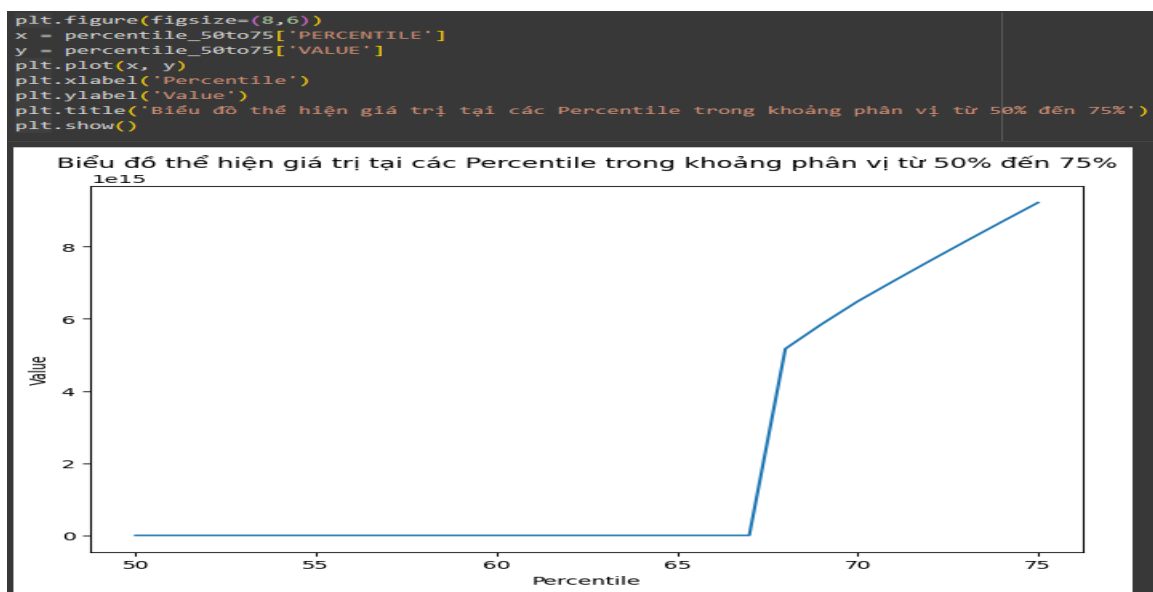
Hình 20 Kiểm tra giá trị trong phân vị nhất định


```
print(percentile_50to75)
```

	PERCENTILE	VALUE
0	50	1611.67
1	51	1681.07
2	52	1757.34
3	53	1838.78
4	54	1929.75
5	55	2031.51
6	56	2115.48
7	57	2206.56
8	58	2308.72
9	59	2426.55
10	60	2561.35
11	61	2721.22
12	62	2918.47
13	63	3168.50
14	64	3509.73
15	65	4026.65
16	66	4757.99
17	67	7494.73
18	68	5170829999999995.00
19	69	5849449999999990.00
20	70	6482049999999990.00
21	71	7047900000000000.00
22	72	7604200000000000.00
23	73	8151199999999995.00
24	74	8690450000000000.00
25	75	9221449999999996.00

Hình 21 Kết quả ra được

- 3 hàm trên được dùng để hiển thị và lưu lại các giá trị của cột TOTALBASKET tại vị trí phân vị nhất định trong khoảng từ 50% đến 75%.
- Ta có thể thấy rõ giá trị tại phân vị 65% nhảy lên giá trị tại phân vị 67% đáng kể như thế nào, và sự khác biệt rất rõ ràng giữa giá trị tại phân vị 67% và 68%.
- Từ các giá trị trên mà ta lưu lại được, ta tiến hành đưa vào biểu đồ đường để trực quan hóa sự thay đổi giá trị của cột TOTALBASKET trên từng phân vị trong khoảng kể trên.



Hình 22 Biểu đồ thể hiện giá trị tại khoảng phân vị nhất định

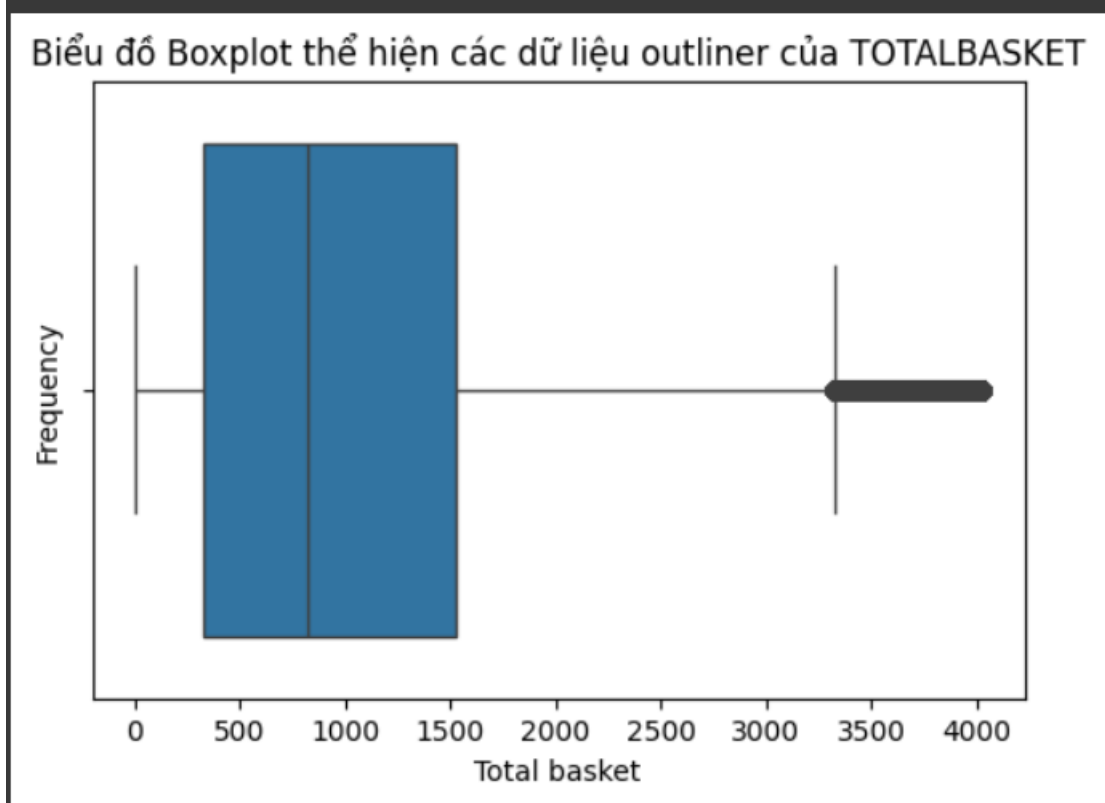
- Khi ta xác định được phân vị nào cần lấy (với phân vị sau đó chứa giá trị lớn hơn một cách bất thường), ta tiến hành giữ lại phần dữ liệu phía bên trái phân vị đó, tức gọt bỏ đi phần dữ liệu phía bên phải phân vị (lúc này ta coi như dữ liệu ngoại lệ), rồi tiến hành đề tập đã xử lý đó lên tập dữ liệu trước quá trình loại bỏ dữ liệu ngoại lệ.

```
df_orders = df_orders[df_orders['TOTALBASKET'] < np.percentile(df_orders['TOTALBASKET'], 65, interpolation='midpoint')]
```

Hình 23 Gọt tập dữ liệu

- Đến đây, coi như quá trình tiền xử lý đã hoàn thành.

```
plt.figure(figsize=(6,4))
sns.boxplot(x=df_orders['TOTALBASKET'])
plt.xlabel('Total basket')
plt.ylabel('Frequency')
plt.title('Biểu đồ Boxplot thể hiện các dữ liệu outlier của TOTALBASKET')
plt.show()
```



Hình 24 Biểu đồ Boxplot thể hiện dữ liệu ngoại lệ sau tiền xử lý

- Ta có thể thấy rõ các vạch tại vị trí các phân vị 0%, 25%, 50%, 75% và 100% trên biểu đồ.

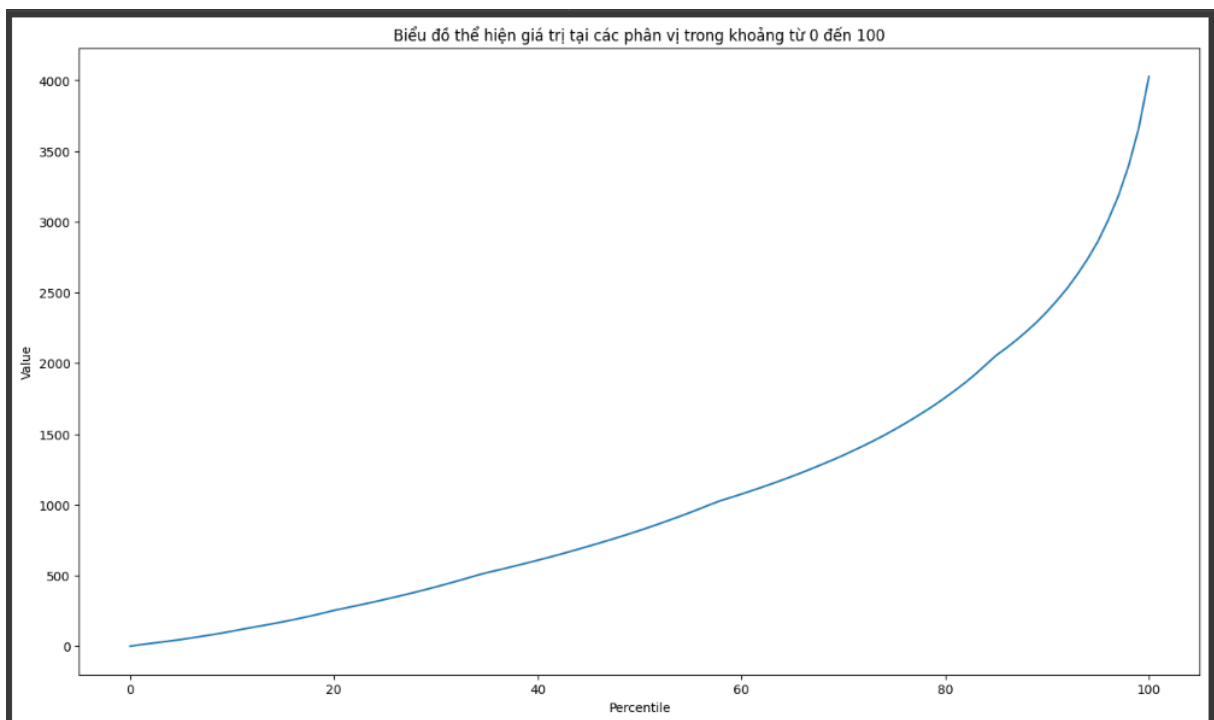
- Vì ta cần đảm bảo tính đa dạng của tập dữ liệu, và vì dữ liệu ngoại lệ đã được xử lý phần lớn, nên ta không cần xử lý dữ liệu ngoại lệ tiếp nữa.

```
percentile_0to100 = pd.DataFrame(columns=['PERCENTILE', 'VALUE'])

for i in range (0, 101):
    new_row = pd.DataFrame({
        'PERCENTILE': [i],
        'VALUE': [np.percentile(df_orders['TOTALBASKET'], i, interpolation='midpoint')]
    })
    percentile_0to100 = pd.concat([percentile_0to100, new_row], ignore_index=True)
    print("Value of total basket at the " + str(i) + "% percentile: ")
    print(np.percentile(df_orders['TOTALBASKET'], i, interpolation='midpoint'))
    print("\n")

plt.figure(figsize=(16, 9))
x = percentile_0to100['PERCENTILE']
y = percentile_0to100['VALUE']
plt.plot(x, y)
plt.xlabel('Percentile')
plt.ylabel('Value')
plt.title('Biểu đồ thể hiện giá trị tại các Percentile trong khoảng từ 75 đến 100')
plt.show()
```

Hình 25 Các hàm để xuất ra biểu đồ



Hình 26 Biểu đồ thể hiện giá trị doanh thu tại khoảng phân vị nhất định

Ba hàm bên trên thực hiện hiện thị và lưu lại các giá trị với các phân vị nhất định giống như ba hàm phía trên, với khác biệt là ta đang sử dụng tập dữ liệu sau tiền xử lý, và khoảng phân vị là từ 0% đến 100%.

```
df_orders.info()

<class 'pandas.core.frame.DataFrame'>
Index: 681574 entries, 1 to 1048573
Data columns (total 3 columns):
#   Column          Non-Null Count  Dtype  
---  -
0   BRANCH_ID       681574 non-null object  
1   DATE_           681574 non-null object  
2   TOTALBASKET     681574 non-null float64  
dtypes: float64(1), object(2)
memory usage: 20.8+ MB
```

Hình 27 Kiểu dữ liệu

- Tập dữ liệu có:
 - + BRANCH_ID và DATE_ vẫn theo kiểu dữ liệu Object (String) như cũ;
 - + TOTALBASKET theo kiểu dữ liệu float64 (float) thay cho Object (String) trước đó.
- Tập dữ liệu hao tốn một lượng tài nguyên hơn 21.4 MB.
- Hàm bao gồm cột DATE_ (với kiểu dữ liệu datetime), và cột TOTALBASKET (chứa kiểu dữ liệu float), với tổng cộng 702545 dòng trong tập dữ liệu;

```
df_orders.describe()

TOTALBASKET
count      681574.00
mean       1048.78
std        887.62
min         0.00
25%        331.13
50%        818.14
75%       1530.20
max       4026.65
```

Hình 28 Các thông số của TOTALBASKET

- Xét riêng cột TOTALBASKET:
 - + Cột có giá trị trung bình là 1166.84;

- + Cột có giá trị nhỏ nhất và lớn nhất lần lượt là 0 và 1152.58;
- + Giá trị tại các phân vị 25%, 50%, và 75% lần lượt là 343.82, 855.9, và 1628.55.

Chương III: PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU VÀ XÂY DỰNG MÔ HÌNH DỰ ĐOÁN

3.1 Phương pháp khai phá dữ liệu

3.1.1 Khái niệm về khai phá dữ liệu

Phân tích khám phá dữ liệu (Exploratory Data Analysis – EDA) là bước đầu tiên trong quy trình khoa học dữ liệu nhằm hiểu rõ đặc điểm, cấu trúc và mối quan hệ giữa các biến trong tập dữ liệu trước khi xây dựng mô hình.

EDA không chỉ dừng ở việc thống kê cơ bản mà còn bao gồm trực quan hóa, kiểm tra phân phối, xác định xu hướng và phát hiện bất thường (outliers).

Đây được xem là giai đoạn “đặt nền móng” cho toàn bộ dự án, giúp người phân tích hiểu dữ liệu trước khi chọn thuật toán hoặc kỹ thuật dự báo phù hợp.

3.1.2 Các phương pháp khai phá dữ liệu

Phương pháp khai phá dữ liệu có rất nhiều trong đó nổi bật nhất gồm các phương pháp:

- Phương pháp phân loại (Classification) – Dự đoán nhãn rời rạc
- Phương pháp hồi quy (Regression) – Dự đoán giá trị liên tục
- Phương pháp phân cụm (Clustering) – Gom nhóm đối tượng tương đồng
- Luật kết hợp (Association Rules) – Tìm quan hệ đồng xuất hiện của các mẫu
- Phương pháp phân tích chuỗi thời gian (Time Series Forecasting) – ARIMA/SARIMA, Prophet, LSTM
- Mô hình xác suất cho CLV – BG/NBD, Gamma-Gamma
- Đánh giá và giải thích mô hình - cross-validation, MAE/RMSE/AUC

3.2 Xây dựng mô hình dự đoán

Trong phạm vi dự án, với đề tài xây dựng mô hình dự đoán chi tiêu của khách hàng và dự đoán doanh thu của từng chi nhánh. Các mô hình được lựa chọn để thực hiện:

- Dự đoán chi tiêu của khách hàng : Sử dụng mô hình BG-NBD, Gamma Gamma để dự đoán chi tiêu khách hàng trong tương lai + mô hình K-mean để phân hạng khách hàng.
- Dự đoán doanh thu của từng chi nhánh: Sử dụng mô hình Prophet và XGBoost để dự đoán doanh thu theo thời gian.

3.2.1. Dự đoán chi tiêu của khách hàng

Mô hình sử dụng: BG-NBD dùng để dự đoán số lần mua hàng trong tương lai của khách hàng, Gamma – Gamma để tính số tiền mà khách hàng tiêu trong mỗi lần mua ở tương lai. Kết hợp mô hình BG-NBD và Gamma – Gamma để tính được chỉ số CLV của khách hàng từ đó sử dụng K-Means để phân loại khách hàng.

Chuẩn bị dữ liệu:

Mô hình BG-NBD và Gamma – Gamma sử dụng dữ liệu dạng RFM cùng với tham số T. Đây là dạng mô hình tóm tắt hành vi mua hàng của khách hàng trong đó gồm có 4 chỉ số: Recency, Frequency, Monetary và T, trong đó:

- Recency: Là khoảng thời gian từ lúc mua đầu tiên đến lần mua cuối cùng
- T: Là thời gian từ lúc mua hàng đầu tiên đến lúc thời gian thực hiện quan sát
- Frequency: Tổng số lần mua lặp lại (chỉ tính khi số lần mua > 1).
- Monetary: Thu nhập trung bình trên mỗi lần mua.

```

#Sử dụng biến today_date để tính ngày theo đối
#Giả sử ngày thực hiện theo đối là 1 tuần sau từ ngày cuối cùng (21-08-2023)

today_date = dt.datetime(2023, 8, 21)

#Nhóm các hóa đơn của user lại và tạo dataframe theo RFM
df_clv = df.groupby("USERID").agg({"DATE_" : [lambda date: (date.max() - date.min()).days, # recency
                                     lambda date: (today_date - date.min()).days], # T
                                   "ORDERID" : lambda order: order.nunique(), # frequency
                                   "TOTALBASKET" : lambda total_basket: total_basket.sum()}) # monetary

```

```

[30]: #Kiểm tra dataframe của mô hình
df_clv.head()

```

```

[30]:
      DATE_  ORDERID  TOTALBASKET
USERID
1      937      945      111      155132.19
2      876      906      112      171249.34
3      941      953      86      117615.85
4      936      948      97      133681.48
5      872      961      103      178823.71

```

Hình 29 Chuyển đổi mô hình RFM

Sau khi đã chuyển đổi sang mô hình RFM bằng cách groupby các hóa đơn của USERID lại cho từng khách hàng.

```
#Xử lý tên cột
```

```
df_clv.columns = df_clv.columns.droplevel(0)
df_clv.head()
```

	<lambda_0>	<lambda_1>	<lambda>	<lambda>
USERID				
1	937	945	111	155132.19
2	876	906	112	171249.34
3	941	953	86	117615.85
4	936	948	97	133681.48
5	923	951	102	128830.21

+ Code

+ Markdown

```
#Đặt tên cho các cột
```

```
df_clv.columns = ["recency", "T", "frequency", "monetary"]
df_clv.head(10)
```

	recency	T	frequency	monetary
USERID				
1	937	945	111	155132.19
2	876	906	112	171249.34
3	941	953	86	117615.85
4	936	948	97	133681.48
5	923	951	102	128830.21
6	944	957	119	149825.63
7	922	937	109	149994.38
8	919	932	111	169805.79
9	933	946	105	127282.41
10	926	958	94	101864.70

Hình 30 Đặt tên cho các cột của RFM


```
[33]: #Kiểm tra các Recency Frequency T thấp nhất và cao nhất

print(f"Recency (Min,Max): ({df_clv['recency'].min()}, {df_clv['recency'].max()})")
print(f"Frequency (Min,Max): ({df_clv['frequency'].min()}, {df_clv['frequency'].max()})")
print(f"T (Min,Max): ({df_clv['T'].min()}, {df_clv['T'].max()})")
print(f"Monetary (Min,Max): ({df_clv['monetary'].min()}, {df_clv['monetary'].max()})")

Recency (Min,Max): (817, 955)
Frequency (Min,Max): (62, 202)
T (Min,Max): (860, 962)
Monetary (Min,Max): (59802.35, 258721.65999999997)
```

Hình 31 Thông tin về dữ liệu mới

Đoạn code trên sử dụng để đặt lại tên cho các cột và hiện giờ có thể nhìn thấy rõ hơn về nội dung của dữ liệu.

Trong bộ dữ liệu này ta có thể thay Recency và T cả 2 có giá trị min và max khá cao nên cần phải xử lý để giảm vùng giá trị khi đem vào huấn luyện model. Giải pháp là chuyển sang dạng tuần hoặc tháng thay vì dạng ngày từ đó dữ liệu sẽ giảm đi đáng kể, ở đây sẽ sử dụng chuyển sang dạng tuần.

Frequency: giá trị min là 62 nên có thể thấy là không có khách hàng vắng lai mà chỉ có những khách hàng thân thiết mua hàng rất nhiều lần trong suốt hơn 955 ngày.

Monetary: Giá trị của Monetary khá cao khi min là 59.802 TRY(đơn vị tiền Thổ Nhĩ Kỳ), giá trị này là giá trị tổng của tất cả đơn hàng của người mua trong suốt 955 ngày. Phải tính lại giá trị Monetary này bằng cách lấy Monetary / Frequency để tìm ra số tiền chi tiêu trung bình của khách hàng trong 1 lần mua.

```
# Tính số tiền khách hàng sử dụng mỗi lần mua
df_clv["monetary"] = df_clv["monetary"] / df_clv["frequency"]
df_clv.head(10)
```

[34]:

	recency	T	frequency	monetary
USERID				
1	937	945	111	1397.587297
2	876	906	112	1529.011964
3	941	953	86	1367.626163
4	936	948	97	1378.159588
5	923	951	102	1263.041275
6	944	957	119	1259.038908
7	922	937	109	1376.095229
8	919	932	111	1529.781892
9	933	946	105	1212.213429
10	926	958	94	1083.667021

+ Code + Markdown

```
#Kiểm tra số tiền nhỏ nhất và lớn nhất mà khách hàng chi trả ở 1 lần mua
print(f"Monetary (Min,Max): ({round(df_clv['monetary'].min(),2)}, {round(df_clv['monetary'].max(),2)})")
```

Monetary (Min,Max): (776.65, 1813.49)

Hình 31 Tính giá trị chi trả trung bình của khách hàng

```
# Chuyển đổi Giá trị của Recency và T sang dạng Week
df_clv["recency"] = round(df_clv["recency"] / 7) #1 week = 7 day
df_clv["T"] = round(df_clv["T"] / 7)
```

```
[37]: df_clv.head()
```

[37]:

	recency	T	frequency	monetary
USERID				
1	134.0	135.0	111	1397.587297
2	125.0	129.0	112	1529.011964
3	134.0	136.0	86	1367.626163
4	134.0	135.0	97	1378.159588
5	132.0	136.0	102	1263.041275

+ Code + Markdown

Hình 32 Chuyển đổi Recency và T sang đơn vị tuần

Dữ liệu đã sẵn sàng, bắt đầu huấn luyện mô hình

Mô hình BG-NBD:

```
#Model BG-NBD
bgf = BetaGeoFitter(penalizer_coef = 0.001)

# Huấn luyện với bộ dữ liệu df_clv: frequency, recency, T
bgf.fit(df_clv["frequency"],
        df_clv["recency"],
        df_clv["T"])

: <lifetimes.BetaGeoFitter: fitted with 99996 subjects, a: 0.00, alpha: 18.65, b: 0.00, r: 14.01>
```

Hình 33 Train mô hình BG-NBD

Sử dụng hàm BetaGeoFitter để tạo mô hình BG-NBD sau đó gọi đến tham số `penalizer_coef` (Đây là chỉ số penalty score, chỉ số này giúp mô hình dự đoán có thêm điểm phạt để tinh chỉnh giá trị dự đoán không quá thấp hoặc không quá cao so với dữ liệu thực tế) với giá trị 0.001. Chỉ số này càng nhỏ thì mô hình sẽ dự đoán các số cao hơn. Ngược lại nếu chỉ số này cao thì nhưng lần dự đoán giá trị dự đoán sẽ thấp hơn giá trị gốc.

Đưa mô hình RFM vào trong BG-NBD để huấn luyện mô hình. Các đặc trưng dùng để huấn luyện là : Recency, Frequency và T. Sau khi đã huấn luyện mô hình thực hiện dự đoán trong 3 tháng người dùng đó sẽ mua bao nhiêu lần.

```
]: #Lưu dữ liệu tần suất mua hàng của khách hàng trong 3 tháng tới

df_clv["pred_freq_3_month"] = round(bgf.predict(4*3,
                                                df_clv["frequency"],
                                                df_clv["recency"],
                                                df_clv["T"]),2)
```

+ Code

+ Markdown

```
]: df_clv.head()
```

```
]:      recency      T frequency  monetary  pred_freq_3_month
USERID
1      134.0    135.0      111  1397.587297              9.76
2      125.0    129.0      112  1529.011964             10.24
3      134.0    136.0       86  1367.626163              7.76
4      134.0    135.0       97  1378.159588              8.67
5      132.0    136.0      102  1263.041275              9.00
```

Hình 34 Dự đoán số lần mua của khách hàng trong tương lai

Sau khi đã thử dự đoán số lần mua hàng của khách trong 3 tháng thì ta chuyển sang mô hình tiếp theo.

Mô hình Gamma Gamma:

Với mô hình này mục tiêu là dự đoán được khả năng chi tiêu của khách hàng trong khoảng bao nhiêu ở mỗi lần mua.

```
[52]: # Mô hình GammaGamma
      ggf = GammaGammaFitter(penalizer_coef = 0.01)

      # Fitting dữ liệu cho model với frequency và monetary
      ggf.fit(df_clv["frequency"], df_clv["monetary"])

[53]: <lifetimes.GammaGammaFitter: fitted with 99996 subjects, p: 3.27, q: 0.22, v: 3.27>

      + Code      + Markdown

[55]: #Thêm dự đoán giá trị trung bình của khách hàng mang lại cho cửa hàng vào df_clv

      df_clv['predict_monetary'] = ggf.conditional_expected_average_profit(df_clv["frequency"],df_clv["monetary"])
```

```
> df_clv
```

```
[56]:
```

	recency	T	frequency	monetary	pred_freq_3_month	pred_freq_6_month	pred_freq_12_month	predict_monetary
USERID								
1	134.0	135.0	111	1397.587297	9.76	19.53	39.05	1400.622249
2	125.0	129.0	112	1529.011964	10.24	20.48	40.96	1532.299857
3	134.0	136.0	86	1367.626163	7.76	15.52	31.04	1371.462606
4	134.0	135.0	97	1378.159588	8.67	17.34	34.68	1381.585828
5	132.0	136.0	102	1263.041275	9.00	18.00	36.01	1266.029716
...
99996	136.0	137.0	98	1211.244388	8.64	17.27	34.54	1214.228904
99997	133.0	134.0	94	1018.442447	8.49	16.98	33.96	1021.064497
99998	133.0	136.0	105	1400.949238	9.23	18.47	36.94	1404.165654
99999	132.0	133.0	97	1280.905670	8.78	17.57	35.14	1284.092509
100000	131.0	136.0	88	1206.036136	7.92	15.83	31.66	1209.346587

Hình 35 Fitting dữ liệu cho mô hình Gamma Gamma

Sau khi đã huấn luyện mô hình thực hiện dự đoán CLV cho khách hàng trong 3 tháng kế tiếp.

```
> #Dự đoán CLV của khách hàng bằng hàm customer_lifetime_value.
#Hàm customer_lifetime_value dùng để tính toán giá trị clv bằng mô hình BG-NBG cho phần dự đoán tần suất mua hàng
# Sau đó sử dụng ggf để dự đoán giá trị trung bình mà khách hàng đó mang lại cho cửa hàng

# CLV = (pred_freq) * (avarage_monetary)
CLV = ggf.customer_lifetime_value(bggf,
                                  df_clv["frequency"],
                                  df_clv["recency"],
                                  df_clv["T"],
                                  df_clv["monetary"],
                                  time = 3, #Thời gian dự đoán (Month)
                                  freq = "W") # Week
```

```
[58]: CLV.head(10)
```

```
[58]: USERID
1      14561.573062
2      16710.533594
3      11333.170927
4      12755.042698
5      12135.680965
6      13776.196611
7      14201.251341
8      16149.094852
9      12025.476159
10     9633.872532
Name: clv, dtype: float64
```

Hình 36 Dự đoán CLV cho khách hàng trong 3 tháng kế tiếp

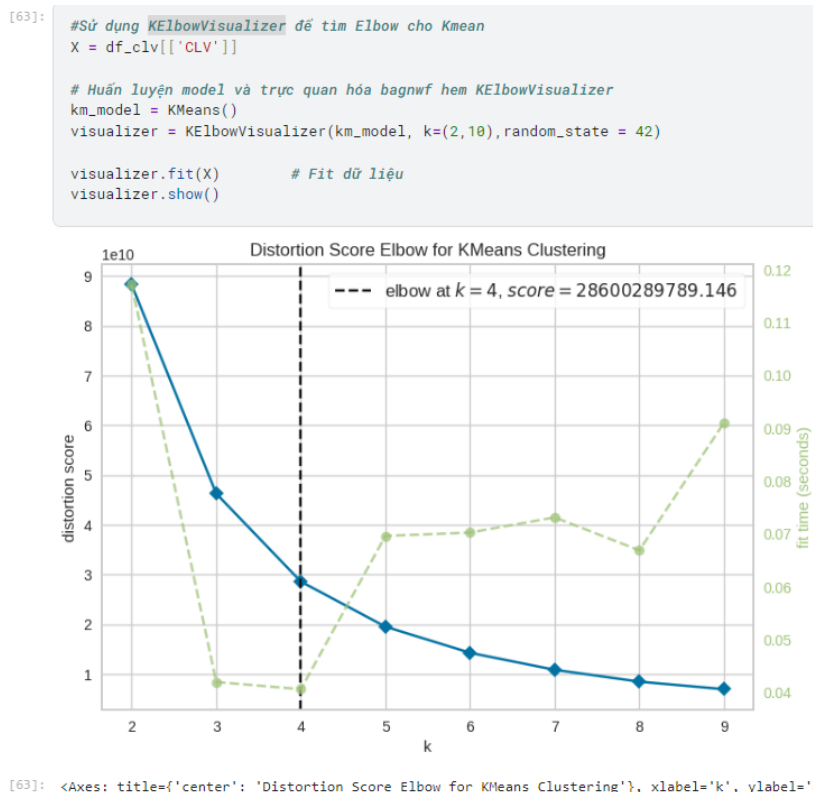
Ta có thể thấy trong vòng 3 tháng khách hàng có mã USERID = 1 chi tiêu khoảng 14.561 TRY cho cửa hàng, trong đó khách hàng số 2 thì có chỉ số CLV cao nhất với 16.710 TRY trong 3 tháng. Ngược lại khách hàng số 10 là khách hàng chi tiêu ít nhất khoảng 9.633 TRY trong thời gian 3 tháng kể tiếp.

Nhìn thấy được sự khác biệt trên nên chúng ta sẽ khai thác thêm thông tin từ dữ liệu CLV đó. Chẳng hạn như dùng để phân hạng khách hàng dựa trên điểm số CLV từ đó đưa ra được những chiến lược hợp lý.

Phân hạng khách hàng:

Sử dụng mô hình K-Means để có thể phân loại khách hàng dựa trên chỉ số CLV. Nhưng để trước tiên thực hiện phân cụm thì sử dụng Elbow Method kiểm tra xem k tối ưu nhất khi phân số cụm là bao nhiêu.

- Elbow Method: Sử dụng hàm KElbowVisualizer để tìm chỉ số k cho mô hình K-means



Hình 37 Sử dụng Elbow Method để tìm chỉ số k cho mô hình K-Means

Có thể nhìn thấy rõ là chỉ số k phù hợp với mô hình K-Means cho bộ dữ liệu này là $k = 4$.

Sau khi đã có được chỉ số k tối ưu thực hiện việc huấn luyện mô hình K-Means sau đó phân hạng khách hàng

- Mô hình K-Means: Huấn luyện mô hình phân cụm K-Means với tham số $k = 4$, cùng với đó là `random_state = 42` để không bị lấy mẫu ngẫu nhiên mỗi lần chạy

```
[64]: #Sử dụng Kmean để phân cụm khách hàng

km_model = KMeans(n_clusters=4, random_state = 42, n_init = 10)
km_model.fit(X)

df_clv['cluster'] = km_model.labels_

+ Code + Markdown

[65]: df_clv
```

	recency	T	frequency	monetary	pred_freq_3_month	pred_freq_6_month	pred_freq_12_month	predict_monetary	CLV	cluster
USERID										
1	134.0	135.0	111	1397.587297	9.76	19.53	39.05	1400.622249	14561.573062	2
2	125.0	129.0	112	1529.011964	10.24	20.48	40.96	1532.299857	16710.533594	2
3	134.0	136.0	86	1367.626163	7.76	15.52	31.04	1371.462606	11333.170927	1
4	134.0	135.0	97	1378.159588	8.67	17.34	34.68	1381.585828	12755.042698	0
5	132.0	136.0	102	1263.041275	9.00	18.00	36.01	1266.029716	12135.680965	1
...
99996	136.0	137.0	98	1211.244388	8.64	17.27	34.54	1214.228904	11165.619707	1
99997	133.0	134.0	94	1018.442447	8.49	16.98	33.96	1021.064497	9231.976287	3
99998	133.0	136.0	105	1400.949238	9.23	18.47	36.94	1404.165654	13807.871893	0
99999	132.0	133.0	97	1280.905670	8.78	17.57	35.14	1284.092509	12011.312604	1
100000	131.0	136.0	88	1206.036136	7.92	15.83	31.66	1209.346587	10193.367526	3

99996 rows x 10 columns

Hình 38 Mô hình K-Means phân loại khách hàng

Kiểm tra xem các giá trị CLV trung bình của từng cụm và phân loại khách hàng theo hạng : Bronze, Silver, Gold, Diamond theo thứ hạng từ thấp đến cao.

```

#Kiểm tra xem thông tin của các cụm

df_clusters = df_clv.groupby(['cluster'])['CLV'].agg(['mean', 'count']).reset_index()

df_clusters.columns = ["cluster", "avg_CLV", "n_customers"]

# Tỷ lệ các khách hàng của cụm

df_clusters['percent_customers'] = (df_clusters['n_customers']/df_clusters['n_customers'].sum())*100
df_clusters

```

```

[66]:
  cluster  avg_CLV  n_customers  percent_customers
0      0  12951.914983      31613          31.614265
1      1  11527.085128      35818          35.819433
2      2  14712.888712      13309          13.309532
3      3   9962.727745      19256          19.256770

```

Hình 39 Kiểm tra thông tin các cụm

```

[67]:
# Phân loại khách hàng với avg_CLV:
# Sắp xếp theo avg_CLV tăng dần
df_cluster = df_clusters.sort_values("avg_CLV")

# Gán thứ hạng theo thứ tự
labels = ["Bronze", "Silver", "Gold", "Diamond"]
df_cluster["segment"] = labels[:len(df_cluster)]

df_cluster

```

```

[67]:
  cluster  avg_CLV  n_customers  percent_customers  segment
3      3   9962.727745      19256          19.256770   Bronze
1      1  11527.085128      35818          35.819433   Silver
0      0  12951.914983      31613          31.614265    Gold
2      2  14712.888712      13309          13.309532   Diamond

```

```

[69]:
mapping = dict(zip(df_cluster["cluster"], df_cluster["segment"]))
df_clv["customer_category"] = df_clv["cluster"].map(mapping)

```

+ Code + Markdown

```

[70]:
df_clv.head(10)

```

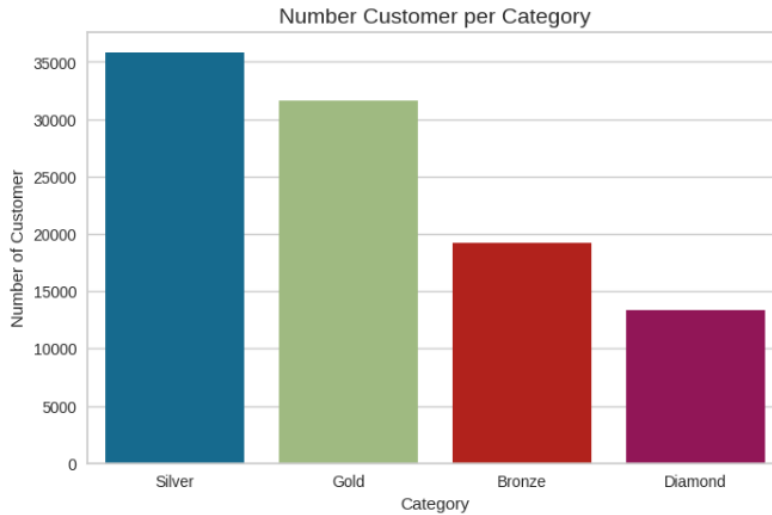
```

[70]:
  recency    T  frequency  monetary  pred_freq_3_month  pred_freq_6_month  pred_freq_12_month  predict_monetary  CLV  cluster  customer_category
USERID
1  134.0  135.0    111  1397.587297          9.76          19.53          39.05      1400.622249  14561.573062    2      Diamond
2  125.0  129.0    112  1529.011964         10.24         20.48         40.96      1532.299857  16710.533594    2      Diamond
3  134.0  136.0     86  1367.626163          7.76         15.52         31.04      1371.462606  11333.170927    1      Silver
4  134.0  135.0     97  1378.159588          8.67         17.34         34.68      1381.585828  12755.042698    0      Gold
5  132.0  136.0    102  1263.041275          9.00         18.00         36.01      1266.029716  12135.680965    1      Silver

```

Hình 40 Phân loại khách hàng


```
3]: #Đồ thị Bar số lượng phân khúc khách hàng
plt.figure(figsize=(8,5))
sns.barplot(x=df_clv['customer_category'].value_counts().index, y=df_clv['customer_category'].value_counts().values)
plt.title('Number Customer per Category', fontsize=14)
plt.xlabel('Category')
plt.ylabel('Number of Customer')
plt.show()
```



Hình 41 Số lượng khách hàng mỗi phân khúc

Như vậy dữ liệu về hóa đơn của khách hàng đã được xử lý và dự đoán khả năng chi tiêu của khách hàng trong 3 tháng. Cùng với đó nhờ vào data đã dự đoán ta có thể làm thêm các tính năng mới như phân loại khách hàng từ đó hỗ trợ rất nhiều cho việc lên các chiến lược tiếp thị cũng như thống kê số lượng khách hàng thân thiết.

3.2.2. Dự đoán doanh thu của từng chi nhánh

3.2.2.1. Mô hình Prophet

```
from prophet import Prophet
```

Hình 42 Nhập mô hình Prophet

- Tạo mảng "all_branch_forecasts" để chứa các dữ liệu được dự đoán dựa trên các tiêu chí nhất định;
- Mảng trên sẽ được sử dụng để đối chiếu với dữ liệu thực tế nhằm đánh giá độ chính xác của mô hình từ việc đối chiếu dữ liệu dự đoán với thực tế.

- Từ đây, ta bắt đầu chạy vòng lặp "for" để huấn luyện mô hình theo từng BRANCH_ID nhất định:

```
all_branch_forecasts = {}

for branch_id in df_orders['BRANCH_ID'].unique():
    print(f"Processing branch: {branch_id}")
    df_branch = df_orders[df_orders['BRANCH_ID'] == branch_id].copy()
    df_branch = df_branch.rename(columns={'DATE_': 'ds', 'TOTALBASKET': 'y'})
    df_branch = df_branch.groupby('ds')['y'].sum().reset_index()

    model_prophet = Prophet()
    model_prophet.fit(df_branch)

    #predicting the total basket 1 month ahead
    future = model_prophet.make_future_dataframe(periods=1, freq='M')
    forecast = model_prophet.predict(future)

    all_branch_forecasts[branch_id] = forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
    print("\n\n")
```

Hình 43 Huấn luyện mô hình Prophet

- Vòng lặp tạo một tập dữ liệu DataFrame mới là df_branch gồm các đối tượng chung một mã nhánh (BRANCH_ID), tương ứng với một địa chỉ duy nhất. Nó hoạt động tương tự DataFrame x, bao gồm cả x_train và x_test.
- Vòng lặp sau đó tiến hành đổi tên cột mục tiêu thành "y", và cột ngày thành "ds".
 - + Mô hình Prophet chỉ chấp nhận các cột với tên được nêu trên, tức nếu để tên cột nào khác thì sẽ sinh ra thông báo lỗi;
 - + Các cột được gán tên "ds" đóng vai trò như dữ liệu hỗ trợ cho cột mục tiêu "y", giống như tập x trong quá trình tách tập dữ liệu thành x và y trong quá trình huấn luyện mô hình dạng cây.
- Sau đó vòng lặp gộp các dữ liệu/dòng lại theo ngày (ds) và tính tổng doanh thu/lợi nhuận (y) của các dòng trên. Quá trình trên áp dụng cho toàn bộ tập dữ liệu df_branch đó. Tiếp theo, chức năng reset_index() đưa dữ liệu đã gộp lại về dạng DataFrame và viết đè lên tập kể trên.
- Một mô hình Prophet với mô-đun cùng tên được tạo ra, và tập dữ liệu df_branch được đưa vào mô hình trên cho việc huấn luyện.
- Sau đó, tạo DataFrame tên "future" chỉ gồm các ngày/tháng/năm trong tương lai, và không có giá trị TOTALBASKET gì trong đó, với số đơn vị thời gian tương lai là 1, và đơn vị thời gian là "M", tức là tháng. Nó hoạt động tương tự y_train.
- Tiếp theo, tạo DataFrame tên "forecast" để lưu lại kết quả dự đoán của mô hình, với quá trình huấn luyện từ trước, và tập dữ liệu ngày "future" cho việc dự đoán. Nó có chức năng tương tự y_test.

- Và rồi, tạo một mảng chứa các đối tượng kết quả thu được từ quá trình huấn luyện, và đưa kết quả vào đó. Một đối tượng trên bao gồm:
 - + Mã nhánh mà ta dùng để gom dữ liệu trước đó;
 - + dt: Ngày tháng năm;
 - + yhat: Giá trị dự đoán (trong khoảng giá trị dự đoán nhất định);
 - + yhat_lower và yhat_upper: Lần lượt là cực tiểu và cực đại của khoảng giá trị dự đoán nhất định.

3.2.2.1. Mô hình XGBoost

```
from xgboost import XGBRegressor
```

Hình 44 Nhập mô hình XGBoost

- Tạo hàm tạo các đặc điểm cho các tập dữ liệu x_train, x_test, y_train, và y_test. Gắn như đặc điểm đều liên quan đến ngày tháng năm, riêng "BRANCH_ID" (đã được mã hóa nhãn) là về địa chỉ nơi nhánh cửa hàng được đặt.

```
def create_features(df):
    df['day'] = df['DATE_'].dt.day
    df['month'] = df['DATE_'].dt.month
    df['year'] = df['DATE_'].dt.year
    df['dayofweek'] = df['DATE_'].dt.dayofweek
    df['quarter'] = df['DATE_'].dt.quarter
    df['dayofyear'] = df['DATE_'].dt.dayofyear
    df['dayofmonth'] = df['DATE_'].dt.day
    df['weekofyear'] = df['DATE_'].dt.isocalendar().week
    df['BRANCH_ID'] = label_encoder.fit_transform(df['BRANCH_ID'])

    x = df[['BRANCH_ID', 'day', 'month', 'year', 'dayofweek', 'quarter', 'dayofyear', 'dayofmonth', 'weekofyear']]
    return x
```

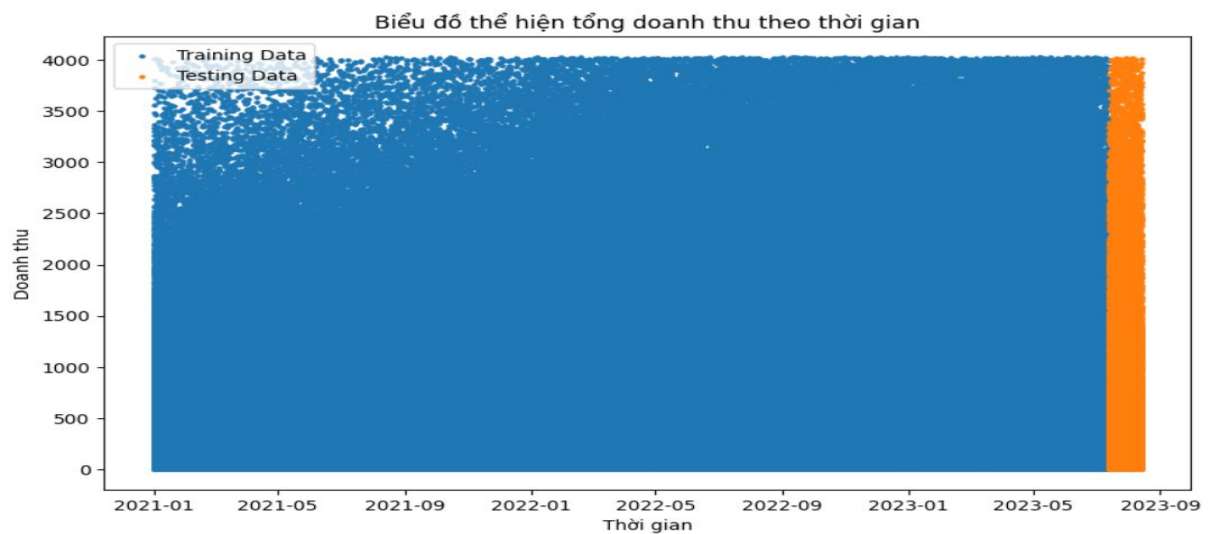
Hình 45 Hàm tạo đặc điểm

- Tách tập dữ liệu Orders làm hai phần, với tập kiểm tra nằm trong khoảng 1 tháng kể từ ngày muộn nhất về phía quá khứ, còn tập huấn luyện là phần còn lại của tập Orders.

```
split_date = '2023-07-14'
train = df_orders[df_orders['DATE_'] < split_date].reset_index(drop=True)
test = df_orders[df_orders['DATE_'] >= split_date].reset_index(drop=True)
```

Hình 46 Tách dữ liệu theo ngày

Trực quan hóa biểu đồ thể hiện doanh thu theo thời gian, với phần xanh dương thể hiện phần dữ liệu được dùng cho việc huấn luyện, và phần cam là phần dữ liệu cho việc kiểm tra.



Hình 47 Biểu đồ thể hiện tổng doanh thu theo thời gian

- Tạo tập dữ liệu `x_train`, `x_test`, `y_train`, và `y_test`, với:
 - + `x_train` và `y_train` đều dùng hàm tạo đặc điểm, với lần lượt tập huấn luyện và tập kiểm tra được áp dụng vào hàm trên;
 - + `y_train` và `y_test` đều là cột `TOTALBASKET` được lấy ra từ hai tập dữ liệu nêu trên.

```
x_train = create_features(train)
y_train = train['TOTALBASKET']
x_test = create_features(test)
y_test = test['TOTALBASKET']
```

Hình 48 Tách tập dữ liệu

- Tạo mô hình XGBoost mang tên "`model_xgb`", với số lần huấn luyện là 1000 lần. Sau đó, tiến hành huấn luyện mô hình "`model_xgb`", với cả 2 tập `x_train` và `y_train`, cũng như `x_test` và `y_test` trong mảng "`eval_set`".

```
model_xgb = XGBRegressor(n_estimators=1000)

model_xgb.fit(
    x_train,
    y_train,
    eval_set=[(x_test, y_test)],
    verbose=False
)
```

Hình 49 Huấn luyện mô hình

Chương IV: KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH

4.1 Dự đoán chi tiêu của khách hàng

4.1.1 Mô hình BG-NBD

[76]:

```
bgf.summary
```

[76]:

	coef	se(coef)	lower 95% bound	upper 95% bound
r	1.400869e+01	4.711272e-02	1.391635e+01	1.410103e+01
alpha	1.865106e+01	6.493704e-02	1.852378e+01	1.877834e+01
a	7.692565e-15	6.206876e-12	-1.215778e-11	1.217317e-11
b	6.644796e-05	4.994851e-02	-9.783263e-02	9.796553e-02

Hình 50 Bảng Summary của BG-NBD

Đây là bảng summary của BG-NBD được dùng để đánh giá mô hình dự đoán tần suất mua hàng của khách. Các tham số được chú thích như sau:

Cột	Giải thích
Coef	Giá trị ước lượng của từng tham số trong mô hình
se(coef)	Sai số chuẩn (standard error) – càng nhỏ, ước lượng càng ổn định
lower 95% bound / upper 95% bound	Khoảng tin cậy 95% cho tham số – giúp kiểm tra độ chắc chắn của ước lượng

Dòng	Giải thích
r	Tham số <i>shape</i> của phân phối Gamma mô tả mức độ biến thiên về tần suất mua hàng giữa các khách hàng.
alpha	Tham số <i>rate</i> của phân phối Gamma, đại diện cho tốc độ mua trung bình.
a	Tham số <i>shape</i> của phân phối Beta mô tả sự khác biệt trong xác suất “rời bỏ” giữa các khách hàng.
b	Tham số <i>rate</i> của phân phối Beta, thể hiện tốc độ trung bình mà khách hàng rời bỏ.


Bảng bgf.summary trả về bốn tham số: $r = 14.00869$ ($SE = 0.047$), $\alpha = 18.65106$ ($SE = 0.065$), $a \approx 7.69 \times 10^{-15}$ (≈ 0), và $b \approx 6.64 \times 10^{-5}$ (≈ 0).

Tham số r và α được ước lượng chính xác (khoảng tin cậy 95% hẹp), cho thấy tốc độ mua trung bình giữa các khách tương đối đồng đều.

Trung bình tốc độ mua ước tính là $r/\alpha \approx 0.75$ (đơn vị phụ thuộc đơn vị thời gian của T).

Tham số a và b gần bằng 0 và có khoảng tin cậy bao gồm 0. Điều này cho thấy mô hình không tìm thấy bằng chứng mạnh về tỷ lệ churn trong dữ liệu quan sát hoặc tham số này bị ước lượng ở biên.

4.1.2 Mô hình Gamma - Gamma



```
ggf.summary
```

[54]:

	coef	se(coef)	lower 95% bound	upper 95% bound
p	3.267184	0.016636	3.234578	3.299791
q	0.221789	0.000765	0.220289	0.223289
v	3.265142	0.016641	3.232526	3.297758

Hình 51 Bảng Summary của mô hình Gamma-Gamma

Đây là bảng summary của Gamma – Gamma được dùng để đánh giá mô hình dự đoán CLV của khách hàng. Các tham số được chú thích như sau:

Cột	Giải thích
Coef	Giá trị ước lượng của từng tham số trong mô hình
se(coef)	Sai số chuẩn (standard error) – càng nhỏ, ước lượng càng ổn định
lower 95% bound / upper 95% bound	Khoảng tin cậy 95% cho tham số – giúp kiểm tra độ chắc chắn của ước lượng

Dòng	Giải thích
p	Tham số <i>shape</i> mô tả sự khác biệt trong giá trị chi tiêu giữa các khách hàng.
q	Tham số <i>shape</i> mô tả mức độ biến động chi tiêu trong từng khách hàng.

v	Tham số <i>scale</i> biểu thị giá trị trung bình chi tiêu kỳ vọng của toàn bộ khách hàng.
---	---

- Sai số chuẩn ($se(coef)$) của cả ba tham số rất nhỏ ($\approx 0.016 - 0.0007$), chứng tỏ ước lượng ổn định và có ý nghĩa thống kê cao.
- Khoảng tin cậy 95% hẹp \rightarrow mô hình hội tụ tốt, không có hiện tượng nhiễu hoặc tham số cực đoan.
- Các tham số đều dương và nằm trong vùng hợp lệ \rightarrow mô hình hợp lý theo giả định Gamma-Gamma.
- Sự gần bằng giữa p và v ($\approx 3.26-3.27$) cho thấy phân phối chi tiêu có dạng gần đối xứng, không bị lệch mạnh về phía các khách hàng chi tiêu quá lớn.

Mô hình Gamma-Gamma đã hội tụ tốt và phản ánh đúng xu hướng chi tiêu của khách hàng:

- Mức chi tiêu trung bình ổn định,
- Chênh lệch chi tiêu giữa khách hàng thấp,
- Sai số mô hình nhỏ.

Khi kết hợp với mô hình BG-NBD, kết quả này cho phép tính CLV (Customer Lifetime Value) chính xác hơn, vì giá trị chi tiêu trung bình (do Gamma-Gamma dự đoán) đáng tin cậy.

4.2 Dự đoán doanh thu của từng chi nhánh

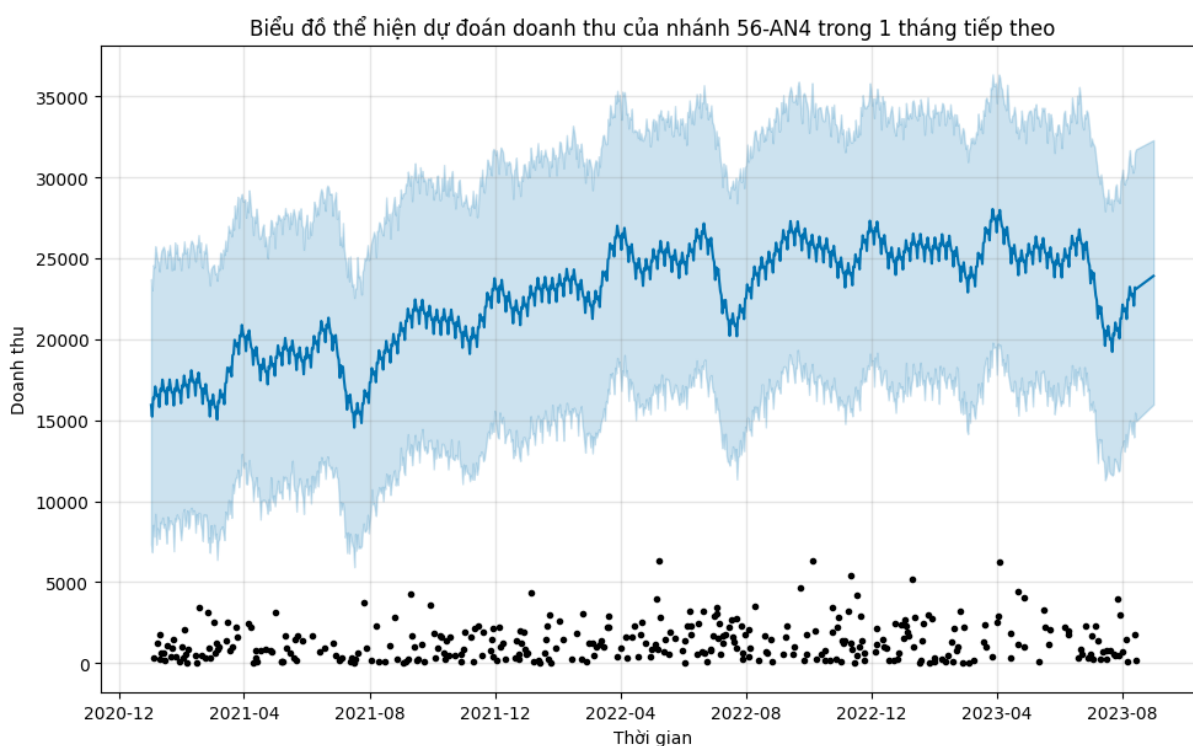
4.2.1. Mô hình Prophet

Thể hiện kết quả có được từ quá trình huấn luyện mô hình Prophet, được lưu trong mảng `all_branch_forecasts` chứa kết quả huấn luyện, bao gồm:

- Mã nhánh mà ta dùng để gom dữ liệu trước đó;
- dt: Ngày tháng năm;
- yhat: Giá trị dự đoán (trong khoảng giá trị dự đoán nhất định);
- yhat_lower và yhat_upper: Lần lượt là cực tiểu và cực đại của khoảng giá trị dự đoán nhất định.

{ '56-AN4' :		ds	yhat	yhat_lower	yhat_upper
0	2021-01-01	15943.37	7114.06	24478.17	
1	2021-01-02	15227.27	6569.78	23496.14	
2	2021-01-03	16389.10	8223.02	24275.20	
3	2021-01-04	16325.31	8036.55	25644.64	
4	2021-01-05	17083.18	8874.58	25641.57	
..	
954	2023-08-13	23206.06	15330.60	31860.89	
955	2023-08-14	23104.62	14447.96	31359.22	
956	2023-08-31	23911.62	15593.84	31949.03	
957	2023-09-30	23459.23	14278.92	31433.79	
958	2023-10-31	23312.02	15437.39	31427.02	

Hình 52 Mô tả dự đoán doanh thu



Hình 53 Biểu đồ dự đoán doanh thu

- Hàm trực quan hóa dữ liệu có từ trước và dữ liệu dự đoán từ chỗ dữ liệu có trước, với trục hoành thể hiện thời gian, và trục tung thể hiện doanh thu/lợi nhuận từ cột TOTALBASKET;
- Hàm trên sử dụng mô hình Prophet đã được huấn luyện từ trước, cũng như mảng all_branch_forecasts nêu trên;
- Trong trường hợp này, mã nhánh 56-AN4 được chọn để trực quan hóa mô hình, vì tính chất dự đoán theo thời gian của mô hình trên, tức không thể dự đoán các nhánh như mô hình cây được.

```
from sklearn.metrics import mean_absolute_error, root_mean_squared_error, r2_score
```

Hình 54 Nhập 3 mô hình điểm số

Nhập 3 mô hình điểm số MAE, RMSE, và R2 từ thư viện sklearn.metric.

- MAE (Mean Absolute Error): Chỉ số hồi quy, dùng để thể hiện độ lớn trung bình của các lỗi trong một tập hợp các dự đoán mà không cần xem xét hướng của chúng;
- RMSE (Root Mean Squared Error): Chỉ số hồi quy, là căn bậc hai của mức trung bình của các sai số bình phương, cũng như độ lệch chuẩn của các phần dư, tức sai số dự đoán;
- R2 (R-Squared): Biểu thị độ phù hợp của mô hình hồi quy.

```
for branch_id, forecast in all_branch_forecasts.items():
    df_branch = df_orders[df_orders['BRANCH_ID'] == branch_id].copy()
    df_branch['DATE_'] = pd.to_datetime(df_branch['DATE_'])
    df_branch = df_branch.rename(columns={'DATE_': 'ds', 'TOTALBASKET': 'y'})
    df_branch = df_branch.groupby('ds')['y'].sum().reset_index()

    merged_df = pd.merge(df_branch, forecast, on='ds', how='inner')

    mae = mean_absolute_error(merged_df['y'], merged_df['yhat'])
    rmse = root_mean_squared_error(merged_df['y'], merged_df['yhat'])
    r2 = r2_score(merged_df['y'], merged_df['yhat'])

    evaluation_results[branch_id] = {'MAE': mae, 'RMSE': rmse, 'R2': r2}
```

Hình 55 Vòng lặp để lấy điểm MAE, RMSE, và R2

Chạy vòng lặp "for" để lưu kết quả gồm 3 điểm trên vào mảng evaluation_results:

- Vòng lặp tạo một tập dữ liệu DataFrame mới là df_branch gồm các đối tượng chung một mã nhánh (BRANCH_ID), tương ứng với một địa chỉ duy nhất. Nó hoạt động tương tự DataFrame x, bao gồm cả x_train và x_test.
- Vòng lặp sau đó tiến hành đổi tên cột mục tiêu thành "y", và cột ngày thành "ds".
 - + Mô hình Prophet chỉ chấp nhận các cột với tên được nêu trên, tức nếu để tên cột nào khác thì sẽ sinh ra thông báo lỗi;
 - + Các cột được gán tên "ds" đóng vai trò như dữ liệu hỗ trợ cho cột mục tiêu "y", giống như tập x trong quá trình tách tập dữ liệu thành x và y trong quá trình huấn luyện mô hình dạng cây.
- Sau đó vòng lặp gộp các dữ liệu/dòng lại theo ngày (ds) và tính tổng doanh thu/lợi nhuận (y) của các dòng trên. Quá trình trên áp dụng cho toàn bộ tập dữ liệu df_branch đó. Tiếp theo, chức năng reset_index() đưa dữ liệu đã gộp lại về dạng DataFrame và viết đè lên tập kể trên.
- Tiếp theo, tạo một DataFrame mới mang tên "merged_df" từ việc gộp tập dữ liệu df_branch (đã nêu trên), cùng với biến dữ liệu dự đoán "forecast", theo ngày (ds);
- Sau đó, 3 điểm số trên được tạo, với cột y (tức TOTALBASKET/doanh thu/lợi nhuận), và cột "yhat" (tức giá trị dự đoán của TOTALBASKET) đều thuộc DataFrame "merged_df";

- Cuối cùng, trả về kết quả là 3 điểm số trên theo mã nhánh nhất định.

```
for branch_id, metrics in evaluation_results.items():  
    print(f"[Nhánh {branch_id}]")  
    print(f"MAE: {metrics['MAE']:.2f}")  
    print(f"RMSE: {metrics['RMSE']:.2f}")  
    print(f"R2: {metrics['R2']:.2f}")  
    print("\n\n")
```

Hình 56 Xuất ba điểm số trên

- Với vòng lặp "for", ta trả về 3 điểm số trên, với mã nhánh liên quan.

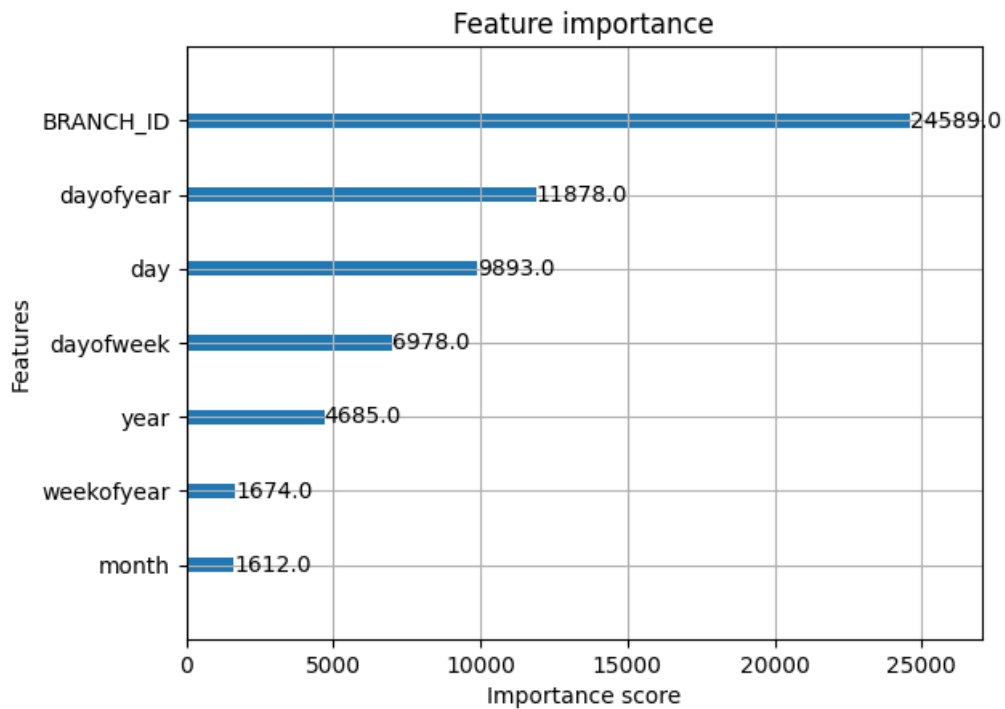
```
[Nhánh 56-AN4]  
MAE: 5130.04  
RMSE: 6472.31  
R2: 0.21  
  
[Nhánh 348-MU1]  
MAE: 1697.81  
RMSE: 2171.46  
R2: 0.05
```

Hình 57 Kết quả trả về

- Nhận xét: Tất cả đều có điểm MAE và RMSE lớn, đồng nghĩa với việc kết quả dự đoán khác xa so với kết quả thực tế. Hơn nữa, điểm R2 dù đều thuộc số dương, phần lớn lại trả về kết quả nhỏ hơn 0 (với một trong số đó là 0.06), nghĩa là kết quả trả về rất khó dự đoán được bất kỳ sự biến thiên trong mô hình, và cũng khó tìm thấy bất kỳ mối quan hệ nào giữa các biến phụ thuộc (tức DATE_ và BRANCH_ID), và biến độc lập (tức TOTALBASKET).
- Vậy, mô hình Prophet không phải là mô hình tốt cho việc dự đoán doanh thu / lợi nhuận trong một tháng tiếp theo theo chi nhánh.

4.2.2. Mô hình XGBoost

- Nhập mô hình "plot_importance" từ thư viện XGBoost. Mô hình này được sử dụng để kiểm tra độ quan trọng của các biến phụ thuộc.
- Áp dụng mô hình XGBoost đã được huấn luyện vào mô hình plot_importance để trực quan hóa độ quan trọng của các biến phụ thuộc.



Hình 58 Biểu đồ thể hiện độ quan trọng của biến phụ thuộc

- Như ta đã thấy, biến mã nhánh là biến phụ thuộc quan trọng nhất, và biến tháng là biến ít quan trọng nhất.
- Tạo biến chứa kết quả dự đoán của mô hình XGBoost là `y_pred = model_xgb.predict(x_test)`, với biến `x_test` làm biến đầu vào.
- Tạo ba điểm số RMSE, MAE, và R2, cũng như xuất các điểm số ra màn hình. Chức năng của ba điểm số trên đã được giải thích ở phần đánh giá mô hình Prophet.

```
rmse = np.sqrt(root_mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

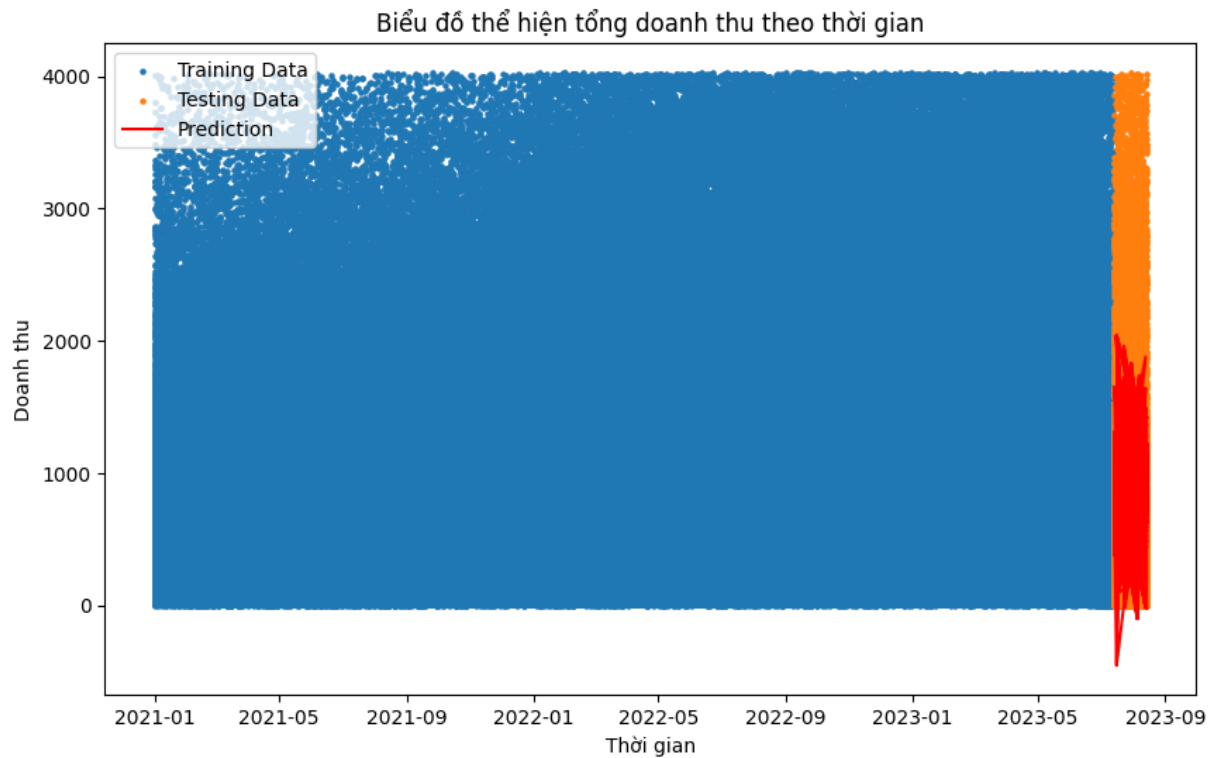
print(f"RMSE: {rmse:.2f}")
print(f"MAE: {mae:.2f}")
print(f"R2: {r2:.2f}")

RMSE: 29.50
MAE: 677.25
R2: -0.05
```

Hình 59 Ba điểm số đánh giá mô hình

- Điểm số RMSE và MAE vẫn cao, dù thấp hơn hai điểm trên của mô hình Prophet, và điểm R2 lần này ở mức âm, tức kết quả trả về rất khó dự đoán được bất kỳ sự biến thiên trong mô hình, và cũng khó tìm thấy bất kỳ mối quan hệ nào giữa các biến phụ thuộc (được nêu trong hàm `create_features`), và biến độc lập (tức `TOTALBASKET`).

- Trực quan hóa hai tập huấn luyện và kiểm tra như trên, đồng thời hiện thị kết quả dự đoán dưới dạng vạch đỏ phía trên tập kiểm tra.



Hình 60 Biểu đồ thể hiện tổng doanh thu theo thời gian

- Mô hình dự đoán ra các giá trị doanh thu trong khoảng đa số từ 0 đến 2000, với một số vạch ngay bên dưới số 0, tức giá trị âm. Điều này thể hiện rằng mô hình không thể hoàn toàn dự đoán chính xác các đơn hàng có doanh thu trên 2000, nhất là khi mô hình lại phán đoán ra doanh thu âm, tức giá trị mà ta không mong muốn.
- Vậy, Mô hình XGBoost cũng không phải là mô hình tốt cho việc dự đoán doanh thu / lợi nhuận trong một tháng tiếp theo theo chi nhánh, giống như mô hình Prophet.

CHƯƠNG V: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Dự đoán chỉ tiêu của khách hàng

Kết luận

Dự án sử dụng hai mô hình chính BG-NBD và Gamma-Gamma để ước lượng tần suất mua và giá trị giao dịch, từ đó tính CLV.

Kết quả chính: BG-NBD hội tụ với $r \approx 14.01$, $\alpha \approx 18.65 \rightarrow$ tốc độ mua trung bình $r/\alpha \approx 0.75$ (đơn vị theo T) và biến thiên giữa khách thấp; tham số churn a, b gần 0 nên ước lượng churn không chắc chắn. Gamma-Gamma hội tụ với $p \approx 3.27$, $q \approx 0.22$, $v \approx 3.27 \rightarrow$ giá trị chỉ tiêu trung bình ổn định và khác biệt giữa khách nhỏ.

Ứng dụng thực tiễn: kết quả đủ tin cậy để phân nhóm khách theo CLV, xác định nhóm ưu tiên giữ chân và tối ưu phân bổ ngân sách marketing. Tuy nhiên cần thận trọng khi dùng kết luận về churn do tham số a, b không ổn định và do thiếu tập kiểm thử độc lập.

Hướng phát triển

Thông qua dự án này, em rút ra được nhiều kinh nghiệm trong việc xử lý dữ liệu giao dịch khách hàng và áp dụng mô hình thống kê để dự đoán giá trị vòng đời khách hàng (CLV). Tuy nhiên, để nâng cao độ chính xác và khả năng tổng quát của mô hình, các hướng phát triển tiếp theo được đề xuất như sau:

- Tối ưu hóa dữ liệu đầu vào: Các dữ liệu hiện đang rất lớn nên cần phải tối ưu hơn để có thể giúp model học tốt hơn.
- Chia nhỏ dữ liệu để kiểm thử: Thực hiện chia dữ liệu thành tập *calibration* và *holdout* theo mốc thời gian, giúp đánh giá khả năng dự báo thực tế của mô hình thay vì chỉ kiểm tra trên dữ liệu huấn luyện.
- So sánh và thử nghiệm mô hình mới: Bên cạnh BG-NBD và Gamma-Gamma, có thể thử các mô hình khác như Pareto/NBD, Hierarchical Bayesian BG-NBD, hoặc Gradient Boosting (LightGBM/XGBoost) để so sánh hiệu suất dự báo CLV.
- Tự động hóa và mở rộng ứng dụng: Xây dựng quy trình tự động cập nhật dữ liệu và tái huấn luyện mô hình định kỳ. Ứng dụng kết quả CLV vào phân khúc khách hàng, tối ưu chiến lược marketing và đo lường hiệu quả giữ chân khách hàng theo thời gian.

5.2 Dự đoán doanh thu của từng chi nhánh

- Kết luận: Cả hai mô hình Prophet và XGBoosting đều không thực hiện cho ra kết quả tốt, với điểm RMSE hoặc MAE cao, hoặc điểm R2 ở mức âm.
- Hướng phát triển:
 - + Cần nhiều dữ liệu liên quan hơn đến việc phân tích dự đoán trên;
 - + Cần được huấn luyện nhiều mô hình hơn, như: Cây quyết định, rừng ngẫu nhiên, Gradient Boosting,...;
 - + Cần thuật toán hiệu quả và tối ưu hơn để xử lý các tập dữ liệu, vì chúng hao phí rất nhiều tài nguyên có hạn của máy.

TÀI LIỆU THAM KHẢO

1. Kaggle CRM Analytics (RFM-CLTV)
<https://www.kaggle.com/code/ihsnknkz/crm-analytics-rfm-cltv#Prepare-CLTV-DataFrame->
2. Scikit-learn Documentation. <https://scikit-learn.org>
3. Projecting Customer Lifetime Value using the BG/NBD and the Gamma-Gamma models. <https://medium.com/@yassirafif/projecting-customer-lifetime-value-using-the-bg-nbd-and-the-gamma-gamma-models-9a937c60fe7f>
4. Lifetimes Documentation <https://lifetimes.readthedocs.io/en/latest/index.html>
5. XGBoosting Documentation <https://xgboost-clone.readthedocs.io/en/latest/>
6. Prophet Documentation <https://prophet.readthedocs.io/en/latest/>