TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

LẬP TRÌNH PYTHON CHO MÁY HỌC (CS116)

BÁO CÁO ĐỒ ÁN CUỐI KỲ

Dự đoán doanh thu chuỗi cửa hàng Rossmann

Giảng viên hướng dẫn: TS. Nguyễn Vinh Tiệp
Sinh viên thực hiện: Phạm Nguyên Tường 23521751
Nguyễn Thanh Tùng 23521745
Tăng Hoàng Phúc 23521219
Chướng Hồng Văn 23521769

TP. Hồ Chí Minh, Tháng 6 năm 2025

Mục lục

1 Tổng quan

1.1 Giới thiệu đề tài

Dự báo doanh thu là một trong những bài toán then chốt trong lĩnh vực kinh doanh, đóng vai trò quan trọng trong việc hỗ trợ doanh nghiệp xây dựng kế hoạch tài chính, phân bổ nguồn lực hiệu quả, tối ưu chuỗi cung ứng và đưa ra các quyết định chiến lược dài hạn. Khả năng dự đoán chính xác doanh thu không chỉ giúp doanh nghiệp giảm thiểu rủi ro mà còn tạo lợi thế cạnh tranh trong môi trường kinh doanh đầy biến động.

Trong bối cảnh đó, bài toán **Rossmann Store Sales Forecasting** được xem là một ví dụ điển hình, phản ánh bài toán thực tế đến từ công ty dược phẩm Rossmann – một trong những chuỗi bán lẻ lớn nhất tại Đức với hàng ngàn cửa hàng trên toàn quốc (trong phạm vi bài toán là hơn 1.000 cửa hàng). Bài toán được công bố trong khuôn khổ một cuộc thi trên nền tảng Kaggle nhằm mục đích dự đoán doanh thu hằng ngày cho mỗi cửa hàng trong hệ thống, dựa trên dữ liệu lịch sử và thông tin liên quan đến hoạt động kinh doanh.

1.2 Lí do chọn đề tài

Trong bối cảnh nền kinh tế toàn cầu ngày càng chịu tác động mạnh mẽ từ dữ liệu và công nghệ, việc ứng dụng các phương pháp Machine Learning vào các bài toán dự báo trong lĩnh vực kinh doanh đã trở thành xu thế tất yếu. Trong số đó, bài toán dự báo doanh thu giữ vai trò đặc biệt quan trọng, do có ảnh hưởng trực tiếp đến hiệu quả hoạch định tài chính, điều phối chuỗi cung ứng, tối ưu hóa nguồn lực và nâng cao năng lực cạnh tranh của doanh nghiệp.

Bài toán Rossmann Store Sales Forecasting không chỉ mang tính thực tiễn cao khi bắt nguồn từ dữ liệu kinh doanh thực tế của một trong những chuỗi bán lẻ lớn tại châu Âu, mà còn thể hiện rõ nét các đặc trưng phức tạp thường gặp trong các bài toán dữ liệu chuỗi thời gian như tính mùa vụ, xu hướng, ảnh hưởng của các yếu tố ngoại sinh (như khuyến mãi, ngày lễ, ngày trong tuần), dữ liệu thiếu, nhiễu, và sự đa dạng về mặt loại hình cửa hàng.

Do đó, đề tài không chỉ đòi hỏi khả năng vận dụng kiến thức chuyên ngành (domain knowledge) trong lĩnh vực bán lẻ và chuỗi thời gian mà còn hướng đến mục tiêu rèn luyện và củng cố các kiến thức, kỹ năng cốt lõi trong việc triển khai một Machine Learning Pipeline, bao gồm: tiền xử lý dữ liệu, phân tích khám phá dữ liệu (EDA), xây dựng và đánh giá mô hình học máy, tối ưu siêu tham số và đánh giá mô hình.

2 Dataset

Bộ dữ liệu được sử dụng trong bài toán Rossmann Store Sales Forecasting gồm tổng cộng **1,017,209 dòng dữ liệu**, chia thành hai tệp chính: `train.csv` và `store.csv`.

- `train.csv`: chứa thông tin bán hàng theo ngày của từng cửa hàng.
  - **Store**: mã định danh của cửa hàng.
  - **DayOfWeek**: thứ trong tuần (1 = Thứ Hai, ..., 7 = Chủ Nhật).
  - **Date**: ngày bán hàng (định dạng YYYY-MM-DD).
  - **Sales**: doanh thu trong ngày (đơn vị tiền tệ không được công bố).
  - **Customers**: số lượng khách hàng đến cửa hàng.
  - **Open**: cửa hàng có mở cửa không (1 = mở, 0 = đóng).
  - **Promo**: có khuyến mãi đang chạy không (1 = có, 0 = không).
  - **StateHoliday**: ngày nghỉ lễ của bang (0 = không nghỉ, a = nghỉ lễ công cộng, b = nghỉ lễ tôn giáo, c = nghỉ lễ trường học).
  - **SchoolHoliday**: có nghỉ học không (1 = nghỉ học, 0 = không).
- `store.csv`: cung cấp thông tin bổ sung về từng cửa hàng.
  - **Store**: mã định danh của cửa hàng (dùng để nối với `train.csv`).
  - **StoreType**: loại cửa hàng (a, b, c, d).
  - **Assortment**: mức độ đa dạng của sản phẩm (a, b, c)
  - **CompetitionDistance**: khoảng cách tới đối thủ gần nhất
  - **CompetitionOpenSinceMonth/Year**: tháng và năm đối thủ bắt đầu hoạt động.
  - **Promo2**: cửa hàng có tham gia chương trình khuyến mãi kéo dài hạn không (1 = có, 0 = không).
  - **Promo2SinceWeek/Year**: tuần và năm bắt đầu áp dụng `Promo2`.
  - **PromoInterval**: các tháng trong năm có áp dụng `Promo2`.

Việc kết hợp hai bộ dữ liệu này cung cấp một cái nhìn toàn diện về hoạt động bán hàng cũng như các yếu tố ảnh hưởng đến doanh thu. Đây là tiền đề quan trọng cho việc tiền xử lý, trích xuất đặc trưng và xây dựng mô hình dự báo hiệu quả.

3 Exploratory Data Analysis

3.1 Kiểm tra dữ liệu bị thiếu hoặc trùng lặp

Đầu tiên, ta cần phải kiểm tra dữ liệu có bị thiếu hoặc trùng lặp không.

```
[ ]  train.isnull().sum()
```

CompetitionDistance	2642
CompetitionOpenSinceMonth	323348
CompetitionOpenSinceYear	323348
Promo2SinceWeek	508031
Promo2SinceYear	508031
PromoInterval	508031

Hình 1: Mô tả các giá trị bị thiếu trong bộ dữ liệu

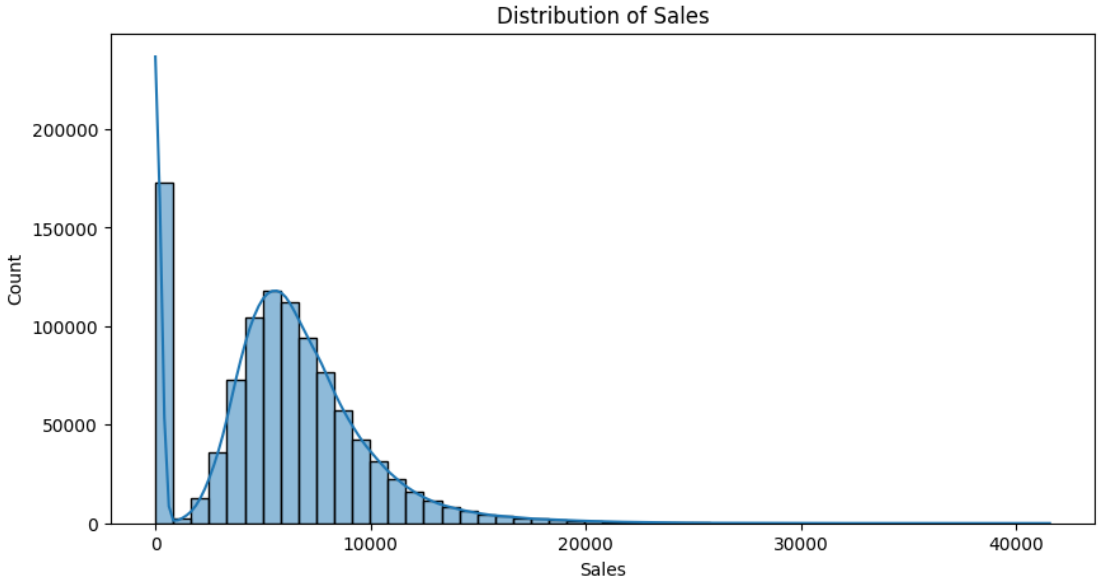
```
[ ]  print(train.duplicated().sum())
     print(test.duplicated().sum())

     0
     0
```

Hình 2: Mô tả các giá trị trùng lặp trong bộ dữ liệu

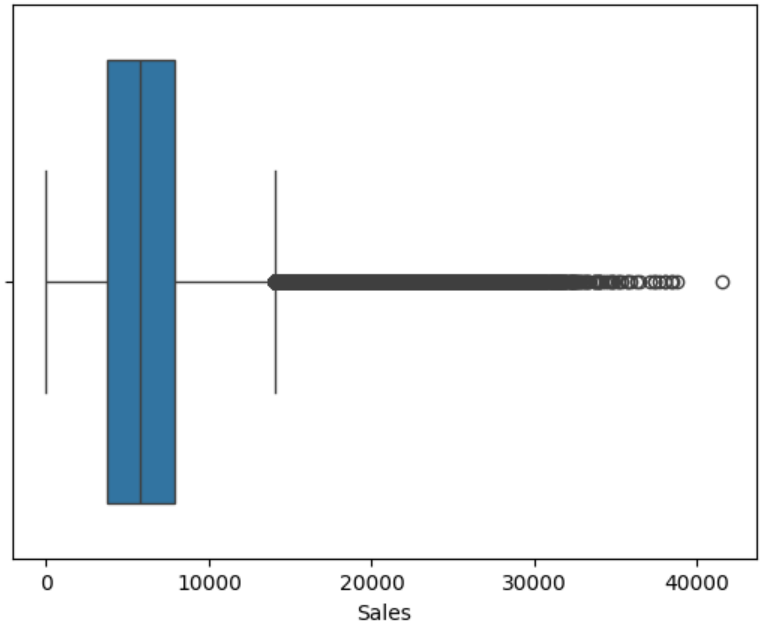
3.2 Phân tích đơn biến

Tiếp theo, ta sẽ tiến hành phân tích biến mục tiêu của bộ dữ liệu này để xem nó có phân phối ra sao.



Hình 3: Phân phối của biến mục tiêu

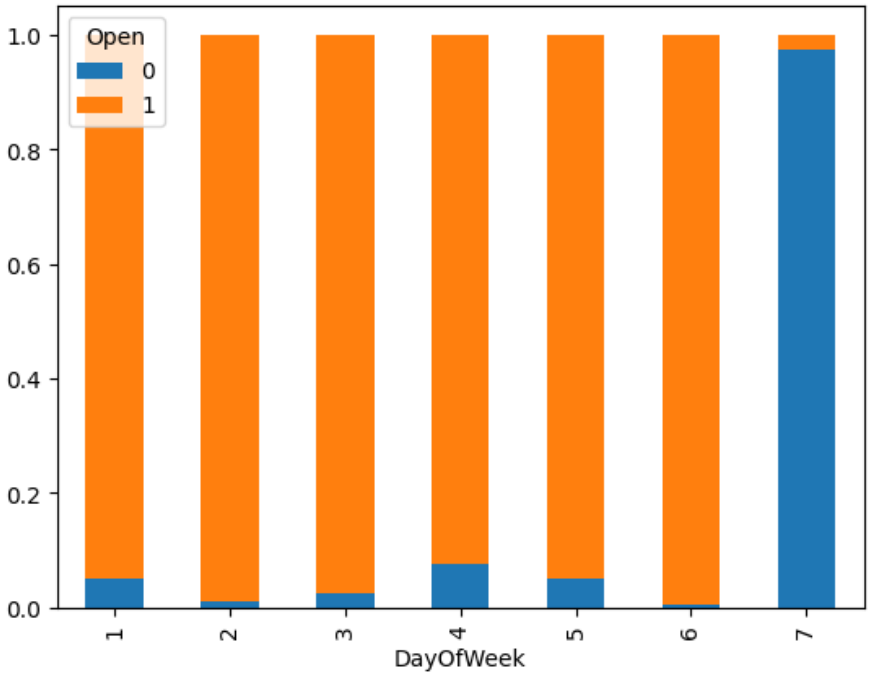
Hình biểu đồ phân phối doanh thu (Sales) cho thấy dữ liệu có phân phối lệch phải (right-skewed), với phần lớn doanh số tập trung dưới 10,000, nhưng có một số giá trị rất lớn, lên đến hơn 40,000. Điều này cho thấy sự chênh lệch lớn trong doanh số, với một số ít giao dịch có giá trị cực cao. Sau đó, thông qua boxplot, ta sẽ xác định outliers của biến mục tiêu.



Hình 4: Outliers của biến mục tiêu

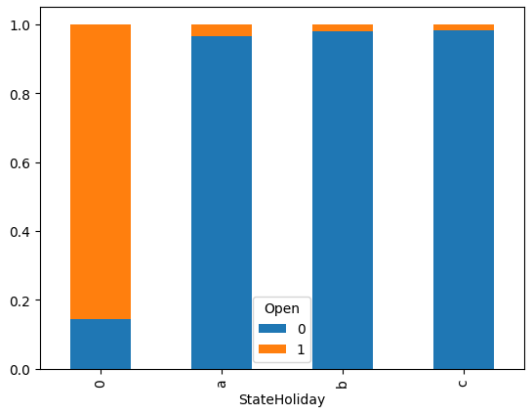
Biểu đồ boxplot cho biến Sales cho thấy phân phối lệch phải rõ rệt với nhiều giá trị ngoại lai ở phía cao. Phần lớn dữ liệu tập trung dưới ngưỡng 15,000 trong khi có những điểm đạt tới hơn 40,000. Điều này gợi ý rằng trung bình của Sales có thể bị ảnh hưởng mạnh bởi một số ít điểm cao bất thường, và trung vị có thể là thước đo phù hợp hơn trong phân tích trung tâm.

3.3 Phân tích 2 biến

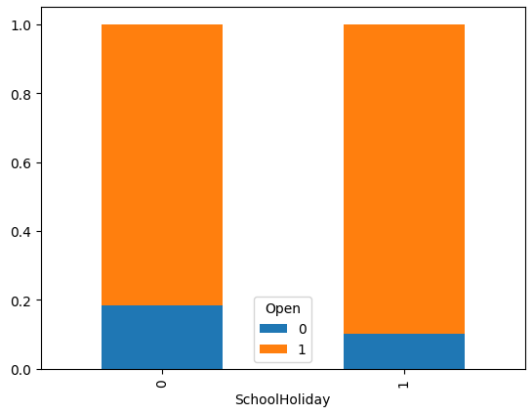


Hình 5: Tần suất mở cửa của các cửa hàng theo ngày

Biểu đồ này cho ta thấy phần lớn cửa hàng đều mở cửa vào những ngày trong tuần và đóng cửa vào chủ nhật.

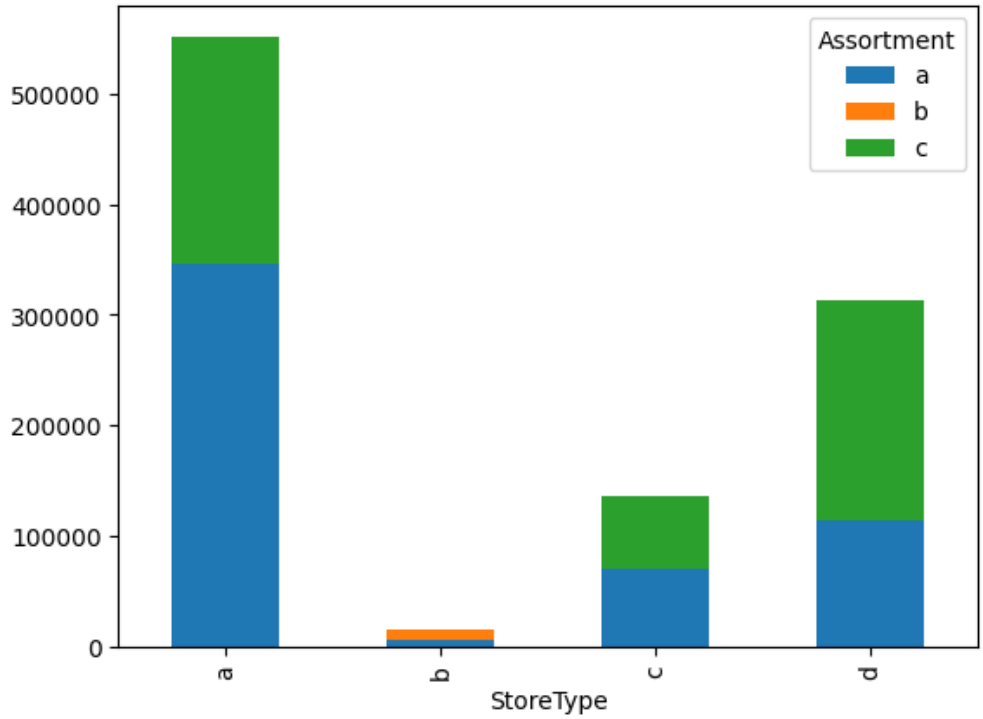


Hình 6: Tần suất mở cửa của các cửa hàng vào các ngày lễ



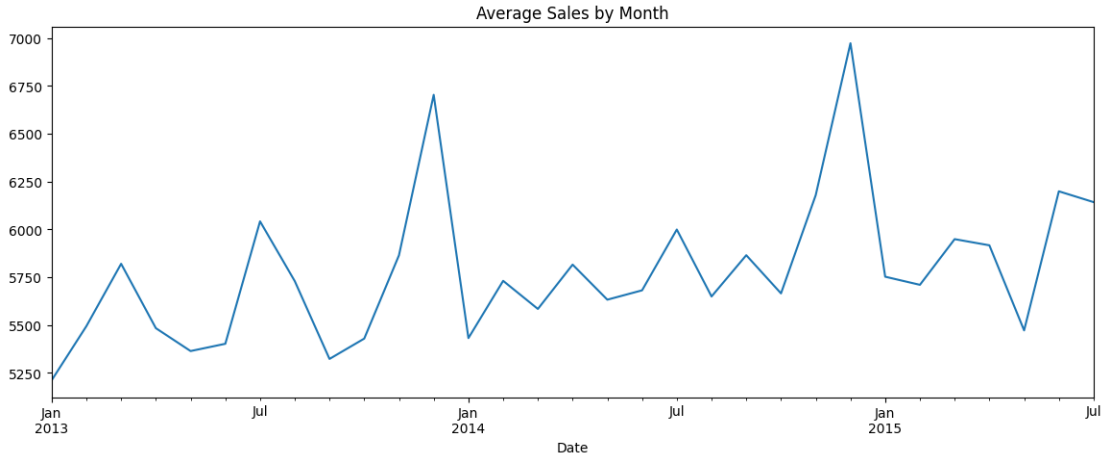
Hình 7: Tần suất mở cửa của các cửa hàng vào các ngày nghỉ ở trường

Qua 2 biểu đồ trên ta dễ thấy rằng các cửa hàng ở đa số đều đóng của và Chủ nhật và các ngày lễ quan trọng như Giáng sinh, Lễ Phục sinh,... còn ngày nghỉ của trường học thì đa số vẫn mở cửa. Điều này xảy ra là vì ở Đức có quy định về việc đóng cửa toàn bộ cửa hàng vào ngày Chủ nhật,ngày lễ lớn tuy nhiên hiện nay cũng có phần nới lỏng hơn.



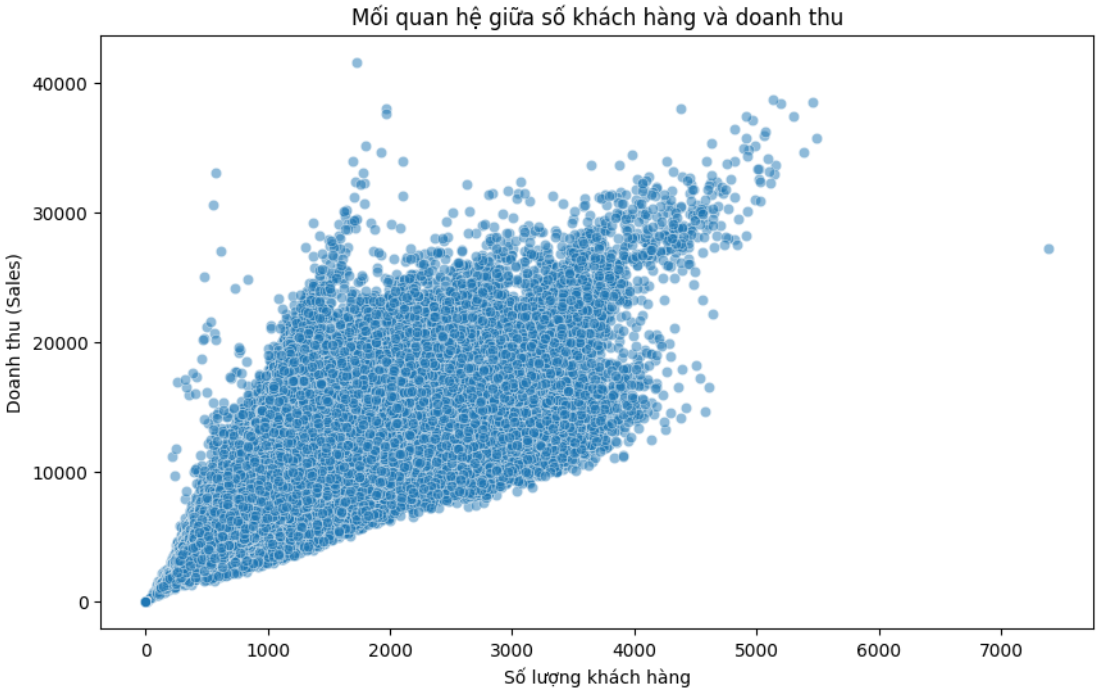
Hình 8: Mối quan hệ giữa StoreType và Assortment

Biểu đồ cho thấy mối quan hệ giữa loại cửa hàng (StoreType) và loại hàng hóa (Assortment) không đồng đều. Cửa hàng loại a là phổ biến nhất, sử dụng nhiều cả Assortment a và c. Trong khi đó, StoreType b rất hiếm và chủ yếu dùng Assortment b, có thể xem như một nhóm ngoại lệ. Việc các nhóm StoreType dùng các Assortment khác nhau cho thấy 2 biến này có mối quan hệ nhất định, và đều có thể mang thông tin hữu ích cho mô hình dự đoán doanh thu.



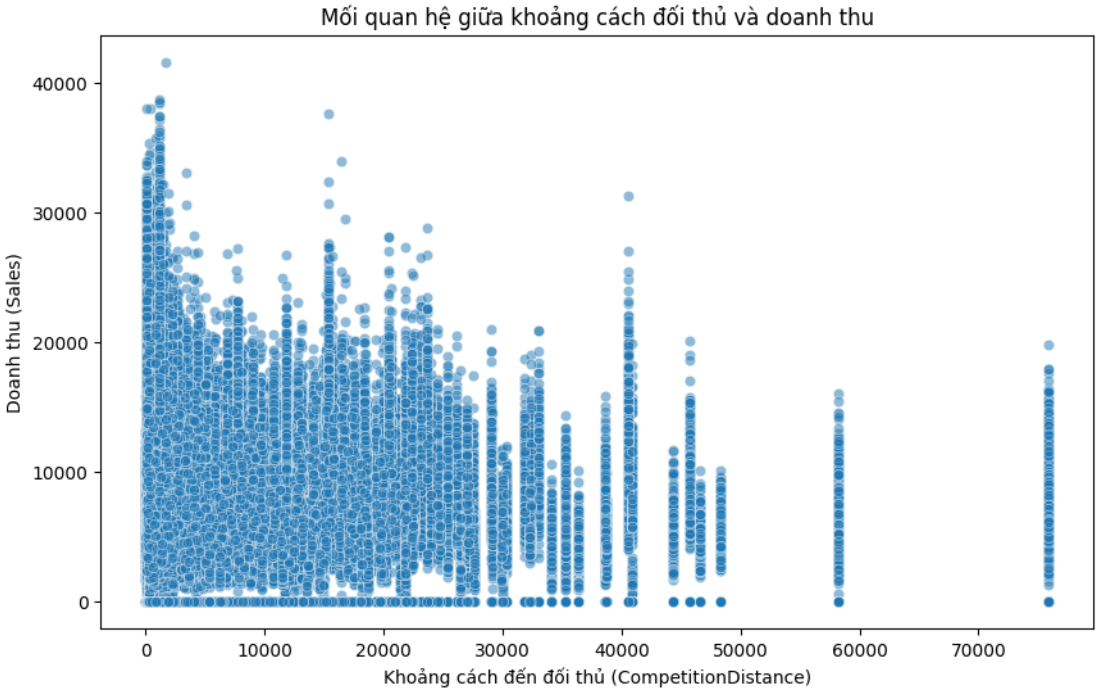
Hình 9: Doanh thu trung bình tháng qua từng năm

Để có cái nhìn rõ hơn thì ta xem xét biểu đồ trực quan hóa chi tiết doanh thu hàng tháng qua từng năm. Ta có thể thấy rằng doanh thu đặc biệt tăng mạnh vào các tháng 3, kỳ nghỉ hè (tháng 6,7) và dịp cuối năm (tháng 12). Từ đó, có thể thấy rằng doanh nghiệp có chu kỳ bán hàng theo mùa rõ rệt — điều này rất quan trọng trong việc dự báo doanh thu và hoạch định chiến lược marketing.



Hình 10: Mối quan hệ giữa số lượng khách hàng và doanh thu

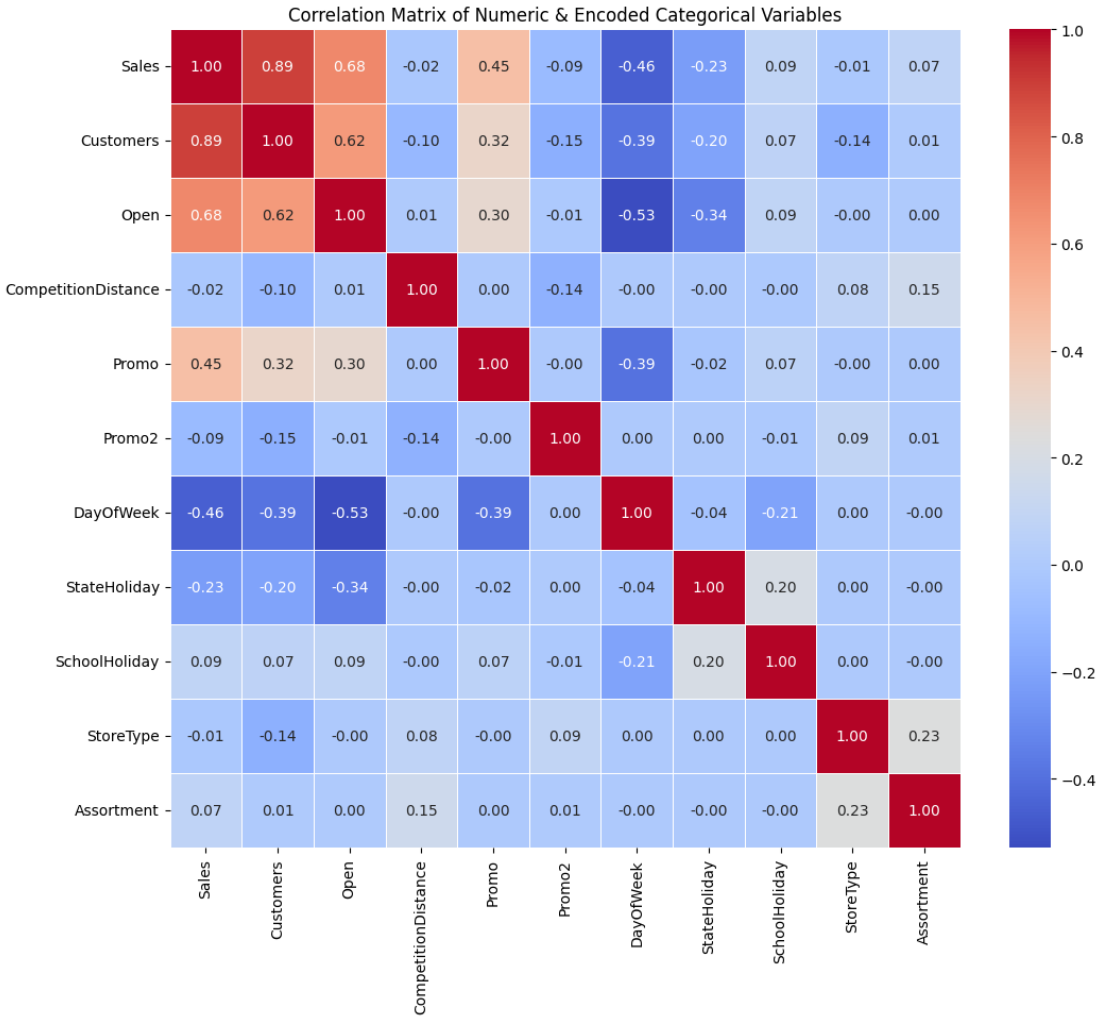
Đối với scatterplot này thì ta thấy số lượng khách hàng có mối quan hệ tuyến tính với doanh thu, nên ta rút ra được kết luận rằng số lượng khách hàng có sức ảnh hưởng lớn đến doanh thu của các cửa hàng.



Hình 11: Mối quan hệ giữa khoảng cách đối thủ và doanh thu

Như ta có thể thấy ở scatterplot trên, do là dữ liệu phân bố rải rác, do đó CompetitionDistance không có mối quan hệ rõ ràng với Sales, hay nói cách khác, khoảng cách với các cửa hàng đối thủ không ảnh hưởng đáng kể đến doanh thu (Ví dụ: Một số siêu thị lớn có thương hiệu mạnh có thể thu hút khách hàng ngay cả khi gần hoặc xa đối thủ).

3.4 Phân tích đa biến



Hình 12: Ma trận tương quan

Một số **insight** ta có thể rút ra được từ ma trận tương quan:

- Khi cửa hàng có **khuyến mãi (Promo)**, khách hàng sẽ quan tâm nhiều hơn → lượng khách (*Customers*) tăng, từ đó kéo theo *doanh thu (Sales)* tăng.
- Nếu cửa hàng **mở cửa (Open = 1)**, hiển nhiên mới có khách đến và phát sinh doanh thu → giải thích mối quan hệ chặt giữa *Open*, *Customers*, và *Sales*.
- **DayOfWeek** ảnh hưởng đến hành vi mua sắm: có những ngày trong tuần mà khách ít đến hơn, dẫn đến doanh thu giảm.

- Các dịp nghỉ lễ như **StateHoliday** hay **SchoolHoliday** không tác động rõ rệt đến doanh thu, có thể do khách vẫn mua sắm bình thường hoặc tùy thuộc bối cảnh cụ thể.
- Do đó, những biến có tương quan cao với *Sales*, đặc biệt là *Customers*, *Open* và *Promo*, nên được **ưu tiên trong quá trình xây dựng mô hình**.

4 Data Preprocessing

4.1 Xử lý dữ liệu bị thiếu

```
[ ]  train.isnull().sum()
```

CompetitionDistance	2642
CompetitionOpenSinceMonth	323348
CompetitionOpenSinceYear	323348
Promo2SinceWeek	508031
Promo2SinceYear	508031
PromoInterval	508031

Hình 13: Mô tả các giá trị bị thiếu trong bộ dữ liệu

Đối với ba cột **Promo2SinceMonth**, **Promo2SinceYear** và **PromoInterval**, các giá trị thiếu luôn xuất hiện đồng thời và chỉ xảy ra khi giá trị của cột Promo2 bằng 0, tức là cửa hàng không tham gia chương trình khuyến mãi định kỳ (Promo2). Do đó, có thể suy luận rằng việc thiếu dữ liệu trong ba cột này thực chất là có chủ đích và mang ý nghĩa thông tin.

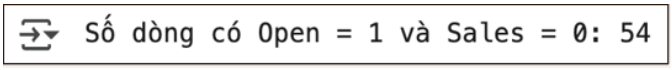
Để bảo toàn ngữ nghĩa này, ta tiến hành điền các giá trị thiếu bằng một giá trị 0 đối với các cột số bị thiếu. Việc điền giá trị đặc biệt giúp mô hình học máy phân biệt rõ ràng giữa "không có chương trình khuyến mãi" và "thiếu dữ liệu".

Đối với cột **CompetitionDistance**, giá trị thiếu (null) thường hàm ý rằng không có cửa hàng đối thủ trong khu vực hoặc khoảng cách đến cửa hàng đối thủ không xác định. Do đó, các giá trị thiếu trong cột này được thay thế bằng giá trị đặc biệt là 0, nhằm thể hiện rõ ràng rằng không tồn tại hoặc không xác định được sự hiện diện của đối thủ cạnh tranh.

Đồng thời, khi **CompetitionDistance** bị thiếu, thì các cột liên quan đến thời điểm đối thủ bắt đầu hoạt động, cụ thể là **CompetitionOpenSinceYear** và **CompetitionOpenSinceMonth** - cũng mất đi ý nghĩa và cần được điền bằng giá trị đặc biệt là 0, để đảm bảo tính nhất quán trong dữ liệu.

Ngược lại, trong trường hợp **CompetitionDistance** có giá trị (tức là tồn tại thông tin về khoảng cách đối thủ), nhưng **CompetitionOpenSinceYear** hoặc **CompetitionOpenSinceMonth** bị thiếu, ta giả định rằng cửa hàng đối thủ có tồn tại nhưng không rõ thời điểm bắt đầu hoạt động. Do đó, các giá trị thiếu này sẽ được thay thế bằng 0, mang hàm ý “không xác định thời điểm bắt đầu cạnh tranh".

4.2 Xử lý outlier



Hình 14: Thông kế số dòng không hợp lệ

Trong quá trình kiểm tra dữ liệu, phát hiện 54 dòng có giá trị Open = 1 nhưng Sales = 0. Điều này là bất thường vì cửa hàng mở cửa nhưng không có doanh thu, có thể do lỗi nhập liệu hoặc ngoại lệ không rõ nguyên nhân. Do các dòng này không mang lại thông tin hữu ích cho quá trình huấn luyện mô hình và có thể gây nhiễu, chúng được loại bỏ khỏi tập dữ liệu huấn luyện.

Như đã thấy trong phần EDA, doanh thu của chuỗi cửa hàng có phân phối lệch phải và chứa rất nhiều các giá trị doanh thu tăng cao đột biến. Do đó để xác định đâu là outlier để loại bỏ thì nhóm đã dùng một phương pháp là **MAD-based Outlier Detection**.

Thông thường để xác định outlier, người ta thường dùng Z-score, tuy nhiên Z-score chỉ áp dụng được cho dữ liệu có phân phối chuẩn hoặc gần chuẩn. Do đó, một phương khác được đề xuất để có thể hoạt động trên cả dữ liệu không có phân phối chuẩn là **Modified Z-score**.

Modified Z-score dựa trên **Median Absolute Deviation (MAD)** – một thước đo bền vững hơn so với độ lệch chuẩn (*standard deviation*), đặc biệt hiệu quả khi dữ liệu chứa nhiều ngoại lệ (*outliers*).

Giá trị MAD được tính theo công thức:

$$\text{MAD} = \text{median}\left(|x_i - \text{median}(x)|\right)$$

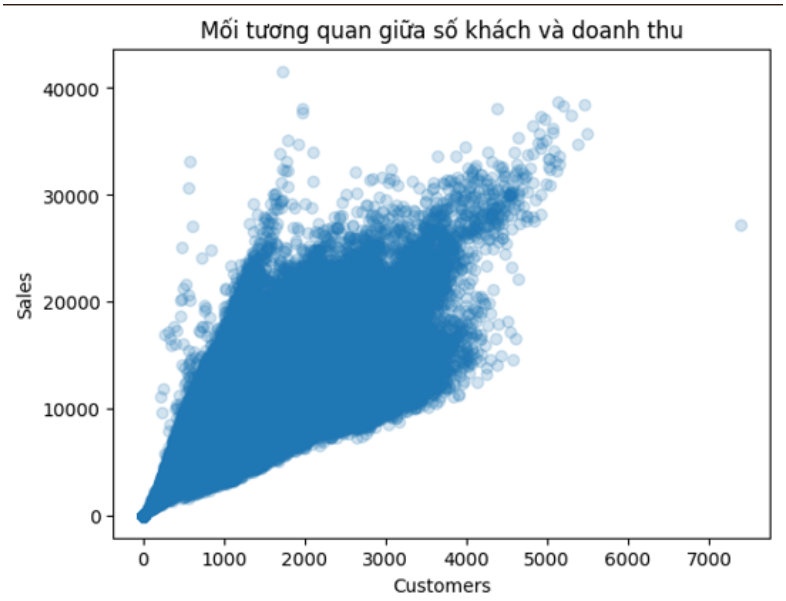
Sau đó, Z-score được hiệu chỉnh theo công thức:

$$\text{Modified Z-score} = \frac{0.6745 \times |x_i - \text{median}(x)|}{\text{MAD}}$$

Nếu giá trị *Modified Z-score* vượt quá một ngưỡng định trước (ở đây nhóm dùng giá trị 3.5), điểm dữ liệu đó được xem là **ngoại lệ (outlier)**.

4.3 Feature Engineering

4.3.1 Customers



Hình 15: Mối tương quan giữa số khách hàng và doanh thu

Biến **Customers** thể hiện số lượng khách mỗi ngày và có mối tương quan mạnh với **Sales**. Tuy nhiên, vì thông tin này không có sẵn trong tập dữ liệu tương lai (tập kiểm tra), nên không thể sử dụng trực tiếp trong huấn luyện để tránh rò rỉ dữ liệu. Do đó, trước khi loại bỏ biến này, ta tận dụng thông tin từ nó để tạo ra các đặc trưng tổng hợp mới có giá trị dự báo cao.

Ta tạo ra các đặc trưng tổng hợp mới theo từng cửa hàng:

- `Store_sales_per_day`: doanh thu trung bình của cửa hàng.
- `Store_customer_per_day`: số lượng khách hàng trung bình của cửa hàng.
- `Store_sales_per_customer_per_day`: doanh thu trung bình trên mỗi khách hàng của cửa hàng.

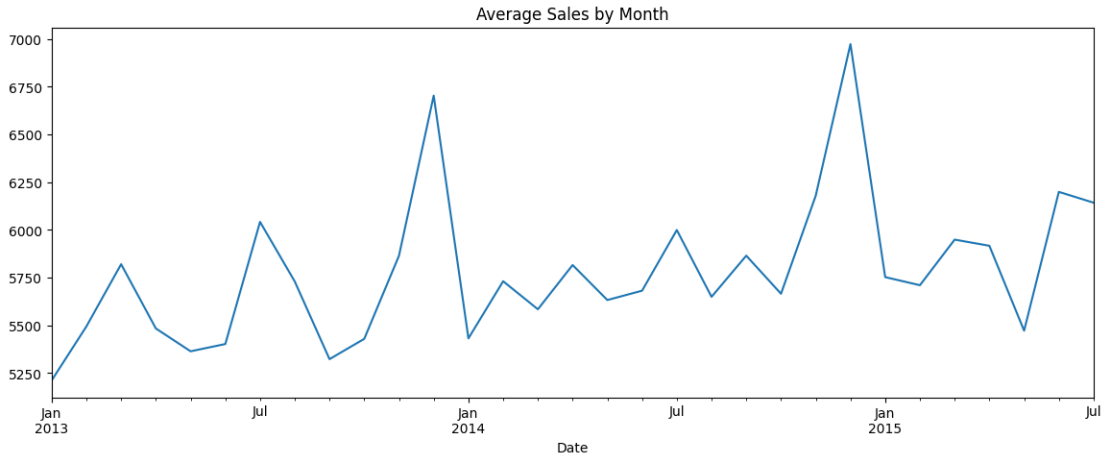
4.3.2 Promo2

Để xử lý dữ liệu liên quan đến chương trình khuyến mãi **Promo2**, tiến hành gộp hai cột có ý nghĩa tương đồng nhau là **Promo2SinceYear** và **Promo2SinceWeek** để tạo thành cột mới có tên **Promo2StartDate**. Từ cột này, tính toán số ngày kể từ khi **Promo2** bắt đầu đến thời điểm hiện tại và lưu vào cột **TimeSincePromo2Start**.

4.3.3 Competition

Tương tự như với **Promo2**, dữ liệu về đối thủ cạnh tranh (**Competition**) cũng được xử lý bằng cách tính toán số ngày kể từ khi đối thủ bắt đầu hoạt động và lưu vào cột **CompetitionAge**.

4.3.4 Xu hướng biến động doanh thu trong năm



Hình 16: Biểu đồ doanh thu trung bình theo tháng

Từ biểu đồ *Average Sales by Month*, có thể nhận thấy doanh thu thường tăng mạnh vào các khoảng thời gian nhất định trong năm, cụ thể là vào tháng 3, kỳ nghỉ hè (tháng 6 và tháng 7), cũng như giai đoạn trước Giáng sinh và năm mới (tháng 11 và tháng 12). Dựa vào đặc điểm này, ta có thể xây dựng thêm các biến đếm để đánh dấu các mốc thời gian quan trọng này nhằm phục vụ cho việc dự báo doanh thu:

- `Days_to_Mar1`: số ngày còn lại đến ngày 1 tháng 3. Mốc này phản ánh giai đoạn doanh thu có xu hướng tăng vào đầu mùa xuân.
- `Days_to_July1`: số ngày còn lại đến ngày 1 tháng 7. Đây là thời điểm kỳ nghỉ hè bắt đầu, thường đi kèm với sự gia tăng doanh thu.
- `Days_to_Dec1`: số ngày còn lại đến ngày 1 tháng 12, trước giai đoạn Giáng sinh và năm mới, khi nhu cầu mua sắm tăng mạnh.

4.4 Mã hóa đặc trưng

4.4.1 Các biến phân loại

Để xử lý các biến phân loại (categorical variables) trong dữ liệu, ta sử dụng phương pháp *Label Encoding*, nhằm chuyển các giá trị phân loại thành dạng số nguyên. Cụ thể, các cột được mã hóa gồm: **StateHoliday, Assortment, PromoInterval**.

```
StateHoliday [0 1 2 3] [0 1]
Assortment [0 2 1] [0 2 1]
PromoInterval [3 1 0 2] [3 1 0 2]
```

Hình 17: Các biến phân loại sau khi được mã hóa

Việc mã hóa này giúp các mô hình học máy có thể tiếp nhận và xử lý các biến phân loại một cách hiệu quả, đồng thời đảm bảo tính nhất quán giữa tập huấn luyện và tập kiểm tra.

4.4.2 Mã hóa đặc trưng thời gian

Các đặc trưng thời gian như `Month`, `DayOfWeek` và `Week` đều mang tính chất tuần hoàn, tức là các giá trị này lặp lại theo chu kỳ (ví dụ: sau tháng 12 sẽ quay lại tháng 1, sau Chủ nhật sẽ quay lại thứ Hai). Nếu giữ nguyên giá trị số thông thường, mô hình có thể hiểu nhầm mối quan hệ tuyến tính giữa các giá trị này, trong khi thực tế mối quan hệ là tuần hoàn.

Để giải quyết vấn đề này, các đặc trưng tuần hoàn được chuyển đổi bằng cách sử dụng hai hàm số *sin* và *cos*, giúp mô hình học được tính chu kỳ của dữ liệu. Công thức chuyển đổi như sau:

$$x_{\sin} = \sin\left(2\pi \cdot \frac{x}{m}\right), \quad x_{\cos} = \cos\left(2\pi \cdot \frac{x}{m}\right) \tag{1}$$

Trong đó:

- x là giá trị của đặc trưng thời gian cần mã hóa.
- m là số giá trị có thể có của đặc trưng đó (ví dụ: 12 đối với tháng, 7 đối với thứ trong tuần).

Nếu chỉ sử dụng một trong hai hàm *sin* hoặc *cos*, có thể xảy ra trường hợp các giá trị khác nhau bị ánh xạ về cùng một giá trị, dẫn đến mất thông tin. Ví dụ, ta có:

$$\sin(0^\circ) = \sin(180^\circ) = 0$$

Trong khi đó, hai góc 0° và 180° lại nằm ở hai vị trí hoàn toàn khác nhau trên vòng tròn lượng giác. Vì vậy, việc kết hợp đồng thời cả hai thành phần *sin* và *cos* giúp biểu diễn vị trí chính xác của mỗi giá trị trong chu kỳ, đảm bảo mô hình học được đầy đủ tính chất tuần hoàn của dữ liệu.

4.5 Biến đổi đặc trưng

4.5.1 Biến mục tiêu Sales

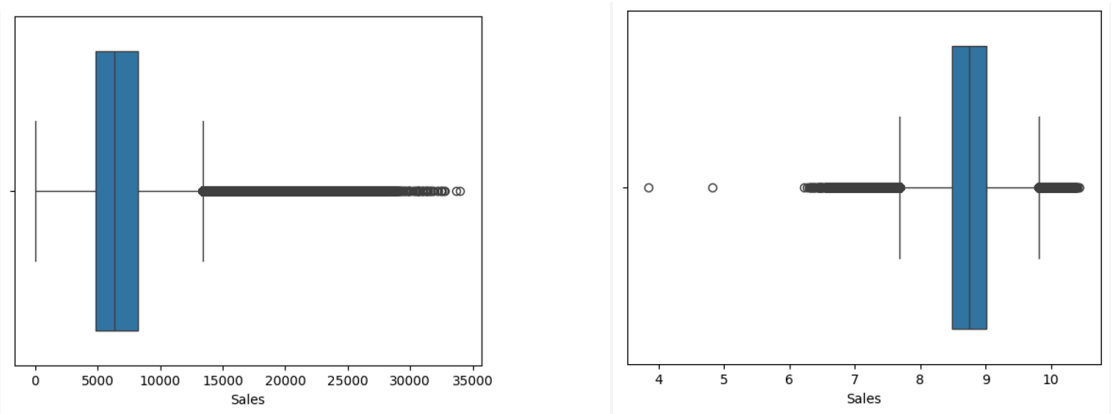
Mặc dù đã loại bỏ khá nhiều outlier ở bước tiền xử lý, tuy nhiên theo như biểu đồ *Hình 18a* thì phân phối doanh thu vẫn bị lệch phải và tồn tại nhiều giá trị tăng cao đột biến. Do đó, để

giảm ảnh hưởng của các giá trị bất thường mà không làm mất thông tin quý giá, phương pháp Log Transform được áp dụng. Cụ thể, giá trị doanh thu được chuyển đổi theo công thức:

$$y' = \log(y + 1) \tag{2}$$

Trong đó, y là giá trị doanh thu ban đầu, và y' là giá trị sau khi chuyển đổi. Việc cộng thêm 1 nhằm xử lý trường hợp giá trị bằng 0.

Kết quả sau khi áp dụng log transform được thể hiện tại *Hình 18b*, cho thấy phân phối dữ liệu trở nên gần chuẩn hơn và giảm ảnh hưởng của các giá trị tăng cao bất thường.



(a) Trước khi áp dụng Log Transform

(b) Sau khi áp dụng Log Transform

Hình 18: So sánh phân phối dữ liệu trước và sau khi áp dụng Log Transform

4.5.2 Các biến khác

Qua quan sát các biểu đồ hộp của các đặc trưng **CompetitionDistance**, **TimeSincePromo2Start** và **CompetitionAge**, có thể nhận thấy các đặc trưng này tồn tại nhiều giá trị ngoại lai (outlier) với độ lệch lớn so với phần còn lại của dữ liệu.

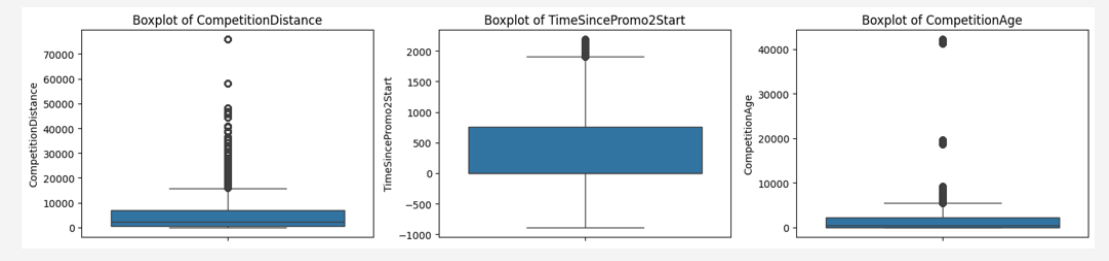
Để giảm ảnh hưởng của các giá trị ngoại lai mà vẫn giữ nguyên được phân bố dữ liệu chính, phương pháp *RobustScaler* được áp dụng. *RobustScaler* sử dụng trung vị và khoảng tứ phân vị (IQR) để chuẩn hóa dữ liệu, giúp giảm thiểu ảnh hưởng của các điểm bất thường. Công thức chuyển đổi như sau:

$$x' = \frac{x - \text{median}(X)}{\text{IQR}(X)} \tag{3}$$

Trong đó:

- x là giá trị gốc của đặc trưng.
- $\text{median}(X)$ là trung vị của đặc trưng.

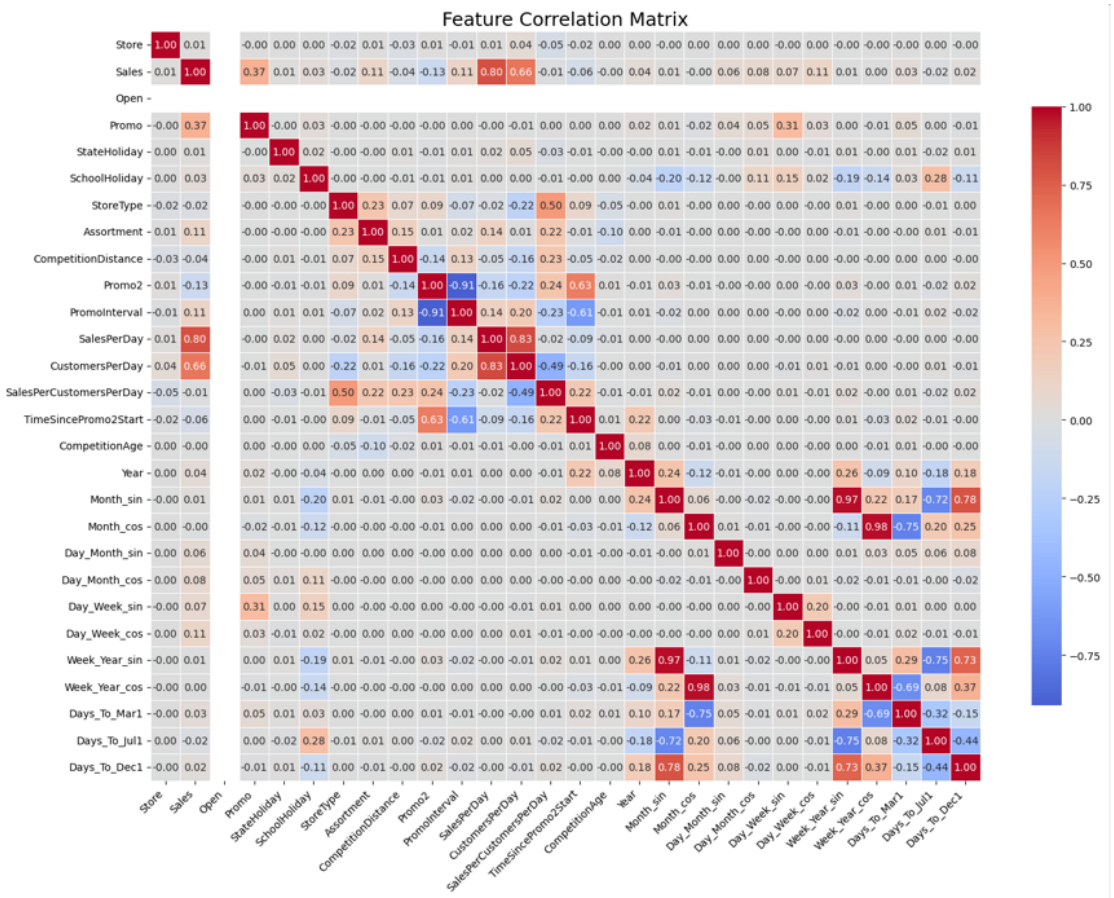
- IQR(X) là khoảng tứ phân vị, được tính bằng hiệu số giữa phân vị thứ ba (Q_3) và phân vị thứ nhất (Q_1).



Hình 19: Boxplot của các biến có phân phối lệch

Việc sử dụng *RobustScaler* giúp đưa dữ liệu về cùng một thang đo, đồng thời hạn chế tối đa ảnh hưởng tiêu cực từ các outlier trong quá trình huấn luyện mô hình.

4.6 Xử lý đa cộng tuyến



Hình 20: Ma trận tương quan

Qua phân tích ma trận tương quan giữa các đặc trưng, nhận thấy các cặp biến tuần hoàn `Week_year_sin/cos` và `Month_sin/cos` có hệ số tương quan rất cao, lên đến khoảng 0.97. Mối tương quan chặt chẽ này có thể gây ra hiện tượng đa cộng tuyến (multicollinearity), làm ảnh hưởng đến tính ổn định và khả năng giải thích của mô hình.

Để khắc phục, ta tiến hành loại bỏ một trong hai cặp đặc trưng này nhằm giảm bớt độ phức tạp của dữ liệu và tránh hiện tượng đa cộng tuyến.

5 Lựa chọn và Huấn luyện Mô hình

5.1 Các mô hình được thử nghiệm

Để xác định mô hình phù hợp nhất cho bài toán hồi quy chuỗi thời gian, nhóm đã triển khai và đánh giá nhiều mô hình khác nhau, bao gồm cả các phương pháp tuyến tính truyền thống và các mô hình boosting hiện đại. Cụ thể:

- **Linear Regression**: Mô hình hồi quy tuyến tính cơ bản, được sử dụng làm baseline. Ưu điểm là đơn giản, dễ huấn luyện và có khả năng giải thích rõ ràng.
- **Ridge Regression**: Biến thể của hồi quy tuyến tính có sử dụng regularization L2, giúp giảm overfitting bằng cách kiểm soát độ lớn của các hệ số.
- **Lasso Regression**: Sử dụng regularization L1 để loại bỏ các đặc trưng không quan trọng (feature selection), đồng thời hỗ trợ khả năng tổng quát hóa tốt hơn.
- **XGBoost (Extreme Gradient Boosting)**: Mô hình boosting mạnh mẽ, có khả năng mô hình hóa các quan hệ phi tuyến và được tối ưu hóa cho cả CPU và GPU. Phù hợp với dữ liệu có cấu trúc phức tạp.
- **CatBoost**: Mô hình boosting dựa trên gradient, hỗ trợ trực tiếp các bài toán hồi quy. Có khả năng xử lý tốt các đặc trưng dạng rời (categorical features) mà không cần one-hot encoding, đồng thời tránh overfitting nhờ các kỹ thuật như ordered boosting.
- **LightGBM (Light Gradient Boosting Machine)**: Mô hình boosting hiệu quả về tốc độ và bộ nhớ, sử dụng chiến lược phân tách theo lá (leaf-wise) để đạt độ chính xác cao. Rất phù hợp cho các bài toán hồi quy trên dữ liệu lớn.
- **Ensemble (Mô hình tổ hợp)**: Kết hợp đầu ra từ các mô hình mạnh bằng các kỹ thuật như trung bình hóa (averaging), voting hoặc stacking. Giúp cải thiện độ chính xác, tăng độ ổn định và giảm sai số tổng thể.

5.2 Chiến lược chia dữ liệu

Vì bài toán là chuỗi thời gian, nên việc chia dữ liệu cần đảm bảo nguyên tắc không để thông tin từ tương lai rò rỉ vào mô hình. Cụ thể:

- **Tập huấn luyện**: Chiếm 75% đầu tiên theo thời gian.
- **Tập kiểm tra (validation)**: 25% cuối cùng, đại diện cho tương lai gần.

Việc chia không ngẫu nhiên (`shuffle=False`) nhằm đảm bảo mô hình học đúng bản chất của chuỗi thời gian.

5.3 Chiến lược tối ưu siêu tham số

Đối với từng nhóm mô hình, nhóm áp dụng chiến lược tối ưu khác nhau nhằm cải thiện độ chính xác:

5.3.1 Grid Search

Grid Search được áp dụng cho các mô hình tuyến tính do có số lượng siêu tham số ít, không gian tìm kiếm nhỏ và thời gian huấn luyện nhanh.

- `Linear Regression`: Mô hình hồi quy tuyến tính cơ bản, không có siêu tham số cần tinh chỉnh. Được sử dụng làm baseline để so sánh với các mô hình khác.
- `Ridge Regression`: Áp dụng regularization L2 để giảm hiện tượng overfitting bằng cách làm nhỏ các hệ số. Siêu tham số được tinh chỉnh là:
  - `alpha`: Hệ số điều chỉnh mức phạt L2, kiểm soát độ lớn của hệ số hồi quy. Các giá trị được thử: $\{0.001, 0.01, 0.1, 1, 10\}$. Giá trị lớn làm mô hình đơn giản hơn nhưng có thể gây underfitting.
- `Lasso Regression`: Áp dụng regularization L1 giúp loại bỏ hoàn toàn các đặc trưng không quan trọng bằng cách đẩy hệ số về 0 (feature selection). Các siêu tham số được tinh chỉnh là:
  - `alpha`: Hệ số điều chỉnh regularization L1, với các giá trị thử nghiệm $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$. Giá trị lớn hơn sẽ mạnh tay hơn trong việc loại bỏ đặc trưng.
  - `max_iter`: Số vòng lặp tối đa trong quá trình tối ưu hóa. Được thử với các giá trị $\{1000, 5000, 10000\}$ để đảm bảo mô hình hội tụ khi gradient nhỏ.

5.3.2 Optuna

Optuna là thư viện tối ưu hóa siêu tham số hiện đại sử dụng chiến lược Bayes (Bayesian Optimization). Thay vì duyệt toàn bộ không gian như Grid Search, Optuna tập trung vào các vùng có tiềm năng tốt. Được sử dụng cho các mô hình boosting với không gian siêu tham số lớn và chi phí huấn luyện cao.

- `XGBoost` Mô hình boosting dựa trên cây, nổi bật về hiệu suất và khả năng xử lý dữ liệu mất cân bằng.
  - `n_estimators`: Số lượng cây được huấn luyện, ảnh hưởng đến độ chính xác và thời gian huấn luyện. Thử trong khoảng $[100, 1000]$.
  - `max_depth`: Độ sâu tối đa của cây, điều khiển độ phức tạp của mô hình. Giá trị thử $[3, 20]$.
  - `learning_rate`: Tốc độ học, kiểm soát mức thay đổi của mô hình trong mỗi lần cập nhật. Giá trị nhỏ (log-scale) từ $[0.01, 0.3]$ giúp học ổn định hơn.
  - `subsample`: Tỷ lệ mẫu dữ liệu được dùng cho mỗi cây, thử trong khoảng $[0.5, 1.0]$, giúp giảm overfitting.
  - `colsample_bytree`: Tỷ lệ đặc trưng được chọn cho mỗi cây, cũng thử trong $[0.5, 1.0]$.
  - `gamma`: Độ lợi tối thiểu để chia một node, dùng để làm cây đơn giản hơn. Giá trị thử $[0, 5]$.
  - `reg_alpha`, `reg_lambda`: Hệ số regularization L1 và L2, giúp kiểm soát overfitting. Cả hai được thử trong $[0, 10]$.

  - `min_child_weight`: Số lượng mẫu tối thiểu cần thiết để chia node, thử trong khoảng $[1, 20]$.
  - `max_delta_step`: Bước nhảy tối đa trong tối ưu hóa logistic, hữu ích cho dữ liệu mất cân bằng. Giá trị thử $[0, 10]$.

- `CatBoost` Mô hình boosting tối ưu cho dữ liệu phân loại, xử lý tốt dữ liệu không cần one-hot encoding.
  - `iterations`: Số vòng boosting, ảnh hưởng trực tiếp đến độ chính xác và thời gian huấn luyện. Giá trị thử $[300, 1000]$.
  - `depth`: Độ sâu của cây quyết định, điều khiển mức độ phức tạp của mô hình. Giá trị từ $[4, 10]$.
  - `learning_rate`: Tốc độ học, thử từ $[0.01, 0.3]$ theo log-scale để tránh học quá nhanh.
  - `l2_leaf_reg`: Hệ số regularization L2 cho mỗi lá, giúp chống overfitting. Thử nghiệm từ $[1, 10]$.
  - `random_strength`: Mức độ ngẫu nhiên trong chia nhánh, giúp tăng đa dạng hóa cây. Thử từ $[1e^{-9}, 10.0]$ (log scale).
  - `bootstrap_type`: Phương pháp lấy mẫu: `Bayesian`, `Bernoulli` hoặc `MVS`, giúp điều khiển chiến lược huấn luyện cây.

- `LightGBM` Mô hình boosting hiệu quả cao, có tốc độ huấn luyện nhanh và phù hợp với dữ liệu lớn.
  - `n_estimators`: Số lượng cây boosting, thử từ $[100, 1000]$.
  - `max_depth`: Độ sâu tối đa của cây, giới hạn độ phức tạp. Thử trong khoảng $[3, 15]$.
  - `learning_rate`: Tốc độ học, thử từ $[0.01, 0.3]$.
  - `num_leaves`: Số lượng lá mỗi cây, giá trị lớn hơn cho phép học phức tạp hơn. Thử $[20, 200]$.
  - `min_data_in_leaf`: Số lượng mẫu tối thiểu trong mỗi lá. Thử $[10, 100]$ để kiểm soát overfitting.
  - `feature_fraction`, `bagging_fraction`: Tỷ lệ đặc trưng và mẫu dữ liệu được sử dụng để huấn luyện mỗi cây, thử từ $[0.5, 1.0]$.
  - `lambda_l1`, `lambda_l2`: Các hệ số regularization L1 và L2, thử từ $[0, 5]$.

5.4 Tiêu chí đánh giá mô hình

Mỗi mô hình được đánh giá dựa trên:

- **MAE (Mean Absolute Error)**: Sai số tuyệt đối trung bình.
- **RMSPE (Root Mean Square Percentage Error)**: Sai số phần trăm bình phương trung bình, phù hợp với dữ liệu lệch.
- **Thời gian huấn luyện**: Tổng thời gian để mô hình huấn luyện và dự đoán.

5.5 Mô hình tổ hợp (Ensemble)

Sau khi huấn luyện và đánh giá các mô hình riêng lẻ, nhóm tiếp tục áp dụng các kỹ thuật tổ hợp để khai thác thế mạnh của từng mô hình và cải thiện hiệu suất dự đoán. Cụ thể, hai phương pháp tổ hợp được sử dụng là: **Voting Ensemble** và **Stacking Ensemble**.

5.5.1 Voting Ensemble

Voting là phương pháp kết hợp các mô hình bằng cách lấy trung bình dự đoán đầu ra. Với bài toán hồi quy, đây là hình thức *Averaging Voting* – giá trị đầu ra cuối cùng là trung bình cộng của các giá trị dự đoán từ các mô hình con.

5.5.2 Stacking Ensemble

Stacking là kỹ thuật tổ hợp nâng cao, trong đó đầu ra của các mô hình con sẽ được sử dụng làm đặc trưng đầu vào cho một mô hình học máy khác gọi là **meta-model** (mô hình cấp cao). Meta-model học cách kết hợp các mô hình con một cách tối ưu.

6 Đánh giá mô hình

Sau khi hoàn tất quá trình EDA, tiền xử lý và huấn luyện mô hình, nhóm đã đánh giá hiệu suất của 8 mô hình gồm 3 mô hình tuyến tính, 3 mô hình boosting, và 2 mô hình ensemble.

Tiêu chí đánh giá:

- **MAE (Mean Absolute Error)** – sai số tuyệt đối trung bình.
- **RMSPE (Root Mean Square Percentage Error)** – sai số phần trăm bình phương trung bình, phù hợp với dữ liệu có phân phối lệch.
- **Thời gian huấn luyện** – tổng thời gian huấn luyện mô hình tính bằng giây.

Kết quả chi tiết được trình bày trong Bảng 1:

Bảng 1: So sánh hiệu năng các mô hình

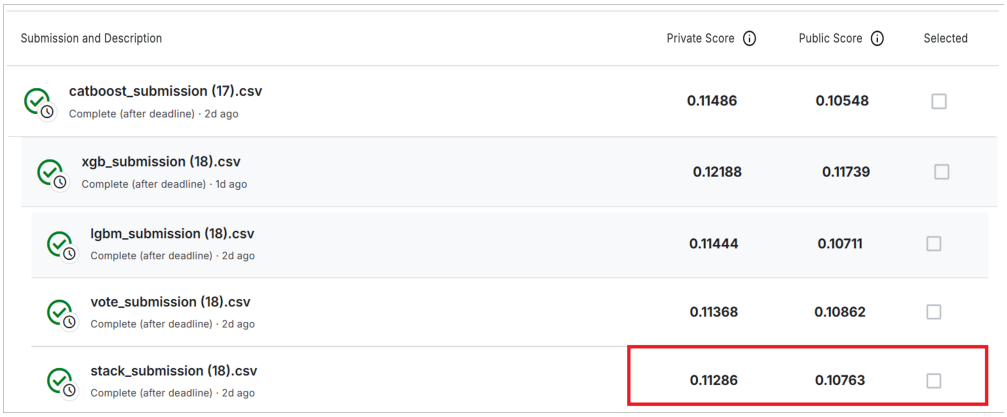
Mô hình	MAE	RMSPE	Training Time (s)
Linear Regression	0.1549	0.0254	1.992
Ridge Regression	0.1549	0.0253	0.920
Lasso Regression	0.1553	0.0253	**0.159**
XGBoost	0.1137	0.0185	85.501
CatBoost	0.1143	0.0183	62.571
LightGBM	0.1114	0.0180	58.584
Voting Regressor	0.1100	0.0179	63.114
Stacking Regressor	**0.1093**	**0.0177**	426.406

Kết quả cho thấy:

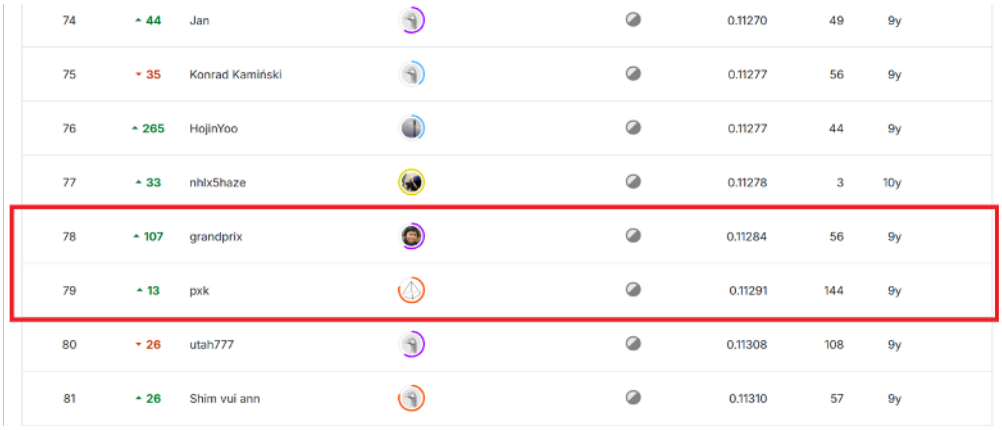
- Các mô hình boosting vượt trội rõ rệt so với tuyến tính.
- LightGBM đạt kết quả tốt nhất trong các mô hình đơn lẻ.
- Stacking Regressor cho hiệu năng cao nhất, nhưng có chi phí tính toán lớn.

Trên nền tảng Kaggle, nhóm đã gửi các mô hình và đạt kết quả tốt với Stacking:

- **Private Score:** 0.11286
- **Public Score:** 0.10763
- **Xếp hạng:** Top **79/3300** ($\approx 2.4\%$)



Hình 21: Kết quả submit trên Kaggle



Hình 22: Vị trí xếp hạng sau khi submit mô hình lên Kaggle

Tổng kết lại, pipeline và phương pháp ensemble mô hình mà nhóm triển khai đã giúp đạt thứ hạng 79/3300 ($\approx 2.4\%$) trên bảng xếp hạng private của Kaggle, vượt qua hơn 3.200 đội thi khác. Kết quả này cho thấy lựa chọn mô hình boosting và ensemble là hướng đi hiệu quả cho bài toán dự báo doanh thu trong lĩnh vực bán lẻ.

7 Kết luận

Bài toán **Rossmann Store Sales Forecasting** là một ví dụ thực tiễn giàu thử thách với dữ liệu lớn, đa dạng và mang tính chuỗi thời gian. Trong suốt quá trình thực hiện, nhóm đã triển khai toàn bộ quy trình học máy: từ EDA, tiền xử lý dữ liệu, khám phá đặc trưng, chọn mô hình, tối ưu siêu tham số đến ensemble và đánh giá mô hình.

Những kết quả nổi bật:

- Việc xử lý dữ liệu hiệu quả (log-transform, robust-scaling, mã hóa đặc trưng thời gian...) đã cải thiện đáng kể đầu vào cho mô hình.
- Các mô hình boosting như LightGBM và CatBoost thể hiện hiệu suất vượt trội.
- Kỹ thuật tổ hợp (Voting và Stacking) giúp cải thiện kết quả đáng kể, với Stacking đạt hiệu năng tốt nhất (RMSPE = 0.0177).
- Submit lên Kaggle đạt Top 79/3300 – một kết quả vượt hơn mong đợi.

Hướng phát triển trong tương lai:

- Bổ sung dữ liệu từ bên ngoài (thời tiết, kinh tế, xu hướng tiêu dùng).
- Thử nghiệm các mô hình học sâu cho chuỗi thời gian như **ARIMA** hoặc **Prophet** để so sánh hiệu quả với các mô hình truyền thống.
- Tự động hóa quá trình tối ưu mô hình với AutoML.

Từ bài toán này, nhóm không chỉ áp dụng lý thuyết đã học vào thực tế mà còn rèn luyện được kỹ năng xử lý dữ liệu, thiết kế pipeline và đánh giá mô hình toàn diện – nền tảng vững chắc cho các dự án học máy trong tương lai.

Tài liệu tham khảo

[1] Kaggle. (2015). *Rossmann Store Sales*. Truy cập từ: https://www.kaggle.com/c/rossmann-store-sales

[2] Taylor, S. J., & Letham, B. (2018). *Forecasting at scale*. The American Statistician, 72(1), 37–45.

[3] Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.

[4] Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. In Proceedings of the 22nd ACM SIGKDD (pp. 785–794).

[5] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree*. In Advances in Neural Information Processing Systems (NeurIPS).

[6] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A next-generation hyperparameter optimization framework*. In Proceedings of the 25th ACM SIGKDD (pp. 2623–2631).