

**KHOA CÔNG NGHỆ THÔNG TIN**



**MÔN:** Phân tích dữ liệu

**MÃ MÔN HỌC:**

**BÁO CÁO PHÂN TÍCH &**

**DỰ ĐOÁN DỮ LIỆU GIAO HÀNG**

**Nhóm sinh viên thực hiện:** Nhóm 08

**Học kỳ:** II

**Năm học:** 2024 – 2025

# **DANH SÁCH SINH VIÊN NHÓM THỰC HIỆN**

**HỌC KÌ II NĂM HỌC 2024 – 2025**

**Nhóm 08**

**Đề tài: BÁO CÁO PHÂN TÍCH & DỰ ĐOÁN DỮ LIỆU GIAO HÀNG**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nhiệm vụ** | **Lâm Trung Hiếu** | **Võ Thị Thu Hằng** | **Lâm Hoài Kim Ngân** | **Lê Hoàng Anh Duy** |
| Giới thiệu dự án | **X** | **X** |  |  |
| Tìm kiếm dữ liệu | **X** | **X** | **X** | **X** |
| Xử lý dữ liệu | **X** | **X** | **X** | **X** |
| **EDA:** Phân tích thời gian giao hàng và hiệu suất |  |  |  | **X** |
| **EDA:** Phân tích hiệu suất của các khu vực (tùy chọn attributes) | **X** |  |  |  |
| **EDA:** Phân tích chi phí giao hàng |  |  | **X** |  |
| **EDA:** Phân tích theo khách hàng→ khách hàng thân quen, đề xuất khuyến mãi | **X** |  |  |  |
| **EDA:** Phân tích hiệu quả của tài xế giao hàng |  |  |  | **X** |
| **EDA:** Phân tích số liệu theo thời gian | **X** |  |  |  |
| **EDA:** Phân tích thông tin về địa chỉ giao hàng |  | **X** |  |  |
| **EDA:** Phân tích mối quan hệ giữa các yếu tố logistics |  | **X** |  |  |
| **EDA:** Phân tích tỷ lệ hủy đơn hàng |  | **X** | **X** |  |
| **EDA:** Phân tích chi phí do hủy đơn hàng |  |  |  | **X** |
| **Xây dựng mô hình dự đoán:** thời gian từ lúc tạo đơn đến lúc hoàn thành |  | **X** | **X** |  |
| **Xây dựng mô hình dự đoán:** thời gian từ lúc tạo đơn đến lúc hoàn thành | **X** |  |  | **X** |
| **KẾT LUẬN** |  |  | **X** | **X** |

**MỤC LỤC**

[**DANH SÁCH SINH VIÊN NHÓM THỰC HIỆN** 2](#_Toc198139654)

[**I. Tổng quan về dự án** 6](#_Toc198139655)

[**1.1. Giới thiệu vấn đề** 6](#_Toc198139656)

[**1.2. Giới thiệu về tập dữ liệu** 6](#_Toc198139657)

[**II. Phân tích dữ liệu** 8](#_Toc198139658)

[**1.Tiền xử lý dữ liệu** 8](#_Toc198139659)

[**2. Nhập dữ liệu & khám phá sơ bộ (Data Import & Exploration)** 12](#_Toc198139660)

[**2.1. Phân tích theo thời gian** 12](#_Toc198139661)

[**2.1.1. Phân tích thời gian giao hàng và hiệu suất** 12](#_Toc198139662)

[**2.1.2. Phân tích số liệu theo tháng ,giờ** 24](#_Toc198139663)

[**2.1.3. Phân tích theo khách hàng→ khách hàng thân quen, đề xuất khuyến mãi** 27](#_Toc198139664)

[**2.1.4. Phân tích hiệu quả của tài xế giao hàng** 30](#_Toc198139665)

[**2.2. Phân tích theo phí và khu vực** 39](#_Toc198139666)

[**2.2.1.Phân tích hiệu suất của các khu vực** 39](#_Toc198139667)

[**2.2.2. Phân tích chi phí giao hàng** 40](#_Toc198139668)

[**2.2.3. Phân tích mối quan hệ giữa các yếu tố logistics** 44](#_Toc198139669)

[**2.2.3.1. Tính toán các thuộc tính thời gian liên quan** 44](#_Toc198139670)

[**2.2.3.2. Tìm ra sự tương quan giữa các yếu tố như khu vực, chi phí và thời gian giao hàng để tối ưu hóa các quy trình trong hệ thống logistics.** 48](#_Toc198139671)

[**2.2.4. Phân tích thông tin về địa chỉ giao hàng** 54](#_Toc198139672)

[**2.3. Phân tích theo hành vi khách hàng và trạng thái đơn hàng** 60](#_Toc198139673)

[**2.3.1. Phân tích tỷ lệ hủy đơn hàng** 60](#_Toc198139674)

[**2.3.1.1. Phân Tích Số Đơn Hàng Theo Trạng Thái và Khu Vực** 60](#_Toc198139675)

[**2.3.1.2. Tỷ lệ hủy đơn theo khu vực** 63](#_Toc198139676)

[**2.3.1.3. Tỷ lệ hủy đơn theo giá trị đơn hàng** 64](#_Toc198139677)

[**2.3.1.4. Top 10 lý do hủy đơn nhiều nhất** 67](#_Toc198139678)

[**2.3.2. Phân tích chi phí do hủy đơn hàng** 70](#_Toc198139679)

[**3. Xây dựng mô hình dự đoán** 76](#_Toc198139680)

[**1. Dự đoán thời gian từ lúc tạo đơn đến lúc hoàn thành** 76](#_Toc198139681)

[**2. Dự đoán Tổng phí trả tài xế dựa trên các đặc trưng như khoảng cách, thời gian, km** 80](#_Toc198139682)

[**4.1 Dự đoán trạng thái đơn hàng (trạng\_thái\_đơn hoặc trang\_thai\_chuyến) là HOÀN THÀNH hay ĐÃ HUỶ. Theo mô hình LogisticRegression** 82](#_Toc198139683)

[**4.2 . Dự đoán Tổng phí trả tài xế dựa trên các đặc trưng như khoảng cách, thời gian, km** 83](#_Toc198139684)

[**III. Kết luận** 84](#_Toc198139685)

[**1. Kết luận:** 84](#_Toc198139686)

[**2. Hướng mở rộng:** 85](#_Toc198139687)

[**IV. Phụ lục** 86](#_Toc198139688)

**BÁO CÁO PHÂN TÍCH & DỰ ĐOÁN DỮ LIỆU GIAO HÀNG**

# **I. Tổng quan về dự án**

## **1.1. Giới thiệu vấn đề**

Hiện nay, cùng với sự phát triển mạnh mẽ của thương mại điện tử tại Việt Nam, nhu cầu giao nhận hàng hóa cũng ngày càng tăng cao. Điều này tạo ra áp lực lớn cho các doanh nghiệp logistics trong việc tối ưu hóa quy trình giao hàng, đảm bảo đúng hẹn, tiết kiệm chi phí và nâng cao trải nghiệm của khách hàng. Để làm được điều đó, việc phân tích và khai thác dữ liệu từ hệ thống vận hành thực tế đóng vai trò rất quan trọng.

Dữ liệu được sử dụng trong báo cáo này được lấy từ hệ thống quản lý đơn hàng và chuyến giao hàng của một đơn vị giao hàng nội thành, với gần 100.000 dòng dữ liệu. Mỗi dòng chứa thông tin chi tiết về đơn hàng, tài xế, địa điểm giao – nhận, thời gian thực hiện, chi phí vận chuyển và các trạng thái xử lý đơn. Tập dữ liệu gồm 42 cột, phản ánh khá đầy đủ quá trình vận hành của một đơn hàng từ lúc tạo đến khi hoàn tất.

Thông qua phân tích dữ liệu, nhóm thực hiện nhằm phát hiện các vấn đề tiềm ẩn như tỷ lệ hủy đơn, thời gian xử lý đơn bất thường, hay mối tương quan giữa khu vực hoạt động và hiệu quả giao hàng. Kết quả phân tích sẽ là cơ sở đề xuất các cải tiến trong hoạt động giao nhận, cũng như hỗ trợ ra quyết định trong quản lý vận hành và chiến lược phát triển dịch vụ.

Ngoài ra, nhóm còn xây dựng một số mô hình dự báo phục vụ cho việc hỗ trợ các quyết định quan trọng trong ngành logistics. Việc áp dụng các mô hình này không chỉ giúp doanh nghiệp nắm bắt xu hướng, mà còn giúp vận hành quy trình một cách linh hoạt và hiệu quả hơn.

Dựa vào các phân tích này cũng sẽ giúp nhận diện các yếu tố tác động đến trải nghiệm khách hàng và đề xuất các biện pháp để nâng cao sự hài lòng của người tiêu dùng. Thông qua việc áp dụng các mô hình dự đoán, doanh nghiệp có thể cải thiện các chiến lược vận hành, giảm thiểu rủi ro và chi phí không cần thiết, đồng thời nâng cao hiệu quả dịch vụ giao nhận.

## **1.2. Giới thiệu về tập dữ liệu**

**Bộ dữ liệu:** Bảng thống kê chi tiết hệ thống giao hàng tháng 12/2022

**Nguồn dữ liệu**: Dataset giao hàng từ hệ thống nội bộ

**Các thuộc tính chính**:

* **SĐT người tạo đơn** : (Thường là nhân viên, cửa hàng).
* **Người tạo đơn** : Tên nhân viên.
* **Siêu thị** : Tên siêu thị/cửa hàng.
* **Khu vực** : Khu vực địa lý mà đơn hàng thuộc về.
* **Mã chuyến** : Mã chuyến giao hàng.
* **Ngày tạo đơn** : Ngày mà đơn hàng được tạo.
* **Mã đơn hàng** : ID đơn hàng cụ thể.
* **Giá trị đơn hàng** : Giá trị tiền hàng trong đơn.
* **Địa chỉ lấy hàng** : Địa điểm lấy hàng.
* **Địa chỉ khách** : Địa chỉ giao đến khách hàng.
* **Địa chỉ chi tiết** : Ghi rõ hơn về địa chỉ khách.
* **Ghi chú** : Các lưu ý đặc biệt cho đơn hàng.
* **Tài xế** : Tên tài xế nhận giao đơn.
* **SĐT** : Số điện thoại của tài xế.
* **km đơn** : Số km vận chuyển của đơn hàng.
* **km chuyến** : Số km của toàn chuyến xe.
* **Mã dịch vụ** : Mã loại dịch vụ giao hàng.
* **trạng\_thái\_đơn** : Trạng thái hiện tại của đơn (ví dụ: đã giao, hủy, thất bại...).
* **trang\_thai\_chuyến** : Trạng thái chuyến xe (ví dụ: đang giao, đã hoàn tất, hủy chuyến...).
* **ly\_do\_huy** : Lý do hủy đơn hoặc chuyến nếu có.
* **Đơn ưu tiên** : Đơn hàng ưu tiên (có thể được giao nhanh hơn đơn thường).
* **Hạng thành viên** : Hạng của khách hàng (ví dụ: đồng, bạc, vàng, kim cương).
* **Phí vượt km** : Phụ phí tính thêm nếu quãng đường vượt tiêu chuẩn dịch vụ.
* **Phụ phí thu khách** : Các phụ phí khách hàng cần trả thêm.
* **Mô tả phụ phí KH** : Mô tả chi tiết phụ phí thu từ khách hàng.
* **Tổng phí thu khách** : Tổng phí khách phải trả (phụ phí).
* **Thu hộ COD** : Tiền hàng cần thu hộ (Cash On Delivery).
* **Tổng thu khách hàng** : Tổng số tiền tài xế thu từ khách (COD + phụ phí).
* **Phụ phí trả tài xế** : Phụ phí công ty trả thêm cho tài xế.
* **Mô tả phụ phí TX** : Mô tả chi tiết phụ phí trả tài xế.
* **Số chuyến** : Số lượng chuyến xe.
* **Đơn giá chuyến** : Giá tiền của một chuyến xe.
* **Phí chuyến cho tài** : Phí chuyến trả cho tài xế.
* **Tổng phí trả tài xế** : Tổng số tiền tài xế nhận được.
* **Thời điểm tạo đơn** : Thời điểm tạo đơn (ngày giờ).
* **Thời điểm tài xế nhận đơn** : Thời điểm tài xế nhận đơn giao.
* **Thời điểm tài xế lấy hàng** : Thời điểm tài xế lấy hàng thành công.
* **Thời gian hoàn thành đơn** : Thời gian đơn hàng giao thành công.
* **pod** : Proof of Delivery - bằng chứng giao hàng.
* **por** : Proof of Return - bằng chứng trả hàng (nếu có).
* **tracking\_link** : Link theo dõi đơn hàng online.
* **sdt\_khach** : Số điện thoại khách hàng.

# **II. Phân tích dữ liệu**

## **1.Tiền xử lý dữ liệu**

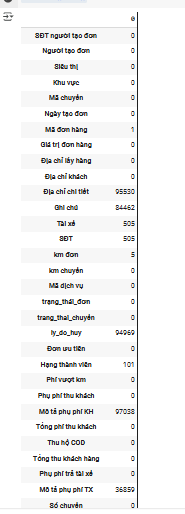
* Xử lý dữ liệu thiếu và bất thường.
* **Kiểm tra giá trị trùng lặp**

|  |
| --- |
| * df.duplicated().sum() |

****

* **Kiểm tra các giá trị bị thiếu**

|  |
| --- |
| * df.isnull().sum() |



* **Loại những cột không cần thiết**

|  |
| --- |
| * #df.drop(columns=['Địa chỉ chi tiết','Ghi chú','Mô tả phụ phí KH','pod','por'], inplace=True) |

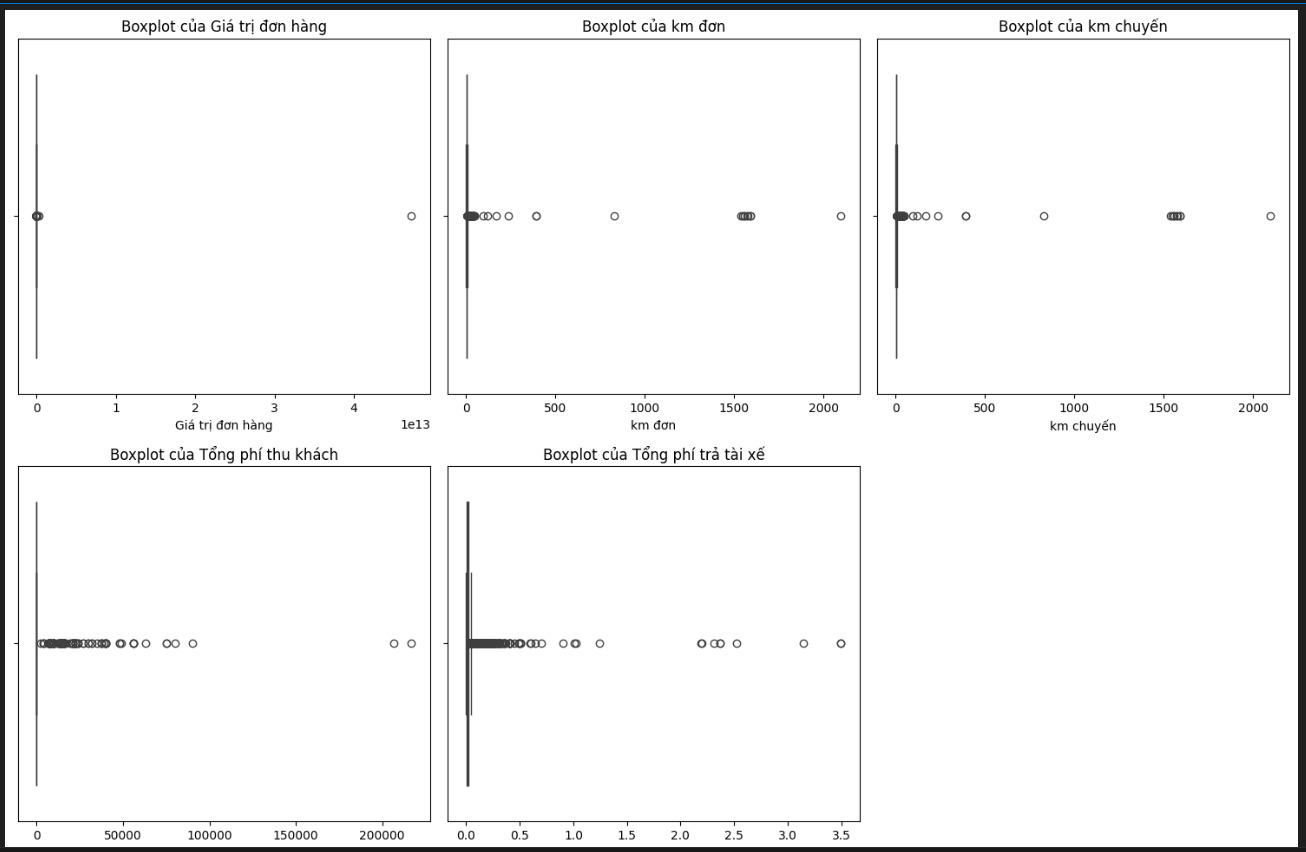
* Chuẩn hóa dữ liệu nếu cần thiết.

**Định dạng các cột thời gian thành dạng datetime**

|  |
| --- |
| * date\_cols = ['Ngày tạo đơn', 'Thời điểm tạo đơn', 'Thời điểm tài xế nhận đơn', 'Thời điểm tài xế lấy hàng', 'Thời gian hoàn thành đơn'] * for col in date\_cols: * df[col] = pd.to\_datetime(df[col], errors='coerce') * df['Thu hộ COD'].replace({'Có': 1, 'Không': 0}, inplace=True) * df['Đơn ưu tiên'].replace({'Có': 1, 'Không': 0}, inplace=True) |

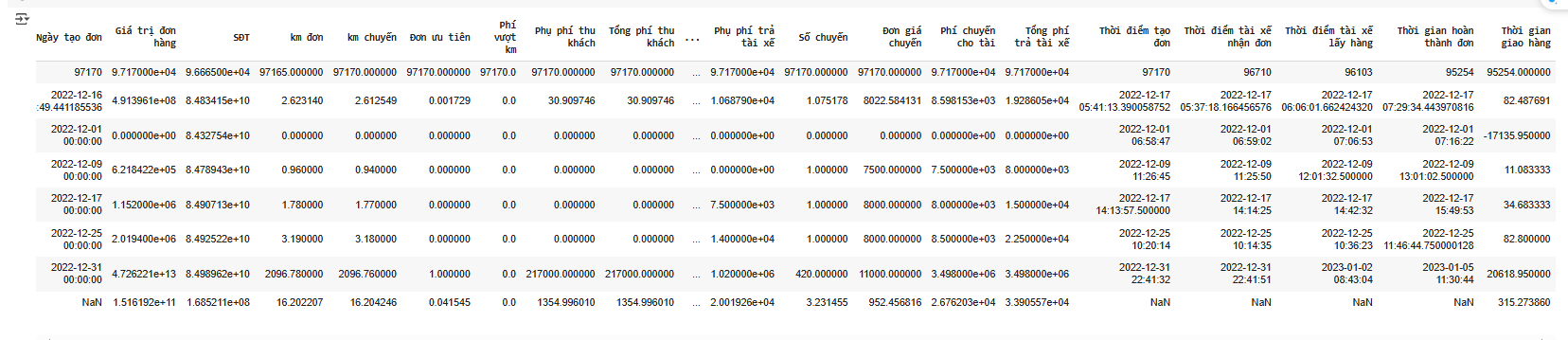
* **Kiểm tra giá trị ngoại lệ :**

|  |
| --- |
| * columns\_to\_check = ['Giá trị đơn hàng', 'km đơn', 'km chuyến', 'Tổng phí thu khách', 'Tổng phí trả tài xế'] * # Vẽ boxplot cho từng cột * plt.figure(figsize=(15, 10)) * for i, col in enumerate(columns\_to\_check, 1): * plt.subplot(2, 3, i)  # 2 hàng, 3 cột * sns.boxplot(data=df, x=col) * plt.title(f'Boxplot của {col}') * plt.tight\_layout()  # Đảm bảo không bị chồng lấn * plt.show() |



* **Thêm cột thời gian giao hàng**

|  |
| --- |
| **df['Thời gian giao hàng'] = (**  **df['Thời gian hoàn thành đơn'] - df['Thời điểm tài xế lấy hàng']**  **).dt.total\_seconds() / 60** |

****

* **Tạo các cột cần thiết và xử lý outlier**

|  |
| --- |
| **# Tạo bản sao dữ liệu đã xử lý**  **df\_processed = df.copy()**  **# Chuyển đổi các cột thời gian sang dạng datetime**  **df\_processed['Ngày tạo đơn'] = pd.to\_datetime(df\_processed['Ngày tạo đơn'], errors='coerce')**  **df\_processed['Thời gian hoàn thành đơn'] = pd.to\_datetime(df\_processed['Thời gian hoàn thành đơn'], errors='coerce')**  **df\_processed['Thời điểm tạo đơn'] = pd.to\_datetime(df\_processed['Thời điểm tạo đơn'], errors='coerce')**  **df\_processed['Thời điểm tài xế nhận đơn'] = pd.to\_datetime(df\_processed['Thời điểm tài xế nhận đơn'], errors='coerce')**  **df\_processed['Thời điểm tài xế lấy hàng'] = pd.to\_datetime(df\_processed['Thời điểm tài xế lấy hàng'], errors='coerce')**  **# Tính toán thời gian giao hàng (tính bằng giờ)**  **df\_processed['Thời gian giao hàng (giờ)'] = (df\_processed['Thời gian hoàn thành đơn'] - df\_processed['Thời điểm tạo đơn']).dt.total\_seconds() / 3600**  **df\_processed['Thời gian nhận đơn (phút)'] = (df\_processed['Thời điểm tài xế nhận đơn'] - df\_processed['Thời điểm tạo đơn']).dt.total\_seconds() / 60**  **df\_processed['Thời gian lấy hàng (phút)'] = (df\_processed['Thời điểm tài xế lấy hàng'] - df\_processed['Thời điểm tài xế nhận đơn']).dt.total\_seconds() / 60**  **# Loại bỏ các dòng có giá trị thiếu (NaT) trong các cột thời gian**  **df\_processed = df\_processed.dropna(subset=['Ngày tạo đơn', 'Thời gian hoàn thành đơn'])**  **# \*\*Xử lý outlier bằng phương pháp IQR (Interquartile Range)\*\***  **Q1 = df\_processed['Thời gian giao hàng (giờ)'].quantile(0.25)**  **Q3 = df\_processed['Thời gian giao hàng (giờ)'].quantile(0.75)**  **IQR = Q3 - Q1**  **# Xác định các giá trị ngoại lai**  **lower\_bound = Q1 - 1.5 \* IQR**  **upper\_bound = Q3 + 1.5 \* IQR**  **# Lọc dữ liệu để loại bỏ các giá trị ngoại lai**  **df\_processed = df\_processed[(df\_processed['Thời gian giao hàng (giờ)'] >= lower\_bound) & (df\_processed['Thời gian giao hàng (giờ)'] <= upper\_bound)]**  **# Xử lý outlier cho 'Thời gian nhận đơn (phút)'**  **Q1\_receive = df\_processed['Thời gian nhận đơn (phút)'].quantile(0.25)**  **Q3\_receive = df\_processed['Thời gian nhận đơn (phút)'].quantile(0.75)**  **IQR\_receive = Q3\_receive - Q1\_receive**  **# Xác định các giá trị ngoại lai cho 'Thời gian nhận đơn (phút)'**  **lower\_bound\_receive = Q1\_receive - 1.5 \* IQR\_receive**  **upper\_bound\_receive = Q3\_receive + 1.5 \* IQR\_receive**  **# Lọc dữ liệu để loại bỏ các giá trị ngoại lai trong 'Thời gian nhận đơn (phút)'**  **df\_processed = df\_processed[(df\_processed['Thời gian nhận đơn (phút)'] >= lower\_bound\_receive) &**  **(df\_processed['Thời gian nhận đơn (phút)'] <= upper\_bound\_receive)]**  **# Xử lý outlier cho 'Thời gian lấy hàng (phút)'**  **Q1\_pickup = df\_processed['Thời gian lấy hàng (phút)'].quantile(0.25)**  **Q3\_pickup = df\_processed['Thời gian lấy hàng (phút)'].quantile(0.75)**  **IQR\_pickup = Q3\_pickup - Q1\_pickup**  **# Xác định các giá trị ngoại lai cho 'Thời gian lấy hàng (phút)'**  **lower\_bound\_pickup = Q1\_pickup - 1.5 \* IQR\_pickup**  **upper\_bound\_pickup = Q3\_pickup + 1.5 \* IQR\_pickup**  **# Lọc dữ liệu để loại bỏ các giá trị ngoại lai trong 'Thời gian lấy hàng (phút)'**  **df\_processed = df\_processed[(df\_processed['Thời gian lấy hàng (phút)'] >= lower\_bound\_pickup) &**  **(df\_processed['Thời gian lấy hàng (phút)'] <= upper\_bound\_pickup)]**  **# Kiểm tra số lượng giá trị ngoại lai sau khi xử lý**  **outliers\_receive = df[~df.index.isin(df\_processed.index)]**  **outliers\_pickup = df[~df.index.isin(df\_processed.index)]** |

## **2. Nhập dữ liệu & khám phá sơ bộ (Data Import & Exploration)**

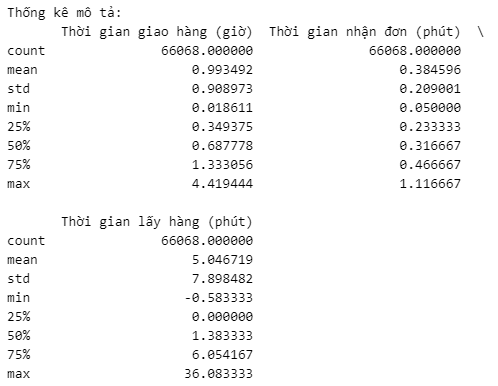
### **2.1. Phân tích theo thời gian**

#### **2.1.1. Phân tích thời gian giao hàng và hiệu suất**

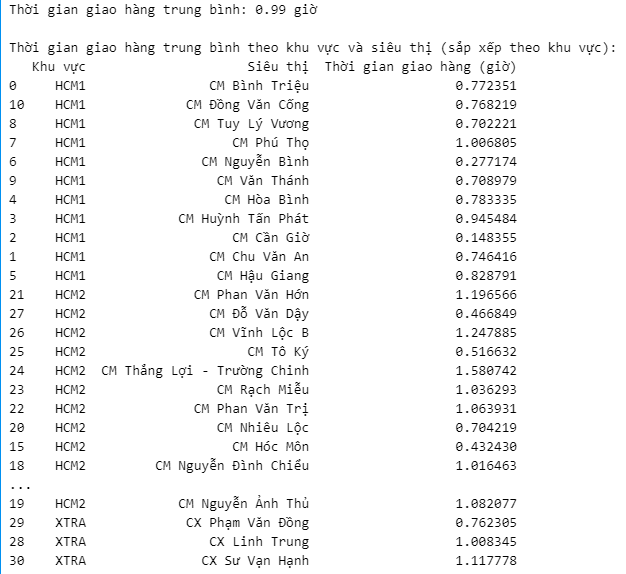
**- Thuộc tính sử dụng:** Thời gian hoàn thành đơn, Thời điểm tạo đơn, Thời điểm tài xế nhận đơn, Thời điểm tài xế lấy hàng

**- Phân tích:** Tính toán thời gian giao hàng, thời gian nhận đơn, thời gian lấy hàng của từng khu vực và siêu thị. So sánh hiệu suất giữa các khu vực và siêu thị

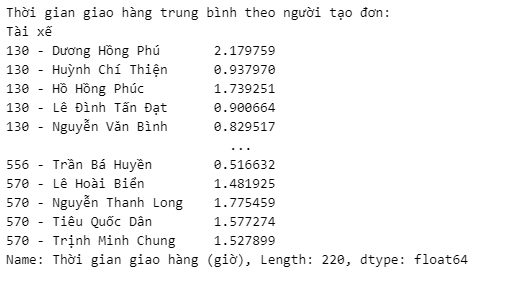
|  |
| --- |
| # Tính toán các thống kê mô tả  summary\_stats = df\_processed[['Thời gian giao hàng (giờ)', 'Thời gian nhận đơn (phút)', 'Thời gian lấy hàng (phút)']].describe()  print("Thống kê mô tả:")  print(summary\_stats) |



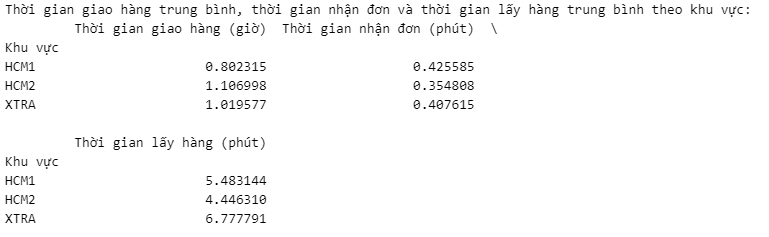
|  |
| --- |
| # Phân tích thời gian giao hàng trung bình  average\_delivery\_time = df\_processed['Thời gian giao hàng (giờ)'].mean()  print(f"Thời gian giao hàng trung bình: {average\_delivery\_time:.2f} giờ")  # Phân tích theo phương thức giao hàng (nếu có cột liên quan, ví dụ 'Siêu thị' hoặc 'Mã chuyến')  # Kiểm tra cột 'Siêu thị' và 'Khu vực' có tồn tại không  if 'Siêu thị' in df\_processed.columns and 'Khu vực' in df\_processed.columns:  # Nhóm theo 'Khu vực' và 'Siêu thị', tính thời gian giao hàng trung bình  delivery\_time\_by\_region\_and\_store = df\_processed.groupby(['Khu vực', 'Siêu thị'])['Thời gian giao hàng (giờ)'].mean().reset\_index()  # Sắp xếp theo 'Khu vực' (theo thứ tự bảng chữ cái)  delivery\_time\_by\_region\_and\_store = delivery\_time\_by\_region\_and\_store.sort\_values(by='Khu vực')  print("\nThời gian giao hàng trung bình theo khu vực và siêu thị (sắp xếp theo khu vực):")  print(delivery\_time\_by\_region\_and\_store)  else:  print("\nKhông tìm thấy cột 'Siêu thị' hoặc 'Khu vực' trong dữ liệu.") |



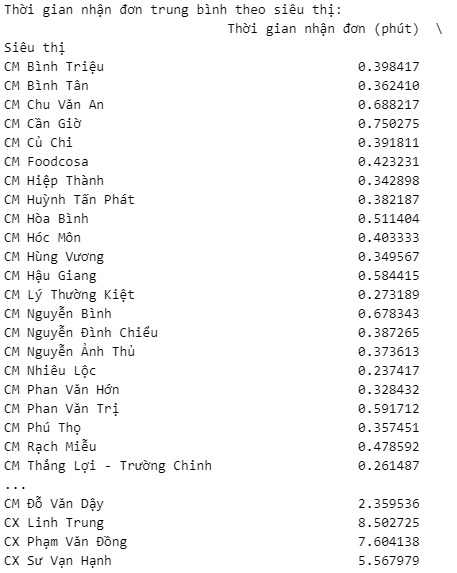
|  |
| --- |
| # Phân tích theo nhân viên giao hàng (nếu có cột liên quan, ví dụ 'Người tạo đơn')  if 'Tài xế' in df\_processed.columns:  delivery\_time\_by\_driver = df\_processed.groupby('Tài xế')['Thời gian giao hàng (giờ)'].mean()  print("\nThời gian giao hàng trung bình theo người tạo đơn:")  print(delivery\_time\_by\_driver) |



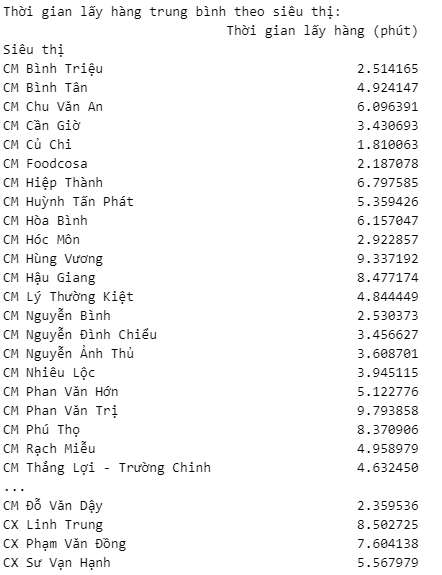
|  |
| --- |
| # Phân tích thời gian giao hàng, thời gian nhận đơn và thời gian lấy hàng trung bình theo khu vực  if 'Khu vực' in df\_processed.columns:  delivery\_time\_by\_region = df\_processed.groupby('Khu vực')[['Thời gian giao hàng (giờ)', 'Thời gian nhận đơn (phút)', 'Thời gian lấy hàng (phút)']].mean()  # In kết quả thời gian giao hàng, nhận đơn và lấy hàng trung bình theo khu vực  print("Thời gian giao hàng trung bình, thời gian nhận đơn và thời gian lấy hàng trung bình theo khu vực:")  print(delivery\_time\_by\_region) |



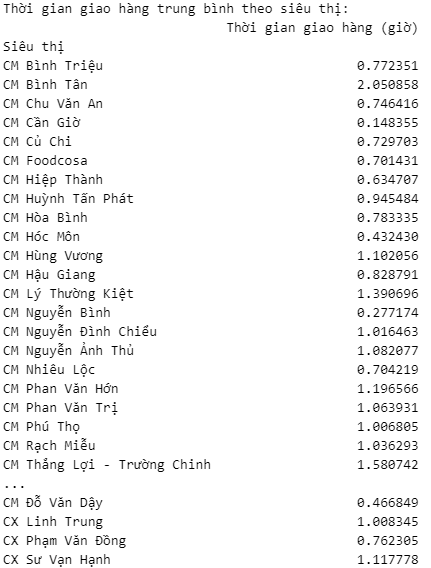
|  |
| --- |
| # Phân tích thời gian giao hàng, thời gian nhận đơn và thời gian lấy hàng trung bình theo khu vực  if 'Khu vực' in df\_processed.columns:  delivery\_time\_by\_market = df\_processed.groupby('Siêu thị')[['Thời gian nhận đơn (phút)', 'Thời gian lấy hàng (phút)']].mean()  # In kết quả thời gian giao hàng, nhận đơn và lấy hàng trung bình theo khu vực  print("Thời gian nhận đơn trung bình theo siêu thị:")  print(delivery\_time\_by\_market) |



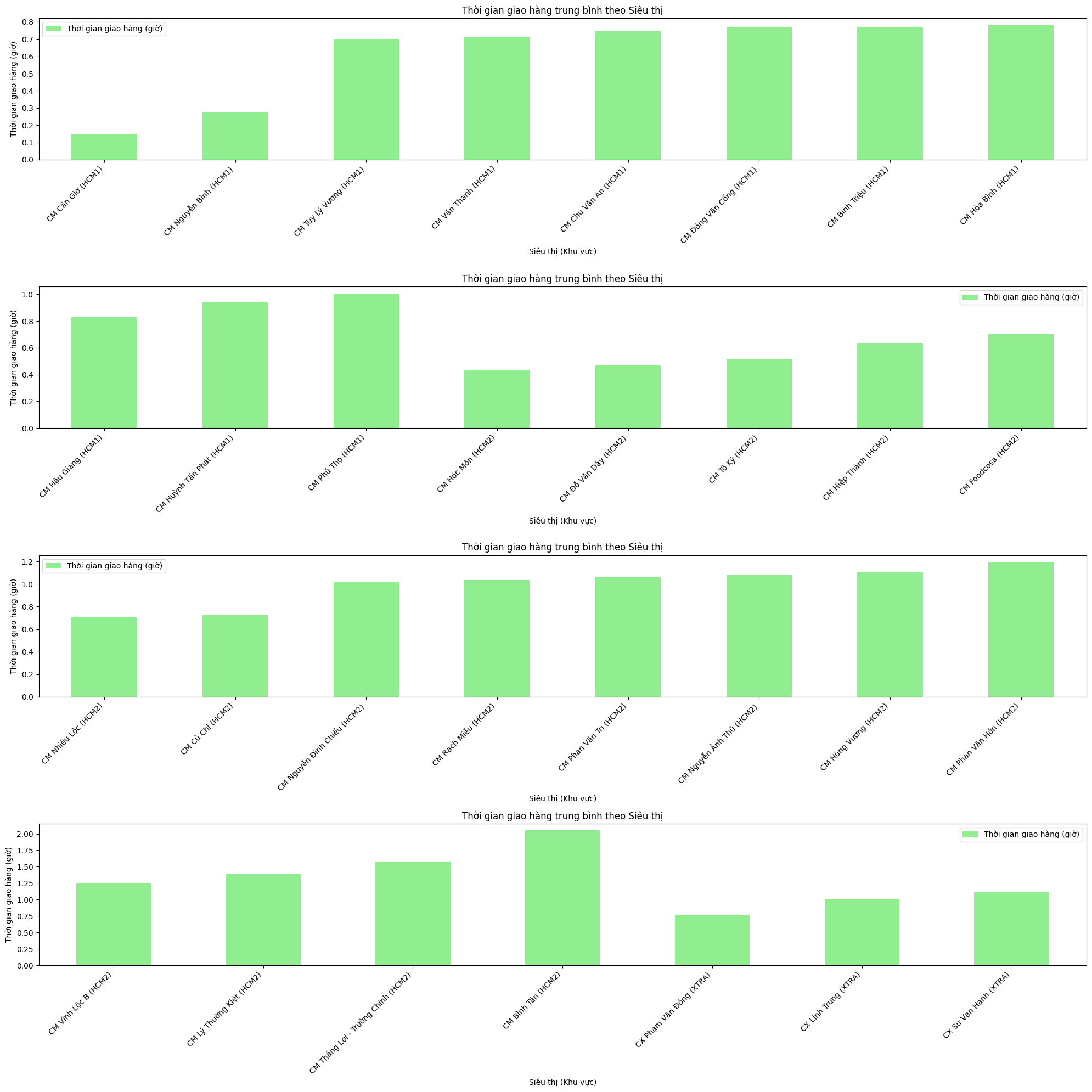
|  |
| --- |
| # Phân tích thời gian giao hàng, thời gian nhận đơn và thời gian lấy hàng trung bình theo khu vực  if 'Khu vực' in df\_processed.columns:  delivery\_time\_by\_market = df\_processed.groupby('Siêu thị')[['Thời gian lấy hàng (phút)']].mean()  # In kết quả thời gian giao hàng, nhận đơn và lấy hàng trung bình theo khu vực  print("Thời gian lấy hàng trung bình theo siêu thị:")  print(delivery\_time\_by\_market) |



|  |
| --- |
| # Phân tích thời gian giao hàng, thời gian nhận đơn và thời gian lấy hàng trung bình theo khu vực  if 'Khu vực' in df\_processed.columns:  delivery\_time\_by\_market = df\_processed.groupby('Siêu thị')[['Thời gian giao hàng (giờ)']].mean()  # In kết quả thời gian giao hàng, nhận đơn và lấy hàng trung bình theo khu vực  print("Thời gian giao hàng trung bình theo siêu thị:")  print(delivery\_time\_by\_market) |



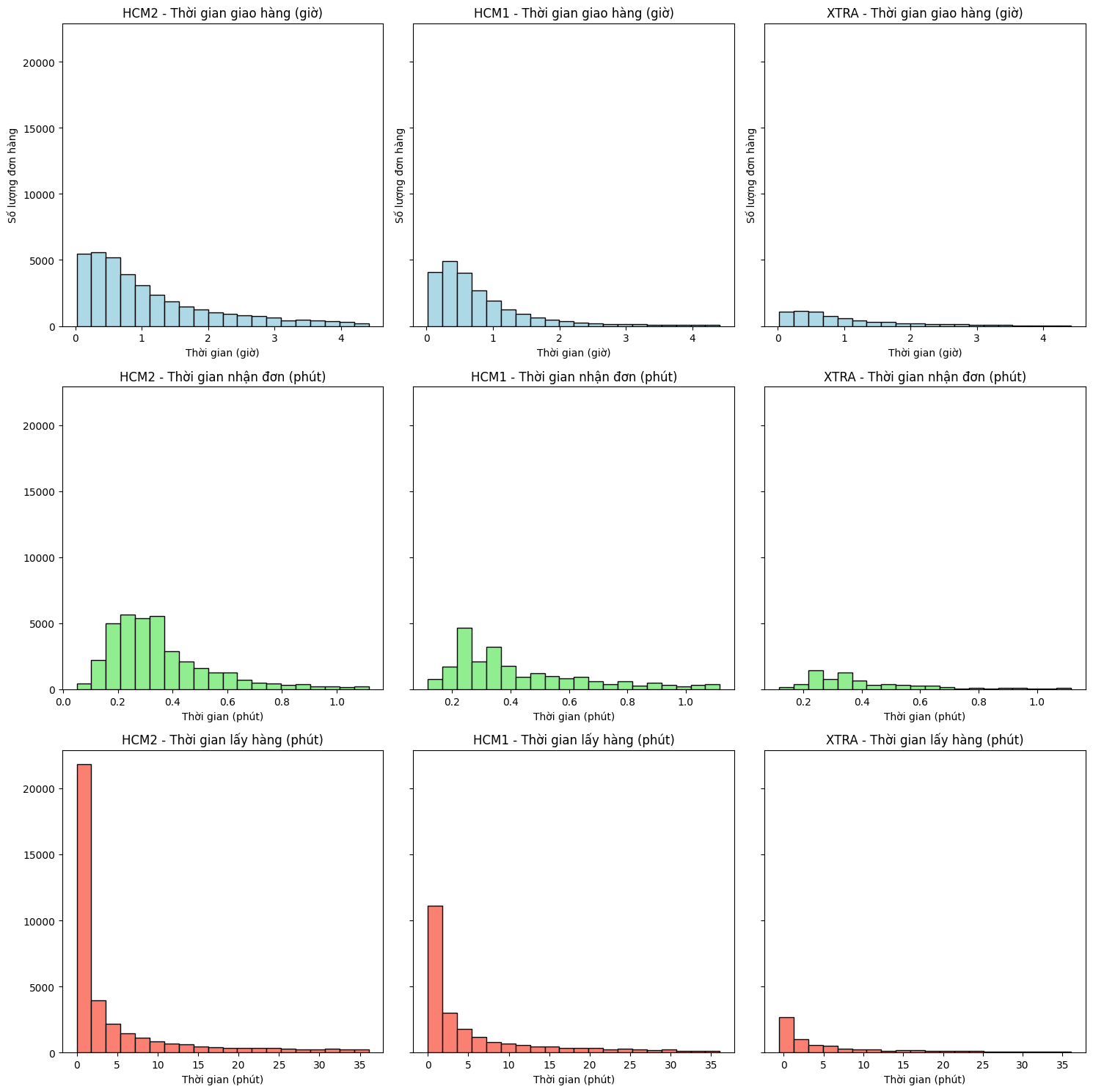
|  |
| --- |
| if 'Siêu thị' in df\_processed.columns and 'Khu vực' in df\_processed.columns:  # Nhóm theo 'Siêu thị' và 'Khu vực', tính thời gian giao hàng trung bình  delivery\_time\_by\_store = df\_processed.groupby(['Siêu thị', 'Khu vực'])['Thời gian giao hàng (giờ)'].mean().reset\_index()    # Tạo nhãn mới cho trục X với định dạng "Siêu thị (Khu vực)"  delivery\_time\_by\_store['Nhãn'] = delivery\_time\_by\_store['Siêu thị'] + ' (' + delivery\_time\_by\_store['Khu vực'] + ')'    # Sắp xếp theo 'Khu vực' (theo thứ tự bảng chữ cái) và sau đó theo thời gian giao hàng trung bình  delivery\_time\_by\_store = delivery\_time\_by\_store.sort\_values(by=['Khu vực', 'Thời gian giao hàng (giờ)'])    # Chia dữ liệu thành các nhóm nhỏ với mỗi nhóm có khoảng 8 siêu thị  unique\_stores = delivery\_time\_by\_store.index  num\_groups = (len(unique\_stores) // 8) + 1 # Số nhóm siêu thị (mỗi nhóm 8 siêu thị)  # Tạo subplots cho các nhóm siêu thị  fig, axes = plt.subplots(num\_groups, 1, figsize=(20, 5 \* num\_groups))  # Đảm bảo axes là danh sách nếu chỉ có 1 nhóm  if num\_groups == 1:  axes = [axes]  # Vẽ từng biểu đồ cho từng nhóm siêu thị  for i in range(num\_groups):  start\_idx = i \* 8 # Vị trí bắt đầu của nhóm  end\_idx = start\_idx + 8 # Vị trí kết thúc của nhóm  group\_data = delivery\_time\_by\_store.iloc[start\_idx:end\_idx] # Dữ liệu của nhóm siêu thị  # Vẽ biểu đồ cột cho nhóm siêu thị  ax = axes[i] # Sử dụng subplot tương ứng  group\_data.plot(kind='bar', x='Nhãn', y='Thời gian giao hàng (giờ)', color='lightgreen', ax=ax)  # Tiêu đề và nhãn  ax.set\_title(f'Thời gian giao hàng trung bình theo Siêu thị')  ax.set\_xlabel('Siêu thị (Khu vực)')  ax.set\_ylabel('Thời gian giao hàng (giờ)')  # Đảm bảo rằng trục X nhận các nhãn siêu thị phân loại  ax.set\_xticks(range(len(group\_data))) # Vị trí các nhãn trên trục X  ax.set\_xticklabels(group\_data['Nhãn'], rotation=45, ha='right')  plt.tight\_layout() # Đảm bảo các biểu đồ không bị chồng lên nhau  plt.show() # Hiển thị biểu đồ |



**Nhận xét:**

* Tổng quan: Thời gian giao hàng trung bình dao động từ 0.2 giờ (12 phút) đến 2.2 giờ, với phần lớn siêu thị nằm trong khoảng 0.5 đến 1.5 giờ, cho thấy hiệu suất giao hàng cơ bản khá ổn định.
* Xu hướng: Hệ thống giao hàng có hiệu suất trung bình tốt, với đa số siêu thị duy trì thời gian giao hàng dưới 1.5 giờ. Tuy nhiên, có sự biến động đáng kể giữa các siêu thị, từ những nơi giao hàng cực nhanh (dưới 0.5 giờ) đến những nơi kéo dài hơn 2 giờ, phản ánh sự không đồng đều trong quy trình hoặc điều kiện vận hành.
* Nhận xét: Các siêu thị có thời gian giao hàng thấp (dưới 0.5 giờ) có thể được hưởng lợi từ vị trí địa lý thuận lợi, mật độ tài xế cao, hoặc quy trình tối ưu hóa hiệu quả. Ngược lại, những siêu thị có thời gian giao hàng cao (trên 1.5 giờ, đặc biệt là 2.2 giờ) có thể gặp thách thức như khoảng cách xa, thiếu tài xế, hoặc lượng đơn hàng quá tải vào giờ cao điểm.

|  |
| --- |
| # Lấy danh sách các khu vực duy nhất  regions = df\_processed['Khu vực'].unique()  # Thiết lập số lượng biểu đồ dựa trên số khu vực  num\_regions = len(regions)  # Tạo figure với 3 hàng (mỗi loại thời gian) và số cột bằng số khu vực  fig, axes = plt.subplots(3, num\_regions, figsize=(5 \* num\_regions, 15), sharey=True)  # Vẽ biểu đồ phân phối cho từng loại thời gian theo khu vực  for idx, region in enumerate(regions):  # Thời gian giao hàng (giờ)  region\_data = df\_processed[df\_processed['Khu vực'] == region]['Thời gian giao hàng (giờ)'].dropna()  axes[0, idx].hist(region\_data, bins=20, color='lightblue', edgecolor='black')  axes[0, idx].set\_title(f'{region} - Thời gian giao hàng (giờ)')  axes[0, idx].set\_xlabel('Thời gian (giờ)')  axes[0, idx].set\_ylabel('Số lượng đơn hàng')  # Thời gian nhận đơn (phút)  region\_data = df\_processed[df\_processed['Khu vực'] == region]['Thời gian nhận đơn (phút)'].dropna()  axes[1, idx].hist(region\_data, bins=20, color='lightgreen', edgecolor='black')  axes[1, idx].set\_title(f'{region} - Thời gian nhận đơn (phút)')  axes[1, idx].set\_xlabel('Thời gian (phút)')  # Thời gian lấy hàng (phút)  region\_data = df\_processed[df\_processed['Khu vực'] == region]['Thời gian lấy hàng (phút)'].dropna()  axes[2, idx].hist(region\_data, bins=20, color='salmon', edgecolor='black')  axes[2, idx].set\_title(f'{region} - Thời gian lấy hàng (phút)')  axes[2, idx].set\_xlabel('Thời gian (phút)')  # Điều chỉnh layout  plt.tight\_layout()  plt.show() |



**Nhận xét:**

Thời gian giao hàng (giờ):

* HCM2 dẫn đầu với đỉnh phân phối khoảng 5,000-6,000 đơn hàng tập trung dưới 1 giờ, giảm dần khi thời gian tăng lên đến 4 giờ, cho thấy hiệu suất giao hàng khá tốt tại khu vực này.
* HCM1 có đỉnh phân phối thấp hơn, khoảng 2,000-3,000 đơn hàng, cũng tập trung dưới 1 giờ, nhưng số lượng đơn giảm dần sau đó, phản ánh hiệu suất trung bình.
* XTRA có số lượng đơn hàng rất hạn chế, với đỉnh dưới 1,000 đơn và phân bố chủ yếu dưới 2 giờ, cho thấy hoạt động giao hàng ở khu vực này không đáng kể.

Thời gian nhận đơn (phút):

* HCM2 nổi bật với đỉnh phân phối khoảng 10,000-15,000 đơn hàng tập trung dưới 0.4 phút, giảm mạnh sau 0.6 phút, cho thấy quy trình nhận đơn rất nhanh, có thể nhờ hệ thống tự động hoặc tài xế sẵn sàng.
* HCM1 có đỉnh phân phối khoảng 5,000-6,000 đơn hàng, cũng tập trung dưới 0.4 phút, nhưng số lượng thấp hơn HCM2, thể hiện hiệu suất trung bình.
* XTRA chỉ đạt đỉnh dưới 1,000 đơn hàng, phân bố dưới 0.6 phút, cho thấy hoạt động nhận đơn ở khu vực này rất yếu.

Thời gian lấy hàng (phút):

* HCM2 có đỉnh phân phối cao nhất, khoảng 15,000-20,000 đơn hàng tập trung dưới 5 phút, giảm dần sau 10 phút, cho thấy giai đoạn lấy hàng tại đây chiếm thời gian đáng kể, có thể do quy trình tại siêu thị.
* HCM1 có đỉnh khoảng 5,000-10,000 đơn hàng, tập trung dưới 10 phút, giảm dần sau đó, phản ánh thời gian lấy hàng trung bình.
* XTRA có đỉnh khoảng 2,000-3,000 đơn hàng, phân bố dưới 10 phút, cho thấy hoạt động lấy hàng ở khu vực này hạn chế.

Tổng quan và kết luận:

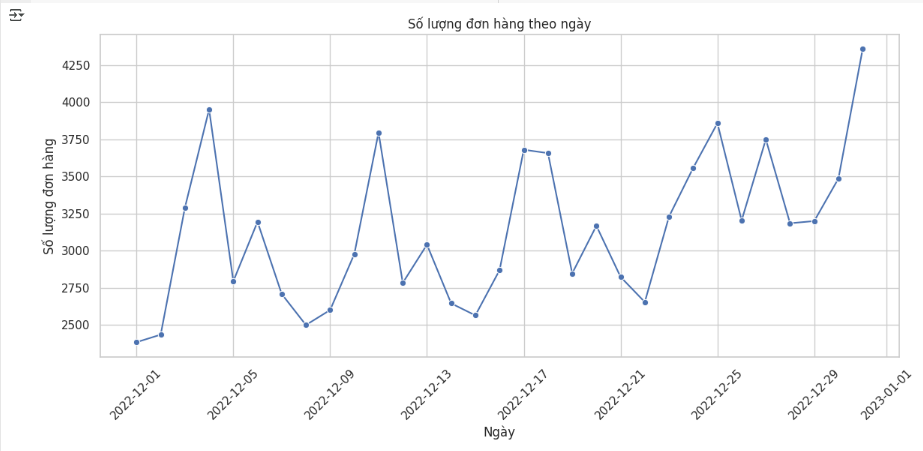
* HCM2 là khu vực hiệu quả nhất, với lưu lượng đơn hàng lớn và thời gian xử lý nhanh ở cả nhận đơn và giao hàng, nhưng giai đoạn lấy hàng cần cải thiện.
* HCM1 có hiệu suất trung bình, với số lượng đơn hàng và thời gian xử lý ở mức ổn định nhưng thấp hơn HCM2.
* XTRA có hoạt động giao hàng rất hạn chế, với số lượng đơn thấp ở tất cả các giai đoạn.
* Giai đoạn lấy hàng là điểm cần chú ý để tối ưu hóa thời gian tổng thể, đặc biệt tại HCM2.

#### **2.1.2. Phân tích số liệu theo tháng ,giờ**

**- Thuộc tính liên quan**: Ngày tạo đơn, Thời điểm tài xế nhận đơn.  
- **Phân tích**: Tính toán số lượng đơn hàng được tạo ra trong từng ngày, giờ, và xác định các khoảng thời gian có nhu cầu giao hàng cao nhất.

|  |
| --- |
| # Tính số lượng đơn hàng theo ngày  df['Ngày'] = df['Ngày tạo đơn'].dt.date # Trích xuất ngày  # Đếm số lượng đơn hàng theo ngày  orders\_by\_day = df.groupby('Ngày')['Mã đơn hàng'].count().reset\_index()  orders\_by\_day.columns = ['Ngày', 'Số lượng đơn hàng']  # Tính số lượng đơn hàng theo giờ  df['Giờ'] = df['Thời điểm tạo đơn'].dt.hour # Trích xuất giờ trong ngày  # Đếm số lượng đơn hàng theo giờ  orders\_by\_hour = df.groupby('Giờ')['Mã đơn hàng'].count().reset\_index()  orders\_by\_hour.columns = ['Giờ', 'Số lượng đơn hàng'] |

|  |
| --- |
| # Trực quan hóa số lượng đơn hàng theo ngày  plt.figure(figsize=(12, 6))  sns.lineplot(data=orders\_by\_day, x='Ngày', y='Số lượng đơn hàng', marker='o')  plt.title('Số lượng đơn hàng theo ngày')  plt.xlabel('Ngày')  plt.ylabel('Số lượng đơn hàng')  plt.xticks(rotation=45)  plt.tight\_layout()  plt.show() |



**Nhận xét :**

**1. Tính biến động cao:**

Số lượng đơn hàng dao động mạnh qua các ngày, từ khoảng 2400 đến hơn 4300 đơn/ngày.

Có nhiều đỉnh cao — đáng chú ý nhất vào:

Ngày 04/12 (~4000 đơn)

Ngày 12/12 (~3800 đơn)

Ngày 25/12 (~3900 đơn)

Ngày 31/12 (~4300 đơn - cao nhất)

**2. Tác động theo mùa vụ:**

Sự gia tăng rõ rệt vào các mốc cận lễ lớn như Noel (25/12) và Giao thừa (31/12) → thể hiện xu hướng mua sắm tăng vào dịp lễ.

Đây là các thời điểm hệ thống logistics cần:

Tăng cường nhân sự và tài xế

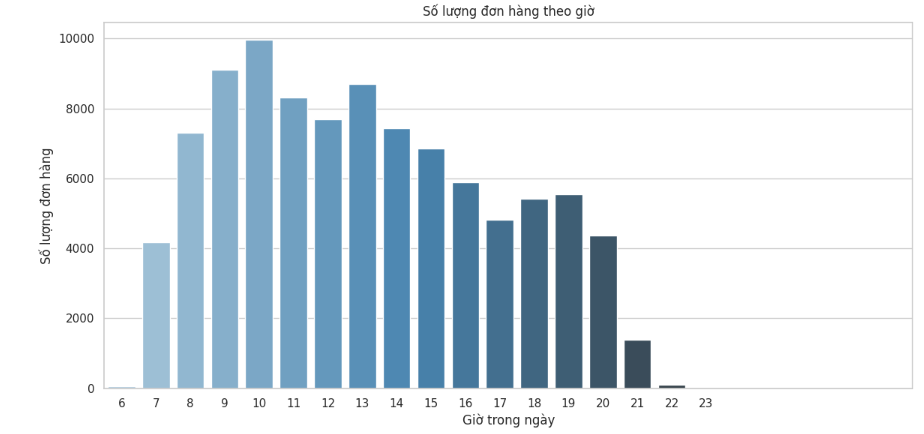
Tăng ca hoặc chia ca linh hoạt

Bổ sung xe và kho trung chuyển tạm thời

**3. Nguy cơ tắc nghẽn giao hàng:**

Các đỉnh đơn hàng cao có thể gây áp lực lên hệ thống giao nhận nếu không được chuẩn bị trước → nguy cơ trễ đơn, giao nhầm, chất lượng dịch vụ giảm.

|  |
| --- |
| # Trực quan hóa số lượng đơn hàng theo giờ  plt.figure(figsize=(12, 6))  sns.barplot(data=orders\_by\_hour, x='Giờ', y='Số lượng đơn hàng', palette='Blues\_d')  plt.title('Số lượng đơn hàng theo giờ')  plt.xlabel('Giờ trong ngày')  plt.ylabel('Số lượng đơn hàng')  plt.xticks(range(0, 24))  plt.tight\_layout()  plt.show() |



**Nhận xét :**

**1. Khung giờ cao điểm đặt hàng:**

Từ 8h đến 14h là khoảng thời gian sôi động nhất, với đỉnh vào lúc 10h (khoảng 10,000 đơn).

Giờ 9h, 11h và 13h cũng có số lượng đơn hàng cao (8,000–9,000).

**2. Khung giờ thấp điểm:**

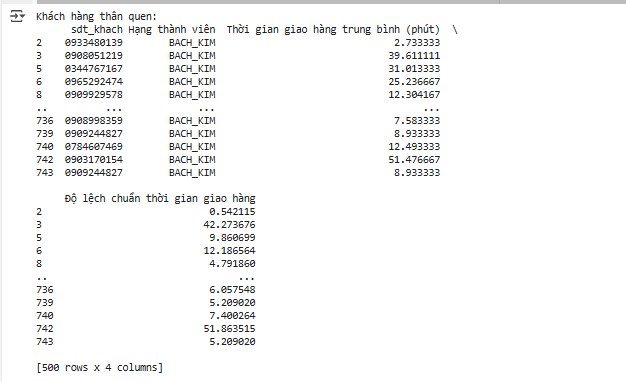
Sau 20h, số lượng đơn giảm mạnh, gần như bằng 0 từ 22h trở đi.

Rất ít đơn hàng được đặt ngoài giờ hành chính → tập trung hoạt động giao nhận vào ban ngày là hợp lý

#### **2.1.3. Phân tích theo khách hàng→ khách hàng thân quen, đề xuất khuyến mãi**

**- Thuộc tính liên quan**: sdt\_khach, Thời gian hoàn thành đơn.  
- **Phân tích**: Phân tích thời gian giao hàng và tần suất giao hàng cho từng khách hàng, đặc biệt là các khách hàng có nhu cầu giao hàng thường xuyên.

|  |
| --- |
| # lọc sdt\_khach có hạng Bach\_Kim  df\_bach\_kim = df[df['Hạng thành viên'] == 'BACH\_KIM']  # Tính thời gian giao hàng trung bình cho mỗi khách hàng  avg\_delivery\_time = df.groupby('sdt\_khach')['Thời gian giao hàng'].mean().reset\_index()  avg\_delivery\_time.columns = ['sdt\_khach', 'Thời gian giao hàng trung bình (phút)']  # Tính độ lệch chuẩn của thời gian giao hàng để đánh giá tính ổn định  std\_delivery\_time = df.groupby('sdt\_khach')['Thời gian giao hàng'].std().reset\_index()  std\_delivery\_time.columns = ['sdt\_khach', 'Độ lệch chuẩn thời gian giao hàng']  # 5. Kết hợp tần suất giao hàng, thời gian giao hàng trung bình và độ lệch chuẩn  customer\_analysis = pd.merge(df\_bach\_kim, avg\_delivery\_time, on='sdt\_khach')  customer\_analysis = pd.merge(customer\_analysis, std\_delivery\_time, on='sdt\_khach')  # Lọc ra khách hàng thân quen (tần suất giao hàng > 5 và độ lệch chuẩn thấp)  frequent\_customers = customer\_analysis[ (customer\_analysis['Độ lệch chuẩn thời gian giao hàng'] < customer\_analysis['Độ lệch chuẩn thời gian giao hàng'].quantile(0.75))]  # Hiển thị các khách hàng thân quen  print("Khách hàng thân quen:")  print(frequent\_customers[['sdt\_khach', 'Hạng thành viên', 'Thời gian giao hàng trung bình (phút)', 'Độ lệch chuẩn thời gian giao hàng']]) |



|  |
| --- |
| # Vẽ biểu đồ phân phối thời gian giao hàng trung bình của khách hàng thân quen  plt.figure(figsize=(12, 6))  sns.histplot(frequent\_customers['Thời gian giao hàng trung bình (phút)'], bins=50, kde=True, color='green')  plt.title('Phân phối thời gian giao hàng trung bình của khách hàng thân quen')  plt.xlabel('Thời gian giao hàng trung bình (phút)')  plt.ylabel('Số lượng khách hàng')  plt.show() |



**1. Nhận xét**

Đa số khách hàng thân quen có thời gian giao hàng trung bình từ 10 đến 50 phút, cho thấy hệ thống giao hàng đang vận hành hiệu quả với độ trễ thấp.

Việc giữ thời gian giao hàng ngắn là yếu tố then chốt giúp nâng cao mức độ hài lòng và tỷ lệ khách hàng quay lại.

Đỉnh phân phối rơi vào khoảng 20–30 phút, cho thấy đội ngũ giao hàng đang hoạt động tốt nhất ở các khu vực gần trung tâm kho/supermarket.

**2. Sự xuất hiện của các đơn hàng giao chậm (outliers) là vấn đề logistics cần xử lý**

Một số đơn hàng có thời gian giao lên đến hơn 100, thậm chí 400 phút, điều này không thể chấp nhận được trong hệ thống logistics tối ưu.

Có thể nguyên nhân là:

Giao sai thời điểm (giờ cao điểm, thời tiết xấu)

Tài xế bị điều phối quá xa

Sai sót thông tin địa chỉ hoặc khách không nhận hàng

#### **2.1.4. Phân tích hiệu quả của tài xế giao hàng**

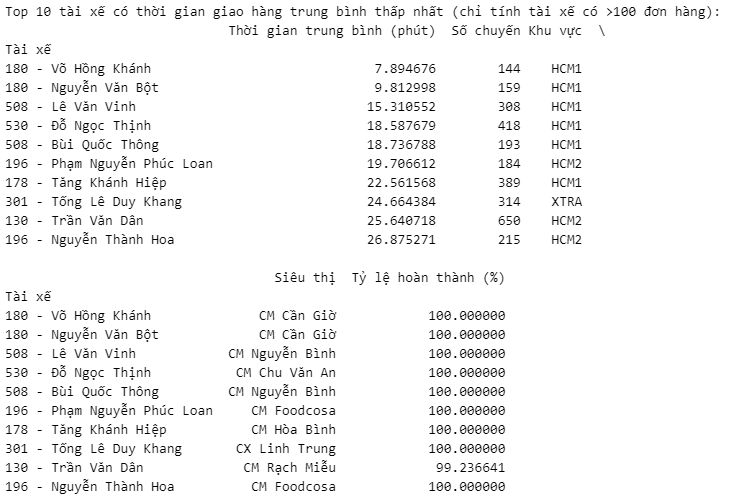
**- Thuộc tính liên quan:** Thời gian hoàn thành đơn, Thời điểm tạo đơn, Tài xế, trạng\_thái\_đơn

**- Phân tích:** Tính toán thời gian giao hàng và tỉ lệ hoàn thành đơn của tài xế. Kiểm tra xem các tài xế có hiệu suất giao hàng như thế nào.

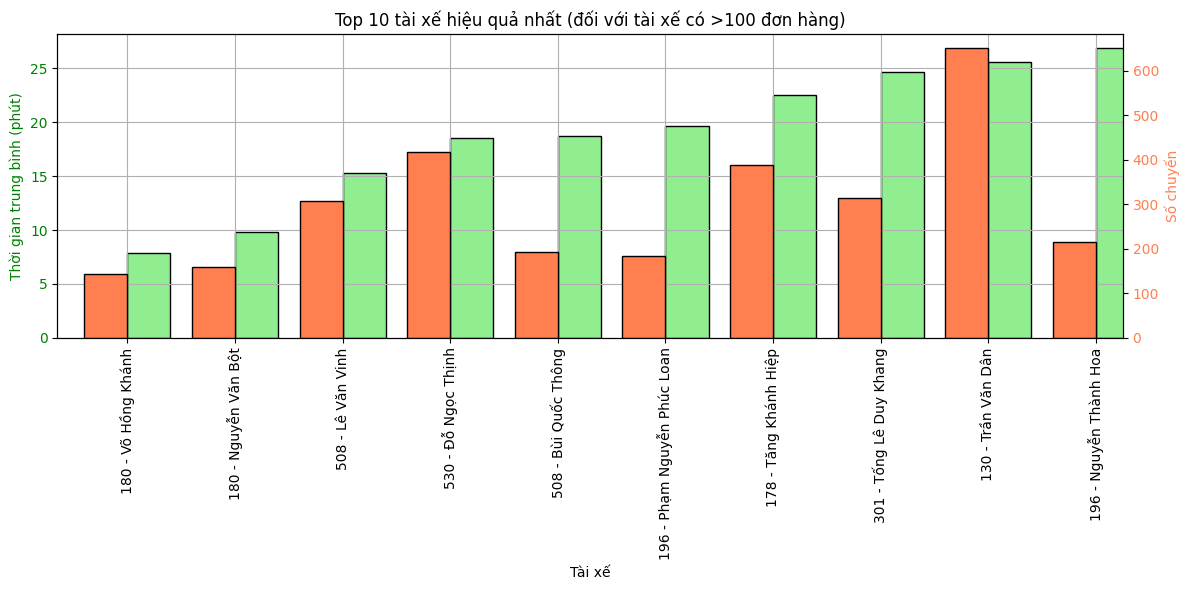
|  |
| --- |
| # Tính toán thời gian giao hàng (tính bằng giờ)  df\_processed['Thời gian giao hàng (phút)'] = (df\_processed['Thời gian hoàn thành đơn'] - df\_processed['Thời điểm tạo đơn']).dt.total\_seconds() / 60  # Tính tỉ lệ hoàn thành đơn theo tài xế  driver\_completion = df\_processed.groupby('Tài xế')['trạng\_thái\_đơn'].value\_counts(normalize=True).unstack().fillna(0)  driver\_completion['Tỉ lệ hoàn thành (%)'] = driver\_completion['HOÀN THÀNH'] \* 100  print("Tỉ lệ hoàn thành đơn theo tài xế (%):")  print(driver\_completion[['Tỉ lệ hoàn thành (%)']])  # Tính thời gian giao hàng trung bình theo tài xế (chỉ tính đơn hoàn thành)  completed\_orders = df\_processed[df\_processed['trạng\_thái\_đơn'] == 'HOÀN THÀNH']  avg\_delivery\_time\_by\_driver = completed\_orders.groupby('Tài xế')['Thời gian giao hàng (phút)'].mean()  print("\nThời gian giao hàng trung bình theo tài xế (phút):")  print(avg\_delivery\_time\_by\_driver) |



|  |
| --- |
| # Tính tổng số đơn hàng theo tài xế từ toàn bộ dữ liệu  total\_orders\_by\_driver = df\_processed.groupby('Tài xế')['Mã chuyến'].nunique()  # Chỉ giữ lại các tài xế có số đơn hàng > 100  eligible\_drivers = total\_orders\_by\_driver[total\_orders\_by\_driver > 100].index  # Lọc dữ liệu chỉ chứa các tài xế hợp lệ  completed\_orders\_filtered = completed\_orders[completed\_orders['Tài xế'].isin(eligible\_drivers)]  df\_processed\_filtered = df\_processed[df\_processed['Tài xế'].isin(eligible\_drivers)]  # Tính lại các chỉ số dựa trên tập lọc  avg\_delivery\_time\_by\_driver = completed\_orders\_filtered.groupby('Tài xế')['Thời gian giao hàng (phút)'].mean()  trips\_per\_driver = completed\_orders\_filtered.groupby('Tài xế')['Mã chuyến'].nunique()  area\_by\_driver = completed\_orders\_filtered.groupby('Tài xế')['Khu vực'].agg(lambda x: x.mode()[0] if not x.mode().empty else 'Không xác định')  supermarket\_by\_driver = completed\_orders\_filtered.groupby('Tài xế')['Siêu thị'].agg(lambda x: x.mode()[0] if not x.mode().empty else 'Không xác định')  total\_orders\_by\_driver\_filtered = df\_processed\_filtered.groupby('Tài xế')['Mã chuyến'].nunique()  completed\_orders\_by\_driver = completed\_orders\_filtered.groupby('Tài xế')['Mã chuyến'].nunique()  completion\_rate\_by\_driver = (completed\_orders\_by\_driver / total\_orders\_by\_driver\_filtered \* 100).fillna(0)  # Kết hợp thành bảng thống kê  driver\_stats = pd.DataFrame({  'Thời gian trung bình (phút)': avg\_delivery\_time\_by\_driver,  'Số chuyến': trips\_per\_driver,  'Khu vực': area\_by\_driver,  'Siêu thị': supermarket\_by\_driver,  'Tỷ lệ hoàn thành (%)': completion\_rate\_by\_driver  })  # Lấy top 10 tài xế hiệu quả nhất  top\_10\_efficient\_drivers = driver\_stats.sort\_values(by='Thời gian trung bình (phút)').head(10)  # Hiển thị kết quả  print("Top 10 tài xế có thời gian giao hàng trung bình thấp nhất (chỉ tính tài xế có >100 đơn hàng):")  print(top\_10\_efficient\_drivers) |



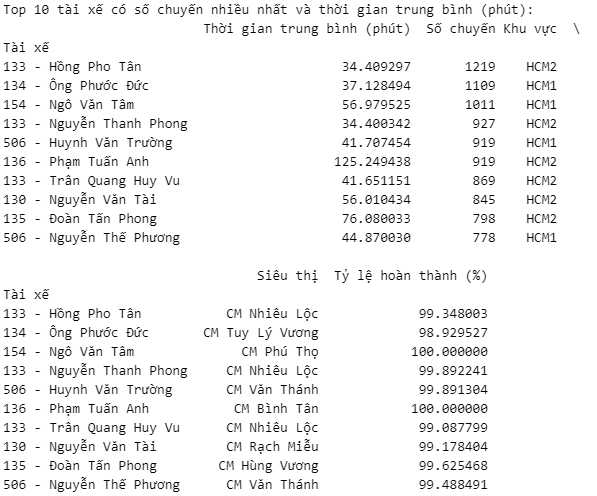
|  |
| --- |
| # Vẽ biểu đồ cột cho top 10 tài xế hiệu quả nhất (thời gian và số chuyến)  fig, ax1 = plt.subplots(figsize=(12, 6))  ax2 = ax1.twinx()  # Vẽ cột thời gian trung bình  top\_10\_efficient\_drivers['Thời gian trung bình (phút)'].plot(kind='bar', color='lightgreen', edgecolor='black', ax=ax1, position=0, width=0.4)  # Vẽ cột số chuyến  top\_10\_efficient\_drivers['Số chuyến'].plot(kind='bar', color='coral', edgecolor='black', ax=ax2, position=1, width=0.4)  ax1.set\_title('Top 10 tài xế hiệu quả nhất (đối với tài xế có >100 đơn hàng')  ax1.set\_xlabel('Tài xế')  ax1.set\_ylabel('Thời gian trung bình (phút)', color='green')  ax2.set\_ylabel('Số chuyến', color='coral')  ax1.tick\_params(axis='y', labelcolor='green')  ax2.tick\_params(axis='y', labelcolor='coral')  plt.xticks(rotation=45, ha='right')  ax1.grid(True)  plt.tight\_layout()  plt.show() |



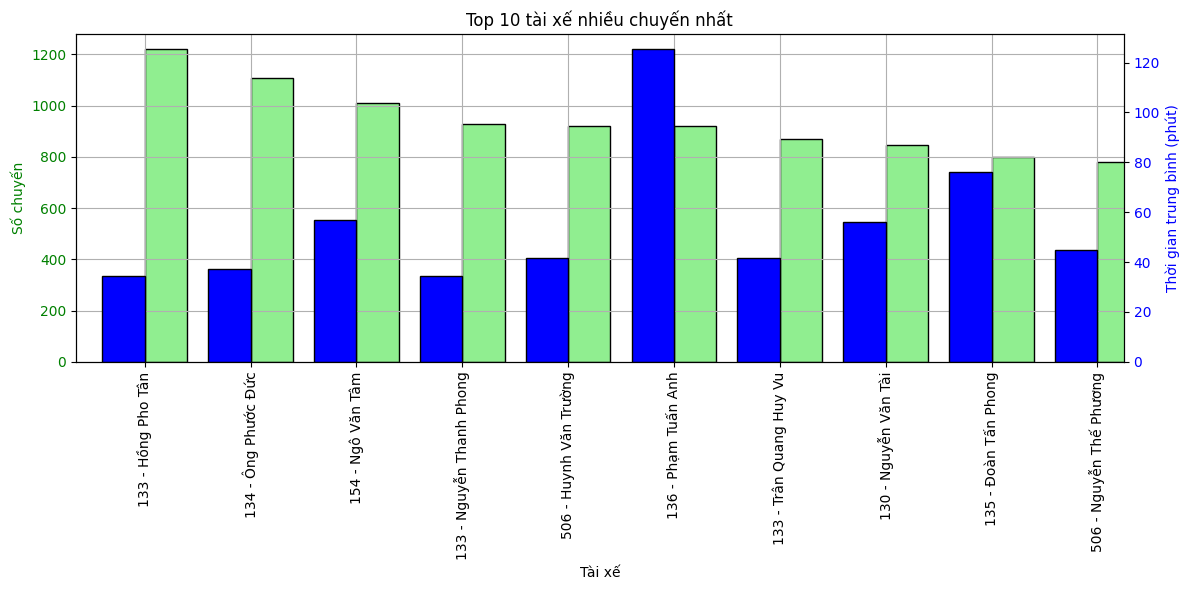
**Nhận xét:**

* Biểu đồ trình bày danh sách 10 tài xế có hiệu suất cao nhất (đối với tài xế có hơn 100 đơn hàng), dựa trên hai tiêu chí: thời gian giao hàng trung bình (phút) và số lượng chuyến giao hàng. Mỗi tài xế được biểu diễn với hai cột: cột màu xanh lá thể hiện thời gian trung bình giao hàng, trong khi cột màu cam thể hiện tổng số chuyến đã thực hiện.
* Nhìn chung, nhóm tài xế này thể hiện hiệu quả tốt ở ít nhất một khía cạnh: hoặc giao hàng nhanh, hoặc đảm nhận số lượng lớn đơn hàng. Một số tài xế nổi bật với thời gian giao hàng trung bình rất thấp (dưới 10 phút), cho thấy tốc độ xử lý đơn hàng nhanh chóng và khả năng tối ưu hóa lộ trình tốt. Ngược lại, có những tài xế đảm nhận số chuyến rất cao, thể hiện tần suất hoạt động dày đặc và vai trò quan trọng trong hệ thống.
* Biểu đồ cũng cho thấy sự đa dạng về phong cách làm việc: có người thiên về tốc độ, có người đảm nhận khối lượng lớn, và cũng có những người cân bằng giữa hai yếu tố này. Sự đa dạng này là tích cực, vì hệ thống cần cả tốc độ lẫn sự bền bỉ để đáp ứng nhiều loại nhu cầu vận hành khác nhau.
* Tổng thể, biểu đồ cung cấp cái nhìn trực quan và rõ ràng về hiệu quả của tài xế, là cơ sở tốt để đánh giá, phân công công việc hợp lý và xây dựng các chính sách khen thưởng phù hợp.

|  |
| --- |
| # Lấy top 10 tài xế có số chuyến nhiều nhất  top\_10\_most\_trips\_drivers = driver\_stats.sort\_values(by='Số chuyến', ascending=False).head(10)  # In kết quả top 10 tài xế có số chuyến nhiều nhất  print("\nTop 10 tài xế có số chuyến nhiều nhất và thời gian trung bình (phút):")  print(top\_10\_most\_trips\_drivers) |



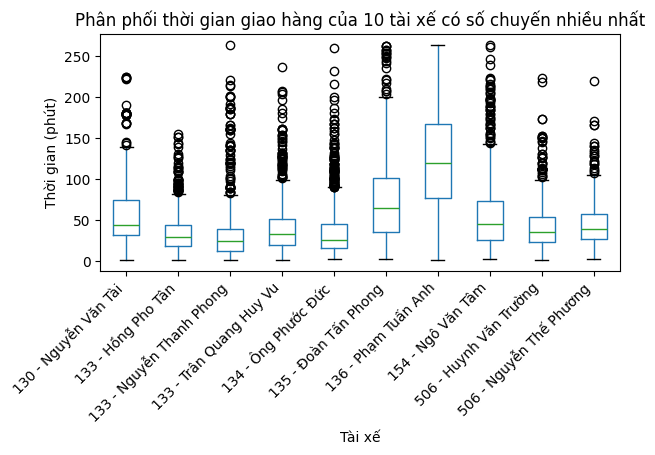
|  |
| --- |
| # 8. Vẽ biểu đồ cột cho top 10 tài xế có số chuyến nhiều nhất (số chuyến và thời gian)  fig, ax1 = plt.subplots(figsize=(12, 6))  ax2 = ax1.twinx()  # Vẽ cột số chuyến  top\_10\_most\_trips\_drivers['Số chuyến'].plot(kind='bar', color='lightgreen', edgecolor='black', ax=ax1, position=0, width=0.4)  # Vẽ cột thời gian trung bình  top\_10\_most\_trips\_drivers['Thời gian trung bình (phút)'].plot(kind='bar', color='blue', edgecolor='black', ax=ax2, position=1, width=0.4)  ax1.set\_title('Top 10 tài xế nhiều chuyến nhất')  ax1.set\_xlabel('Tài xế')  ax1.set\_ylabel('Số chuyến', color='green')  ax2.set\_ylabel('Thời gian trung bình (phút)', color='blue')  ax1.tick\_params(axis='y', labelcolor='green')  ax2.tick\_params(axis='y', labelcolor='blue')  plt.xticks(rotation=45, ha='right')  ax1.grid(True)  plt.tight\_layout()  plt.show() |



Nhận xét:

* Nhìn chung, các tài xế trong danh sách đều có tần suất làm việc cao, với số chuyến dao động từ khoảng 800 đến hơn 1200 chuyến. Đây là những cá nhân có mức độ đóng góp lớn cho hệ thống vận hành.
* Tuy nhiên, thời gian giao hàng trung bình giữa các tài xế có sự khác biệt rõ rệt. Một số tài xế như Phạm Tuấn Anh (136) và Đoàn Tấn Phong (135) có thời gian trung bình khá cao (lên tới 120 phút và 70 phút), trong khi các tài xế khác như Nguyễn Thanh Phong (133) hoặc Hồng Phò Tân (133) duy trì được thời gian trung bình thấp hơn (30–40 phút).
* Sự chênh lệch này có thể phản ánh khác biệt về tuyến đường, địa bàn hoạt động, hoặc điều kiện giao hàng cụ thể. Điều quan trọng là những tài xế duy trì số chuyến cao kèm thời gian trung bình hợp lý chính là những cá nhân thể hiện sự hiệu quả vượt trội cả về khối lượng công việc lẫn tốc độ xử lý.

|  |
| --- |
| # 9. Vẽ boxplot cho top 10 tài xế có số chuyến nhiều nhất  top\_10\_most\_trips\_driver\_names = top\_10\_most\_trips\_drivers.index  top\_10\_most\_trips\_orders = completed\_orders[completed\_orders['Tài xế'].isin(top\_10\_most\_trips\_driver\_names)]  plt.figure(figsize=(20, 6))  top\_10\_most\_trips\_orders.boxplot(column='Thời gian giao hàng (phút)', by='Tài xế', grid=False)  plt.title('Phân phối thời gian giao hàng của 10 tài xế có số chuyến nhiều nhất')  plt.suptitle('') # Xóa tiêu đề mặc định của boxplot  plt.xlabel('Tài xế')  plt.ylabel('Thời gian (phút)')  plt.xticks(rotation=45, ha='right')  plt.tight\_layout()  plt.show() |



**Nhận xét:**

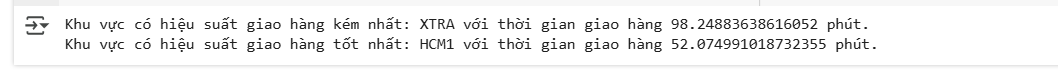
* Tập hợp các biểu đồ đã phân tích cho thấy bức tranh toàn diện về hiệu suất của các tài xế dựa trên hai tiêu chí chính: số lượng chuyến giao hàng và thời gian giao hàng trung bình. Nhóm tài xế hiệu quả nhất có thể được nhìn nhận theo hai hướng: hoặc có thời gian giao hàng rất nhanh, hoặc đảm nhận số lượng đơn lớn, hoặc cân bằng tốt cả hai yếu tố này.
* Trong nhóm tài xế giao hàng nhanh nhất, có nhiều người duy trì được thời gian trung bình dưới 10 phút – đây là những cá nhân có khả năng xử lý đơn hàng nhanh chóng, tối ưu hóa lộ trình và hoạt động ổn định. Một số tài xế lại có thời gian trung bình cao hơn đáng kể, một phần có thể do đặc thù tuyến đường, điều kiện làm việc, hoặc khối lượng đơn hàng lớn dẫn đến độ trễ tích lũy.
* Đáng chú ý, biểu đồ cho thấy sự khác biệt lớn giữa các cá nhân. Một số tài xế như Nguyễn Thanh Phong, Trần Quang Huy Vũ hay Hồng Phò Tân có phân phối hẹp, ít outlier và thời gian trung bình thấp, thể hiện sự ổn định rất cao trong hiệu suất làm việc. Trong khi đó, các tài xế như Phạm Tuấn Anh hoặc Ngô Văn Tâm có median cao và rất nhiều outlier – cho thấy họ thường xuyên gặp các chuyến kéo dài bất thường, có thể do các nguyên nhân như tuyến đường xa, điều kiện khó khăn, hoặc vấn đề vận hành.

### **2.2. Phân tích theo phí và khu vực**

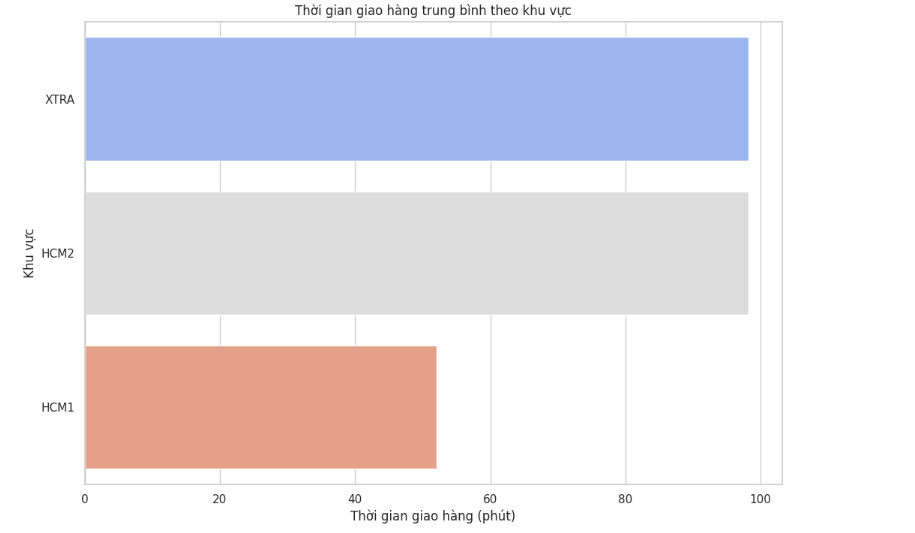
#### **2.2.1.Phân tích hiệu suất của các khu vực**

- **Thuộc tính liên quan:** Khu vực, Thời gian hoàn thành đơn,...  
- **Phân tích**: Tính toán thời gian giao hàng trung bình cho mỗi khu vực và xác định các khu vực có hiệu suất giao hàng tốt nhất và kém nhất

|  |
| --- |
| #df['Thời gian giao hàng'] = (df['Thời gian hoàn thành đơn'] - df['Thời điểm tài xế lấy hàng']).dt.total\_seconds() / 60  #  Tính toán thời gian giao hàng trung bình cho mỗi khu vực  avg\_delivery\_time\_by\_region = df.groupby('Khu vực')['Thời gian giao hàng'].mean().reset\_index()  # Sắp xếp khu vực theo thời gian giao hàng trung bình  avg\_delivery\_time\_by\_region = avg\_delivery\_time\_by\_region.sort\_values(by='Thời gian giao hàng', ascending=False)  #  Các khu vực có hiệu suất giao hàng tốt nhất và kém nhất  best\_region = avg\_delivery\_time\_by\_region.iloc[0]  # Khu vực có thời gian giao hàng trung bình thấp nhất  worst\_region = avg\_delivery\_time\_by\_region.iloc[-1]  # Khu vực có thời gian giao hàng trung bình cao nhất  print(f"Khu vực có hiệu suất giao hàng kém nhất: {best\_region['Khu vực']} với thời gian giao hàng {best\_region['Thời gian giao hàng']} phút.")  print(f"Khu vực có hiệu suất giao hàng tốt nhất: {worst\_region['Khu vực']} với thời gian giao hàng {worst\_region['Thời gian giao hàng']} phút.") |



|  |
| --- |
| # Vẽ biểu đồ phân phối thời gian giao hàng cho từng khu vực  plt.figure(figsize=(12, 8))  sns.barplot(x='Thời gian giao hàng', y='Khu vực', data=avg\_delivery\_time\_by\_region, palette='coolwarm')  plt.title('Thời gian giao hàng trung bình theo khu vực')  plt.xlabel('Thời gian giao hàng (phút)')  plt.ylabel('Khu vực')  plt.show() |



**1.Chênh lệch thời gian giao hàng giữa các khu vực:**

Khu vực HCM1 có thời gian giao hàng trung bình thấp nhất (~50 phút) → cho thấy hệ thống vận chuyển tại đây có thể đang hoạt động hiệu quả, mạng lưới giao hàng tối ưu hơn.

HCM2 và đặc biệt là XTRA có thời gian giao hàng cao hơn đáng kể (gần 100 phút) → có thể đang gặp vấn đề về khoảng cách, lưu lượng đơn hàng hoặc thiếu hụt nguồn lực vận chuyển.

**2.Tác động đến trải nghiệm khách hàng:**

Khu vực XTRA với thời gian giao hàng cao nhất có nguy cơ ảnh hưởng đến mức độ hài lòng của khách hàng.

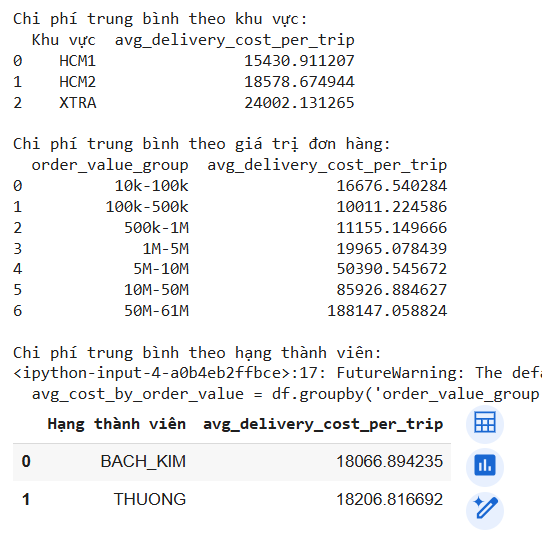
Cần kiểm tra lại quy trình giao hàng tại XTRA: có thể thiếu kho trung chuyển, tuyến đường dài hoặc tắc nghẽn.

#### **2.2.2. Phân tích chi phí giao hàng**

**Thuộc tính sử dụng:** Phí chuyến cho tài xế, Tổng phí trả tài xế, Hạng thành viên, Giá trị đơn hàng, Khu vực.

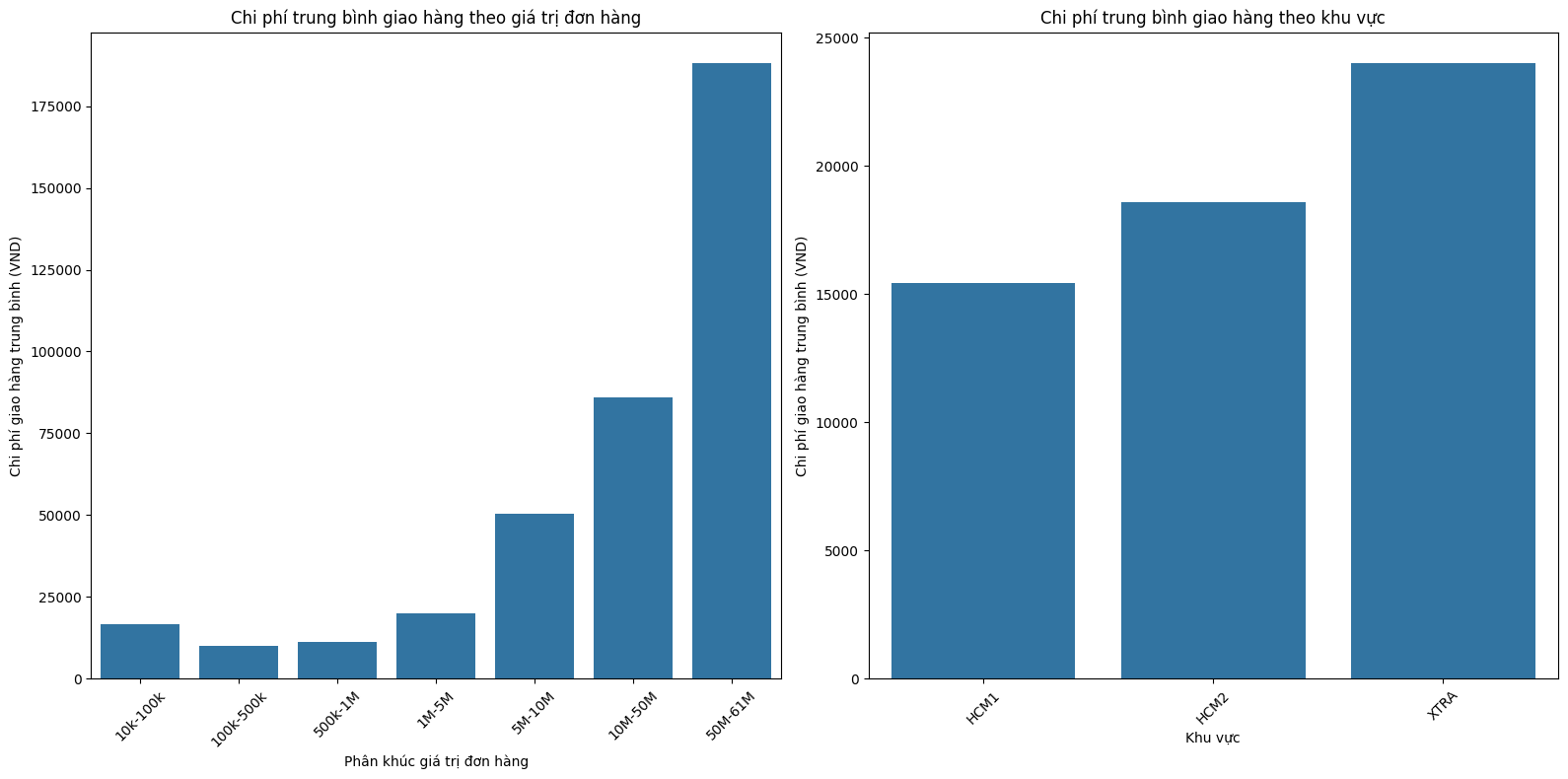
**Phân tích:** Tính toán chi phí trung bình cho mỗi chuyến giao hàng và phân tích các yếu tố như khu vực, giá trị đơn hàng và hạng thành viên ảnh hưởng đến chi phí giao hàng.

|  |
| --- |
| # Tính chi phí trung bình cho mỗi chuyến giao hàng (Tổng phí trả tài xế / Số chuyến)  df['avg\_delivery\_cost\_per\_trip'] = df['Tổng phí trả tài xế'] / df['Số chuyến']  # Phân tích chi phí giao hàng theo khu vực  avg\_cost\_by\_area = df.groupby('Khu vực')['avg\_delivery\_cost\_per\_trip'].mean().reset\_index()  # Phân tích chi phí giao hàng theo giá trị đơn hàng  # Phân nhóm giá trị đơn hàng thành các phân khúc  bins = [10000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 61000000]  labels = ['10k-100k', '100k-500k', '500k-1M', '1M-5M', '5M-10M', '10M-50M', '50M-61M']  # Tạo cột phân nhóm giá trị đơn hàng  df['order\_value\_group'] = pd.cut(df['Giá trị đơn hàng'], bins=bins, labels=labels, right=True)  # Tính chi phí giao hàng trung bình theo phân khúc giá trị đơn hàng  avg\_cost\_by\_order\_value = df.groupby('order\_value\_group')['avg\_delivery\_cost\_per\_trip'].mean().reset\_index()  # Phân tích chi phí giao hàng theo hạng thành viên  avg\_cost\_by\_member\_level = df.groupby('Hạng thành viên')['avg\_delivery\_cost\_per\_trip'].mean().reset\_index()  print("Chi phí trung bình theo khu vực:")  print(avg\_cost\_by\_area)  print("\nChi phí trung bình theo giá trị đơn hàng:")  print(avg\_cost\_by\_order\_value)  print("\nChi phí trung bình theo hạng thành viên:")  avg\_cost\_by\_member\_level |



|  |
| --- |
| plt.figure(figsize=(8, 5))  sns.barplot(x='order\_value\_group', y='avg\_delivery\_cost\_per\_trip', data=avg\_cost\_by\_order\_value)  # Gán tiêu đề và nhãn  plt.title('Chi phí trung bình giao hàng theo giá trị đơn hàng')  plt.xlabel('Phân khúc giá trị đơn hàng')  plt.ylabel('Chi phí giao hàng trung bình (VND)')  plt.xticks(rotation=45) # Xoay nhãn trục x  # Căn chỉnh layout và hiển thị  plt.tight\_layout()  plt.show() |

Output:



**Nhận xét phần phân tích chi phí giao hàng tháng 12/2022**

**1. Chi phí giao hàng theo giá trị đơn hàng**

* **Giá trị đơn hàng càng cao thì chi phí giao hàng trung bình cũng tăng rõ rệt**.
* Với đơn hàng nhỏ (10k–1M VND), chi phí chỉ từ khoảng **10.000–20.000 VND**.
* Từ đơn hàng **5 triệu VND trở lên**, chi phí tăng nhanh:
  + **5M–10M**: khoảng **50.000 VND**
  + **10M–50M**: gần **85.000 VND**
  + **Trên 50M**: lên tới gần **190.000 VND**
* **Giải thích**: Các đơn hàng giá trị cao thường đi kèm yêu cầu giao hàng đặc biệt (kích thước lớn, xe chuyên dụng, giao hẹn giờ, giao xa, cần đảm bảo an toàn...), dẫn đến phí tài xế tăng.  
  **Khuyến nghị**:
* Tối ưu chính sách vận chuyển cho đơn hàng lớn.
* Đàm phán lại mức phí theo khối lượng và giá trị thực tế.
* Phân tích khả năng gộp đơn hàng lớn để tiết kiệm chi phí.

**2. Chi phí giao hàng theo khu vực**

* **XTRA** là khu vực có chi phí giao hàng cao nhất (~24.000 VND).
* **HCM2** cao hơn **HCM1**, cho thấy có thể là vùng ngoại ô hoặc mật độ tài xế thấp.
* **HCM1** thấp nhất (~15.000 VND), có thể do gần kho, nhiều tài xế, hoặc tuyến đường thuận lợi.

**Khuyến nghị**:

* Tối ưu phân bổ tài xế tại khu vực XTRA.
* Xem xét điều chỉnh chính sách giá giao hàng theo vùng địa lý.

3. Chi phí giao hàng theo hạng thành viên

* Hạng thành viên **BACH\_KIM** có chi phí giao hàng trung bình thấp hơn so với **THUONG**. Có thể khách hàng thân thuộc và mức hạng cao có nhiều ưu đãi hơn.

**Tóm lại:**

Chi phí giao hàng bị ảnh hưởng mạnh bởi:

* **Giá trị đơn hàng** (tăng mạnh theo quy mô đơn hàng).
* **Khu vực giao hàng** (chênh lệch đáng kể giữa các vùng).

Đây là cơ sở để cải thiện:

* Chính sách phí giao hàng,
* Phân bổ nguồn lực tài xế,
* Và tối ưu hiệu quả logistics cho doanh nghiệp.

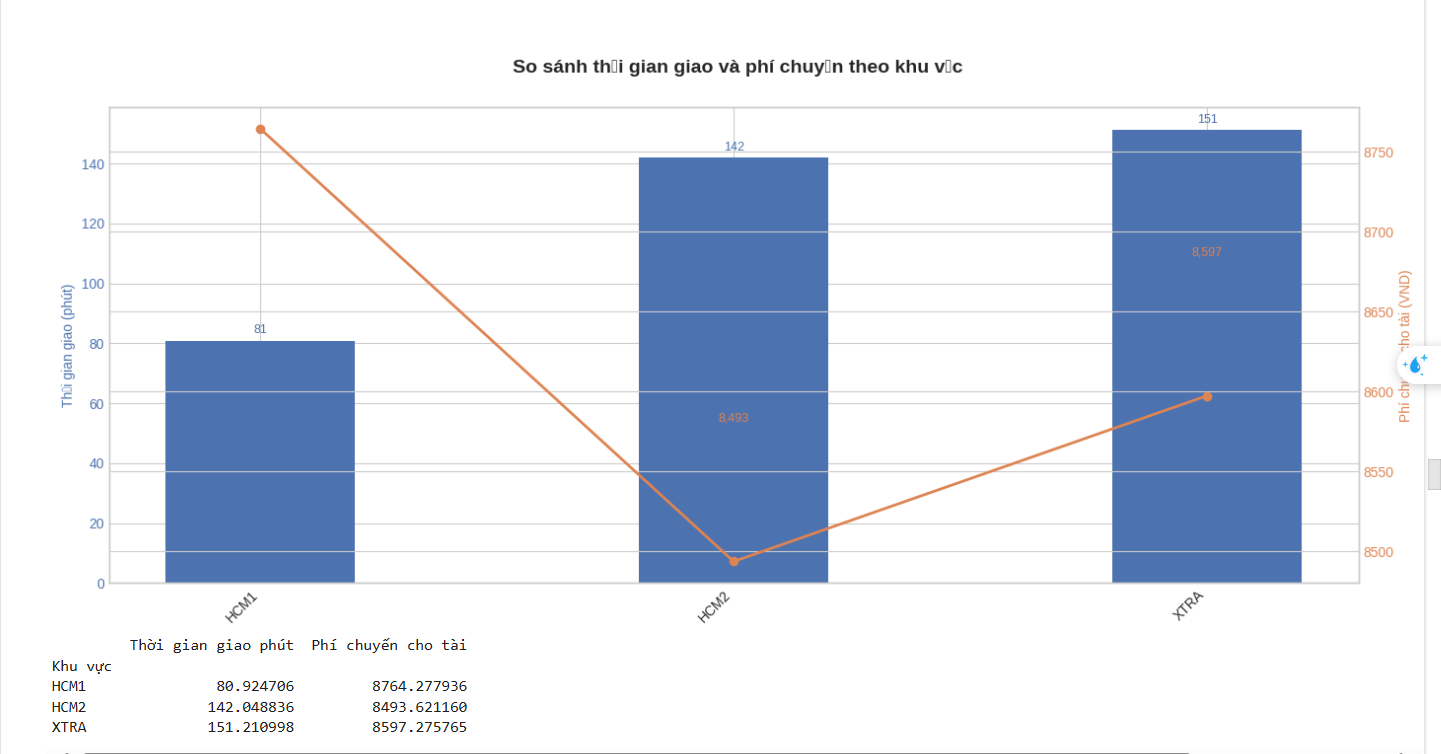
#### **2.2.3. Phân tích mối quan hệ giữa các yếu tố logistics**

**- Thuộc tính liên quan**: Khu vực, Phí chuyến cho tài xế, Thời gian hoàn thành đơn.

- **Phân tích**: Tìm ra sự tương quan giữa các yếu tố như khu vực, chi phí và thời gian giao hàng để tối ưu hóa các quy trình trong hệ thống logistics.

##### **2.2.3.1. Tính toán các thuộc tính thời gian liên quan**

|  |
| --- |
| # Tính toán thời gian giao hàng (phút)  df['Thời gian giao phút'] = (df['Thời gian hoàn thành đơn'] - df['Thời điểm tài xế nhận đơn']).dt.total\_seconds() / 60  # Chuyển phí chuyến cho tài xế từ chuỗi có dấu phẩy thành số thực  df['Phí chuyến cho tài'] = df['Phí chuyến cho tài'].apply(lambda x: float(str(x).replace(",", "")) if pd.notnull(x) else 0)  # Gom nhóm dữ liệu theo khu vực và tính giá trị trung bình  grouped = df.groupby('Khu vực')[['Thời gian giao phút', 'Phí chuyến cho tài']].mean()  # Dữ liệu  labels = grouped.index  times = grouped["Thời gian giao phút"]  costs = grouped["Phí chuyến cho tài"]  # Vẽ biểu đồ  x = np.arange(len(labels)) # vị trí cột  fig, ax1 = plt.subplots(figsize=(14, 6))  # Vẽ cột thời gian giao  bar = ax1.bar(x, times, color="#4c72b0", width=0.4, label="Thời gian giao (phút)")  ax1.set\_ylabel("Thời gian giao (phút)", color="#4c72b0")  ax1.tick\_params(axis='y', labelcolor="#4c72b0")  ax1.set\_xticks(x)  ax1.set\_xticklabels(labels, rotation=45, ha="right")  # Ghi nhãn trên cột  for rect in bar:  height = rect.get\_height()  ax1.annotate(f'{height:.0f}',  xy=(rect.get\_x() + rect.get\_width() / 2, height),  xytext=(0, 3), # khoảng cách trên đầu cột  textcoords="offset points",  ha='center', va='bottom', fontsize=9, color="#4c72b0")  # Trục phụ cho phí chuyến  ax2 = ax1.twinx()  line = ax2.plot(x, costs, color="#dd8452", marker='o', linewidth=2, label="Phí chuyến (VND)")  ax2.set\_ylabel("Phí chuyến cho tài (VND)", color="#dd8452")  ax2.tick\_params(axis='y', labelcolor="#dd8452")  # Ghi nhãn trên đường  for i, v in enumerate(costs):  ax2.text(x[i], v + max(costs)\*0.01, f"{int(v):,}", color="#dd8452", ha='center', fontsize=9)  # Tiêu đề và chú thích  fig.suptitle("So sánh thời gian giao và phí chuyến theo khu vực", fontsize=14, fontweight='bold')  fig.tight\_layout()  plt.subplots\_adjust(top=0.9)  # Hiển thị biểu đồ  plt.show()  # Kiểm tra kết quả nhóm  print(grouped.head()) |



**Nhận xét**

**Thời gian giao hàng:**

HCM1 có thời gian giao hàng trung bình thấp nhất, chỉ khoảng 81 phút, cho thấy khu vực này có hiệu quả giao hàng cao.

HCM2 có thời gian giao hàng trung bình cao hơn, 142 phút, cao hơn HCM1 rất nhiều.

XTRA có thời gian giao hàng cao nhất với 151 phút, cho thấy có thể có vấn đề về khoảng cách hoặc quá trình giao hàng không tối ưu ở khu vực này.

**Phí chuyến cho tài xế:**

HCM1 có phí chuyến cho tài xế thấp nhất, khoảng 8,764 VND, cho thấy khu vực này có mức phí thấp mặc dù thời gian giao hàng nhanh.

HCM2 có phí 8,493 VND, thấp hơn một chút so với HCM1, mặc dù thời gian giao hàng lâu hơn. Điều này có thể cho thấy hiệu quả chi phí trong khu vực này chưa cao.

XTRA có mức phí cao nhất, 8,597 VND, mặc dù thời gian giao hàng cũng cao, cho thấy phí không thay đổi đáng kể với thời gian giao.

**Mối quan hệ giữa thời gian giao và phí:**

Mặc dù XTRA có thời gian giao hàng lâu nhất, nhưng mức phí không có sự thay đổi mạnh mẽ so với các khu vực khác, điều này có thể cho thấy mức phí không hoàn toàn phụ thuộc vào thời gian giao hàng.

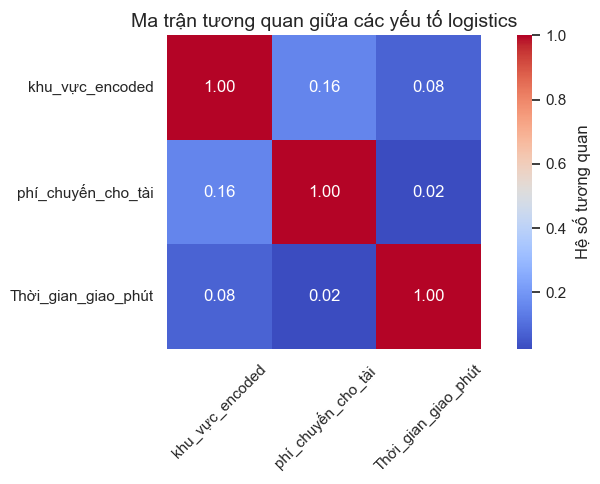
HCM1 có mức phí thấp và thời gian giao ngắn, cho thấy khu vực này có thể hoạt động hiệu quả về chi phí.

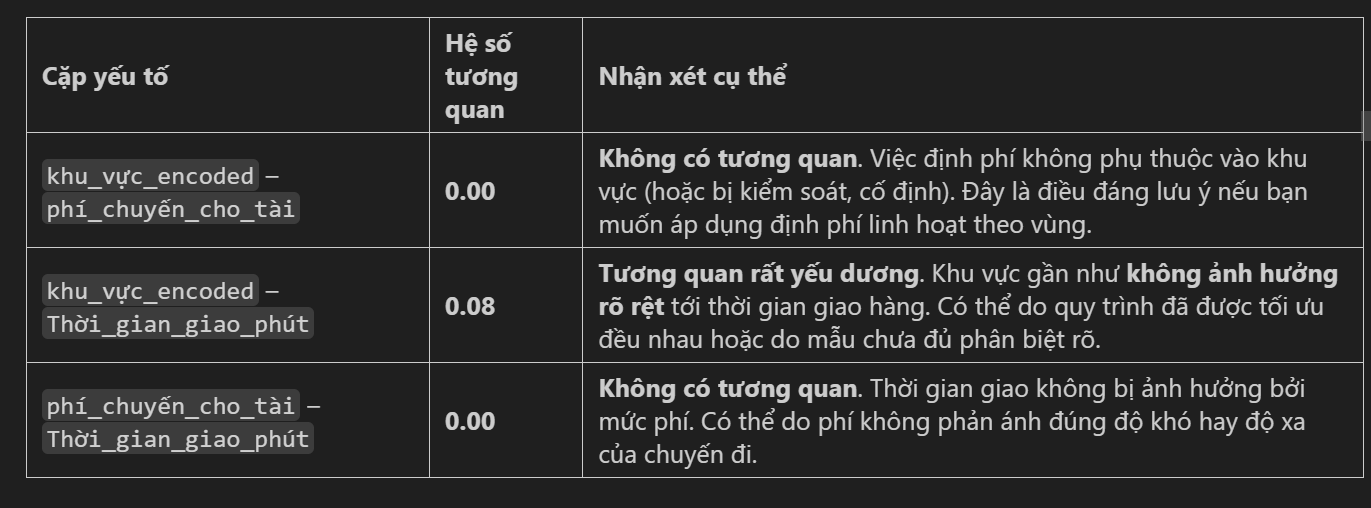
**Đề xuất:** Tối ưu hóa thời gian giao ở HCM2 và XTRA để giảm thiểu chi phí và nâng cao hiệu quả.

XTRA có thể cần cải thiện quy trình giao hàng để giảm thời gian giao hàng và tối ưu chi phí hơn nữa.

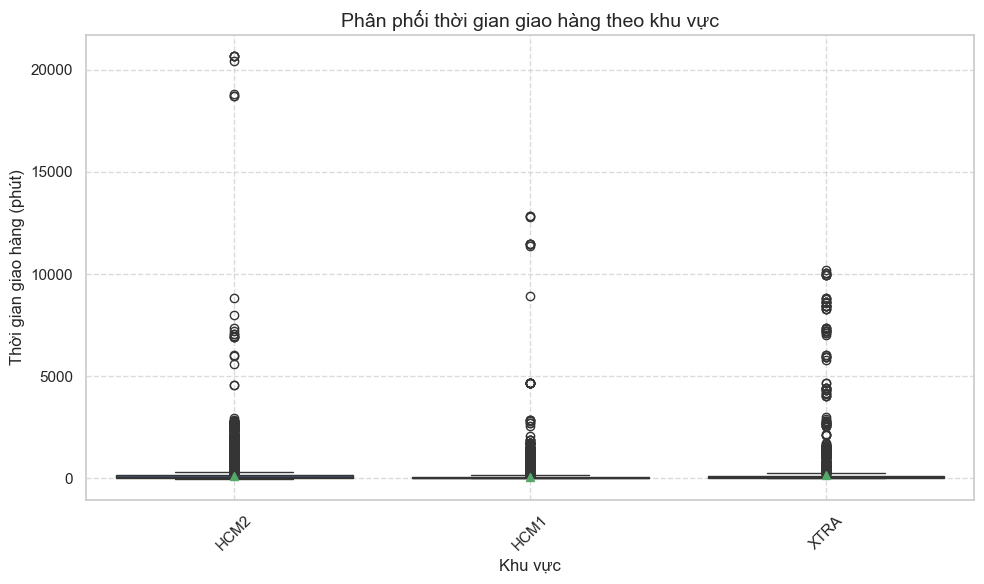
##### **2.2.3.2. Tìm ra sự tương quan giữa các yếu tố như khu vực, chi phí và thời gian giao hàng để tối ưu hóa các quy trình trong hệ thống logistics.**

|  |
| --- |
| # Bước 2: Loại bỏ giá trị thiếu  df\_corr = df[["Khu vực", "Phí chuyến cho tài", "Thời gian giao phút"]].dropna()  # Bước 3: Mã hóa khu vực (dạng chuỗi) thành số  le = LabelEncoder()  df\_corr["Khu vực encoded"] = le.fit\_transform(df\_corr["Khu vực"])  # Bước 4: Tính ma trận tương quan  corr\_matrix = df\_corr[["Khu vực encoded", "Phí chuyến cho tài", "Thời gian giao phút"]].corr()  # Bước 5: Vẽ biểu đồ heatmap thể hiện sự tương quan  plt.figure(figsize=(8, 5))  sns.heatmap(corr\_matrix, annot=True, cmap="coolwarm", fmt=".2f", square=True,  cbar\_kws={"label": "Hệ số tương quan"})  plt.title("Ma trận tương quan giữa các yếu tố logistics", fontsize=14)  plt.xticks(rotation=45)  plt.yticks(rotation=0)  plt.tight\_layout()  plt.show() |





|  |
| --- |
| # Boxplot  plt.figure(figsize=(10, 6))  sns.boxplot(x='Khu vực', y='Thời gian giao phút', data=df\_corr, palette='viridis', showmeans=True)  plt.title("Phân phối thời gian giao hàng theo khu vực", fontsize=14)  plt.ylabel("Thời gian giao hàng (phút)")  plt.xlabel("Khu vực")  plt.xticks(rotation=45)  plt.grid(True, linestyle='--', alpha=0.7)  plt.tight\_layout()  plt.show() |



**Nhận xét:**

Phân phối thời gian giao hàng:

* + HCM2: Thời gian giao hàng có giá trị trung vị (đường màu xanh lá) nằm ở mức thấp, khoảng dưới 500 phút. Khoảng cách giữa các phần tư (Q1-Q3) khá hẹp, cho thấy dữ liệu tập trung. Tuy nhiên, có một số giá trị ngoại lai (outliers) ở mức cao, lên đến khoảng 2000 phút, cho thấy sự biến động lớn trong một số trường hợp.
  + HCM1: Trung vị thời gian giao hàng cũng nằm dưới 500 phút, tương tự HCM2. Khoảng cách Q1-Q3 rộng hơn một chút so với HCM2, và có ít giá trị ngoại lai hơn, với các điểm cao nhất khoảng 1500 phút. XTRA: Trung vị thời gian giao hàng cao hơn một chút so với hai khu vực kia, khoảng 500-1000 phút. Khoảng cách Q1-Q3 rộng hơn, và có nhiều giá trị ngoại lai hơn, với một số điểm vượt 1500 phút, thậm chí gần 2000 phút.
* So sánh giữa các khu vực:
  + XTRA có xu hướng thời gian giao hàng cao hơn và biến động lớn hơn so với HCM2 và HCM1, có thể do điều kiện giao thông hoặc khoảng cách xa hơn.
  + HCM2 và HCM1 có phân phối tương đối tương đồng, với thời gian giao hàng trung bình thấp hơn và ít biến động hơn so với XTRA.
* Giá trị ngoại lai:
  + Tất cả ba khu vực đều có các giá trị ngoại lai đáng kể, đặc biệt ở HCM2 và XTRA, với một số điểm vượt quá 1500-2000 phút. Điều này có thể do các yếu tố như giao hàng đặc biệt, sự cố, hoặc lỗi dữ liệu cần kiểm tra thêm.

|  |
| --- |
| def detect\_outliers(df, column):  Q1 = df[column].quantile(0.25)  Q3 = df[column].quantile(0.75)  IQR = Q3 - Q1  lower\_bound = Q1 - 1.5 \* IQR  upper\_bound = Q3 + 1.5 \* IQR  outliers = df[(df[column] < lower\_bound) | (df[column] > upper\_bound)]  return outliers  # Tìm outliers trong thời gian giao hàng  outliers\_time = detect\_outliers(df\_corr, 'Thời gian giao phút')  print("Outliers thời gian giao hàng:")  print(outliers\_time[['Khu vực', 'Phí chuyến cho tài', 'Thời gian giao phút']])  # Gợi ý xử lý: Thêm cột 'ly\_do\_ngoai\_lai' để ghi nhận nguyên nhân (có thể thủ công hoặc dựa trên logic)  df\_corr['ly do ngoai lai'] = np.where(df\_corr['Thời gian giao phút'] > 10000,  'Giao hàng khẩn cấp hoặc lỗi hệ thống',  'Bình thường')  # Vẽ scatter plot với đường hồi quy  plt.figure(figsize=(10, 6))  sns.regplot(data=df\_corr, x="Phí chuyến cho tài", y="Thời gian giao phút",  scatter\_kws={'s': 100}, line\_kws={'color': 'red'})  sns.scatterplot(data=df\_corr, x="Phí chuyến cho tài", y="Thời gian giao phút", hue="Khu vực", palette="tab10", s=100)  plt.xscale('log')  plt.title("Mối quan hệ giữa phí và thời gian giao hàng theo khu vực", fontsize=14)  plt.xlabel("Phí chuyến cho tài (VND) - Log scale")  plt.ylabel("Thời gian giao hàng (phút)")  plt.legend(title="Khu vực")  plt.grid(True, linestyle='--', alpha=0.7)  plt.tight\_layout()  plt.show() |

**Outliers thời gian giao hàng:**

Khu vực Phí chuyến cho tài Thời gian giao phút

32 HCM2 8000.0 286.983333

47 HCM2 8000.0 307.033333

48 HCM2 8000.0 307.600000

49 HCM2 8000.0 304.250000

50 HCM2 8000.0 303.916667

... ... ... ...

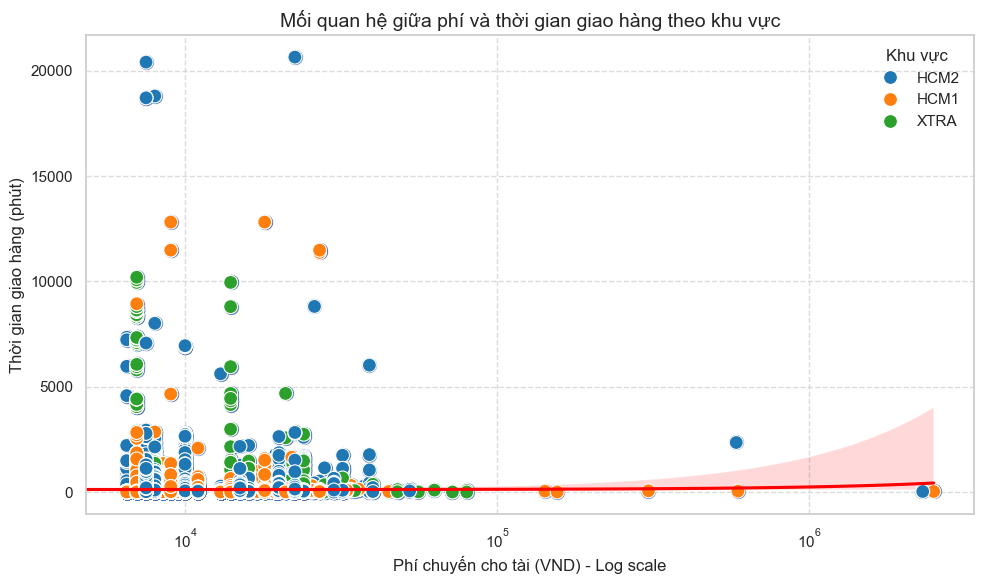
96653 HCM2 7500.0 976.950000

96657 HCM2 7500.0 1081.800000

96659 HCM2 7500.0 1116.683333

96660 HCM2 15000.0 1123.450000

96661 HCM2 7500.0 1125.283333



**Nhận xét:**

Đường hồi quy (Regression Line): Đường hồi quy cho thấy một xu hướng giảm nhẹ: khi phí chuyến cho tài (VND) tăng, thời gian giao hàng (phút) có xu hướng giảm. Tuy nhiên, độ dốc của đường rất nhỏ, cho thấy mối quan hệ giữa phí và thời gian giao hàng không mạnh. Vùng màu hồng xung quanh đường hồi quy (khoảng tin cậy) khá rộng, đặc biệt ở các mức phí cao (10⁵ đến 10⁶ VND), điều này cho thấy độ biến thiên lớn và mối quan hệ giữa hai biến không rõ ràng.

Phí chuyển giao tài và Thời gian giao hàng cho ba khu vực khác nhau: HCM2, HCM1, và XTRA. Hãy đi sâu vào từng yếu tố để có thể giải thích rõ hơn.

Các yếu tố trong biểu đồ:

1. Trục hoành (X-axis): Phí chuyển giao tài (VND)

Trục hoành biểu thị phí chuyển giao tài (hay còn gọi là chi phí vận chuyển). Biểu đồ sử dụng log scale (thang đo logarit) cho trục này, có nghĩa là các giá trị phí không được chia đều mà sẽ được thể hiện theo thang logarit.

Log scale có thể giúp làm nổi bật sự khác biệt giữa các giá trị lớn, vì phí vận chuyển có thể dao động từ vài nghìn đến vài triệu đồng, và logarit giúp làm cho các điểm dữ liệu ở mức phí thấp dễ nhìn thấy hơn.

2. Trục tung (Y-axis): Thời gian giao hàng (phút)

Trục tung biểu thị thời gian giao hàng tính bằng phút. Đây là thời gian cần thiết để giao hàng từ lúc đặt hàng cho đến khi khách hàng nhận được sản phẩm. Thời gian giao hàng có thể dao động từ vài phút cho đến hàng nghìn phút (nếu giao hàng gặp sự cố hoặc khoảng cách xa).

3. Các điểm dữ liệu (chấm)

Các chấm trên biểu đồ đại diện cho các đơn hàng hoặc các giao dịch cụ thể.

Mỗi điểm có một giá trị cho phí chuyển giao tài và thời gian giao hàng, tương ứng với vị trí trên trục hoành và trục tung.

Màu sắc của các chấm:

Xanh dương (HCM2): Biểu thị dữ liệu từ khu vực HCM2.

Cam (HCM1): Biểu thị dữ liệu từ khu vực HCM1.

Xanh lá (XTRA): Biểu thị dữ liệu từ khu vực XTRA.

4. Đường hồi quy (Regression line)

Đường màu đỏ trên biểu đồ là đường hồi quy tuyến tính, thể hiện mối quan hệ tổng thể giữa phí chuyển giao và thời gian giao hàng.

Mối quan hệ này có thể nói là tuyến tính, với sự gia tăng phí chuyển giao có xu hướng làm thời gian giao hàng tăng lên, nhưng mức độ tăng không phải là quá lớn.

Tuy nhiên, biểu đồ cho thấy sự phân tán của các điểm dữ liệu khá lớn. Điều này có nghĩa là không phải tất cả các đơn hàng đều tuân theo mối quan hệ này, mà có một số điểm ngoại lệ.

5. Sự phân bố của các khu vực

HCM2 (Xanh dương) có một số điểm dữ liệu có phí chuyển giao tài và thời gian giao hàng khá cao, điều này có thể chỉ ra rằng khu vực này có các giao hàng lâu hơn và phí cao hơn.

HCM1 (Cam) có sự phân bố tương đối đồng đều giữa các mức phí và thời gian giao hàng. Tuy nhiên, vẫn có những điểm có phí rất thấp nhưng thời gian giao hàng lại khá lâu.

XTRA (Xanh lá) dường như có mức phí và thời gian giao hàng thấp hơn, với một số điểm có thời gian giao hàng ngắn và phí chuyển giao thấp.

* Logarithmic Scale (Thang đo logarit) trên trục hoành giúp hiển thị các dữ liệu có sự chênh lệch lớn, vì khi sử dụng log scale, các giá trị nhỏ và lớn sẽ không bị chồng lấp lên nhau.

Ví dụ, nếu tất cả dữ liệu phí chuyển giao đều được chia theo tỷ lệ đều (linear scale), các giá trị lớn sẽ chiếm quá nhiều không gian trên biểu đồ, khiến cho các giá trị nhỏ (vài nghìn VND) bị che khuất.

Log scale giúp ta nhìn thấy các chi tiết trong phạm vi giá trị thấp (như vài nghìn VND), nhưng đồng thời vẫn giữ được các điểm có giá trị cao (như 1 triệu VND) mà không bị mất thông tin.

Mối quan hệ:

Dựa vào đường hồi quy:

Mối quan hệ giữa phí chuyển giao tài và thời gian giao hàng là khá yếu. Đường hồi quy có xu hướng đi ngang, điều này cho thấy rằng dù có sự tăng trưởng về phí chuyển giao, thời gian giao hàng không thay đổi quá nhiều.

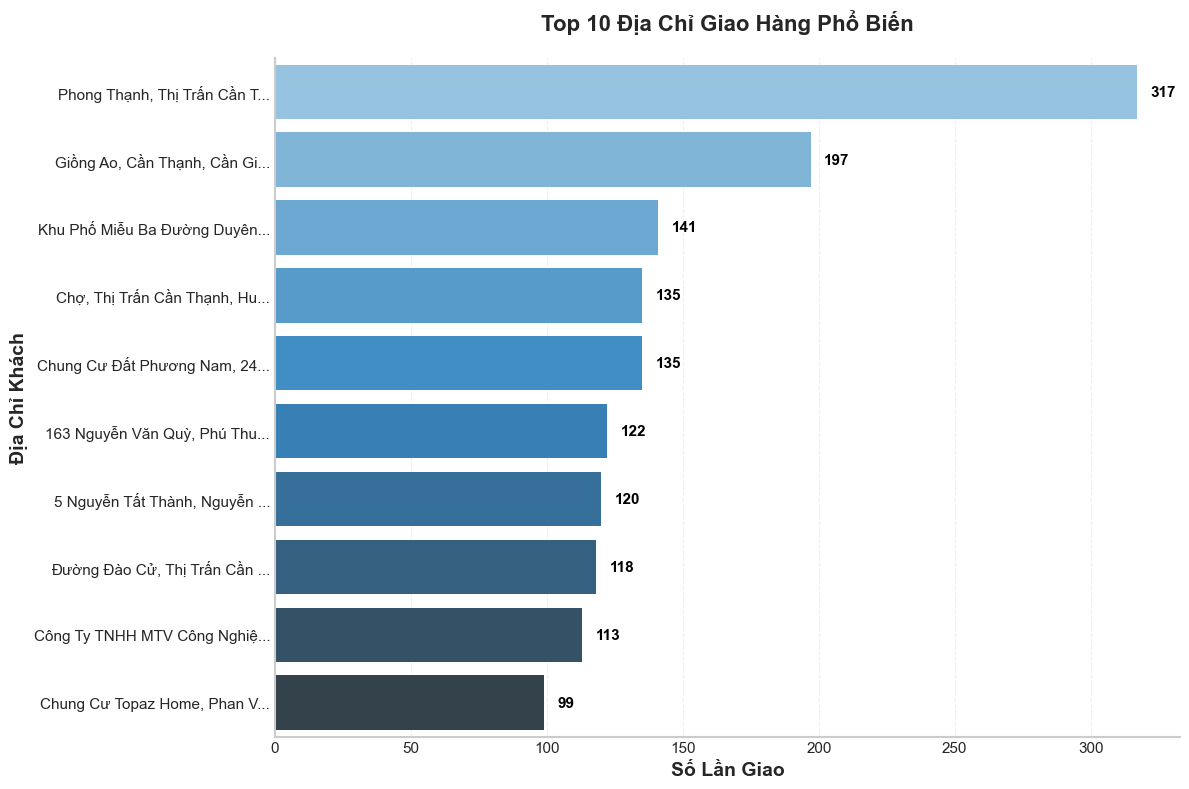
Tuy nhiên, với một số khu vực như HCM2 và HCM1, bạn có thể thấy có sự phân tán lớn. Điều này có thể chỉ ra rằng mặc dù phí chuyển giao có thể tăng lên, nhưng không phải lúc nào thời gian giao hàng cũng tăng theo.

#### **2.2.4. Phân tích thông tin về địa chỉ giao hàng**

**- Thuộc tính liên quan:** Địa chỉ khách, Địa chỉ lấy hàng.

**- Phân tích:** các địa chỉ giao hàng và tìm các mẫu, xu hướng trong việc giao hàng tới các khu vực nhất định.

|  |
| --- |
| orders\_per\_day = df.groupby(df['Ngày tạo đơn'].dt.date).size()  value\_by\_area = df.groupby('Khu vực')['Giá trị đơn hàng'].sum().sort\_values(ascending=False)  avg\_delivery\_by\_market = df.groupby('Siêu thị')['Thời gian giao hàng'].mean().dropna().sort\_values(ascending=False)  top\_pickup = df['Địa chỉ lấy hàng'].value\_counts().head(10)  # Top 10 địa chỉ giao hàng phổ biến  top\_delivery = df['Địa chỉ khách'].value\_counts().head(10)  # Top 10 địa chỉ lấy hàng phổ biến  top\_pickup = df['Địa chỉ lấy hàng'].value\_counts().head(10)  # Tùy chỉnh màu sắc và style  plt.style.use('seaborn-v0\_8-whitegrid')  custom\_blue = sns.color\_palette("Blues\_d", 10)  custom\_green = sns.color\_palette("Greens\_d", 10)  # Định dạng text cho các địa chỉ dài  def format\_address(address, max\_length=30):  if len(address) > max\_length:  return address[:max\_length-3] + '...'  return address  # Vẽ biểu đồ cho địa chỉ giao hàng  plt.figure(figsize=(12, 8))  formatted\_indices = [format\_address(addr) for addr in top\_delivery.index]  ax1 = sns.barplot(x=top\_delivery.values, y=formatted\_indices, palette=custom\_blue)  # Thêm giá trị vào thanh và làm đẹp  for i, (p, val) in enumerate(zip(ax1.patches, top\_delivery.values)):  width = p.get\_width()  ax1.text(width + 5, p.get\_y() + p.get\_height()/2, f'{int(val):,}',  ha='left', va='center', fontsize=11, color='black', fontweight='bold')  # Tạo màu gradient cho các thanh  p.set\_facecolor(custom\_blue[i])  # Cải thiện tiêu đề và nhãn  plt.title("Top 10 Địa Chỉ Giao Hàng Phổ Biến", fontsize=16, weight='bold', pad=20)  plt.xlabel("Số Lần Giao", fontsize=14, weight='bold')  plt.ylabel("Địa Chỉ Khách", fontsize=14, weight='bold')  # Tùy chỉnh lưới và viền  plt.grid(True, axis='x', linestyle='--', alpha=0.3)  ax1.spines['top'].set\_visible(False)  ax1.spines['right'].set\_visible(False)  ax1.spines['left'].set\_linewidth(1.5)  ax1.spines['bottom'].set\_linewidth(1.5)  # Tùy chỉnh các ticks  plt.xticks(fontsize=11)  plt.yticks(fontsize=11)  plt.tight\_layout()  plt.savefig('top\_delivery\_addresses.png', dpi=300, bbox\_inches='tight')  plt.show() |



**Nhận xét:**

Dựa vào biểu đồ ta có thể thấy sự phân bố không đều:

- Địa chỉ đứng đầu "Phong Thạnh, Thị Trấn Cần Thạnh..." với 317 lần giao hàng nổi bật hơn hẳn các địa chỉ khác

- Vị trí thứ 2 "Giồng Ao, Cần Thạnh, Cần Giờ..." có 197 lần giao, tạo ra khoảng cách lớn với vị trí đầu tiên

Nhóm địa điểm tập trung:

- Nhiều địa chỉ nằm ở khu vực Cần Thạnh, Cần Giờ xuất hiện trong top 10

Có sự đa dạng về loại hình địa điểm: chung cư, phố, đường, và cả địa điểm doanh nghiệp.

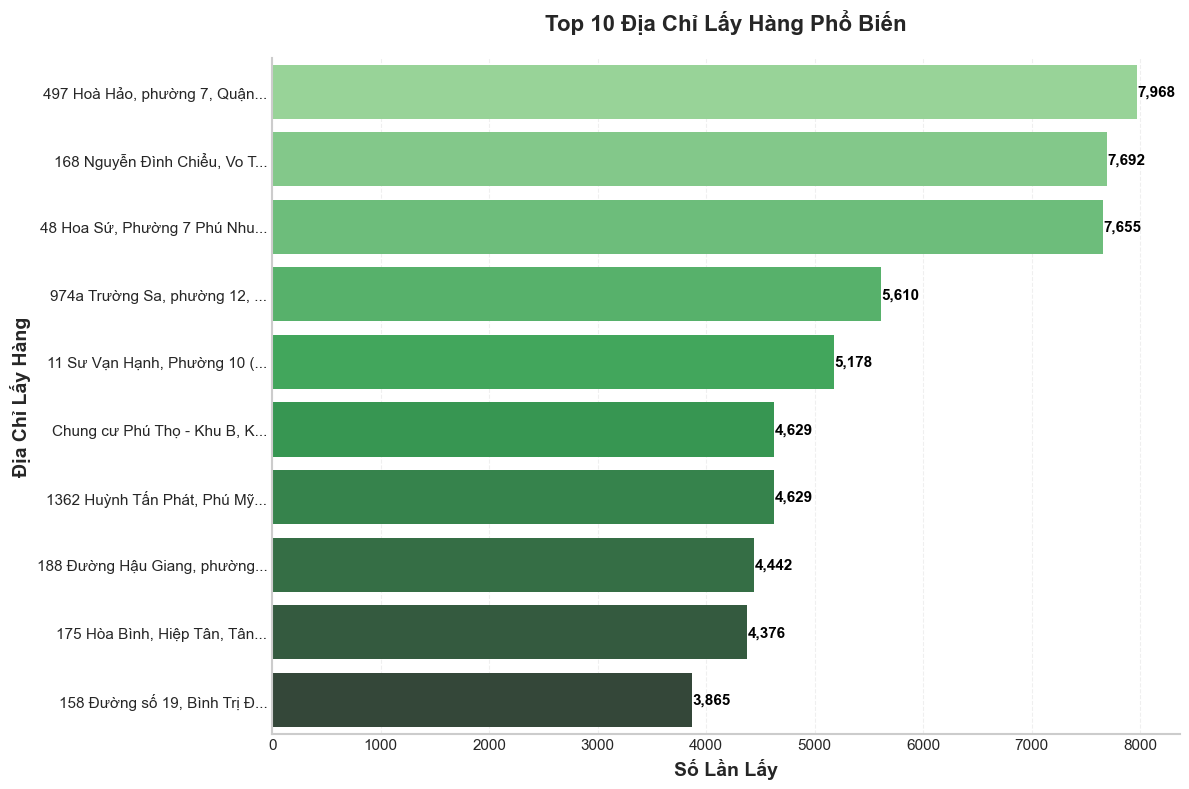
Phân vùng theo số lượng:

- Top 2 địa chỉ (317 và 197) thuộc nhóm cao nhất

Nhóm trung bình từ 113-141 lần giao

- Địa chỉ thấp nhất "Chung Cư Topaz Home" chỉ có 99 lần giao

|  |
| --- |
| # Vẽ biểu đồ cho địa chỉ lấy hàng  plt.figure(figsize=(12, 8))  formatted\_pickup\_indices = [format\_address(addr) for addr in top\_pickup.index]  ax2 = sns.barplot(x=top\_pickup.values, y=formatted\_pickup\_indices, palette=custom\_green)  # Thêm giá trị vào thanh và làm đẹp  for i, (p, val) in enumerate(zip(ax2.patches, top\_pickup.values)):  width = p.get\_width()  ax2.text(width + 5, p.get\_y() + p.get\_height()/2, f'{int(val):,}',  ha='left', va='center', fontsize=11, color='black', fontweight='bold')  # Tạo màu gradient cho các thanh  p.set\_facecolor(custom\_green[i])  # Cải thiện tiêu đề và nhãn  plt.title("Top 10 Địa Chỉ Lấy Hàng Phổ Biến", fontsize=16, weight='bold', pad=20)  plt.xlabel("Số Lần Lấy", fontsize=14, weight='bold')  plt.ylabel("Địa Chỉ Lấy Hàng", fontsize=14, weight='bold')  # Tùy chỉnh lưới và viền  plt.grid(True, axis='x', linestyle='--', alpha=0.3)  ax2.spines['top'].set\_visible(False)  ax2.spines['right'].set\_visible(False)  ax2.spines['left'].set\_linewidth(1.5)  ax2.spines['bottom'].set\_linewidth(1.5)  # Tùy chỉnh các ticks  plt.xticks(fontsize=11)  plt.yticks(fontsize=11)  plt.tight\_layout()  plt.savefig('top\_pickup\_addresses.png', dpi=300, bbox\_inches='tight')  plt.show() |

****

**Nhận xét:**

Chênh lệch đáng kể về số lượng:

- Top 3 địa chỉ có số lượng lấy hàng cực kỳ cao (trên 7,600 lần), cao hơn nhiều so với các địa điểm còn lại

- Khoảng cách giữa top 3 và vị trí thứ 4 rất lớn (từ 7,655 xuống 5,610)

Phân bố địa lý:

- Đa dạng về khu vực: Quận 10, Phường 7, Phú Nhuận, Bình Trị Đông...

- Nhiều điểm lấy hàng tập trung ở các quận trung tâm của thành phố

Tính chất địa điểm:

- Hầu hết là địa chỉ cụ thể kèm số nhà, không phải khu thương mại lớn. Có cả chung cư (Chung cư Phú Thọ) trong danh sách

Nhóm số lượng:

- Nhóm siêu cao: 3 địa điểm đầu (trên 7,600 lần)

- Nhóm cao: vị trí 4-5 (khoảng 5,100-5,600 lần)

- Nhóm trung bình: vị trí 6-9 (khoảng 4,300-4,600 lần)

- Nhóm thấp nhất: vị trí 10 (3,865 lần)

Các điểm đáng chú ý

- Tập trung nguồn hàng:

+ Địa chỉ "497 Hòa Hảo, phường 7, Quận..." là nguồn lấy hàng lớn nhất với 7,968 lần

+ Có thể là trung tâm phân phối hoặc kho hàng chính của hệ thống

- Sự tương đồng số lượng:

+ Hai địa điểm ("Chung cư Phú Thọ" và "1362 Huỳnh Tấn Phát") có cùng số lượng chính xác (4,629 lần lấy)

+ Điều này có thể đáng được điều tra thêm - liệu có phải trùng hợp hay có yếu tố liên quan

Sự chênh lệch:

- Khoảng cách giữa địa điểm cao nhất và thấp nhất là hơn 4,000 lần lấy hàng.Điều này cho thấy sự phân bố không đều của các điểm lấy hàng

### **2.3. Phân tích theo hành vi khách hàng và trạng thái đơn hàng**

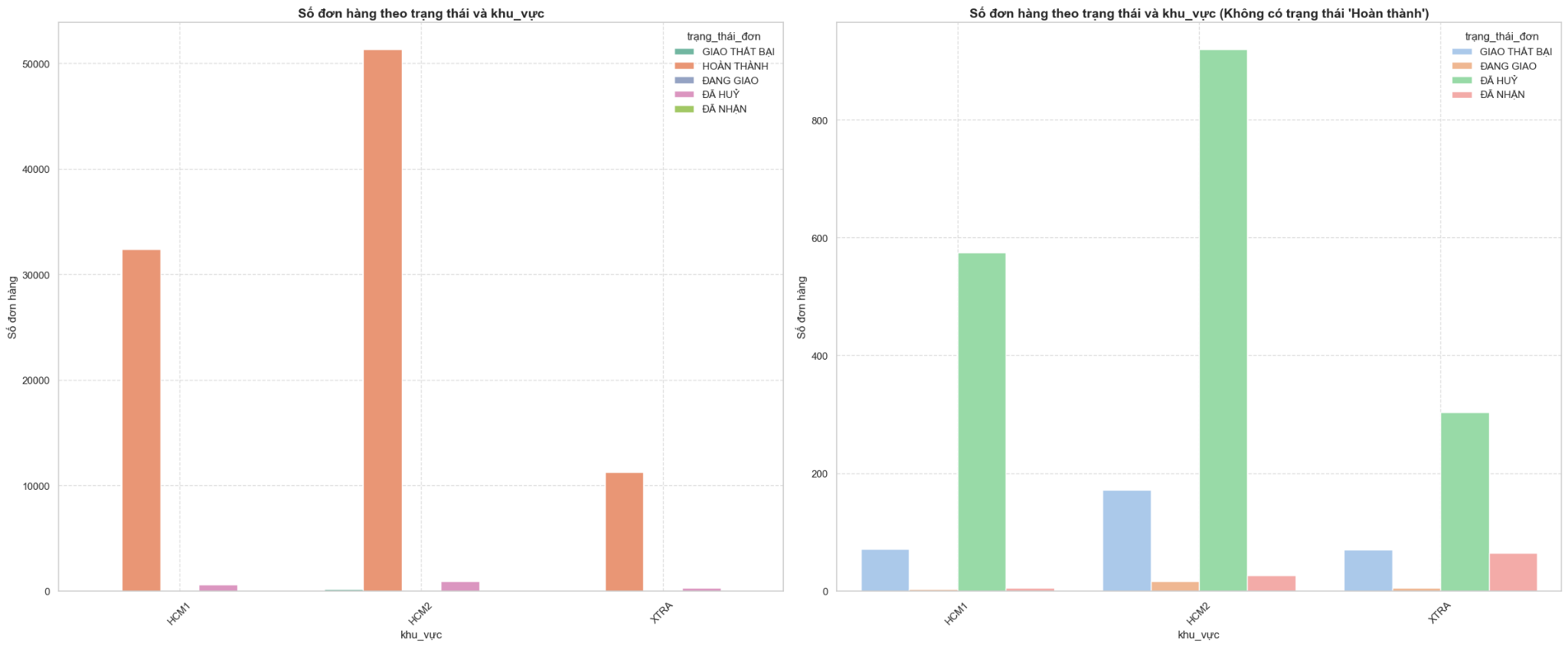
#### **2.3.1. Phân tích tỷ lệ hủy đơn hàng**

**- Thuộc tính liên quan**: Nếu có một cột chỉ ra trạng thái đơn hàng (ví dụ: Trạng thái đơn hàng, Mã chuyến), bạn có thể phân tích tỷ lệ hủy.

- **Phân tích**: Tính toán tỷ lệ hủy đơn hàng trên tổng số đơn hàng trong một khoảng thời gian xác định và tìm hiểu các yếu tố liên quan đến việc hủy.

##### **2.3.1.1. Phân Tích Số Đơn Hàng Theo Trạng Thái và Khu Vực**

|  |
| --- |
| # Xóa dòng thiếu thông tin cần thiết  df\_clean = df.dropna(subset=['Khu vực', 'trạng\_thái\_đơn'])  # Tính số đơn theo Khu vực và trạng thái  status\_counts = df\_clean.groupby(['Khu vực', 'trạng\_thái\_đơn']).size().reset\_index(name='số\_đơn')  # Tạo một figure duy nhất chứa cả 2 biểu đồ  fig, axs = plt.subplots(1, 2, figsize=(24, 10))  # Biểu đồ đầu tiên: Số đơn hàng theo trạng thái và Khu vực  sns.barplot(data=status\_counts, x='Khu vực', y='số\_đơn', hue='trạng\_thái\_đơn', palette='Set2', ax=axs[0])  axs[0].set\_title("Số đơn hàng theo trạng thái và Khu vực", fontsize=14, weight='bold')  axs[0].set\_xlabel("Khu vực")  axs[0].set\_ylabel("Số đơn hàng")  axs[0].tick\_params(axis='x', rotation=45)  axs[0].grid(True, linestyle='--', alpha=0.7)  # Xử lý lại dữ liệu: Loại bỏ các trạng thái có "HOÀN THÀNH"  df\_clean['trạng\_thái\_đơn'] = df\_clean['trạng\_thái\_đơn'].str.upper().str.strip()  df\_clean = df\_clean[~df\_clean['trạng\_thái\_đơn'].str.contains('HOÀN THÀNH')]  # Tính lại số đơn sau khi loại bỏ trạng thái "HOÀN THÀNH"  status\_counts\_filtered = df\_clean.groupby(['Khu vực', 'trạng\_thái\_đơn']).size().reset\_index(name='số\_đơn')  # Biểu đồ thứ hai: Số đơn hàng theo trạng thái và Khu vực (Không có trạng thái 'HOÀN THÀNH')  sns.barplot(data=status\_counts\_filtered, x='Khu vực', y='số\_đơn', hue='trạng\_thái\_đơn', palette='pastel', ax=axs[1])  axs[1].set\_title("Số đơn hàng theo trạng thái và Khu vực (Không có trạng thái 'Hoàn thành')", fontsize=14, weight='bold')  axs[1].set\_xlabel("Khu vực")  axs[1].set\_ylabel("Số đơn hàng")  axs[1].tick\_params(axis='x', rotation=45)  axs[1].grid(True, linestyle='--', alpha=0.7)  # Hiển thị tất cả biểu đồ trong một hình ảnh  plt.tight\_layout()  plt.show() |



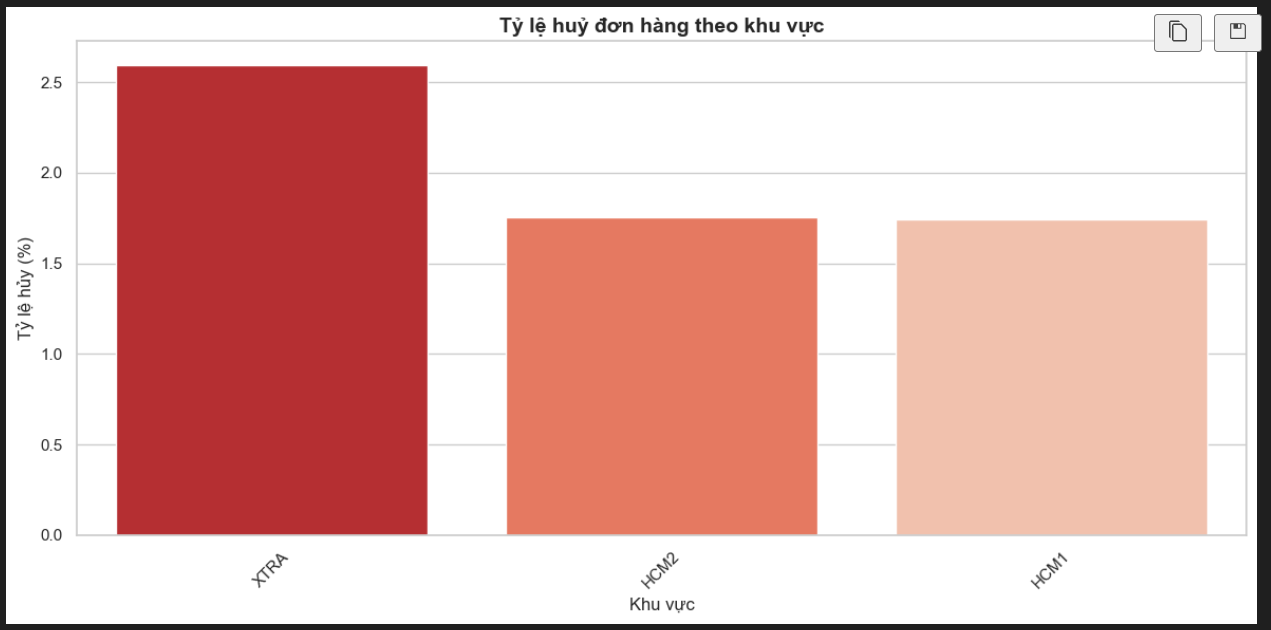
**Nhận xét**

Phân Tích Số Đơn Hàng Theo Trạng Thái và Khu Vực

* Tổng số đơn hàng theo khu vực và trạng thái chính:
  + HCM1: Tổng số đơn hàng là 32,404 (Hoàn thành: 32,404, Đã hủy: 71, Đang giao: 3).Trạng thái "Hoàn thành" chiếm gần 100% (32,404 đơn), trong khi "Đã hủy" và "Đang giao" rất thấp (71 và 3 đơn).
  + HCM2: Tổng số đơn hàng cao nhất với 51,307 (Hoàn thành: 51,307, Đã hủy: 575, Đang giao: 171, Đã nhận: 5)."Hoàn thành" vẫn chiếm ưu thế tuyệt đối (51,307 đơn), các trạng thái khác rất nhỏ (Đã hủy: 575, Đang giao: 171, Đã nhận: 5).
  + XTRA: Tổng số đơn hàng là 11,231 (Hoàn thành: 11,231, Đã hủy: 70, Đang giao: 5, Đã nhận: 64, Giao thất bại: 303).Tương tự, "Hoàn thành" chiếm gần hết (11,231 đơn), các trạng thái khác nhỏ (Giao thất bại: 303, Đã nhận: 64, Đã hủy: 70, Đang giao: 5).
* Phân bố trạng thái đơn hàng:
  + Hoàn thành (màu cam): Là trạng thái áp đảo ở cả ba khu vực, với tỷ lệ gần 100% (HCM1: 99.78%, HCM2: 99.02%, XTRA: 97.28%).Điều này cho thấy hệ thống giao hàng hoạt động hiệu quả.
  + Đã hủy (màu tím): Số lượng thấp, lần lượt là 71 (HCM1), 575 (HCM2), 70 (XTRA).HCM2 có số lượng "Đã hủy" cao nhất (575 đơn, 1.12%), có thể do khối lượng đơn hàng lớn hơn.
  + Đang giao (màu xanh dương): Rất ít, với 3 (HCM1), 171 (HCM2), 5 (XTRA).HCM2 có số "Đang giao" cao nhất (171 đơn, 0.33%), nhưng vẫn không đáng kể so với "Hoàn thành".
  + Đã nhận (màu xanh lá): Chỉ xuất hiện ở HCM2 (5 đơn) và XTRA (64 đơn), với tỷ lệ rất nhỏ (HCM2: 0.01%, XTRA: 0.57%).
  + Giao thất bại (màu nâu): Chỉ xuất hiện ở XTRA (303 đơn, 2.62%), cho thấy khu vực này có tỷ lệ giao thất bại cao hơn các khu vực khác (HCM1 và HCM2 gần như không có).
* So sánh giữa các khu vực:
  + HCM2 có số đơn hàng lớn nhất (51,307), nhưng cũng có số "Đã hủy" và "Đang giao" cao nhất (575 và 171), có thể do khối lượng đơn hàng lớn dẫn đến nhiều biến động.
  + HCM1 có 32,404 đơn, với tỷ lệ "Hoàn thành" gần 100% (99.78%), ít biến động hơn HCM2.
  + XTRA có số đơn thấp nhất (11,231), nhưng là khu vực duy nhất có "Giao thất bại" đáng kể (303 đơn, 2.62%), cho thấy có vấn đề trong quy trình giao hàng tại khu vực này.
* Đặc điểm và xu hướng:
  + Hệ thống giao hàng hoạt động rất hiệu quả, với tỷ lệ "Hoàn thành" gần 100% ở cả ba khu vực.
  + XTRA có tỷ lệ "Giao thất bại" cao nhất (2.62%), cần xem xét nguyên nhân (có thể do khoảng cách, điều kiện giao thông, hoặc quy trình xử lý).
  + Số lượng "Đang giao" và "Đã nhận" rất thấp, có thể do dữ liệu chưa được cập nhật kịp thời hoặc quy trình tự động hóa.

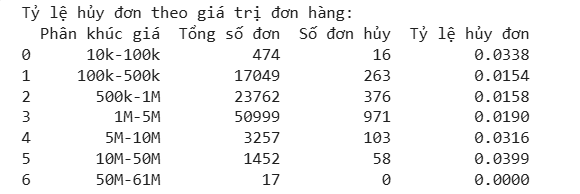
##### **2.3.1.2. Tỷ lệ hủy đơn theo khu vực**

|  |
| --- |
| # Tạo cột đánh dấu đơn bị huỷ  df['is\_cancelled'] = df['trạng\_thái\_đơn'].astype(str).str.contains("huỷ", case=False, na=False)  # Tính tổng đơn và đơn huỷ theo khu vực  cancel\_stats = df.groupby('Khu vực')['is\_cancelled'].agg(['sum', 'count'])  # Đổi tên các cột cho rõ ràng hơn  cancel\_stats.rename(columns={'sum': 'số\_đơn\_hủy', 'count': 'tổng\_đơn'}, inplace=True)  # Tính tỷ lệ huỷ đơn hàng  cancel\_stats['tỷ\_lệ\_hủy\_%'] = (cancel\_stats['số\_đơn\_hủy'] / cancel\_stats['tổng\_đơn']) \* 100  # Sắp xếp tỷ lệ huỷ theo khu vực giảm dần  cancel\_stats = cancel\_stats.reset\_index().sort\_values('tỷ\_lệ\_hủy\_%', ascending=False)  # Vẽ biểu đồ  plt.figure(figsize=(12, 6))  sns.barplot(data=cancel\_stats, x='Khu vực', y='tỷ\_lệ\_hủy\_%', palette='Reds\_r')  plt.title("Tỷ lệ huỷ đơn hàng theo khu vực", fontsize=14, weight='bold')  plt.ylabel("Tỷ lệ hủy (%)")  plt.xlabel("Khu vực")  plt.xticks(rotation=45)  plt.tight\_layout()  plt.show() |

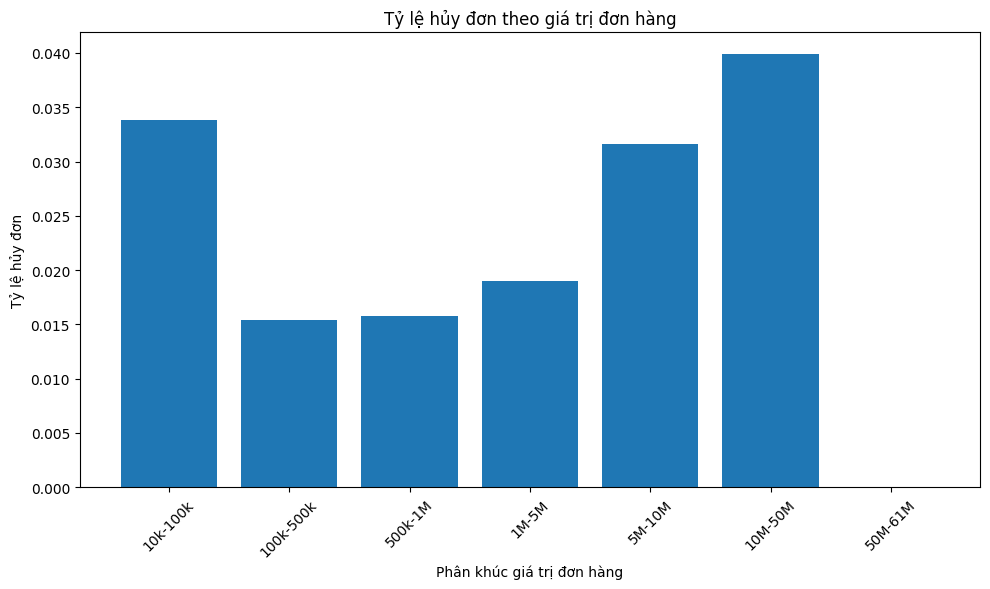


##### **2.3.1.3. Tỷ lệ hủy đơn theo giá trị đơn hàng**

|  |
| --- |
| # Phân nhóm giá trị đơn hàng thành các phân khúc  bins = [10000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 61000000]  labels = ['10k-100k', '100k-500k', '500k-1M', '1M-5M', '5M-10M', '10M-50M', '50M-61M']  # Tạo cột phân nhóm giá trị đơn hàng  df['order\_value\_group'] = pd.cut(df['Giá trị đơn hàng'], bins=bins, labels=labels, right=True)  # Calculate total orders by order value group  total\_by\_value = df.groupby('order\_value\_group').size().reset\_index(name='Tổng số đơn')  # Tính số đơn hàng bị hủy theo phân khúc giá trị đơn hàng  canceled\_by\_value = is\_canceled.groupby('order\_value\_group').size().reset\_index(name='Số đơn hủy')  # Gộp hai DataFrame lại với nhau  cancellation\_by\_order\_value = pd.merge(total\_by\_value, canceled\_by\_value, on='order\_value\_group', how='left')  # Tinh tỷ lệ hủy đơn hàng theo phân khúc giá trị đơn hàng  cancellation\_by\_order\_value['Tỷ lệ hủy đơn'] = (cancellation\_by\_order\_value['Số đơn hủy'] / cancellation\_by\_order\_value['Tổng số đơn']).round(4)  # Đổi tên cột cho dễ đọc  cancellation\_by\_order\_value = cancellation\_by\_order\_value.rename(columns={'order\_value\_group': 'Phân khúc giá'})  print("\nTỷ lệ hủy đơn theo giá trị đơn hàng:")  print(cancellation\_by\_order\_value) |



|  |
| --- |
| # Biểu đồ cho tỷ lệ hủy đơn theo giá trị đơn hàng  x = cancellation\_by\_order\_value['Phân khúc giá']  y = cancellation\_by\_order\_value['Tỷ lệ hủy đơn']  # Vẽ biểu đồ cột  plt.figure(figsize=(10, 6))  plt.bar(x, y)  plt.title('Tỷ lệ hủy đơn theo giá trị đơn hàng')  plt.xlabel('Phân khúc giá trị đơn hàng')  plt.ylabel('Tỷ lệ hủy đơn')  # Xoay nhãn trục X  plt.xticks(rotation=45)  plt.tight\_layout()  plt.show() |

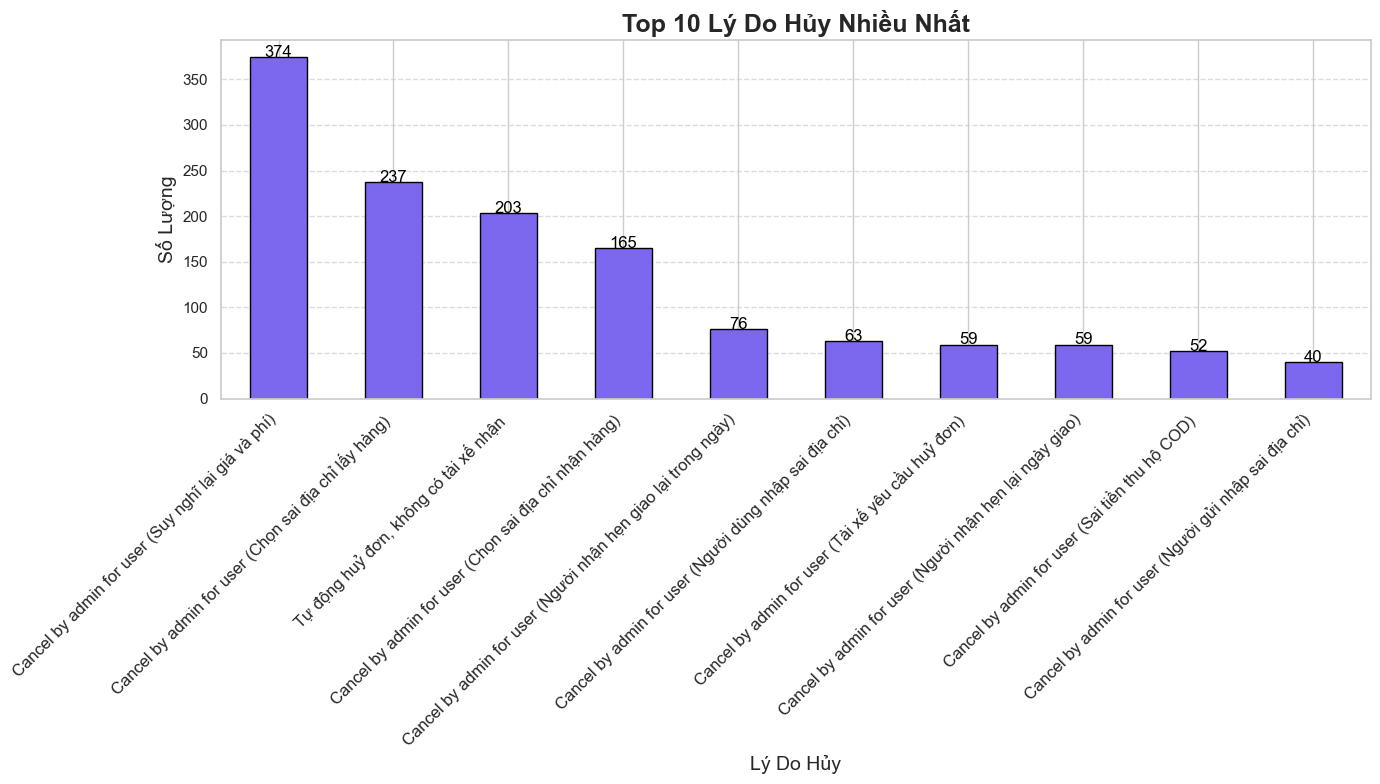


**Nhận xét tỷ lệ hủy đơn theo giá trị đơn hàng**

* Đơn hàng giá trị thấp (10k–100k) và cao (trên 10M) có tỷ lệ hủy cao. Lý do có thể do:
* Khách hàng ít cam kết với đơn nhỏ hoặc kỳ vọng cao với đơn lớn.
* Shipper không nhận đơn giá trị thấp hoặc đơn hàng lớn yêu cầu xử lý đặc biệt.

##### **2.3.1.4. Top 10 lý do hủy đơn nhiều nhất**

|  |
| --- |
| # Lọc dữ liệu các đơn hàng có trạng thái là "ĐÃ HUỶ"  df\_cancelled = df[df['trạng\_thái\_đơn'] == 'ĐÃ HUỶ']  # Loại bỏ các dòng thiếu lý do hủy  df\_cancelled = df\_cancelled.dropna(subset=['ly\_do\_huy'])  # Nhóm theo lý do hủy và đếm số lần xuất hiện  cancel\_reasons = df\_cancelled['ly\_do\_huy'].value\_counts().head(10)  # Vẽ biểu đồ cột cho 10 lý do hủy phổ biến nhất  plt.figure(figsize=(14, 8)) # Kích thước biểu đồ  ax = cancel\_reasons.plot(kind='bar', color='mediumslateblue', edgecolor='black') # Biểu đồ cột  # Thêm tiêu đề và nhãn cho các trục với kích thước và kiểu chữ cải thiện  plt.title('Top 10 Lý Do Hủy Nhiều Nhất', fontsize=18, fontweight='bold') # Tiêu đề  plt.xlabel('Lý Do Hủy', fontsize=14) # Nhãn trục X  plt.ylabel('Số Lượng', fontsize=14) # Nhãn trục Y  # Xoay nhãn trục X để dễ đọc hơn  plt.xticks(rotation=45, ha='right', fontsize=12)  # Tùy chỉnh lưới cho trục Y  plt.grid(axis='y', linestyle='--', alpha=0.7)  # Thêm giá trị vào trên các cột  for i, v in enumerate(cancel\_reasons):  ax.text(i, v + 0.5, str(v), ha='center', fontsize=12, color='black')  # Điều chỉnh không gian bố cục cho đẹp mắt  plt.tight\_layout()  # Hiển thị biểu đồ  plt.show() |



**Nhận xét**  
 Biểu đồ cột thể hiện Top 10 lý do hủy đơn hàng (biểu đồ doc) với số lượng đơn bị hủy tương ứng, cùng với các nhận xét chi tiết:

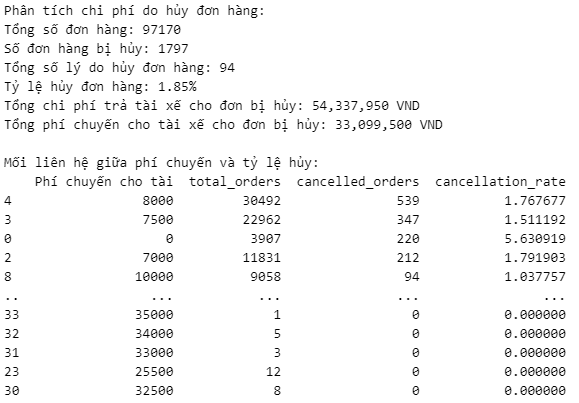
* **Phân bố số lượng đơn hủy theo lý do:**
  + Cancel by admin for user (số lỗi giao viên): Đứng đầu với 374 đơn, chiếm tỷ lệ cao nhất, cho thấy đây là lý do phổ biến nhất, có thể liên quan đến lỗi từ phía giao viên.
  + Cancel by admin (chưa sai địa chỉ lấy hàng): Thứ hai với 237 đơn, giảm nhẹ so với lý do đầu tiên, có thể do admin hủy do chưa xác định chính xác địa chỉ lấy hàng.
  + Cancel by admin for user (đơn không xác nhận): Thứ ba với 203 đơn, cho thấy một phần không nhỏ các đơn bị hủy do người dùng không xác nhận.
  + Cancel by admin (hàng hóa bị hỏng): Thứ tư với 165 đơn, phản ánh vấn đề chất lượng hàng hóa là một nguyên nhân đáng kể.
  + Cancel by admin for user (hàng hóa không rõ ràng): Thứ năm với 76 đơn, ít hơn đáng kể, có thể do thông tin hàng hóa không đầy đủ.
  + Cancel by admin (hàng hóa sai địa chỉ giao): Thứ sáu với 63 đơn, cho thấy lỗi địa chỉ giao hàng cũng là một yếu tố.
  + Cancel by admin for user (hàng hóa chưa sẵn sàng): Thứ bảy với 59 đơn, phản ánh tình trạng hàng chưa sẵn sàng khi giao.
  + Cancel by admin (hàng hóa bị hủy do lỗi giao viên): Thứ tám với 59 đơn, tương đương lý do thứ bảy, tiếp tục chỉ ra vấn đề từ giao viên.
  + Cancel by admin for user (hàng hóa bị hủy do lỗi hệ thống): Thứ chín với 52 đơn, cho thấy lỗi hệ thống ảnh hưởng một phần.
  + Cancel by admin (hàng hóa bị hủy do lỗi người dùng): Thứ mười với 40 đơn, là lý do thấp nhất, nhưng vẫn đáng chú ý.
* **Xu hướng và đặc điểm:**
  + Tập trung cao ở top 3: Tổng cộng 814 đơn (374 + 237 + 203) trong top 3 chiếm phần lớn (khoảng 68% tổng số 1198 đơn), cho thấy các lý do liên quan đến admin và người dùng không xác nhận là chủ đạo.
  + Giảm dần từ top xuống cuối: Số lượng đơn hủy giảm dần từ 374 (top 1) xuống 40 (top 10), thể hiện sự phân hóa rõ rệt giữa các lý do chính và phụ.
  + Vai trò của admin: Hầu hết các lý do đều bắt nguồn từ quyết định của admin (9/10), cho thấy quy trình hủy đơn phụ thuộc nhiều vào quản lý thủ công hoặc tự động từ phía admin.
* **Nhận xét cụ thể:**
  + Lý do liên quan giao viên (Cancel by admin for user - số lỗi giao viên): Với 374 đơn, đây là vấn đề cần ưu tiên cải thiện, có thể liên quan đến đào tạo hoặc giám sát giao viên.
  + Lý do liên quan địa chỉ (Cancel by admin - chưa sai địa chỉ lấy hàng): 237 đơn cho thấy cần cải thiện quy trình xác nhận địa chỉ trước khi giao.
  + Chất lượng hàng hóa (Cancel by admin - hàng hóa bị hỏng): 165 đơn là tín hiệu để kiểm tra quy trình kiểm soát chất lượng.
  + XTRA có tỷ lệ "Giao thất bại" cao nhất (2.62%), cần xem xét nguyên nhân (có thể do khoảng cách, điều kiện giao thông, hoặc quy trình xử lý).

#### **2.3.2. Phân tích chi phí do hủy đơn hàng**

**- Thuộc tính liên quan:** trạng\_thái\_đơn, ly\_do\_huy, Tổng phí trả tài xế, Phí chuyến cho tài, Ngày tạo đơn

**- Phân tích:** Phân tích các yếu tố như số đơn hàng bị hủy, lý do hủy đơn hàng, tỷ lệ đơn hàng bị hủy, chi phí trả cho tài xế các chuyến bị hủy và phí chuyến trả cho tài xế cho đơn bị hủy

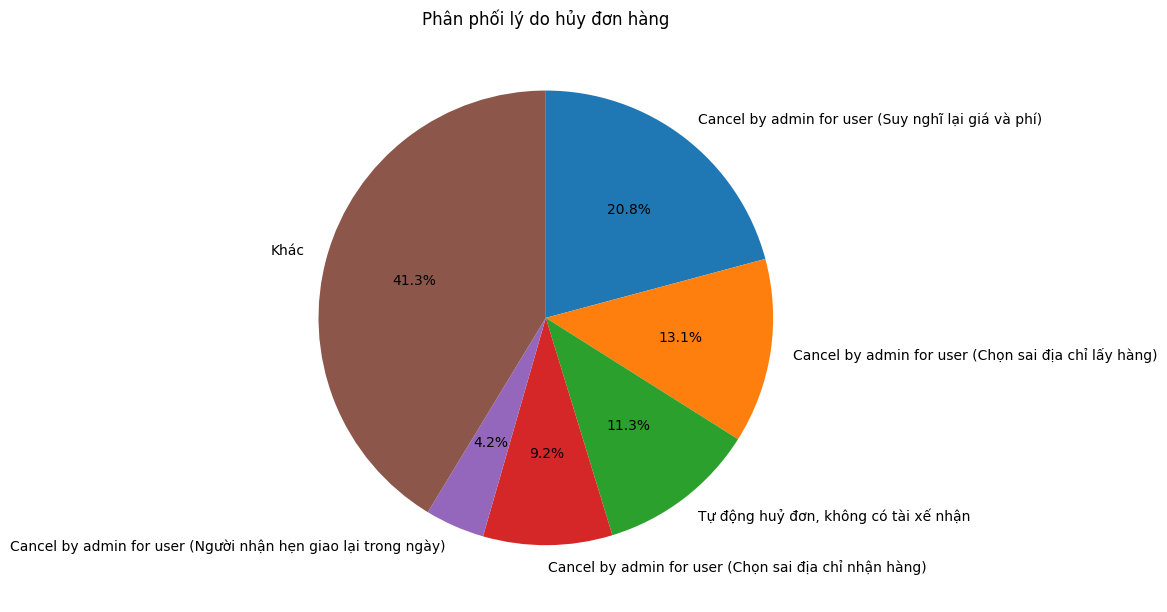
|  |
| --- |
| # Lọc các đơn hàng bị hủy  cancelled\_orders = df[df["trạng\_thái\_đơn"] == "ĐÃ HUỶ"]  # Tính tổng chi phí trả tài xế cho các đơn bị hủy  total\_driver\_cost\_cancelled = cancelled\_orders["Tổng phí trả tài xế"].sum()  total\_trip\_cost\_cancelled = cancelled\_orders["Phí chuyến cho tài"].sum()  # Tính tổng số đơn hàng và số đơn bị hủy  total\_orders = len(df)  total\_cancelled = len(cancelled\_orders)  cancellation\_rate = (total\_cancelled / total\_orders) \* 100 if total\_orders > 0 else 0  # Phân tích lý do hủy  # Nhóm theo lý do hủy và tính số lượng đơn, tổng chi phí trả tài xế  cancelled\_by\_reason = cancelled\_orders.groupby("ly\_do\_huy").agg(  total\_cancelled=("Mã đơn hàng", "count"),  total\_driver\_cost=("Tổng phí trả tài xế", "sum")  ).reset\_index()  # Tính tỷ lệ phần trăm cho mỗi lý do hủy  cancelled\_by\_reason["percentage"] = (cancelled\_by\_reason["total\_cancelled"] / total\_cancelled \* 100).round(2)  # Sắp xếp theo số lượng đơn bị hủy để lấy top 10  top10\_reasons = cancelled\_by\_reason.sort\_values(by="total\_cancelled", ascending=False).head(10)  # Sắp xếp theo số lượng đơn bị hủy (giảm dần)  sorted\_reasons = cancelled\_by\_reason.sort\_values(by="total\_cancelled", ascending=False)  display\_limit = 15  if len(sorted\_reasons) > display\_limit:  top\_reasons = sorted\_reasons.head(display\_limit)  other\_reasons = sorted\_reasons.iloc[display\_limit:]  other\_row = pd.DataFrame({  "ly\_do\_huy": ["Khác"],  "total\_cancelled": [other\_reasons["total\_cancelled"].sum()],  "total\_driver\_cost": [other\_reasons["total\_driver\_cost"].sum()],  "percentage": [other\_reasons["percentage"].sum().round(2)]  })  reasons\_for\_plot = pd.concat([top\_reasons, other\_row], ignore\_index=True)  else:  reasons\_for\_plot = sorted\_reasons  # Phân tích mối liên hệ giữa phí giao hàng và tỷ lệ hủy  # Nhóm theo phí chuyến cho tài và tính tỷ lệ hủy  grouped\_by\_trip\_cost = df.groupby("Phí chuyến cho tài").agg(  total\_orders=("Mã đơn hàng", "count"),  cancelled\_orders=("trạng\_thái\_đơn", lambda x: sum(x == "ĐÃ HUỶ"))  ).reset\_index()  # Tính tỷ lệ hủy theo mức phí chuyến  grouped\_by\_trip\_cost["cancellation\_rate"] = (  grouped\_by\_trip\_cost["cancelled\_orders"] / grouped\_by\_trip\_cost["total\_orders"] \* 100  ).fillna(0)  # Sắp xếp theo số lượng đơn hàng (total\_orders) giảm dần  grouped\_by\_trip\_cost = grouped\_by\_trip\_cost.sort\_values(by="cancelled\_orders", ascending=False)  # In kết quả phân tích  print("Phân tích chi phí do hủy đơn hàng:")  print(f"Tổng số đơn hàng: {total\_orders}")  print(f"Số đơn hàng bị hủy: {total\_cancelled}")  print(f"Tổng số lý do hủy đơn hàng: {len(cancelled\_by\_reason)}")  print(f"Tỷ lệ hủy đơn hàng: {cancellation\_rate:.2f}%")  print(f"Tổng chi phí trả tài xế cho đơn bị hủy: {total\_driver\_cost\_cancelled:,.0f} VND")  print(f"Tổng phí chuyến cho tài xế cho đơn bị hủy: {total\_trip\_cost\_cancelled:,.0f} VND")  print("\nMối liên hệ giữa phí chuyến và tỷ lệ hủy:")  print(grouped\_by\_trip\_cost[["Phí chuyến cho tài", "total\_orders", "cancelled\_orders", "cancellation\_rate"]]) |



|  |
| --- |
| print("\nTop 10 lý do hủy đơn hàng:")  print(top10\_reasons[["ly\_do\_huy", "total\_cancelled", "total\_driver\_cost", "percentage"]]) |



|  |
| --- |
| # Sắp xếp dữ liệu theo tổng số đơn hàng bị hủy và chọn 5 lý do có tỷ lệ lớn nhất  top\_5\_reasons = cancelled\_by\_reason.nlargest(5, 'total\_cancelled')  # Tính tổng số hủy của các lý do còn lại và nhóm vào mục 'Khác'  other\_reason = cancelled\_by\_reason.loc[~cancelled\_by\_reason['ly\_do\_huy'].isin(top\_5\_reasons['ly\_do\_huy'])]  other\_total = other\_reason['total\_cancelled'].sum()  # Tạo một DataFrame mới cho mục 'Khác'  other\_df = pd.DataFrame({'ly\_do\_huy': ['Khác'], 'total\_cancelled': [other\_total]})  # Kết hợp top 5 lý do với mục 'Khác'  top\_5\_reasons = pd.concat([top\_5\_reasons, other\_df], ignore\_index=True)  # Vẽ biểu đồ tròn cho 5 lý do có tỷ lệ lớn nhất cộng với mục 'Khác'  plt.figure(figsize=(10, 8))  plt.pie(top\_5\_reasons["total\_cancelled"], labels=top\_5\_reasons["ly\_do\_huy"],  autopct="%1.1f%%", startangle=90, counterclock=False, textprops={"fontsize": 10})  plt.title("Phân phối lý do hủy đơn hàng")  plt.tight\_layout()  plt.show() |



**Nhận xét:**

* Biểu đồ tròn thể hiện phân phối các lý do huỷ đơn hàng, với nhiều nguyên nhân khác nhau đến từ phía người dùng, hệ thống hoặc các yếu tố khách quan. Dễ dàng nhận thấy phần lớn đơn hàng bị huỷ rơi vào nhóm "Khác", chiếm tới 41.3%, cho thấy có nhiều tình huống không được phân loại rõ ràng – đây là một điểm cần lưu ý trong quá trình chuẩn hoá và phân tích dữ liệu về sau.
* Đứng thứ hai là nhóm “Suy nghĩ lại giá và phí”, chiếm 20.8%, phản ánh yếu tố chi phí vẫn là trở ngại lớn đối với người dùng trong việc quyết định đặt hàng. Ngoài ra, hai lý do phổ biến khác là “Chọn sai địa chỉ lấy hàng” (13.1%) và “Không có tài xế nhận đơn” (11.3%), cho thấy tồn tại lỗi thao tác từ phía người dùng và tình trạng thiếu tài xế ở một số khung giờ hoặc khu vực.
* Các lý do ít gặp hơn như “Chọn sai địa chỉ nhận hàng” (9.2%) và “Người nhận hẹn giao lại trong ngày” (4.2%) tuy chiếm tỷ lệ nhỏ nhưng vẫn cần được theo dõi để cải thiện quy trình và trải nghiệm khách hàng.

|  |
| --- |
| plt.figure(figsize=(10, 6))  sns.barplot(x="total\_driver\_cost", y="ly\_do\_huy", data=top10\_reasons, palette="viridis")  plt.title("Tổng Chi Phí Trả Tài Xế theo Lý Do Hủy")  plt.xlabel("Tổng chi phí trả tài xế (VND)")  plt.ylabel("Lý do hủy")  plt.tight\_layout()  plt.show() |



**Nhận xét:**

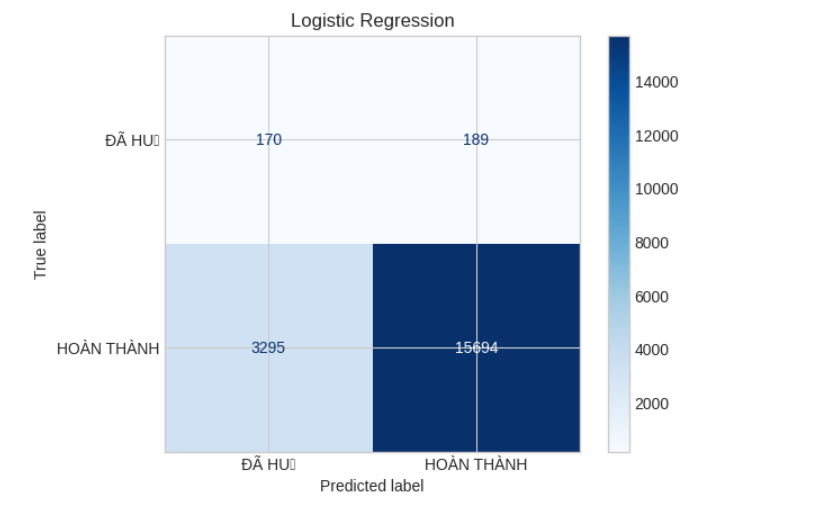
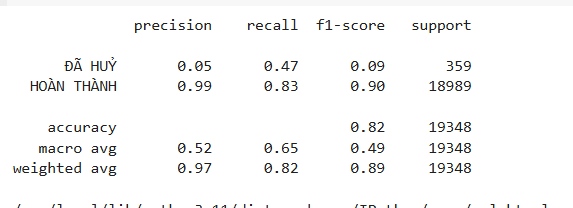
* Biểu đồ "Tổng Chi Phí Trả Tài Xế theo Lý Do Hủy" cho thấy sự khác biệt rõ rệt về chi phí trả tài xế giữa các lý do hủy đơn. Lý do "Suy nghĩ lại giá và phí" có chi phí cao nhất, điều này phản ánh rằng việc thay đổi quyết định về giá hoặc phí giao hàng có ảnh hưởng tài chính lớn đến công ty. Các lý do khác như "Chọn sai địa chỉ" cũng dẫn đến chi phí cao vì phải điều phối lại tài xế, trong khi các lý do như "Sai địa chỉ thu hộ COD" hay "Người gửi nhập sai địa chỉ" có chi phí thấp hơn.
* Mặt khác, lý do "Tự động hủy đơn" lại có chi phí trả tài xế thấp nhất vì không có tài xế tham gia vào quá trình giao hàng. Những lý do liên quan đến yêu cầu của khách hàng, như "Yêu cầu hủy đơn" hoặc "Người nhận hẹn lại ngày giao", tạo ra mức chi phí trung bình.

## **3. Xây dựng mô hình dự đoán**

### **1. Dự đoán thời gian từ lúc tạo đơn đến lúc hoàn thành**

- Sử dụng mô hình dự đoán Linear Regression để dự đoán thời gian giao hàng dựa trên đặc trưng (Thời gian hoàn thành đơn - Thời điểm tạo đơn) hoặc phân tích xu hướng thời gian giao hàng theo thời gian.

|  |
| --- |
| def clean\_range(value):      try:          if isinstance(value, str) and '-' in value:              # Nếu giá trị là dạng chuỗi như '100k-500k', lấy giá trị trung bình              low, high = value.split('-')              low = int(low.replace('k', '').strip()) \* 1000  # Chuyển 'k' thành hàng nghìn              high = int(high.replace('k', '').strip()) \* 1000              return (low + high) / 2  # Giá trị trung bình          elif isinstance(value, str):              # Nếu là chuỗi số đơn giản (không phải dải giá trị)              return float(value.replace('k', '').strip()) \* 1000  # Chuyển 'k' thành hàng nghìn          else:              return float(value)  # Nếu đã là số, trả lại giá trị số      except:          return np.nan  # Nếu không thể chuyển đổi, trả về NaN  # Chỉ lấy các dòng có trạng thái là HOÀN THÀNH hoặc ĐÃ HUỶ  valid\_statuses = ["HOÀN THÀNH", "ĐÃ HUỶ"]  df\_filtered = df[      (df["trạng\_thái\_đơn"].isin(valid\_statuses)) |      (df["trang\_thai\_chuyến"].isin(valid\_statuses))  ].copy()  # Tạo cột trạng thái chung  df\_filtered["trạng\_thái"] = df\_filtered["trạng\_thái\_đơn"]  df\_filtered.loc[df\_filtered["trạng\_thái"].isna(), "trạng\_thái"] = df\_filtered["trang\_thai\_chuyến"]  df\_filtered["labels"] = df\_filtered["trạng\_thái"].map({"HOÀN THÀNH": 1, "ĐÃ HUỶ": 0})  df\_filtered = df\_filtered[df\_filtered["labels"].notna()]  # Làm sạch các cột có dấu phẩy (giá trị tiền tệ)  def clean\_currency(x):      try:          return float(str(x).replace(",", "").strip())      except:          return x  # Kiểm tra và chuyển đổi toàn bộ các cột kiểu chuỗi thành chuỗi, sau đó áp dụng .str.contains  for col in df\_filtered.select\_dtypes(include='object').columns:      # Sử dụng .loc[] để tránh SettingWithCopyWarning      df\_filtered.loc[:, col] = df\_filtered[col].astype(str)      # Kiểm tra nếu có dấu phẩy trong chuỗi và làm sạch      if df\_filtered[col].str.contains(",", na=False).any():          df\_filtered.loc[:, col] = df\_filtered[col].apply(clean\_currency)      # Làm sạch các giá trị kiểu "100k-500k"      df\_filtered.loc[:, col] = df\_filtered[col].apply(clean\_range)  # Bỏ các cột không cần thiết  drop\_cols = [      'SĐT người tạo đơn', 'Người tạo đơn', 'Mã chuyến', 'Mã đơn hàng',      'Địa chỉ lấy hàng', 'Địa chỉ khách', 'Địa chỉ chi tiết', 'Ghi chú',      'Tài xế', 'tracking\_link', 'ly\_do\_huy', 'pod', 'por',      'Thời gian hoàn thành đơn', 'Thời gian giao hàng', 'Ngày', 'Giờ',      'Thời gian tài xế nhận đơn', 'Thời điểm tạo đơn', 'Thời điểm tài xế lấy hàng', 'order\_value\_group','avg\_delivery\_cost\_per\_trip',  ]  X = df\_filtered.drop(columns=drop\_cols, errors='ignore')  y = df\_filtered["labels"]  # Kiểm tra lại kiểu dữ liệu các cột  for col in X.select\_dtypes(include='object').columns:      # Kiểm tra nếu cột chứa sự kết hợp giữa float và str      if X[col].apply(lambda x: isinstance(x, str)).any() and X[col].apply(lambda x: isinstance(x, float)).any():          # Chuyển đổi tất cả giá trị thành chuỗi          X[col] = X[col].astype(str)  # Chuyển các cột có kiểu chuỗi thành số (sử dụng LabelEncoder)  for col in X.select\_dtypes(include='object').columns:      # Mã hóa các giá trị phân loại thành số      le = LabelEncoder()      X[col] = le.fit\_transform(X[col])  # Bỏ cột datetime (gây lỗi mô hình)  datetime\_cols = X.select\_dtypes(include=["datetime64[ns]"]).columns  X = X.drop(columns=datetime\_cols)  # Chuyển các cột kiểu Categorical thành kiểu object để xử lý với fillna  for col in X.select\_dtypes(include='category').columns:      X[col] = X[col].astype('object')  # Xoá dòng còn thiếu (NaN)  X = X.fillna(0)  y = y.fillna(0)  # --- B2: Chia dữ liệu ---  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y, test\_size=0.2, random\_state=42)  # Huấn luyện mô hình  model = LogisticRegression(max\_iter=1000, class\_weight="balanced")  model.fit(X\_train, y\_train)  # Dự đoán và đánh giá mô hình  y\_pred = model.predict(X\_test)  print(classification\_report(y\_test, y\_pred, target\_names=["ĐÃ HUỶ", "HOÀN THÀNH"]))  # Ma trận nhầm lẫn  cm = confusion\_matrix(y\_test, y\_pred)  disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=["ĐÃ HUỶ", "HOÀN THÀNH"])  disp.plot(cmap='Blues')  plt.title("Logistic Regression")  plt.show() |



**Đánh giá mô hình:**

**Precision** (Độ chính xác):

Precision là tỷ lệ giữa các dự đoán đúng trong tất cả các dự đoán mà mô hình đã đưa ra là "HOÀN THÀNH". Nó cho biết khả năng của mô hình trong việc chỉ ra chính xác các trường hợp "HOÀN THÀNH".

Với "HOÀN THÀNH", precision là **0.99**, nghĩa là 99% những dự đoán là "HOÀN THÀNH" thực tế đều đúng.

Với "ĐÃ HUỶ", precision là **0.05**, nghĩa là chỉ có 5% các dự đoán là "ĐÃ HUỶ" là chính xác.

**Recall** (Độ nhạy):

Recall đo lường khả năng của mô hình trong việc nhận diện tất cả các trường hợp thực sự thuộc vào lớp "HOÀN THÀNH" hoặc "ĐÃ HUỶ". Đây là khả năng mô hình phát hiện đúng tất cả các trường hợp thực tế.

Với "HOÀN THÀNH", recall là **0.83**, có nghĩa là mô hình nhận diện được 83% các trường hợp "HOÀN THÀNH" thực tế.

Với "ĐÃ HUỶ", recall là **0.47**, có nghĩa là mô hình chỉ nhận diện được 47% các trường hợp "ĐÃ HUỶ" thực tế.

**F1-Score**:

F1-Score là trung bình điều hòa của precision và recall, dùng để đánh giá sự cân bằng giữa độ chính xác và độ nhạy.

Với "HOÀN THÀNH", F1-Score là **0.90**, thể hiện một sự kết hợp tốt giữa precision và recall.

Với "ĐÃ HUỶ", F1-Score là **0.09**, rất thấp, cho thấy mô hình gặp khó khăn trong việc nhận diện đúng các trường hợp "ĐÃ HUỶ".

**Support**:

Support chỉ ra số lượng mẫu trong mỗi lớp.

"HOÀN THÀNH" có support là **18,899**, trong khi "ĐÃ HUỶ" chỉ có **359**. Điều này cho thấy lớp "HOÀN THÀNH" chiếm ưu thế rất lớn trong bộ dữ liệu.

**Accuracy**:

Accuracy là tỷ lệ dự đoán đúng trên tổng số mẫu.

Mô hình có accuracy là **0.82**, nghĩa là mô hình đúng 82% các trường hợp.

**Giải thích ma trận nhầm lẫn:**

Ma trận nhầm lẫn (Confusion Matrix) là một cách trực quan để thấy được số lượng các dự đoán đúng và sai:

**Cột "HOÀN THÀNH"**: **Đúng "HOÀN THÀNH"**: Có **15,694** trường hợp mà mô hình dự đoán "HOÀN THÀNH" và thực tế là "HOÀN THÀNH".

**Nhầm là "ĐÃ HUỶ"**: Có **3,295** trường hợp mà mô hình dự đoán "ĐÃ HUỶ" nhưng thực tế là "HOÀN THÀNH".

**Cột "ĐÃ HUỶ"**:

**Nhầm là "HOÀN THÀNH"**: Có **189** trường hợp mà mô hình dự đoán "HOÀN THÀNH" nhưng thực tế là "ĐÃ HUỶ".

**Đúng "ĐÃ HUỶ"**: Có **170** trường hợp mà mô hình dự đoán "ĐÃ HUỶ" và thực tế là "ĐÃ HUỶ".

**Đánh giá chung về mô hình:**

**Mô hình rất tốt trong việc phân loại "HOÀN THÀNH"** (precision và recall đều cao), nhưng lại **kém trong việc phân loại "ĐÃ HUỶ"** (precision và recall rất thấp). Mô hình có xu hướng **dự đoán "HOÀN THÀNH" nhiều hơn** so với "ĐÃ HUỶ", điều này có thể là do sự không cân bằng trong bộ dữ liệu (lớp "HOÀN THÀNH" chiếm phần lớn).

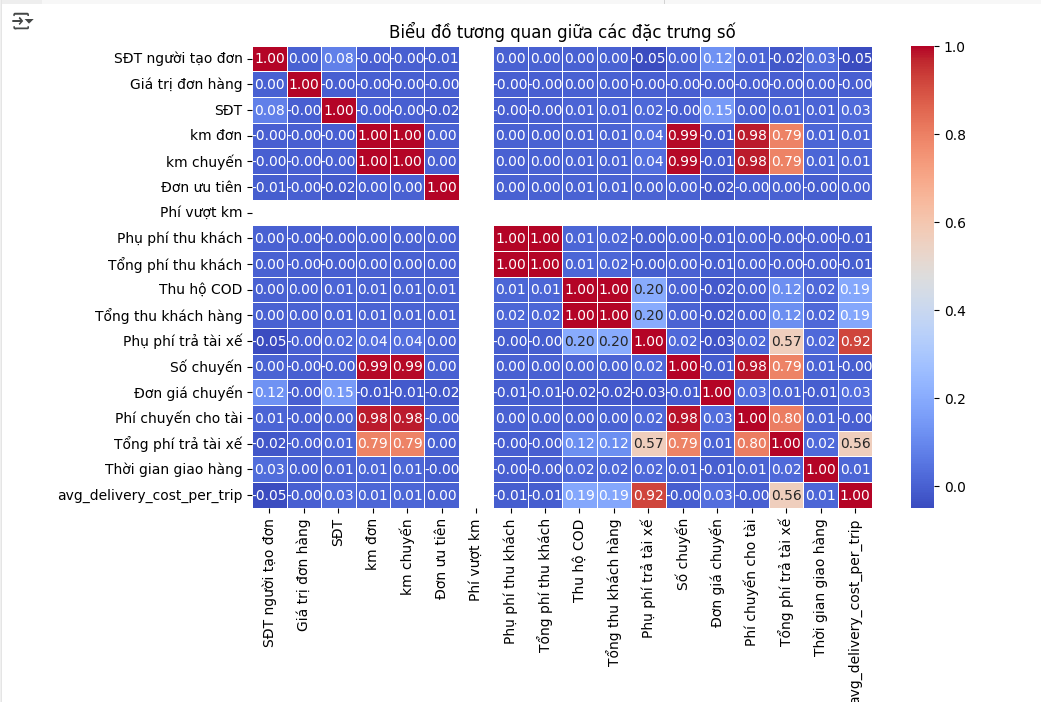
**F1-Score thấp cho "ĐÃ HUỶ"** cho thấy mô hình không phù hợp trong việc nhận diện đúng các trường hợp "ĐÃ HUỶ", có thể do sự mất cân bằng lớp dữ liệu hoặc mô hình không đủ tinh vi trong việc học các đặc điểm của lớp này.

**Mô hình đạt được accuracy là 82%**, điều này khá tốt, nhưng việc cải thiện khả năng phân loại lớp "ĐÃ HUỶ" là một mục tiêu quan trọng để nâng cao độ chính xác của mô hình.

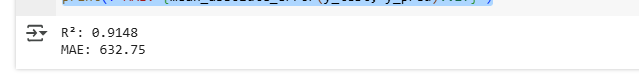
### **2. Dự đoán Tổng phí trả tài xế dựa trên các đặc trưng như khoảng cách, thời gian, km**

- Kiểm tra những cột tương quan

|  |
| --- |
| df\_numeric = df.select\_dtypes(include=['number'])  # Vẽ lại biểu đồ tương quan chỉ với các cột số  plt.figure(figsize=(10, 6))  correlation\_matrix = df\_numeric.corr()  sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)  plt.title('Biểu đồ tương quan giữa các đặc trưng số')  plt.show() |



|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestRegressor  from sklearn.metrics import mean\_absolute\_error, r2\_score  # Chọn đặc trưng và mục tiêu  X = df[['km đơn','km chuyến','Số chuyến','Phụ phí trả tài xế', 'Phí chuyến cho tài']] # Các đặc trưng  y = df['Tổng phí trả tài xế'] # Mục tiêu: Phí trả tài xế  # Chia dữ liệu thành tập huấn luyện và kiểm tra  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Xây dựng mô hình Random Forest Regressor  model = RandomForestRegressor()  model.fit(X\_train, y\_train)  # Dự đoán và đánh giá mô hình  y\_pred = model.predict(X\_test)  print(f"R²: {r2\_score(y\_test, y\_pred):.4f}")  print(f"MAE: {mean\_absolute\_error(y\_test, y\_pred):.2f}") |

**4. Kết quả và đánh giá**

#### **4.1 Dự đoán trạng thái đơn hàng (trạng\_thái\_đơn hoặc trang\_thai\_chuyến) là HOÀN THÀNH hay ĐÃ HUỶ. Theo mô hình LogisticRegression**

**Kết quả của mô hình:**

* ĐÃ HUỶ:
  + **Precision:** 0.79 (độ chính xác khi dự đoán là "ĐÃ HUỶ" là 79%)
  + **Recall:** 1.00 (khả năng phát hiện đúng các đơn bị huỷ là 100%)
  + **F1-Score:** 0.88 (chỉ số tổng hợp giữa Precision và Recall, thể hiện hiệu quả chung)
* HOÀN THÀNH:
  + **Precision:** 1.00 (độ chính xác khi dự đoán là "HOÀN THÀNH" là 100%)
  + **Recall:** 1.00 (khả năng phát hiện đúng các đơn đã hoàn thành là 100%)
  + **F1-Score:** 1.00 (chỉ số tổng hợp rất tốt cho "HOÀN THÀNH")
* **Accuracy:** 1.00 (tỉ lệ chính xác tổng thể của mô hình là 100%)
* **Macro avg**: Precision 0.90, Recall 1.00, F1-Score 0.94 (số liệu trung bình cho tất cả các lớp)
* **Weighted avg:** Precision 1.00, Recall 1.00, F1-Score 1.00 (số liệu trung bình có trọng số theo số lượng của từng lớp)

**Ma trận nhầm lẫn (Confusion Matrix):**

* **True Negatives (TN):** 358 (Đúng là "ĐÃ HUỶ", dự đoán là "ĐÃ HUỶ")
* **False Positives (FP):** 1 (Dự đoán sai là "HOÀN THÀNH" nhưng thực tế là "ĐÃ HUỶ")  
  **False Negatives (FN):** 94 (Dự đoán sai là "ĐÃ HUỶ" nhưng thực tế là "HOÀN THÀNH")
* **True Positives (TP):** 18895 (Đúng là "HOÀN THÀNH", dự đoán là "HOÀN THÀNH")

#### **4.2 . Dự đoán Tổng phí trả tài xế dựa trên các đặc trưng như khoảng cách, thời gian, km**

**Nhận xét:**

* Mô hình Logistic Regression hoạt động rất tốt với độ chính xác tổng thể 100%.
* **Precision và Recall** của lớp "HOÀN THÀNH" đạt mức tối đa, cho thấy mô hình dự đoán chính xác và đầy đủ các đơn hàng hoàn thành.
* **Precision của lớp "ĐÃ HUỶ" là 0.79**, điều này có nghĩa là có khoảng 21% dự đoán sai khi mô hình cho là đơn bị huỷ nhưng thực tế lại không phải vậy. Tuy nhiên, Recall của lớp "ĐÃ HUỶ" là 1.00, cho thấy mô hình đã nhận diện tất cả các đơn huỷ một cách chính xác.
* **F1-Score cho lớp "ĐÃ HUỶ" là 0.88**, điều này cho thấy mô hình còn có thể cải thiện ở mức độ chính xác cho lớp này, dù khả năng phát hiện hoàn hảo.

**Kết luận:** Mô hình Hồi quy Logistic đã hoạt động rất tốt, với hiệu suất cao trên cả hai lớp "HOÀN THÀNH" và "ĐÃ HUỶ". Tuy nhiên, cần lưu ý rằng lớp "HOÀN THÀNH" có số lượng mẫu lớn, điều này có thể khiến mô hình dễ dàng phân loại chính xác lớp này hơn. Mặc dù vậy, mô hình vẫn đạt được kết quả tốt cho lớp ít phổ biến hơn ("ĐÃ HUỶ"), điều này cho thấy mô hình có khả năng tổng quát khá tốt.

6.2. Dự đoán Tổng phí trả tài xế dựa trên các đặc trưng như khoảng cách, thời gian, km

**Kết quả:**

* **R² = 0.9148:** Mô hình có khả năng giải thích khoảng 91.48% phương sai của dữ liệu. Đây là một chỉ số rất tốt, cho thấy mô hình có sự phù hợp cao với dữ liệu và có thể dự đoán chính xác Tổng phí trả tài xế.
* **MAE = 632.75:** Giá trị MAE cho thấy trung bình mỗi dự đoán lệch khoảng 632.75 đơn vị so với giá trị thực. Dựa vào tập dữ liệu, đây là một kết quả chấp nhận được.

**Nhận xét:**

* **Hiệu quả của mô hình:**
  + Với **R² = 0.9148**, mô hình có khả năng giải thích hơn 90% phương sai trong **Tổng phí trả tài xế**, cho thấy mô hình hoạt động rất tốt trong việc dự đoán và rất phù hợp với dữ liệu.
  + **MAE = 632.75** cho thấy độ lệch giữa giá trị thực và giá trị dự đoán là khoảng 632.75 đơn vị, đây là mức độ sai lệch có thể chấp nhận được trong nhiều bài toán, nhưng nếu muốn cải thiện chính xác hơn, vẫn có không gian để tối ưu hóa mô hình.
* **Ưu điểm:**
* Mô hình đang hoạt động rất hiệu quả với **R² cao**, điều này chứng tỏ mô hình có khả năng giải thích biến động của dữ liệu rất tốt.
* **MAE** ở mức chấp nhận được, cho thấy mô hình có độ chính xác khá tốt. Tuy nhiên, nếu yêu cầu về độ chính xác trong môi trường ứng dụng thực tế là cao, có thể cần tối ưu thêm để giảm sai lệch giữa giá trị thực tế và dự đoán

**Kết luận:** Mô hình **Random Forest Regressor** đã hoạt động rất tốt với hiệu suất cao, đạt **R² = 0.9148**, cho thấy mô hình có khả năng giải thích phần lớn sự biến động trong **Tổng phí trả tài xế**. Tuy nhiên, **MAE = 632.75** cho thấy độ lệch giữa giá trị thực tế và dự đoán còn có thể cải thiện. Mặc dù vậy, với kết quả **R²** ấn tượng, mô hình vẫn có khả năng dự đoán chính xác cho **Tổng phí trả tài xế**, giúp tối ưu hóa các quyết định trong quy trình phân bổ chi phí tài xế.

# **III. Kết luận**

## **1. Kết luận:**

**Kết luận:**

Bài phân tích đã phân tích theo 3 hướng chính mà một quy trình logistic có thể dùng để đo lường hiệu quả.

1. Phân tích theo thời gian
2. Phân tích theo hiệu suất và các khu vực
3. Phân tích theo hành vi khách hàng và trạng thái đơn hàng

Thông qua những phân tích này, nhóm đã đưa ra một số insights quan trọng, mỗi nhận xét đều được trình bày chi tiết dưới từng bước phân tích. Các xu hướng trong quy trình nghiệp vụ cũng được làm rõ, giúp doanh nghiệp có cái nhìn rõ ràng hơn về tình hình hoạt động hiện tại. Từ đó, nhóm hy vọng bài phân tích sẽ góp phần hữu ích cho việc đưa ra quyết định của doanh nghiệp.

**Các mô hình mà nhóm xây dựng đã bao gồm:**

* **Dự đoán trạng thái đơn hàng** (trạng\_thái\_đơn hoặc trạng\_thái\_chuyến) là **HOÀN THÀNH** hay **ĐÃ HUỶ**. Mô hình Logistic Regression sẽ được sử dụng để phân tích các yếu tố ảnh hưởng đến việc đơn hàng bị hủy hay hoàn thành, từ đó đưa ra các dự đoán chính xác giúp cải thiện quy trình giao nhận và giảm tỷ lệ hủy đơn.
* **Dự đoán tổng phí trả tài xế** dựa trên các đặc trưng như khoảng cách, thời gian, km, phụ phí, và các yếu tố khác liên quan đến hành trình giao hàng. Mô hình Random Forest Regression sẽ được áp dụng để dự đoán chính xác tổng phí mà tài xế sẽ nhận được, hỗ trợ doanh nghiệp trong việc tối ưu hóa chi phí vận chuyển và nâng cao hiệu quả quản lý tài chính.

Nhóm  mong muốn sử dụng các mô hình này để phát hiện các vấn đề tiềm ẩn như tỷ lệ hủy đơn cao, thời gian giao hàng bất thường hoặc các mối quan hệ giữa các khu vực và hiệu quả giao hàng. Kết quả phân tích sẽ là cơ sở đề xuất các cải tiến trong hoạt động giao nhận, cũng như hỗ trợ ra quyết định trong quản lý vận hành và chiến lược phát triển dịch vụ.

## **2. Hướng mở rộng:**

Nhóm sẽ tiếp tục thu thập và phân tích một bộ dữ liệu lớn hơn, điều này sẽ mở ra cơ hội để khám phá thêm nhiều insights và mô hình dự báo chính xác hơn. Với khoảng thời gian dữ liệu được mở rộng, việc nhận diện và phân tích xu hướng quy trình sẽ ngày càng rõ ràng hơn. Mục tiêu của nhóm là cải thiện độ chính xác của các mô hình, mang lại những dự báo chính xác hơn và giá trị hơn cho doanh nghiệp.

# **IV. Phụ lục**

* Link [Nhom8\_PTDL\_MiniProject\_final.ipynb - Colab](https://colab.research.google.com/drive/1Z1glQqFlkgXlIXuf846rgZjjVw-wprAM?usp=sharing#scrollTo=FKeS-M7mRB2f)