

Mục tiêu dự án với tập dữ liệu Amazon Sale Report

1. Mục tiêu tổng quát

Khai phá dữ liệu bán hàng để phát hiện các yếu tố ảnh hưởng đến trạng thái đơn hàng, doanh thu và hiệu suất giao hàng.

Dự đoán kết quả trong tương lai như: đơn hàng có bị hủy hay không, hoặc doanh thu của một đơn hàng.

Đề xuất giải pháp cải thiện hoạt động kinh doanh: tối ưu hóa vận chuyển, giảm đơn hàng bị hủy, tăng doanh thu.

2. Các câu hỏi nghiên cứu có thể đặt ra

Phân tích mô tả

- Sản phẩm nào bán chạy nhất?
- Thành phố nào có doanh số cao nhất?
- Tỷ lệ đơn hàng bị hủy là bao nhiêu?
- Những yếu tố nào thường xuất hiện trong các đơn bị hủy?

Hồi quy

- Những yếu tố nào ảnh hưởng đến doanh thu (Amount)?
- Có thể dự đoán doanh thu của một đơn hàng dựa trên sản phẩm, khu vực, kiểu giao hàng...?

Phân loại

- Đơn hàng này có bị hủy (Cancelled) không?
- Mô hình có thể phân loại trạng thái đơn hàng dựa trên thông tin đầu vào như thành phố, SKU, dịch vụ vận chuyển...?

3. Mục đích xây dựng mô hình

✓ Hồi quy (Regression) Dự đoán số tiền (Amount) của một đơn hàng dựa trên các đặc trưng như loại hàng, thành phố, dịch vụ vận chuyển

✓ Phân loại (Classification) Phân loại trạng thái đơn hàng: Shipped, Delivered, Cancelled, v.v.

Triển Khai

Đọc và xem dữ liệu

In [33]: `import pandas as pd`

```
# Đọc dữ liệu
df = pd.read_csv("Amazon Sale Report.csv")

# Xem kích thước và 5 dòng đầu tiên
print("♦ Kích thước dữ liệu:", df.shape)
display(df.head())


# Xem thông tin tổng quan về kiểu dữ liệu và giá trị thiếu
print("\n🔍 Thông tin các cột:")
df_info = pd.DataFrame({
    'Data Type': df.dtypes,
    'Missing Values': df.isnull().sum(),
    'Unique Values': df.nunique()
})
display(df_info)
```

♦ Kích thước dữ liệu: (128975, 24)

D:\Temp\ipykernel_10684\2360503588.py:4: DtypeWarning: Columns (23) have mixed type s. Specify dtype option on import or set low_memory=False.
df = pd.read_csv("Amazon Sale Report.csv")

index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	C
0	0	405-8078784-5731545	04-30-22	Cancelled	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL
2	2	404-0687676-7273146	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL
3	3	403-9615377-8133951	04-30-22	Cancelled	Merchant	Amazon.in	Standard	J0341	J0341-DR-L
4	4	407-1069790-7240320	04-30-22	Shipped	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL

5 rows × 24 columns

 Thông tin các cột:

	Data Type	Missing Values	Unique Values
index	int64	0	128975
Order ID	object	0	120378
Date	object	0	91
Status	object	0	13
Fulfilment	object	0	2
Sales Channel	object	0	2
ship-service-level	object	0	2
Style	object	0	1377
SKU	object	0	7195
Category	object	0	9
Size	object	0	11
ASIN	object	0	7190
Courier Status	object	6872	3
Qty	int64	0	10
currency	object	7795	1
Amount	float64	7795	1410
ship-city	object	33	8955
ship-state	object	33	69
ship-postal-code	float64	33	9459
ship-country	object	33	1
promotion-ids	object	49153	5787
B2B	bool	0	2
fulfilled-by	object	89698	1
Unnamed: 22	object	49050	1

```
In [34]: # Tạo bảng mô tả thuộc tính
column_descriptions = {
    'Order ID': 'Mã đơn hàng duy nhất',
    'Date': 'Ngày tạo đơn hàng',
    'Status': 'Trạng thái đơn hàng (Shipped, Cancelled, Delivered, ...)',
    'Fulfilment': 'Hình thức hoàn tất đơn hàng (Merchant hoặc Amazon)',
    'Sales Channel': 'Kênh bán hàng (Online, Offline)',
    'Ship Service Level': 'Mức độ dịch vụ vận chuyển (Standard, Express, ...)',
    'Style': 'Loại sản phẩm hoặc biến thể',
    'SKU': 'Mã định danh sản phẩm',
```

```
'Category': 'Danh mục sản phẩm',
'Size': 'Kích thước sản phẩm',
'ASIN': 'Mã định danh sản phẩm trên Amazon',
'Courier Status': 'Trạng thái của đơn vị vận chuyển (Shipped, Delivered, Cancel)',
'Qty': 'Số lượng mua',
'Currency': 'Loại tiền tệ',
'Amount': 'Tổng số tiền của đơn hàng',
'ship-city': 'Thành phố nhận hàng',
'ship-state': 'Bang hoặc khu vực nhận hàng',
'ship-postal-code': 'Mã bưu điện nơi nhận',
'ship-country': 'Quốc gia nhận',
'promotion-ids': 'Mã khuyến mãi nếu có',
'B2B': 'Đơn hàng B2B (doanh nghiệp) hay B2C',
'fulfilled-by': 'Đơn vị hoàn tất đơn (Merchant hoặc Amazon)',
'Unnamed: 22': 'Cột rác từ Excel, không có thông tin'
}
```

Chuyển sang DataFrame để dễ đọc

```
desc_df = pd.DataFrame.from_dict(column_descriptions, orient='index', columns=['Description'])
display(desc_df)
```

	Description
Order ID	Mã đơn hàng duy nhất
Date	Ngày tạo đơn hàng
Status	Trạng thái đơn hàng (Shipped, Cancelled, Deliv...
Fulfilment	Hình thức hoàn tất đơn hàng (Merchant hoặc Ama...
Sales Channel	Kênh bán hàng (Online, Offline)
Ship Service Level	Mức độ dịch vụ vận chuyển (Standard, Express, ...
Style	Loại sản phẩm hoặc biến thể
SKU	Mã định danh sản phẩm
Category	Danh mục sản phẩm
Size	Kích thước sản phẩm
ASIN	Mã định danh sản phẩm trên Amazon
Courier Status	Trạng thái của đơn vị vận chuyển (Shipped, Del...
Qty	Số lượng mua
Currency	Loại tiền tệ
Amount	Tổng số tiền của đơn hàng
ship-city	Thành phố nhận hàng
ship-state	Bang hoặc khu vực nhận hàng
ship-postal-code	Mã bưu điện nơi nhận
ship-country	Quốc gia nhận
promotion-ids	Mã khuyến mãi nếu có
B2B	Đơn hàng B2B (doanh nghiệp) hay B2C
fulfilled-by	Đơn vị hoàn tất đơn (Merchant hoặc Amazon)
Unnamed: 22	Cột rác từ Excel, không có thông tin

✅ Tiền xử lý dữ liệu (Data Preprocessing)

1. Xoá cột dư thừa

Các cột như index, Unnamed: 22 không mang thông tin hữu ích (có thể do lỗi xuất file từ Excel).

```
In [35]: # Xoá các cột không cần thiết
df.drop(columns=['index', 'Unnamed: 22', 'Courier Status'], errors='ignore', inplace=True)
```

2. Chuyển đổi kiểu dữ liệu

- Một số cột như Date đang ở dạng chuỗi, cần chuyển sang datetime để phân tích thời gian.
- ship-postal-code đang ở dạng float, nên chuyển về string để thể hiện mã vùng rõ ràng.

```
In [36]: # Chuyển 'Date' sang kiểu datetime
df['Date'] = pd.to_datetime(df['Date'], format='%m-%d-%y', errors='coerce')

# Chuyển 'ship-postal-code' thành chuỗi (nếu có)
if 'ship-postal-code' in df.columns:
    df['ship-postal-code'] = df['ship-postal-code'].astype('Int64').astype('string')
```

3. Xử lý giá trị thiếu (Missing Values)

- Nhiều mô hình học máy không xử lý được dữ liệu thiếu (NaN).
- Loại bỏ cột có quá nhiều giá trị thiếu
- Cột phân loại → thay thế bằng giá trị phổ biến hoặc 'Unknown' hoặc 'None'
- Cột số → thay thế bằng 0 hoặc trung bình

```
In [37]: # Hiển thị tổng số giá trị thiếu
print("🔍 Thiếu dữ liệu:")
display(df.isnull().sum())

# Loại bỏ cột có hơn 60% giá trị thiếu
df = df.dropna(axis=1, thresh=len(df) * 0.4)
#df = df[df['Amount'].notnull()]
df = df[df['ship-postal-code'].notnull()]
df = df[df['ship-city'].notnull()]
df = df[df['ship-state'].notnull()]
df = df[df['ship-country'].notnull()]

# Điền giá trị thiếu có định hướng
fill_defaults = {
    'Courier Status': 'Unknown',
    'currency': 'INR',
    'Amount': 0,
    'promotion-ids': 'None'
}

for col, value in fill_defaults.items():
    if col in df.columns:
```

```
df[col].fillna(value, inplace=True)
display(df.isnull().sum())
```

🔍 Thiếu dữ liệu:

Order ID	0
Date	0
Status	0
Fulfilment	0
Sales Channel	0
ship-service-level	0
Style	0
SKU	0
Category	0
Size	0
ASIN	0
Qty	0
currency	7795
Amount	7795
ship-city	33
ship-state	33
ship-postal-code	33
ship-country	33
promotion-ids	49153
B2B	0
fulfilled-by	89698

dtype: int64

D:\Temp\ipykernel_10684\3185221319.py:23: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df[col].fillna(value, inplace=True)
```



```

Order ID      0
Date          0
Status        0
Fulfilment    0
Sales Channel 0
ship-service-level 0
Style         0
SKU           0
Category      0
Size          0
ASIN          0
Qty           0
currency      0
Amount        0
ship-city     0
ship-state    0
ship-postal-code 0
ship-country  0
promotion-ids 0
B2B           0
dtype: int64

```

4. Xử lý giá trị trùng lặp

- Trùng lặp dòng có thể làm sai lệch phân tích, đặc biệt khi tính tổng doanh thu hoặc số đơn hàng.

```

In [38]: # Kiểm tra và xóa dòng trùng lặp
print("Số dòng trùng lặp:", df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("Số dòng trùng lặp:", df.duplicated().sum())

```

Số dòng trùng lặp: 6

Số dòng trùng lặp: 0

5. Kiểm tra & mã hóa dữ liệu phân loại (Categorical Encoding - nếu dùng mô hình)

Các mô hình ML cần đầu vào số (numeric)

Các cột như Status, Fulfilment, Category, ... cần được mã hoá

```

In [39]: # Kiểm tra các cột phân loại
cat_columns = df.select_dtypes(include='object').columns
print("Các cột phân loại:\n", cat_columns.tolist())

```

Các cột phân loại:

```

['Order ID', 'Status', 'Fulfilment', 'Sales Channel ', 'ship-service-level', 'Style', 'SKU', 'Category', 'Size', 'ASIN', 'currency', 'ship-city', 'ship-state', 'ship-country', 'promotion-ids']

```



Exploratory Data Analysis (EDA) & Trực quan hóa

Đơn hàng bị hủy có đặc điểm gì?

🔍 Mục đích: So sánh giữa đơn hàng Cancelled và không bị hủy theo các yếu tố như Amount, ship-state, Courier Status.

📊 Phân phối số đơn hàng bị hủy theo bang:

```
In [40]: import matplotlib.pyplot as plt
import seaborn as sns
cancelled_by_state = df[df['Status'] == 'Cancelled']['ship-state'].value_counts().h

plt.figure(figsize=(10, 5))
sns.barplot(x=cancelled_by_state.values, y=cancelled_by_state.index, palette='Reds')
plt.title('🔴 Top 10 bang có nhiều đơn hàng bị hủy nhất')
plt.xlabel('Số đơn hàng bị hủy')
plt.ylabel('Bang')
plt.show()
```

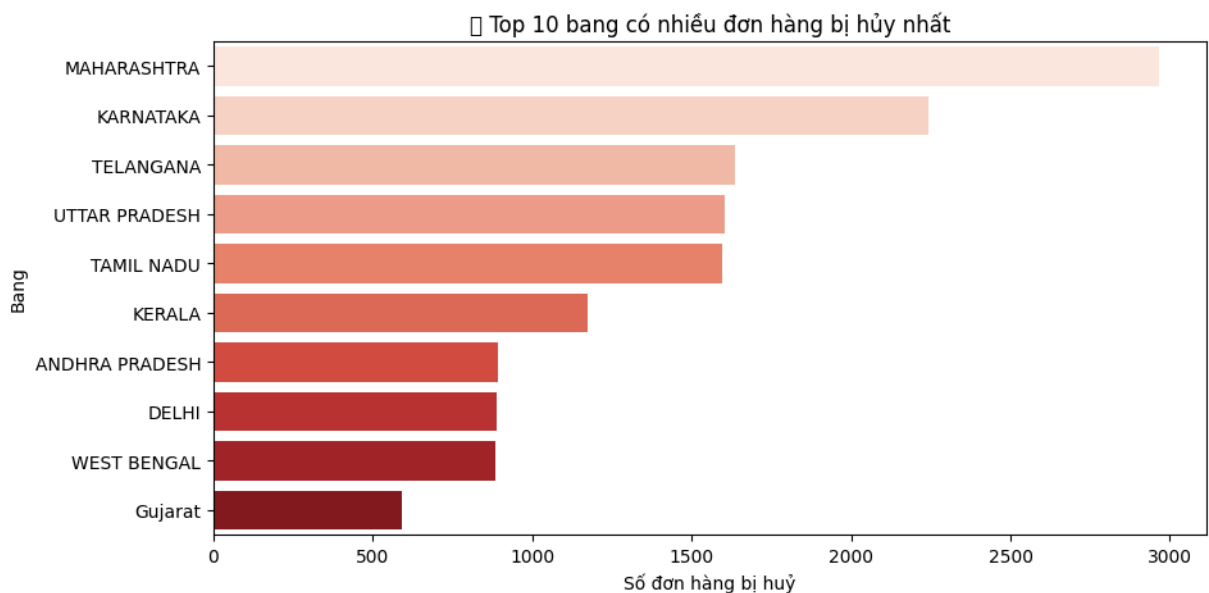
D:\Temp\ipykernel_10684\3642839011.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=cancelled_by_state.values, y=cancelled_by_state.index, palette='Reds')
```

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128721 (\N{OCTAGONAL SIGN}) missing from font(s) DejaVu Sans.

```
fig.canvas.print_figure(bytes_io, **kw)
```



Biểu đồ hiển thị "Top 10 bang có nhiều đơn hàng bị hủy nhất". Dưới đây là những nhận xét về biểu đồ này:

- Tiêu đề: Tiêu đề rõ ràng, cho biết nội dung chính của biểu đồ là "Top 10 bang có nhiều đơn hàng bị hủy nhất".
- Trục tung (Bang): Liệt kê 10 bang theo thứ tự từ bang có số lượng đơn hàng bị hủy ít nhất trong top 10 (Gujarat) đến bang có số lượng đơn hàng bị hủy nhiều nhất (Maharashtra).
- Trục hoành (Số đơn hàng bị hủy): Biểu diễn số lượng đơn hàng bị hủy, với các mốc từ 0 đến 3000.
- Màu sắc: Biểu đồ sử dụng thang màu đỏ/cam, với màu đỏ đậm hơn cho các bang có số lượng đơn hàng bị hủy ít hơn và màu nhạt dần sang cam/hồng cho các bang có số lượng đơn hàng bị hủy nhiều hơn. Điều này giúp dễ dàng phân biệt mức độ giữa các bang.

Dữ liệu nổi bật:

- Maharashtra là bang có số lượng đơn hàng bị hủy cao nhất, gần 3000 đơn hàng.
- Karnataka đứng thứ hai với số lượng đơn hàng bị hủy trên 2000.
- Các bang như Telangana, Uttar Pradesh, Tamil Nadu có số lượng đơn hàng bị hủy tương đối gần nhau, khoảng 1500-1600.
- Kerala có số lượng đơn hàng bị hủy hơn 1000.
- Các bang như Andhra Pradesh, Delhi, West Bengal có số lượng đơn hàng bị hủy dưới 1000 nhưng vẫn nằm trong top 10.
- Gujarat là bang có số lượng đơn hàng bị hủy ít nhất trong top 10, khoảng 500-600 đơn hàng.



So sánh doanh thu giữa đơn hủy và không hủy:

```
In [41]: # Nhóm trạng thái đơn hàng
def group_status(status):
    status = str(status).strip().lower()
    if 'delivered to buyer' in status:
        return 'Đã giao'
    elif 'returned' in status or 'rejected' in status or 'returning' in status:
        return 'Trả hàng'
    elif 'damaged' in status or 'lost' in status:
        return 'Lỗi / Mất hàng'
    elif 'cancelled' in status:
        return 'Đã hủy'
    elif 'shipped' in status or 'pending' in status or 'picked up' in status or 'ou
        return 'Đang giao'
    else:
        return 'Khác'
```

```
# Tạo cột nhóm
df1 = df.copy()
df1['StatusGroup'] = df1['Status'].apply(group_status)

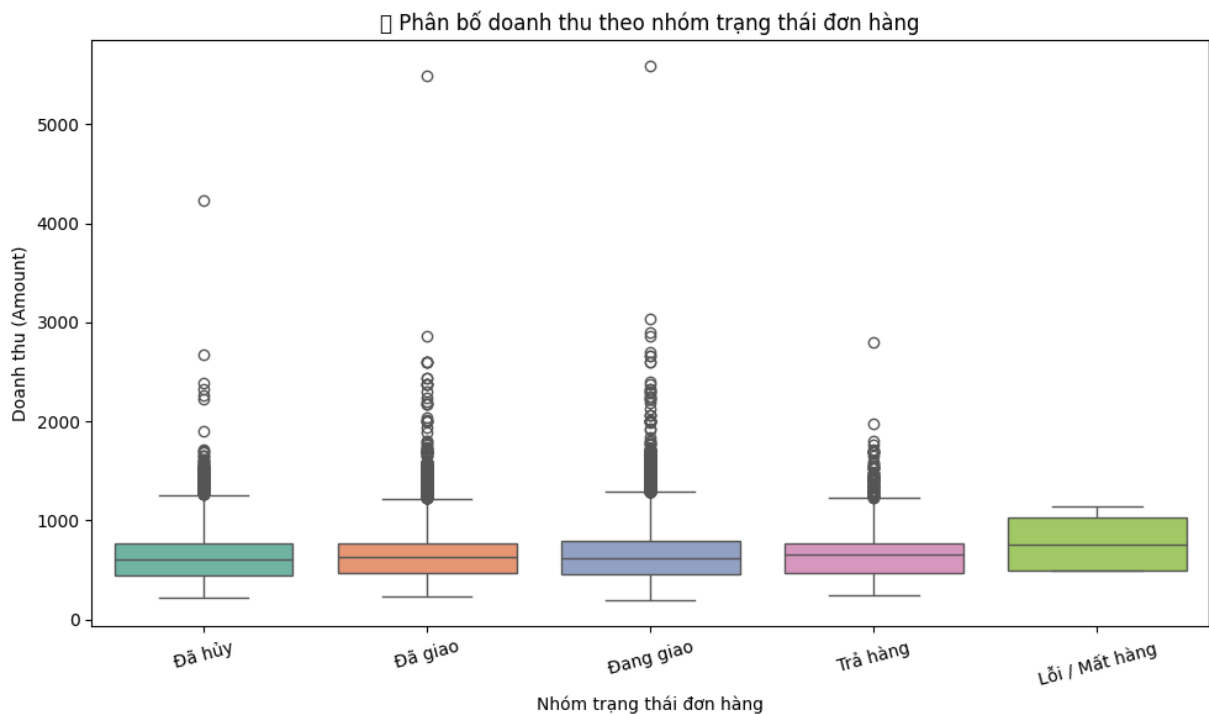
plt.figure(figsize=(10, 6))
sns.boxplot(data=df1[df1['Amount'] > 0],
            x='StatusGroup', y='Amount', palette='Set2')

plt.title('📦 Phân bố doanh thu theo nhóm trạng thái đơn hàng')
plt.xlabel('Nhóm trạng thái đơn hàng')
plt.ylabel('Doanh thu (Amount)')
plt.xticks(rotation=15)
plt.tight_layout()
plt.show()
```

D:\Temp\ipykernel_10684\4236743576.py:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.boxplot(data=df1[df1['Amount'] > 0],
D:\Temp\ipykernel_10684\4236743576.py:29: UserWarning: Glyph 128230 (\N{PACKAGE}) missing from font(s) DejaVu Sans.
plt.tight_layout()
C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128230 (\N{PACKAGE}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)
```



Nhận xét chi tiết từng nhóm trạng thái:

Đã hủy:

- Trung vị doanh thu khá thấp, khoảng 500-600.

- Phạm vi doanh thu cũng hẹp.
- Có nhiều điểm ngoại lệ (outliers) với doanh thu rất cao, thậm chí lên đến hơn 4000, cho thấy có những đơn hàng bị hủy nhưng giá trị doanh thu lại rất lớn. Đây là một điểm đáng chú ý cần được điều tra.

Đã giao:

- Trung vị doanh thu tương tự như "Đã hủy", khoảng 500-600.
- Phạm vi doanh thu rộng hơn so với "Đã hủy" (khoảng từ 200-1000).
- Có nhiều điểm ngoại lệ với doanh thu cao, nhưng không quá cao như "Đã hủy".

Đang giao:

- Trung vị doanh thu cao hơn một chút so với "Đã hủy" và "Đã giao", khoảng 600-700.
- Phạm vi doanh thu cũng tương tự "Đã giao".
- Có nhiều điểm ngoại lệ với doanh thu cao.

Trả hàng:

- Trung vị doanh thu thấp nhất trong các nhóm, khoảng 500.
- Phạm vi doanh thu cũng khá hẹp.
- Có một số điểm ngoại lệ với doanh thu cao, nhưng ít hơn các nhóm khác. Lỗi / Mất hàng:
- Đây là nhóm có phân bố doanh thu khác biệt nhất.
- Trung vị doanh thu cao nhất, khoảng 1000.
- Phạm vi doanh thu cũng cao hơn (khoảng 800-1100).
- Điều đáng chú ý là nhóm này không có ngoại lệ được hiển thị, cho thấy doanh thu của các đơn hàng trong trạng thái này có sự tập trung hơn và ít biến động lớn.

Doanh thu theo thời gian như thế nào?

```
In [42]: # Thêm cột tháng-năm
df['Month'] = df['Date'].dt.to_period('M')

revenue_by_month = df.groupby('Month')['Amount'].sum()

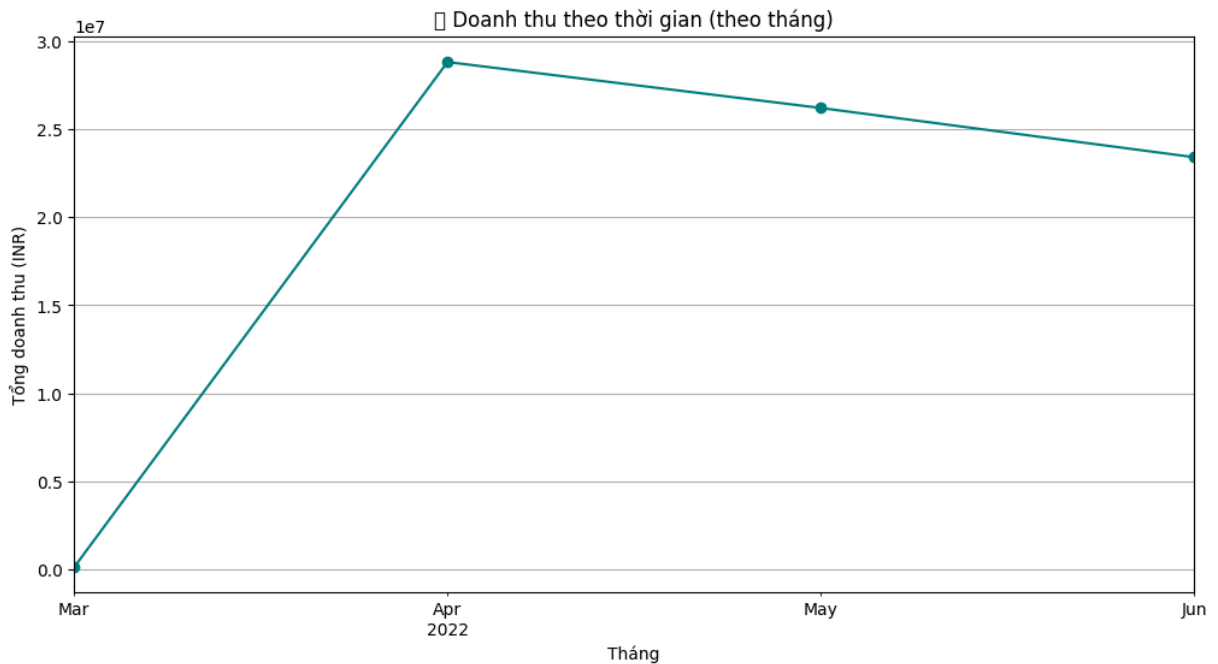
plt.figure(figsize=(12, 6))
revenue_by_month.plot(marker='o', color='teal')
plt.title('📅 Doanh thu theo thời gian (theo tháng)')
```

```
plt.xlabel('Tháng')
plt.ylabel('Tổng doanh thu (INR)')
plt.grid(True)
plt.show()
```

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128197 (\N{CALENDAR}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128197 (\N{CALENDAR}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128197 (\N{CALENDAR}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)



Xu hướng doanh thu:

- Tháng 3 (Mar): Doanh thu rất thấp, gần như bằng 0. Có thể là điểm khởi đầu của dữ liệu hoặc tháng đầu tiên hoạt động.
- Tháng 4 (Apr): Doanh thu tăng vọt một cách ấn tượng, đạt đỉnh điểm ở mức gần 29,000,000 INR (29 triệu INR). Đây là mức doanh thu cao nhất trong giai đoạn được biểu đồ thể hiện.
- Tháng 5 (May): Doanh thu có sự sụt giảm nhẹ so với tháng 4, xuống còn khoảng 26,000,000 INR (26 triệu INR). Mặc dù giảm, đây vẫn là một mức doanh thu rất cao.
- Tháng 6 (Jun): Doanh thu tiếp tục giảm so với tháng 5, xuống còn khoảng 23,000,000 INR (23 triệu INR).

Kết luận chung:

- Biểu đồ cho thấy một sự tăng trưởng vượt bậc về doanh thu từ tháng 3 lên tháng 4 năm 2022.
- Sau khi đạt đỉnh vào tháng 4, doanh thu có xu hướng giảm dần trong tháng 5 và tháng 6.
- Mặc dù có sự sụt giảm trong hai tháng gần đây nhất, mức doanh thu trong tháng 5 và tháng 6 vẫn rất đáng kể so với tháng 3.
- Cần phân tích thêm để hiểu lý do của sự tăng trưởng mạnh trong tháng 4 và sự sụt giảm nhẹ trong các tháng tiếp theo (ví dụ: các chiến dịch marketing, thay đổi chính sách, yếu tố mùa vụ, hoặc sự cạnh tranh).

Xử lý dữ liệu đầu vào cho mô hình ML (Hồi quy & Phân loại)

✓ 1. Chọn biến mục tiêu (target) và loại mô hình

Xác định mục tiêu và bài toán

- **Mô hình phân loại:** Dự đoán đơn hàng có bị **hủy (Status = "Cancelled")** hay không. Chuyển `Status` thành biến nhị phân 0/1.
- **Mô hình hồi quy:** Dự đoán **Amount** (doanh thu đơn hàng) dựa trên các đặc trưng như sản phẩm, khu vực, dịch vụ vận chuyển, v.v.

```
In [43]: # Phân Loại: Biến mục tiêu nhị phân
df['Cancelled'] = df['Status'].apply(lambda x: 1 if x == 'Cancelled' else 0)

# Hồi quy: Biến mục tiêu là Amount (đã làm sạch trước đó)
```

✓ 2. Lọc dữ liệu không hợp lệ

Lọc dữ liệu

- Bỏ các đơn hàng có `Amount <= 0` (lỗi nhập hoặc đơn không có giá trị).
- Chỉ giữ các dòng có `Qty > 0`.

```
In [44]: df_model = df[(df['Amount'] > 0) & (df['Qty'] > 0)].copy()
```

✓ 3. Mã hoá biến phân loại (Categorical Encoding)

Mã hoá biến phân loại

- Sử dụng **Label Encoding** để biến các cột phân loại thành dạng số.
- Giúp mô hình ML hiểu và xử lý các biến phân loại.

```
In [45]: from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Tách các cột dạng chuỗi
cat_cols = df_model.select_dtypes(include='object').columns.tolist()

# Khởi tạo bộ Lưu encoder
label_encoders = {}
df_encoded = df_model.copy()

# Duyệt từng cột phân loại
for col in cat_cols:
    n_unique = df_model[col].nunique()

    if n_unique > 5:
        # Dùng Label Encoding
        le = LabelEncoder()
        df_encoded[col] = le.fit_transform(df_model[col].astype(str))
        label_encoders[col] = le # Lưu encoder nếu muốn inverse sau
    else:
        # Dùng One-Hot Encoding
        dummies = pd.get_dummies(df_model[col], prefix=col)
        df_encoded = pd.concat([df_encoded.drop(columns=[col]), dummies], axis=1)

# Kết quả
display(df_encoded.head())
df_model = df_encoded.copy()
```

	Order ID	Date	Status	Style	SKU	Category	Size	ASIN	Qty	Amount	...	B2B	Montl
1	12248	2022-04-30	5	844	4378	8	0	5217	1	406.0	...	False	20220
2	40793	2022-04-30	3	530	2588	8	8	666	1	329.0	...	True	20220
4	81106	2022-04-30	3	751	3781	6	0	4297	1	574.0	...	False	20220
5	41815	2022-04-30	3	1229	6199	5	8	2894	1	824.0	...	False	20220
6	100431	2022-04-30	3	180	647	5	5	1653	1	653.0	...	False	20220

5 rows × 24 columns



4. Lựa chọn đặc trưng (Feature Selection)


```
In [46]: # Loại bỏ các cột không phù hợp
cols_to_drop = [
    'order-id', 'customer-name', 'ship-address', 'ship-city',
    'ship-phone-number', 'Date', 'Status', 'order-item-id', 'currency', 'Month', 'Co
]
df_model = df_model.drop(columns=[col for col in cols_to_drop if col in df.columns])
df_model.head()
```

```
Out[46]:
```

	Order ID	Style	SKU	Category	Size	ASIN	Qty	Amount	ship-state	ship-postal-code	promotion-ids
1	12248	844	4378	8	0	5217	1	406.0	22	560085	1999
2	40793	530	2588	8	8	666	1	329.0	27	410210	5705
4	81106	751	3781	6	0	4297	1	574.0	54	600073	5716
5	41815	1229	6199	5	8	2894	1	824.0	57	201102	5705
6	100431	180	647	5	5	1653	1	653.0	9	160036	5705

✓ 5. Thực hiện xử lý mất cân bằng dữ liệu và chọn biến mục tiêu

```
In [47]: from imblearn.over_sampling import SMOTE

X = df_model.drop(columns=['Amount', 'Cancelled'], errors='ignore')
y_class = df_model['Cancelled']
y_reg = df_model['Amount']
print("Trước SMOTE:")
print(y_class.value_counts())
# Cân bằng dữ liệu bằng SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_class = smote.fit_resample(X, y_class)
print("Sau SMOTE:")
print(y_class.value_counts())
```

Trước SMOTE:

Cancelled

0 108042

1 5628

Name: count, dtype: int64

Sau SMOTE:

Cancelled

0 108042

1 108042

Name: count, dtype: int64

✓ 6. Chuẩn hoá dữ liệu số (Feature Scaling)



Chuẩn hoá dữ liệu số

- Các mô hình như hồi quy tuyến tính hoặc logistic thường yêu cầu dữ liệu ở cùng một thang đo.
- Dùng StandardScaler để chuẩn hoá các biến số.

In [48]: `print(X.head())`

	Order ID	Style	SKU	Category	Size	ASIN	Qty	ship-state	\
1	12248	844	4378	8	0	5217	1	22	
2	40793	530	2588	8	8	666	1	27	
4	81106	751	3781	6	0	4297	1	54	
5	41815	1229	6199	5	8	2894	1	57	
6	100431	180	647	5	5	1653	1	9	

	ship-postal-code	promotion-ids	B2B	Fulfilment_Amazon	\
1	560085	1999	False	False	
2	410210	5705	True	True	
4	600073	5716	False	True	
5	201102	5705	False	True	
6	160036	5705	False	True	

	Fulfilment_Merchant	Sales Channel	_Amazon.in	\
1	True		True	
2	False		True	
4	False		True	
5	False		True	
6	False		True	

	ship-service-level_Expedited	ship-service-level_Standard	currency_INR	\
1	False	True	True	
2	True	False	True	
4	True	False	True	
5	True	False	True	
6	True	False	True	

	ship-country_IN
1	True
2	True
4	True
5	True
6	True

In [49]: `from sklearn.preprocessing import StandardScaler`

```

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_resampled)
X_scaled_reg = pd.DataFrame(X)
display(X_scaled[:5, :])

```

```
array([[ -1.32884207e+00,  2.52709948e-01,  3.03807214e-01,
         1.17459245e+00, -2.39925060e+00,  8.42732016e-01,
        -3.88856778e-02, -5.84570922e-01,  4.86319793e-01,
        -2.87612493e+00, -9.36085045e-02, -2.41715713e+00,
         2.41715713e+00,  0.00000000e+00, -2.41442672e+00,
         2.40863257e+00,  0.00000000e+00,  0.00000000e+00],
       [-3.98739163e-01, -6.10276435e-01, -6.16532139e-01,
         1.17459245e+00,  6.76967594e-01, -1.50743979e+00,
        -3.88856778e-02, -3.03130762e-01, -3.05283744e-01,
         3.37984891e-01,  1.06827900e+01,  4.13709141e-01,
        -4.13709141e-01,  0.00000000e+00,  4.14176994e-01,
        -4.15173328e-01,  0.00000000e+00,  0.00000000e+00],
       [ 9.14809220e-01, -2.88792945e-03, -3.14395568e-03,
        -2.28850956e-01, -2.39925060e+00,  3.67636858e-01,
        -3.88856778e-02,  1.21664611e+00,  6.97526747e-01,
         3.47524883e-01, -9.36085045e-02,  4.13709141e-01,
        -4.13709141e-01,  0.00000000e+00,  4.14176994e-01,
        -4.15173328e-01,  0.00000000e+00,  0.00000000e+00],
       [-3.65438579e-01,  1.31083020e+00,  1.24008540e+00,
        -9.30572659e-01,  6.76967594e-01, -3.56883257e-01,
        -3.88856778e-02,  1.38551020e+00, -1.40974167e+00,
         3.37984891e-01, -9.36085045e-02,  4.13709141e-01,
        -4.13709141e-01,  0.00000000e+00,  4.14176994e-01,
        -4.15173328e-01,  0.00000000e+00,  0.00000000e+00],
       [ 1.54449003e+00, -1.57220393e+00, -1.61450906e+00,
        -9.30572659e-01, -4.76614228e-01, -9.97745312e-01,
        -3.88856778e-02, -1.31631534e+00, -1.62664236e+00,
         3.37984891e-01, -9.36085045e-02,  4.13709141e-01,
        -4.13709141e-01,  0.00000000e+00,  4.14176994e-01,
        -4.15173328e-01,  0.00000000e+00,  0.00000000e+00]])
```

✓ 7. Chia tập huấn luyện và kiểm tra

```
In [50]: from sklearn.model_selection import train_test_split

# Cho mô hình phân loại
X_train_cls, X_test_cls, y_train_cls, y_test_cls = train_test_split(
    X_scaled, y_class, test_size=0.2, random_state=42)

# Cho mô hình hồi quy
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    X_scaled_reg, y_reg, test_size=0.2, random_state=42)
```

🤖 Mô hình phân loại: Logistic Regression

- Phù hợp cho bài toán nhị phân (Cancelled: 1 hoặc 0)
- Cho xác suất dự đoán → dễ diễn giải

Tham số quan trọng:

- `penalty='l2'` : chuẩn hóa L2 (mặc định) giúp giảm overfitting.
- `C=1.0` : hệ số điều chỉnh phạt regularization. Càng nhỏ thì regularization càng mạnh.
- `solver='liblinear'` : phù hợp với dữ liệu nhỏ và ít đặc trưng.

- `random_state=42` : để kết quả có thể tái lập được.

```
In [51]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Huấn luyện Logistic Regression
logreg = LogisticRegression(penalty='l2', C=1.0, solver='liblinear', random_state=42)
logreg.fit(X_train_cls, y_train_cls)

# Dự đoán và đánh giá
y_pred_cls = logreg.predict(X_test_cls)
print("🔍 Classification Report:")
print(classification_report(y_test_cls, y_pred_cls))
print("📊 Confusion Matrix:")
print(confusion_matrix(y_test_cls, y_pred_cls))
```

🔍 Classification Report:

	precision	recall	f1-score	support
0	0.99	0.72	0.83	21526
1	0.78	1.00	0.88	21691
accuracy			0.86	43217
macro avg	0.89	0.86	0.85	43217
weighted avg	0.89	0.86	0.85	43217


📊 Confusion Matrix:

```
[[15477  6049]
 [  102 21589]]
```


```
In [52]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Huấn luyện Logistic Regression
logreg = LogisticRegression(penalty='l1', C=1.0, solver='liblinear', random_state=42)
logreg.fit(X_train_cls, y_train_cls)

# Dự đoán và đánh giá
y_pred_cls = logreg.predict(X_test_cls)
print("🔍 Classification Report:")
print(classification_report(y_test_cls, y_pred_cls))
print("📊 Confusion Matrix:")
print(confusion_matrix(y_test_cls, y_pred_cls))
```

 Classification Report:

	precision	recall	f1-score	support
0	0.99	0.73	0.84	21526
1	0.79	0.99	0.88	21691
accuracy			0.86	43217
macro avg	0.89	0.86	0.86	43217
weighted avg	0.89	0.86	0.86	43217

 Confusion Matrix:

```
[[15624  5902]
 [  127 21564]]
```

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\sklearn\svm_base.py:1249: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
warnings.warn(

Mô hình hồi quy: Linear Regression

- Dự đoán biến liên tục: `Amount`
- Dễ hiểu, nhanh huấn luyện, dễ diễn giải hệ số.

Không cần nhiều tham số: chỉ cần mặc định.

```
In [53]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Huấn luyện mô hình hồi quy tuyến tính
linreg = LinearRegression()
linreg.fit(X_train_reg, y_train_reg)

# Dự đoán và đánh giá
y_pred_reg = linreg.predict(X_test_reg)

# Đánh giá kết quả
mse = mean_squared_error(y_test_reg, y_pred_reg)
r2 = r2_score(y_test_reg, y_pred_reg)

print(f"❌ MSE: {mse:.2f}")
print(f"❌ R² Score: {r2:.2f}")
```

❌ MSE: 47396.07
❌ R² Score: 0.34

Mô hình cây quyết định

Tham số được chọn:

- `min_samples_split=10` : Một nút chỉ được chia nếu có ít nhất 10 mẫu → kiểm soát số lượng nhánh, tránh phân tách quá nhỏ.
- `max_depth=25` : Giới hạn độ sâu của mỗi cây để tránh overfitting. Độ sâu nhỏ hơn giúp mô hình tổng quát hơn.

- `random_state=42` : Đảm bảo tính lặp lại.

In [54]: `from sklearn.tree import DecisionTreeClassifier`
`from sklearn.metrics import classification_report, confusion_matrix`

```
# Mô hình cây quyết định
dt_cls = DecisionTreeClassifier(
    max_depth=25,
    min_samples_split=10,
    random_state=42
)

# Huấn luyện mô hình
dt_cls.fit(X_train_cls, y_train_cls)

# Dự đoán
y_pred_dt_cls = dt_cls.predict(X_test_cls)

# Đánh giá
print("🌳 Decision Tree Classifier Report:")
print(classification_report(y_test_cls, y_pred_dt_cls))
print("📊 Confusion Matrix:")
print(confusion_matrix(y_test_cls, y_pred_dt_cls))
```

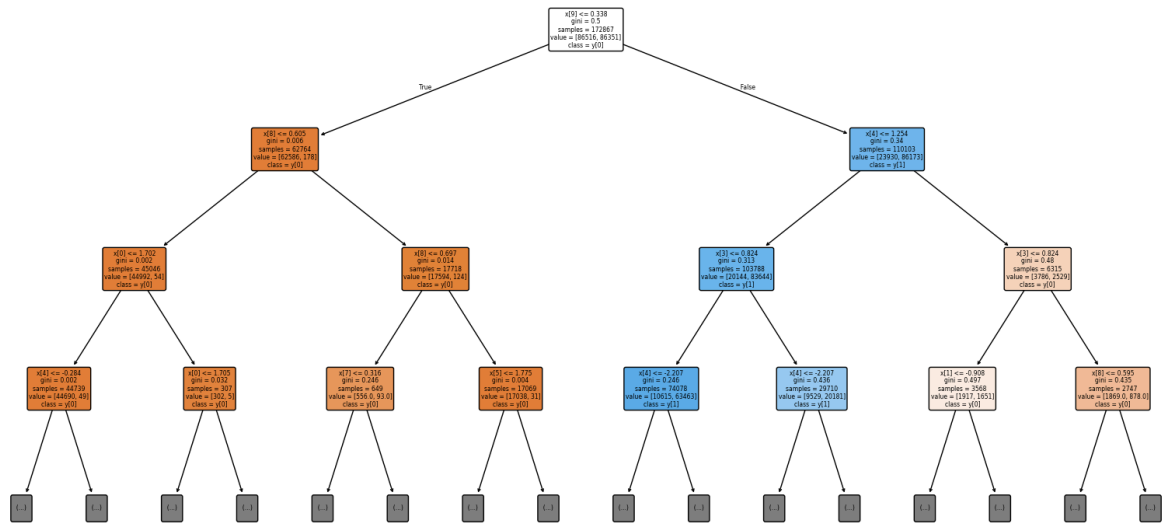
🌳 Decision Tree Classifier Report:

	precision	recall	f1-score	support
0	0.94	0.94	0.94	21526
1	0.94	0.95	0.94	21691
accuracy			0.94	43217
macro avg	0.94	0.94	0.94	43217
weighted avg	0.94	0.94	0.94	43217

📊 Confusion Matrix:
[[20200 1326]
[1181 20510]]

In [55]: `from sklearn.tree import plot_tree`
`import matplotlib.pyplot as plt`

```
plt.figure(figsize=(20,10)) # Kích thước hình lớn để nhìn rõ
plot_tree(
    dt_cls,
    filled=True,           # tô màu các nút theo lớp
    rounded=True,         # bo tròn các ô
    class_names=True,     # hiện tên lớp (nếu có)
    feature_names=X_train_cls.columns if hasattr(X_train_cls, 'columns') else None,
    max_depth=3           # chỉ vẽ sâu tối đa 3 tầng để dễ nhìn, bạn có thể bỏ hoặc
)
plt.show()
```



```
In [56]: y_train_pred_rf_cls = dt_cls.predict(X_train_cls)
print("Train Accuracy:", dt_cls.score(X_train_cls, y_train_cls))
print("Test Accuracy:", dt_cls.score(X_test_cls, y_test_cls))
```

Train Accuracy: 0.9750501830887329

Test Accuracy: 0.9419904204364024

Mô hình phân loại: RandomForestClassifier

Mô hình rừng ngẫu nhiên là một tập hợp nhiều cây quyết định. Ưu điểm:

- Xử lý tốt dữ liệu phức tạp, không tuyến tính
- Kháng nhiễu và overfitting tốt hơn so với cây đơn

Tham số được chọn:

- `n_estimators=100`: Sử dụng 100 cây quyết định trong rừng → giá trị mặc định ổn định cho nhiều bài toán.
- `min_samples_split=10`: Một nút chỉ được chia nếu có ít nhất 10 mẫu → kiểm soát số lượng nhánh, tránh phân tách quá nhỏ.
- `max_depth=25`: Giới hạn độ sâu của mỗi cây để tránh overfitting. Độ sâu nhỏ hơn giúp mô hình tổng quát hơn.
- `random_state=42`: Đảm bảo tính lặp lại.

```
In [57]: from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Mô hình phân loại
rf_cls = RandomForestClassifier(
    max_depth=25,
    n_estimators=100,
    min_samples_split=10,
    random_state=42
```

```
)

rf_cls.fit(X_train_cls, y_train_cls)
y_pred_rf_cls = rf_cls.predict(X_test_cls)

# Đánh giá
print(" 🌲 Random Forest Classifier Report:")
print(classification_report(y_test_cls, y_pred_rf_cls))
print(" 📊 Confusion Matrix:")
print(confusion_matrix(y_test_cls, y_pred_rf_cls))
```

```
🌲 Random Forest Classifier Report:
              precision    recall  f1-score   support

     0       0.97       0.92       0.94       21526
     1       0.92       0.97       0.95       21691

 accuracy          0.95          0.95          0.95          43217
 macro avg         0.95          0.95          0.95          43217
weighted avg         0.95          0.95          0.95          43217
```

```
📊 Confusion Matrix:
[[19755  1771]
 [  546 21145]]
```

```
In [58]: y_train_pred_rf_cls = rf_cls.predict(X_train_cls)
print("Train Accuracy:", rf_cls.score(X_train_cls, y_train_cls))
print("Test Accuracy:", rf_cls.score(X_test_cls, y_test_cls))
```

```
Train Accuracy: 0.9695372743207206
Test Accuracy: 0.9463868385126224
```

🔍 Mô hình hồi quy: RandomForestRegressor

Rừng ngẫu nhiên hồi quy cũng là một tổ hợp nhiều cây quyết định, nhưng trả về trung bình kết quả dự đoán.

Tham số được chọn:

- `n_estimators=100` : Dùng 100 cây quyết định, tăng độ chính xác.
- `max_depth=15` : Hạn chế độ sâu để giảm overfitting.
- `min_samples_split=10` : Giúp cây không chia khi số lượng mẫu nhỏ.
- `random_state=42` : Đảm bảo kết quả có thể tái lập.

```
In [27]: from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Mô hình hồi quy
rf_reg = RandomForestRegressor(
    n_estimators=100,
    max_depth=25,
    min_samples_split=10,
    random_state=42
)
```



```

rf_reg.fit(X_train_reg, y_train_reg)
y_pred_rf_reg = rf_reg.predict(X_test_reg)

# Đánh giá
mse_rf = mean_squared_error(y_test_reg, y_pred_rf_reg)
r2_rf = r2_score(y_test_reg, y_pred_rf_reg)

print(f" 🌲 Random Forest Regressor - MSE: {mse_rf:.2f}")
print(f" 🌲 Random Forest Regressor - R² Score: {r2_rf:.2f}")

```

🌲 Random Forest Regressor - MSE: 3084.55
 🌲 Random Forest Regressor - R² Score: 0.96

🎯 ROC Curve và AUC Score

- **ROC (Receiver Operating Characteristic)** thể hiện sự đánh đổi giữa TPR (True Positive Rate) và FPR (False Positive Rate).
- **AUC (Area Under Curve)** là diện tích dưới đường ROC, đánh giá độ phân biệt của mô hình.
 - AUC ~ 1: mô hình tốt
 - AUC ~ 0.5: mô hình ngẫu nhiên

```

In [28]: from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

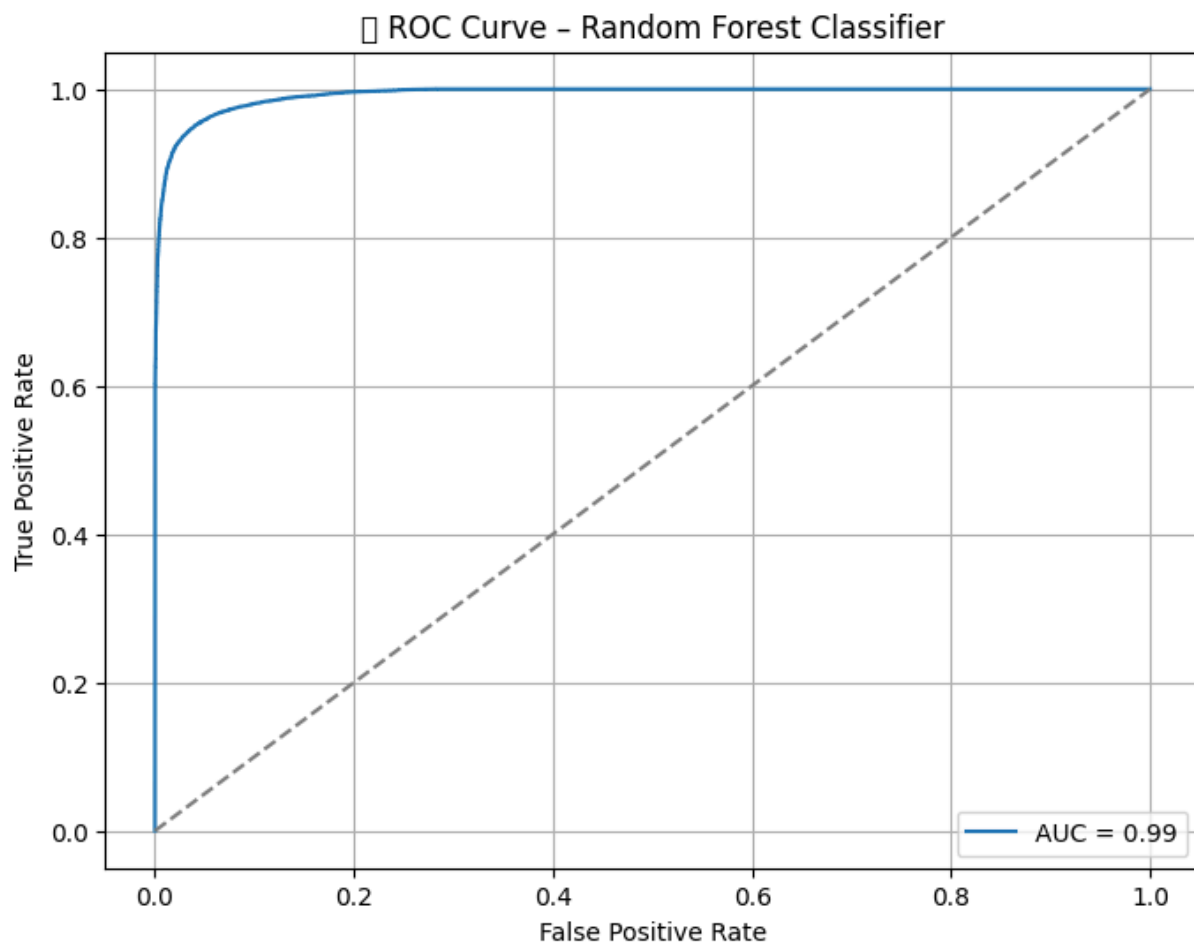
# Dự đoán xác suất
y_proba_rf = rf_cls.predict_proba(X_test_cls)[: , 1]

# Tính ROC
fpr, tpr, thresholds = roc_curve(y_test_cls, y_proba_rf)
auc_score = roc_auc_score(y_test_cls, y_proba_rf)

# Vẽ ROC
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0,1], [0,1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(' 🌲 ROC Curve - Random Forest Classifier')
plt.legend()
plt.grid(True)
plt.show()

```

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtool
 s.py:170: UserWarning: Glyph 127795 (\N{DECIDUOUS TREE}) missing from font(s) DejaVu
 Sans.
 fig.canvas.print_figure(bytes_io, **kw)



🔍 Feature Importance

Random Forest cung cấp độ quan trọng của từng thuộc tính đầu vào:

- Đặc trưng càng quan trọng → càng ảnh hưởng đến dự đoán.
- Giúp chọn đặc trưng tốt, loại bỏ nhiễu.

```
In [29]: import pandas as pd
import numpy as np

# Feature names
feature_names = X.columns

# Phân Loại - Feature Importance
importances_cls = rf_cls.feature_importances_
indices_cls = np.argsort(importances_cls)[::-1]

plt.figure(figsize=(10,6))
plt.title("🌿 Feature Importance - Phân loại")
plt.bar(range(len(importances_cls)), importances_cls[indices_cls], align='center')
plt.xticks(range(len(importances_cls)), feature_names[indices_cls], rotation=90)
plt.tight_layout()
plt.show()

# Hồi quy - Feature Importance
```

```

importances_reg = rf_reg.feature_importances_
indices_reg = np.argsort(importances_reg)[::-1]

plt.figure(figsize=(10,6))
plt.title(" 🌲 Feature Importance - Hồi quy")
plt.bar(range(len(importances_reg)), importances_reg[indices_reg], align='center')
plt.xticks(range(len(importances_reg)), feature_names[indices_reg], rotation=90)
plt.tight_layout()
plt.show()

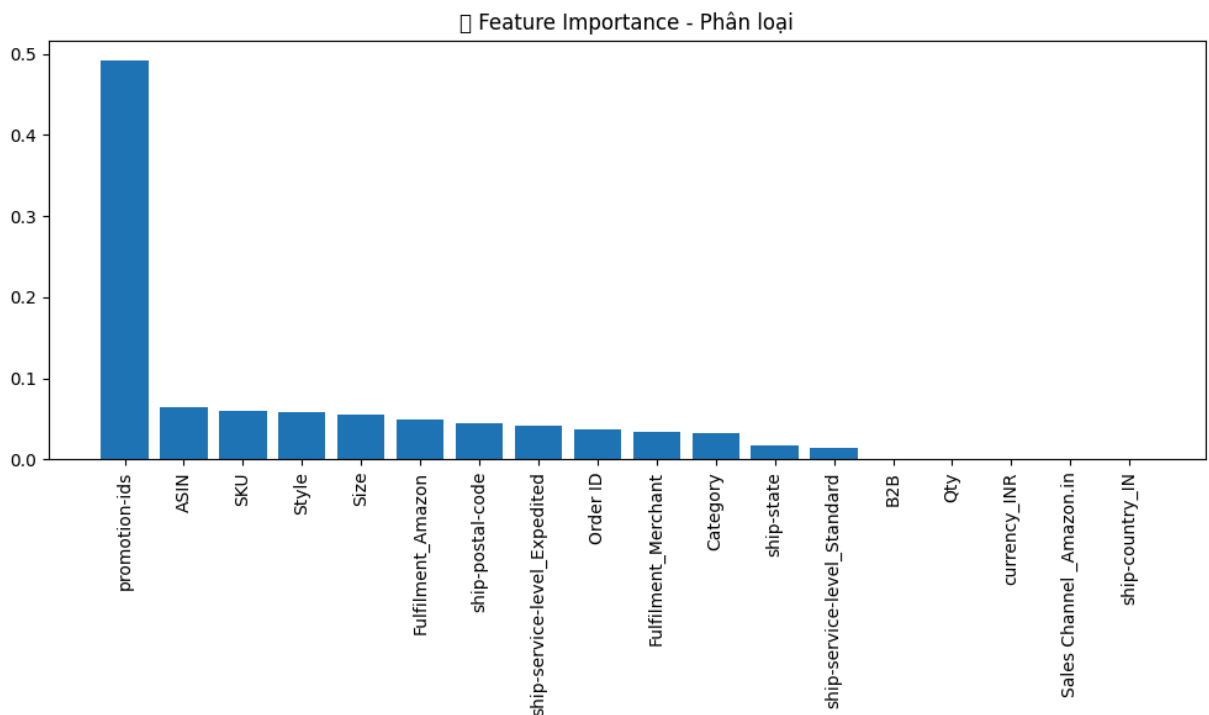
```

D:\Temp\ipykernel_10684\99954139.py:15: UserWarning: Glyph 127795 (\N{DECIDUOUS TREE}) missing from font(s) DejaVu Sans.

```

plt.tight_layout()
C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127795 (\N{DECIDUOUS TREE}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)

```

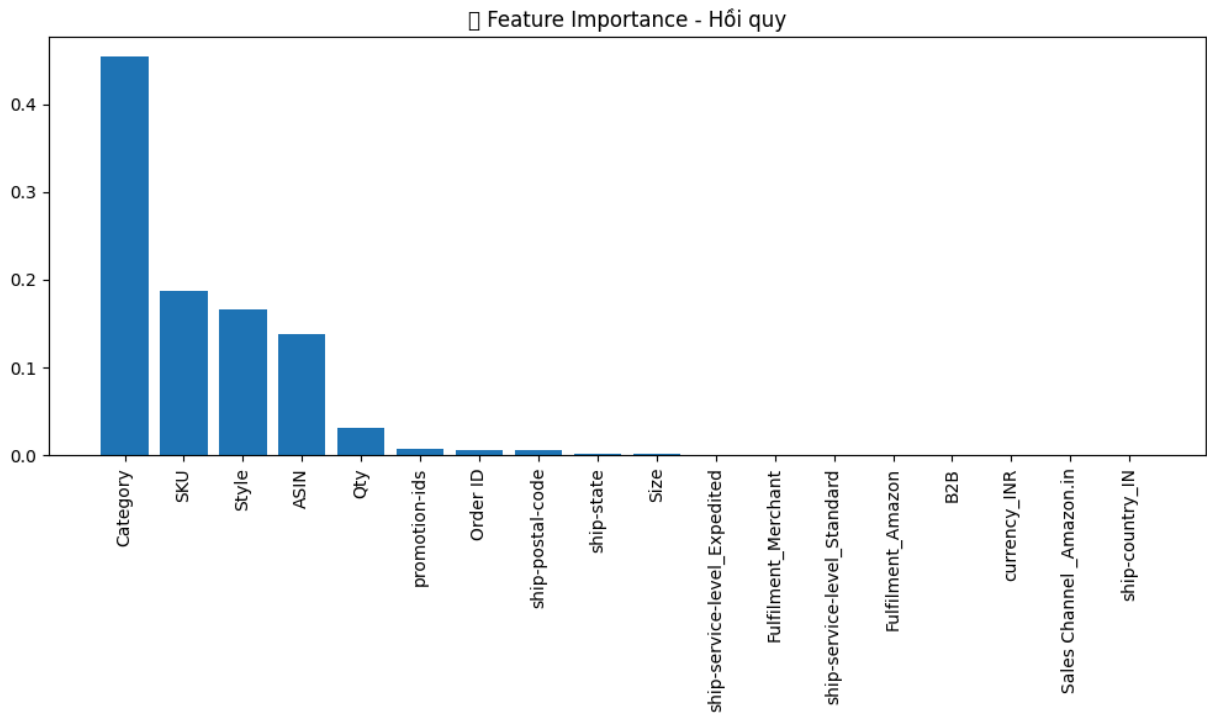


D:\Temp\ipykernel_10684\99954139.py:26: UserWarning: Glyph 127794 (\N{EVERGREEN TREE}) missing from font(s) DejaVu Sans.

```

plt.tight_layout()
C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127794 (\N{EVERGREEN TREE}) missing from font(s) DejaVu Sans.
fig.canvas.print_figure(bytes_io, **kw)

```



Phân cụm dữ liệu (Data Clustering)

Phần này áp dụng thuật toán học không giám sát K-Means để khám phá các nhóm tự nhiên (cụm) trong dữ liệu đơn hàng. Mục tiêu là hiểu rõ hơn về các phân khúc đơn hàng dựa trên đặc điểm của chúng.

```
In [60]: # Chuẩn bị thư viện cần thiết
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA # Để trực quan hóa
from sklearn.metrics import silhouette_score # Tạm thời giữ lại, có thể bỏ sau
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

# Sử dụng df_model đã được mã hóa và tiền xử lý ở các bước trên
df_for_clustering_input = df_model.copy()

cols_to_drop_for_clustering = [
    'Amount', 'Cancelled', 'Order ID', 'SKU', 'ASIN', 'ship-postal-code'
]
X_cluster = df_for_clustering_input.drop(columns=[col for col in cols_to_drop_for_c

print("Các cột được sử dụng cho phân cụm (trước khi chuẩn hóa):")
print(X_cluster.columns.tolist())

for col in X_cluster.columns:
    if X_cluster[col].dtype == 'bool':
        X_cluster[col] = X_cluster[col].astype(int)
    elif X_cluster[col].nunique() == 2 and X_cluster[col].min() == 0 and X_cluster[
```

```

        if X_cluster[col].dtype not in ['int64', 'int32']:
            X_cluster[col] = X_cluster[col].astype(int)

cols_to_scale = [col for col in X_cluster.columns
                  if (X_cluster[col].dtype in ['int64', 'int32', 'float64']) and (

print("\nCác cột sẽ được StandardScaler áp dụng:")
print(cols_to_scale)

X_cluster_scaled = X_cluster.copy()
if cols_to_scale:
    scaler_cluster = StandardScaler()
    X_cluster_scaled[cols_to_scale] = scaler_cluster.fit_transform(X_cluster[cols_to_scale])
else:
    print("Không có cột nào cần áp dụng StandardScaler.")

print("\nKích thước dữ liệu đầu vào cho phân cụm (đã chuẩn hóa):", X_cluster_scaled.shape)
print("\n5 dòng đầu của dữ liệu đã chuẩn hóa:")
display(X_cluster_scaled.head())

```

Các cột được sử dụng cho phân cụm (trước khi chuẩn hóa):

```
['Style', 'Category', 'Size', 'Qty', 'ship-state', 'promotion-ids', 'B2B', 'Fulfilment_Amazon', 'Fulfilment_Merchant', 'Sales_Channel_Amazon.in', 'ship-service-level_Express', 'ship-service-level_Standard', 'currency_INR', 'ship-country_IN']
```

Các cột sẽ được StandardScaler áp dụng:

```
['Style', 'Category', 'Size', 'Qty', 'ship-state', 'promotion-ids']
```

Kích thước dữ liệu đầu vào cho phân cụm (đã chuẩn hóa): (113670, 14)

5 dòng đầu của dữ liệu đã chuẩn hóa:

	Style	Category	Size	Qty	ship-state	promotion-ids	B2B	Fulfilment_Amazon
1	0.227321	1.043055	-2.206075	-0.052501	-0.590945	-1.969148	0	
2	-0.602453	1.043055	0.573685	-0.052501	-0.311753	0.500187	1	
4	-0.018440	-0.273468	-2.206075	-0.052501	1.195881	0.507516	0	
5	1.244719	-0.931729	0.573685	-0.052501	1.363396	0.500187	0	
6	-1.527360	-0.931729	-0.468725	-0.052501	-1.316843	0.500187	0	

1. Phân cụm bằng K-Means

K-Means là một thuật toán phân cụm phổ biến nhằm mục đích phân chia N quan sát vào K cụm sao cho mỗi quan sát thuộc về cụm có giá trị trung bình (tâm cụm) gần nhất.

1.1. Xác định số cụm (k) tối ưu

Để xác định số lượng cụm (k) phù hợp cho thuật toán K-Means, phương pháp Elbow thường được sử dụng. Phương pháp này dựa trên việc tính toán tổng bình phương sai số trong từng

cụm (WCSS - Within-Cluster Sum of Squares) cho các giá trị k khác nhau. Điểm "khuyết tật" trên biểu đồ WCSS theo k thường được xem là một chỉ báo cho số cụm tối ưu, nơi việc tăng thêm số cụm không còn mang lại sự cải thiện đáng kể trong việc giảm WCSS.

(Tùy chọn: Có thể bỏ đoạn Silhouette Score nếu muốn đơn giản) Bên cạnh đó, Silhouette Score cũng là một thước đo hữu ích để đánh giá chất lượng của các cụm. Silhouette Score đo lường mức độ tương đồng của một đối tượng với cụm của chính nó so với các cụm khác. Giá trị Silhouette dao động từ -1 đến 1, với giá trị cao hơn cho thấy các cụm được phân tách tốt và các đối tượng được gán đúng cụm.

```
In [61]: # Elbow Method
wcss = []
k_range = range(2, 11) # Thử từ 2 đến 10 cụm
for i in k_range:
    kmeans_elbow = KMeans(n_clusters=i, init='k-means++', random_state=42, n_init='
    kmeans_elbow.fit(X_cluster_scaled) # Sử dụng dữ liệu đã chuẩn hóa
    wcss.append(kmeans_elbow.inertia_)

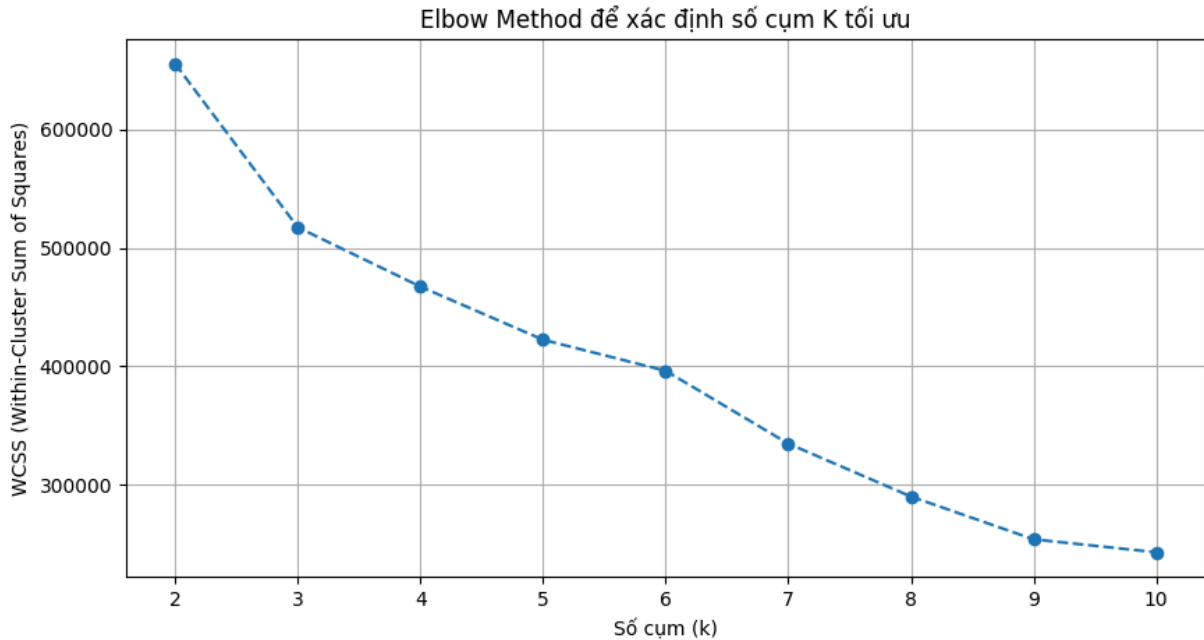
plt.figure(figsize=(10, 5))
plt.plot(k_range, wcss, marker='o', linestyle='--')
plt.title('Elbow Method để xác định số cụm K tối ưu')
plt.xlabel('Số cụm (k)')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
plt.grid(True)
plt.show()

# --- PHẦN SILHOUETTE SCORE (TÙY CHỌN - CÓ THỂ BỎ ĐI) ---
# silhouette_scores = []
# sample_size_silhouette = min(10000, X_cluster_scaled.shape[0])
# if X_cluster_scaled.shape[0] > sample_size_silhouette:
#     X_sample_silhouette = X_cluster_scaled.sample(sample_size_silhouette, random_
#     print(f"Tính Silhouette Score trên mẫu con {sample_size_silhouette} điểm.")
# else:
#     X_sample_silhouette = X_cluster_scaled
#     print("Tính Silhouette Score trên toàn bộ dữ liệu.")

# for i in k_range:
#     kmeans_silhouette = KMeans(n_clusters=i, init='k-means++', random_state=42, n
#     cluster_labels_silhouette = kmeans_silhouette.fit_predict(X_sample_silhouette)
#     if len(np.unique(cluster_labels_silhouette)) > 1:
#         silhouette_avg = silhouette_score(X_sample_silhouette, cluster_labels_silho
#         silhouette_scores.append(silhouette_avg)
#         print(f"Với k={i}, Silhouette Score là {silhouette_avg:.4f}")
#     else:
#         silhouette_scores.append(-1)
#         print(f"Với k={i}, không thể tính Silhouette Score (chỉ tạo ra 1 cụm).")

# plt.figure(figsize=(10, 5))
# plt.plot(k_range, silhouette_scores, marker='o', linestyle='--')
# plt.title('Silhouette Score để xác định số cụm K tối ưu')
# plt.xlabel('Số cụm (k)')
# plt.ylabel('Silhouette Score')
# plt.grid(True)
```

```
# plt.show()
# --- KẾT THÚC PHẦN SILHOUETTE SCORE ---
```



1.2. Huấn luyện mô hình K-Means

Quan sát biểu đồ Elbow Method, có một điểm uốn cong ("khủy tay") rõ rệt tại $k=3$. Tại điểm này, việc tăng thêm số cụm không làm giảm WCSS một cách đáng kể nữa. Do đó, $k=3$ được chọn là số cụm để huấn luyện mô hình K-Means. (Nếu bạn vẫn dùng Silhouette và nó cho kết quả khác, ví dụ $k=6$ tốt hơn, thì diễn giải ở đây sẽ cần kết hợp cả hai: "Mặc dù Silhouette Score có thể cao hơn ở $k=X$, phương pháp Elbow cho thấy $k=3$ là điểm mà sự cải thiện về WCSS bắt đầu giảm dần, đồng thời số cụm ít hơn giúp việc diễn giải dễ dàng hơn. Do đó, $k=3$ được chọn.")

```
In [62]: optimal_k = 3 # Đã quyết định dựa trên phân tích Elbow (và Silhouette nếu có)

kmeans = KMeans(n_clusters=optimal_k, init='k-means++', random_state=42, n_init='auto')
kmeans_labels_raw = kmeans.fit_predict(X_cluster_scaled)

df_kmeans_analysis = X_cluster.copy()
df_kmeans_analysis['KMeans_Cluster_Raw'] = kmeans_labels_raw

if 'Amount' in df_for_clustering_input.columns:
    df_kmeans_analysis['Amount'] = df_for_clustering_input.loc[df_kmeans_analysis.index, 'Amount']
if 'Qty' in df_for_clustering_input.columns:
    df_kmeans_analysis['Qty'] = df_for_clustering_input.loc[df_kmeans_analysis.index, 'Qty']

print(f"Số lượng điểm dữ liệu trong mỗi cụm (K-Means, k={optimal_k}) theo nhãn gốc")
print(pd.Series(kmeans_labels_raw).value_counts().sort_index())

# Mapping nhãn K-Means gốc sang nhãn Legend để biểu đồ khớp màu với bảng đặc điểm
# Bảng đặc điểm: Cụm 0 (28463), Cụm 1 (56859), Cụm 2 (28348)
# Biểu đồ (palette 'viridis'): màu tím (nhãn gốc 0), xanh ngọc (nhãn gốc 2), vàng (nhãn gốc 1)
# Legend 0 (Tím) = Cụm 0 Bảng (nhãn gốc 0)
```

```

# Legend 1 (Xanh ngọc) = Cụm 2 Bảng (nhãn gốc 2)
# Legend 2 (Vàng) = Cụm 1 Bảng (nhãn gốc 1)
label_mapping_for_plot = {
    0: 0,
    2: 1,
    1: 2
}
kmeans_labels_for_plot = np.array([label_mapping_for_plot[label] for label in kmeans_labels_raw])

df_kmeans_analysis['KMeans_Cluster_for_Profiling'] = kmeans_labels_raw

pca_kmeans_viz = PCA(n_components=2, random_state=42)
X_pca_kmeans_viz = pca_kmeans_viz.fit_transform(X_cluster_scaled)

plt.figure(figsize=(10, 7))
scatter_plot = sns.scatterplot(x=X_pca_kmeans_viz[:, 0], y=X_pca_kmeans_viz[:, 1],
                              hue=kmeans_labels_for_plot,
                              palette='viridis', s=50, alpha=0.7, legend='full')

centers_pca_viz = pca_kmeans_viz.transform(kmeans.cluster_centers_)
# Vẽ tâm cụm theo thứ tự nhãn gốc K-Means để khớp với màu sắc đã map
plt.scatter(centers_pca_viz[0, 0], centers_pca_viz[0, 1], marker='X', s=250, color='blue')
plt.scatter(centers_pca_viz[2, 0], centers_pca_viz[2, 1], marker='X', s=250, color='yellow')
plt.scatter(centers_pca_viz[1, 0], centers_pca_viz[1, 1], marker='X', s=250, color='green')

plt.title(f'Trực quan hóa các cụm K-Means (k={optimal_k}) bằng PCA')
plt.xlabel('Thành phần chính 1 (PC1)')
plt.ylabel('Thành phần chính 2 (PC2)')
plt.legend(title='K-Means Cluster')
plt.grid(True)
plt.show()

df_kmeans_analysis.rename(columns={'KMeans_Cluster_for_Profiling': 'KMeans_Cluster'})
df_kmeans_analysis.drop(columns=['KMeans_Cluster_Raw'], inplace=True, errors='ignore')

print("\nSố lượng điểm dữ liệu trong mỗi cụm (K-Means, k=3) dùng cho profiling (theo nhãn gốc K-Means):")
print(df_kmeans_analysis['KMeans_Cluster'].value_counts().sort_index())

```

Số lượng điểm dữ liệu trong mỗi cụm (K-Means, k=3) theo nhãn gốc K-Means:

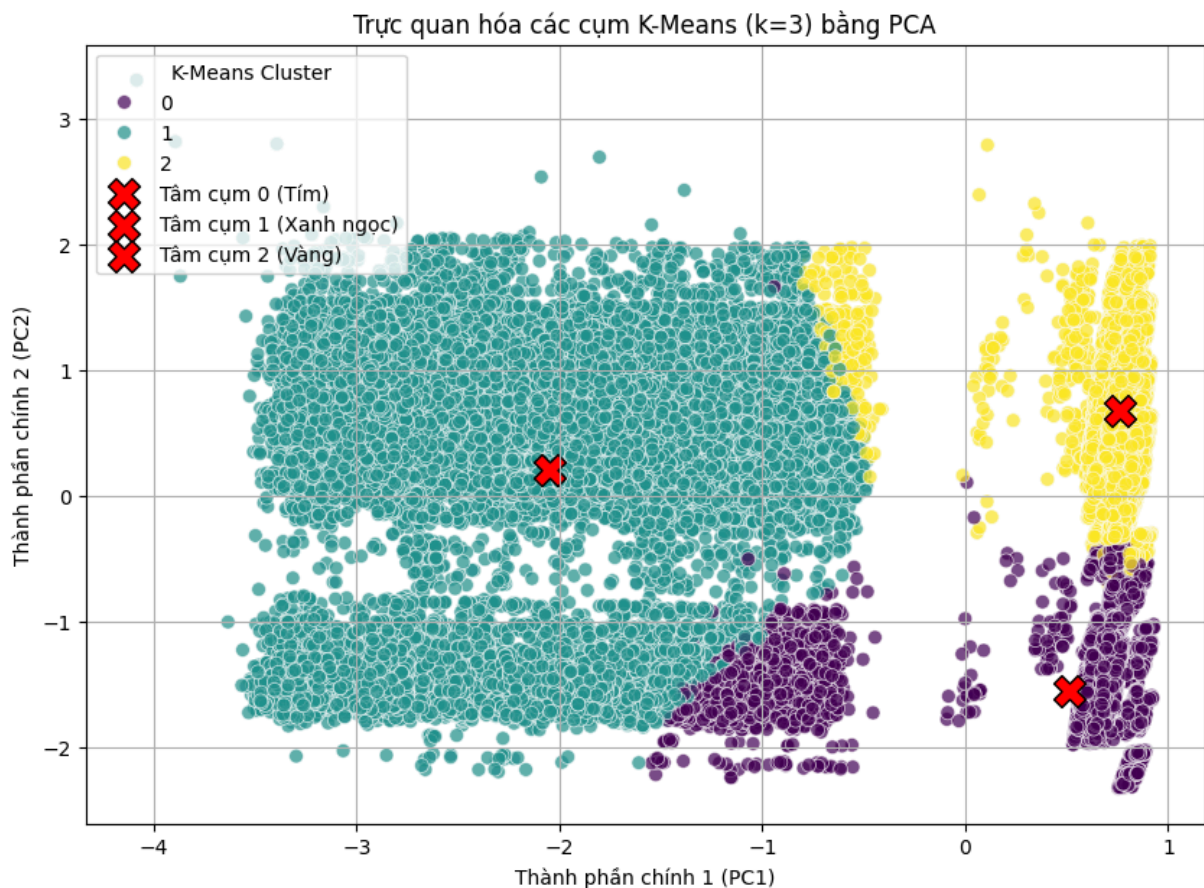
```

0    28463
1    56859
2    28348

```

Name: count, dtype: int64

C:\Users\thanh\AppData\Roaming\Python\Python312\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but PCA was fitted with feature names
warnings.warn(



Số lượng điểm dữ liệu trong mỗi cụm (K-Means, k=3) dùng cho profiling (theo nhãn gốc K-Means):

KMeans_Cluster

0 28463

1 56859

2 28348

Name: count, dtype: int64

1.3. Phân tích đặc điểm các cụm K-Means

Sau khi phân cụm, việc kiểm tra đặc điểm của từng cụm là cần thiết để hiểu rõ hơn về ý nghĩa của chúng. Bảng dưới đây tóm tắt các thông tin chính cho mỗi cụm, bao gồm số lượng đơn hàng, giá trị và số lượng trung bình, tỷ lệ các phương thức hoàn thành đơn hàng, dịch vụ vận chuyển, và các loại sản phẩm, kích thước, bang nhận hàng, khuyến mãi phổ biến nhất. (Lưu ý: Đối với các cột đã được mã hóa bằng Label Encoding như 'Style', 'Category', 'Size', 'ship-state', 'promotion-ids', giá trị hiển thị là tên đã giải mã kèm theo mã số gốc để tiện đối chiếu).

```
In [63]: # Label_encoders được giả định là đã có từ cell [13] của notebook gốc của bạn
# df_kmeans_analysis đã có cột 'KMeans_Cluster' là nhãn gốc từ K-Means
# df_for_clustering_input là df_model (dữ liệu trước khi drop cột và scale cho clus
# X_cluster là df_for_clustering_input sau khi drop các cột không cần thiết (nhưng

kmeans_profile_list = []
# optimal_k đã được định nghĩa ở cell C3
for cluster_id_kmeans in range(optimal_k):
    current_cluster_indices = df_kmeans_analysis[df_kmeans_analysis['KMeans_Cluster
```

```

if current_cluster_indices.empty:
    print(f"Cụm K-Means {cluster_id_kmeans} không có điểm dữ liệu nào.")
    continue

profile = {'KMeans_Cluster': cluster_id_kmeans,
           'Count': len(current_cluster_indices)}

# Tính trung bình cho 'Amount' và 'Qty' gốc
if 'Amount' in df_for_clustering_input.columns:
    profile['Avg_Amount'] = df_for_clustering_input.loc[current_cluster_indices
if 'Qty' in df_for_clustering_input.columns:
    profile['Avg_Qty_Original'] = df_for_clustering_input.loc[current_cluster_

# Tính tỷ lệ cho các cột one-hot/boolean (đã là 0/1 trong X_cluster)
for col in X_cluster.columns:
    if X_cluster[col].nunique() == 2 and X_cluster[col].min() == 0 and X_cluste
        if col in X_cluster.loc[current_cluster_indices].columns:
            col_data_for_cluster = X_cluster.loc[current_cluster_indices, col]
            profile[f'Ratio_{col}'] = col_data_for_cluster.mean()

# Tìm mode cho các cột đã Label encode và diễn giải chúng
label_encoded_cols_original = ['Style', 'Category', 'Size', 'ship-state', 'prom
for col_le in label_encoded_cols_original:
    if col_le in df_for_clustering_input.columns and col_le in label_encoders:
        original_encoded_vals_for_mode = df_for_clustering_input.loc[current_cl
        mode_series = original_encoded_vals_for_mode.mode()

        if not mode_series.empty:
            mode_encoded = mode_series[0]
            try:
                mode_decoded = label_encoders[col_le].inverse_transform([int(mo
                profile[f'Most_Common_{col_le}'] = f"{mode_decoded} (Code: {int
            except Exception as e:
                profile[f'Most_Common_{col_le}'] = f"Code: {int(mode_encoded)}
        else:
            profile[f'Most_Common_{col_le}'] = 'N/A'
    elif col_le in X_cluster.columns:
        mode_series = X_cluster.loc[current_cluster_indices, col_le].mode()
        profile[f'Most_Common_{col_le}_Code'] = mode_series[0] if not mode_seri

kmeans_profile_list.append(profile)

if kmeans_profile_list:
    kmeans_cluster_profile_df = pd.DataFrame(kmeans_profile_list)
    print("\nĐặc điểm các cụm K-Means:")

    ordered_profile_cols = ['Count']
    if 'Avg_Amount' in kmeans_cluster_profile_df.columns: ordered_profile_cols.append('Avg_Amount')
    if 'Avg_Qty_Original' in kmeans_cluster_profile_df.columns: ordered_profile_cols.append('Avg_Qty_Original')

    ordered_profile_cols.extend(sorted([col for col in kmeans_cluster_profile_df.co
    ordered_profile_cols.extend(sorted([col for col in kmeans_cluster_profile_df.co

    final_ordered_cols = [col for col in ordered_profile_cols if col in kmeans_clus

```

```

if 'KMeans_Cluster' in kmeans_cluster_profile_df.columns:
    display(kmeans_cluster_profile_df.set_index('KMeans_Cluster')[final_ordered_cols])
else:
    display(kmeans_cluster_profile_df[final_ordered_cols])
else:
    print("Không có cụm K-Means nào để phân tích.")

```

Đặc điểm các cụm K-Means:

	Count	Avg_Amount	Avg_Qty_Original	Ratio_B2B	Ratio_Fulfilment_Amazon
KMeans_Cluster					
0	28463	780.235323	1.003092	0.007027	0.900608
1	56859	608.733323	1.003517	0.006947	0.992930
2	28348	654.809299	1.005291	0.008113	0.000000

```

In [66]: plt.figure(figsize=(8, 5))
sns.barplot(data=kmeans_cluster_profile_df, x='KMeans_Cluster', y='Avg_Amount', palette='viridis')
plt.title('Giá trị đơn hàng trung bình theo cụm K-Means')
plt.xlabel('Cụm K-Means')
plt.ylabel('Giá trị đơn hàng trung bình (Avg_Amount)')
plt.show()

```

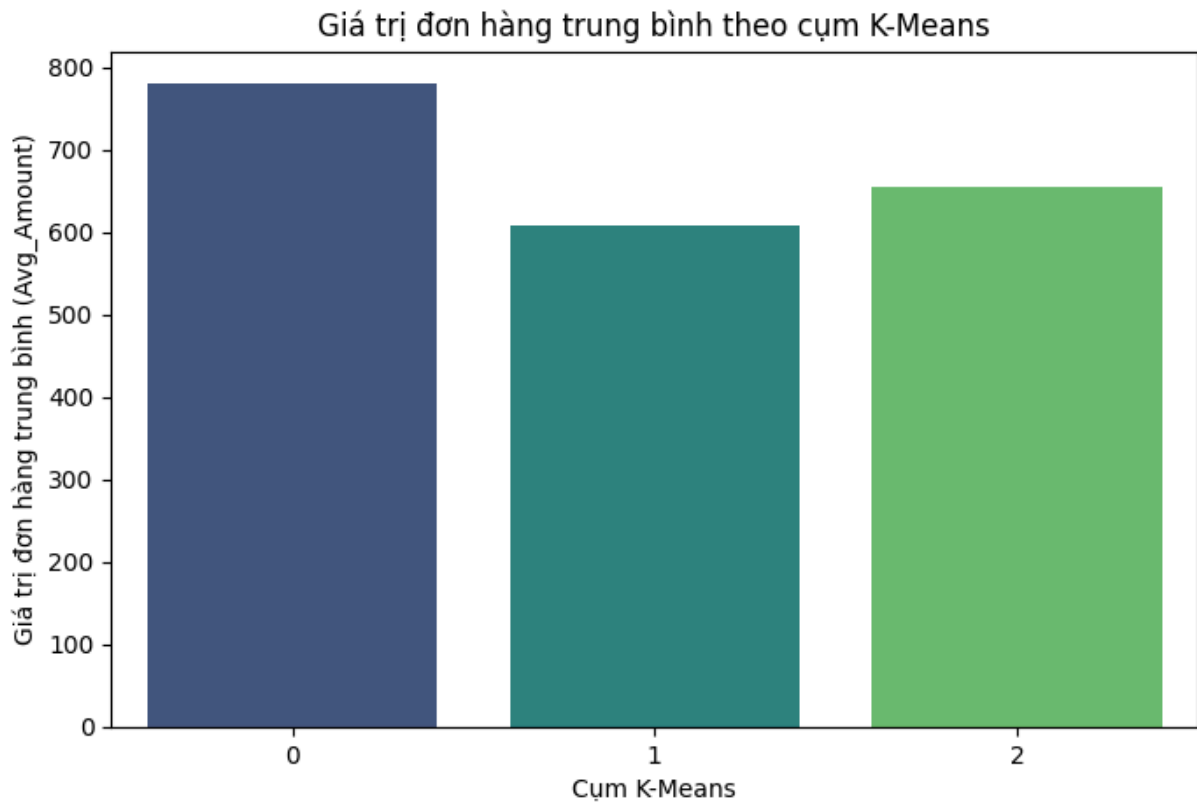
D:\Temp\ipykernel_10684\3089064379.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

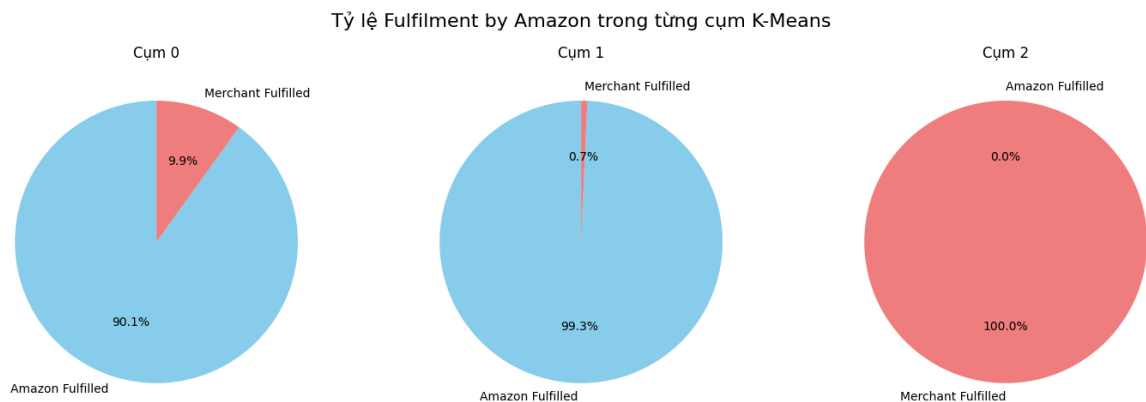
```

sns.barplot(data=kmeans_cluster_profile_df, x='KMeans_Cluster', y='Avg_Amount', palette='viridis')

```



```
In [67]: fig, axes = plt.subplots(1, optimal_k, figsize=(5 * optimal_k, 5), sharey=True)
fig.suptitle('Tỷ lệ Fulfilment by Amazon trong từng cụm K-Means', fontsize=16)
for i in range(optimal_k):
    ax = axes[i] if optimal_k > 1 else axes # Xử lý trường hợp chỉ có 1 cụm
    ratio_amazon = kmeans_cluster_profile_df.loc[i, 'Ratio_Fulfilment_Amazon']
    sizes = [ratio_amazon, 1 - ratio_amazon]
    labels = ['Amazon Fulfilled', 'Merchant Fulfilled']
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'red'])
    ax.set_title(f'Cụm {i}')
plt.tight_layout(rect=[0, 0, 1, 0.96]) # Điều chỉnh để tiêu đề không bị che
plt.show()
```



1.4. Diễn giải các cụm K-Means và Kết luận

Dựa trên bảng đặc điểm các cụm ở trên:

- **Cụm 0 (Số lượng: [Điền số lượng từ output], Màu tím trên PCA):**

- Đặc trưng bởi giá trị đơn hàng trung bình cao nhất ([Điền Avg_Amount]).
- Sản phẩm phổ biến nhất là "Set" (Loại hàng: SET268).
- Phần lớn (khoảng 90%) được hoàn thành bởi Amazon và sử dụng dịch vụ giao hàng nhanh.
- *Tóm lại: Cụm này có thể đại diện cho các đơn hàng sản phẩm "Set" có giá trị cao, được ưu tiên xử lý và vận chuyển nhanh bởi Amazon.*
- **Cụm 1 (Số lượng: [Điền số lượng từ output], Màu vàng trên PCA):**
 - Đây là cụm có số lượng đơn hàng lớn nhất.
 - Giá trị đơn hàng trung bình thấp nhất ([Điền Avg_Amount]).
 - Sản phẩm phổ biến nhất là "kurta" (Loại hàng: JNE3405, Size L).
 - Gần như toàn bộ (trên 99%) được hoàn thành bởi Amazon và vận chuyển nhanh.
 - *Tóm lại: Cụm này đại diện cho phân khúc phổ thông nhất, chủ yếu là sản phẩm "kurta" giá trị thấp, được Amazon xử lý và giao hàng rất hiệu quả.*
- **Cụm 2 (Số lượng: [Điền số lượng từ output], Màu xanh ngọc trên PCA):**
 - Giá trị đơn hàng trung bình ở mức giữa.
 - Sản phẩm phổ biến cũng là "kurta" (Loại hàng: JNE3797, Size M).
 - Điểm khác biệt cốt lõi: **100% đơn hàng trong cụm này được hoàn thành bởi Người bán (Merchant) và sử dụng dịch vụ giao hàng tiêu chuẩn.**
 - Khuyến mãi phổ biến liên quan đến chương trình tài chính của Amazon (Amazon PLCC Free-Financing).
 - *Tóm lại: Cụm này đại diện cho các đơn hàng "kurta" do người bán tự xử lý và vận chuyển, có thể gắn liền với các ưu đãi tài chính cụ thể.*

Kết luận cho K-Means:

Phân cụm K-Means với $k=3$ đã giúp phân tách dữ liệu thành ba nhóm đơn hàng có ý nghĩa, khác biệt rõ rệt dựa trên phương thức hoàn thành đơn hàng, tốc độ giao hàng, loại sản phẩm chính và giá trị đơn hàng. Điều này cung cấp cái nhìn sâu sắc về các phân khúc hoạt động khác nhau trên nền tảng. Bang nhận hàng phổ biến nhất cho cả ba cụm là MAHARASHTRA, cho thấy tầm quan trọng của thị trường này. Tỷ lệ đơn hàng B2B trong tất cả các cụm đều rất thấp.