VNUHCM – UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY _ K20

Phân tích dữ liệu ứng dụng-CSC12110_20HTTT

ĐỒ ÁN THỰC HÀNH #1

Sinh viên

20127217 – Nguyễn Trung Kiên
20127401 – Quách Đỗ Gia Huy
20127432 – Nguyễn Hoài An
20127675 – Dương Đình Hiếu

Học Kỳ III

Mục Lục

I.Phân công công việc

MSSV	Họ và tên	Công việc	Đánh giá
20127217	Nguyễn Trung Kiên	**Quan sát và phân tích dữ liệu (EDA)** ★ Cho biết kiểu dữ liệu của các thuộc tính ★ Đối với cột Price, cho biết ★ Mô tả số lượng missing value của mỗi thuộc tính ★ Tính tỷ lệ % missing value của mỗi thuộc tính ★ Xử lý dữ liệu missing	23%
20127401	Quách Đỗ Gia Huy	**EDA** : ★ Outlier: sử dụng đồ thị boxplot để quan sát cột Price có tồn tại outlier không? **Visualize** : ★ Giá vé tăng khi nào? ★ So sánh tỷ lệ các chuyến bay thẳng và trung chuyển (có dừng) ★ Ngoài các đặc trưng trong tập dữ liệu, các yếu tố nào có khả năng ảnh hưởng đến giá vé chuyến bay?	23%
20127432	Nguyễn Hoài An	**Visualize** : ★ Giá có thay đổi tùy theo hãng hàng không hay không? ★ Giá vé bị ảnh hưởng như thế nào khi mua vé chỉ 1 hoặc 2 ngày trước ngày khởi hành? ★ Giá vé có thay đổi theo thời gian đi và đến không? ★ Dự đoán giá vé	31%
20127675	Dương Đình Hiếu	**Visualize** : ★ Nước nào có lượng chuyến bay nhiều nhất? ★ Hãng máy bay nào được khách đặt vé nhiều/thấp nhất? ★ Tháng nào là tháng cao điểm	23%

A. Quan sát và phân tích tập dữ liệu (EDA)

1. Kiểu dữ liệu của các thuộc tính

Bước đầu sử dụng thư viện pandas và matplotlib để đọc dữ liệu từ hai tệp Excel (Data_Train_p1.xlsx và Data_Train_p2.xlsx), gộp chúng thành một DataFrame duy nhất và hiển thị sau đó sử dụng phương thức dtypes của pandas để lấy thông tin về kiểu dữ liệu của từng cột trong DataFrame merged_data và sau đó in thông tin này ra

```
Airline            object
Date_of_Journey    object
Source             object
Destination        object
Route              object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Additional_Info    object
Price               int64
dtype: object
```

2. Đối với cột Price. Cho biết:

- **Giá cao nhất, thấp nhất, giá trung bình, mức giá xuất hiện nhiều**

Sử dụng

max_price: Lấy giá trị lớn nhất trong cột 'Price'.
min_price: Lấy giá trị nhỏ nhất trong cột 'Price'.
mean_price: Tính giá trị trung bình của cột 'Price'.
most_price: Lấy giá trị xuất hiện nhiều nhất (mode) trong cột 'Price' của DataFrame merged_data. Do mode có thể trả về nhiều giá trị nếu có nhiều giá trị có số lần xuất hiện như nhau, nên chọn phần tử đầu tiên của mode bằng cách sử dụng [0].
Kết quả thu được trên màn hình :

```
Giá cao nhất:  79512
Giá thấp nhất:  1759
Giá trung bình:  9087.545582178958
Mức giá xuất hiện nhiều:  10262
```

- **Độ lệch chuẩn của cột price?**

  price_std: Tính độ lệch chuẩn của cột 'Price' trong DataFrame merged_data bằng cách sử dụng phương thức std() của pandas.

  sau đó hiển thị ra màn hình với kết quả :

```
Độ lệch chuẩn của cột Price:  4611.411872913706
```

- **Cho biết giá phân phối xung quanh trung tâm hay cách xa trung tâm? Nhận xét về giá trị mean và median của tập dữ liệu.**

```
median_price = merged_data['Price'].median()
if mean_price < median_price:
  price_dis = "phân phối xung quanh trung tâm"
else:
  price_dis = "phân phối cách xa trung tâm"

print("Phân phối giá: ",price_dis)
print("Median: ",median_price)
```

median_price = merged_data['Price'].median(): Tính giá trị trung vị của cột 'Price'. Sau đó, sử dụng một câu lệnh điều kiện để xác định xem giá trị trung bình (mean_price) có nhỏ hơn giá trị trung vị hay không. Nếu có, thì được cho rằng dữ liệu phân phối xung quanh trung tâm; ngược lại, được cho rằng dữ liệu phân phối cách xa trung tâm.

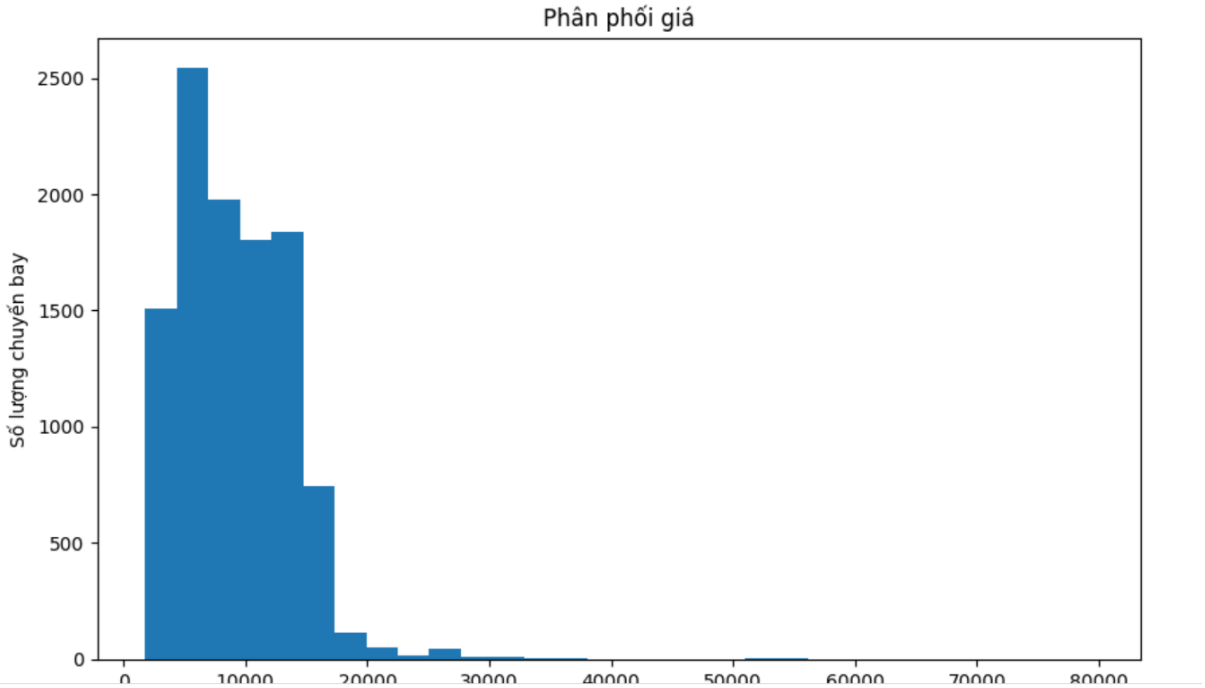
So sánh giữa giá trị trung bình và giá trị trung vị để đưa ra nhận xét về tính đối xứng của phân phối dữ liệu. Nếu giá trị trung bình bằng giá trị

trung vị, thì dữ liệu được coi là có phân phối đối xứng. Nếu giá trị trung bình lớn hơn giá trị trung vị, có sự lệch lớn (lệch dương) trong phân phối. Ngược lại, nếu giá trị trung bình nhỏ hơn giá trị trung vị, có sự lệch nhỏ (lệch âm) trong phân phối.

```
Phân phối giá:  phân phối cách xa trung tâm
Median:  8372.0
Mean lớn hơn Median -> có sự lệch lớn (lệch dương) trong phân phối.
```

- **Vẽ đồ thị histogram để nhận xét dữ liệu có bị lệch không?**

histogram nghiêng về trái,dữ liệu có thể có sự lệch trái, và giá trị trung bình có thể nhỏ hơn giá trung vị



3, Mô tả số lượng missing value của mỗi thuộc tính

1. `merged_data.isnull()`:- Phương thức này tạo ra một DataFrame với cùng kích thước như `merged_data`, nhưng các ô dữ liệu được đánh dấu là `True` nếu chúng chứa giá trị thiếu và `False` nếu chúng không chứa giá trị thiếu.

2. `.sum(): - Phương thức này được gọi trên DataFrame thu được từ bước trước. Nó tính tổng của các giá trị `True` trên mỗi cột, do đó trả về một Series với chỉ mục là tên của các cột và giá trị là tổng số lượng giá trị thiếu trong từng cột.

```
Số lượng missing value của mỗi thuộc tính:
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

Bảng hiển thị số lượng giá trị thiếu cho mỗi thuộc tính như sau:

- Không có giá trị thiếu (0): Airline, Date_of_Journey, Source, Destination, Dep_Time, Arrival_Time, Duration, Additional_Info, Price.
- Có một giá trị thiếu (1): Route và Total_Stops.

4. Tính tỷ lệ % missing value của mỗi thuộc tính

- missing_values_percent: Tạo một Series chứa phần trăm giá trị thiếu cho mỗi cột, với chỉ mục là tên cột và giá trị là phần trăm giá trị thiếu.
- (missing_values / len(merged_data)) * 100: Tính phần trăm giá trị thiếu bằng cách chia số lượng giá trị thiếu cho tổng số lượng dòng và nhân với 100.

```
Airline            0.00000
Date_of_Journey    0.00000
Source             0.00000
Destination        0.00000
Route              0.00936
Dep_Time           0.00000
Arrival_Time       0.00000
Duration           0.00000
Total_Stops        0.00936
Additional_Info    0.00000
Price              0.00000
dtype: float64
```

5. Xử lý dữ liệu missing

- Xử lý giá trị thiếu trong cột 'Route' và 'Total_Stops' của DataFrame merged_data. Dưới đây là giải thích quy trình:

```
merged_data.dropna(subset=['Route'])
merged_data.dropna(subset=['Total_Stops'])
```

Cả hai dòng mã này giống nhau và nhằm loại bỏ các hàng chứa giá trị thiếu trong cột tương ứng ('Route' và 'Total_Stops').

```
mode_route = merged_data['Route'].mode()[0]
mode_stops = merged_data['Total_Stops'].mode()[0]
```

Dòng mã trên lấy giá trị mode của cột 'Route' và 'Total_Stops'. Giá trị mode là giá trị xuất hiện nhiều nhất trong cột.

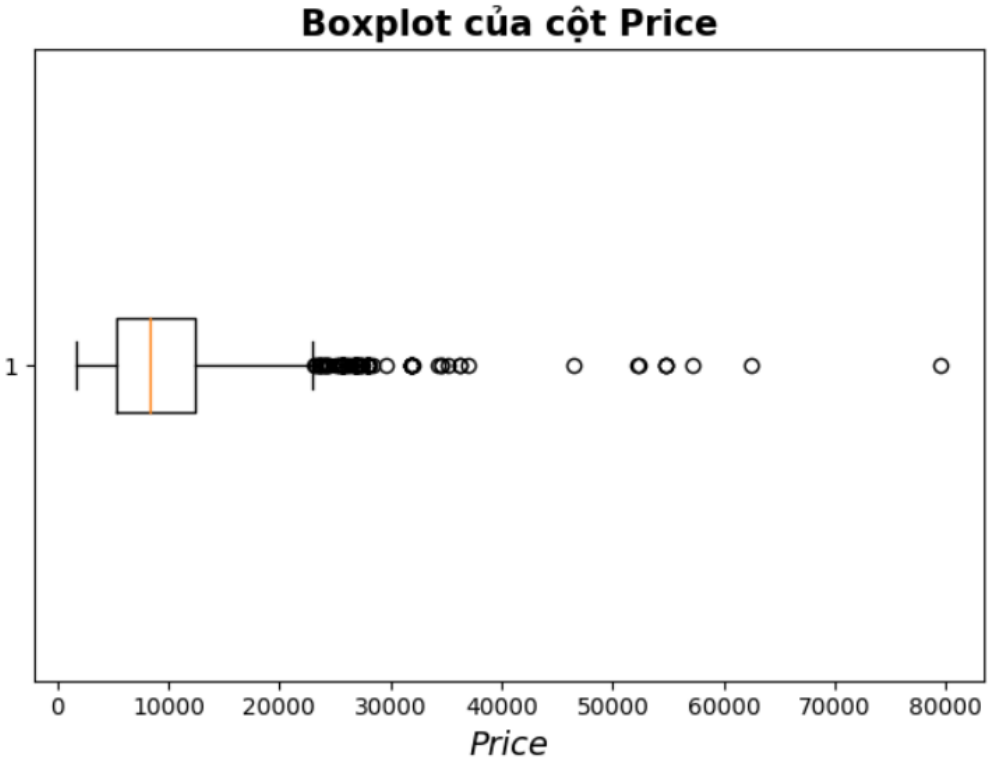
```
merged_data['Route'].fillna(mode_route, inplace=True)
merged_data['Total_Stops'].fillna(mode_stops, inplace=True)
```

Dòng mã trên thực hiện việc thay thế giá trị thiếu trong cột 'Route' và 'Total_Stops' bằng giá trị mode tương ứng. Phương thức fillna được sử dụng để điền giá trị thiếu

Quy trình này giúp điền giá trị thiếu bằng giá trị mode, giả sử giá trị mode là giá trị xuất hiện nhiều nhất và thường được sử dụng để điền vào giá trị thiếu trong trường hợp này.

6. Outlier: sử dụng đồ thị boxplot để quan sát cột Price có tồn tại outlier không?

- **Giá trị cao nhất**: Giá trị max của biểu đồ boxplot là 23017.0
- **Giá trị thấp nhất**: Giá trị min nhất biểu đồ boxplot là 1759
- **Giá trị trung vị** : Giá trị trung vị của biểu đồ boxplot là 8372
- **Outliers**:
  - Qua biểu đồ cho thấy cột Price có tồn tại outlier .
  - Có rất nhiều outliers nằm ở phía bên phải của biểu đồ, cho thấy có một số giá vé cao hơn hẳn so với phần lớn dữ liệu. Cụ thể, những giá vé này có giá trị từ 23017 đến 79512, đều lớn hơn giá trị max của biểu đồ boxplot, cho thấy là outliers.
  - Qua đó cho thấy rằng có một số chuyến bay đặc biệt có giá vé cao hơn hẳn so với phần lớn các chuyến bay khác.



➔ **Giữ lại các outliers này vì : biểu thị sự đa dạng của giá vé và có thể phản ánh các yếu tố đặc biệt như sự kiện lớn, chênh lệch giá theo thời gian, hay các điều kiện đặc biệt.**

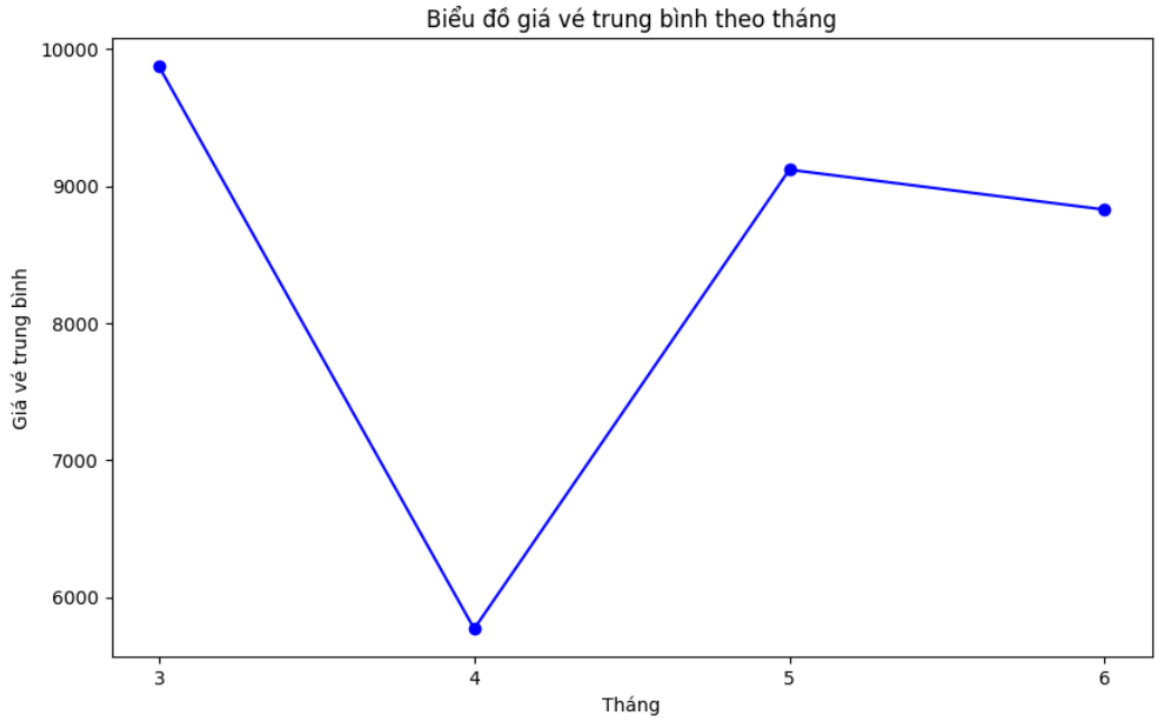
Các giá vé cao đặc biệt có thể phản ánh sự biến động của thị trường. Việc giữ lại outliers có thể giúp mô hình thấy sự biến động và phản ánh thực tế thị trường hơn.

B. Visualize dữ liệu dạng bảng hoặc đồ thị

1. Giá vé tăng khi nào?

❖ Mục tiêu phân tích:

- Phân tích sự biến động giá vé theo tháng để xác định những tháng có xu hướng tăng giá vé.
- Cách phân tích: gom nhóm giá vé theo năm , tiếp theo gom nhóm theo các tháng của năm đó , lấy giá vé trung bình của mỗi tháng trong năm và trực quan hóa dữ liệu



	Year	Month	mean_price
0	2019	3	9759.455882
1	2019	4	5770.847081
2	2019	5	9109.506066
3	2019	6	8828.796134

❖ **Nhận xét :**

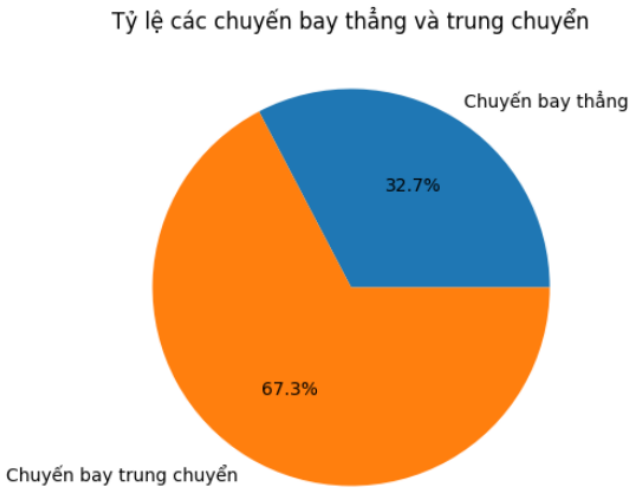
1. **Có sự thay đổi giá vé qua các tháng của năm 2019:**
   - Tháng 3 có giá vé trung bình cao nhất (9759.45).
   - Tháng 4 có giá vé trung bình thấp nhất (5770.85).
   - Tháng 5 và 6 có giá vé trung bình ở mức trung bình (9109.5 và 8828.80).
2. **Biểu đồ**

- Ở biểu đồ line plot có : trục x là các tháng và trục y là giá vé trung bình của các tháng trong năm tương ứng. và các đường ở biểu đồ sẽ biểu thị về sự biến động giá vé theo tháng và so sánh giữa các tháng.
- Dựa trên biểu đồ đường , giá vé trung bình cao nhất vào Tháng 3 có giá vé là (10674.51)
- Giá vé giảm vào tháng 4 khi có giá vé trung bình thấp nhất (5770.85).Qua đường biến động giá vé cho thấy giá vé tăng từ tháng 4 đến tháng 5, giảm vào giữa tháng 5 đến tháng 6.

2. So sánh tỷ lệ các chuyến bay thẳng và trung chuyển (có dừng)

❖ Mục tiêu của phân tích:

- So sánh tỷ lệ các chuyến bay thẳng và trung chuyển
- Cách phân tích : Xác định các loại chuyến bay : qua cột 'Total_Stops'. Qua đó sẽ xác định số lượng chuyến bay thẳng(non - stop) và chuyến bay trung chuyển (1-stop, 2-stop, 3-stop,...).
  - direct_flights = merged_data[merged_data['Total_Stops'] == 'non-stop'].shape[0]
  - transit_flights = merged_data[merged_data['Total_Stops'] != 'non-stop'].shape[0]
  - Qua đó tính tỷ lệ theo các loại chuyến bay và vẽ biểu đồ tròn theo tỷ lệ



❖ **Nhận xét**

- Biểu đồ này cho thấy tổng quan về tỷ lệ giữa số lượng chuyến bay thẳng (non-stop) và chuyến bay trung chuyển (1-stop, 2-stop, 3-stop,...).
- Nhưng không cho thấy số lượng chuyến bay cụ thể của mỗi loại
- Qua biểu đồ có thể thấy tỷ lệ chuyến bay trung chuyển (67.3 %) cao hơn tỉ lệ các chuyến bay thẳng (32.7%)

3. Nước nào có lượng chuyến bay nhiều nhất?

I) Import Thư Viện:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

- Dòng code này import các thư viện cần thiết: **pandas** để xử lý dữ liệu, **seaborn** và **matplotlib.pyplot** để vẽ biểu đồ.

II) Xử Lý Dữ Liệu:

```
country_counts = merge_data['Source'].value_counts()
```

- Sử dụng **value_counts()** để đếm số lượng chuyến bay theo nước và lưu kết quả vào biến **country_counts**. Kết quả sẽ là Series có chỉ mục là tên nước và giá trị là số lượng chuyến bay tương ứng.

III) Tạo Mảng Màu:

```
colors = ['grey'] * len(country_counts) max_index = country_counts.idxmax()
colors[country_counts.index.get_loc(max_index)] = '#FF0000'
```

- Tạo một mảng màu **colors** có chiều dài bằng số lượng nước. Mặc định là màu xám cho tất cả các cột.
- Xác định chỉ mục của nước có số lượng chuyến bay nhiều nhất bằng **idxmax()** và đặt màu đỏ đậm cho cột đó trong mảng màu.

IV) Vẽ Biểu Đồ Cột:

```
plt.figure(figsize=(10, 6)) sns.barplot(x=country_counts.index,
y=country_counts.values, palette=colors)
```

- Tạo một biểu đồ cột sử dụng thư viện seaborn (**sns.barplot**). Trục x là tên nước (**country_counts.index**), trục y là số lượng chuyến bay (**country_counts.values**), và sử dụng màu từ mảng **colors**.

V) Thêm Tiêu Đề và Mô Tả:

plt.title('Biểu Đồ Cột Thể Hiện Số Lượng Chuyến Bay Theo Nước', fontsize=16, fontweight='bold', pad=20, color='#FF0000')
plt.text(1.05, 0.8, 'Trục Ox: Số lượng chuyến bay\nTrục Oy: Nước',transform=plt.gca().transAxes, fontsize=12, va='center', bbox=dict(facecolor='white', edgecolor='white', boxstyle='round,pad=0.5'))

- Thêm tiêu đề cho biểu đồ với các thuộc tính như kích thước font, độ đậm và màu sắc.
- Thêm mô tả bên phải của biểu đồ để giải thích ý nghĩa của trục x và trục y.

VI) Hiển Thi Biểu Đồ:

plt.show()

- Dòng code này hiển thị biểu đồ cột đã tạo.

Þ Nhận Xét:

¬ Lý do cho việc chọn bar chart (Biểu đồ cột) và hiệu quả nó mang lại :

1. So Sánh Số Lượng Chuyến Bay:

o Mục tiêu chính là so sánh số lượng chuyến bay giữa các nước khác nhau. Biểu đồ cột là lựa chọn phù hợp để thể hiện sự chênh lệch giữa các giá trị tuyệt đối.

2. Dữ Liệu Phân Loại (Categorical):

o Dữ liệu trên trục x là các nước, là dạng dữ liệu phân loại, và không liên tục. Biểu đồ cột phản ánh rõ ràng mối quan hệ giữa các biến phân loại này.

3.Mô Tả Chính Xác:

o Biểu đồ cột cung cấp một cái nhìn tổng quan về số lượng chuyến bay của mỗi nước một cách chính xác và dễ đọc.

4.Màu Đỏ Đậm để Làm Nổi Bật:

o Sự sử dụng màu đỏ đậm cho cột của nước có số lượng chuyến bay nhiều nhất làm nổi bật thông tin quan trọng, giúp người đọc dễ dàng nhận diện.

5.Hiệu Quả Trực Quan:

o Biểu đồ cột là một trong những loại biểu đồ trực quan và dễ hiểu, giúp người đọc có cái nhìn tổng quan về phân bố số lượng chuyến bay.

6.Kích Thước và Khả Năng Hiển Thị:

o Kích thước của biểu đồ được đặt là 10x6 để đảm bảo rõ ràng và không gian đủ để hiển thị thông tin.

4. Hãng máy bay nào được khách đặt vé nhiều/thấp nhất?

I) Tính Số Lượng Đặt Vé Cho Mỗi Hãng Máy Bay:

```
airline_counts = merged_data['Airline'].value_counts()
```

- Sử dụng **value_counts()** để đếm số lượng đặt vé cho mỗi hãng máy bay. Kết quả là một Series với chỉ mục là tên hãng máy bay và giá trị là số lượng đặt vé tương ứng.

II) Tao Mảng Màu:

```
colors_airline = ['grey'] * len(airline_counts)
max_index_airline = airline_counts.idxmax()
colors_airline[airline_counts.index.get_loc(max_index_airline)] = '#FF0000'
```

- Tạo một mảng màu **colors_airline** có chiều dài bằng số lượng hãng máy bay. Mặc định là màu xám cho tất cả các cột.
- Xác định chỉ mục của hãng máy bay có số lượng đặt vé nhiều nhất bằng **idxmax()** và đặt màu đỏ đậm cho cột đó trong mảng màu.

III) Vẽ Biểu Đồ Cột Cho Số Lượng Đặt Vé Theo Hãng Máy Bay:

```
plt.figure(figsize=(10, 6))
sns.barplot(x=airline_counts.index, y=airline_counts.values, palette=colors_airline)
plt.title('Biểu Đồ Cột Thể Hiện Số Lượng Đặt Vé Theo Hãng Máy Bay', fontsize=16,
fontweight='bold', pad=20, color='#FF0000')
plt.text(1.05, 0.8, 'Trục Ox: Hãng Máy Bay\nTrục Oy: Số Lượng Đặt Vé',
transform=plt.gca().transAxes, fontsize=12, va='center', bbox=dict(facecolor='white',
edgecolor='white', boxstyle='round,pad=0.5'))
```

plt.show()

- Tạo một biểu đồ cột sử dụng thư viện seaborn (**sns.barplot**). Trục x là tên hãng máy bay (**airline_counts.index**), trục y là số lượng đặt vé (**airline_counts.values**), và sử dụng màu từ mảng **colors_airline**.

IV) Phần Thêm Tiêu Đề và Mô Tả , Hiển Thị Biểu Đồ:

· Thực hiện tương tự câu 3

Þ **Nhận Xét:**

¬ Lý do cho việc chọn bar chart (Biểu đồ cột) và hiệu quả nó mang lại :

1. Tương tự như câu 3
2. Nhưng có điểm khác biệt là thay đổi trục toạ độ biến biểu đồ cột nằm ngang thuận tiện cho việc hiển thị và quan sát

¬ Biểu đồ đã sắp xếp theo thứ từ lớn đến bé giúp ta có thể dễ dàng tìm ra được hãng máy bay có số lượng vé đặt lớn nhất và nhỏ nhất

5. Tháng nào là tháng cao điểm?

**Theo biểu đồ tròn **

I)Chuyển Đổi Cột 'Date_of_Journey' Sang Đinh Dang datetime:

merged_data['Date_of_Journey'] = pd.to_datetime(merged_data['Date_of_Journey'], format='%d/%m/%Y')

- Dòng code này sử dụng hàm **pd.to_datetime** để chuyển đổi cột 'Date_of_Journey' từ dạng chuỗi sang định dạng datetime. Tham số **format='%d/%m/%Y'** xác định định dạng ngày tháng trong chuỗi.

II)Tính Số Lương Chuyến Bay Theo Tháng:

month_counts =merged_data['Date_of_Journey'].dt.month.value_counts()

- Dòng code này sử dụng thuộc tính **dt.month** để trích xuất thông tin tháng từ cột 'Date_of_Journey'. Sau đó, **value_counts()** được sử dụng để đếm số lượng chuyến bay cho mỗi tháng.

III)Vẽ Biểu Đồ Tròn:

```
plt.figure(figsize=(8, 8))
plt.pie(month_counts, labels=month_counts.index, autopct='%1.1f%%',
startangle=140, colors=sns.color_palette('pastel'))
plt.title('Biểu Đồ Tròn Phân Phối Số Lượng Chuyến Bay Theo Tháng', fontsize=14,
fontweight='bold', pad=10, color='#FF0000')
plt.show()
```

- Dòng code này vẽ biểu đồ tròn sử dụng **plt.pie**. **month_counts** được sử dụng làm dữ liệu, **labels** là các nhãn tháng, **autopct** là hiển thị phần trăm trên biểu đồ, **startangle** xác định góc bắt đầu, và **colors** là một mảng màu sắc được tạo bằng **sns.color_palette('pastel')**.

Theo biểu đồ đường

I)Chuyển Đổi Cột 'Date_of_Journey' Sang Đinh Dang datetime:

Thực hiện tương tự biểu đồ tròn trên

II)Tính Số Lượng Chuyến Bay Theo Tháng:

```
month_counts =
merged_data['Date_of_Journey'].dt.to_period('M').value_counts().sort_index()
```

- Dòng code này sử dụng **dt.to_period('M')** để chuyển đổi thông tin ngày thành chu kỳ tháng. Sau đó, sử dụng **value_counts()** để đếm số lượng chuyến bay cho mỗi chu kỳ tháng. Cuối cùng, **sort_index()** được sử dụng để sắp xếp theo thứ tự thời gian.

III)Vẽ Biểu Đồ Đường:

```
plt.figure(figsize=(10, 6))
sns.lineplot(x=month_counts.index.astype(str), y=month_counts.values)
```

```
plt.title('Biểu Đồ Đường Phân Phối Số Lượng Chuyến Bay Theo Tháng', fontsize=14, fontweight='bold', pad=10, color='#FF0000')
plt.xticks(rotation=45)
plt.xlabel('Tháng')
plt.ylabel('Số lượng chuyến bay')
plt.show()
```

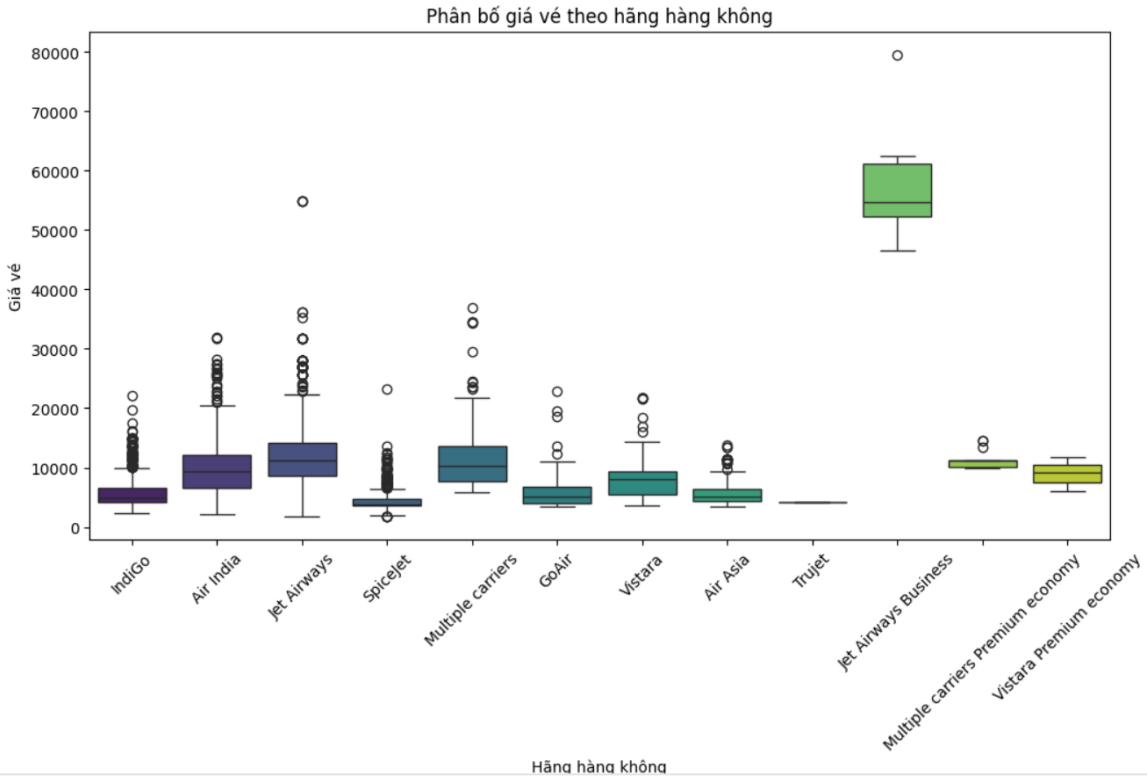
- Dòng code này sử dụng thư viện Seaborn để vẽ biểu đồ đường (**sns.lineplot**). Trục x là các chu kỳ tháng (đã chuyển đổi thành chuỗi để hiển thị), và trục y là số lượng chuyến bay.
- **plt.title**, **plt.xticks**, **plt.xlabel**, và **plt.ylabel** được sử dụng để thêm tiêu đề và điều chỉnh các thành phần khác của biểu đồ để làm cho nó dễ đọc và hiểu.

Þ **Nhận Xét:**

Theo câu hỏi tháng nào là tháng cao điểm thì việc sử dụng biểu đồ tròn, cột, đường, ... điều có thể phân tích được vấn đề. Nhưng sự lựa chọn phù hợp và tối ưu thể hiện phân phối theo tháng và xu hướng biến động qua thời gian là biểu đồ đường

6. Giá có thay đổi tùy theo hãng hàng không hay không?

a) Cách 1: sử dụng boxplot để đánh giá



- Nhận xét:
  - Biểu đồ cho thấy sự phân bố giá tùy theo từng hãng
  - Biểu đồ chỉ cho cái nhìn tổng quan về giá vé theo hãng nhưng lại chưa chi tiết đến các chuyến khác nhau
  - Biểu đồ cho thấy phân bố giá của các hãng không giống nhau , trong khi phân bố giá của hãng Jet airways Bussiness có phân bố giá rất cao so với tất cả các hãng còn lại lại có phân bố giá thấp hơn rất nhiều như IndiGo,, Spicejet
  - Hãng Trujet không có có phân bố giá rộng

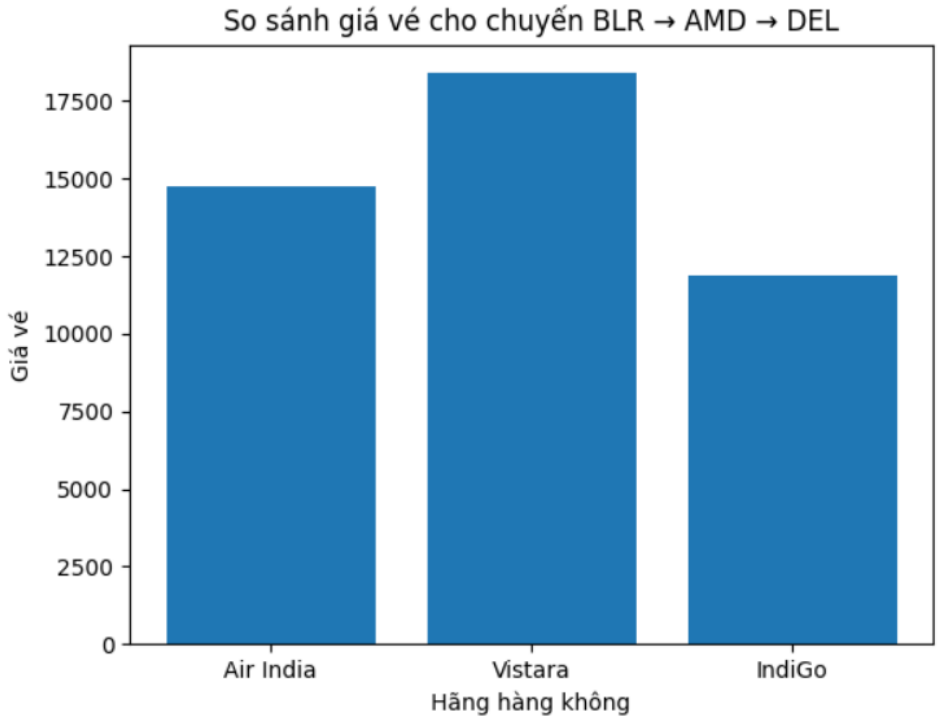
b) Cách 2: so sánh giá theo từng chuyến

- Đầu tiên lọc ra các chuyến bay giống nhau có những giá khác nhau

```
BLR → AMD → DEL                [12599, 10394, 6590, 18387, 13859, 10008, 1247...
BLR → BBI → DEL                                        [12704, 11654, 8714]
BLR → BDQ → DEL                          [28097, 11297, 9623, 6163, 5853, 10037]
BLR → BOM → AMD → DEL                              [17345, 10573, 17135, 11570]
BLR → BOM → BHO → DEL          [25430, 11518, 10783, 12515, 11728, 13250, 264...
                                                          ...
DEL → PNQ → COK                [5830, 5264, 4812, 10112, 4760, 8488, 6356, 51...
DEL → RPR → NAG → BOM → COK              [10493, 15586, 9128, 10703, 12383, 11543]
DEL → TRV → COK                [7165, 8425, 5947, 6587, 7480, 6937, 12677, 11...
DEL → UDR → BOM → COK          [8708, 13082, 13538, 20747, 12121, 15812, 1191...
MAA → CCU                      [4667, 3687, 7414, 6297, 3332, 3540, 3597, 385...
```

- Chọn random 1 trong các chuyến có giá khác nhau giữa các hãng vừa lọc ở trên để visualize

...



- Nhận xét : Chọn chuyến từ BLR-> AMD -> DEL thì có 3 hãng có tuyến bay này và giá của 3 hãng là khác nhau với Vistara có giá vé cao nhất và IndiGo có giá vé thấp nhất

c) Cách 3: Xem dưới dạng Table:

Airline	Air Asia	Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	Multiple carriers Premium economy	SpiceJet	Trujet	Vistara	Vistara Premium economy
Route												
BLR → AMD → DEL	NaN	12599.0	NaN	7708.0	NaN	NaN	NaN	NaN	NaN	NaN	6590.0	NaN
BLR → BOM → DEL	NaN	8714.0	NaN	22153.0	11087.0	79512.0	NaN	NaN	NaN	NaN	NaN	NaN
BLR → BOM → JDH → DEL	NaN	12778.0	NaN	NaN	11245.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BLR → DEL	3383.0	6121.0	3898.0	3897.0	7229.0	NaN	NaN	NaN	3527.0	NaN	4668.0	5969.0
BLR → GOI → DEL	NaN	8767.0	7657.0	7280.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BLR → HYD → DEL	NaN	4943.0	NaN	8738.0	NaN	NaN	NaN	NaN	6315.0	NaN	NaN	NaN
BLR → MAA → DEL	NaN	5932.0	NaN	8434.0	11507.0	57209.0	NaN	NaN	NaN	NaN	NaN	NaN
BOM → DEL → HYD	NaN	11678.0	NaN	NaN	19595.0	NaN	NaN	NaN	NaN	NaN	12395.0	NaN
BOM → HYD	NaN	3625.0	NaN	4049.0	5678.0	NaN	NaN	NaN	1965.0	NaN	NaN	NaN
BOM → IDR → DEL → HYD	NaN	13253.0	NaN	NaN	20845.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BOM → JDH → DEL → HYD	NaN	25139.0	NaN	NaN	23843.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CCU → BBI → BLR	5162.0	7662.0	NaN	4226.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CCU → BLR	4409.0	6245.0	3514.0	4174.0	NaN	NaN	NaN	NaN	3873.0	NaN	NaN	NaN
CCU → BOM → BLR	NaN	8366.0	6686.0	9319.0	9663.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CCU → DEL → BLR	5192.0	11096.0	NaN	NaN	12121.0	NaN	NaN	NaN	NaN	NaN	9397.0	NaN
CCU → GAU → BLR	NaN	8030.0	NaN	6991.0	9689.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CCU → HYD → BLR	NaN	6117.0	4773.0	5170.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CCU → IXR → DEL → BLR	5192.0	12303.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CCU → MAA → BLR	NaN	6528.0	NaN	5268.0	NaN	NaN	NaN	NaN	4649.0	NaN	NaN	NaN
CCU → PNQ → BLR	NaN	NaN	NaN	7605.0	NaN	NaN	NaN	NaN	6013.0	NaN	NaN	NaN
DEL → AMD → BOM → COK	NaN	14011.0	NaN	NaN	11150.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Biểu đồ cho thấy với cùng chuyến thì sẽ có hãng nào bay và có giá là bao nhiêu dễ dàng so sánh được giá của từng chuyến giữa các hãng
- Nhận xét : Có thể thấy ở nhiều chuyến bay giữa các hãng với nhau thì có sự chênh lệch về giá , ví dụ như chuyến BLR → AMD → DEL có 3 hãng bay và giá của mỗi hãng cũng khác nhau

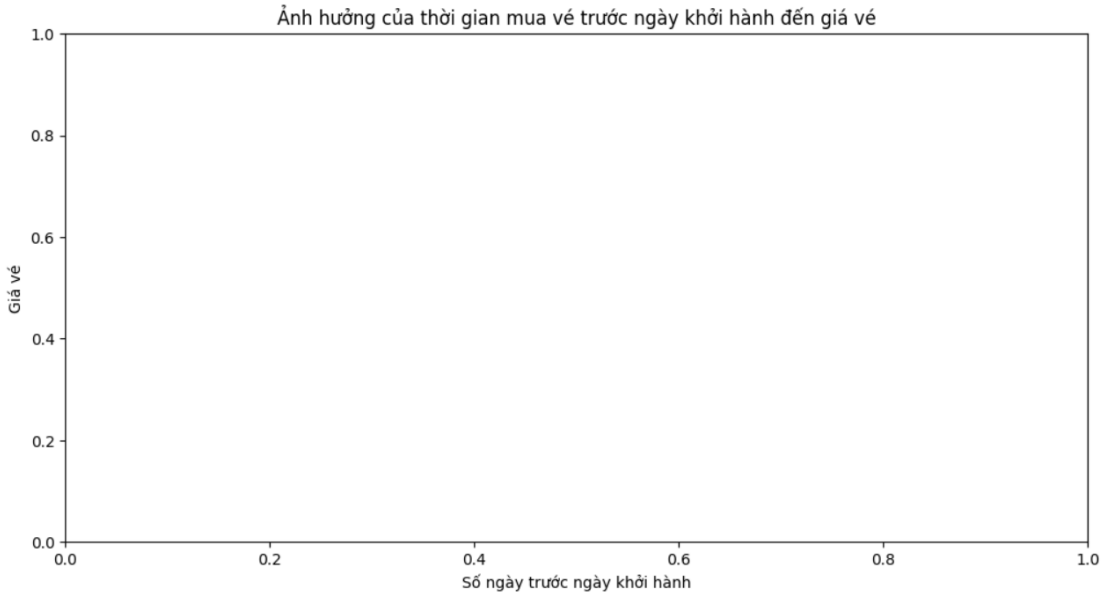
7. Giá vé bị ảnh hưởng như thế nào khi mua vé chỉ 1 hoặc 2 ngày trước ngày khởi hành?

- Đầu tiên để so sánh giá vé có ảnh hưởng như nào khi 1 vé trước 1 2 ngày khởi hành hay không ta lọc ra các giá vé đã bán trước cách đó 1 2 ngày

```
# Tính số ngày giữa ngày hiện tại và ngày khởi hành
df['days_until_departure'] = (df['Date_of_Journey'] - pd.to_datetime('today')).dt.days

# Lọc dữ liệu cho những chuyến bay chỉ 1 hoặc 2 ngày trước ngày khởi hành
df_filtered = df[df['days_until_departure'].isin([-2, -1])]
```

- Vì dữ liệu không có các dữ liệu nào liên quan tới 1 chuyến bay nào có ngày khởi hành gần nhau 1 2 ngày nên giá trị trả về là rỗng không thể đánh giá được giá vé có bị ảnh hưởng hay không



8. Giá vé có thay đổi theo thời gian đi và đến không?

a) Cách 1: Xem giá theo các mốc thời gian dạng bảng

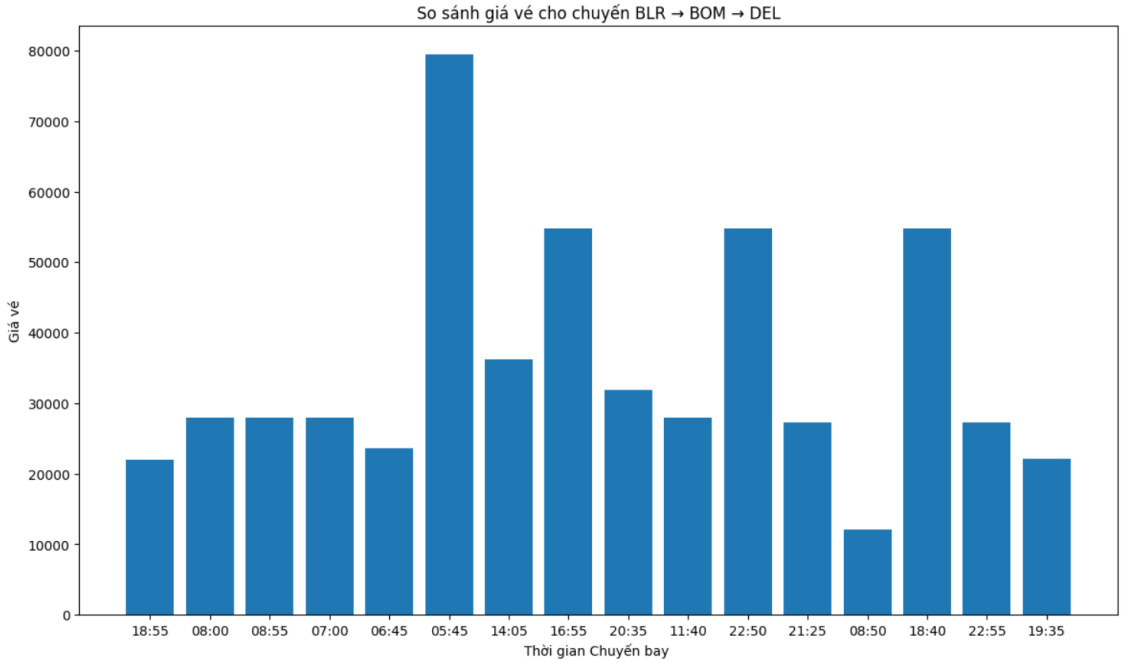
departure_hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Route																					
BLR → AMD → DEL	12599.0	NaN	NaN	NaN	NaN	NaN	6590.0	NaN	NaN	NaN	...	7708.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
BLR → BOM → AMD → DEL	NaN	NaN	NaN	NaN	NaN	NaN	10573.0	NaN	17345.0	NaN	...	NaN	NaN	NaN	11570.0	NaN	NaN	NaN	NaN	NaN	NaN
BLR → BOM → BHO → DEL	NaN	NaN	NaN	NaN	NaN	NaN	12515.0	NaN	25430.0	NaN	...	NaN	NaN	NaN	11518.0	NaN	NaN	NaN	NaN	NaN	NaN
BLR → BOM → DEL	NaN	NaN	NaN	NaN	NaN	9134.0	8714.0	19225.0	22270.0	NaN	...	13712.0	NaN	13817.0	NaN	11087.0	22153.0	11087.0	17471.0	13555.0	NaN
BLR → BOM → IDR → DEL	NaN	NaN	NaN	NaN	NaN	NaN	12599.0	NaN	19372.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
DEL → NAG → BOM → COK	NaN	NaN	NaN	NaN	NaN	7711.0	13376.0	NaN	NaN	NaN	...	10919.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
DEL → PNQ → COK	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5830.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	10112.0
DEL → TRV → COK	NaN	NaN	NaN	NaN	NaN	7165.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5947.0	NaN	NaN
DEL → UDR → BOM → COK	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	15812.0	20747.0	NaN	NaN	NaN	NaN	NaN
MAA → CCU	NaN	NaN	NaN	NaN	NaN	3540.0	NaN	3687.0	3332.0	3543.0	...	3597.0	3850.0	NaN	11982.0	NaN	3597.0	NaN	NaN	5277.0	NaN

74 rows × 24 columns

- Các thời gian được gom lại thành các mốc 0 = 00:00 - 00:59, 1 = 1:00 - 1;59,….
- Nhận xét:
    - Qua bảng trên quan sát được có sự khác nhau về giá giữa các mốc thời gian khởi hành khác nhau
    - Ví dụ như cùng chuyến BLR-> BOM -> AMD- DEL ở mốc 5,6,7,8,…. Có sự khác nhau về giá vé

b) Cách 2: Xem giá theo thời gian của một chuyến

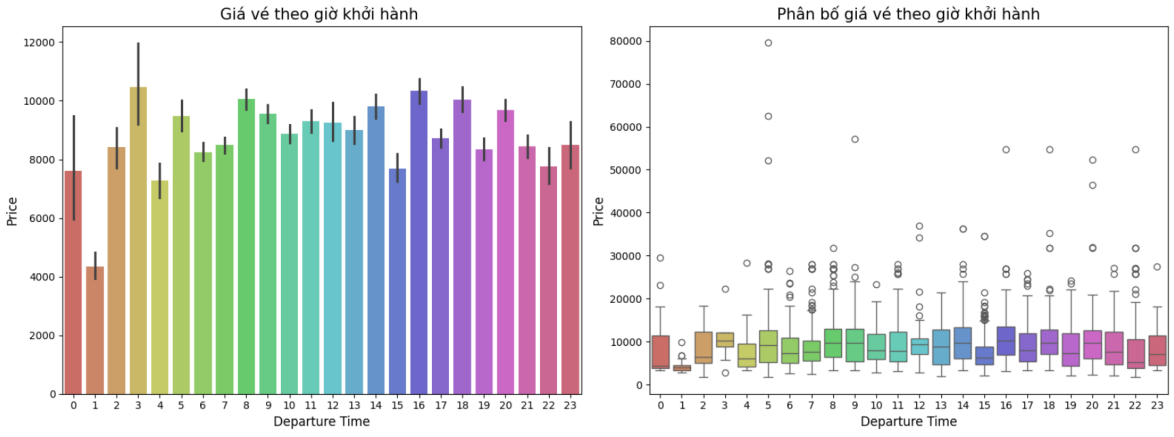
- Lọc ra các chuyến có giá vé khác nhau
- Chọn ra 1 chuyến có giá vé khác nhau và visualize theo thời gian



- Nhận xét : có nhiều khoảng thời gian xuất phát khác nhau của chuyến BLR-> BOM-> DEL và ở các khoảng thời gian khác nhau thì các giá cũng không giống nhau . Ở chuyến này giá vé vào lúc khởi hành từ 05:45 là cao nhất và rẻ nhất vào lúc 08:50

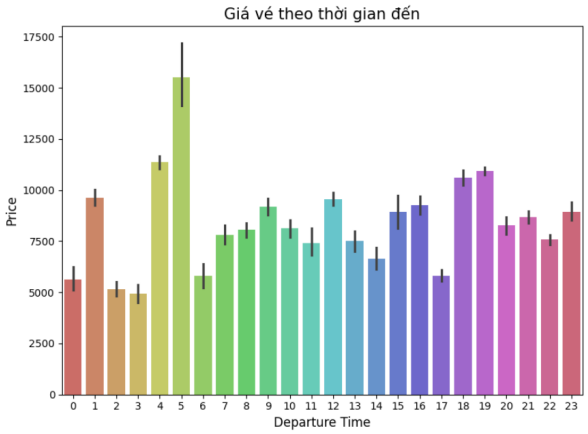
c) Cách 3: Sử dụng biểu đồ Boxplot và barplot

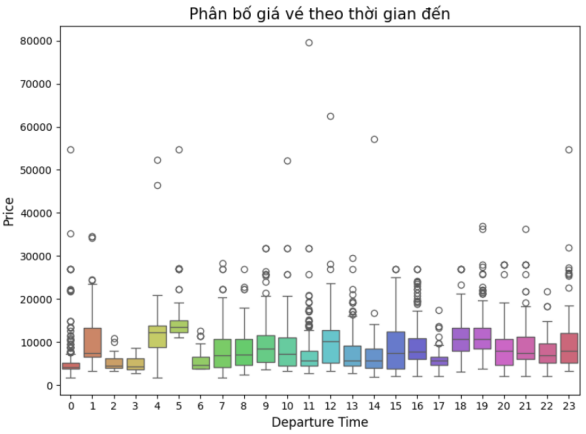
- Đầu tiên cũng gom các giờ thành mốc thời gian như cách 1 (các mốc 0 = 00:00-00:59 , ..)
- Visualize dưới dạng barplot và boxplot trong đó barplot để xem giá vé theo giờ và boxplot để xem phân bố giá vé theo giờ



- Nhận xét theo thời gian khởi hành :
  - Có sự khác nhau về giá vé giữa các giờ khác nhau
  - Vào khoảng thời gian khởi hành từ lúc 1:00 - 1:59 giá vé là rẻ nhất và cũng có phân bố giá hẹp nhất nghĩa là giá trong khoảng đó không có sự chênh lệch quá lớn

- Vào khoảng thời gian khởi hành từ lúc 3:00-3:59 có giá cao hơn so với tất cả các khoảng giờ còn lại và phân bố giá của khoảng thời gian này cũng nằm ở mức cao hơn so các khoảng giờ kia

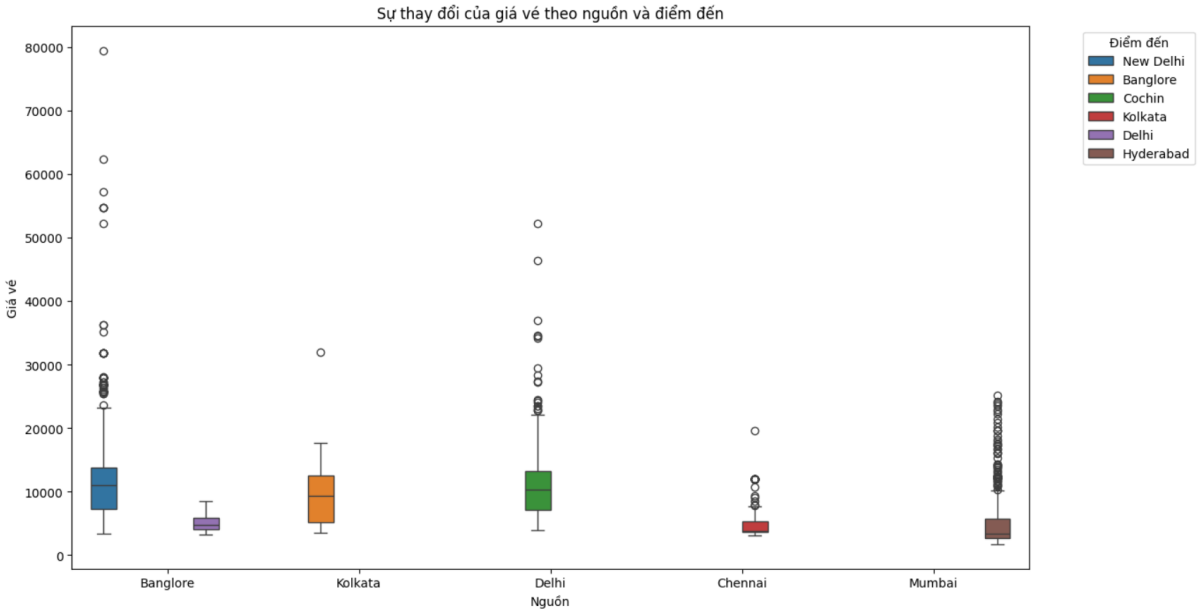




- Nhận xét theo thời gian đến :
  - Có sự khác nhau về giá vé giữa các giờ khác nhau
  - Vào khoảng thời gian đến lúc 3:00 - 3:59 giá vé là rẻ nhất
  - Phân bố giá của thời gian đến lúc 00:00-00:59 là hẹp nhất
  - Vào khoảng thời gian khởi hành từ lúc 5:00-5:59 có giá cao hơn so với tất cả các khoảng giờ còn lại và phân bố giá của khoảng thời gian này cũng nằm ở mức cao hơn so các khoảng giờ kia
-

9. Giá thay đổi như thế nào khi thay đổi Nguồn và Điểm đến?

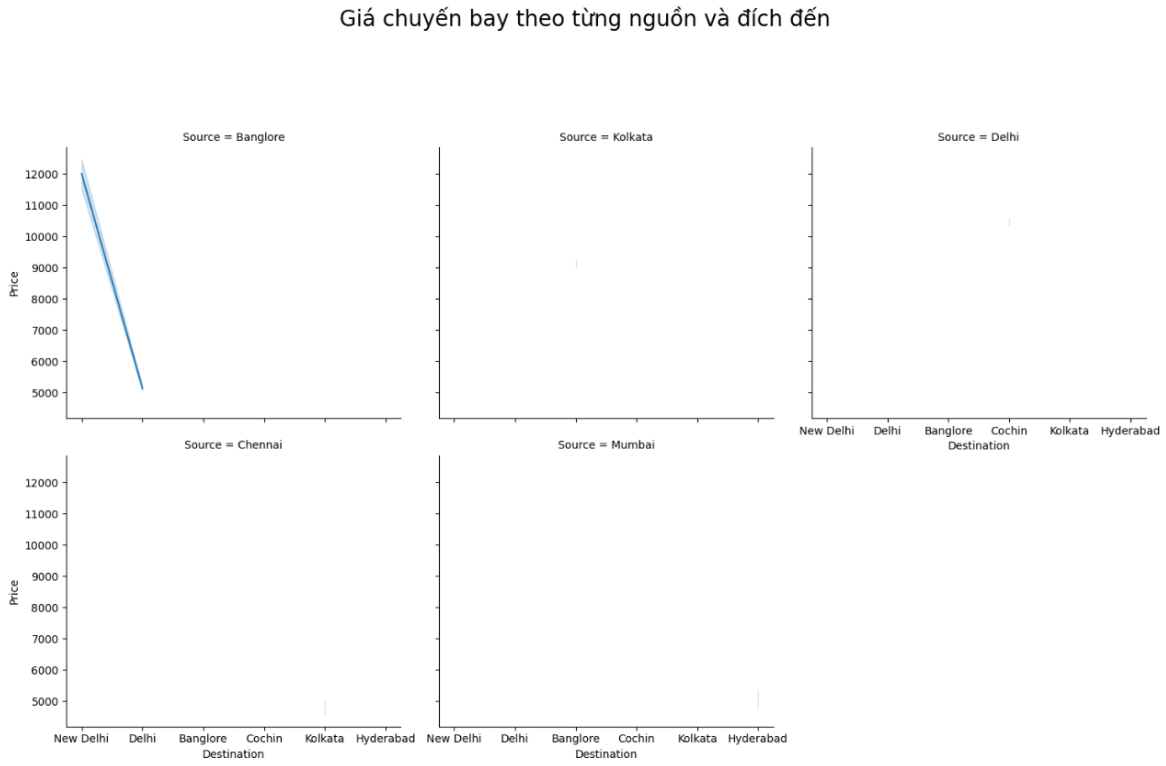
a) Dùng Boxplot để xem phân bố giá theo nguồn đến đích



- Dùng Boxplot để visualize dữ liệu với x là Source và hue = Destination
- Nhận xét :

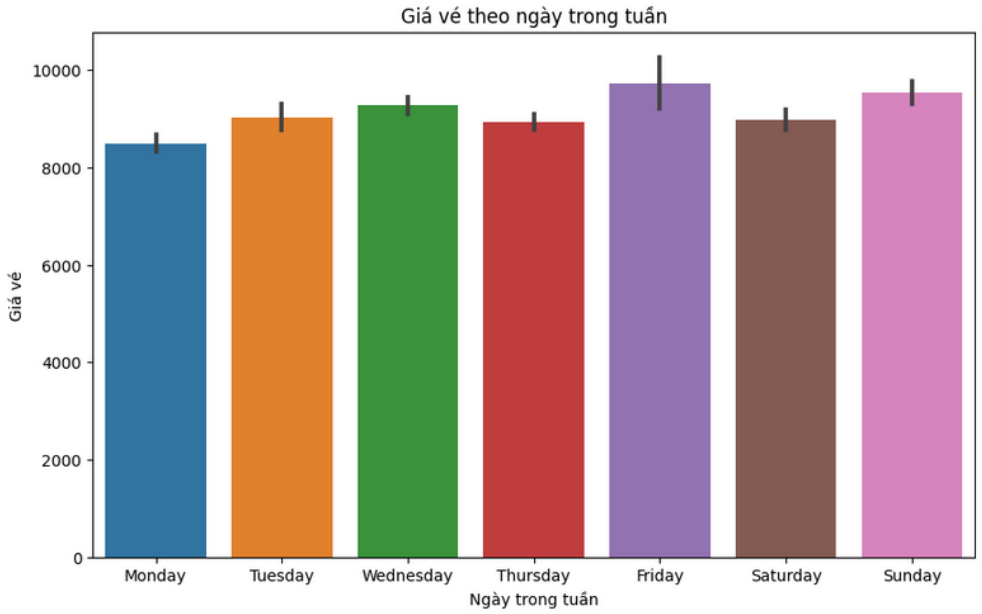
- Có 5 nguồn và chỉ có Banglore là có 2 điểm đến trong tập dữ liệu
- Phân bố giá cùng nguồn Banglore giữa 2 đích đến là New Delhi và Delhi có sự chênh lệch về giá. Phân bố của Delhi Thấp hơn so với New Delhi
- Giữa các nguồn khác nhau và đích đến cũng khác nhau thì phân bố giá cũng khác nhau cụ thể là nguồn Mumbai đến hyderabad có min giá thấp nhất so với min giá các nguồn và đích còn lại nhưng phân bố giá của nguồn này lại rộng
- Không có Nguồn và Đích cùng chung chuyến , ví dụ như nguồn là Banglore đích là NewDelhi và ngược lại nguồn là New Delhi đích là Banglore nên không có sự so sánh về giá giữa khi đảo nguồn và đích

b) Dùng Relplot để so sánh riêng từ nguồn và đích



- Nhận xét: do dữ liệu chỉ có một nguồn và một đích nên biểu đồ sẽ không biểu thị so sánh được nếu cùng nguồn , riêng có nguồn Banglore là có 2 đích nên ta thấy được giá từ Banglore đến New Delhi cao hơn so với giá đến Delhi

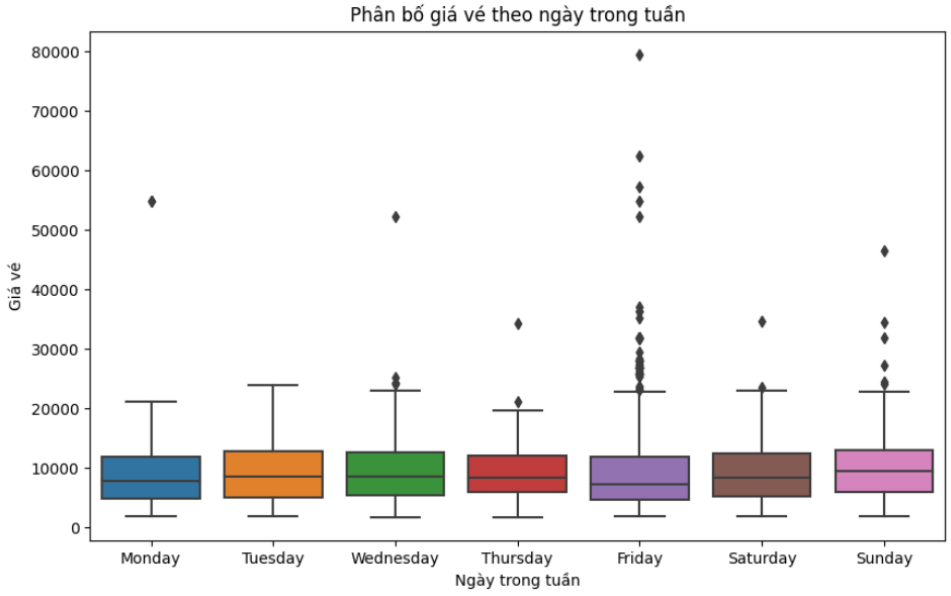
- Ngoài các đặc trưng trong tập dữ liệu, các yếu tố nào có khả năng ảnh hưởng đến giá vé chuyến bay ?



a) Giá vé thay đổi theo ngày trong tuần

Dùng bar và Boxplot để visualize dữ liệu với x là ngày trong tuần và y là giá vé
Nhận xét :
Qua biểu đồ cho thấy có sự phân bố khác nhau của giá vé ngày trong tuần

- Ngày thứ sáu là ngày có giá vé cao hơn so với các ngày khác trong tuần , Ngày thứ 2 là ngày có giá vé thấp nhất.



Qua biểu đồ boxplot:

- Vào ngày thứ sáu có nhiều vé có giá rất cao so với các ngày khác .

- Dự đoán giá vé
  - Chuyển đổi các cột có thuộc tính là object sang dữ liệu số bằng label encoding

```
le=LabelEncoder()
for col in merged_data.columns:
    if merged_data[col].dtype=='object':
        merged_data[col]=le.fit_transform(merged_data[col])
merged_data
```

  - Dữ liệu sau khi Label Encoding :

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Year	Month	Day	days_until_departure	departure_hour	arrival_hour	Day_of_Week
0	3	2019-03-24	0	5	18	211	233	240	4	8	3897	2019	3	24	-1746	22	1	3
1	1	2019-05-01	3	0	84	31	906	336	1	8	7662	2019	5	1	-1708	5	13	6
2	4	2019-06-09	2	1	118	70	413	106	1	8	13882	2019	6	9	-1669	9	4	3
3	3	2019-05-12	3	0	91	164	1324	311	0	8	6218	2019	5	12	-1697	18	23	3
4	3	2019-03-01	0	5	29	149	1237	303	0	8	13302	2019	3	1	-1769	16	21	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0679	1	2019-03-27	2	1	118	45	1124	26	1	8	10913	2019	3	27	-1743	7	19	6
0680	3	2019-06-27	1	4	127	208	114	234	4	8	5277	2019	6	27	-1651	22	0	4
0681	6	2019-05-21	2	1	104	80	1217	7	0	8	7485	2019	5	21	-1688	10	21	5
0682	4	2019-03-24	3	0	66	150	1004	160	0	8	14231	2019	3	24	-1746	16	16	3
0683	4	2019-03-24	3	0	66	150	1004	160	0	8	14231	2019	3	24	-1746	16	16	3

  - Đánh giá các biến độc lập và xây dựng mô hình hồi quy bằng statsmodel

```
X = merged_data.drop(['Price','Date_of_Journey','Dep_Time','Arrival_Time'], axis=1)
X = sm.add_constant(X)
y = merged_data['Price']
lin_reg = sm.OLS(y, X).fit()
print(lin_reg.summary())
X
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.389
Model:                            OLS   Adj. R-squared:                  0.389
Method:                 Least Squares   F-statistic:                     523.6
Date:                Tue, 02 Jan 2024   Prob (F-statistic):               0.00
Time:                        07:46:57   Log-Likelihood:            -1.0266e+05
No. Observations:               10684   AIC:                         2.053e+05
Df Residuals:                   10670   BIC:                         2.054e+05
Df Model:                          13
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
Airline                    -13.6795     14.923     -0.917      0.359     -42.932      15.573
Source                    -400.1473     38.979    -10.266      0.000    -476.554    -323.740
Destination                78.8388     33.500      2.353      0.019      13.172     144.506
Route                      -1.0842      1.200     -0.903      0.366      -3.437       1.269
Duration                   -2.5223      0.337     -7.488      0.000      -3.183      -1.862
Total_Stops             -1520.3818     23.463    -64.799      0.000   -1566.374   -1474.390
Additional_Info           129.3474     29.676      4.359      0.000      71.176     187.518
Year                    -2116.5228    168.011    -12.598      0.000   -2445.855   -1787.190
Month                    7.014e+04   5579.554     12.570      0.000    5.92e+04    8.11e+04
Day                      2220.3395    181.505     12.233      0.000    1864.555    2576.124
days_until_departure    -2303.1145    182.198    -12.641      0.000   -2660.257   -1945.972
departure_hour             34.7835      6.159      5.648      0.000      22.711      46.856
arrival_hour              -20.0190      5.124     -3.907      0.000     -30.062      -9.976
Day_of_Week              -128.8750     18.129     -7.109      0.000    -164.412     -93.338
```

Chọn tập các thuộc tính làm các biến độc lập ngoại trừ price,date_of_journey, dep_time, arrival_time

Kết quả thấy được các biến có ảnh hưởng đến mô hình , nhưng giá trị R-square của mô hình trên khá thấp 0.368 nên mô hình chưa thực sự tốt để sử dụng dự đoán

- Sử dụng 3 model khác để đánh giá : RandomForestRegressor, ExtraTreesRegressor, DecisionTreeRegressor

```
X = merged_data.drop(['Price','Date_of_Journey','Dep_Time','Arrival_Time'], axis=1)
y = merged_data['Price']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=42)

#scale dữ liệu biến độc lập train và test về 0-1
mmscaler=MinMaxScaler(feature_range=(0,1))
x_train=mmscaler.fit_transform(X_train)
x_test=mmscaler.fit_transform(X_test)
x_train=pd.DataFrame(x_train)
x_test=pd.DataFrame(x_test)

# Khởi tạo mô hình
rf_model = RandomForestRegressor()
et_model = ExtraTreesRegressor()
dt_model = DecisionTreeRegressor()
ln_model = LinearRegression()


# Fit dữ liệu vào mô hình
rf_model.fit(X_train, y_train)
et_model.fit(X_train, y_train)
dt_model.fit(X_train, y_train)
ln_model.fit(X_train, y_train)


# Dự đoán
rf_pred = rf_model.predict(X_test)
et_pred = et_model.predict(X_test)
dt_pred = dt_model.predict(X_test)
ln_pred = ln_model.predict(X_test)
```

- Kết quả thu được

```
RandomForestRegressor Evaluation:
MAE: 644.91
MSE: 2891853.23
RMSE: 1700.54
R^2: 0.8641

ExtraTreesRegressor Evaluation:
MAE: 646.11
MSE: 2792989.18
RMSE: 1671.22
R^2: 0.8687

DecisionTreeRegressor Evaluation:
MAE: 721.36
MSE: 3793072.35
RMSE: 1947.58
R^2: 0.8217

LinearRegression Evaluation:
MAE: 2611.06
MSE: 13342122.60
RMSE: 3652.69
R^2: 0.3730
```

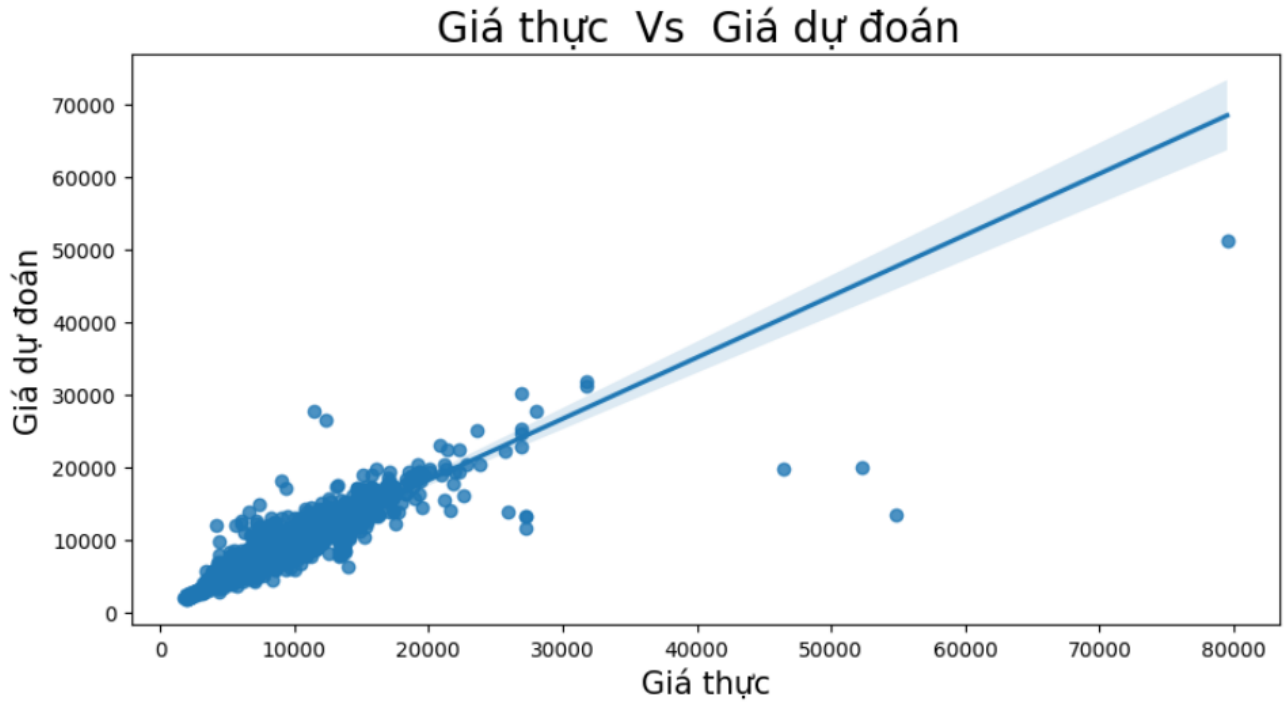
Với 3 mô hình trên thì có thể thấy giá trị R-square đã tăng nhiều hơn so với sử dụng statsmodel và cũng có giá trị cao tiệm cận 1 trong đó giá trị R-square của RandomForest và ExtraTrees cao hơn DecisionTree và LinearRegression và RandomForest có MAE nhỏ hơn các MAE của các mô hình còn lại .Vì vậy RandomForest phù hợp nhất để sử dụng

- Dự đoán và xem kết quả dự đoán bằng RandomForest

```
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)
out=pd.DataFrame({'Price_actual':y_test,'Price_pred':y_pred})
result=merged_data.merge(out,left_index=True,right_index=True)


plt.figure(figsize=(10,5))
sns.regplot(x='Price_actual',y='Price_pred',data=result)
plt.title('Giá thực  Vs  Giá dự đoán ',fontsize=20)
plt.xlabel('Giá thực',fontsize=15)
plt.ylabel('Giá dự đoán',fontsize=15)
plt.show()
```



C. Quá trình phân công công việc