TRƯỜNG ĐẠI HỌC KHOA HỌC XÃ HỘI VÀ NHÂN VĂN, ĐHQG-HCM
KHOA THƯ VIỆN-THÔNG TIN HỌC
NGÀNH QUẢN LÝ THÔNG TIN

--------📖--------

ĐỒ ÁN CUỐI KỲ

ĐỀ TÀI: ĐỊNH GIÁ VÉ MÁY BAY THEO LỊCH TRÌNH VÀ LOẠI BAY

MÔN: KHOA HỌC DỮ LIỆU & ỨNG DỤNG

Giảng viên: Th.S Trần Đình Anh Huy

STT	MSSV	HỌ & TÊN	EMAIL
01	1956210100	Nguyễn Minh Tuấn	1956210100@hcmussh.edu.vn
02	1956210102	Tạ Thị Diệu Thắm	1956210102@hcmussh.edu.vn
03	1956210109	Phạm Thu Trang	1956210109@hcmussh.edu.vn

Thành phố Hồ Chí Minh, ngày 17 tháng 6 năm 2022

MỤC LỤC

MỤC LỤC **2**

TÓM TẮT **3**

I. TỔNG QUAN ĐỀ TÀI **4**

1. Giới thiệu dữ liệu **4**

2. Mô tả dataset **4**

II. PHÂN TÍCH DỮ LIỆU THEO YÊU CẦU **6**

1. Mô tả dữ liệu **6**

2. Mối liên quan giữa thông số của các chuyến bay và giá tiền của chúng **12**

3. Xây dựng mô hình định giá vé máy bay theo yêu cầu của người dùng **19**

4. Chạy các mô hình máy học liên quan với bộ dữ liệu huấn luyện **24**

5. Kiểm định mô hình với bộ dữ liệu thực tế **26**

III. KẾT LUẬN **28**

TÀI LIỆU THAM KHẢO **28**

TÓM TẮT

Ngày nay việc di chuyển bằng đường hàng không đã trở nên rất phổ biến tại Việt Nam nói riêng và trên thế giới nói chung. Đây là phương tiện di chuyển thuận tiện và nhanh nhất. Giá vé máy bay thay đổi theo các yếu tố trong thực tế như hãng hàng không, hạng vé, thời điểm mua vé, dịch vụ hàng không…

Để làm rõ điều đó, đồ án này sẽ thực hiện phân tích tập dữ liệu đặt vé máy bay thu được từ trang web “Ease My Trip” và thực hiện các thử nghiệm giả thuyết thống kê khác nhau để có được thông tin có ý nghĩa từ đó. Thuật toán thống kê 'Hồi quy tuyến tính' sẽ được sử dụng để đào tạo tập dữ liệu và dự đoán một biến mục tiêu liên tục.

'Easemytrip' là một nền tảng internet để đặt vé máy bay và do đó là một nền tảng mà hành khách tiềm năng sử dụng để mua vé. Nghiên cứu kỹ lưỡng về dữ liệu sẽ giúp khám phá những hiểu biết sâu sắc có giá trị to lớn đối với hành khách và mục tiêu nghiên cứu của nhóm sẽ là thực hiện các yêu cầu dưới đây

- Tìm ra mối liên quan giữa thông số của các chuyến bay và giá tiền của chúng
- Xây dựng mô hình định giá vé máy bay theo yêu cầu của người dùng
- Chạy các mô hình máy học liên quan với bộ dữ liệu huấn luyện
- Kiểm định mô hình với bộ dữ liệu thực tế

I. TỔNG QUAN ĐỀ TÀI

1. Giới thiệu dữ liệu

Tên dataset: Flight Price Prediction

Nội dung: Dataset này là một tập dữ liệu được trích xuất từ trang web "Ease My Trip" – một nền tảng internet được sử dụng để khách hàng đặt vé máy bay. Dữ liệu đã được thu thập theo 2 hạng vé là hạng phổ thông (Economy) và hạng thương gia (Business). Ngoài ra, dataset cũng chứa thông tin cụ thể về các tùy chọn đặt vé cho những chuyến bay giữa 6 thành phố lớn của Ấn Độ, bao gồm: Bangalore, Chennai, Delhi, Hyderabad, Kolkata và Mumbai.

2. Mô tả dataset

Dataset bao gồm 3 file:

- Clean_Dataset.csv: là tập dữ liệu đã được làm sạch, chứa thông tin tổng hợp của cả 2 hạng vé là hạng phổ thông và thương gia. File dữ liệu có 300153 dòng và 11 cột (ngoại trừ cột số thứ tự).
- business.csv: Tập dữ liệu này chứa thông tin chi tiết về các chuyến bay hạng thương gia.
- economy.csv: Tập dữ liệu này chứa thông tin chi tiết về các chuyến bay hạng phổ thông.

Sau đây là chi tiết các thực thể có trong tập dữ liệu sau khi đã được làm sạch:

STT	TÊN THỰC THỂ	MÔ TẢ
1	Airline	Lưu trữ tên của hãng hàng không. Có tổng cộng 6 hãng hàng không khác nhau trong cột này.
2	Flight	Lưu trữ mã chuyến bay
3	Source City	Nơi khởi hành của chuyến bay. Có 6 thuộc tính tương ứng với 6 thành phố như đã trình bày ở trên

4	Departure Time	Lưu trữ thông tin về thời gian khởi hành. Có 6 mốc thời gian được lưu trữ trong cột này
5	Stops	Lưu trữ số lượng điểm dừng giữa địa điểm khởi hành và điểm đến của các chuyến bay
6	Arrival Time	Lưu trữ thông tin về thời gian hạ cánh của chuyến bay
7	Destination City	Thành phố nơi chuyến bay sẽ hạ cánh
8	Class	Chứa thông tin về hạng ghế của máy bay, ở đây chỉ gồm ghế hạng phổ thông và hạng thương gia
9	Duration	Hiển thị tổng thời gian của một chuyến bay, tính theo giờ
10	Days Left	Số ngày từ thời điểm đặt vé đến khi khởi hành
11	Price	Lưu trữ thông tin về giá vé

II. PHÂN TÍCH DỮ LIỆU THEO YÊU CẦU

1. Mô tả dữ liệu

- *Thêm các thư viện và file dữ liệu vào Google Colab*

```
[1] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import matplotlib.pyplot as plt
    import seaborn as sns
    from plotly.offline import init_notebook_mode, iplot, plot
    import plotly as py
    init_notebook_mode(connected=True)
    import plotly.graph_objs as go
```

```
[2] #from google.colab import files
    #upload = files.upload()
```

```
[3] from google.colab import drive
    drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).
```

- *Đọc dữ liệu*

```
[4] #df = pd.read_csv('Clean_Dataset.csv')
    #df1 = pd.read_csv('business.csv')
    #df2 = pd.read_csv('economy.csv')
```

```
[5] import pandas as pd
    df= pd.read_csv("/content/drive/MyDrive/Colab Notebooks/dulieu/Clean_Dataset.csv")
    df1= pd.read_csv("/content/drive/MyDrive/Colab Notebooks/dulieu/business.csv")
    df2= pd.read_csv("/content/drive/MyDrive/Colab Notebooks/dulieu/economy.csv")
```

```
[6] df.head()
```

	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

```
[7] df1.head()
```

	date	airline	ch_code	num_code	dep_time	from	time_taken	stop	arr_time	to	price
0	11-02-2022	Air India	AI	868	18:00	Delhi	02h 00m	non-stop	20:00	Mumbai	25,612
1	11-02-2022	Air India	AI	624	19:00	Delhi	02h 15m	non-stop	21:15	Mumbai	25,612
2	11-02-2022	Air India	AI	531	20:00	Delhi	24h 45m	1-stop\n\t\t\t\t\t\t\t\t\t\n\t\t\t\t\t\t...	20:45	Mumbai	42,220
3	11-02-2022	Air India	AI	839	21:25	Delhi	26h 30m	1-stop\n\t\t\t\t\t\t\t\t\t\n\t\t\t\t\t\t...	23:55	Mumbai	44,450
4	11-02-2022	Air India	AI	544	17:15	Delhi	06h 40m	1-stop\n\t\t\t\t\t\t\t\t\t\n\t\t\t\t\t\t...	23:55	Mumbai	46,690

```
[8] df2.head()
```

	date	airline	ch_code	num_code	dep_time	from	time_taken	stop	arr_time	to	price
0	11-02-2022	SpiceJet	SG	8709	18:55	Delhi	02h 10m	non-stop	21:05	Mumbai	5,953
1	11-02-2022	SpiceJet	SG	8157	06:20	Delhi	02h 20m	non-stop	08:40	Mumbai	5,953
2	11-02-2022	AirAsia	I5	764	04:25	Delhi	02h 10m	non-stop	06:35	Mumbai	5,956
3	11-02-2022	Vistara	UK	995	10:20	Delhi	02h 15m	non-stop	12:35	Mumbai	5,955
4	11-02-2022	Vistara	UK	963	08:50	Delhi	02h 20m	non-stop	11:10	Mumbai	5,955

- *Liên kết 3 file dữ liệu với nhau và định dạng lại các trường dữ liệu*

LIÊN KẾT CÁC FILE

```
[9]  df1['price'] = df1['price'].str.replace(',', '')
     df1['price'] = df1['price'].astype(int)

     df2['price'] = df2['price'].str.replace(',', '')
     df2['price'] = df2['price'].astype(int)
```

```
[10] df1['stop'] = df1['stop'].str.replace('\n', '')
     df1['stop'] = df1['stop'].str.replace('\t', '')
     df1['stop'].str.strip()

     df2['stop'] = df2['stop'].str.replace('\n', '')
     df2['stop'] = df2['stop'].str.replace('\t', '')
     df2['stop'].str.strip()
```

```
0         non-stop
1         non-stop
2         non-stop
3         non-stop
4         non-stop
           ...
206769     1-stop
206770     1-stop
206771     1-stop
206772     1-stop
206773     1-stop
Name: stop, Length: 206774, dtype: object
```

```
[11] df1['time_taken'] = df1['time_taken'].apply(lambda x: x.split("h")[0])
     df1['time_taken'] = df1['time_taken'].astype(float)

     df2['time_taken'] = df2['time_taken'].apply(lambda x: x.split("h")[0])
     df2['time_taken'] = df2['time_taken'].astype(float)
```

```
[12] df1['time_taken'] = df1['time_taken'].astype(int)
     df2['time_taken'] = df2['time_taken'].astype(int)
```

ĐỊNH DẠNG LẠI NGÀY THÁNG

```
[13] df1['date'] = pd.to_datetime(df1['date'], format='%d-%m-%Y')
     df1.dtypes
```

```
date          datetime64[ns]
airline               object
ch_code               object
num_code               int64
dep_time              object
from                  object
time_taken             int64
stop                  object
arr_time              object
to                    object
price                  int64
dtype: object
```

- *Số lượng dòng, cột của các file data*

SỐ DÒNG, CỘT CỦA CÁC FILE

```
[15] print("Cleand Dataset Shape:", df.shape)
     print("Business Dataset Shape:", df1.shape)
     print("Economy Dataset Shape:", df2.shape)
```

```
Cleand Dataset Shape: (300153, 12)
Business Dataset Shape: (93487, 11)
Economy Dataset Shape: (206774, 11)
```

- *Thêm cột Class (Hạng vé)*

```
df1['class'] = 'Business'
df2['class'] = 'Economy'
```

```
[17] display(df2.head())
```

	date	airline	ch_code	num_code	dep_time	from	time_taken	stop	arr_time	to	price	class
0	2022-02-11	SpiceJet	SG	8709	18:55	Delhi	2	non-stop	21:05	Mumbai	5953	Economy
1	2022-02-11	SpiceJet	SG	8157	06:20	Delhi	2	non-stop	08:40	Mumbai	5953	Economy
2	2022-02-11	AirAsia	I5	764	04:25	Delhi	2	non-stop	06:35	Mumbai	5956	Economy
3	2022-02-11	Vistara	UK	995	10:20	Delhi	2	non-stop	12:35	Mumbai	5955	Economy
4	2022-02-11	Vistara	UK	963	08:50	Delhi	2	non-stop	11:10	Mumbai	5955	Economy

```
[18] df1.head()
```

	date	airline	ch_code	num_code	dep_time	from	time_taken	stop	arr_time	to	price	class
0	2022-02-11	Air India	AI	868	18:00	Delhi	2	non-stop	20:00	Mumbai	25612	Business
1	2022-02-11	Air India	AI	624	19:00	Delhi	2	non-stop	21:15	Mumbai	25612	Business
2	2022-02-11	Air India	AI	531	20:00	Delhi	24	1-stop	20:45	Mumbai	42220	Business
3	2022-02-11	Air India	AI	839	21:25	Delhi	26	1-stop	23:55	Mumbai	44450	Business
4	2022-02-11	Air India	AI	544	17:15	Delhi	6	1-stop	23:55	Mumbai	46690	Business

```
[19] display(df.head())
```

	Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

- *Tạo bảng mới dựa trên sự kết hợp của Business và Economy*

```
[20] new_df=pd.concat([df1, df2])
     new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 300261 entries, 0 to 206773
Data columns (total 12 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   date        300261 non-null  datetime64[ns]
 1   airline     300261 non-null  object        
 2   ch_code     300261 non-null  object        
 3   num_code    300261 non-null  int64         
 4   dep_time    300261 non-null  object        
 5   from        300261 non-null  object        
 6   time_taken  300261 non-null  int64         
 7   stop        300261 non-null  object        
 8   arr_time    300261 non-null  object        
 9   to          300261 non-null  object        
 10  price       300261 non-null  int64         
 11  class       300261 non-null  object        
dtypes: datetime64[ns](1), int64(3), object(8)
memory usage: 29.8+ MB
```

```
[21] new_df.head()
```

	date	airline	ch_code	num_code	dep_time	from	time_taken	stop	arr_time	to	price	class
0	2022-02-11	Air India	AI	868	18:00	Delhi	2	non-stop	20:00	Mumbai	25612	Business
1	2022-02-11	Air India	AI	624	19:00	Delhi	2	non-stop	21:15	Mumbai	25612	Business
2	2022-02-11	Air India	AI	531	20:00	Delhi	24	1-stop	20:45	Mumbai	42220	Business
3	2022-02-11	Air India	AI	839	21:25	Delhi	26	1-stop	23:55	Mumbai	44450	Business
4	2022-02-11	Air India	AI	544	17:15	Delhi	6	1-stop	23:55	Mumbai	46690	Business

- *Đếm số lượng chuyến bay theo ký hiệu*

```
[23] new_df.ch_code.value_counts()
```

```
UK    127859
AI     80894
6E     43120
G8     23177
I5     16098
SG      9011
S5        61
2T        41
Name: ch_code, dtype: int64
```

- *Đếm theo số hiệu*

```
[24] new_df['num_code'].value_counts()
```

0 giây

```
808     3313
706     3235
772     2860
774     2808
720     2650
        ...
1058       1
6474       1
9974       1
405        1
8193       1
Name: num_code, Length: 1255, dtype: int64
```

- *Đếm theo chặng bay*

```
[25] new_df.stop.value_counts()
```

0 giây

```
1-stop                    243603
non-stop                   36044
2+-stop                    13288
1-stopVia IXU               1839
1-stopVia IDR               1398
1-stopVia Patna              674
1-stopVia Indore             381
1-stopVia PAT                354
1-stopVia MYQ                321
1-stopVia Bhubaneswar        301
1-stopVia KLH                284
1-stopVia JGB                193
1-stopVia JRG                175
1-stopVia STV                169
1-stopVia BBI                158
1-stopVia Delhi              153
1-stopVia Hyderabad          143
1-stopVia IXE                120
```

- *Đếm theo thời gian bay*

```
[31] df3.duration.value_counts().sort_values()
```

```
41.50       1
47.08       1
35.67       1
44.50       1
42.00       1
           ...
2.83     2323
2.08     2755
2.75     2879
2.25     4036
2.17     4242
Name: duration, Length: 476, dtype: int64
```

- *Kiểm tra sự đầy đủ dữ liệu*

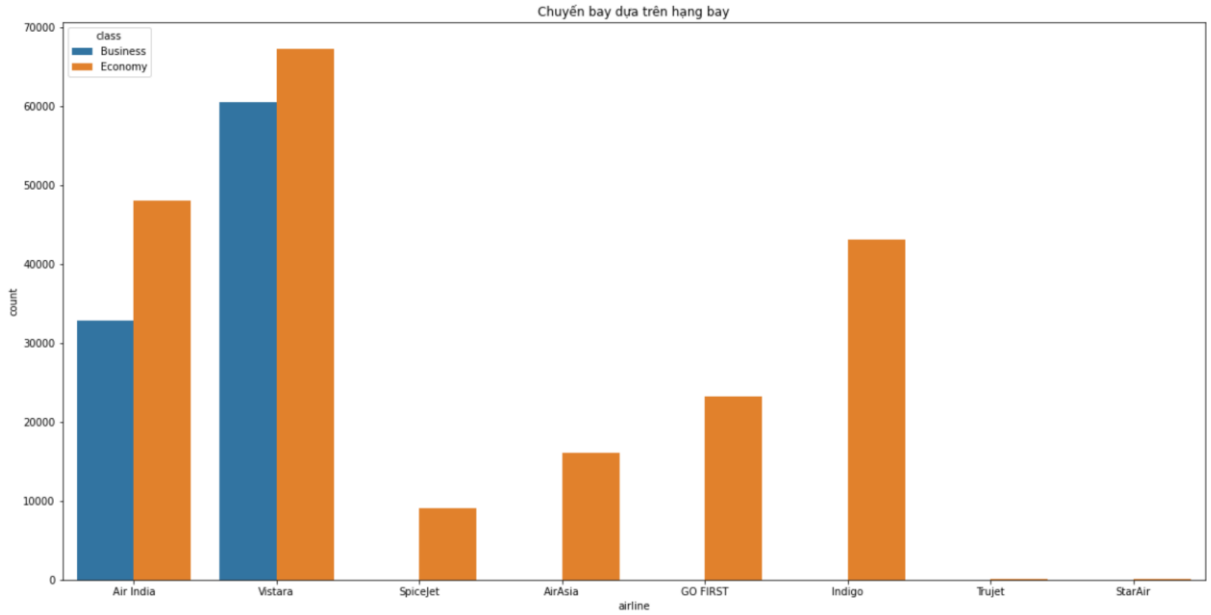
```
[33] df.isnull().sum()
```

```
Unnamed: 0          0
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64
```

2. Mối liên quan giữa thông số của các chuyến bay và giá tiền của chúng

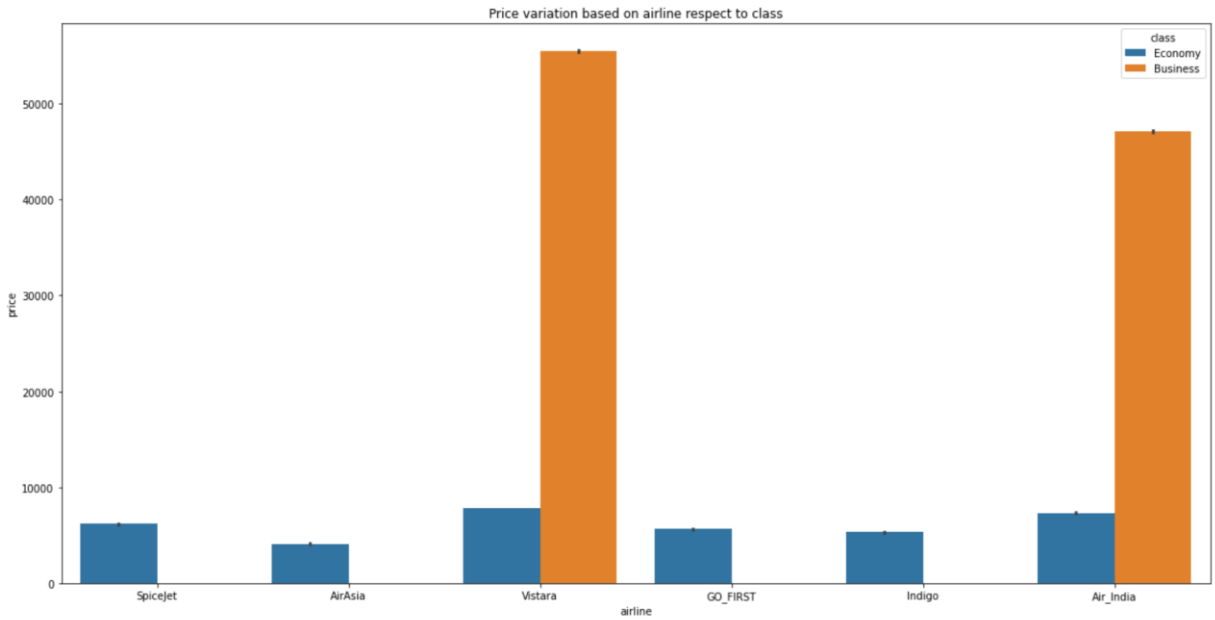
- *Hạng vé đối với giá vé*

```
[39] plt.figure(figsize=(20, 10))
     sns.countplot(x='airline',hue='class' ,data=new_df).set(title='Chuyến bay dựa trên hạng bay')
```



⇨ Số lượng chuyến bay theo cả hạng Thương gia và hạng Phổ thông nhiều nhất là của hãng Vistara, đứng nhì là hãng Air India

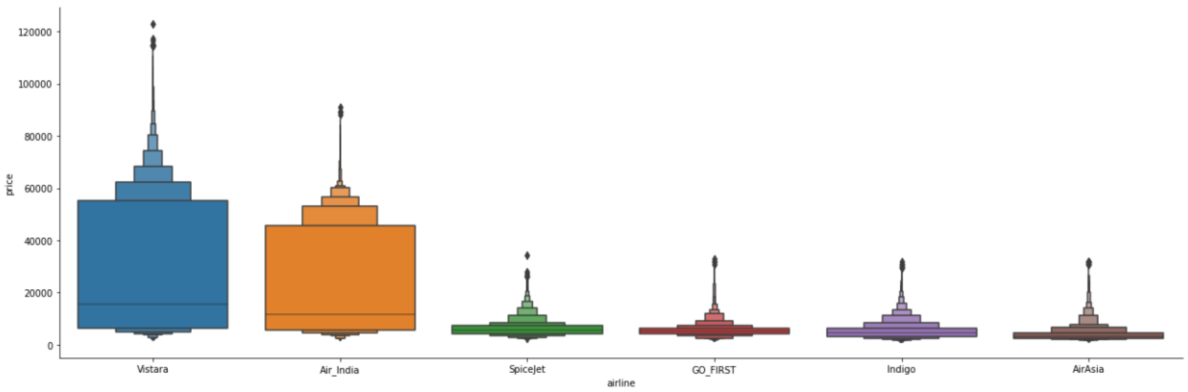
```
[40] plt.figure(figsize=(20, 10))
     sns.barplot(x='airline', y='price',hue='class' ,data=df3).set(title='Price variation based on airline respect to class')
```



⇨ Ở đây ta có thể thấy một số khác biệt lớn về hạng Phổ thông và hạng Thương gia nhưng điều này chỉ xảy ra ở hai hãng hàng không Vistara và Air India.

- *Hãng bay đối với giá vé*

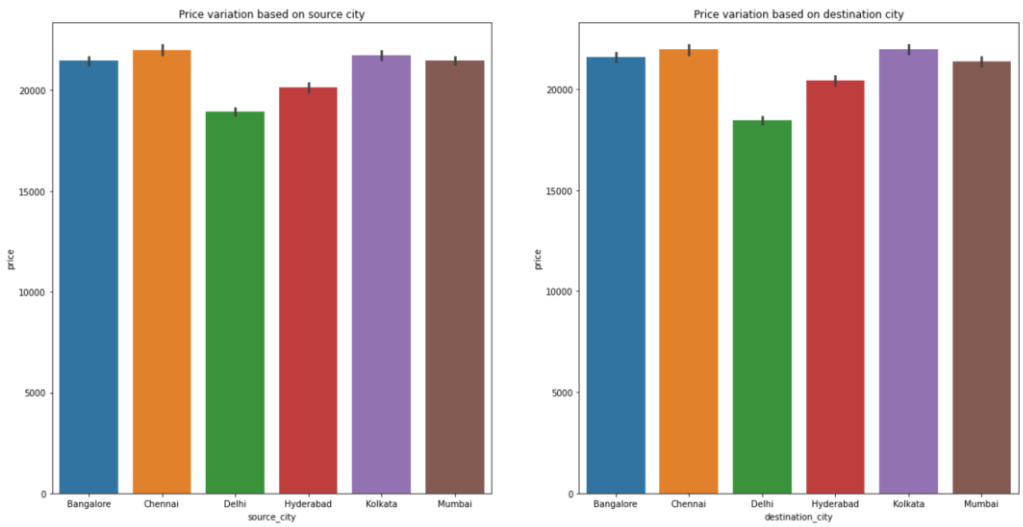
```
sns.catplot(y = "price", x = "airline", data = df3.sort_values("price", ascending = False), kind="boxen", height = 6, aspect = 3)
plt.show()
```



⇨ Giá vé của hãng Vistara là cao nhất, kế đến là Air India
Các hãng còn lại có giá trị vé ở mức tương đương với nhau

- *Nơi xuất phát, nơi hạ cánh đối với giá vé*

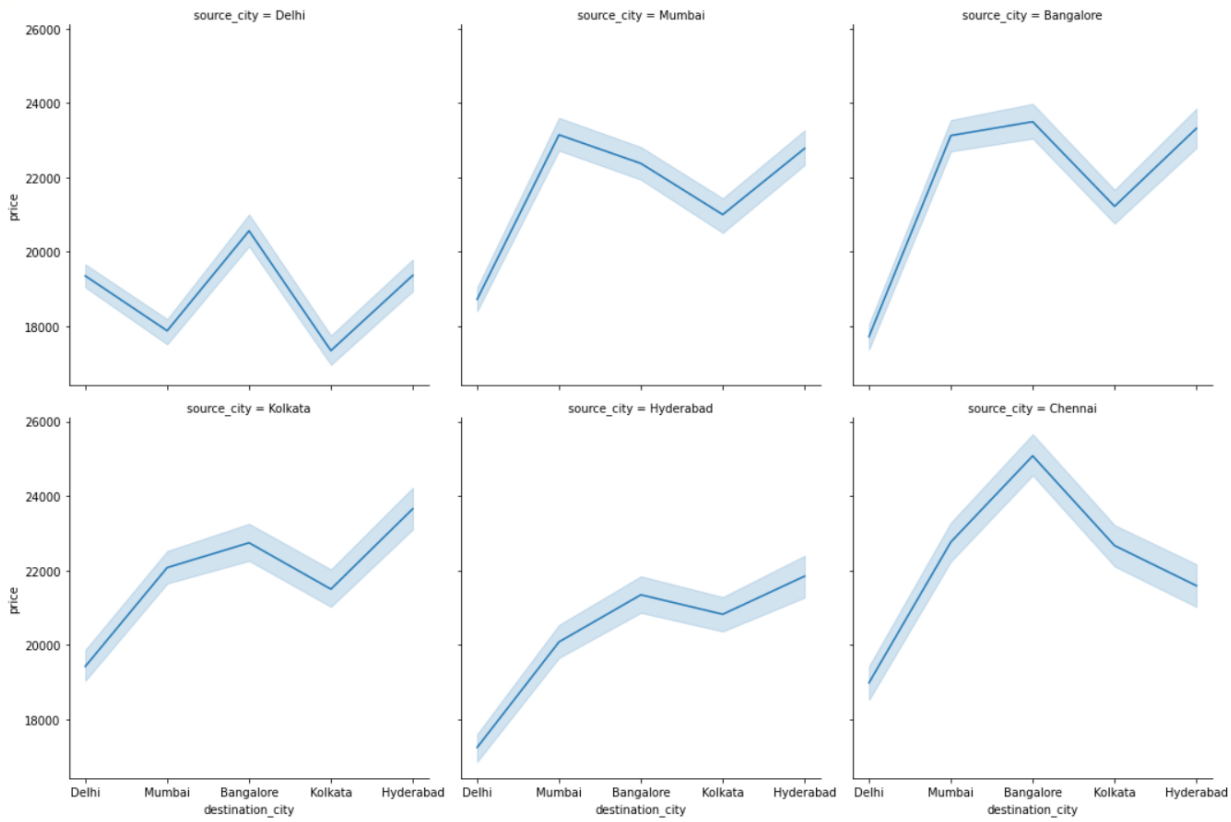
```
[42] fig, axes = plt.subplots(1,2, figsize=(20,10))
     sns.barplot(x='source_city', y='price', data=df3.sort_values('source_city', axis=0), ax=axes[0])
     axes[0].set(title='Price variation based on source city')

     sns.barplot(x='destination_city', y='price', data=df3.sort_values('destination_city', axis=0), ax=axes[1])
     axes[1].set(title='Price variation based on destination city');
```



⇨ Giá vé ở nơi xuất phát và nơi hạ cánh cao nhất là thành phố Chennai và thấp nhất là Delhi

Tuy nhiên sự chênh lệch giá vé giữa các thành phố không quá lớn

```
[43] sns.relplot(col="source_city", y="price", kind="line",x='destination_city', data=df3, col_wrap=3)
```



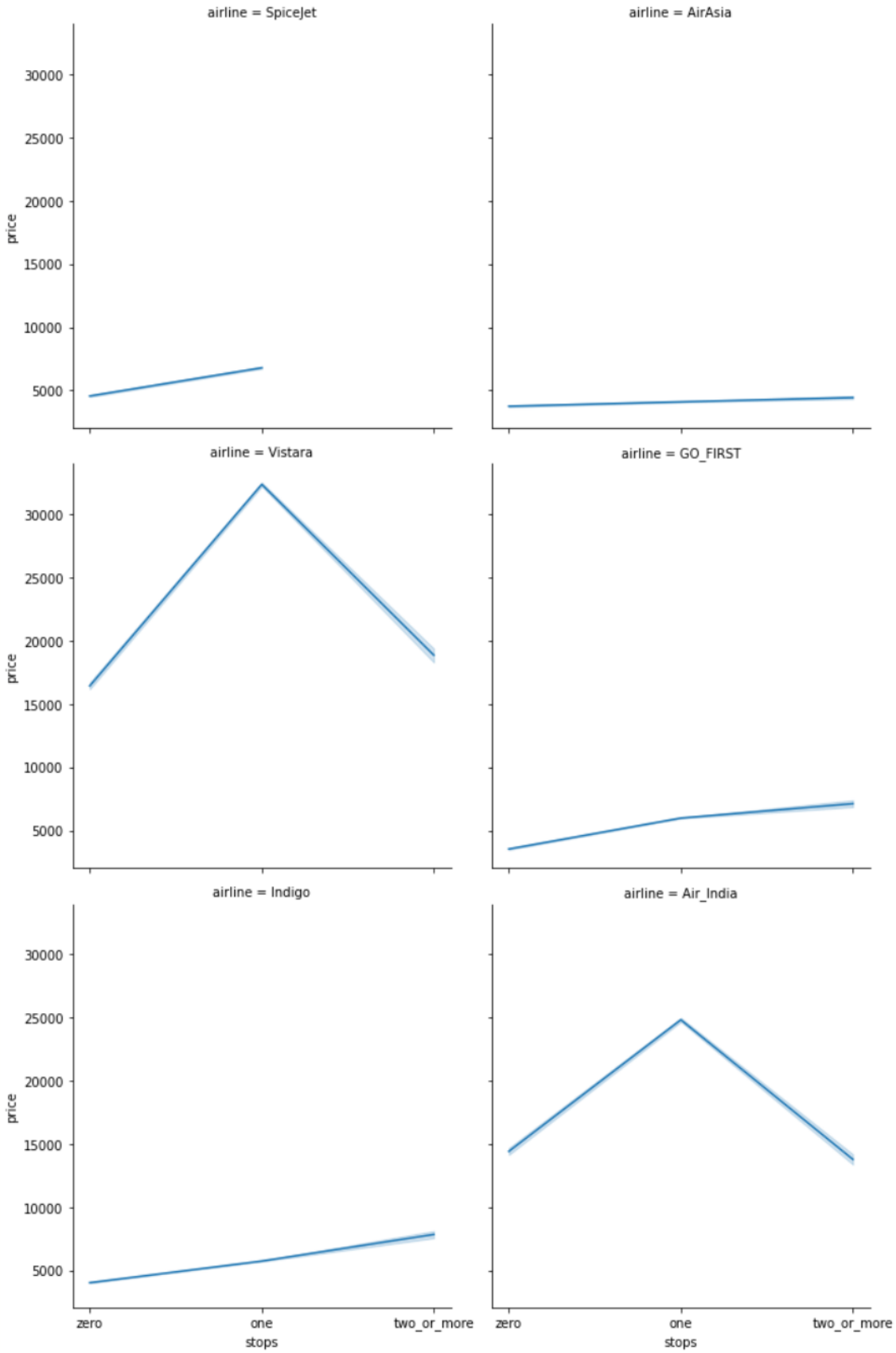
⇨ Biểu đồ cho thấy, cùng một điểm xuất phát tới những điểm hạ cánh khác nhau hoặc ngược lại, giá vé sẽ khác nhau

Khoảng cách giữa 2 thành phố càng cào thì giá vé càng cao

- *Chặng bay đối với giá vé*

```
[44] sns.relplot(col="airline", y="price", kind="line",x='stops', data=df3, col_wrap=2)
```

- Biểu đồ bên dưới giải thích một chút về lý do tại sao Vistara và Air India là những hãng hàng không đắt nhất, vì chúng ta có thể thấy các chuyến bay dừng một chặng là giá cao nhất trong số những hãng hàng không khác


airline = SpiceJet
airline = AirAsia
airline = Vistara
airline = GO_FIRST
airline = Indigo
airline = Air_India
price
30000
25000
20000
15000
10000
5000
zero
one
two_or_more
stops

- *Thời gian đặt vé đối với giá vé*

Ta có thể yếu tố này như sau, đây là khoảng thời gian từ lúc đặt vé tới lúc khởi hành.

```
[47] gby1 = df3.groupby(['source_city', 'destination_city', 'days_left'])['price'].mean()
     gby1
```

```
source_city  destination_city  days_left
Bangalore    Chennai           1            22647.710526
                               2            32438.043956
                               3            30463.322917
                               4            25613.474227
                               5            24440.059322
                                               ...
Mumbai       Kolkata           45           20009.839552
                               46           21356.389961
                               47           20899.023715
                               48           20550.008032
                               49           20710.546875
Name: price, Length: 1470, dtype: float64
```

```
[48] gby1 = gby1.reset_index()
     gby1.index.values
```

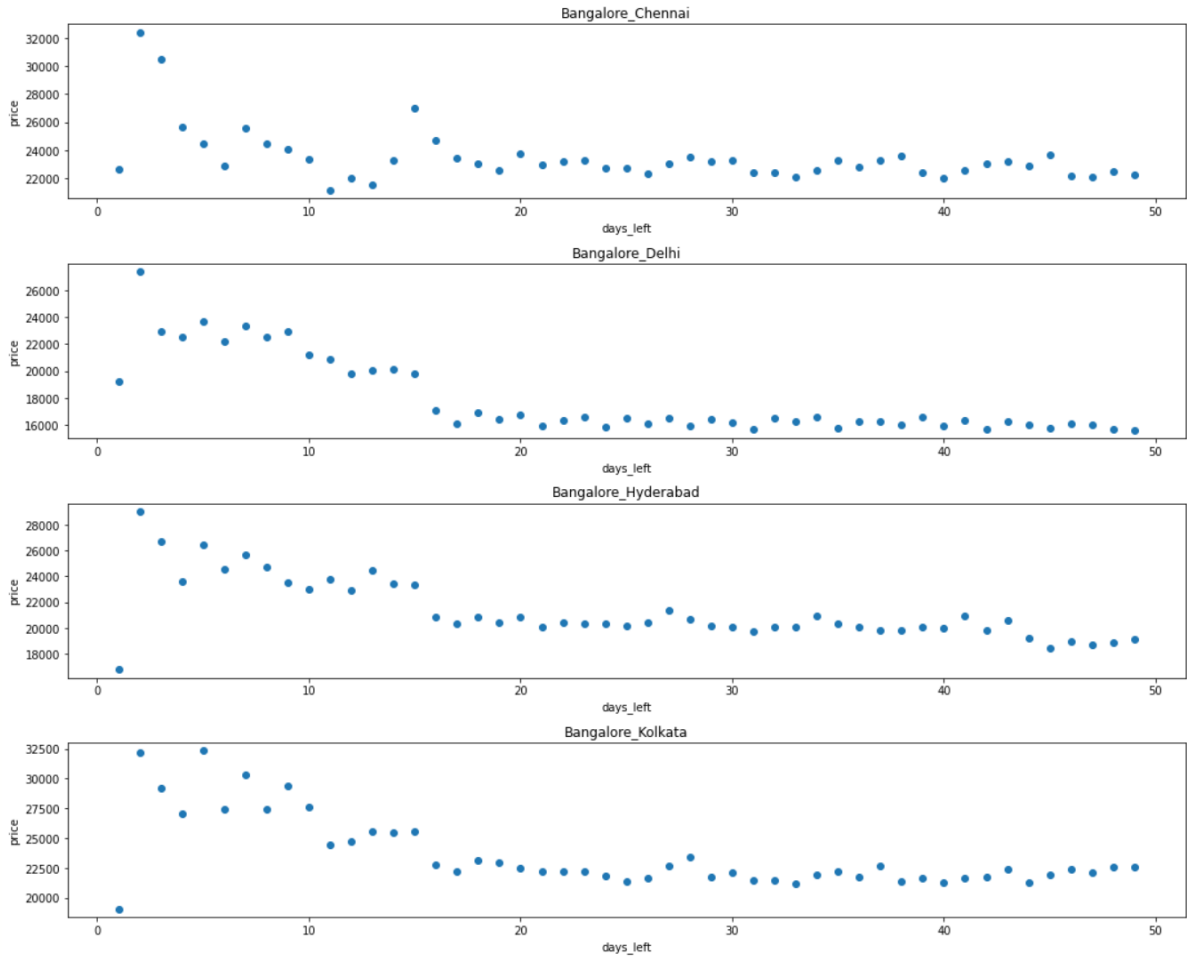
```
array([   0,    1,    2, ..., 1467, 1468, 1469])
```

```
[49] gby1['combined_col'] = gby1['source_city'] + '_' + gby1['destination_city']
```

```
[50] gby1.combined_col.nunique()
```

```
30
```

```
[51] plt.figure(figsize=(15,90))
     for indx, val in enumerate(gby1.combined_col.unique()):
         plt.subplot(30,1,indx+1)
         temp_df = gby1.loc[gby1['combined_col'] == val]
         plt.scatter(temp_df['days_left'], temp_df['price'])
         plt.title(val)
         plt.xlabel('days_left')
         plt.ylabel('price')

     plt.tight_layout()
```

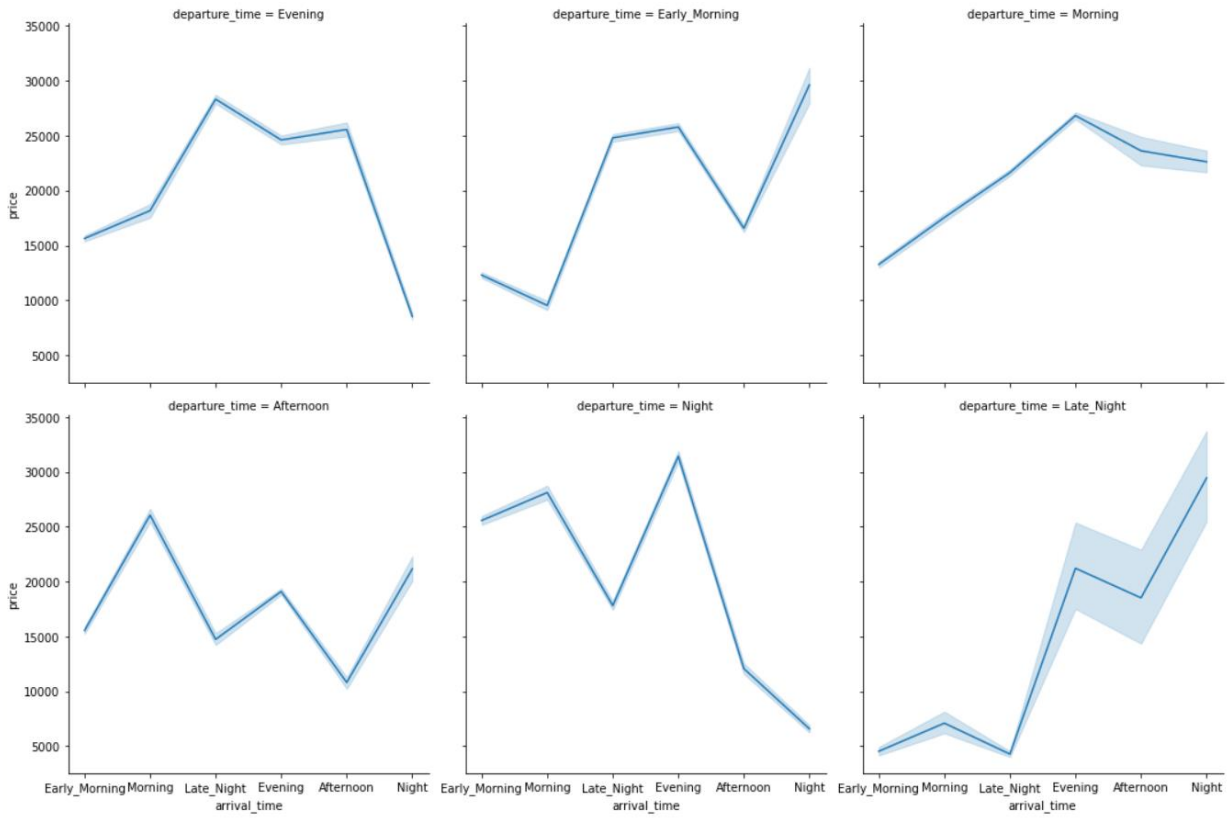


⇨ Trên đây là 4/30 biểu đồ thể hiện thời gian còn lại từ khi đặt vé tới lúc khởi hành

Ta có thể thấy, mức giá thay đổi tùy thuộc vào ngày đặt vé, đặc biệt nếu đặt trước ngày bay một hoặc hai ngày thì giá sẽ càng cao

- *Thời điểm trong ngày đối với giá vé*

```
[52] sns.relplot(col="departure_time", y="price", kind="line",x='arrival_time', data=df3, col_wrap=3)
```



⇨ Giá thay đổi dựa trên thời gian khởi hành và thời gian đến, ta có thể thấy nếu thời gian khởi hành là buổi tối và thời gian đến là ban đêm, thì giá vé sẽ rẻ hơn.

❖ Giá vé máy bay thay đổi phụ thuộc vào nhiều yếu tố: *Hãng máy bay*, *Hạng vé ngồi, Chặng bay, Thời gian bay, Thời gian đặt vé*, *Khoảng cách*… nhưng đối với bộ dữ liệu này ta thấy sự khác biệt rõ nhất là ở yếu tố Hạng vé máy bay, giá vé hạng Thương gia lớn hơn rất nhiều so với hạng Phổ thông. Kế đến là yếu tố hãng Máy bay, Hãng Vistara có mức giá vé cao nhất và thứ nhì là hãng Air India.

3. Xây dựng mô hình định giá vé máy bay theo yêu cầu của người dùng

- *Chuẩn bị dữ liệu*

```
[55] df3.columns
```

```
Index(['airline', 'flight', 'source_city', 'departure_time', 'stops',
       'arrival_time', 'destination_city', 'class', 'duration', 'days_left',
       'price', 'price_per_minute'],
      dtype='object')
```

```
[56] df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           300153 non-null  object 
 1   flight            300153 non-null  object 
 2   source_city       300153 non-null  object 
 3   departure_time    300153 non-null  object 
 4   stops             300153 non-null  object 
 5   arrival_time      300153 non-null  object 
 6   destination_city  300153 non-null  object 
 7   class             300153 non-null  object 
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64  
 10  price             300153 non-null  int64  
 11  price_per_minute  300153 non-null  float64
dtypes: float64(2), int64(2), object(8)
memory usage: 27.5+ MB
```

- *Vì các trường dữ liệu sau đây là dữ liệu phân loại, nên tạo dummy cho nó*

```
[58] airline = df3[["airline"]]

     airline = pd.get_dummies(airline, drop_first= True)

     airline.head()
```

	airline_Air_India	airline_GO_FIRST	airline_Indigo	airline_SpiceJet	airline_Vistara
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	0	0
3	0	0	0	0	1
4	0	0	0	0	1

```
[59]
source = df3[["source_city"]]

source = pd.get_dummies(source, drop_first= True)

source.head()
```

	source_city_Chennai	source_city_Delhi	source_city_Hyderabad	source_city_Kolkata	source_city_Mumbai
0	0	1	0	0	0
1	0	1	0	0	0
2	0	1	0	0	0
3	0	1	0	0	0
4	0	1	0	0	0

```
[60] departure = df3[["departure_time"]]

departure = pd.get_dummies(departure, drop_first= True)

departure.head()
```

	departure_time_Early_Morning	departure_time_Evening	departure_time_Late_Night	departure_time_Morning	departure_time_Night
0	0	1	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	0	0	0	1	0
4	0	0	0	1	0

```
[61] arrival = df3[["arrival_time"]]

arrival = pd.get_dummies(arrival, drop_first= True)

arrival.head()
```

	arrival_time_Early_Morning	arrival_time_Evening	arrival_time_Late_Night	arrival_time_Morning	arrival_time_Night
0	0	0	0	0	1
1	0	0	0	1	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	1	0

```
[62] destination = df3[["destination_city"]]

destination = pd.get_dummies(destination, drop_first= True)

destination.head()
```

	destination_city_Chennai	destination_city_Delhi	destination_city_Hyderabad	destination_city_Kolkata	destination_city_Mumbai
0	0	0	0	0	1
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1

```
[63] df3.replace({"zero": 0, "one": 1, "two_or_more": 2}, inplace = True)
```

```
[64] df3.replace({"Economy": 0, "Business": 1}, inplace = True)
```

- *Tạo bảng kết hợp dữ liệu*

```
[65] df4 = pd.concat([df3, airline, source, departure, arrival, destination], axis = 1)
```

```
[66] df4.head()
```

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	...	arrival_time_Early_Morning	arrival_time_
0	SpiceJet	SG-8709	Delhi	Evening	0	Night	Mumbai	0	2.17	1	...	0	
1	SpiceJet	SG-8157	Delhi	Early_Morning	0	Morning	Mumbai	0	2.33	1	...	0	
2	AirAsia	I5-764	Delhi	Early_Morning	0	Early_Morning	Mumbai	0	2.17	1	...	1	
3	Vistara	UK-995	Delhi	Morning	0	Afternoon	Mumbai	0	2.25	1	...	0	
4	Vistara	UK-963	Delhi	Morning	0	Morning	Mumbai	0	2.33	1	...	0	

5 rows × 37 columns

- *Xoá các cột thừa và cân bằng lại dữ liệu*

```
[67] df4.drop(["airline", "flight", 'source_city', "departure_time", 'arrival_time', 'destination_city', 'price_per_minute'], axis = 1, inplace = True)
```

```
[68] from sklearn.preprocessing import MinMaxScaler

     scaler = MinMaxScaler()
     num_vars = ['duration', 'days_left']
     df4[num_vars] = scaler.fit_transform(df4[num_vars])
```

```
[69] df4.head()
```

	stops	class	duration	days_left	price	airline_Air_India	airline_GO_FIRST	airline_Indigo	airline_SpiceJet	airline_Vistara	...	arrival_time_Early_Morning	arrival_time_Ev
0	0	0	0.027347	0.0	5953	0	0	0	1	0	...	0	
1	0	0	0.030612	0.0	5953	0	0	0	1	0	...	0	
2	0	0	0.027347	0.0	5956	0	0	0	0	0	...	1	
3	0	0	0.028980	0.0	5955	0	0	0	0	1	...	0	
4	0	0	0.030612	0.0	5955	0	0	0	0	1	...	0	

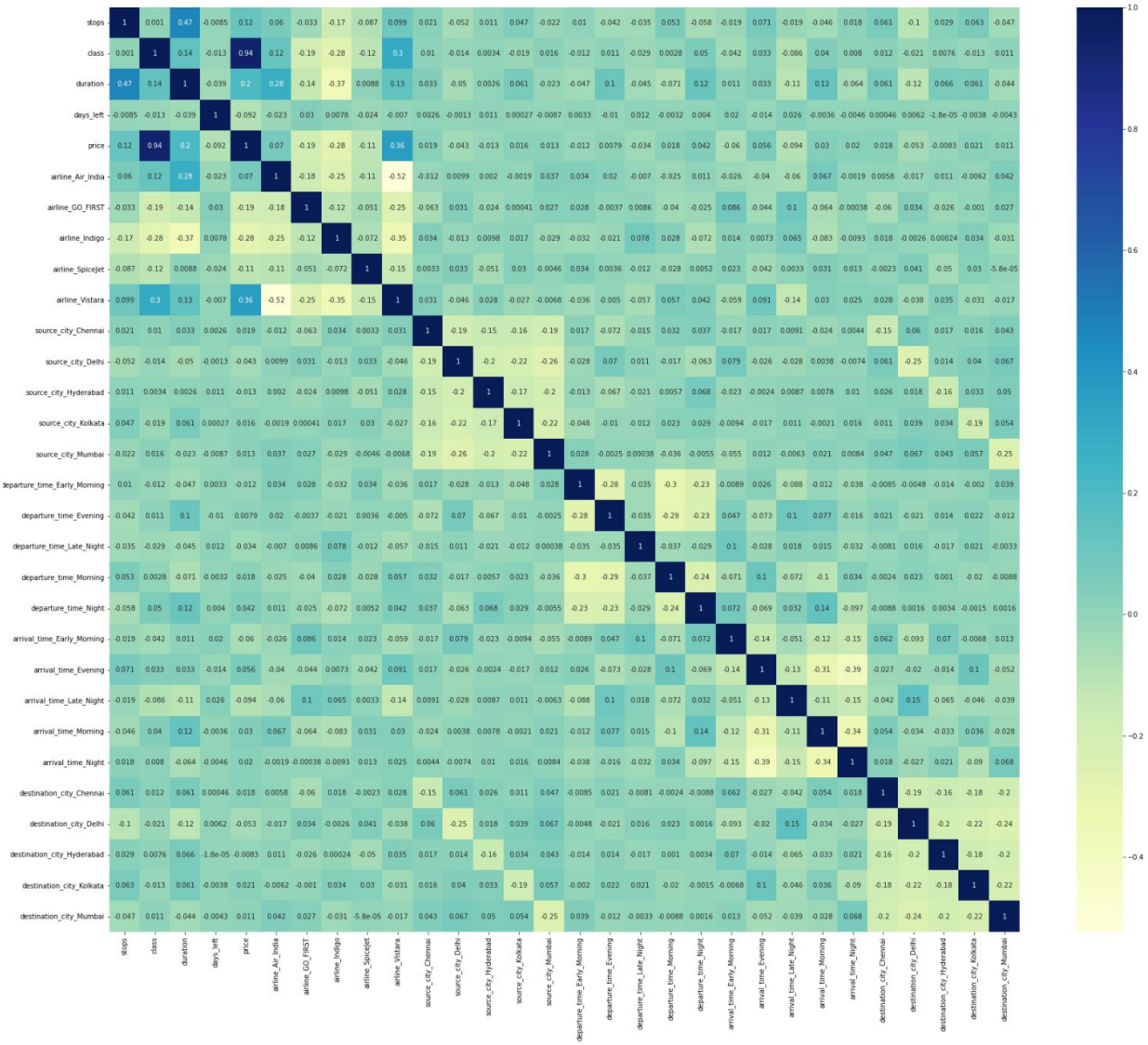
5 rows × 30 columns

- *Xây dựng bộ dữ liệu để tạo mô hình*

```
[70] X = df4.drop(['price'], axis=1)
     y = df4['price']
```

- *Tương quan giữa các biến trong bộ dữ liệu*

```
[73] plt.figure(figsize = (30, 25))
     sns.heatmap(df4.corr(), annot = True, cmap="YlGnBu")
     plt.show()
```



⇨ Biểu đồ trên cho thấy mối liên hệ giữa Giá vé và Hạng vé rất lớn

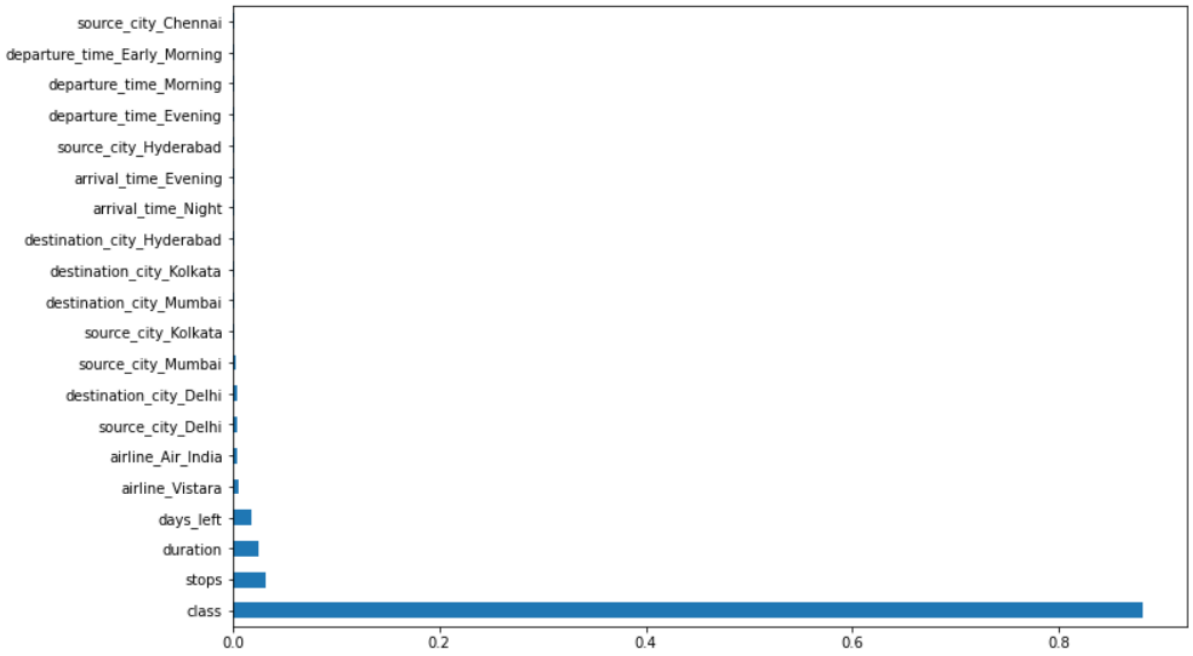
- *Chọn các biến quan trọng*

```
[74] from sklearn.ensemble import ExtraTreesRegressor
     selection = ExtraTreesRegressor()
     selection.fit(X, y)

     ExtraTreesRegressor()
```

```
[75] print(selection.feature_importances_)
```

```
[3.16673008e-02 8.80371106e-01 2.47675540e-02 1.87818574e-02
 4.88809894e-03 1.86480436e-04 1.69102127e-04 1.46847428e-04
 6.06825701e-03 1.04319230e-03 4.40801154e-03 1.47550476e-03
 2.07641622e-03 2.34887697e-03 1.08698994e-03 1.47436030e-03
 5.48665451e-05 1.35643113e-03 9.95558960e-04 9.49009662e-04
 1.50471682e-03 2.15280337e-04 1.03537547e-03 1.74075205e-03
 1.02964300e-03 4.17920487e-03 1.92298720e-03 1.99668322e-03
 2.05953464e-03]
```

```
[76] plt.figure(figsize = (12,8))
     feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
     feat_importances.nlargest(20).plot(kind='barh')
     plt.show()
```




```
[77] X = df4[['class', 'stops', 'duration', 'days_left']]
     y = df4['price']
```

4. Chạy các mô hình máy học liên quan với bộ dữ liệu huấn luyện

- *Chia dữ liệu để huấn luyện*

```
[82] X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state= 42)
```

- *Mô hình Hồi quy tuyến tính*

```
[83] model = LinearRegression()
     model.fit(X_train,y_train)
     y_pred = model.predict(X_test)
     print(y_pred)
```

```
[ 5788.13701482 50637.56023357  5041.43241229 ...  6937.69883365
 -1579.62770683 56749.66898952]
```

```
[84] print("R2 Score: ",r2_score(y_test,y_pred))
     print("Mean Squared Error: ",mean_squared_error(y_test, y_pred))
     print('Mean Absolute Error', mean_absolute_error(y_test, y_pred))
     print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
R2 Score:  0.9001487797463545
Mean Squared Error:  51471534.03497546
Mean Absolute Error 4595.674349880799
Root Mean Squared Error: 7174.3664553028975
```

- *Mô hình Hồi quy Random Forest*

```
[85] from sklearn.ensemble import RandomForestRegressor

     regr = RandomForestRegressor()
     regr.fit(X_train, y_train)
     y_pred = regr.predict(X_test)
     print(y_pred)
```

```
[ 5212.31502381 63275.31117509  5140.63797173 ...  5847.72063312
  2795.09784133 78967.07036386]
```

```
[86] print("R2 Score: ",r2_score(y_test,y_pred))
     print("Mean Squared Error: ",mean_squared_error(y_test, y_pred))
     print('Mean Absolute Error', mean_absolute_error(y_test, y_pred))
     print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
```

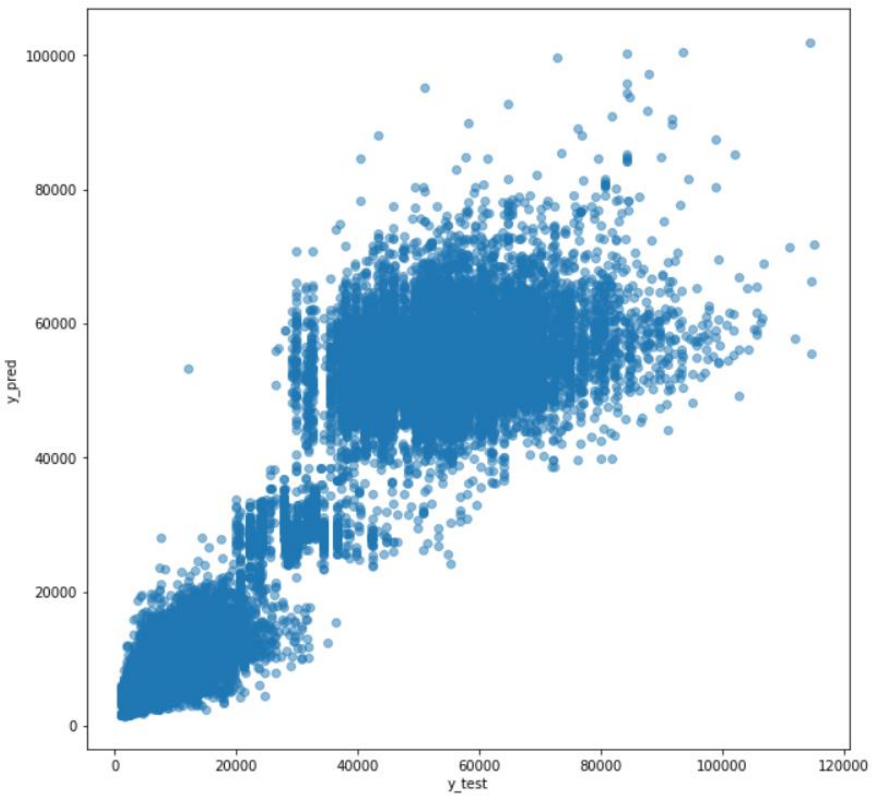
```
R2 Score:  0.9233583656072071
Mean Squared Error:  39507403.9468312
Mean Absolute Error 3673.7287846396966
Root Mean Squared Error: 6285.491543772149
```

⇨ Ta thấy được mô hình Random Forest có các chỉ số cao nhất

Các chỉ số R2 và RMSE cho thấy mô hình này đạt độ chính xác đến 92% cho bộ dữ liệu huấn luyện và mức chênh lệch so với bộ dữ liệu dự báo nằm ở khoảng ± 12 600

- *Biểu đồ dự đoán so với dữ liệu huấn luyện*

```
[88] plt.figure(figsize = (10,10))
     plt.scatter(y_test, y_pred, alpha = 0.5)
     plt.xlabel("y_test")
     plt.ylabel("y_pred")
     plt.show()
```



5. Kiểm định mô hình với bộ dữ liệu thực tế

- *Bộ dữ liệu thực tế*

```
[90] cat_columns = ['airline','source_city','departure_time','stops','arrival_time','destination_city','class']
     num_columns = ['duration','days_left']
```

```
[91] encoder = OrdinalEncoder().fit_transform(df3[cat_columns])
     encoder = pd.DataFrame(encoder,columns = cat_columns)
```

```
[92] print(encoder)
```

```
        airline  source_city  departure_time  stops  arrival_time  \
0           4.0          2.0             2.0    0.0           5.0
1           4.0          2.0             1.0    0.0           4.0
2           0.0          2.0             1.0    0.0           1.0
3           5.0          2.0             4.0    0.0           0.0
4           5.0          2.0             4.0    0.0           4.0
...         ...          ...             ...    ...           ...
300148      5.0          1.0             4.0    1.0           2.0
300149      5.0          1.0             0.0    1.0           5.0
300150      5.0          1.0             1.0    1.0           5.0
300151      5.0          1.0             1.0    1.0           2.0
300152      5.0          1.0             4.0    1.0           2.0

        destination_city  class
0                    5.0    0.0
1                    5.0    0.0
2                    5.0    0.0
3                    5.0    0.0
4                    5.0    0.0
...                  ...    ...
300148               3.0    1.0
300149               3.0    1.0
300150               3.0    1.0
300151               3.0    1.0
300152               3.0    1.0

[300153 rows x 7 columns]
```

```
X = pd.concat([encoder,df3[num_columns]],axis=1)
y = df3['price']
```

- *Lấy dữ liệu kiểm định*

```
[94] X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state= 42)
```

- *Hồi quy Random Forest*

```
[98] from sklearn.ensemble import RandomForestRegressor

     regr = RandomForestRegressor()
     regr.fit(X_train, y_train)
     y_pred = regr.predict(X_test)
     print(y_pred)
```

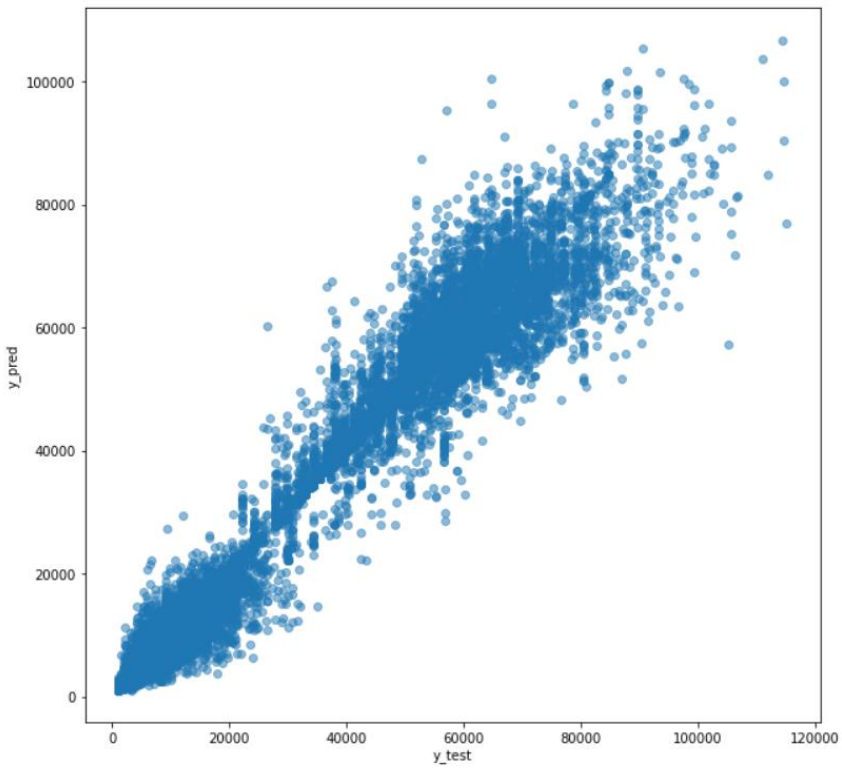
```
[ 6342.93 69846.36  6195.   ...  6589.61  3762.9  71053.96]
```

```
[99] print("R2 Score: ",r2_score(y_test,y_pred))
     print("Mean Squared Error: ",mean_squared_error(y_test, y_pred))
     print('Mean Absolute Error', mean_absolute_error(y_test, y_pred))
     print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
R2 Score:  0.9850444253725149
Mean Squared Error:  7709333.6115049515
Mean Absolute Error 1087.2709639228844
Root Mean Squared Error: 2776.568675812819
```

- *Biểu đồ dự đoán với dữ liệu thực tế*

```
plt.figure(figsize = (10,10))
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



- ⇨ Các chỉ số R2 và RMSE cho thấy mô hình này đạt độ chính xác đến 98% cho bộ dữ liệu huấn luyện và mức chênh lệch so với bộ dữ liệu thực tế nằm ở khoảng ± 5 600.

III. KẾT LUẬN

Qua bộ dữ liệu vé máy bay, nhóm đã phần nào phân tích được những đặc điểm cơ bản việc đặt vé máy bay cũng như các yếu tố ảnh hưởng đến giá vé. Ngoài ra, nhóm cũng đã xây dựng được mô hình dự báo giá vé máy bay cơ bản nhưng chỉ mới dừng lại ở việc kiểm định khả năng chính xác của dự báo. Tuy nhiên đây cũng là một bước tiến mới của nhóm và nhóm cũng đã học hỏi được nhiều hơn về kỹ năng phân tích cũng như tìm tòi, khám phá những cách phân tích khác nhau đối với một bộ dữ liệu nhất định.

TÀI LIỆU THAM KHẢO

Flight Price Prediction. (n.d). Retrieved June 17, 2022, from https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction.

Flight Price Solution Challenge. (n.d). Retrieved June 17, 2022, from https://www.kaggle.com/code/omarmohamedelgharib/flightpricesolutionchallenge.