

Thêm các thư viện

In [1]:

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from IPython.display import Markdown, display
```

In [2]:

```
1 from sklearn.model_selection import StratifiedShuffleSplit
2 from sklearn.pipeline import Pipeline
3 from sklearn.preprocessing import MinMaxScaler
4 from sklearn.impute import SimpleImputer
```

In [3]:

```
1 from sklearn.compose import ColumnTransformer
2 from sklearn.preprocessing import OneHotEncoder
3 from sklearn.preprocessing import LabelEncoder
4 from sklearn.preprocessing import OrdinalEncoder
```

In [4]:

```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn import tree
4 from sklearn.model_selection import GridSearchCV
```

In [5]:

```
1 from sklearn.metrics import classification_report
2 from sklearn.metrics import confusion_matrix
3 from sklearn.metrics import roc_curve
4 from sklearn.metrics import accuracy_score
5 from sklearn.metrics import roc_auc_score
6 from sklearn.metrics import RocCurveDisplay
7 from sklearn import metrics
```

I. Chuẩn bị dữ liệu

1. Đọc dữ liệu

```
In [6]: 1 df = pd.read_csv("Invistico_Airline.csv")
```

```
In [7]: 1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   satisfaction    129880 non-null   object  
 1   Gender          129880 non-null   object  
 2   Customer Type  129880 non-null   object  
 3   Age             129880 non-null   int64  
 4   Type of Travel 129880 non-null   object  
 5   Class           129880 non-null   object  
 6   Flight Distance 129880 non-null   int64  
 7   Seat comfort    129880 non-null   int64  
 8   Departure/Arrival time convenient 129880 non-null   int64  
 9   Food and drink  129880 non-null   int64  
 10  Gate location   129880 non-null   int64  
 11  Inflight wifi service 129880 non-null   int64  
 12  Inflight entertainment 129880 non-null   int64  
 13  Online support   129880 non-null   int64  
 14  Ease of Online booking 129880 non-null   int64  
 15  On-board service  129880 non-null   int64  
 16  Leg room service 129880 non-null   int64  
 17  Baggage handling 129880 non-null   int64  
 18  Checkin service  129880 non-null   int64  
 19  Cleanliness     129880 non-null   int64  
 20  Online boarding  129880 non-null   int64  
 21  Departure Delay in Minutes 129880 non-null   int64  
 22  Arrival Delay in Minutes 129487 non-null   float64
dtypes: float64(1), int64(17), object(5)
memory usage: 22.8+ MB
```

```
In [8]: 1 df.head()
```

Out[8]:

	satisfaction	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Seat comfort	Departure/Arrival time convenient	Food and drink	...	Online support	Ease of Online booking	On-board service	Leg room service	Bag han
0	satisfied	Female	Loyal Customer	65	Personal Travel	Eco	265	0	0	0	...	2	3	3	0	0
1	satisfied	Male	Loyal Customer	47	Personal Travel	Business	2464	0	0	0	...	2	3	4	4	4
2	satisfied	Female	Loyal Customer	15	Personal Travel	Eco	2138	0	0	0	...	2	2	3	3	3
3	satisfied	Female	Loyal Customer	60	Personal Travel	Eco	623	0	0	0	...	3	1	1	0	0
4	satisfied	Female	Loyal Customer	70	Personal Travel	Eco	354	0	0	0	...	4	2	2	0	0

5 rows × 23 columns

Chúng ta có thể thấy tập dữ liệu này gồm các nhóm liên quan đến thuộc tính của khách hàng (Age, Gender,...), thông tin về chuyến bay (Class, Inflight wifi service, ...), đánh giá của khách hàng (Seat comfort, Food and drink, ...).

Loại dữ liệu của từng nhóm

- Thuộc tính khách hàng: Object
- Thông tin chuyến bay: Integer and Object
- Đánh giá của khách hàng: Integer (1->5)
- Độ hài lòng: Binary (0, 1)

In [9]: 1 df.columns

```
Out[9]: Index(['satisfaction', 'Gender', 'Customer Type', 'Age', 'Type of Travel',  
              'Class', 'Flight Distance', 'Seat comfort',  
              'Departure/Arrival time convenient', 'Food and drink', 'Gate location',  
              'Inflight wifi service', 'Inflight entertainment', 'Online support',  
              'Ease of Online booking', 'On-board service', 'Leg room service',  
              'Baggage handling', 'Checkin service', 'Cleanliness', 'Online boarding',  
              'Departure Delay in Minutes', 'Arrival Delay in Minutes'],  
              dtype='object')
```

2. Tổng quan về dữ liệu

Phân phối của dữ liệu

In [10]: 1 df.describe()

Out[10]:

In [11]:

```
1 #function to see values in each column
2 def list_values_column(df,num_value):
3     display(Markdown('**RESULT:**'))
4     print('DataFrame has {} column '.format(len(df.columns)))
5     for col in df.columns:
6         if df[col].nunique() <= num_value:
7             print('Column {} has :'.format(col)+ str(df[col].unique().tolist()),)
```

In [12]:

```
1 [[j, df[j].unique()] for j in [i for i in df.columns] if df[j].nunique() <= 5]
```

Out[12]:

```
[['satisfaction', array(['satisfied', 'dissatisfied'], dtype=object)],
['Gender', array(['Female', 'Male'], dtype=object)],
['Customer Type',
array(['Loyal Customer', 'disloyal Customer'], dtype=object)],
['Type of Travel',
array(['Personal Travel', 'Business travel'], dtype=object)],
['Class', array(['Eco', 'Business', 'Eco Plus'], dtype=object)],
['Baggage handling', array([3, 4, 1, 2, 5], dtype=int64)]]
```

```
In [13]: 1 list_values_column(df,6)
```

**RESULT:

```
DataFrame has 23 column
Column satisfaction has :['satisfied', 'dissatisfied']
Column Gender has :['Female', 'Male']
Column Customer Type has :['Loyal Customer', 'disloyal Customer']
Column Type of Travel has :['Personal Travel', 'Business travel']
Column Class has :['Eco', 'Business', 'Eco Plus']
Column Seat comfort has :[0, 1, 4, 5, 2, 3]
Column Departure/Arrival time convenient has :[0, 1, 2, 3, 4, 5]
Column Food and drink has :[0, 1, 2, 3, 4, 5]
Column Gate location has :[2, 3, 4, 1, 5, 0]
Column Inflight wifi service has :[2, 0, 3, 4, 5, 1]
Column Inflight entertainment has :[4, 2, 0, 3, 5, 1]
Column Online support has :[2, 3, 4, 5, 1, 0]
Column Ease of Online booking has :[3, 2, 1, 5, 4, 0]
Column On-board service has :[3, 4, 1, 2, 5, 0]
Column Leg room service has :[0, 4, 3, 2, 5, 1]
Column Baggage handling has :[3, 4, 1, 2, 5]
Column Checkin service has :[5, 2, 4, 3, 1, 0]
Column Cleanliness has :[3, 4, 1, 2, 5, 0]
Column Online boarding has :[2, 3, 5, 4, 1, 0]
```

```
In [14]: 1 #function to see ratio of values in each columns
2 def values_count_column(df,num_value):
3     display(Markdown('**RESULT:**'))
4     for col in df.columns:
5         if df[col].nunique() <= num_value:
6             print( 'Column ',col,':')
7             print(df[col].value_counts()/len(df))
```

```
In [15]: 1 values_count_column(df,3)
```

**RESULT:

```
Column satisfaction :  
satisfied      0.547328  
dissatisfied   0.452672  
Name: satisfaction, dtype: float64  
Column Gender :  
Female       0.507384  
Male        0.492616  
Name: Gender, dtype: float64  
Column Customer Type :  
Loyal Customer    0.816908  
disloyal Customer 0.183092  
Name: Customer Type, dtype: float64  
Column Type of Travel :  
Business travel  0.690584  
Personal Travel  0.309416  
Name: Type of Travel, dtype: float64  
Column Class :  
Business      0.478596  
Eco          0.448945  
Eco Plus     0.072459  
Name: Class, dtype: float64
```

```
In [16]: 1 [df[j].value_counts()/len(df[j]) for j in [i for i in df.columns] if df[j].nunique() <= 5]
```

```
Out[16]: [satisfied      0.547328
dissatisfied   0.452672
Name: satisfaction, dtype: float64,
Female        0.507384
Male          0.492616
Name: Gender, dtype: float64,
Loyal Customer    0.816908
disloyal Customer  0.183092
Name: Customer Type, dtype: float64,
Business travel   0.690584
Personal Travel    0.309416
Name: Type of Travel, dtype: float64,
Business        0.478596
Eco            0.448945
Eco Plus       0.072459
Name: Class, dtype: float64,
4             0.371420
5             0.275239
3             0.188520
2             0.103419
1             0.061403
Name: Baggage handling, dtype: float64]
```

3. Chuẩn hóa tên cột

```
In [17]: 1 def rename_column(df,from_c,to_c = '_'):
2     display(Markdown('**RESULT:**'))
3     print('Columns changed from {} to {}'.format(from_c))
4     print(df.columns[df.columns.str.contains(from_c)])
5     df.columns = [label.replace(from_c, to_c) for label in df.columns]
```

In [18]:

```
1 rename_column(df, ' ')
2 rename_column(df, '-')
```

RESULT:

```
Columns changed from _ to -
Index(['Customer Type', 'Type of Travel', 'Flight Distance', 'Seat comfort',
       'Departure/Arrival time convenient', 'Food and drink', 'Gate location',
       'Inflight wifi service', 'Inflight entertainment', 'Online support',
       'Ease of Online booking', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Online boarding',
       'Departure Delay in Minutes', 'Arrival Delay in Minutes'],
      dtype='object')
```

RESULT:

```
Columns changed from - to _
Index(['On-board_service'], dtype='object')
```

4. Xem xét dữ liệu rỗng

```
In [19]: 1 total = df.isna().sum()  
2 total
```

```
Out[19]: satisfaction          0  
Gender                  0  
Customer_Type            0  
Age                     0  
Type_of_Travel           0  
Class                   0  
Flight_Distance          0  
Seat_comfort             0  
Departure/Arrival_time_convenient 0  
Food_and_drink           0  
Gate_location             0  
Inflight_wifi_service    0  
Inflight_entertainment   0  
Online_support            0  
Ease_of_Online_booking    0  
On_board_service          0  
Leg_room_service          0  
Baggage_handling          0  
Checkin_service           0  
Cleanliness               0  
Online_boarding           0  
Departure_Delay_in_Minutes 0  
Arrival_Delay_in_Minutes 393  
dtype: int64
```

-> Arrival_Delay_in_Minutes có 393 dữ liệu bị rỗng

```
In [20]: 1 def Missing_table(df):  
2     sum_na = df.isna().sum()  
3     miss_percent = sum_na/len(df)  
4     Na_table = pd.concat([sum_na,miss_percent],axis = 1)  
5     Na_table = Na_table.rename(columns = {0:'Sum_of_na',1:'Percent_na'})  
6     Na_table = Na_table[Na_table.iloc[:,1]!=0]  
7     Na_table  
8     return Na_table
```

```
In [21]: 1 Missing_table(df)
```

Out[21]:

	Sum_of_na	Percent_na
Arrival_Delay_in_Minutes	393	0.003026

5. Mã hóa biến phụ thuộc

```
In [22]: 1 en = LabelEncoder()
2 df['satisfaction'] = en.fit_transform(df['satisfaction'])
```

In [23]: 1 df

Out[23]:

	satisfaction	Gender	Customer_Type	Age	Type_of_Travel	Class	Flight_Distance	Seat_comfort	Departure/Arrival_time_convenient	Food_
0	1	Female	Loyal Customer	65	Personal Travel	Eco	265	0		0
1	1	Male	Loyal Customer	47	Personal Travel	Business	2464	0		0
2	1	Female	Loyal Customer	15	Personal Travel	Eco	2138	0		0
3	1	Female	Loyal Customer	60	Personal Travel	Eco	623	0		0
4	1	Female	Loyal Customer	70	Personal Travel	Eco	354	0		0
...
129875	1	Female	disloyal Customer	29	Personal Travel	Eco	1731	5		5
129876	0	Male	disloyal Customer	63	Personal Travel	Business	2087	2		3
129877	0	Male	disloyal Customer	69	Personal Travel	Eco	2320	3		0
129878	0	Male	disloyal Customer	66	Personal Travel	Eco	2450	3		2
129879	0	Female	disloyal Customer	38	Personal Travel	Eco	4307	3		4

129880 rows × 23 columns

II. Trực quan hóa dữ liệu

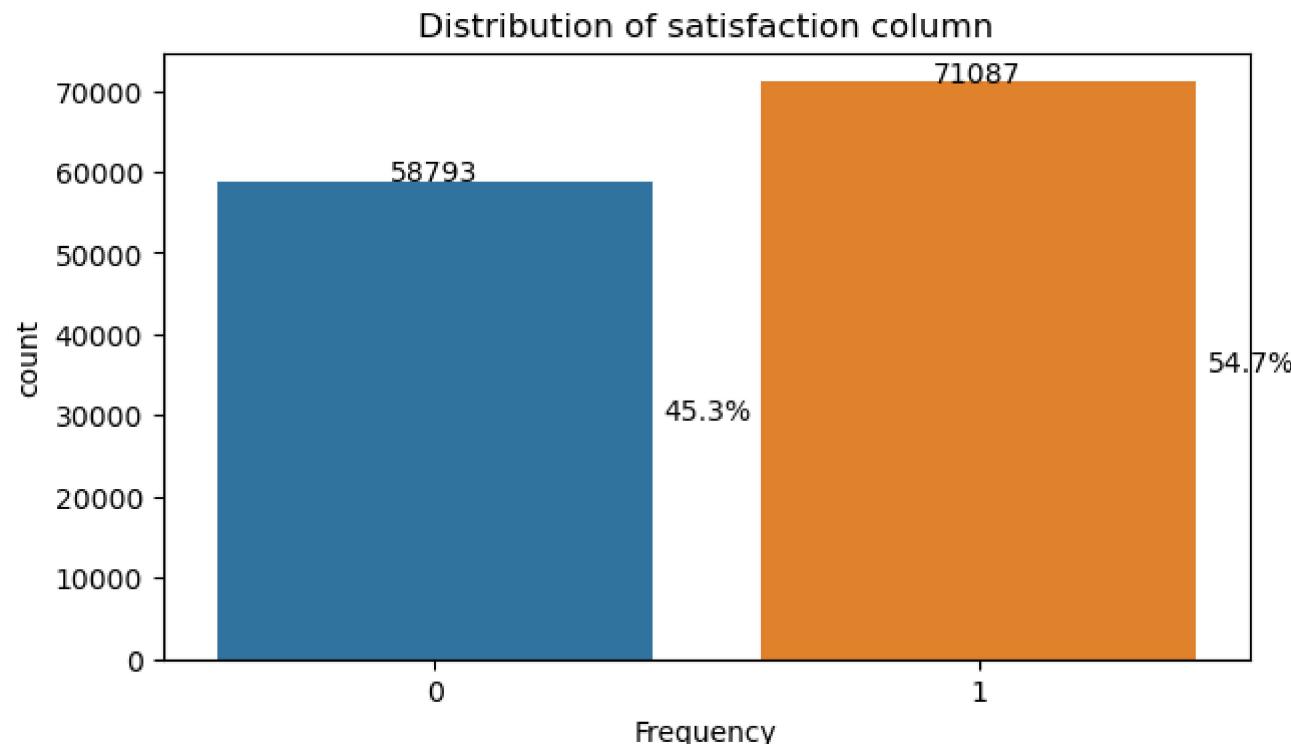
1. Tỷ lệ giữa các giá trị của biến phụ thuộc

In [24]:

```
1 def plot_target_column (df, target):  
2     ax = sns.countplot(x=target, data=df)  
3     plt.title('Distribution of ' + target + ' column')  
4     plt.xlabel('Frequency')  
5     ax.set_xticklabels(ax.get_xticklabels(),rotation=0)  
6     total = len(df[target])  
7     for p in ax.patches:  
8         percentage = '{:.1f}%'.format(100 * p.get_height()/total)  
9         x = p.get_x() + p.get_width() + 0.02  
10        y = p.get_y() + p.get_height()/2  
11        ax.annotate(percentage, (x, y))  
12        ax.text(p.get_x()+p.get_width()/2., p.get_height() + 0.2, p.get_height(),ha="center")  
13    plt.show()
```

In [25]:

```
1 plt.figure(figsize= (7,4), dpi= 100)  
2 plot_target_column(df, "satisfaction")
```



2. Trực quan hóa các thuộc tính của khách hàng theo biến phụ thuộc

- Select customer's properties columns

```
In [26]: 1 df["Age_cat"] = pd.cut(df["Age"], bins= [0, 27, 57, 77, np.inf]
2           , labels= ["7-27", "27-57", "57-77", "77-85"])
```

- Separate customers into 4 groups according to Age

```
In [27]: 1 bin_user_prop = [i for i in df.loc[:, "satisfaction": "Class"].columns if i not in ('Age', 'Type_of_Travel', 'Class')]
```

```
In [28]: 1 bin_user_prop.append("Age_cat")
```

```
In [29]: 1 bin_user_prop
```

```
Out[29]: ['satisfaction', 'Gender', 'Customer_Type', 'Age_cat']
```

```
In [30]: 1 df[bin_user_prop]
```

Out[30]:

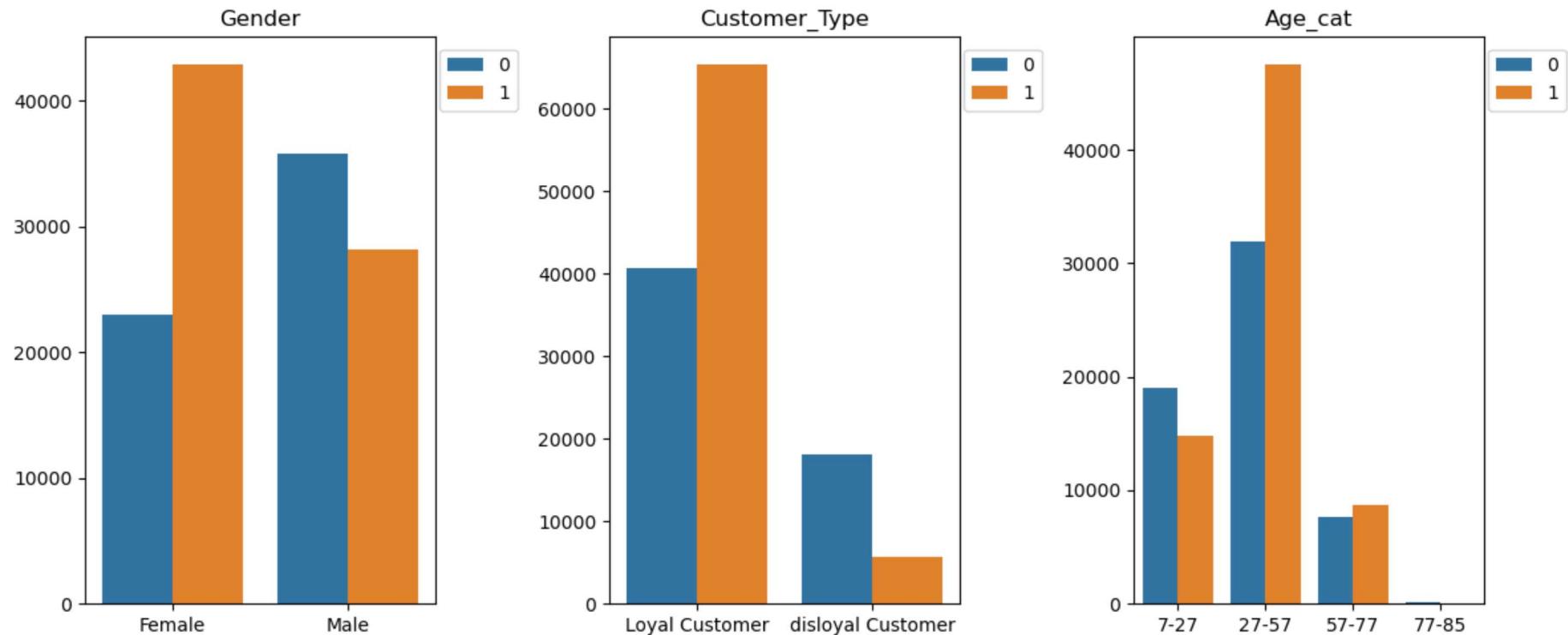
	satisfaction	Gender	Customer_Type	Age_cat
0	1	Female	Loyal Customer	57-77
1	1	Male	Loyal Customer	27-57
2	1	Female	Loyal Customer	7-27
3	1	Female	Loyal Customer	57-77
4	1	Female	Loyal Customer	57-77
...
129875	1	Female	disloyal Customer	27-57
129876	0	Male	disloyal Customer	57-77
129877	0	Male	disloyal Customer	57-77
129878	0	Male	disloyal Customer	57-77
129879	0	Female	disloyal Customer	27-57

129880 rows × 4 columns

- Using countplot to visualize customer's properties along with satisfaction

In [31]:

```
1 fig, axes = plt.subplots(nrows=1,ncols=3,figsize=(12, 5), dpi= 100)
2 for i in range(0,3):
3     sns.countplot(data= df[bin_user_prop], x= bin_user_prop[i + 1], hue= "satisfaction", ax= axes[i])
4     axes[i].legend(loc= (1.01, 0.87))
5     axes[i].set_xlabel("")
6     axes[i].set_ylabel("")
7     axes[i].set(title= bin_user_prop[i + 1])
8
9 plt.tight_layout()
10 plt.show()
```

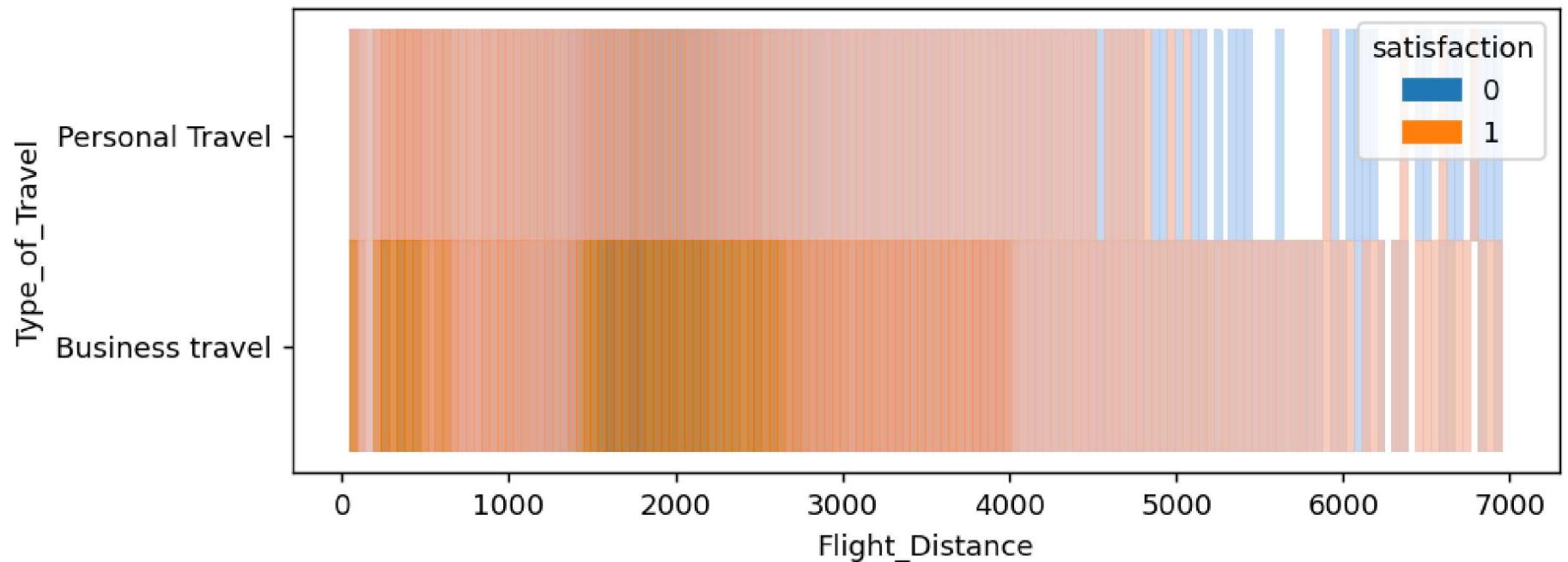


Look like an easygoing female like a male ...

3. Trực quan hóa thông tin của chuyến bay theo biến phụ thuộc

```
In [32]: 1 plt.figure(figsize= (8, 3), dpi = 130)
2 sns.histplot(data= df, x= 'Flight_Distance', y= 'Type_of_Travel', hue= 'satisfaction', alpha = 0.7)
```

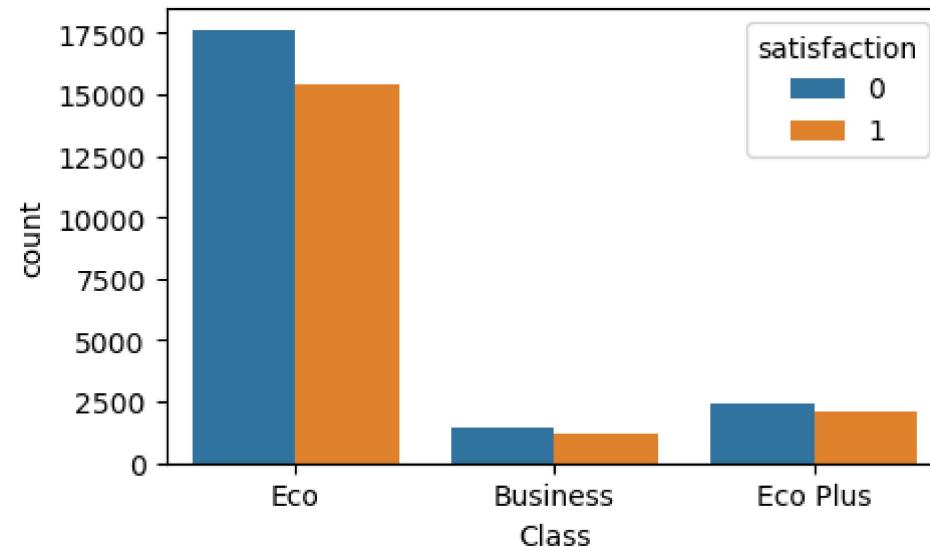
```
Out[32]: <AxesSubplot:xlabel='Flight_Distance', ylabel='Type_of_Travel'>
```



- Có sự khác biệt về phân bố của khoảng cách bay và độ hài lòng giữa các mục đích du lịch

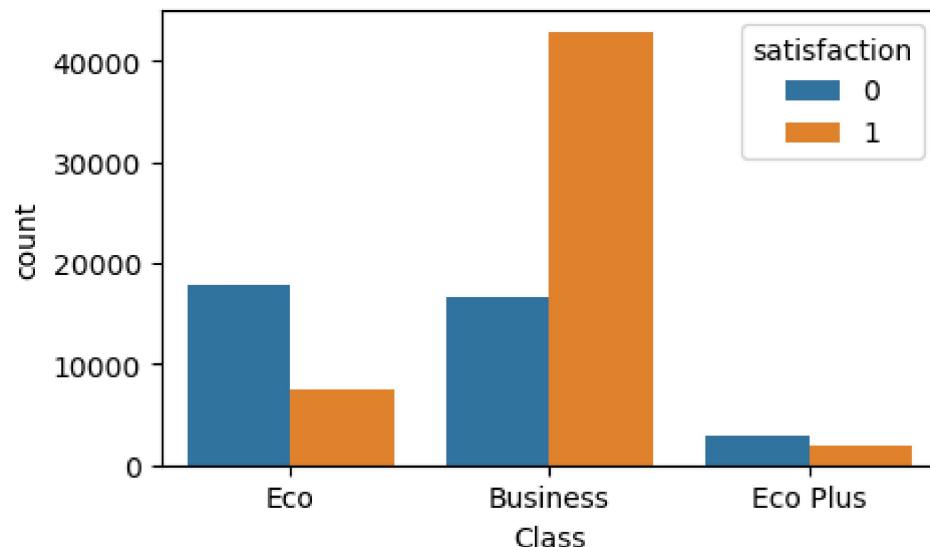
In [33]:

```
1 plt.figure(figsize= (5, 3), dpi = 100)
2 sns.countplot(data= df[df[ 'Type_of_Travel' ] == 'Personal Travel'], x= 'Class', hue= 'satisfaction')
3 plt.show()
```



In [34]:

```
1 plt.figure(figsize= (5, 3), dpi = 100)
2 sns.countplot(data= df[df['Type_of_Travel'] == 'Business travel'], x= 'Class', hue= 'satisfaction')
3 plt.show()
```



- Vấn đề phát hiện qua trực quan hóa:
 - Loại vé Eco thường sử dụng cho mục đích Personal Travel và nhìn chung, khách hàng đang không hài lòng với dịch vụ ở hạng mục vé này khi đi xa
 - Loại vé Business thường được khách hàng đánh giá tốt ở nhiều mục đích sử dụng
 - Loại vé Eco Plus không có quá nhiều khác biệt đối với các mục đích sử dụng khác nhau và số lượng hành khách thuộc loại vé này còn khá ít
- Giải pháp đề xuất:
 - Cần xem xét thay đổi dịch vụ với hạng vé Eco dành cho những chuyến bay xa đặc biệt đối với những dịch vụ ở trên chuyến bay như giải trí, chỗ ngồi, đồ ăn, ...
 - Cung cấp thêm nhiều dịch vụ đặc biệt cho hạng vé Eco Plus, cần thêm marketing, PR ... cho loại vé này

```
In [35]: 1 corr_matrix = df.corr()
2 corr_matrix['satisfaction'].sort_values(ascending = False)
```

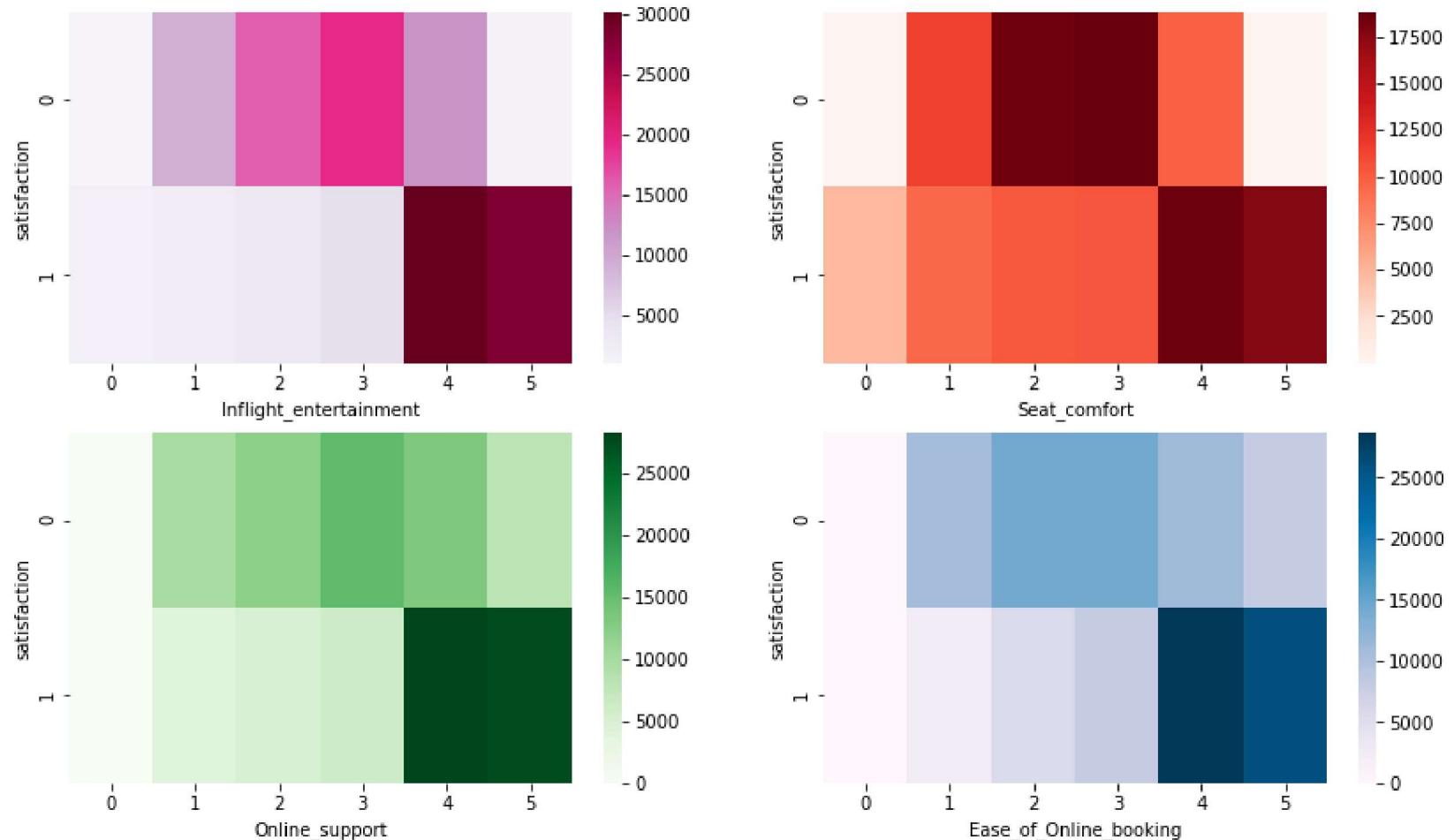
```
Out[35]: satisfaction           1.000000
Inflight_entertainment      0.523496
Ease_of_Online_booking       0.431772
Online_support                0.390143
On_board_service              0.352047
Online_boarding                0.338147
Leg_room_service               0.304928
Checkin_service                  0.266179
Baggage_handling                 0.260347
Cleanliness                      0.259330
Seat_comfort                      0.242384
Inflight_wifi_service            0.227062
Food_and_drink                   0.120677
Age                            0.117971
Gate_location                     -0.012071
Departure/Arrival_time_convenient -0.015507
Flight_Distance                    -0.039224
Departure_Delay_in_Minutes        -0.073909
Arrival_Delay_in_Minutes          -0.080691
Name: satisfaction, dtype: float64
```

Vẽ biểu đồ heatmap với những cột có điểm corr tương đối cao

In [36]:

```
1 fig, axarr = plt.subplots(2, 2, figsize=(14, 8))
2
3 table1 = pd.crosstab(df['satisfaction'], df['Inflight_entertainment'])
4 sns.heatmap(table1, cmap='PuRd', ax = axarr[0][0])
5 table2 = pd.crosstab(df['satisfaction'], df['Seat_comfort'])
6 sns.heatmap(table2, cmap='Reds', ax = axarr[0][1])
7 table3 = pd.crosstab(df['satisfaction'], df['Online_support'])
8 sns.heatmap(table3, cmap='Greens', ax = axarr[1][0])
9 table4 = pd.crosstab(df['satisfaction'], df['Ease_of_Online_booking'])
10 sns.heatmap(table4, cmap='PuBu', ax = axarr[1][1])
```

Out[36]: <AxesSubplot:xlabel='Ease_of_Online_booking', ylabel='satisfaction'>



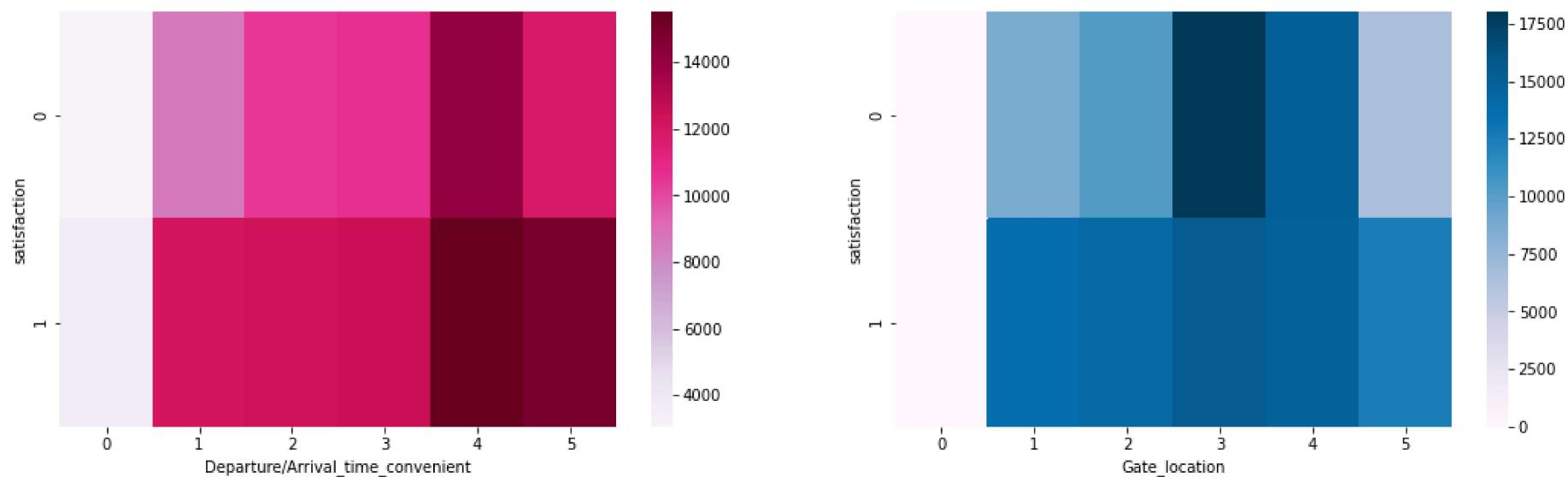
Có sự tương quan rõ rệt với satisfaction. Rất nhiều đánh giá 4,5 được xếp vào loại hài lòng Ngược lại, rất nhiều đánh giá 1-3 thì xếp vào loại không hài lòng.

Cho thấy những feature này có tác động lớn vào đánh giá độ hài lòng của khách hàng

In [37]:

```
1 #biểu đồ heatmap với những cột có hệ số tương quan cực thấp
2 #Gate_Location -0.012071
3 #Departure/Arrival_time_convenient -0.015507
4 fig, axarr = plt.subplots(1, 2, figsize=(18, 5))
5
6 table1 = pd.crosstab(df['satisfaction'], df['Departure/Arrival_time_convenient'])
7 sns.heatmap(table1, cmap='PuRd', ax = axarr[0])
8 table2 = pd.crosstab(df['satisfaction'], df['Gate_location'])
9 sns.heatmap(table2, cmap='PuBu', ax = axarr[1])
10
```

Out[37]: <AxesSubplot:xlabel='Gate_location', ylabel='satisfaction'>



Cả 2 biểu đồ trên đều không cho thấy sự tương quan. Những đánh giá 1,2,3 không khác biệt so với 4,5 sao về đánh giá độ hài lòng khách hàng.

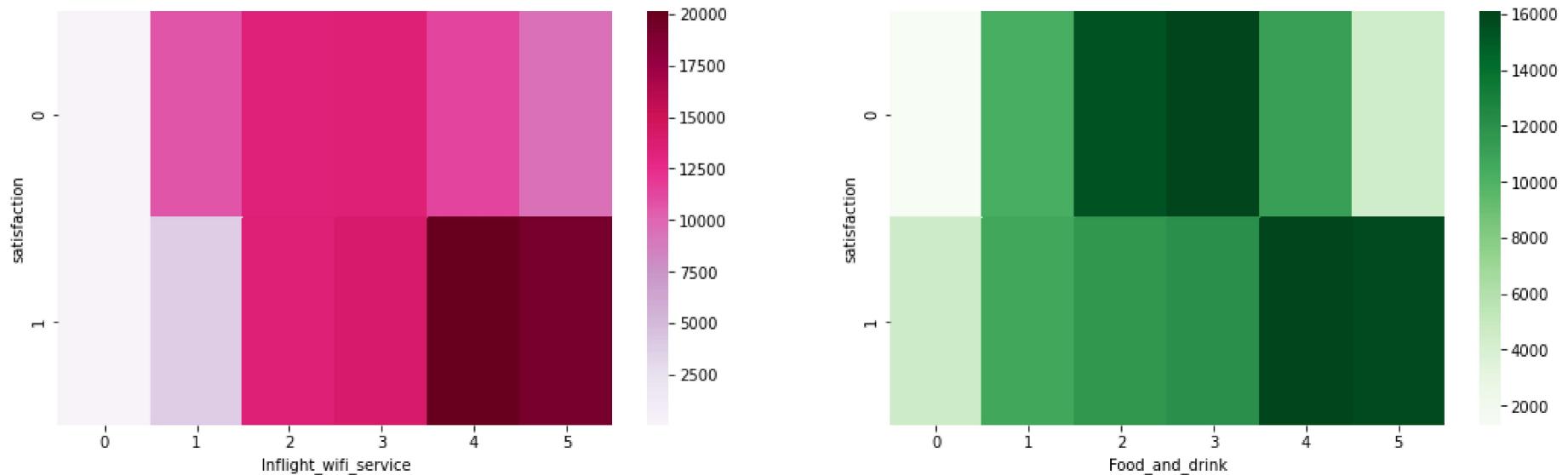
- Ta kết luận được rằng 2 feature trên không ảnh hưởng nhiều đến mức độ hài lòng khách hàng

Biểu đồ heatmap với những cột có hệ số tương quan so với satisfaction không cao

In [38]:

```
1 fig, axarr = plt.subplots(1, 2, figsize=(18, 5))
2
3 table = pd.crosstab(df['satisfaction'], df['Inflight_wifi_service'])
4 sns.heatmap(table, cmap='PuRd', ax = axarr[0])
5 table1 = pd.crosstab(df['satisfaction'], df['Food_and_drink'])
6 sns.heatmap(table1, cmap='Greens', ax = axarr[1])
```

Out[38]: <AxesSubplot:xlabel='Food_and_drink', ylabel='satisfaction'>



Sự hài lòng của khách hàng không quá phụ thuộc vào Inflight_wifi_service và Food_and_drink

4. Tương quan giữa các tính tăng trong bộ dữ liệu

In [39]:

```
1 corr = df.corr()
2 mask = np.triu(np.ones_like(corr, dtype=bool))
3 f, ax = plt.subplots(figsize=(20, 20))
4 cmap = sns.diverging_palette(1000, 200, as_cmap=True)
5 sns.heatmap(corr, mask=mask, cmap=cmap, vmax=None, center=0,square=True, annot=True, linewidths=.5, cbar_kws={"shrin
```

Out[39]: <AxesSubplot:>



Cleanliness -	0.26	-0.018	0.0094	0.11	0.067	0.033	-0.0017	0.038	0.11	0.096	0.42	0.55	0.41	0.63	0.24		
Online_boarding -	0.34	0.038	0.0096	0.13	-0.00062	0.014	-0.003	0.63	0.36	0.67	0.68	0.14	0.11	0.11	0.18	0.11	
Departure_Delay_in_Minutes -	-0.074	-0.009	0.11	-0.024	0.0044	-0.013	0.004	-0.033	-0.03	-0.034	-0.037	-0.038	0.0037	-0.01	-0.021	-0.062	-0.02
Arrival_Delay_in_Minutes -	-0.081	-0.011	0.11	-0.026	0.0026	-0.015	0.0036	-0.035	-0.033	-0.036	-0.04	-0.041	0.00047	-0.014	-0.024	-0.067	-0.022
satisfaction -		Age -	Flight_Distance -	Seat_comfort -	Departure/Arrival_time_convenient -	Food_and_drink -	Gate_location -	Inflight_wifi_service -	Inflight_entertainment -	Online_support -	Ease_of_Online_booking -	On_board_service -	Leg_room_service -	Baggage_handling -	Checkin_service -	Cleanliness -	Online_boarding -
																Arrival_Delay_in_Minutes -	Departure_Delay_in_Minutes -

Departure_Delay_in_Minutes tương quan cao với Arrival_Delay_in_Minutes từ 0.97 ~ 1

Kết luận: kết hợp Arrival_Delay_in_minutes và Departure_Delay_in_Minutes thành Delay_in_Minutes

```
In [40]: 1 df['Delay_in_Minutes'] = df['Arrival_Delay_in_Minutes'] + df['Departure_Delay_in_Minutes']
2 df.drop(['Arrival_Delay_in_Minutes', 'Departure_Delay_in_Minutes'], axis = 1, inplace=True)
```

III. Chuẩn bị dữ liệu cho mô hình

1. Chia thành tập huấn luyện, phát triển và kiểm thử

Lấy mẫu theo thuộc tính

```
In [41]: 1 #df['Age_cat'] = df['Age_cat'].astype('object')
```

In [42]:

```
1 # stratified sampling by Age_cat
2 split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=101)
3 for train_index, group_index in split.split(df, df["Age_cat"]):
4     strat_train_set = df.loc[train_index]
5     strat_group_set = df.loc[group_index]
6 # test size = 0.2
7 split_2 = StratifiedShuffleSplit(n_splits=1, test_size=0.5, random_state=101)
8 for dev_index, test_index in split_2.split(strat_group_set, strat_group_set["Age_cat"]):
9     strat_dev_set = strat_group_set.iloc[dev_index]
10    strat_test_set = strat_group_set.iloc[test_index]
```

In [43]:

```
1 strat_train_set.drop("Age_cat", axis=1, inplace=True)
2 strat_dev_set.drop("Age_cat", axis=1, inplace=True)
3 strat_test_set.drop("Age_cat", axis=1, inplace=True)
```

C:\Users\Taichi\anaconda3\envs\my-env-name\lib\site-packages\pandas\core\frame.py:4169: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
errors=errors,

Đổi tên các tập dữ liệu

In [44]:

```
1 # Tập huấn Luyện
2 y_train = strat_train_set['satisfaction']
3 X_train = strat_train_set.drop("satisfaction",axis =1 )
```

In [45]:

```
1 # Tập phát triển
2 y_dev = strat_dev_set['satisfaction']
3 X_dev = strat_dev_set.drop("satisfaction",axis = 1)
```

In [46]:

```
1 # Tập kiểm thử
2 y_test = strat_test_set['satisfaction']
3 X_test = strat_test_set.drop("satisfaction",axis = 1)
```

2. Xây dựng Pipeline xử lý dữ liệu cho mô hình

```
In [47]: 1 # Rút trích tên các thuộc tính phân loại và số liệu  
2 col_cat = X_train.columns[X_train.dtypes==object]  
3 col_num = X_train.columns[X_train.dtypes!=object]
```

```
In [48]: 1 onehot_col = ['Gender', 'Customer_Type', 'Type_of_Travel']  
2 ordi_col = col_cat.drop(['Gender', 'Customer_Type', 'Type_of_Travel'])
```

Pipeline cho số liệu

```
In [49]: 1 #pipeline handle missing values and scaling feature  
2 num_pipeline = Pipeline([  
3     ('imputer', SimpleImputer(strategy="mean")),  
4     ('std_scaler', MinMaxScaler()),  
5 ])
```

Pipeline tổng thể

```
In [50]: 1 # Pipeline tổng thể xử lý dữ liệu phân loại và số học  
2 full_pipeline = ColumnTransformer([  
3     ("num", num_pipeline, col_num),  
4     ("ordi", OrdinalEncoder(), ordi_col),  
5     ("onehot", OneHotEncoder(), onehot_col)  
6 ])
```

```
In [51]: 1 full_pipeline.fit(X_train)
```

```
Out[51]: ColumnTransformer(transformers=[('num',
                                         Pipeline(steps=[('imputer', SimpleImputer()),
                                                         ('std_scaler',
                                                          MinMaxScaler())]),
                                         Index(['Age', 'Flight_Distance', 'Seat_comfort',
                                                'Departure/Arrival_time_convenient', 'Food_and_drink', 'Gate_location',
                                                'Inflight_wifi_service', 'Inflight_entertainment', 'Online_support',
                                                'Ease_of_Online_booking', 'On_board_service', 'Leg_room_service',
                                                'Baggage_handling', 'Checkin_service', 'Cleanliness', 'Online_boarding',
                                                'Delay_in_Minutes'],
                                               dtype='object')),
                                         ('ordi', OrdinalEncoder(),
                                          Index(['Class'], dtype='object')),
                                         ('onehot', OneHotEncoder(),
                                          ['Gender', 'Customer_Type',
                                           'Type_of_Travel']))])
```

Xử lý tên cột cho các thuộc tính khi đi qua Pipeline

```
In [52]: 1 col_name = list(col_num) + list(ordi_col)
2 full_col_name = col_name + list(full_pipeline.named_transformers_.onehot.get_feature_names())
```

```
In [53]: 1 # Đưa các tập dữ liệu qua Pipeline
2 X_train= full_pipeline.transform(X_train)
3 X_train = pd.DataFrame(X_train,columns=full_col_name)
4
5 X_test = full_pipeline.transform(X_test)
6 X_test = pd.DataFrame(X_test,columns=full_col_name)
7
8 X_dev = full_pipeline.transform(X_dev)
9 X_dev = pd.DataFrame(X_dev,columns=full_col_name)
```

IV. Lựa chọn và tối ưu mô hình máy học

1. Logistic Regression

```
In [54]: 1 log_reg = LogisticRegression(max_iter= 1000)
```

```
In [55]: 1 log_reg.fit(X_train, y_train)
```

```
Out[55]: LogisticRegression(max_iter=1000)
```

```
In [56]: 1 predict_dev = log_reg.predict(X_dev)
```

```
In [57]: 1 print(confusion_matrix(predict_dev, y_dev))
```

```
[[4731  996]
 [1117 6144]]
```

```
In [58]: 1 print(classification_report(predict_dev, y_dev))
```

	precision	recall	f1-score	support
0	0.81	0.83	0.82	5727
1	0.86	0.85	0.85	7261
accuracy			0.84	12988
macro avg	0.83	0.84	0.84	12988
weighted avg	0.84	0.84	0.84	12988

2. Neural Network

```
In [59]: 1 import tensorflow as tf
2 from tensorflow import keras
3 from tensorflow.keras import layers
```

```
In [60]: 1 tf.random.set_seed(1234)
```

In [61]:

```
1 # Define Sequential model with 3 layers
2 model = keras.Sequential(
3     [
4         layers.Dense(256, activation="relu", name="layer0"),
5         layers.Dense(256, activation="relu", name="layer1"),
6
7         layers.Dense(128, activation="linear", name="layer4"),
8         layers.Dense(1, activation='sigmoid')
9     ]
10 )
```

In [62]:

```
1 model.compile(
2     optimizer="adam",
3     loss='BinaryCrossentropy',
4     metrics=[tf.keras.metrics.BinaryAccuracy()],
5
6 )
```

```
In [63]: 1 history = model.fit(  
2     X_train,  
3     y_train,  
4     batch_size=256,  
5     epochs=10,  
6     validation_data=(X_dev, y_dev)  
7 )  
...  
val_binary_accuracy: 0.9474  
Epoch 5/10  
406/406 [=====] - 1s 2ms/step - loss: 0.1183 - binary_accuracy: 0.9477 - val_loss: 0.1242 -  
val_binary_accuracy: 0.9447  
Epoch 6/10  
406/406 [=====] - 1s 2ms/step - loss: 0.1140 - binary_accuracy: 0.9494 - val_loss: 0.1210 -  
val_binary_accuracy: 0.9423  
Epoch 7/10  
406/406 [=====] - 1s 2ms/step - loss: 0.1113 - binary_accuracy: 0.9513 - val_loss: 0.1183 -  
val_binary_accuracy: 0.9485  
Epoch 8/10  
406/406 [=====] - 1s 2ms/step - loss: 0.1076 - binary_accuracy: 0.9523 - val_loss: 0.1196 -  
val_binary_accuracy: 0.9465  
Epoch 9/10  
406/406 [=====] - 1s 2ms/step - loss: 0.1052 - binary_accuracy: 0.9530 - val_loss: 0.1139 -  
val_binary_accuracy: 0.9505  
Epoch 10/10  
406/406 [=====] - 1s 2ms/step - loss: 0.1020 - binary_accuracy: 0.9546 - val_loss: 0.1158 -  
val_binary_accuracy: 0.9503
```

```
In [64]: 1 y_proba_neural = model.predict(X_test)
```

```
In [65]: 1 print(classification_report(y_proba_neural > 0.5 , y_test))
```

	precision	recall	f1-score	support
False	0.96	0.93	0.95	6006
True	0.94	0.97	0.95	6982
accuracy			0.95	12988
macro avg	0.95	0.95	0.95	12988
weighted avg	0.95	0.95	0.95	12988

3. Randomforest

```
In [66]: 1 from sklearn.ensemble import RandomForestClassifier
```

```
In [67]: 1 rdf_model = RandomForestClassifier(verbose= 1 , n_jobs= -1, random_state= 101)
```

```
In [68]: 1 param_grid = {  
2     'bootstrap': [True],  
3     'max_depth': [30, 50],  
4     'max_features': [5],  
5     'min_samples_leaf': [2],  
6     'min_samples_split': [2],  
7     'n_estimators': [120, 140]  
8 }
```

```
In [70]: 1 grid_search.fit(X_train, y_train)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 8 concurrent workers.  
[Parallel(n_jobs=-1)]: Done 34 tasks | elapsed: 1.3s  
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed: 4.2s finished
```

```
Out[70]: GridSearchCV(cv=3,  
                      estimator=RandomForestClassifier(n_jobs=-1, random_state=101,  
                      verbose=1),  
                      n_jobs=-1,  
                      param_grid={'bootstrap': [True], 'max_depth': [30, 50],  
                      'max_features': [5], 'min_samples_leaf': [2],  
                      'min_samples_split': [2], 'n_estimators': [120, 140]})
```

```
In [72]: 1 grid_search.best_score_
```

```
Out[72]: 0.9574414741576524
```

```
In [73]: 1 y_proba_grid = grid_search.predict(X_dev)  
2 print(classification_report(y_proba_grid, y_dev))
```

	precision	recall	f1-score	support
0	0.96	0.94	0.95	5967
1	0.95	0.97	0.96	7021
accuracy			0.96	12988
macro avg	0.96	0.96	0.96	12988
weighted avg	0.96	0.96	0.96	12988

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.  
[Parallel(n_jobs=8)]: Done 34 tasks | elapsed: 0.0s  
[Parallel(n_jobs=8)]: Done 120 out of 120 | elapsed: 0.0s finished
```

```
In [75]: 1 rdf_model = RandomForestClassifier(n_jobs= -1, random_state= 101,
2                                     bootstrap= True,
3                                     max_features= 5,
4                                     max_depth= 30,
5                                     min_samples_split= 2,
6                                     n_estimators= 120,
7                                     verbose= 0)
```

```
In [76]: 1 rdf_model.fit(X_train, y_train)
```

```
Out[76]: RandomForestClassifier(max_depth=30, max_features=5, n_estimators=120,
n_jobs=-1, random_state=101)
```

```
In [76]: 1 rdf_model
```

```
[Parallel(n_jobs=8)]: Using backend ThreadingBackend with 8 concurrent workers.
[Parallel(n_jobs=8)]: Done 34 tasks      | elapsed:    0.0s
[Parallel(n_jobs=8)]: Done 120 out of 120 | elapsed:    0.0s finished
```

```
In [77]: 1 y_proba_rdf = rdf_model.predict(X_test)
2 print(classification_report(y_proba_rdf, y_test))
```

	precision	recall	f1-score	support
0	0.96	0.95	0.96	5957
1	0.96	0.97	0.96	7031
accuracy			0.96	12988
macro avg	0.96	0.96	0.96	12988
weighted avg	0.96	0.96	0.96	12988

Accuracy giữa các tập train-dev-test set không có sự chênh lệch, mô hình không bị bias và variance.

Thông dịch mô hình

```
In [78]: 1 log_ = pd.Series(log_reg.coef_.reshape(-1), X_dev.columns).sort_values(ascending= False)
```

```
In [79]: 1 rdf_ = pd.Series(rdf_model.feature_importances_, index= X_dev.columns).sort_values(ascending= False)
```

In [80]:

```
1 compare = pd.concat([rdf_,log_],axis= 1)
2 compare.columns = ['Random Forest','Logistic Regression']
3 compare
```

Out[80]:

	Random Forest	Logistic Regression
Inflight_entertainment	0.206430	3.436761
Seat_comfort	0.129908	1.405997
Ease_of_Online_booking	0.072443	1.040326
Online_support	0.064847	0.479436
On_board_service	0.038131	1.545003
Food_and_drink	0.037768	-1.049059
Leg_room_service	0.033105	1.089688
Online_boarding	0.032984	0.888470
Flight_Distance	0.031631	-0.708445
x1_Disloyal Customer	0.031288	-1.037317
x1_Loyal Customer	0.029772	1.017243
Age	0.028098	-0.593676
Class	0.027530	-0.508755
Checkin_service	0.025163	1.482836
x0_Female	0.024302	0.472870
Baggage_handling	0.024002	0.478272
Cleanliness	0.022591	0.447113
x0_Male	0.022104	-0.492944
x2_Business travel	0.021986	0.436882
x2_Personal Travel	0.021184	-0.456957
Departure/Arrival_time_convenient	0.020965	-1.012912
Delay_in_Minutes	0.020392	-7.563532
Gate_location	0.019193	0.554058

Random Forest Logistic Regression

Inflight_wifi_service	0.014182	-0.340966
------------------------------	----------	-----------

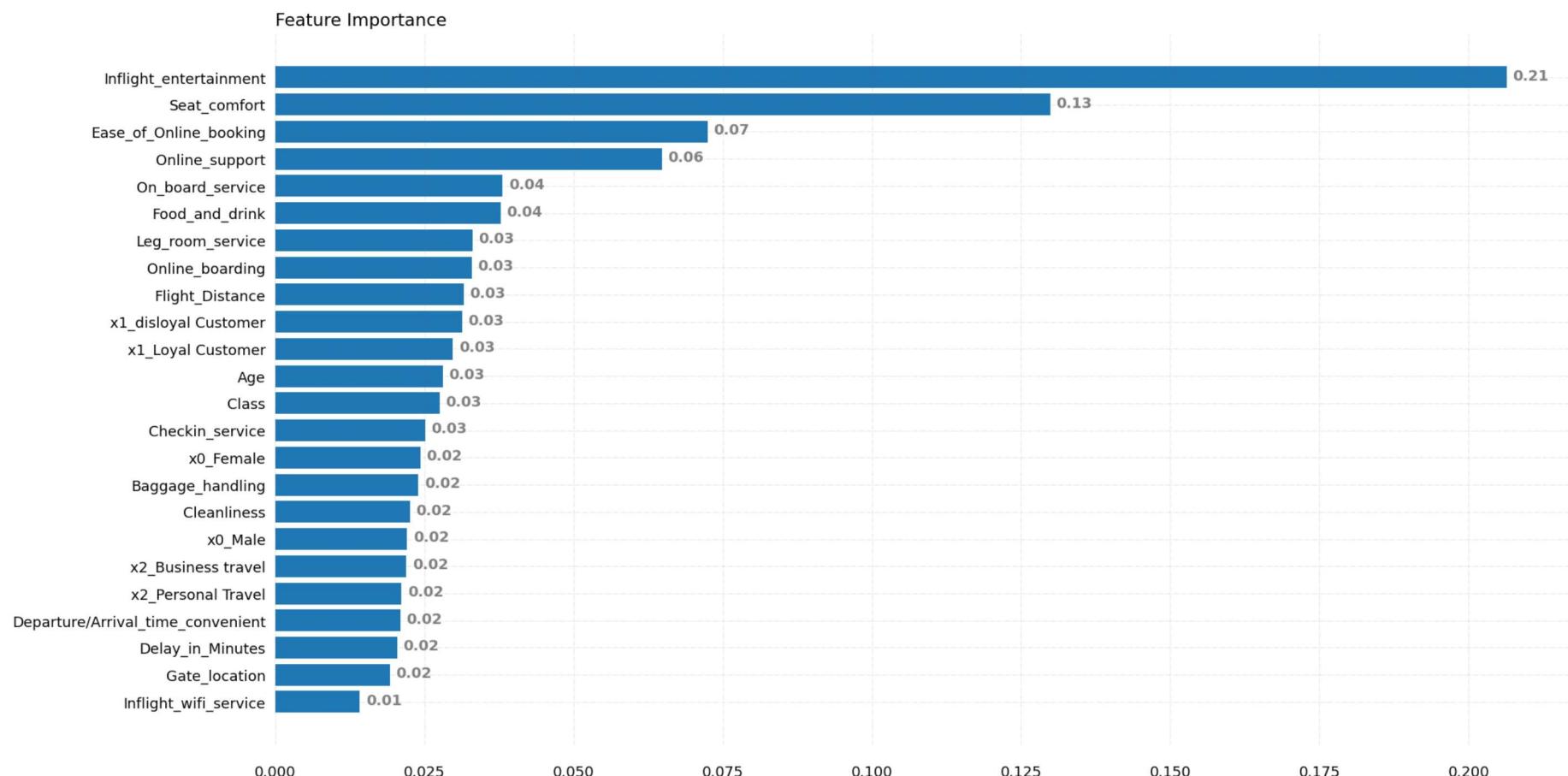


In [81]:

```
1 name = rdf_.index
2 price = rdf_.values
3
4 # Figure Size
5 fig, ax = plt.subplots(figsize =(16,9), dpi= 130)
6
7 # Horizontal Bar Plot
8 ax.barh(name, price)
9
10 # Remove axes splines
11 for s in ['top', 'bottom', 'left', 'right']:
12     ax.spines[s].set_visible(False)
13
14 # Remove x, y Ticks
15 ax.xaxis.set_ticks_position('none')
16 ax.yaxis.set_ticks_position('none')
17
18 # Add padding between axes and Labels
19 ax.xaxis.set_tick_params(pad = 10)
20 ax.yaxis.set_tick_params(pad = 3)
21
22 # Add x, y gridlines
23 ax.grid(b = True, color ='grey',
24          linestyle ='-.', linewidth = 0.5,
25          alpha = 0.2)
26
27 # Show top values
28 ax.invert_yaxis()
29
30 # Add annotation to bars
31 for i in ax.patches:
32     plt.text(i.get_width()+0.001, i.get_y()+0.5,
33               str(round((i.get_width()), 2)),
34               fontsize = 10, fontweight ='bold',
35               color ='grey')
36
37 # Add Plot Title
38 ax.set_title('Feature Importance',
39              loc ='left', )
40
41 # Show Plot
```

```
42 | plt.show()
```

```
43
```



Từ phân tích trực quan và tham số của mô hình, ta thấy rằng chưa làm được hài lòng các khách hàng mới và các dịch vụ chưa tốt làm ảnh hưởng đến độ hài lòng như đồ ăn, thức uống và độ delay của chuyến bay. Độ chênh lệch về sự hài lòng giữa các loại vé còn khá lớn và đang nghiêng về phần không hài lòng, cần xem xét và đưa ra các phương án cải thiện cho từng loại vé khác nhau.

