Classification Project - Hotel Booking Cancellation Prediction

Use the **Danyyen Hotels dataset** for this project.

Context

A significant number of hotel bookings are called off due to cancellations or no-shows. The typical reasons for cancellations include changes of plans, scheduling conflicts, etc. This is often made easier by the option to do so free of charge or preferably at a low cost which is beneficial to hotel guests but it is a less desirable and possibly revenue-diminishing factor for hotels to deal with. Such losses are particularly high on last-minute cancellations.

The new technologies involving online booking channels have dramatically changed customers' booking possibilities and behavior. This adds a further dimension to the challenge of how hotels handle cancellations, which are no longer limited to traditional booking and guest characteristics.

The cancellation of bookings potentially impacts a hotel on various fronts:

1. Loss of resources (revenue) when the hotel cannot resell the room.
2. Additional costs of distribution channels by increasing commissions or paying for publicity to help sell these rooms.
3. Lowering prices at last minute, so the hotel can resell a room, resulting in reducing the profit margin.
4. Human resources to make arrangements for the guests.

Objective

The increasing number of cancellations calls for a Machine Learning based solution that can help in predicting **which booking is likely to be canceled**. Dannyen Hotels Group has a chain of hotels in Lagos Nigeria, they are facing problems with the high number of booking cancellations and have reached out to your firm for data-driven solutions. As a data scientist, you have to analyze the data provided to find which **factors have a high influence on booking cancellations, build a predictive model that can predict which booking is going to be canceled in advance, and help in formulating profitable policies for cancellations and refunds.**

Data Description

The data contains the different attributes of customers' booking details. The detailed data dictionary is given below.

Data Dictionary

- Booking_ID: The unique identifier of each booking
- no_of_adults: The number of adults
- no_of_children: The number of children
- no_of_weekend_nights: The number of weekend nights (Saturday and Sunday) the guest stayed or booked to stay at the hotel
- no_of_week_nights: The number of weeknights (Monday to Friday) the guest stayed or booked to stay at the hotel
- type_of_meal_plan: The type of meal plan booked by the customer:
  - Not Selected – No meal plan selected
  - Meal Plan 1 – Breakfast
  - Meal Plan 2 – Half board (breakfast and one other meal)
  - Meal Plan 3 – Full board (breakfast, lunch, and dinner)
- required_car_parking_space: Does the customer require a car parking space? (0 - No, 1- Yes)
- room_type_reserved: The type of room reserved by the customer. The values are ciphered (encoded) by INN Hotels.
- lead_time: The number of days between the date of booking and the arrival date
- arrival_year: The year of arrival date
- arrival_month: The month of arrival date
- arrival_date: The date of the month
- market_segment_type: Market segment designation.
- repeated_guest: Is the customer a repeated guest? (0 - No, 1- Yes)
- no_of_previous_cancellations: The number of previous bookings that were canceled by the customer before the current booking
- no_of_previous_bookings_not_canceled: The number of previous bookings not canceled by the customer before the current booking
- avg_price_per_room: The average price per day for the reservation; prices of the rooms are dynamic. (in euros)
- no_of_special_requests: The total number of special requests made by the customer (e.g. high floor, view from the room, etc.)
- booking_status: Flag indicating if the booking was canceled or not.

- booking_status: Flag indicating if the booking was canceled or not.

Importing the necessary libraries and overview of the dataset

In [76]:

```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# To scale the data using z-score
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

# Algorithms to use
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

# To tune the model
from sklearn.model_selection import GridSearchCV

# Metrics to evaluate the model
from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve
```

Loading the data

In [77]:

```
hotel = pd.read_csv("DanyyenHotelsGroup.csv")
```

In [78]:

```
# Copying data to another variable to avoid any changes to original data
data = hotel.copy()
```

View the first and the last 5 rows of the dataset

In [79]:

```
data.head()
```

Out[79]:

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_type_reserved
0	INN00001	2	0	1	2	Meal Plan 1	0	Room_Type 1
1	INN00002	2	0	2	3	Not Selected	0	Room_Type 1
2	INN00003	1	0	2	1	Meal Plan 1	0	Room_Type 1
3	INN00004	2	0	0	2	Meal Plan 1	0	Room_Type 1
4	INN00005	2	0	1	1	Not Selected	0	Room_Type 1

In [80]:

```
data.tail()
```

Out[80]:

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_type_reser
36270	INN36271	3	0	2	6	Meal Plan 1	0	Room_Typ
36271	INN36272	2	0	1	3	Meal Plan 1	0	Room_Typ
36272	INN36273	2	0	2	6	Meal Plan 1	0	Room_Typ
36273	INN36274	2	0	0	3	Not Selected	0	Room_Typ
36274	INN36275	2	0	1	2	Meal Plan 1	0	Room_Typ

Checking the info of the data

In [81]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Booking_ID                            36275 non-null  object
 1   no_of_adults                          36275 non-null  int64
 2   no_of_children                        36275 non-null  int64
 3   no_of_weekend_nights                  36275 non-null  int64
 4   no_of_week_nights                     36275 non-null  int64
 5   type_of_meal_plan                     36275 non-null  object
 6   required_car_parking_space            36275 non-null  int64
 7   room_type_reserved                    36275 non-null  object
 8   lead_time                             36275 non-null  int64
 9   arrival_year                          36275 non-null  int64
 10  arrival_month                         36275 non-null  int64
 11  arrival_date                          36275 non-null  int64
 12  market_segment_type                   36275 non-null  object
 13  repeated_guest                        36275 non-null  int64
 14  no_of_previous_cancellations          36275 non-null  int64
 15  no_of_previous_bookings_not_canceled  36275 non-null  int64
 16  avg_price_per_room                    36275 non-null  float64
 17  no_of_special_requests                36275 non-null  int64
 18  booking_status                        36275 non-null  object
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
```

- The dataset has **36,275 rows and 19 columns**.
- Booking_ID , type_of_meal_plan , room_type_reserved , market_segment_type , and booking_status are of **object type** while the rest of the columns are numeric in nature.
- There are **no null values** in the dataset.
- **Booking_ID column is an identifier**. Let's check if each entry of the column is unique.

In [82]:

```
data.Booking_ID.nunique()
```

Out[82]:

```
36275
```

Observations:

- We can see that **all the entries of this column are unique**. Hence, this column would not add any value to our analysis.
- Let's drop this column.

Dropping the Booking_ID column

In [83]:

```
data = data.drop(["Booking_ID"], axis = 1)
```

In [84]:

```
data.head()
```

Out[84]:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_type_reserved	lead_time	a
0	2	0	1	2	Meal Plan 1	0	Room_Type 1	224	
1	2	0	2	3	Not Selected	0	Room_Type 1	5	
2	1	0	2	1	Meal Plan 1	0	Room_Type 1	1	
3	2	0	0	2	Meal Plan 1	0	Room_Type 1	211	
4	2	0	1	1	Not Selected	0	Room_Type 1	48	

Exploratory Data Analysis and Data Preprocessing

Summary Statistics for numerical columns

In [85]:

```
# Selecting numerical columns and checking the summary statistics
num_cols = data.select_dtypes('number').columns

data[num_cols].describe().T
```

Out[85]:

	count	mean	std	min	25%	50%	75%	max
no_of_adults	36275.0	1.844962	0.518715	0.0	2.0	2.00	2.0	4.0
no_of_children	36275.0	0.105279	0.402648	0.0	0.0	0.00	0.0	10.0
no_of_weekend_nights	36275.0	0.810724	0.870644	0.0	0.0	1.00	2.0	7.0
no_of_week_nights	36275.0	2.204300	1.410905	0.0	1.0	2.00	3.0	17.0
required_car_parking_space	36275.0	0.030986	0.173281	0.0	0.0	0.00	0.0	1.0
lead_time	36275.0	85.232557	85.930817	0.0	17.0	57.00	126.0	443.0
arrival_year	36275.0	2017.820427	0.383836	2017.0	2018.0	2018.00	2018.0	2018.0
arrival_month	36275.0	7.423653	3.069894	1.0	5.0	8.00	10.0	12.0
arrival_date	36275.0	15.596995	8.740447	1.0	8.0	16.00	23.0	31.0
repeated_guest	36275.0	0.025637	0.158053	0.0	0.0	0.00	0.0	1.0
no_of_previous_cancellations	36275.0	0.023349	0.368331	0.0	0.0	0.00	0.0	13.0
no_of_previous_bookings_not_canceled	36275.0	0.153411	1.754171	0.0	0.0	0.00	0.0	58.0
avg_price_per_room	36275.0	103.423539	35.089424	0.0	80.3	99.45	120.0	540.0
no_of_special_requests	36275.0	0.619655	0.786236	0.0	0.0	0.00	1.0	5.0

**Observations:

--some customers booked for as much as 17 week nights and 6 weekend nights; the data was gathered from customers who were to arrive between year 2017 and 2018; customers having 10 children might seem to be an outlier or an error.

-- some customers had no cancellations while some had as high 13 cancellations; the average price per room is approx 103 Euros; while some made no special requests, some had up to 5 special requests.

In [86]:

```
# Checking the rows where the avg_price_per_room is 0
data[data["avg_price_per_room"] == 0]
```

Out[86]:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal_plan	required_car_parking_space	room_type_reserved	lead_tim
63	1	0	0	1	Meal Plan 1	0	Room_Type 1	
145	1	0	0	2	Meal Plan 1	0	Room_Type 1	1
209	1	0	0	0	Meal Plan 1	0	Room_Type 1	
266	1	0	0	2	Meal Plan 1	0	Room_Type 1	
267	1	0	2	1	Meal Plan 1	0	Room_Type 1	
...	...	...	...	...	...	...	...	
35983	1	0	0	1	Meal Plan 1	0	Room_Type 7	
36080	1	0	1	1	Meal Plan 1	0	Room_Type 7	
36114	1	0	0	1	Meal Plan 1	0	Room_Type 1	
36217	2	0	2	1	Meal Plan 1	0	Room_Type 2	
36250	1	0	0	2	Meal Plan 2	0	Room_Type 1	

545 rows × 18 columns

- In the market segment column, it looks like **many values are complementary**. Let's check the market segment where the room prices are equal to 0.

In [87]:

```
data.loc[data["avg_price_per_room"] == 0, "market_segment_type"].value_counts()
```

Out[87]:

```
Complementary    354
Online           191
Name: market_segment_type, dtype: int64
```

Observations:

- It makes sense that most values with room prices equal to 0 are the rooms given as a complimentary service by the hotel.
- The rooms booked online might be a part of some promotional campaign done by the hotel. We will not treat these rows as we don't have the data to test this claim.

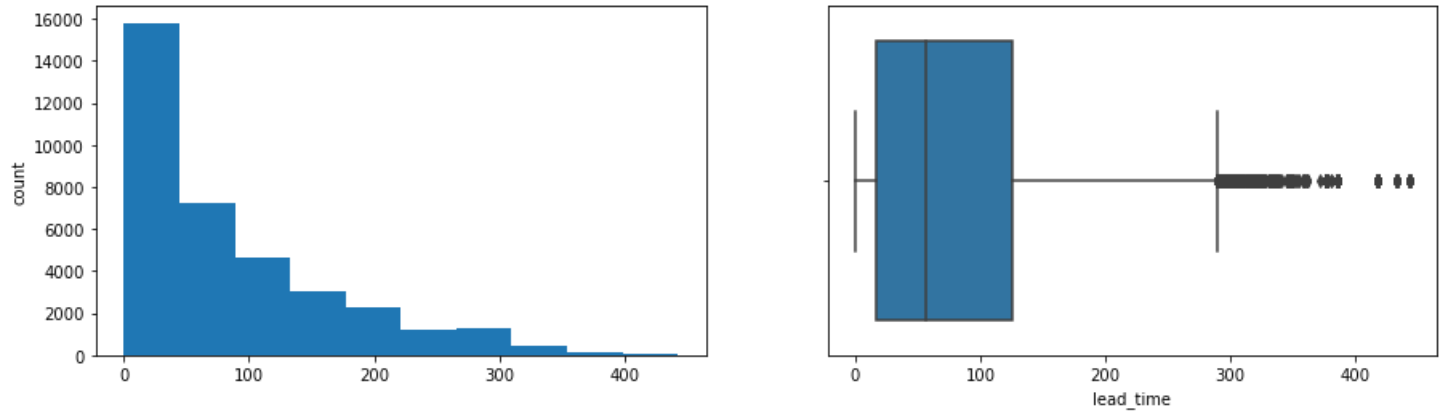
Checking the distribution and outliers for numerical columns in the data

In [88]:

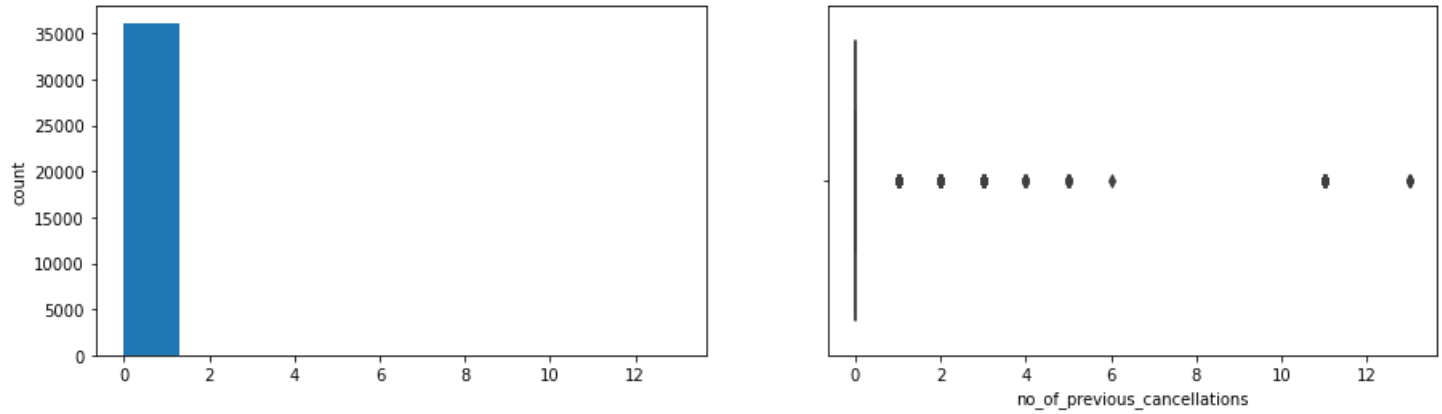
```
for col in ['lead_time', 'no_of_previous_cancellations', 'no_of_previous_bookings_not_canceled', 'avg_price_per_room']:
    print(col)
```

```
print('Skew :', round(data[col].skew(), 2))

plt.figure(figsize = (15, 4))

plt.subplot(1,2,1)

data[col].hist(bins = 10, grid = False)

plt.ylabel('count')

plt.subplot(1, 2, 2)

sns.boxplot(x = data[col])

plt.show()
```

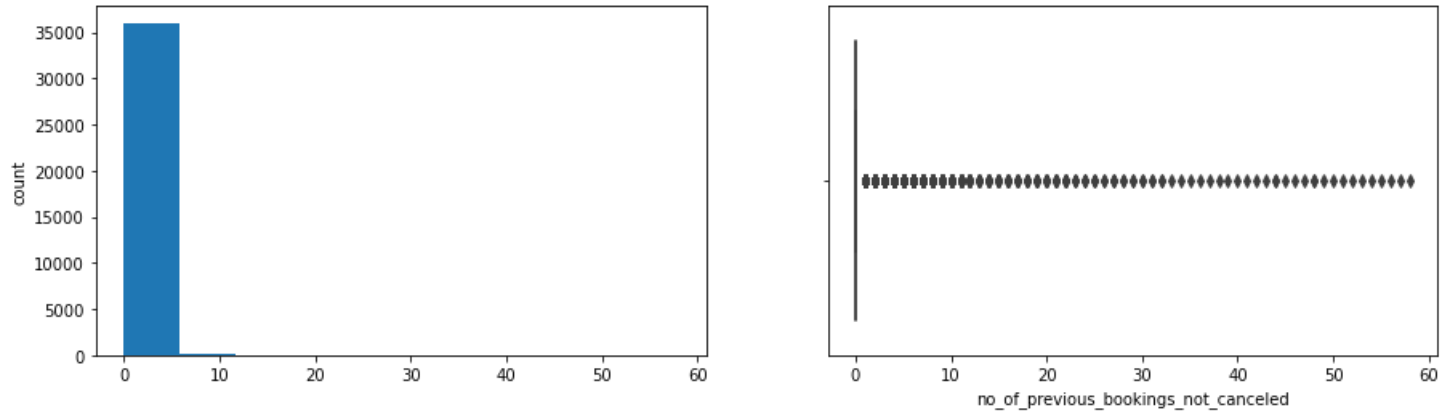
lead_time
Skew : 1.29



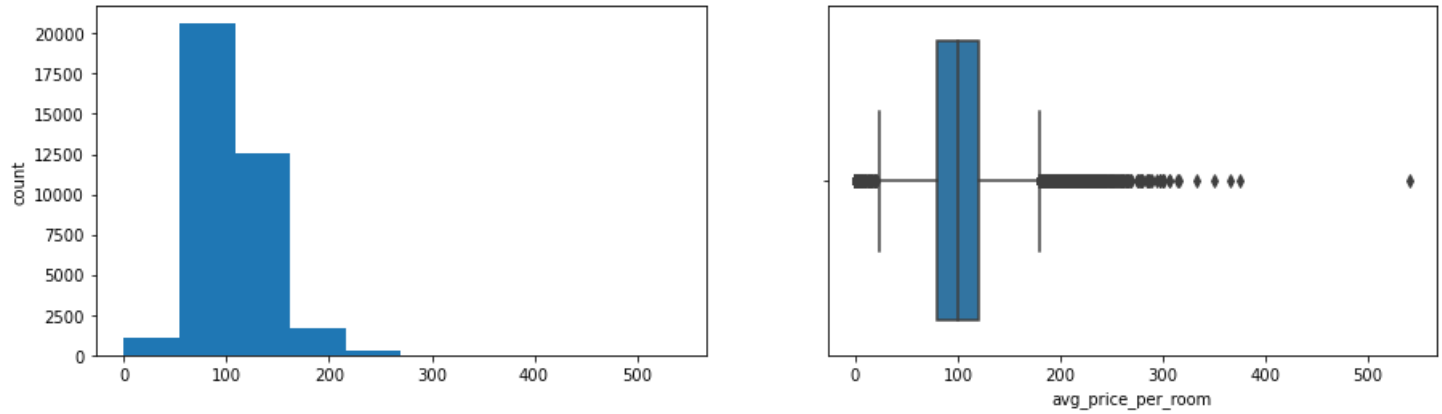
no_of_previous_cancellations
Skew : 25.2



no_of_previous_bookings_not_canceled
Skew : 19.25



avg_price_per_room
Skew : 0.67



- The distribution of **lead time is right-skewed** implies the majority of customer make bookings close to the arrival date. Many customers have made the booking on the same day of arrival as well. There are many outliers, **some customers made booking more than 400 days in advance**.
- **Very few customers have more than one cancellation**. Some customers canceled more than 12 times.
- **Very few customers have more than 1 booking not canceled previously**.
- **The distribution of average price per room is skewed to right**. The boxplot shows that there are outliers on both sides. The median price of a room is around ~100 euros. There is 1 observation where the average price of the room is more than 500 dollars. This observation is quite far away from the rest of the values. We can treat this by clipping the value to the upper whisker (Q3 + 1.5 * IQR).

In [89]:

```
# Calculating the 25th quantile
```

```
Q1 = data["avg_price_per_room"].quantile(0.25)

# Calculating the 75th quantile
Q3 = data["avg_price_per_room"].quantile(0.75)

# Calculating IQR
IQR = Q3 - Q1

# Calculating value of upper whisker
Upper_Whisker = Q3 + 1.5 * IQR
Upper_Whisker
```

Out[89]:

```
179.55
```

In [90]:

```
# Assigning the value of upper whisker to outliers
data.loc[data["avg_price_per_room"] >= 500, "avg_price_per_room"] = Upper_Whisker
```

- **check the percentage of each category for columns mentioned below (cat_cols)**

In [91]:

```
cat_cols = ['no_of_adults', 'no_of_children', 'no_of_week_nights', 'no_of_weekend_nights', 'required_car_parking_space',
       'type_of_meal_plan', 'room_type_reserved', 'arrival_month', 'market_segment_type', 'no_of_special_requests',
       'booking_status']

# check the percentage of each category for columns
for col in cat_cols:
    print('+-----> percentage of each category for',col)
    print((data[col].value_counts() / data[col].value_counts().sum()) * 100)
```

```
+-----> percentage of each category for no_of_adults
2    71.972433
1    21.212957
3     6.387319
0     0.383184
4     0.044108
Name: no_of_adults, dtype: float64
+-----> percentage of each category for no_of_children
0     92.562371
1      4.460372
2      2.916609
3      0.052378
9      0.005513
10     0.002757
Name: no_of_children, dtype: float64
+-----> percentage of each category for no_of_week_nights
2     31.547898
1     26.155755
3     21.609924
4      8.242591
0      6.580289
5      4.449345
6      0.521020
7      0.311509
10     0.170917
8      0.170917
9      0.093728
11     0.046864
15     0.027567
12     0.024810
14     0.019297
13     0.013784
17     0.008270
16     0.005513
Name: no_of_week_nights, dtype: float64
+-----> percentage of each category for no_of_weekend_nights
0    46.511371
1    27.553411
2    25.006203
3     0.421778
4     0.355617
5     0.093728
6     0.055134
7     0.002757
Name: no_of_weekend_nights, dtype: float64
+-----> percentage of each category for required_car_parking_space
0    96.901447
1     3.098553
```

```
Name: required_car_parking_space, dtype: float64
+-----> percentage of each category for type_of_meal_plan
Meal Plan 1      76.733287
Not Selected     14.141971
Meal Plan 2       9.110958
Meal Plan 3       0.013784
Name: type_of_meal_plan, dtype: float64
+-----> percentage of each category for room_type_reserved
Room_Type 1     77.546520
Room_Type 4     16.697450
Room_Type 6      2.662991
Room_Type 2      1.907650
Room_Type 5      0.730531
Room_Type 7      0.435562
Room_Type 3      0.019297
Name: room_type_reserved, dtype: float64
+-----> percentage of each category for arrival_month
10    14.657478
9     12.711234
8     10.511371
6      8.829773
12     8.328050
11     8.215024
7      8.049621
4      7.542385
5      7.161957
3      6.500345
2      4.697450
1      2.795314
Name: arrival_month, dtype: float64
+-----> percentage of each category for market_segment_type
Online           63.994487
Offline          29.022743
Corporate         5.560303
Complementary     1.077877
Aviation          0.344590
Name: market_segment_type, dtype: float64
+-----> percentage of each category for no_of_special_requests
0    54.519642
1    31.352171
2    12.030324
3     1.860786
4     0.215024
5     0.022054
Name: no_of_special_requests, dtype: float64
+-----> percentage of each category for booking_status
Not_Canceled    67.236389
Canceled        32.763611
Name: booking_status, dtype: float64
```

**Observations:

-- customer bookings of 2 adults had the highest percentage while 4 adults had the least; 67% didn't cancel while approx 33% canceled; most customers didn't have special requests; online most market_segment; most customers arrival bookings were in the months of October and September; most customers booked room_type1; most didn't want parking space; most wanted breakfast; most booking customers didn't have children; most bookings for week nights were 2.

Replacing values 9 and 10 for the number of children with 3 and encoding the target variable

In [92]:

```
# Replacing values 9 and 10 with 3 for the column no_of_children
data["no_of_children"] = data["no_of_children"].replace([9, 10], 3)
```

In [93]:

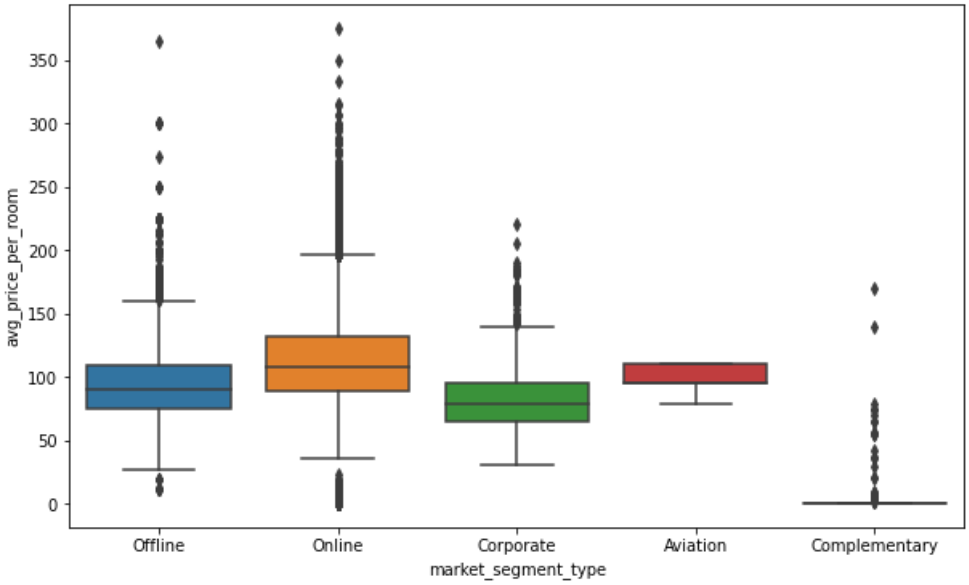
```
data["booking_status"] = data["booking_status"].apply(lambda x: 1 if x == "Canceled" else 0)
```

Bivariate analysis.

Check the relationship of market segment type with the average price per room.

In [94]:

```
plt.figure(figsize = (10, 6))

sns.boxplot(data = data, x = "market_segment_type", y = "avg_price_per_room")

plt.show()
```



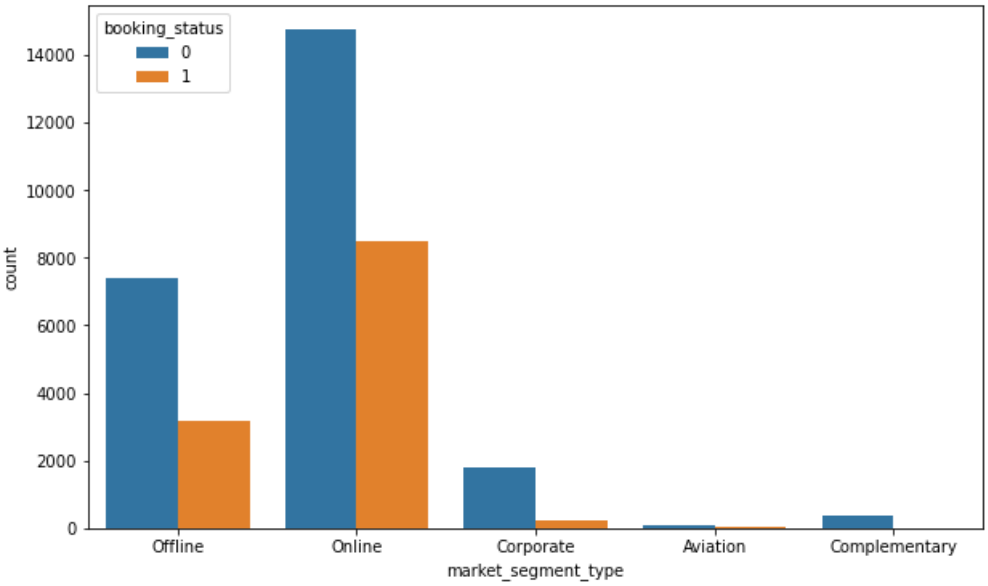
Observations:

- **Rooms booked online have the highest variations in prices.**
- The distribution for offline and corporate room prices are almost similar except for some outliers.
- Complementary market segment gets the rooms at very low prices, which makes sense.

Check how booking status varies across different market segments and also lead time impacts booking status.

In [95]:

```
plt.figure(figsize = (10, 6))

sns.countplot(x = 'market_segment_type', hue = 'booking_status', data = data)

plt.show()
```

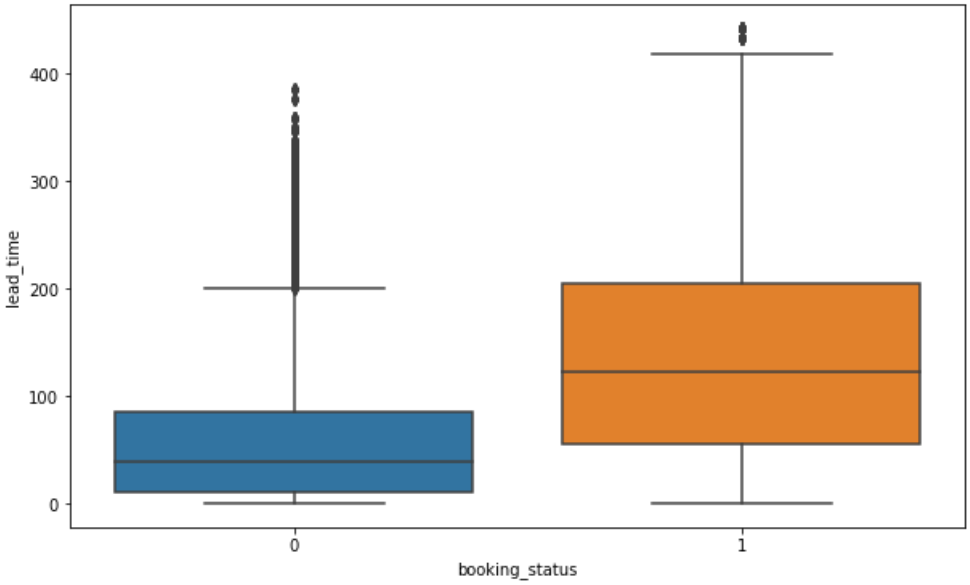


Observations:

- **Online bookings have the highest number of cancellations.**
- Bookings made offline are less prone to cancellations.
- Corporate and complementary segment also show very low number of cancellations.

In [96]:

```
plt.figure(figsize = (10, 6))

sns.boxplot(data = data, x = "booking_status", y = "lead_time")

plt.show()
```



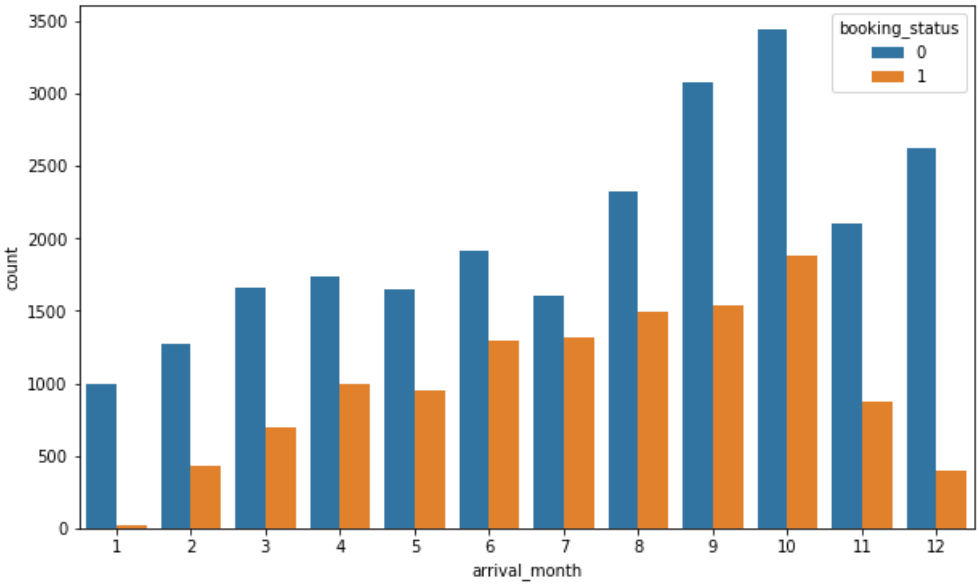
Observations:

- There's a big difference in the median value of lead time for bookings that were canceled and bookings that were not canceled.
- **The higher the lead time, the higher are the chances of a booking being canceled.**

Check how the arrival month impacts the booking status.

In [97]:

```
plt.figure(figsize = (10, 6))

sns.countplot(x = 'arrival_month', hue = 'booking_status', data = data)

plt.show()
```



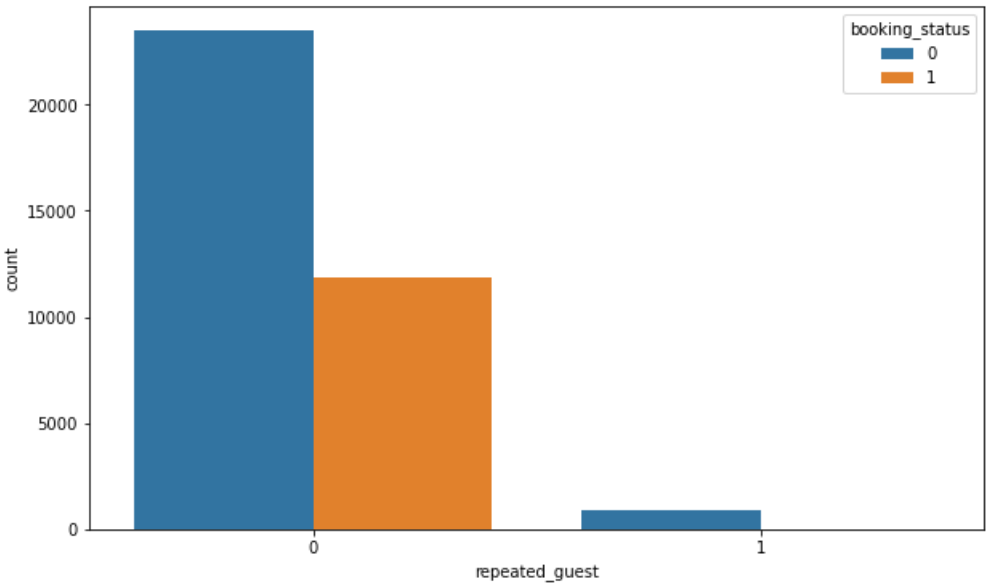
Observations:

- We observed earlier that the month of October has the highest number of bookings but the above plot shows that **October has the highest number of cancellations** as well.
- Bookings made for **December and January are least prone to cancellations**.

Repeating guests are the guests who stay in the hotel often and are important to brand equity. Check the percentage of repeating guests cancel

In [98]:

```
plt.figure(figsize = (10, 6))

sns.countplot(x = 'repeated_guest', hue = 'booking_status', data = data)

plt.show()
```



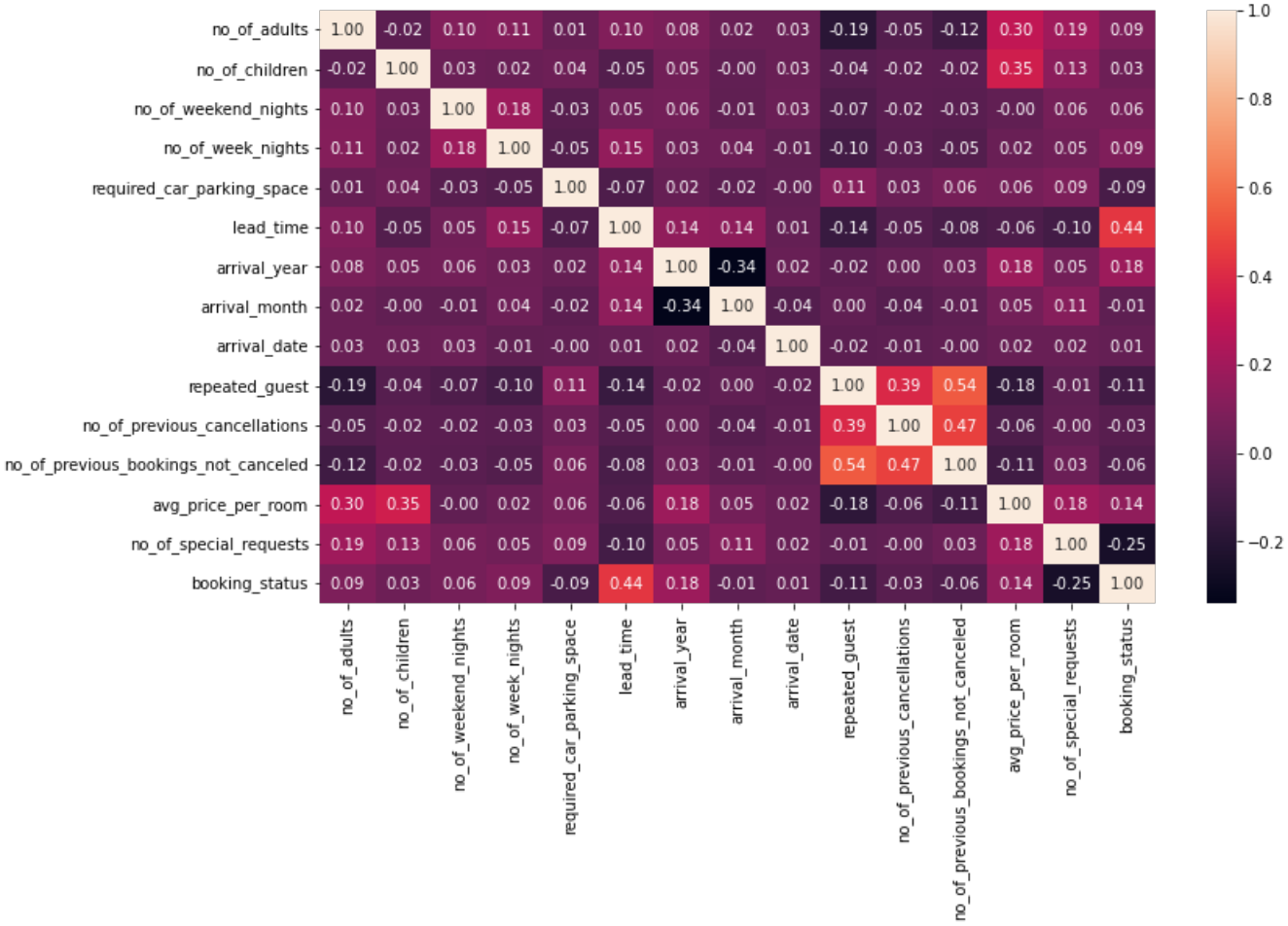
Observations:

- There are **very few repeat customers but the cancellation among them is very less**.
- This is a good indication as repeat customers are important for the hospitality industry as they can help in spreading the word of mouth.

Check the pairwise correlations between all the variables.

In [99]:

```
plt.figure(figsize = (12, 7))

sns.heatmap(data.corr(), annot = True, fmt = ".2f")

plt.show()
```



Observations:

- There's a **weak positive correlation between the number of customers (adults and children) and the average price per room**. This makes sense as more the number of customers more the price of the rooms.
- There's a **weak negative correlation between average room price and repeated guests**. The hotel might be giving some loyalty benefits to the customers.
- **Repeated guests have a positive correlation with the number of previous bookings canceled and previous bookings not canceled.** This implies that repeated customers are also likely to cancel their bookings.
- There's a weak positive correlation between lead time and the number of weeknights a customer is planning to stay in the hotel.
- There's a **positive correlation between booking status and lead time**, indicating higher the lead time higher are the chances of cancellation.
- There's a weak negative correlation between the number of special requests from the customer and the booking status, indicating **if a customer has made some special requests the chances of cancellation might decrease**.

Preparing the data for modeling

- Models cannot take non-numeric inputs. So, we will first create dummy variables for all the categorical variables.
- Then split the data into train and test sets.

- **Drop the target variable from the original data and store it in a separate DataFrame X**
- **Store the target variable in a separate series Y**

In [100]:

```
# Remove the blanks and complete the below code
X = data.drop(columns = ['booking_status'])
Y = data['booking_status']
```

In [101]:

```
# Creating dummy variables, drop_first = True is used to avoid redundant variables
X = pd.get_dummies(X, drop_first = True)
```

In [102]:

```
# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.30, random_state = 1)
```

Building Classification Models

Before training the model, choose the appropriate model evaluation criterion as per the problem at hand.

Model evaluation criterion

Model can make wrong predictions as:

1. Predicting a customer will not cancel their booking but in reality, the customer cancels their booking.
2. Predicting a customer will cancel their booking but in reality, the customer does not cancel their booking.

Which case is more important?

- Both the cases are important as:
- If we predict that a booking will not be canceled and the booking gets canceled, then the hotel will lose resources and will have to bear additional costs of unsold rooms. The hotel might also have to bear an additional cost of advertising the room again on different distribution channels.
- If we predict that a booking will get canceled and the booking doesn't get canceled, then the hotel might not be able to provide satisfactory services to the customer by assuming that this booking will be canceled. This might damage the brand equity.

How to reduce the losses?

- Hotel would want F1 Score to be maximized, greater the F1 score, higher are the chances of minimizing False Negatives and False Positives.

Create a function to calculate and print the classification report and confusion matrix so that we don't have to rewrite the same code repeatedly for each model.

In [103]:

```
# Function to print classification report and get confusion matrix in a proper format

def metrics_score(actual, predicted):
    print(classification_report(actual, predicted))

    cm = confusion_matrix(actual, predicted)

    plt.figure(figsize = (8, 5))

    sns.heatmap(cm, annot = True,  fmt = '.2f', xticklabels = ['Not Canceled', 'Canceled'], yticklabels = ['Not Canceled', 'Canceled'])

    plt.ylabel('Actual')

    plt.xlabel('Predicted')
```

```
plt.show()
```

Logistic Regression

Fit the logistic regression model on the train dataset using random_state = 1

In [104]:

```
# Define Logistic Regression model
log_reg= LogisticRegression()

# Fit the model
log_reg.fit(X_train,y_train)
```

Out[104]:

LogisticRegression()

Check the coefficient of each dependent variable in the data.

In [105]:

```
pd.Series(log_reg.coef_[0], index = X_train.columns).sort_values(ascending = False)
```

Out[105]:

```
market_segment_type_Online              0.623273
type_of_meal_plan_Not Selected          0.321190
no_of_weekend_nights                    0.180888
avg_price_per_room                      0.019750
lead_time                               0.015394
no_of_adults                            0.011349
no_of_week_nights                       0.006807
arrival_date                            0.001291
type_of_meal_plan_Meal Plan 3           0.000400
room_type_reserved_Room_Type 3          0.000341
arrival_year                           -0.001733
room_type_reserved_Room_Type 2         -0.004280
market_segment_type_Complementary      -0.008772
room_type_reserved_Room_Type 5         -0.011115
room_type_reserved_Room_Type 7         -0.017796
no_of_previous_cancellations           -0.024654
market_segment_type_Corporate          -0.031241
room_type_reserved_Room_Type 4         -0.032696
repeated_guest                         -0.043363
room_type_reserved_Room_Type 6         -0.045539
no_of_children                         -0.053783
arrival_month                          -0.057509
type_of_meal_plan_Meal Plan 2          -0.094188
required_car_parking_space             -0.138919
no_of_previous_bookings_not_canceled   -0.212120
market_segment_type_Offline            -0.597497
no_of_special_requests                 -1.548694
dtype: float64
```

**Observations:

booking features that are likely to lead to booking cancellations are:

--market_segment_type_Online

--type_of_meal_plan_Not Selected

--no_of_weekend_nights

booking features that are likely to lead to customers showing up (not cancelling booking) are:

--no_of_special_requests

--market_segment_type_Offline

--no_of_previous_bookings_not_canceled

In [106]:

```
# Finding the odds
odds = np.exp(log_reg.coef_[0])

# Adding the odds to a dataframe and sorting the values
pd.DataFrame(odds, X_train.columns, columns = ['odds']).sort_values(by = 'odds', ascending = False)
```

Out[106]:

	odds
market_segment_type_Online	1.865023
type_of_meal_plan_Not Selected	1.378768
no_of_weekend_nights	1.198281
avg_price_per_room	1.019947
lead_time	1.015513
no_of_adults	1.011414
no_of_week_nights	1.006830
arrival_date	1.001292
type_of_meal_plan_Meal Plan 3	1.000400
room_type_reserved_Room_Type 3	1.000341
arrival_year	0.998268
room_type_reserved_Room_Type 2	0.995730
market_segment_type_Complementary	0.991267
room_type_reserved_Room_Type 5	0.988947
room_type_reserved_Room_Type 7	0.982361
no_of_previous_cancellations	0.975648
market_segment_type_Corporate	0.969242
room_type_reserved_Room_Type 4	0.967832
repeated_guest	0.957564
room_type_reserved_Room_Type 6	0.955482
no_of_children	0.947638
arrival_month	0.944114
type_of_meal_plan_Meal Plan 2	0.910112
required_car_parking_space	0.870298
no_of_previous_bookings_not_canceled	0.808868
market_segment_type_Offline	0.550187
no_of_special_requests	0.212525

**Observations:

-- the odds of a customer in the online market segment who didn't show up booking is 1.86(85% higher) times higher than other market segments

-- the odd of a customer who didn't select any meal not showing up after booking is 1.35(35% higher) times higher than other customers that selected other type of meal plans

--customers who booked for weekend nights have a 19% chance of not showing up after booking.

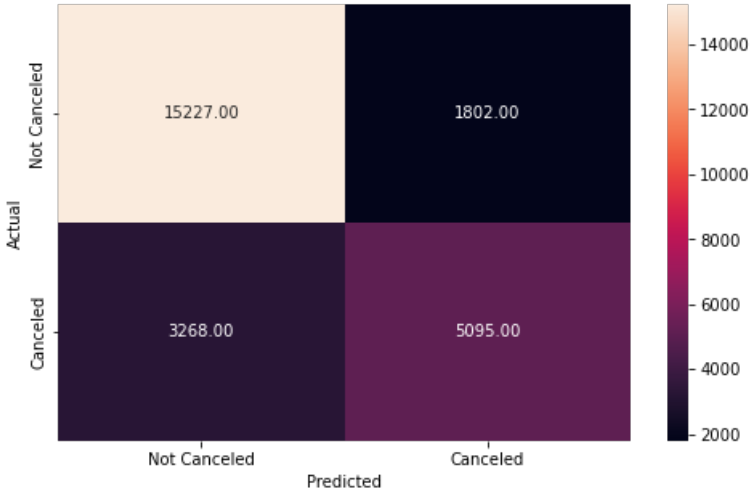
Check the performance of the model on the training set.

Check the performance on the training data and write your observations from the below classification report and confusion matrix for the training set

In [107]:

```
# Checking the performance on the training data

y_pred_train = log_reg.predict(X_train)

metrics_score(y_train, y_pred_train)
```

	precision	recall	f1-score	support
0	0.82	0.89	0.86	17029
1	0.74	0.61	0.67	8363
accuracy			0.80	25392
macro avg	0.78	0.75	0.76	25392
weighted avg	0.80	0.80	0.79	25392



Reading confusion matrix (clockwise):

- **True Positive**: Predicting the customer will not cancel the booking and the customer does not cancel the booking.
- **False Negative**: Predicting the customer will cancel the booking but the customer does not cancel the booking.
- **True Negative**: Predicting the customer will cancel the booking and the customer cancels the booking.
- **False Positive**: Predicting the customer will not cancel the booking but the customer cancels the booking.

**Observations:

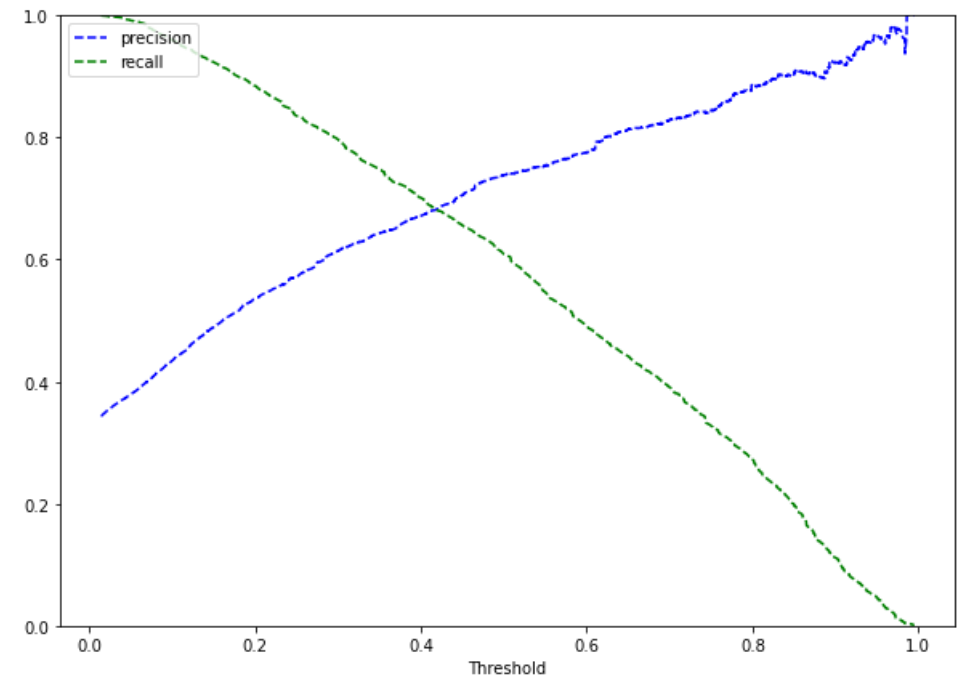
--The model is performing well in terms of accuracy (80%)

--The F1 Score did better for predicting bookings that will not cancelled (86%) than predicting bookings that will be cancelled (67%); the macro avg F1 score is 76%, which means the model can give a 76% accuracy for False positive and false negative in predicting for both bookings that will be cancelled and bookings that won't be cancelled.

Precision-Recall Curve for Logistic Regression

In [108]:

```
# predict_proba gives the probability of each observation belonging to each class

y_scores = log_reg.predict_proba(X_train)

precisions, recalls, thresholds = precision_recall_curve(y_train, y_scores[:,1])

# Plotting values of precisions, recalls, and thresholds
plt.figure(figsize = (10, 7))

plt.plot(thresholds, precisions[:-1], 'b--', label = 'precision')

plt.plot(thresholds, recalls[:-1], 'g--', label = 'recall')

plt.xlabel('Threshold')

plt.legend(loc = 'upper left')

plt.ylim([0, 1])

plt.show()
```



Observations:

- We can see that **the precision and the recall are balanced for the threshold of about 0.4.**
- Try to calculate the exact threshold where precision and recall are equal.

In [109]:

```
# Calculating the exact threshold where precision and recall are equal
for i in np.arange(len(thresholds)):
    if precisions[i] == recalls[i]:
        print(thresholds[i])
```

0.41785529601946925

- The threshold of 0.42 would give a balanced precision and recall.

Compare the performance of the model on the training set after changing the threshold and check the performance on the testing set

In [110]:

```
optimal_threshold1 = 0.42

metrics_score(y_train, y_scores[:, 1] > optimal_threshold1)
```

	precision	recall	f1-score	support
0	0.84	0.84	0.84	17029
1	0.68	0.68	0.68	8363
accuracy			0.79	25392
macro avg	0.76	0.76	0.76	25392
weighted avg	0.79	0.79	0.79	25392

**Observation:

--using a threshold of 0.42 the F1 score for bookings likely not cancelled decreased while bookings likely to be cancelled increased and accuracy was down to 79% compared to the previous confusion matrix. this matrix will also likely do better in predicting bookings that were not cancelled compared to bookings that were cancelled.

Check the performance of the model on the test data.

In [111]:

```
# Checking performance on the testing data
y_pred_test = log_reg.predict_proba(X_test)

metrics_score(y_test, y_pred_test[:, 1] > optimal_threshold1)
```

```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      7361
           1       0.68      0.68      0.68      3522

    accuracy                           0.79     10883
   macro avg       0.77      0.77      0.77     10883
weighted avg       0.79      0.79      0.79     10883
```

**Observations:

--The model is giving a similar performance on the test datasets, i.e., the model is giving a generalized good performance with a macro average F1 score of 77%, accuracy of 79% on test data.

K-Nearest Neighbors (K-NN)

- K-NN is a distance-based algorithm and all distance-based algorithms are affected by the scale of the data.
- Scale the attributes (DataFrame X defined above) before building the K-NN model.
- Then identify the value of K to be used in K-NN. We will use **GridSearchCV** to find the optimal value of K along with other hyperparameters.

In [112]:

```
# Scaling the data
scaler = StandardScaler()

# fit_transform the training data
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns = X_train.columns)

# Transform the testing data
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns = X_test.columns)
```

Using GridSearchCV for find the value of K and other hyperparameters

Points to note:

- Hyperparameter tuning is tricky in the sense that there is no direct way to calculate how a change in the hyperparameter value will reduce the loss of your model, so we usually resort to experimentation.
- **Grid search** is a tuning technique that attempts to compute the optimum values of hyperparameters.
- Grid search is an exhaustive search of values that tries many iterations to compute the optimum values of hyperparameters. So, **it might take up to 30 minutes for the code to run depending on the number of values and hyperparameters passed.**
- The hyperparameters that we are tuning are:
  - **n_neighbors**: Number of neighbors to use.
  - **weights={'uniform', 'distance'}**
    - uniform : uniform weights. All points in each neighborhood are weighted equally.
    - distance : weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors that are further away.
  - **p**: When p = 1, this is equivalent to using Manhattan_distance (L1), and Euclidean_distance (L2) is used for p = 2.

In [113]:

```
knn = KNeighborsClassifier()

params_knn = {'n_neighbors':np.arange(2, 20, 2), 'weights':['uniform','distance'], 'p':[1, 2]}

grid_knn = GridSearchCV(estimator = knn, param_grid = params_knn, scoring = 'f1', cv = 10)

model_knn = grid_knn.fit(X_train_scaled,y_train)

knn_estimator = model_knn.best_estimator_

print(knn_estimator)
```

```
KNeighborsClassifier(n_neighbors=14, p=1, weights='distance')
```

- **Fit the KNN model on the scaled training data using the optimal values of hyperparameters obtained from GridSearchCV**
- **Check the performance of the model on the scaled training and testing sets**
- **Compare the performance and write your observations**

In [114]:

```
# Fit the best estimator on the training data
knn_estimator.fit(X_train, y_train)
```

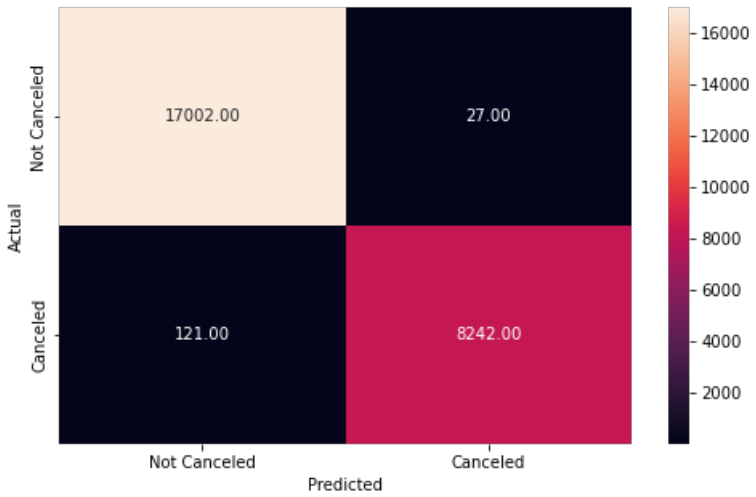
Out[114]:

```
KNeighborsClassifier(n_neighbors=14, p=1, weights='distance')
```

In [115]:

```
# Make predictions on the scaled training data and check the performance (using metrics_score function)

y_pred_train = y_pred_train_knn_estimator = knn_estimator.predict(X_train)

metrics_score(y_train, y_pred_train_knn_estimator)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     17029
           1       1.00      0.99      0.99      8363

    accuracy                           0.99     25392
   macro avg       0.99      0.99      0.99     25392
weighted avg       0.99      0.99      0.99     25392
```



In [116]:

```
# Make predictions on the scaled testing data and check the performance (using metrics_score function)
y_pred_test = knn_estimator.predict(X_test)

metrics_score(y_test, y_pred_test)
```

	precision	recall	f1-score	support
0	0.86	0.93	0.89	7361
1	0.82	0.68	0.74	3522
accuracy			0.85	10883
macro avg	0.84	0.81	0.82	10883
weighted avg	0.85	0.85	0.84	10883

**Observations--

--This model seems to be overfitting for training dataset on bookings that were not cancelled and almost overfitting for bookings that were cancelled; the results have significantly improved in comparison to previous models.

--Accuracy and F1 score have significantly increased on test data by the tunning classifier..There is an 89% chance that the model will detect bookings that wont be cancelled and 74% chance the model will detect bookings that will be cancelled. Macro average of F1 score is 82%.

Conclusion:

--We performed EDA, univariate and bivariate analysis, on all the variables in the dataset.

--Created Dummy Variables because models cannot take non-numeric inputs; Split the data into training and testing set; then determined that the classification is imbalanced so we used F1-Score in confusion Matrix

--We built the Logistic Regession model with relevant features; checked model performance with test data.

--Applied KNN model (using GridSeachCV for optimal K value) for comparison with Logistic Regression; checked model performance with test data

Recommendations:

--We can see that bookings that are likely to be cancelled are mainly customers who are under the online_market_segment; these are customers who made reservations online. These kind of customers should be giving less priority compared to other market segments in an event were they book same date with other market segment types.

--Also Customers that won't select any breakfast are also likely not to show up, probably because they have a high lead time that can make them in decisive of what to eat.The Hotel should likely focus on those that selected a meal because are more likely to show up in an event were they book same time with those who don't select any meal

-- Customers who are likely to show up after booking are those who made Special requests; these special requests can be a result of them being old customers or new customers based on recommendations or good reviews. These kind of customers are more assertive of their bookings

--those who book offline have a high chance of showing up, they took the energy to come to the hotel front desk to make this reservation instead of doing it online; they should be encourage with other hotel services that are not free on a norm.

-- those who have not previously canceled there bookings are likely to show up again; these kind of customized are likely to be those who plan ahead and keep to time/appointments.

WAYS TO REDUCE CANCELLATION OR LOSS TO THE HOTEL AND ALSO IMPROVE CUSTOMER RELATIONSHIPS

Customers could be giving giving discounts or addtional complimentary meals if they show up just reduce their cancellations or make them pay an initial deposit to secure the booking or the hotel can come up with a booking reservation time frame for payment to be made or else the reservation will be giving to someone else.

In [ ]: