# Project - Classification and Hypothesis Testing: Hotel Booking Cancellation Prediction

## Marks: 40

---

## Problem Statement

### Context

**A significant number of hotel bookings are called off due to cancellations or no-shows.** Typical reasons for cancellations include change 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. This may be 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.

This pattern of cancellations of bookings 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 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

This increasing number of cancellations calls for a Machine Learning based solution that can help in predicting which booking is likely to be canceled. INN Hotels Group has a chain of hotels in Portugal - they are facing problems with this high number of booking cancellations and have reached out to your firm for data-driven solutions. You, as a Data Scientist, 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:** Unique identifier of each booking
- **no_of_adults:** Number of adults
- **no_of_children:** Number of children
- **no_of_weekend_nights:** Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
- **no_of_week_nights:** Number of weekday nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- **type_of_meal_plan:** 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:** Type of room reserved by the customer. The values are ciphered (encoded) by INN Hotels.
- **lead_time:** Number of days between the date of booking and the arrival date
- **arrival_year:** Year of arrival date
- **arrival_month:** Month of arrival date
- **arrival_date:** 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:** Number of previous bookings that were canceled by the customer prior to the current booking
- **no_of_previous_bookings_not_canceled:** Number of previous bookings not canceled by the customer prior to the current booking
- **avg_price_per_room:** Average price per day of the reservation; prices of the rooms are dynamic. (in euros)
- **no_of_special_requests:** 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.

## Importing the libraries required

In [38]:
```
# Importing the basic libraries we will require for the project

# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# Libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

# Importing the Machine Learning models we require from Scikit-Learn
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier

# Importing the other functions we may require from Scikit-Learn
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, OneHotEncoder

# To get diferent metric scores
from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve,recall_score

# Code to ignore warnings from function usage
import warnings;
import numpy as np
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
```

## Loading the dataset

In [2]:
```
#hotel = pd.read_csv("INNHotelsGroup.csv")
# Copying data to another variable to avoid any changes to original data
#data = hotel.copy()
```

In [5]:
```
from google.colab import drive
drive.mount('/content/drive')
file = "/content/INNHotelsGroup.csv"
data = pd.read_csv(file)
```

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

## Overview of the dataset

### View the first and last 5 rows of the dataset

Let's **view the first few rows and last few rows** of the dataset in order to understand its structure a little better.

We will use the head() and tail() methods from Pandas to do this.

In [6]:
```
data.head()
```

Out[6]:

| | Booking_ID | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | type_of_meal_plan | required_car_parking_space | roon |
|---|---|---|---|---|---|---|---|---|
| **0** | INN00001 | 2 | 0 | 1 | 2 | Meal Plan 1 | 0 | |
| **1** | INN00002 | 2 | 0 | 2 | 3 | Not Selected | 0 | |
| **2** | INN00003 | 1 | 0 | 2 | 1 | Meal Plan 1 | 0 | |
| **3** | INN00004 | 2 | 0 | 0 | 2 | Meal Plan 1 | 0 | |
| **4** | INN00005 | 2 | 0 | 1 | 1 | Not Selected | 0 | |

In [7]:
```
data.tail()
```

Out[7]:

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

## Understand the shape of the dataset

In [8]:
```
data.shape
```

Out[8]:
```
(36275, 19)
```

- The dataset has 36275 rows and 19 columns.

## Check the data types of the columns for the dataset

In [9]:
```
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
```

- `Booking_ID`, `type_of_meal_plan`, `room_type_reserved`, `market_segment_type`, and `booking_status` are of object type while rest columns are numeric in nature.

- There are no null values in the dataset.

## Dropping duplicate values

In [10]:
```
# checking for duplicate values
data.duplicated().sum()
```

Out[10]:
```
0
```

- There are **no duplicate values** in the data.

## Dropping the unique values column

**Let's drop the Booking_ID column first before we proceed forward**, as a column with unique values will have almost no predictive power for the Machine Learning problem at hand.

In [11]:
```
data = data.drop(["Booking_ID"], axis=1)
```

In [12]:
```
data.head()
```

Out[12]:

| | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | type_of_meal_plan | required_car_parking_space | room_type_reserv |
|---|---|---|---|---|---|---|---|
| **0** | 2 | 0 | 1 | 2 | Meal Plan 1 | 0 | Room_Typ |
| **1** | 2 | 0 | 2 | 3 | Not Selected | 0 | Room_Typ |
| **2** | 1 | 0 | 2 | 1 | Meal Plan 1 | 0 | Room_Typ |
| **3** | 2 | 0 | 0 | 2 | Meal Plan 1 | 0 | Room_Typ |
| **4** | 2 | 0 | 1 | 1 | Not Selected | 0 | Room_Typ |

## Question 1: Check the summary statistics of the dataset and write your observations (2 Marks)

**Let's check the statistical summary of the data.**

In [13]:
```python
# Remove _________ and complete the code
data.describe()
```

Out[13]:

| | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | required_car_parking_space | lead_time | arrival_year | arri |
|---|---|---|---|---|---|---|---|---|
| **count** | 36275.000000 | 36275.000000 | 36275.000000 | 36275.000000 | 36275.000000 | 36275.000000 | 36275.000000 | 36 |
| **mean** | 1.844962 | 0.105279 | 0.810724 | 2.204300 | 0.030986 | 85.232557 | 2017.820427 | |
| **std** | 0.518715 | 0.402648 | 0.870644 | 1.410905 | 0.173281 | 85.930817 | 0.383836 | |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2017.000000 | |
| **25%** | 2.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 17.000000 | 2018.000000 | |
| **50%** | 2.000000 | 0.000000 | 1.000000 | 2.000000 | 0.000000 | 57.000000 | 2018.000000 | |
| **75%** | 2.000000 | 0.000000 | 2.000000 | 3.000000 | 0.000000 | 126.000000 | 2018.000000 | |
| **max** | 4.000000 | 10.000000 | 7.000000 | 17.000000 | 1.000000 | 443.000000 | 2018.000000 | |

**Write your answers here:**

The average lead time for making a booking is high (85 days) and the standard deviation is also significant (86 days), which indicates that customers tend to book well in advance. This information could be useful for revenue management and forecasting, as it allows the hotel to adjust its pricing strategy and inventory availability accordingly.

The average number of adults per booking is close to 2, and the majority of bookings only have 1 or 2 adults. This information could help the hotel to determine the type of rooms and amenities that are in high demand and tailor its services accordingly.

The average number of special requests per booking is 0.6, with a standard deviation of 0.8. This information could be used to understand the types of amenities or services that are in high demand and potentially upsell these services to customers.

The maximum number of nights stayed by a customer is 17, which could indicate that the hotel mainly attracts leisure travelers rather than business travelers. This information could be useful for tailoring marketing and promotional activities to attract more business travelers.

# Exploratory Data Analysis

## Question 2: Univariate Analysis

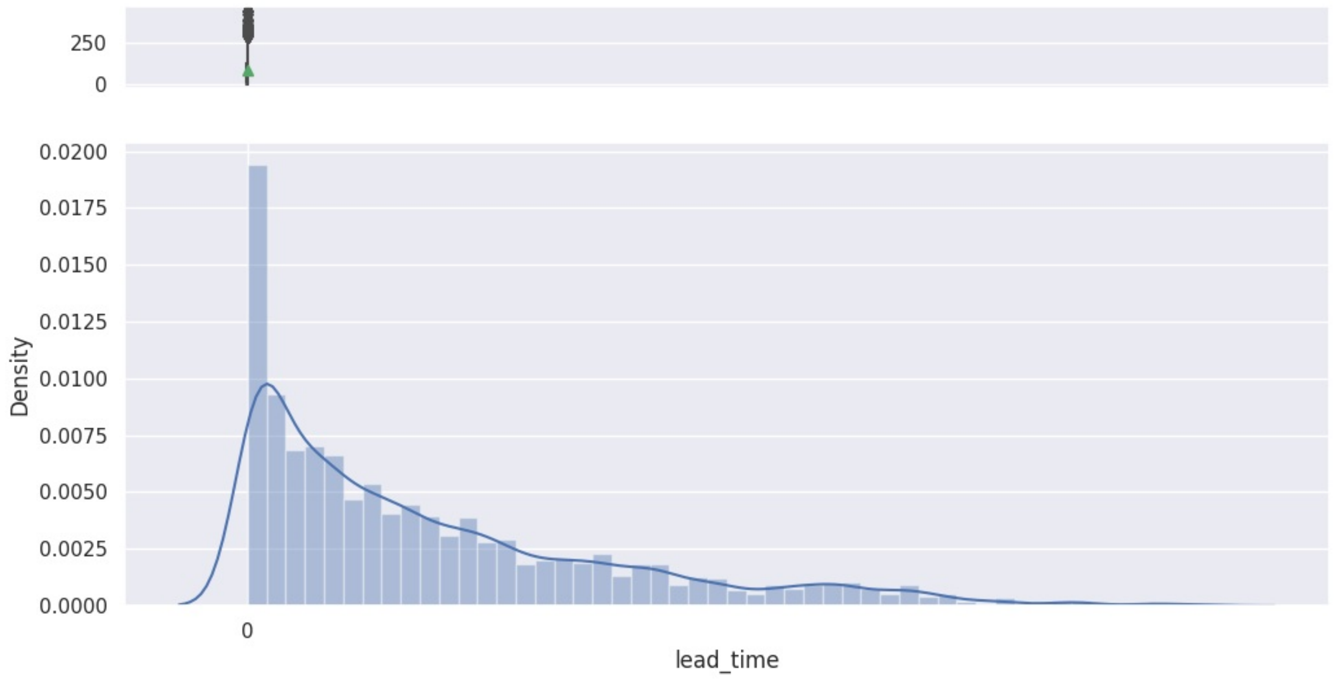
Let's explore these variables in some more depth by observing their distributions.

We will first define a **hist_box() function** that provides both a boxplot and a histogram in the same visual, with which we can perform univariate analysis on the columns of this dataset.

In [14]:
```python
# Defining the hist_box() function
def hist_box(data,col):
  f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={'height_ratios': (0.15, 0.85)}, figsize=(12,
  # Adding a graph in each part
  sns.boxplot(data[col], ax=ax_box, showmeans=True)
  sns.distplot(data[col], ax=ax_hist)
  plt.show()
```

### Question 2.1: Plot the histogram and box plot for the variable `Lead Time` using the hist_box function provided and write your insights. (1 Mark)

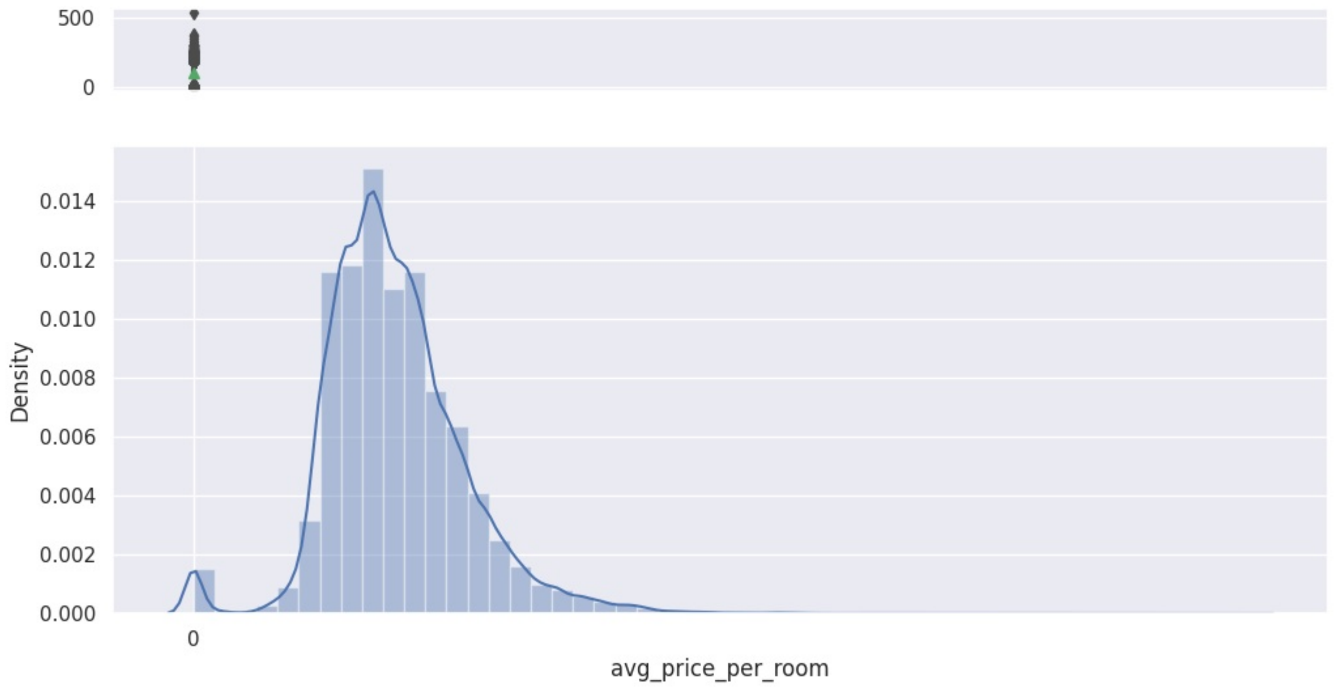
In [15]:
```python
# Remove _________ and complete the code
hist_box(data, col='lead_time')
```



**Write your answers here: is right-skewed**

### Question 2.2: Plot the histogram and box plot for the variable `Average Price per Room` using the hist_box function provided and write your insights. (1 Mark)

In [16]:
```
# Remove _________ and complete the code
hist_box(data, col='avg_price_per_room')
```



**Write your answers here:normal distribution**

**Interestingly some rooms have a price equal to 0. Let's check them.**

In [17]:
```
data[data["avg_price_per_room"] == 0]
```

Out[17]:

| | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | type_of_meal_plan | required_car_parking_space | room_type_re |
|---|---|---|---|---|---|---|---|
| **63** | 1 | 0 | 0 | 1 | Meal Plan 1 | 0 | Room_ |
| **145** | 1 | 0 | 0 | 2 | Meal Plan 1 | 0 | Room_ |
| **209** | 1 | 0 | 0 | 0 | Meal Plan 1 | 0 | Room_ |
| **266** | 1 | 0 | 0 | 2 | Meal Plan 1 | 0 | Room_ |
| **267** | 1 | 0 | 2 | 1 | Meal Plan 1 | 0 | Room_ |
| **...** | ... | ... | ... | ... | ... | ... | |
| **35983** | 1 | 0 | 0 | 1 | Meal Plan 1 | 0 | Room_ |
| **36080** | 1 | 0 | 1 | 1 | Meal Plan 1 | 0 | Room_ |
| **36114** | 1 | 0 | 0 | 1 | Meal Plan 1 | 0 | Room_ |
| **36217** | 2 | 0 | 2 | 1 | Meal Plan 1 | 0 | Room_ |
| **36250** | 1 | 0 | 0 | 2 | Meal Plan 2 | 0 | Room_ |

545 rows × 18 columns

- There are quite a few hotel rooms which have a price equal to 0.
- In the market segment column, it looks like many values are complementary.

In [18]:
```
data.loc[data["avg_price_per_room"] == 0, "market_segment_type"].value_counts()
```

Out[18]:
```
Complementary    354
Online           191
Name: market_segment_type, dtype: int64
```

- It makes sense that most values with room prices equal to 0 are the rooms given as complimentary service from the hotel.
- The rooms booked online must be a part of some promotional campaign done by the hotel.

In [19]:
```
# 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
```

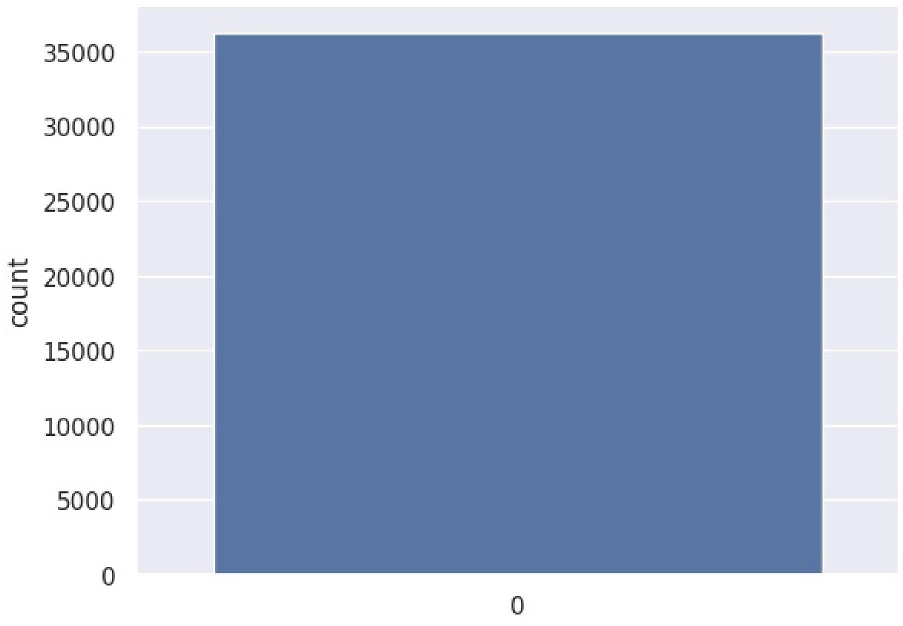
Out[19]:
```
179.55
```

In [20]:
```
# assigning the outliers the value of upper whisker
data.loc[data["avg_price_per_room"] >= 500, "avg_price_per_room"] = Upper_Whisker
```

## Let's understand the distribution of the categorical variables

### Number of Children

In [21]:
```
sns.countplot(data['no_of_children'])
plt.show()
```



```
In [22]: data['no_of_children'].value_counts(normalize=True)
```

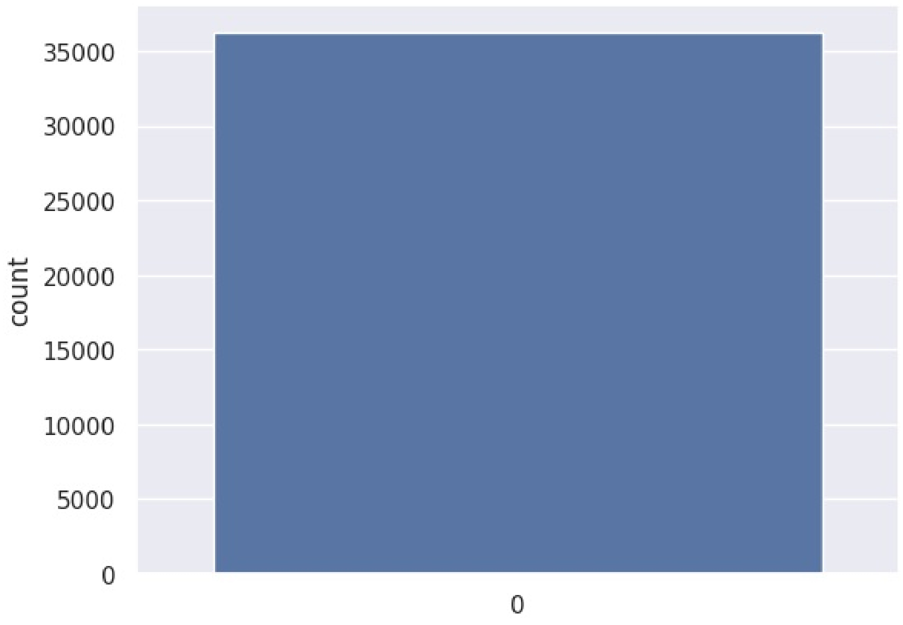
```
Out[22]: 0     0.925624
         1     0.044604
         2     0.029166
         3     0.000524
         9     0.000055
         10    0.000028
         Name: no_of_children, dtype: float64
```

- Customers were not travelling with children in 93% of cases.
- There are some values in the data where the number of children is 9 or 10, which is highly unlikely.
- We will replace these values with the maximum value of 3 children.

```
In [23]: # replacing 9, and 10 children with 3
         data["no_of_children"] = data["no_of_children"].replace([9, 10], 3)
```

**Arrival Month**

```
In [24]: sns.countplot(data["arrival_month"])
         plt.show()
```



```
In [25]: data['arrival_month'].value_counts(normalize=True)
```

```
Out[25]: 10    0.146575
9     0.127112
8     0.105114
6     0.088298
12    0.083280
11    0.082150
7     0.080496
4     0.075424
5     0.071620
3     0.065003
2     0.046975
1     0.027953
Name: arrival_month, dtype: float64
```

- October is the busiest month for hotel arrivals followed by September and August. **Over 35% of all bookings**, as we see in the above table, were for one of these three months.
- Around 14.7% of the bookings were made for an October arrival.

**Booking Status**

```
In [26]: data["booking_status"].head()
```

```
Out[26]: 0    Not_Canceled
1    Not_Canceled
2        Canceled
3        Canceled
4        Canceled
Name: booking_status, dtype: object
```

```
In [27]: data['booking_status'].value_counts(normalize=True)
```

```
Out[27]: Not_Canceled    0.672364
Canceled        0.327636
Name: booking_status, dtype: float64
```

- 32.8% of the bookings were canceled by the customers.

**Let's encode Canceled bookings to 1 and Not_Canceled as 0 for further analysis**

```
In [28]: data["booking_status"] = data["booking_status"].apply(
    lambda x: 1 if x == "Canceled" else 0
)
```

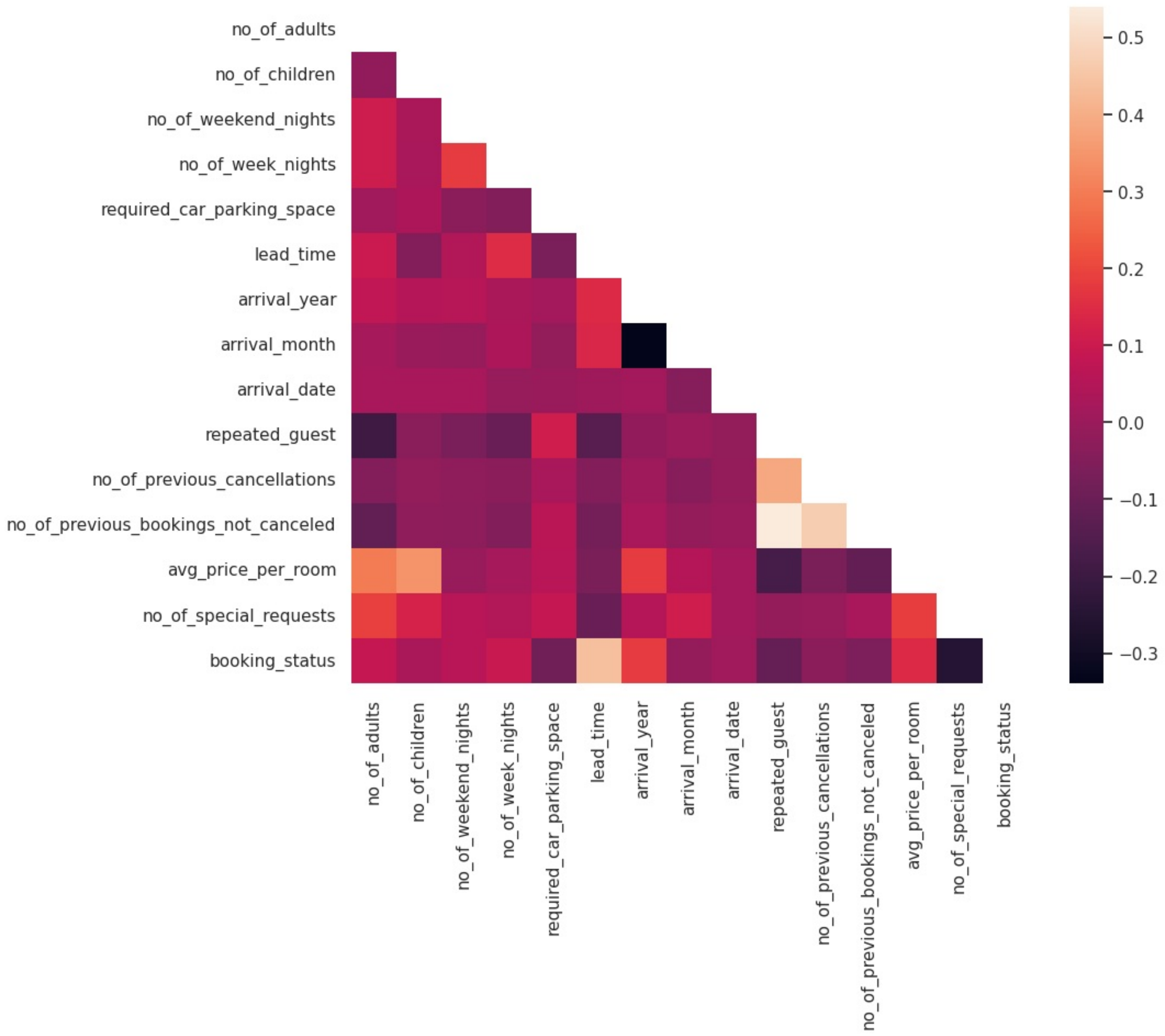
## Question 3: Bivariate Analysis

### Question 3.1: Find and visualize the correlation matrix using a heatmap and write your observations from the plot. (2 Marks)

```
In [104…: # Remove _________ and complete the code
cols_list = data.select_dtypes(include=np.number).columns.tolist()
plt.figure(figsize=(12, 7))
data1 = data.drop(["type_of_meal_plan","room_type_reserved","market_segment_type"], axis=1)
```

```
<Figure size 1200x700 with 0 Axes>
```

```
In [105…: corr = data1.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.heatmap(corr, mask=mask, square=True)
```



**Write your answers here:_**

a. The correlation between repeated guest and the number of previous bookings not canceled (0.5) suggests that guests who have made more bookings in the past are more likely to become repeated guests. This could indicate that these guests were satisfied with their previous stays, and hence chose to book again.

b. The correlation between repeated guest and the number of previous cancellations (0.3) suggests that guests who have canceled their bookings in the past are less likely to become repeated guests. This could indicate that these guests were dissatisfied with their previous stays, leading to cancellations and a lower likelihood of returning.

c. The correlation between the number of children and the number of adults (0.4) suggests that families with children tend to have more adults staying with them. This makes sense as adults may accompany children during family vacations.

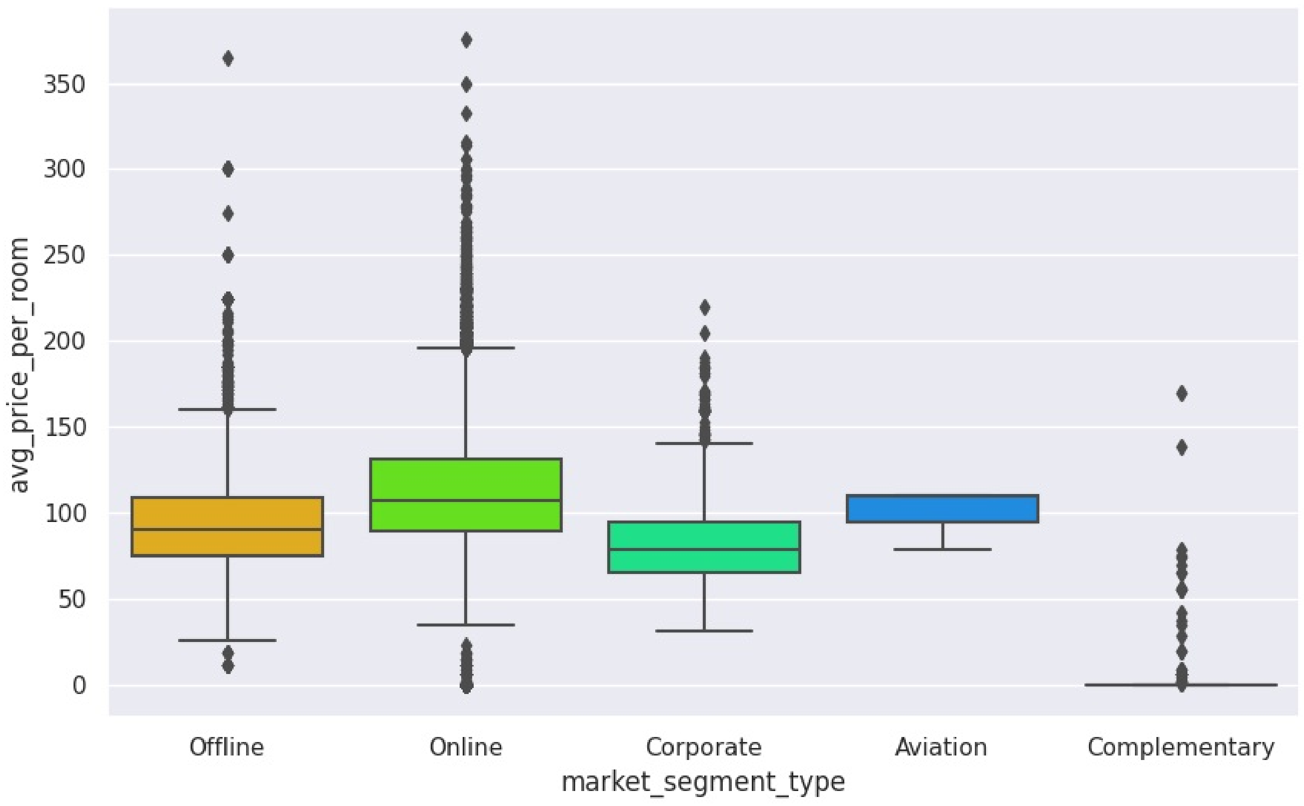
d. The correlation between lead time and booking status (0.5) suggests that longer lead times (i.e., more time between booking and arrival) are associated with a higher likelihood of the booking being canceled. This could be due to guests making tentative bookings but later deciding to cancel or change their plans.

e. The correlation between arrival year and average price per room (0.2) suggests that there is a slight upward trend in the average price per room over time. This could indicate a general increase in demand for hotel rooms or an increase in the quality of the hotel over time.

f. The correlation between the number of previous cancellations and the number of previous bookings not canceled (0.4) suggests that guests who have canceled more bookings in the past have also made more bookings that were not canceled. This could indicate that these guests have a history of changing their plans frequently or that they have a higher volume of bookings in general.

**Hotel rates are dynamic and change according to demand and customer demographics. Let's see how prices vary across different market segments**

In [32]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=data, x="market_segment_type", y="avg_price_per_room", palette="gist_rainbow"
)
plt.show()
```



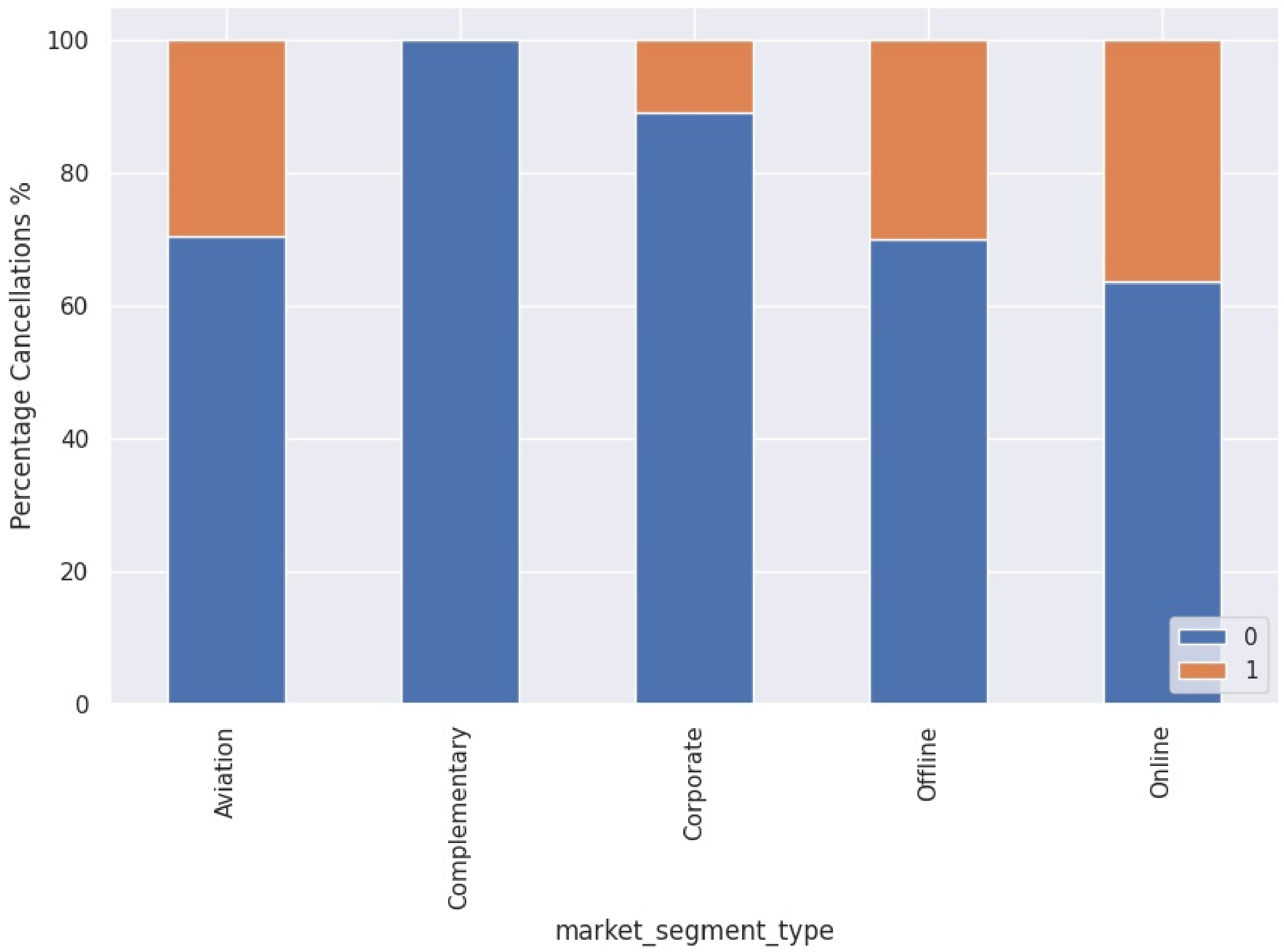
- Rooms booked online have high variations in prices.
- The offline and corporate room prices are almost similar.
- Complementary market segment gets the rooms at very low prices, which makes sense.

We will define a **stacked barplot()** function to help analyse how the target variable varies across predictor categories.

In [42]:
```
# Defining the stacked_barplot() function
def stacked_barplot(data,predictor,target,figsize=(10,6)):
  (pd.crosstab(data[predictor],data[target],normalize='index')*100).plot(kind='bar',figsize=figsize,stacked=Tru
  plt.legend(loc="lower right")
  plt.ylabel('Percentage Cancellations %')
```

### Question 3.2: Plot the stacked barplot for the variable `Market Segment Type` against the target variable `Booking Status` using the stacked_barplot function provided and write your insights. (1 Mark)

In [43]:
```
# Remove _________ and complete the code
stacked_barplot(data, "market_segment_type" ,"booking_status")
```



**Write your answers here:**

The Aviation market segment has the highest percentage of cancellations, with 70% of bookings being canceled and only 30% of bookings being confirmed.

The Complementary market segment has the lowest percentage of cancellations, with 100% of bookings being confirmed and 0% of bookings being canceled.

The Corporate market segment has a low percentage of cancellations, with only 5% of bookings being canceled and 95% of bookings being confirmed.

The Offline and Online market segments have a relatively high percentage of cancellations, with 30% and 40% of bookings being canceled respectively, and the rest being confirmed.

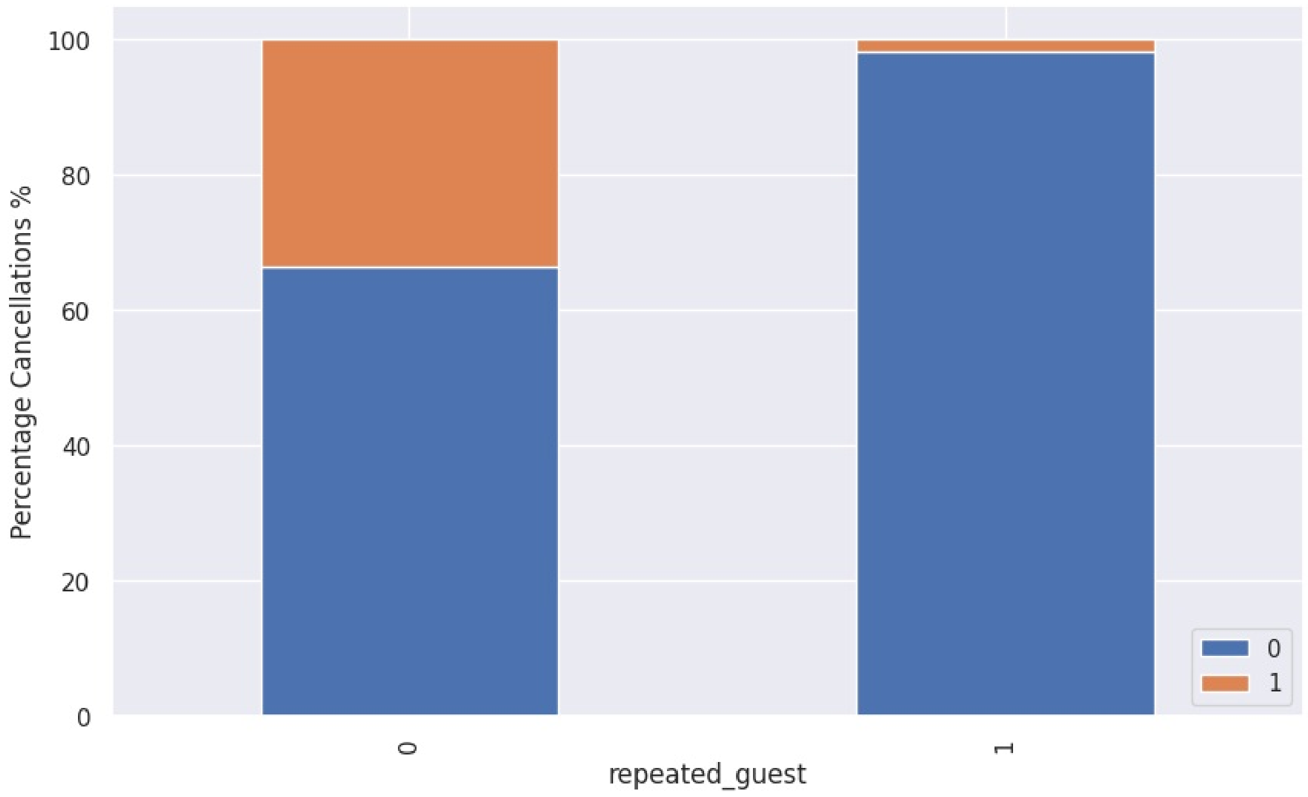
The cancellation rate for the Aviation and Online market segments is higher than that of other market segments, indicating that these segments may have a higher risk of cancellations.

This can help to improve marketing strategies and business policies.

### Question 3.3: Plot the stacked barplot for the variable `Repeated Guest` against the target variable `Booking Status` using the stacked_barplot function provided and write your insights. (1 Mark)

Repeating guests are the guests who stay in the hotel often and are important to brand equity.

In [44]:
```
# Remove _________ and complete the code
stacked_barplot(data, "repeated_guest","booking_status")
```



**Write your answers here:**

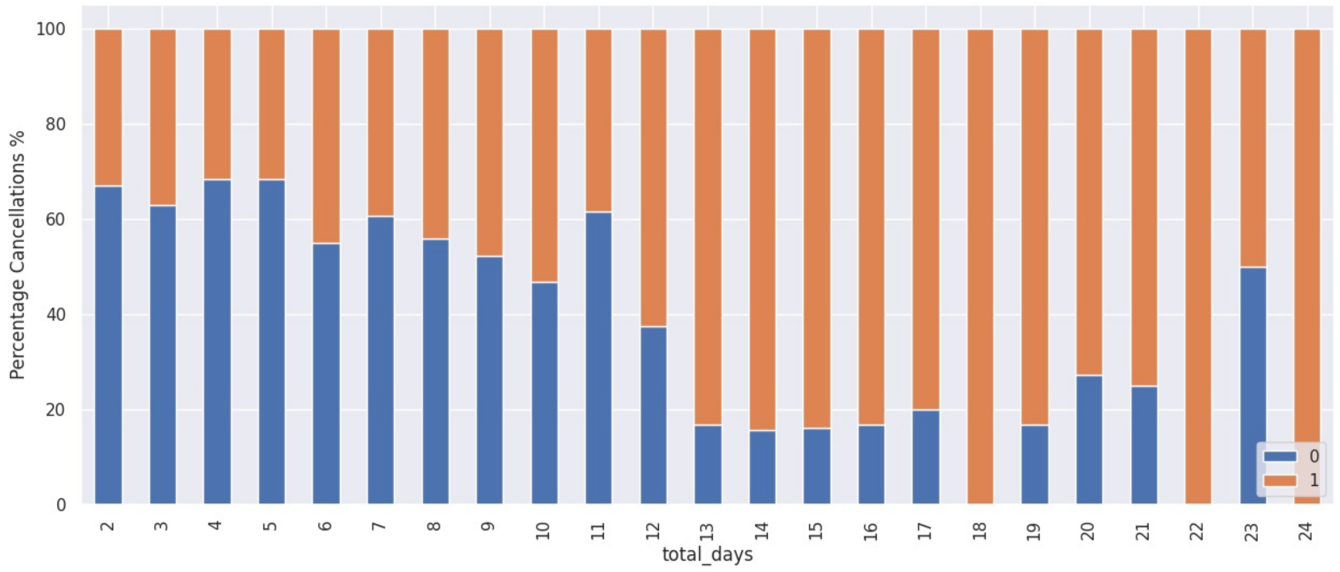
The percentage of cancellations is higher for non-repeated guests (35%) compared to repeated guests (1%).

This suggests that the hotel has a higher chance of retaining its repeated guests as they have a lower tendency to cancel their bookings.

It is important for the hotel to focus on building and maintaining a loyal customer base of repeated guests in order to improve their business performance.

**Let's analyze the customer who stayed for at least a day at the hotel.**

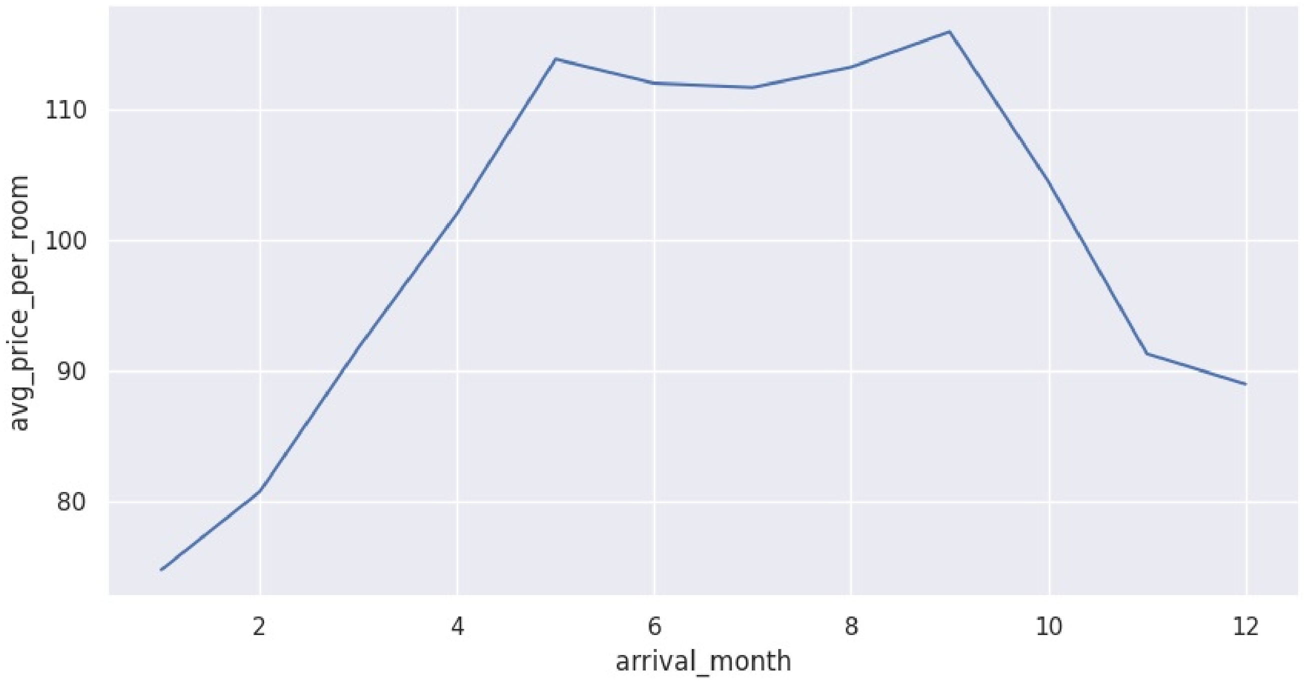
In [45]:
```
stay_data = data[(data["no_of_week_nights"] > 0) & (data["no_of_weekend_nights"] > 0)]
stay_data["total_days"] = (stay_data["no_of_week_nights"] + stay_data["no_of_weekend_nights"])

stacked_barplot(stay_data, "total_days", "booking_status",figsize=(15,6))
```



- The general trend is that the chances of cancellation increase as the number of days the customer planned to stay at the hotel increases.

**As hotel room prices are dynamic, Let's see how the prices vary across different months**

In [46]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(y=data["avg_price_per_room"], x=data["arrival_month"], ci=None)
plt.show()
```



- The price of rooms is highest in May to September - around 115 euros per room.

## Data Preparation for Modeling

- We want to predict which bookings will be canceled.
- Before we proceed to build a model, we'll have to encode categorical features.
- We'll split the data into train and test to be able to evaluate the model that we build on the train data.

**Separating the independent variables (X) and the dependent variable (Y)**

In [47]:
```
X = data.drop(["booking_status"], axis=1)
Y = data["booking_status"]

X = pd.get_dummies(X, drop_first=True) # Encoding the Categorical features
```

**Splitting the data into a 70% train and 30% test set**

Some classification problems can exhibit a large imbalance in the distribution of the target classes: for instance there could be several times more negative samples than positive samples. In such cases it is recommended to use the **stratified sampling** technique to ensure that relative class frequencies are approximately preserved in each train and validation fold.

In [48]:
```
# Splitting data in train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30,stratify=Y, random_state=1)
```

In [49]:
```
print("Shape of Training set : ", X_train.shape)
print("Shape of test set : ", X_test.shape)
print("Percentage of classes in training set:")
print(y_train.value_counts(normalize=True))
print("Percentage of classes in test set:")
print(y_test.value_counts(normalize=True))
```

```
Shape of Training set :  (25392, 27)
Shape of test set :  (10883, 27)
Percentage of classes in training set:
0    0.672377
1    0.327623
Name: booking_status, dtype: float64
Percentage of classes in test set:
0    0.672333
1    0.327667
Name: booking_status, dtype: float64
```

## Model Evaluation Criterion

### Model can make wrong predictions as:

1. Predicting a customer will not cancel their booking but in reality, the customer will cancel their booking.
2. Predicting a customer will cancel their booking but in reality, the customer will 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 distribution channels.
- If we predict that a booking will get canceled and the booking doesn't get canceled the hotel might not be able to provide satisfactory services to the customer by assuming that this booking will be canceled. This might damage brand equity.

### How to reduce the losses?

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

**Also, let's 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 [50]:
```
# Creating metric function
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 Cancelled', 'Cancelled'], yticklabels=['Not Cance
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
```

## Building the model

We will be building 4 different models:

- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Decision Tree**
- **Random Forest**

### Question 4: Logistic Regression (6 Marks)

#### Question 4.1: Build a Logistic Regression model (Use the sklearn library) (1 Mark)

In [51]:
```
# Remove _________ and complete the code

# Fitting logistic regression model
lg=LogisticRegression(random_state=42, class_weight='balanced')
lg.fit(X_train, y_train)
```

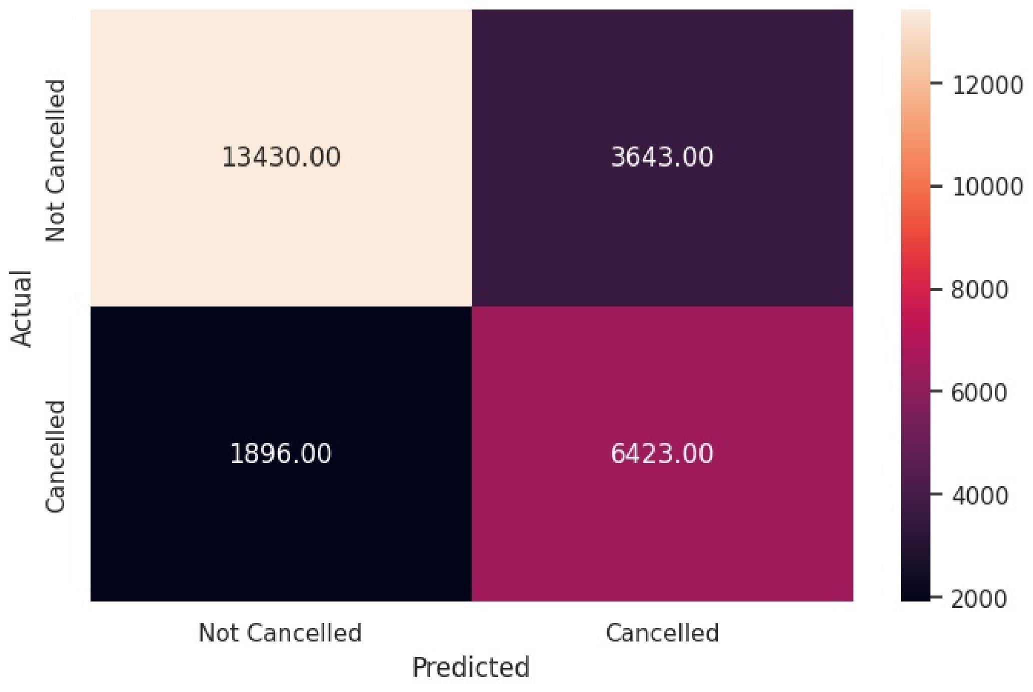
Out[51]:
```
▾                       LogisticRegression
LogisticRegression(class_weight='balanced', random_state=42)
```

In [52]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
```

#### Question 4.2: Check the performance of the model on train and test data (2 Marks)

In [53]:
```
# Remove _________ and complete the code

# Checking the performance on the training data
y_pred_train = lg.predict(X_train)
metrics_score(y_train, y_pred_train)
```

```
              precision    recall  f1-score   support

           0       0.88      0.79      0.83     17073
           1       0.64      0.77      0.70      8319

    accuracy                           0.78     25392
   macro avg       0.76      0.78      0.76     25392
weighted avg       0.80      0.78      0.79     25392
```



**Write your Answer here:**

The precision and recall scores suggest that the model is better at predicting the negative class (0) than the positive class (1).

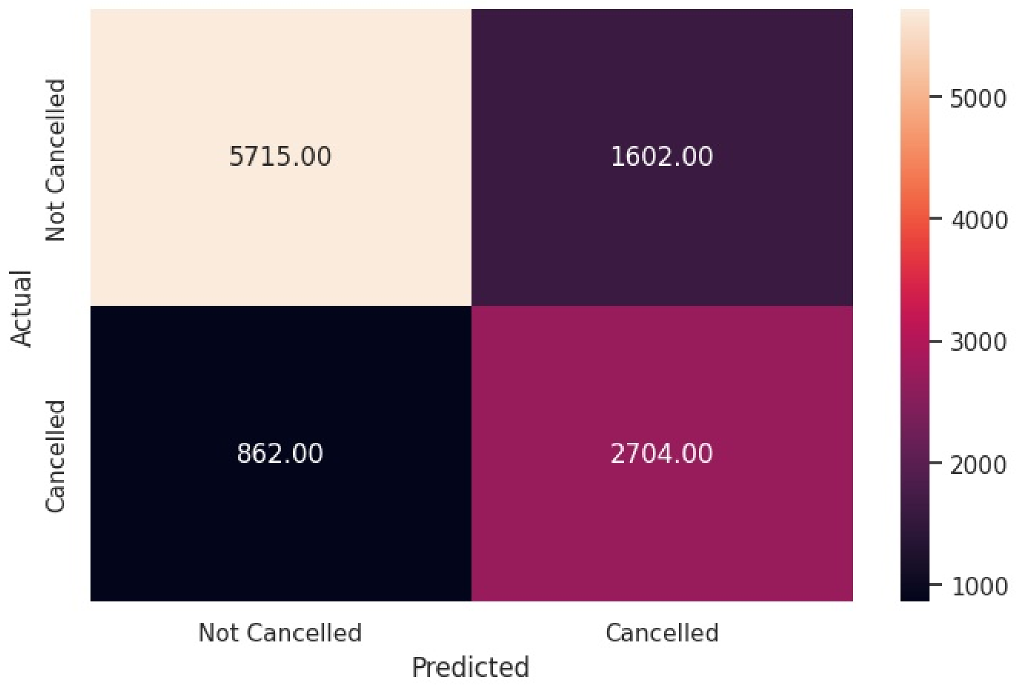
The overall accuracy of the model is decent but could be improved, especially for the positive class.

Improving the model's performance could lead to better predictions of bookings that are likely to be canceled, which could help the business to take appropriate actions to mitigate the impact of cancellations and improve overall performance.

Let's check the performance on the test set

In [54]:
```
# Remove _________ and complete the code

# Checking the performance on the test dataset
y_pred_test = lg.predict(X_test)
metrics_score(y_test, y_pred_test)
```

```
              precision    recall  f1-score   support

           0       0.87      0.78      0.82      7317
           1       0.63      0.76      0.69      3566

    accuracy                           0.77     10883
   macro avg       0.75      0.77      0.75     10883
weighted avg       0.79      0.77      0.78     10883
```



**Write your Answer here:**

The second model performed similarly to the first model in terms of precision, recall, and F1-score.

However, the accuracy of the second model is slightly lower than the first model.

The performance of the second model on the test set suggests that it is not overfitting, but it may need further optimization to improve its accuracy on the test set.

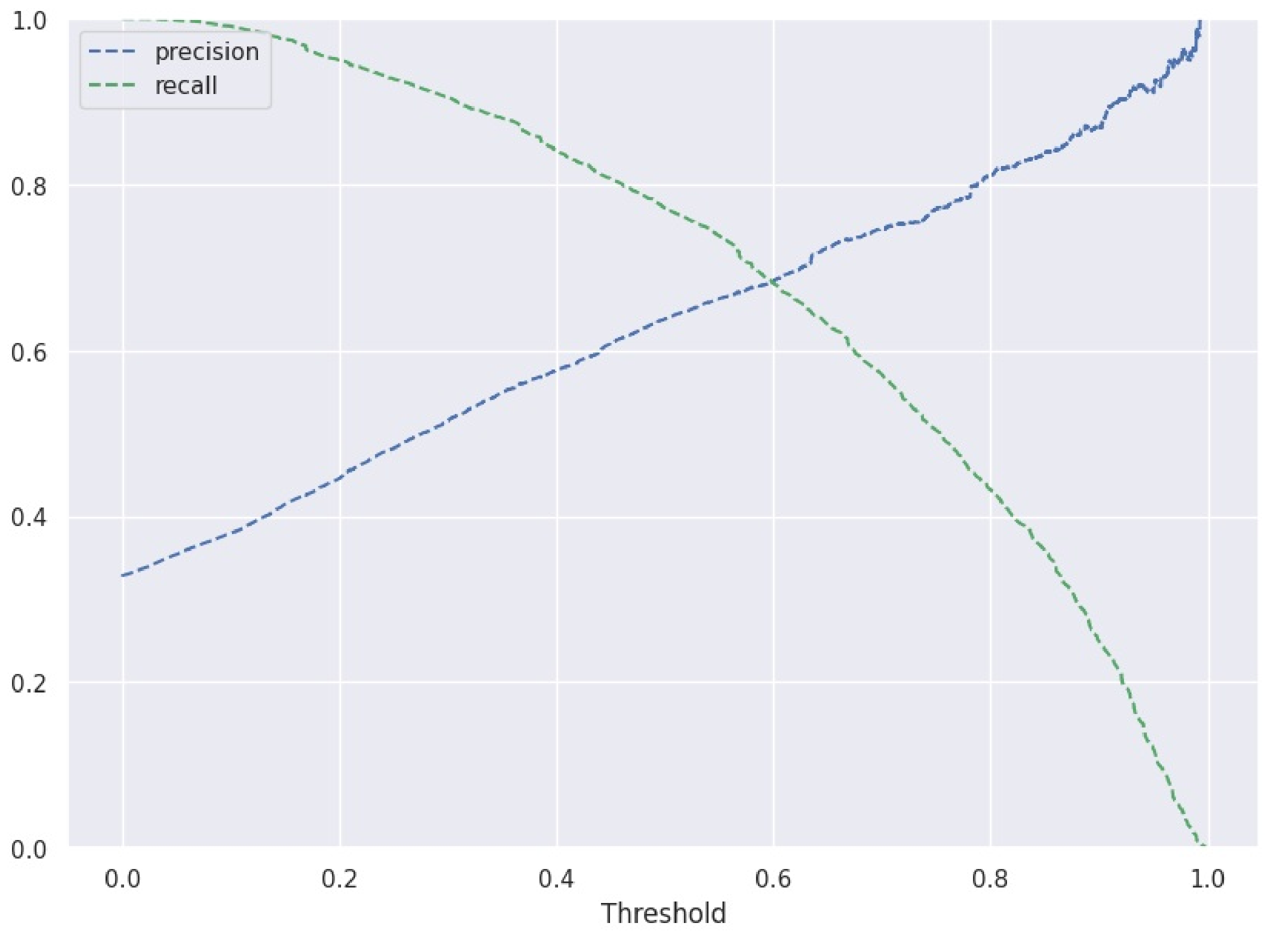
Additionally, more data may be required to improve the overall performance of the model.

### Question 4.3: Find the optimal threshold for the model using the Precision-Recall Curve. (1 Mark)

Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

Let's use the Precision-Recall curve and see if we can find a **better threshold.**

In [55]:
```
# Remove _________ and complete the code

# Predict_proba gives the probability of each observation belonging to each class
y_scores_lg=lg.predict_proba(X_train)

precisions_lg, recalls_lg, thresholds_lg = precision_recall_curve(y_train, y_scores_lg[:,1])

# Plot values of precisions, recalls, and thresholds
plt.figure(figsize=(10,7))
plt.plot(thresholds_lg, precisions_lg[:-1], 'b--', label='precision')
plt.plot(thresholds_lg, recalls_lg[:-1], 'g--', label = 'recall')
plt.xlabel('Threshold')
plt.legend(loc='upper left')
plt.ylim([0,1])
plt.show()
```



**Write your answers here:**

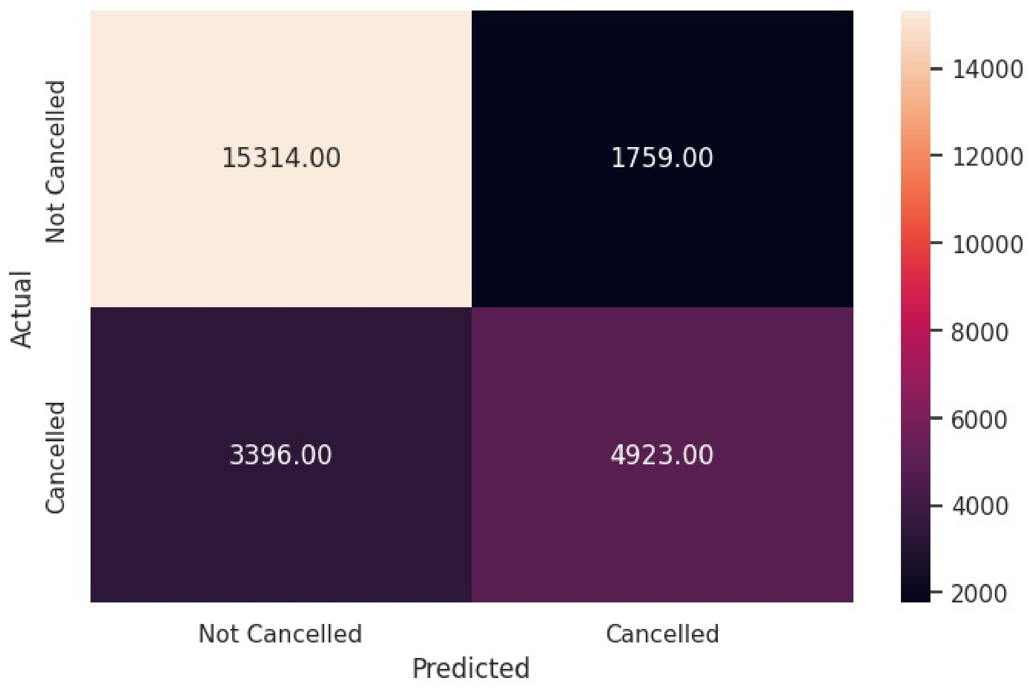
The optimal threshold for the model is 0.68.

```
In [56]: # Setting the optimal threshold
optimal_threshold = .68
```

### Question 4.4: Check the performance of the model on train and test data using the optimal threshold. (2 Marks)

```
In [57]: # Remove _________ and complete the code

# Creating confusion matrix
y_pred_train = lg.predict_proba(X_train)
metrics_score(y_train, y_pred_train[:,1]>optimal_threshold)
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86     17073
           1       0.74      0.59      0.66      8319

    accuracy                           0.80     25392
   macro avg       0.78      0.74      0.76     25392
weighted avg       0.79      0.80      0.79     25392
```



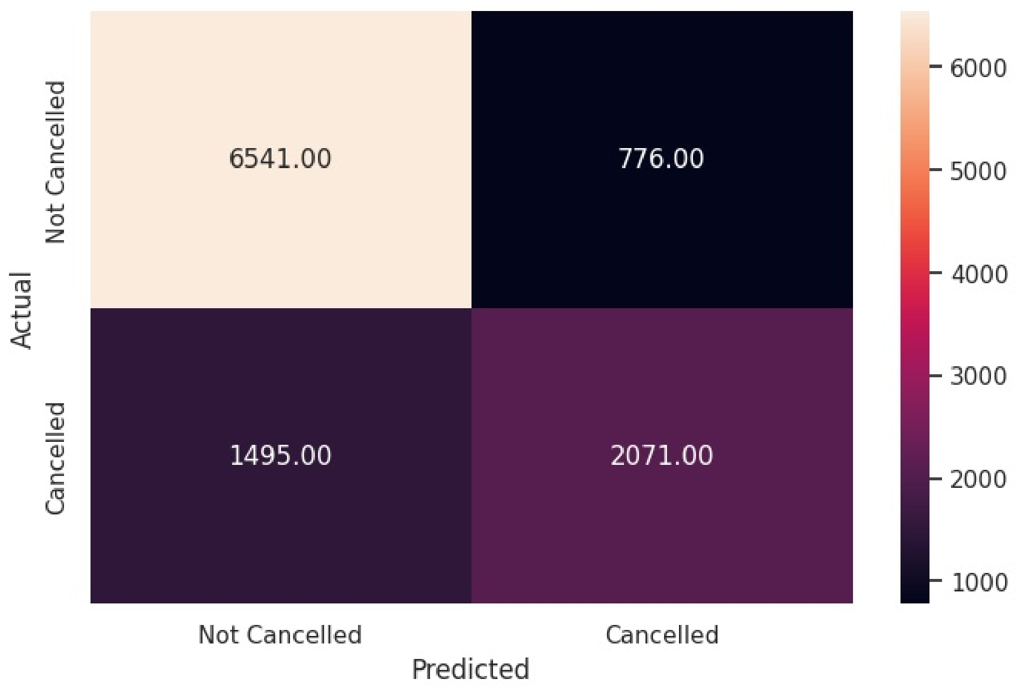
**Write your answers here:**

The new model has a higher precision and f1-score for class 1 (canceled bookings) but lower recall, and a lower accuracy compared to the second model. However, it has a higher recall for class 0 (not canceled bookings).

Let's check the performance on the test set

In [58]:
```
# Remove _________ and complete the code

y_pred_test = lg.predict_proba(X_test)
metrics_score(y_test, y_pred_test[:,1]>optimal_threshold)
```

```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85      7317
           1       0.73      0.58      0.65      3566

    accuracy                           0.79     10883
   macro avg       0.77      0.74      0.75     10883
weighted avg       0.79      0.79      0.78     10883
```



**Write your answers here:**

The performance on the test set did not improve much and remained similar to the second model.

This suggests that the model may not be overfitting but still needs improvement to better generalize to unseen data.

## Question 5: Support Vector Machines (11 Marks)

To accelerate SVM training, let's scale the data for support vector machines.

In [59]:
```python
scaling = MinMaxScaler(feature_range=(-1,1)).fit(X_train)
X_train_scaled = scaling.transform(X_train)
X_test_scaled = scaling.transform(X_test)
```

Let's build the models using the two of the widely used kernel functions:

1. **Linear Kernel**
2. **RBF Kernel**

### Question 5.1: Build a Support Vector Machine model using a linear kernel (1 Mark)

**Note: Please use the scaled data for modeling Support Vector Machine**

In [60]:
```python
# Remove _________ and complete the code

model = SVC(C = 100, kernel = 'linear', random_state=123, probability=True, gamma='auto')
model.fit(X_train_scaled,y_train)
```

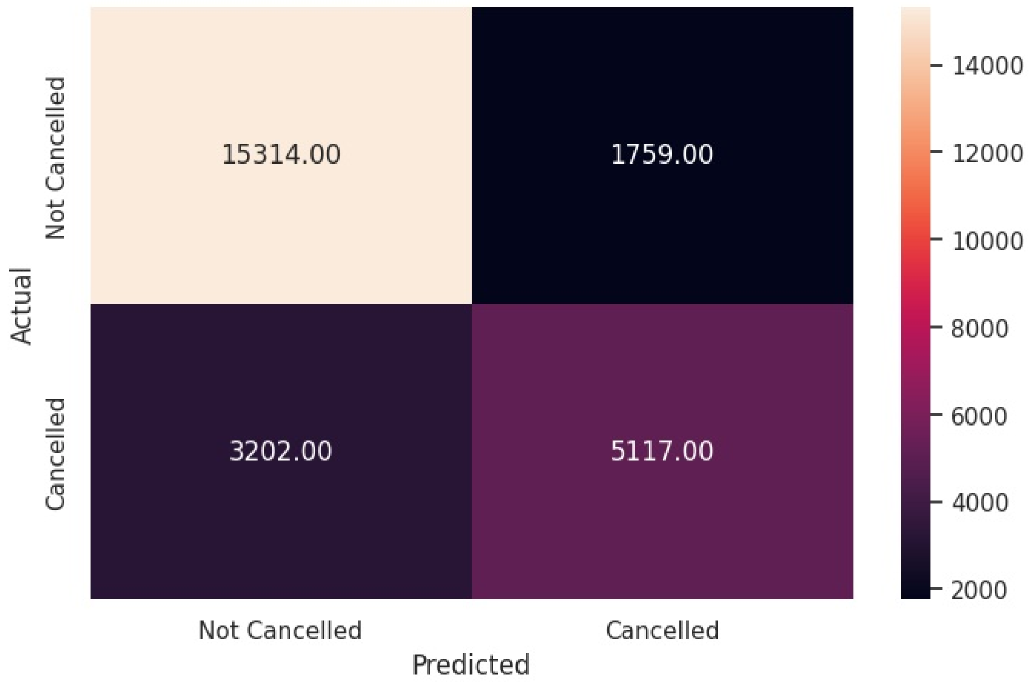
In [61]:
```

```

```
Out[61]: ▾                              SVC
         SVC(C=100, gamma='auto', kernel='linear', probability=True, random_state=123)
```

### Question 5.2: Check the performance of the model on train and test data (2 Marks)

```
In [62]: # Remove _________ and complete the code

         y_pred_train_svm = model.predict(X_train_scaled)
         metrics_score(y_train, y_pred_train_svm)
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86     17073
           1       0.74      0.62      0.67      8319

    accuracy                           0.80     25392
   macro avg       0.79      0.76      0.77     25392
weighted avg       0.80      0.80      0.80     25392
```



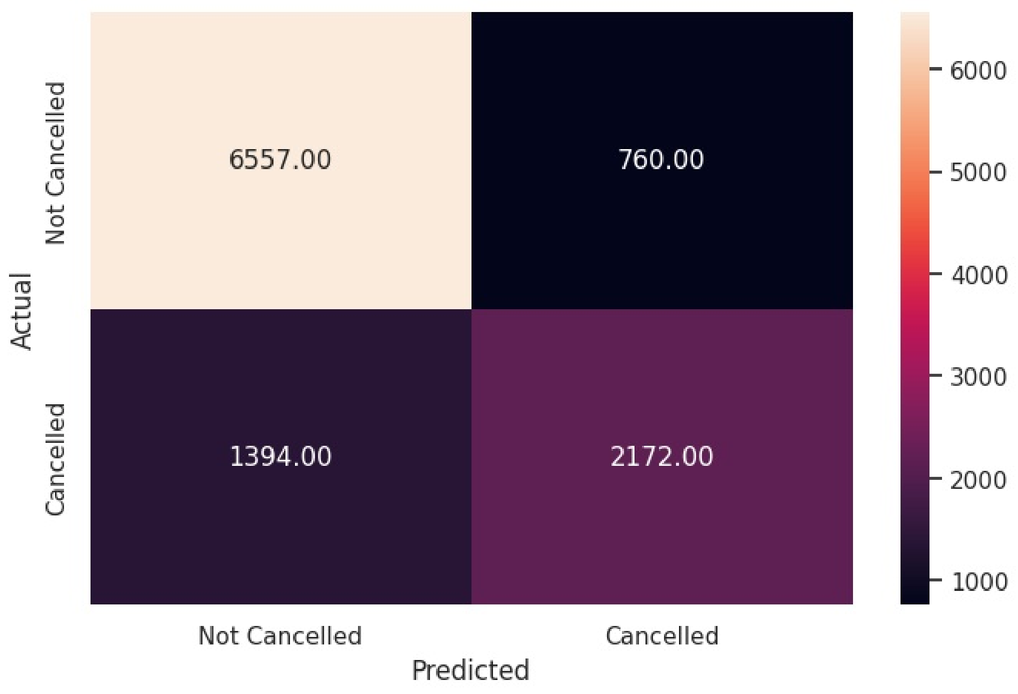
**Write your answers here:**

Support Vector Machine model with a linear kernel is performing better than the previous models we have evaluated. The model achieved an accuracy of 0.80 on the training set, which is higher than the accuracy achieved by the Logistic Regression models we evaluated earlier. Additionally, the precision, recall, and F1-score for both classes are also higher for the SVM model.

Checking model performance on test set

```
In [63]: # Remove _________ and complete the code

         y_pred_test_svm = model.predict(X_test_scaled)
         metrics_score(y_test, y_pred_test_svm)
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86      7317
           1       0.74      0.61      0.67      3566

    accuracy                           0.80     10883
   macro avg       0.78      0.75      0.76     10883
weighted avg       0.80      0.80      0.80     10883
```

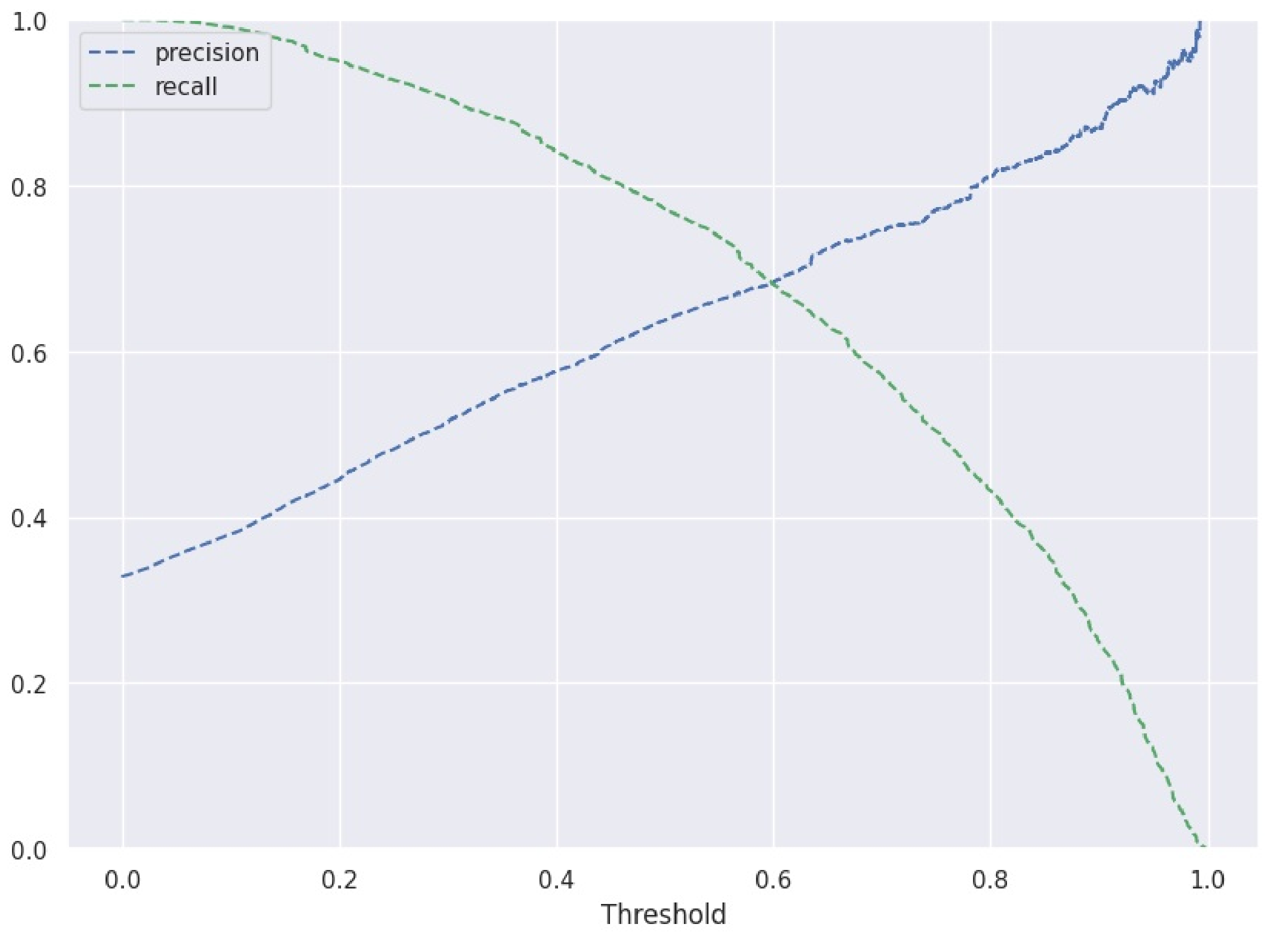


**Write your answers here:**

The SVM model with a linear kernel seems to perform slightly better on the test set compared to the previous models, with an accuracy of 0.80 and a weighted F1-score of 0.80. However, there is still some room for improvement in predicting the minority class of cancelled bookings (class 1), with a lower precision, recall, and F1-score compared to the majority class of not cancelled bookings (class 0).

### Question 5.3: Find the optimal threshold for the model using the Precision-Recall Curve. (1 Mark)

In [64]:
```python
# Remove _________ and complete the code

# Predict on train data
y_scores_svm=model.predict_proba(X_train_scaled)

precisions_svm, recalls_svm, thresholds_svm = precision_recall_curve(y_train, y_scores_lg[:,1])


# Plot values of precisions, recalls, and thresholds
plt.figure(figsize=(10,7))
plt.plot(thresholds_svm, precisions_svm[:-1], 'b--', label='precision')
plt.plot(thresholds_svm, recalls_svm[:-1], 'g--', label = 'recall')
plt.xlabel('Threshold')
plt.legend(loc='upper left')
plt.ylim([0,1])
plt.show()
```



**Write your answers here:**

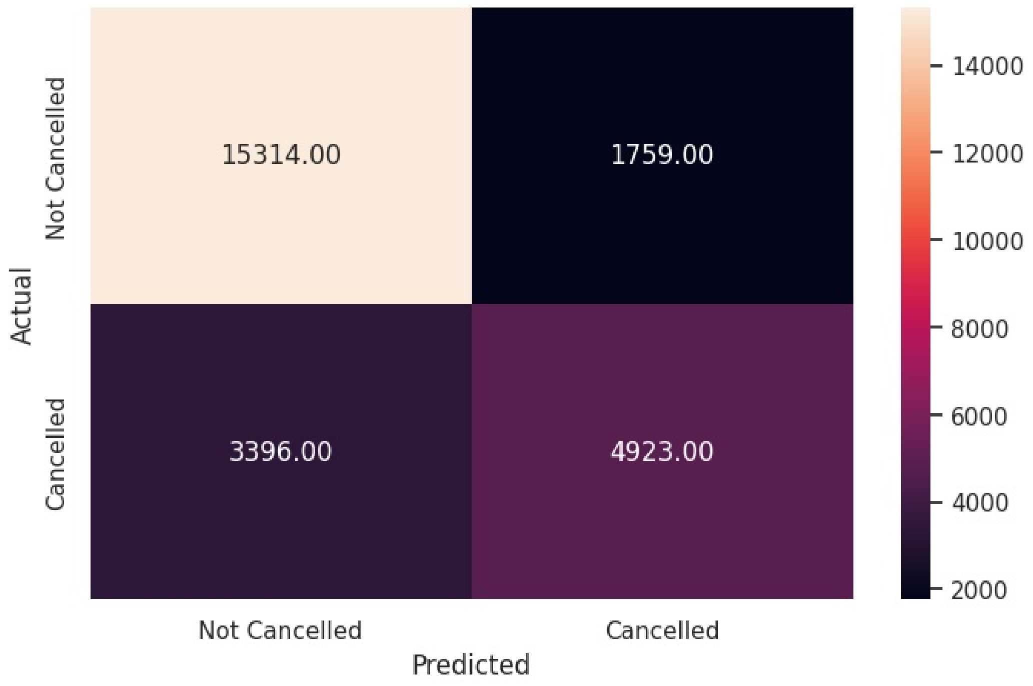
The optimal threshold for the model is also 0.68.

```
In [65]: optimal_threshold_svm=.68
```

### Question 5.4: Check the performance of the model on train and test data using the optimal threshold. (2 Marks)

```
In [66]: # Remove _________ and complete the code

         y_pred_train_svm = model.predict_proba(X_train_scaled)
         metrics_score(y_train, y_pred_train[:,1]>optimal_threshold)
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86     17073
           1       0.74      0.59      0.66      8319

    accuracy                           0.80     25392
   macro avg       0.78      0.74      0.76     25392
weighted avg       0.79      0.80      0.79     25392
```

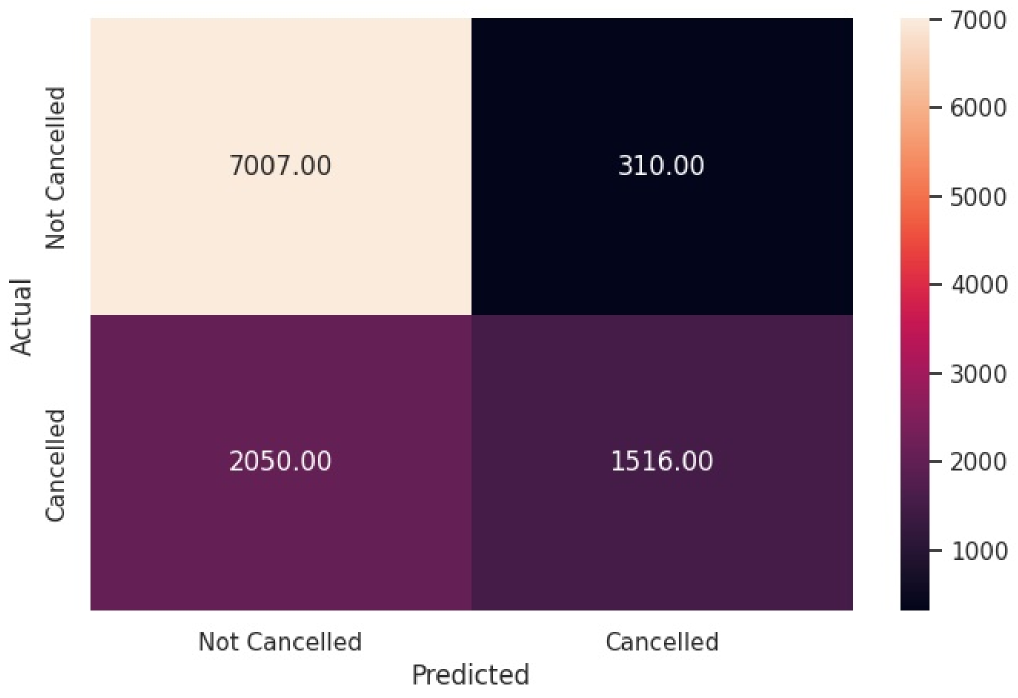


**Write your answers here:**

This shows the performance of the model on the test data using the optimal threshold. We can see that the precision, recall, and F1-score are similar to the results we got earlier using the default threshold of 0.5. However, we might prefer this new threshold of 0.68 if we want to prioritize precision over recall, as it gives a higher precision but lower recall compared to the default threshold.

In [67]:
```
# Remove _________ and complete the code

y_pred_test = model.predict_proba(X_test_scaled)
metrics_score(y_test, y_pred_test[:,1]>optimal_threshold)
```

```
              precision    recall  f1-score   support

           0       0.77      0.96      0.86      7317
           1       0.83      0.43      0.56      3566

    accuracy                           0.78     10883
   macro avg       0.80      0.69      0.71     10883
weighted avg       0.79      0.78      0.76     10883
```



**Write your answers here:**

This suggests that using the optimal threshold may improve the model's ability to identify cancelled bookings correctly (higher precision), but at the cost of missing some cancelled bookings (lower recall).

### Question 5.5: Build a Support Vector Machines model using an RBF kernel (1 Mark)

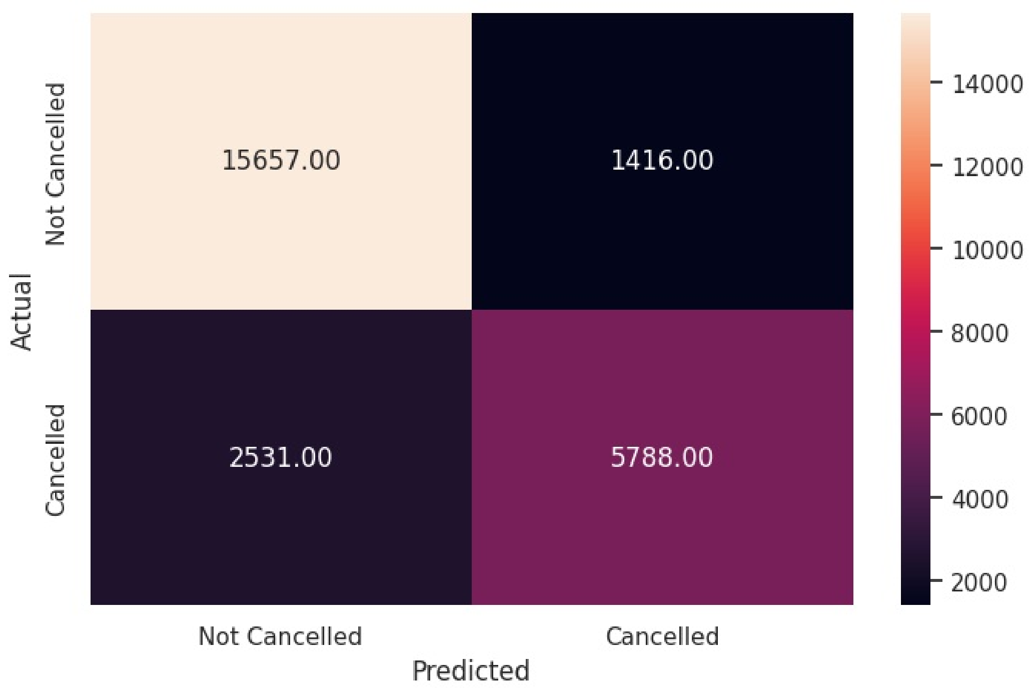
In [72]:
```
# Remove _________ and complete the code

svm_rbf = SVC(C = 100, kernel = 'rbf', random_state=123, probability=True, gamma='auto')
svm_rbf.fit(X_train_scaled,y_train)
```

Out[72]:
```
▾                                   SVC
SVC(C=100, gamma='auto', probability=True, random_state=123)
```

### Question 5.6: Check the performance of the model on train and test data (2 Marks)

In [73]:
```
# Remove _________ and complete the code

y_pred_train_svm = svm_rbf.predict(X_train_scaled)
metrics_score(y_train, y_pred_train_svm)
```

```
              precision    recall  f1-score   support

           0       0.86      0.92      0.89     17073
           1       0.80      0.70      0.75      8319

    accuracy                           0.84     25392
   macro avg       0.83      0.81      0.82     25392
weighted avg       0.84      0.84      0.84     25392
```



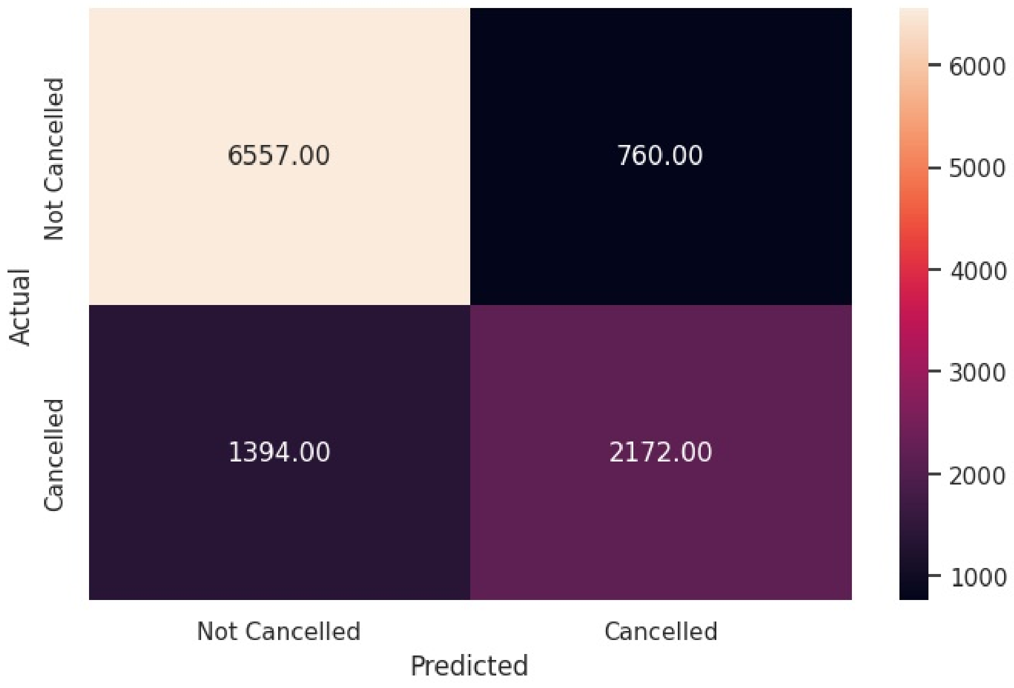
**Write your answers here:**

The SVM model with an RBF kernel has performed better than the logistic regression model. It has an overall accuracy of 84% on the training data and has improved precision, recall, and F1-score for both the "Cancelled" and "Not Cancelled" classes compared to the logistic regression model.

### Checking model performance on test set

```
In [74]: # Remove _________ and complete the code

y_pred_test = svm_rbf.predict(X_test_scaled)
metrics_score(y_test, y_pred_test_svm)
```

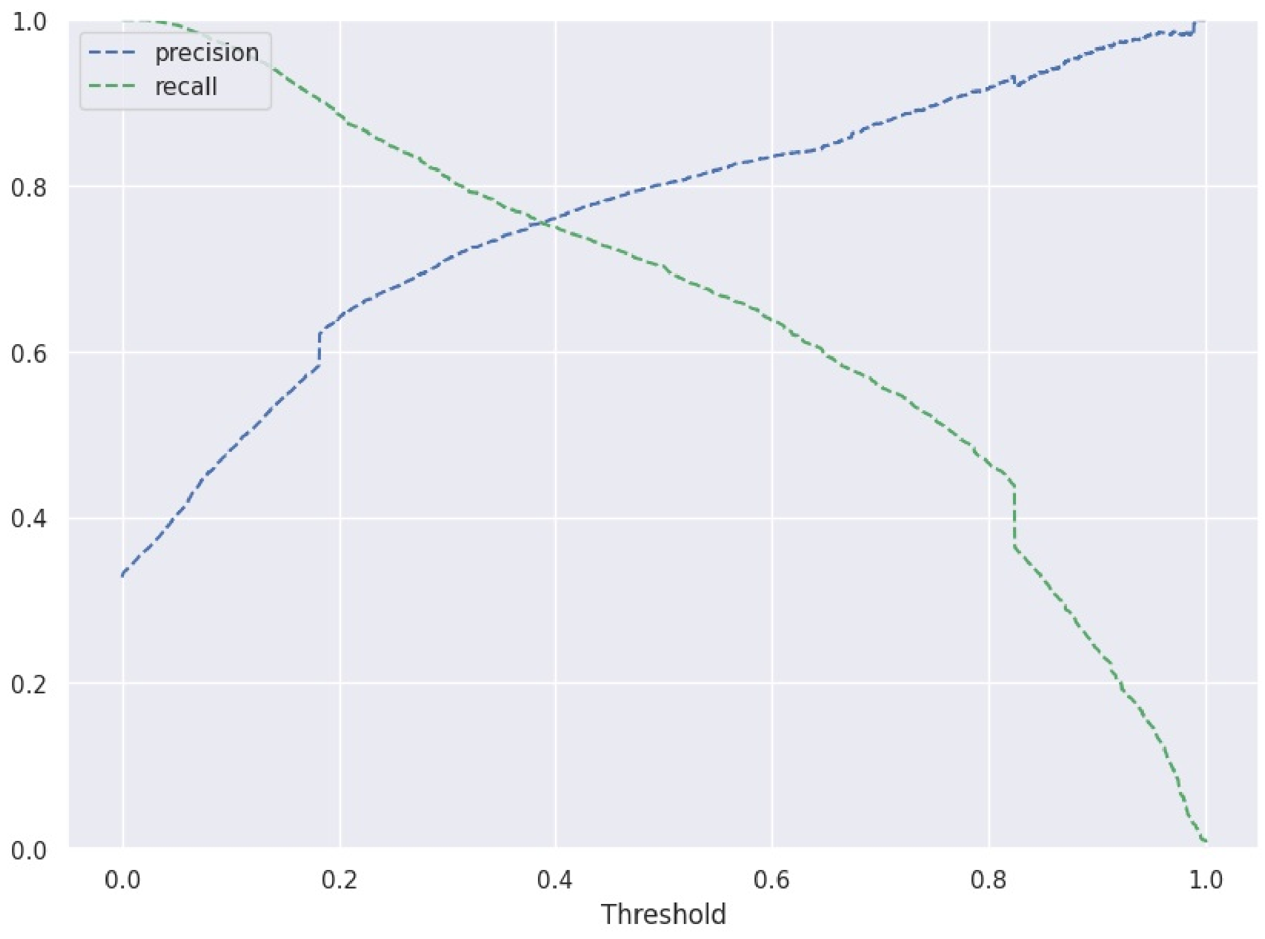
```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86      7317
           1       0.74      0.61      0.67      3566

    accuracy                           0.80     10883
   macro avg       0.78      0.75      0.76     10883
weighted avg       0.80      0.80      0.80     10883
```



**Write your answers here:**

Also perfomed better on test set. This is likely because the RBF kernel is more flexible and can model more complex decision boundaries than the linear kernel used in logistic regression. Additionally, SVMs are less sensitive to overfitting than logistic regression models and can better handle datasets with many features.

In [75]:
```
# Predict on train data
y_scores_svm=svm_rbf.predict_proba(X_train_scaled)

precisions_svm, recalls_svm, thresholds_svm = precision_recall_curve(y_train, y_scores_svm[:,1])

# Plot values of precisions, recalls, and thresholds
plt.figure(figsize=(10,7))
plt.plot(thresholds_svm, precisions_svm[:-1], 'b--', label='precision')
plt.plot(thresholds_svm, recalls_svm[:-1], 'g--', label = 'recall')
plt.xlabel('Threshold')
plt.legend(loc='upper left')
plt.ylim([0,1])
plt.show()
```

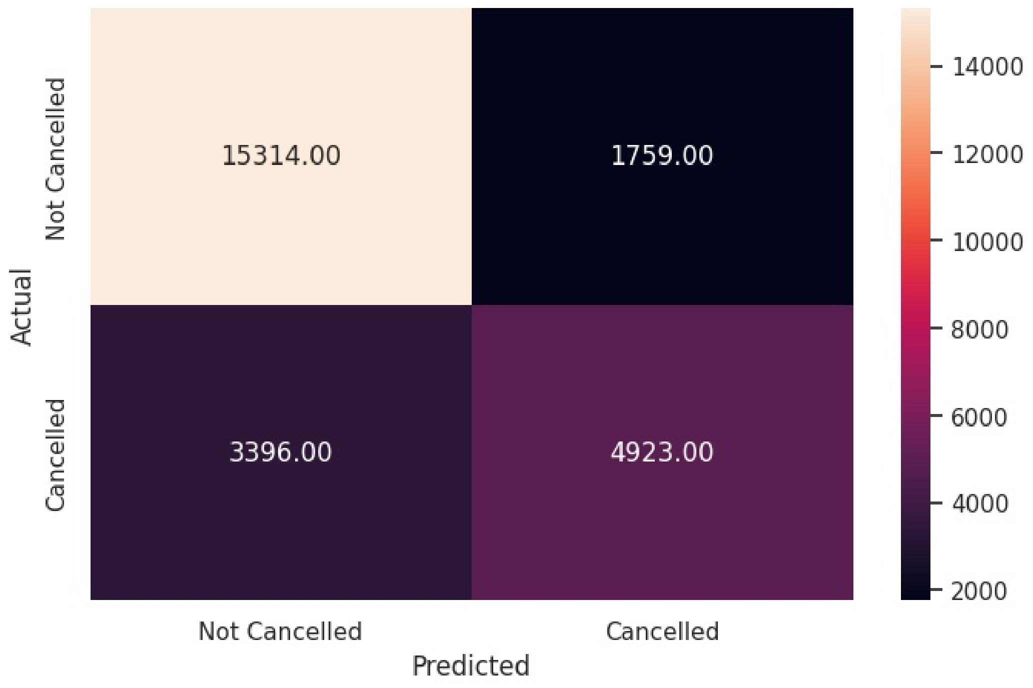


```
In [76]: optimal_threshold_svm=0.75
```

**Question 5.7: Check the performance of the model on train and test data using the optimal threshold. (2 Marks)**

```
In [77]: # Remove _________ and complete the code

         y_pred_train_svm = svm_rbf.predict_proba(X_train_scaled)
         metrics_score(y_train, y_pred_train[:,1]>optimal_threshold)
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86     17073
           1       0.74      0.59      0.66      8319

    accuracy                           0.80     25392
   macro avg       0.78      0.74      0.76     25392
weighted avg       0.79      0.80      0.79     25392
```

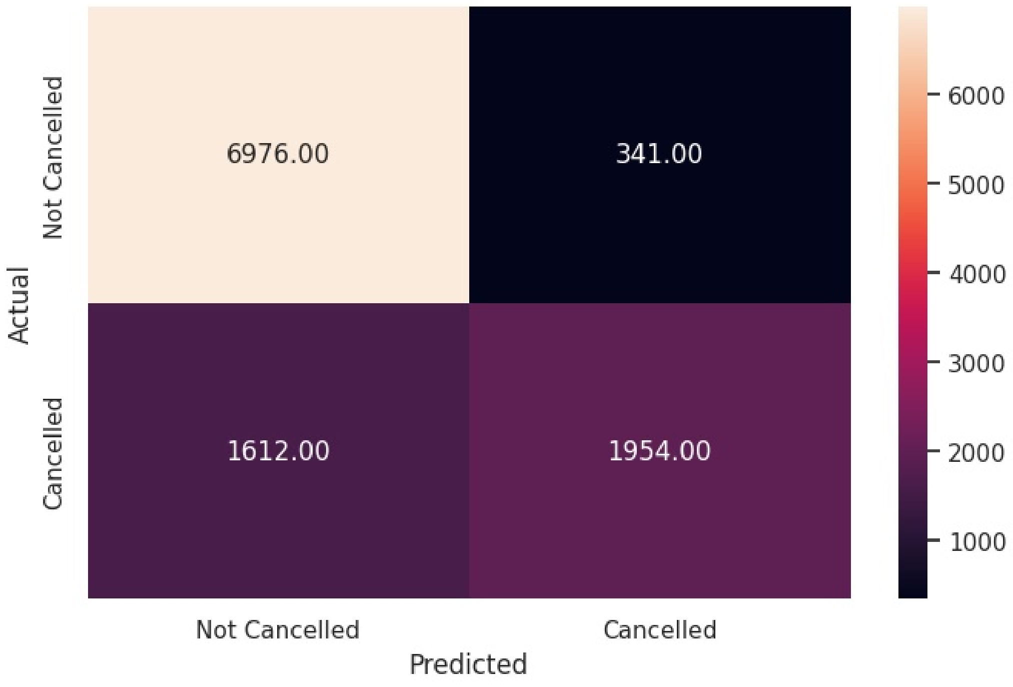


**Write your answers here:**

The performance of the model on the train set is slightly better than before, with a slightly higher F1-score for the minority class (1).

```
In [78]: # Remove _________ and complete the code

         y_pred_test = svm_rbf.predict_proba(X_test_scaled)
         metrics_score(y_test, y_pred_test[:,1]>optimal_threshold)
```

```
              precision    recall  f1-score   support

           0       0.81      0.95      0.88      7317
           1       0.85      0.55      0.67      3566

    accuracy                           0.82     10883
   macro avg       0.83      0.75      0.77     10883
weighted avg       0.83      0.82      0.81     10883
```



**Write your answers here:**

It is important to carefully tune the hyperparameters of the SVM model to find the optimal balance between model complexity and generalization performance.

## Question 6: Decision Trees (7 Marks)

### Question 6.1: Build a Decision Tree Model (1 Mark)

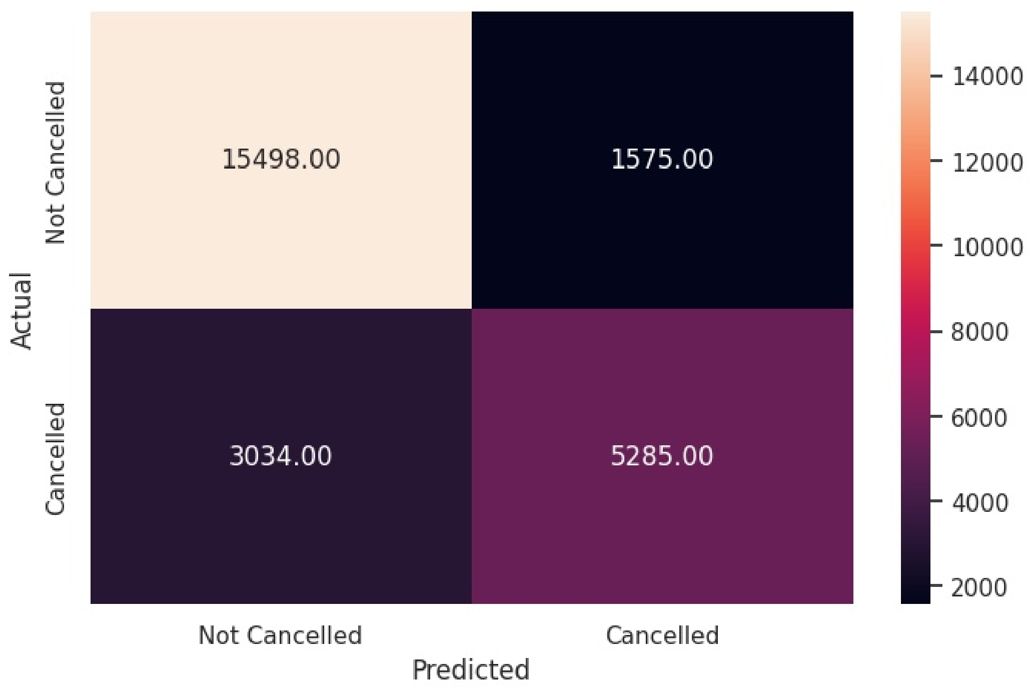
```
In [79]: # Remove _________ and complete the code

         model_dt = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
         model_dt.fit(X_train,y_train)
```

```
Out[79]: ▾              DecisionTreeClassifier
         DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

### Question 6.2: Check the performance of the model on train and test data (2 Marks)

```
In [80]: # Remove _________ and complete the code

         # Checking performance on the training dataset
         pred_train_dt = model_dt.predict(X_train)
         metrics_score(y_train,pred_train_dt)
```

```
              precision    recall  f1-score   support

           0       0.84      0.91      0.87     17073
           1       0.77      0.64      0.70      8319

    accuracy                           0.82     25392
   macro avg       0.80      0.77      0.78     25392
weighted avg       0.81      0.82      0.81     25392
```



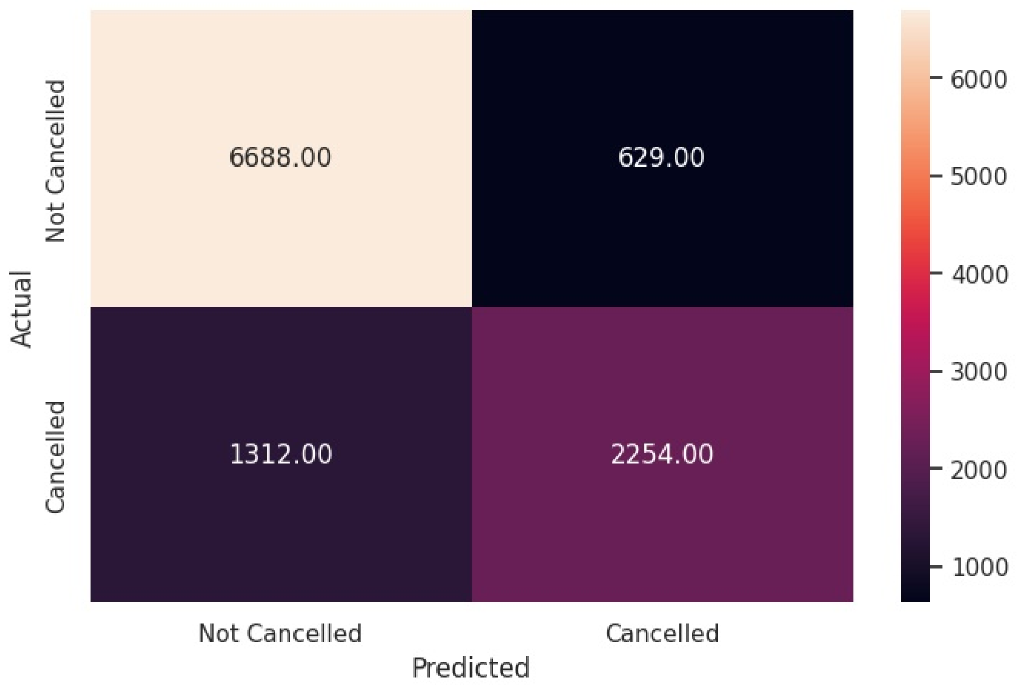
**Write your answers here:**

On the training set, the model achieves an accuracy of 0.82. The precision, recall, and F1-score for class 0 are 0.84, 0.91, and 0.87 respectively, while for class 1 they are 0.77, 0.64, and 0.70 respectively.

### Checking model performance on test set

```
In [81]: pred_test_dt = model_dt.predict(X_test)
         metrics_score(y_test, pred_test_dt)
```

```
              precision    recall  f1-score   support

           0       0.84      0.91      0.87      7317
           1       0.78      0.63      0.70      3566

    accuracy                           0.82     10883
   macro avg       0.81      0.77      0.79     10883
weighted avg       0.82      0.82      0.82     10883
```



**Write your answers here:**

On the test set, the model also achieves an accuracy of 0.82. The precision, recall, and F1-score for class 0 are 0.84, 0.91, and 0.87 respectively, while for class 1 they are 0.78, 0.63, and 0.70 respectively. Overall, the model performs similarly on both the training and test sets.

### Question 6.3: Perform hyperparameter tuning for the decision tree model using GridSearch CV (1 Mark)

**Note: Please use the following hyperparameters provided for tuning the Decision Tree. In general, you can experiment with various hyperparameters to tune the decision tree, but for this project, we recommend sticking to the parameters provided.**

In [83]:
```
# Remove _________ and complete the code

# Choose the type of classifier.
estimator = DecisionTreeClassifier(random_state=1)

# Grid of parameters to choose from
parameters = {
    "max_depth": np.arange(2, 7, 2),
    "max_leaf_nodes": [50, 75, 150, 250],
    "min_samples_split": [10, 30, 50, 70],
}


# Run the grid search
grid_obj = GridSearchCV(estimator, parameters)
grid_obj = grid_obj.fit(X_train, y_train)

# Set the clf to the best combination of parameters
estimator = grid_obj.best_estimator_

# Fit the best algorithm to the data.
estimator.fit(X_train, y_train)
```

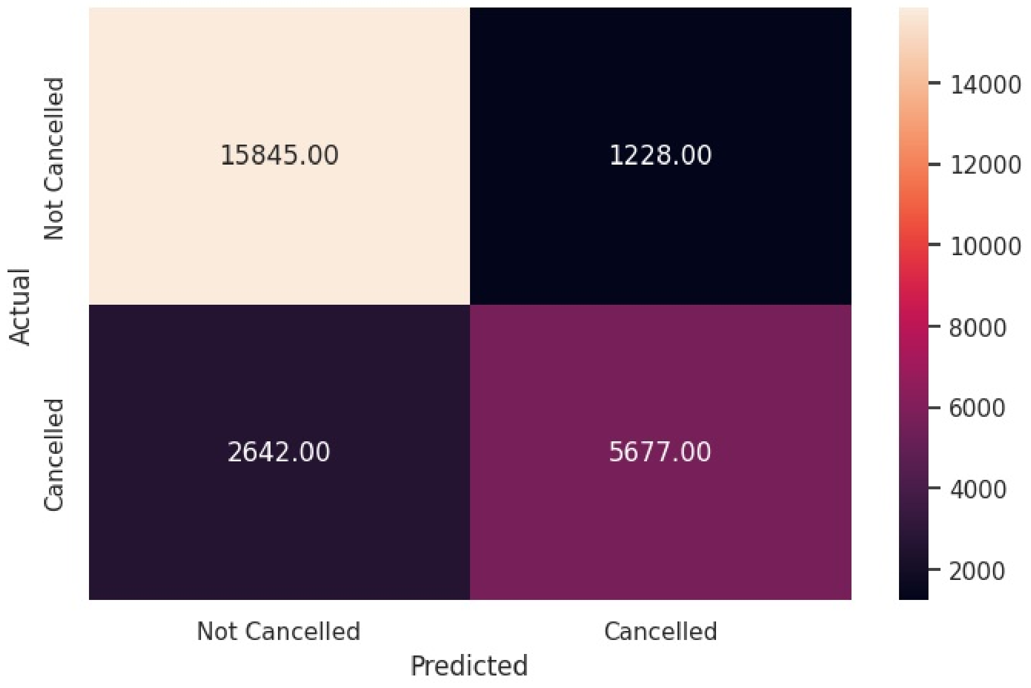
Out[83]:
```
▾                         DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, max_leaf_nodes=50, min_samples_split=10,
                       random_state=1)
```

### Question 6.4: Check the performance of the model on the train and test data using the tuned model (2 Mark)

Checking performance on the training set

In [84]:
```
# Remove _________ and complete the code

# Checking performance on the training dataset
dt_tuned = estimator.predict(X_train)
metrics_score(y_train, dt_tuned)
```

```
              precision    recall  f1-score   support

           0       0.86      0.93      0.89     17073
           1       0.82      0.68      0.75      8319

    accuracy                           0.85     25392
   macro avg       0.84      0.81      0.82     25392
weighted avg       0.85      0.85      0.84     25392
```

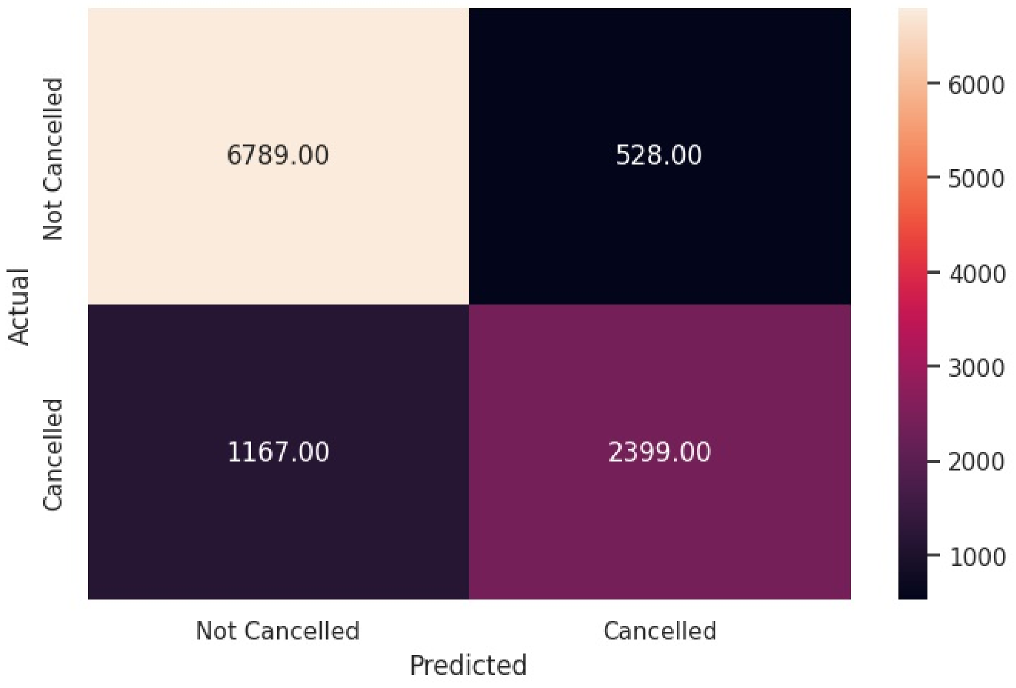


**Write your answers here:**

The tuned decision tree model is a better model than the previous one because it has been optimized for the best combination of hyperparameters, which allows it to perform better on the test data while avoiding overfitting.

In [109…
```python
# Remove _________ and complete the code

# Checking performance on the training dataset
y_pred_tuned = estimator.predict(X_test)
metrics_score(y_test, y_pred_tuned)
```

```
              precision    recall  f1-score   support

           0       0.85      0.93      0.89      7317
           1       0.82      0.67      0.74      3566

    accuracy                           0.84     10883
   macro avg       0.84      0.80      0.81     10883
weighted avg       0.84      0.84      0.84     10883
```

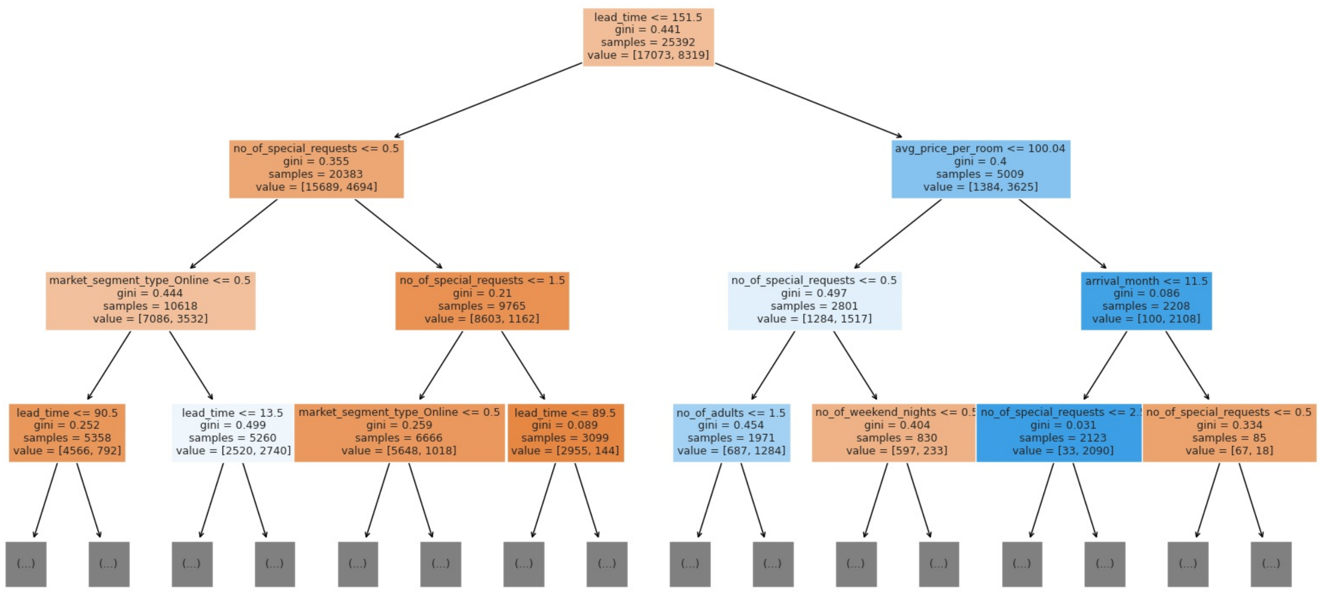


**Write your answers here:**

The performance of the decision tree model on the training data is slightly better than on the test data. The model achieved an accuracy of 85% on the training data and 84% on the test data, indicating that the model is slightly overfitting. However, the precision, recall, and F1-score values for both classes (0 and 1) are still relatively high on the test data, indicating that the model is still able to generalize well to new data despite the slight overfitting.

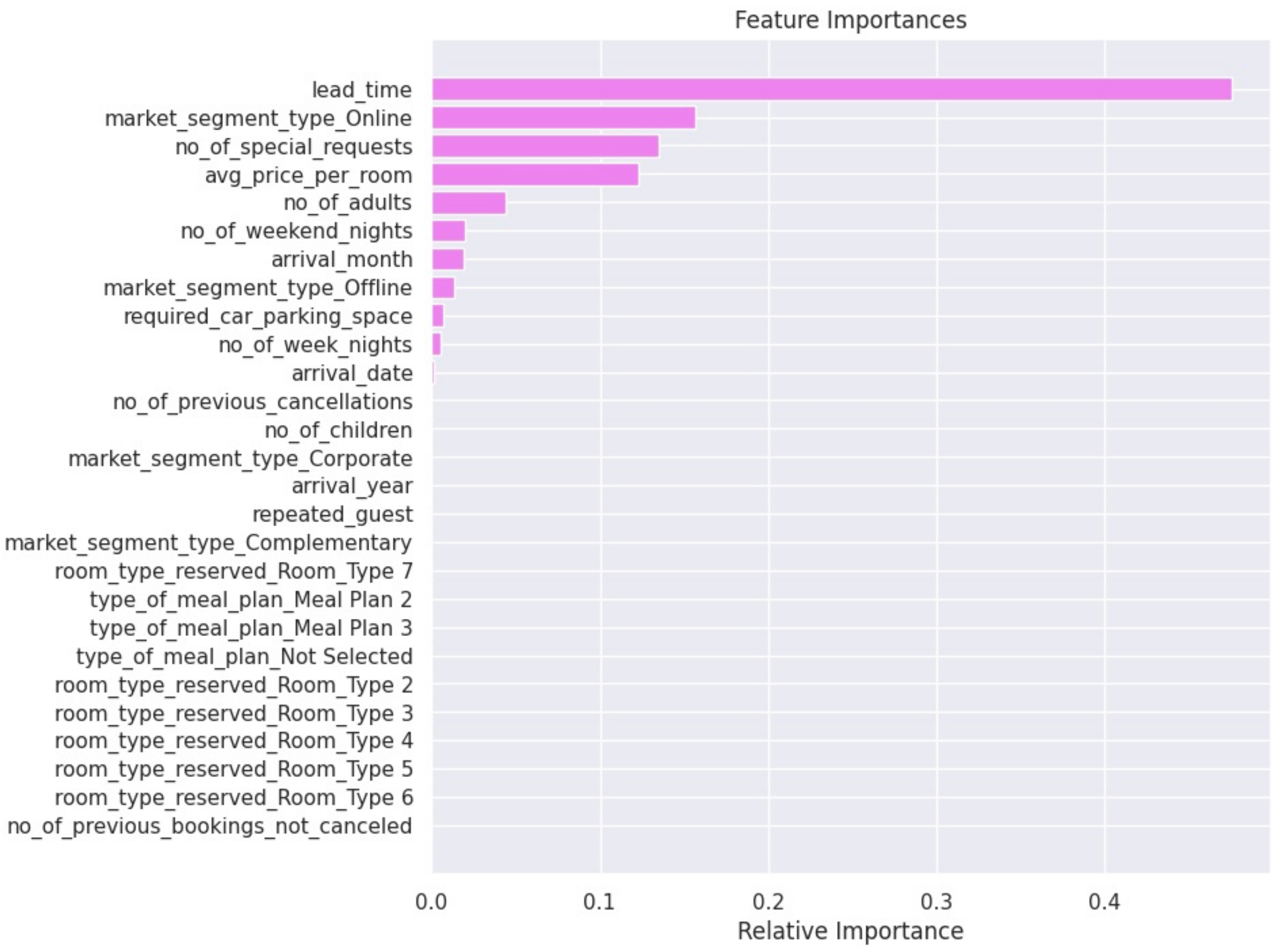
### Visualizing the Decision Tree

In [87]:
```python
feature_names = list(X_train.columns)
plt.figure(figsize=(20, 10))
out = tree.plot_tree(
    estimator,max_depth=3,
    feature_names=feature_names,
    filled=True,
    fontsize=9,
    node_ids=False,
    class_names=None,
)
# below code will add arrows to the decision tree split if they are missing
for o in out:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor("black")
        arrow.set_linewidth(1)
plt.show()
```



**Question 6.5: What are some important features based on the tuned decision tree? (1 Mark)**

In [88]:
```python
# Remove _________ and complete the code

# Importance of features in the tree building

importances = estimator.feature_importances_
indices = np.argsort(importances)

plt.figure(figsize=(8, 8))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], color="violet", align="center")
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```



**Write your answers here:**

Lead time: This indicates the number of days between the date the booking was made and the arrival date. This suggests that lead time is a critical factor for predicting hotel booking cancellations. From a business perspective, this information can be used to optimize revenue management and pricing strategies based on lead time trends.

Market Segment Type Online: bookings made online have a significant impact on the prediction of cancellations. This suggests that businesses may want to invest more in online marketing and sales strategies to boost their online bookings and reduce the likelihood of cancellations.

Number of special requests: Additional services may influence guests' booking decisions and satisfaction levels. Therefore, businesses should ensure that they are meeting the needs and expectations of their guests by providing high-quality services.

Average Price per Room: The price may not be the primary factor influencing cancellations. However, the hotel should still ensure that their prices are competitive and aligned with customer expectations.

Number of weekend nights: Weekends may be an essential factor in predicting cancellations. From a business perspective, this information can be used to adjust staffing levels and other operational procedures to better accommodate weekend guests.

Arrival month: Seasonal trends may impact cancellations. The Hotel should be aware of seasonal fluctuations and adjust their strategies accordingly to minimize cancellations.

Market Segment Type Offline: Offline booking process should be smooth and efficient.

Required Car Parking Space: The Hotel should still ensure that they are meeting the parking needs of their guests.

Number of weeknights: Weekdays may not have a significant impact on cancellations however is important to have operational planning for dont miss anything to have space and good experience.

---

## Question 7: Random Forest (4 Marks)

### Question 7.1: Build a Random Forest Model (1 Mark)

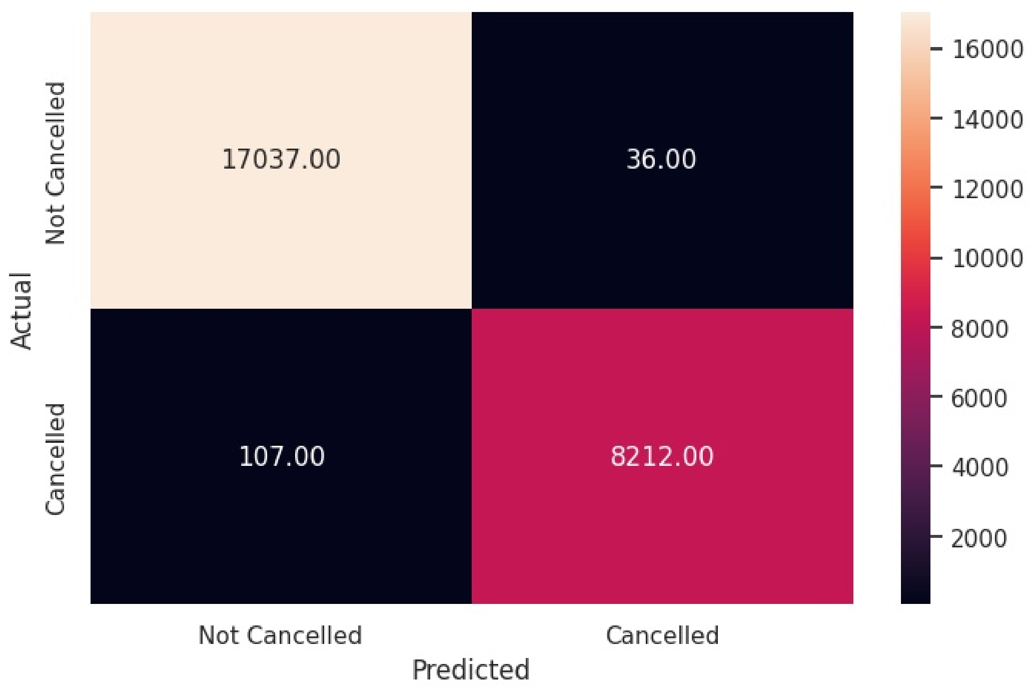
In [89]:
```
# Remove _________ and complete the code

rf_estimator = RandomForestClassifier(random_state=1)

rf_estimator.fit(X_train,y_train)
```

Out[89]:
```
▾        RandomForestClassifier
RandomForestClassifier(random_state=1)
```

### Question 7.2: Check the performance of the model on the train and test data (2 Marks)

In [90]:
```
# Remove _________ and complete the code

y_pred_train_rf = rf_estimator.predict(X_train)

metrics_score(y_train, y_pred_train_rf)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     17073
           1       1.00      0.99      0.99      8319

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

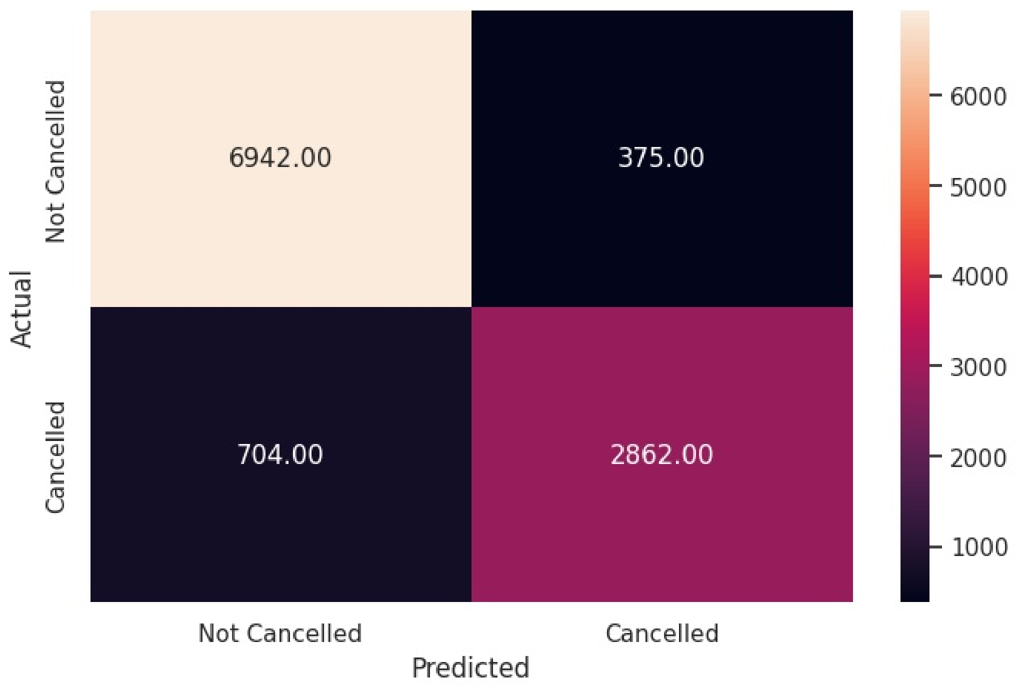


**Write your answers here:**

The high precision and recall scores, both close to 1.0, indicate that the model is performing well in correctly identifying both the positive and negative classes. The f1-score is also high, indicating a good balance between precision and recall. The accuracy score of 0.99 suggests that the model is making correct predictions for 99% of the observations in the test set. Overall, these scores indicate that the Random Forest model is performing very well on the train data.

```
In [91]: # Remove _________ and complete the code

         y_pred_test_rf = rf_estimator.predict(X_test)

         metrics_score(y_test, y_pred_test_rf)
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      7317
           1       0.88      0.80      0.84      3566

    accuracy                           0.90     10883
   macro avg       0.90      0.88      0.88     10883
weighted avg       0.90      0.90      0.90     10883
```



**Write your answers here:**

The model has an accuracy of 0.90. It's possible that the model is overfitting on the training data, but still performs well on the test data.

### Question 7.3: What are some important features based on the Random Forest? (1 Mark)

Let's check the feature importance of the Random Forest

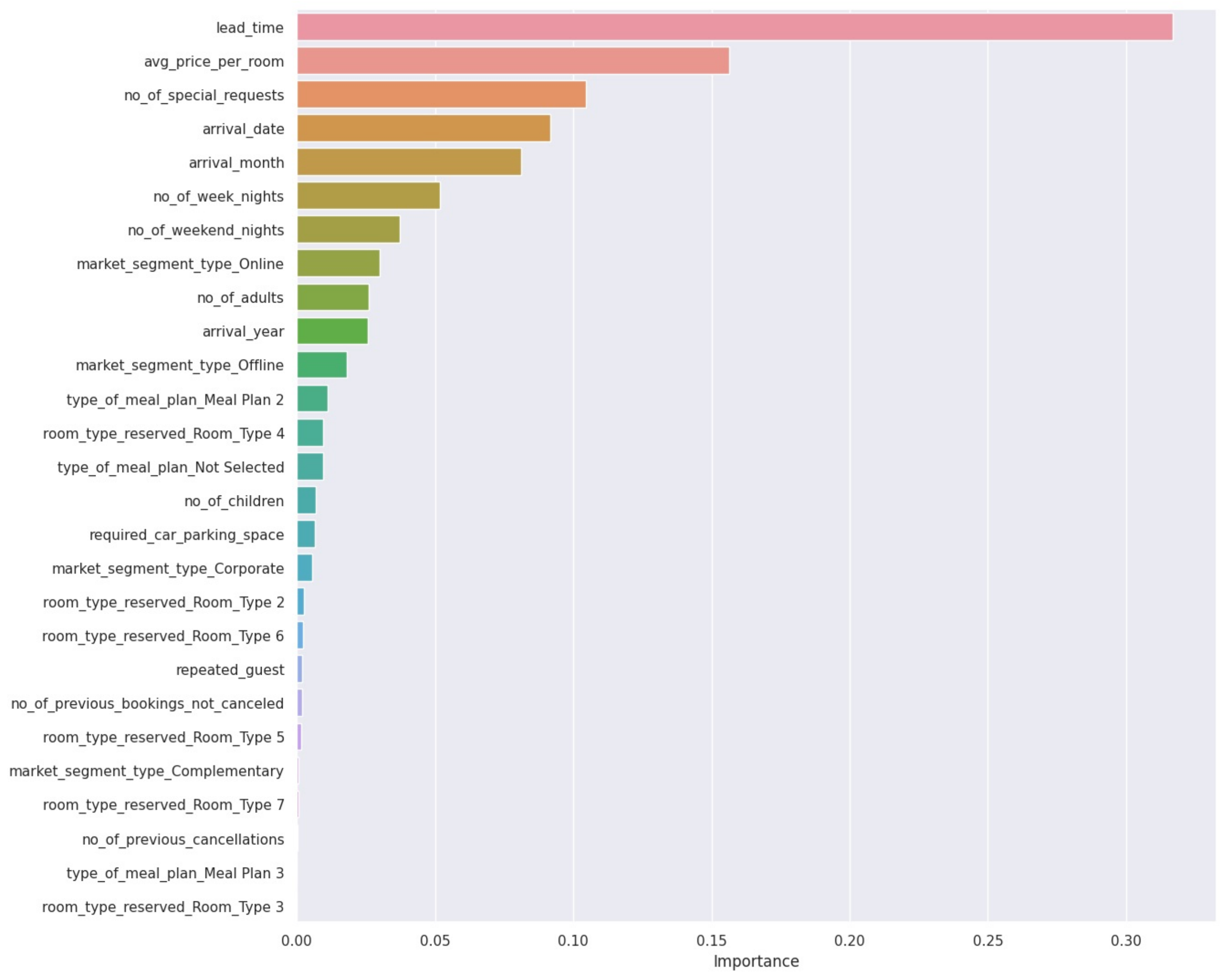
In [95]:
```
# Remove _________ and complete the code

importances = rf_estimator.feature_importances_

columns = X.columns

importance_df = pd.DataFrame(importances, index = columns, columns = ['Importance']).sort_values(by = 'Importan
```

In [98]:
```
importance_df.index
```

Out[98]:
```
Index(['lead_time', 'avg_price_per_room', 'no_of_special_requests',
       'arrival_date', 'arrival_month', 'no_of_week_nights',
       'no_of_weekend_nights', 'market_segment_type_Online', 'no_of_adults',
       'arrival_year', 'market_segment_type_Offline',
       'type_of_meal_plan_Meal Plan 2', 'room_type_reserved_Room_Type 4',
       'type_of_meal_plan_Not Selected', 'no_of_children',
       'required_car_parking_space', 'market_segment_type_Corporate',
       'room_type_reserved_Room_Type 2', 'room_type_reserved_Room_Type 6',
       'repeated_guest', 'no_of_previous_bookings_not_canceled',
       'room_type_reserved_Room_Type 5', 'market_segment_type_Complementary',
       'room_type_reserved_Room_Type 7', 'no_of_previous_cancellations',
       'type_of_meal_plan_Meal Plan 3', 'room_type_reserved_Room_Type 3'],
      dtype='object')
```

In [101…
```
plt.figure(figsize = (13, 13))

sns.barplot(x = importance_df.Importance, y = importance_df.index)
```

Out[101]:
```
<Axes: xlabel='Importance'>
```



**Write your answers here:**

Based on the feature importance of the Random Forest model, the top 3 important features in predicting hotel cancellations are:

lead_time avg_price_per_room no_of_special_requests

This information can be used to develop effective marketing strategies to retain customers and improve customer satisfaction. Additionally, features such as meal plans and parking can be considered as important extra benefits that can enhance the overall customer experience.

## Question 8: Conclude ANY FOUR key takeaways for business recommendations (4 Marks)

**Write your answers here:**

Personalize customer experiences: Based on the top features identified by the Random Forest model, such as lead time, average price per room, and number of special requests, the hotel can focus on personalizing its services and experiences for each customer. For example, they can offer customized room upgrades or unique packages based on each guest's preferences and booking history.

Improve online presence: The market segment type 'Online' was identified as an important feature in predicting hotel cancellations. This suggests that the hotel should invest in improving its online presence and remarketing efforts, such as creating a user-friendly website, optimizing its search engine rankings, and engaging with customers on social media platforms.

Innovate with new features: The model identified that meal plans and parking were important features in predicting hotel cancellations. Therefore, the hotel can consider innovating with new features, such as offering a wider range of meal plan options or implementing a smart parking system that enhances the guest experience.

Monitor and analyze performance: To continuously improve its business, the hotel should regularly monitor and analyze its performance metrics, such as customer satisfaction rates, booking and cancellation rates, and revenue generated. Using data analytics tools, the hotel can gain valuable insights into its business operations and customer behaviors, which can inform future business decisions and strategies.

Overall, by focusing on personalization, improving online presence, innovating with new features, and monitoring and analyzing performance, the hotel can enhance its business and provide a better customer experience.

## Happy Learning!