# INN Hotels Project

Marks : 60

## Context

A significant number of hotel bookings are called off due to cancellations or no-shows. The 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 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 impact 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

The 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 the 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 week 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.

## Please read the instructions carefully before starting the project.

This is a commented Jupyter IPython Notebook file in which all the instructions and tasks to be performed are mentioned.

- Blanks '**___**' are provided in the notebook that needs to be filled with an appropriate code to get the correct result. With every '**___**' blank, there is a comment that briefly describes what needs to be filled in the blank space.
- Identify the task to be performed correctly, and only then proceed to write the required code.
- Fill the code wherever asked by the commented lines like "# write your code here" or "# complete the code". Running incomplete code may throw error.
- Please run the codes in a sequential manner from the beginning to avoid any unnecessary errors.
- Add the results/observations (wherever mentioned) derived from the analysis in the presentation and submit the same.

In [1]:
```
# this will help in making the Python code more structured automatically
'%load_ext nb_black'

import warnings

warnings.filterwarnings("ignore")
from statsmodels.tools.sm_exceptions import ConvergenceWarning

warnings.simplefilter("ignore", ConvergenceWarning)

# 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

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)
# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)
# setting the precision of floating numbers to 5 decimal points
pd.set_option("display.float_format", lambda x: "%.5f" % x)

# Library to split data
from sklearn.model_selection import train_test_split

# To build model for prediction
import statsmodels.stats.api as sms
from statsmodels.stats.outliers_influence import variance_inflation_facto
import statsmodels.api as sm
from statsmodels.tools.tools import add_constant
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# To tune different models
from sklearn.model_selection import GridSearchCV


# To get diferent metric scores
from sklearn.metrics import (
    f1_score,
    accuracy_score,
    recall_score,
    precision_score,
    confusion_matrix,
    roc_auc_score,
    plot_confusion_matrix,
    precision_recall_curve,
    roc_curve,
    make_scorer,
)
```

# Import Dataset

In [7]:
```
hotel = pd.read_csv('INNHotelsGroup.csv') ##  Fill the blank to read the
```

In [8]:
```
# copying data to another variable to avoid any changes to original data
data = hotel.copy()
```

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

In [9]:
```
data.head() ##  Complete the code to view top 5 rows of the data
```

Out[9]:

| | Booking_ID | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights |
|---|---|---|---|---|---|
| **0** | INN00001 | 2 | 0 | 1 | 2 |
| **1** | INN00002 | 2 | 0 | 2 | 3 |
| **2** | INN00003 | 1 | 0 | 2 | 1 |
| **3** | INN00004 | 2 | 0 | 0 | 2 |
| **4** | INN00005 | 2 | 0 | 1 | 1 |

In [13]:
```
data.tail() ##  Complete the code to view last 5 rows of the data
```

Out[13]:

| | Booking_ID | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_ni |
|---|---|---|---|---|---|
| **36270** | INN36271 | 3 | 0 | 2 | |
| **36271** | INN36272 | 2 | 0 | 1 | |
| **36272** | INN36273 | 2 | 0 | 2 | |
| **36273** | INN36274 | 2 | 0 | 0 | |
| **36274** | INN36275 | 2 | 0 | 1 | |

## Understand the shape of the dataset

In [14]:
```
data.shape ##  Complete the code to view dimensions of the data
```

Out[14]:
```
(36275, 19)
```

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

In [15]:
```
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
```

In [16]:
```
# checking for duplicate values
data.duplicated().sum() ##  Complete the code to check duplicate entries
```

Out[16]: 0

**Let's drop the Booking_ID column first before we proceed forward.**

In [17]:
```
data = data.drop('Booking_ID',axis=1) ## Complete the code to drop the Bo
```

In [18]:
```
data.head()
```

Out[18]:

| | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | type_of_mea |
|---|---|---|---|---|---|
| **0** | 2 | 0 | 1 | 2 | Mea |
| **1** | 2 | 0 | 2 | 3 | Not S |
| **2** | 1 | 0 | 2 | 1 | Mea |
| **3** | 2 | 0 | 0 | 2 | Mea |
| **4** | 2 | 0 | 1 | 1 | Not S |

# Exploratory Data Analysis

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

In [19]:
```
data.describe()##  Complete the code to print the statistical summary of
```

Out[19]:

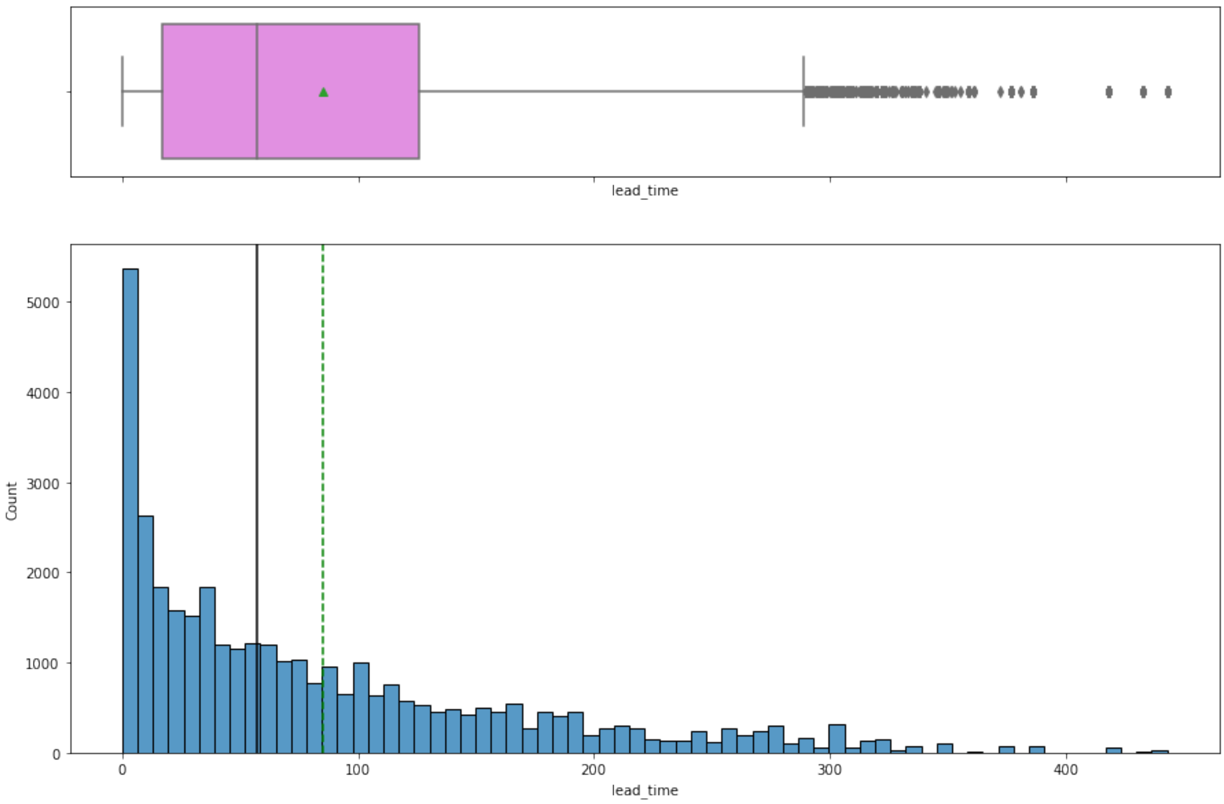
| | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | required |
|---|---|---|---|---|---|
| count | 36275.00000 | 36275.00000 | 36275.00000 | 36275.00000 | |
| mean | 1.84496 | 0.10528 | 0.81072 | 2.20430 | |
| std | 0.51871 | 0.40265 | 0.87064 | 1.41090 | |
| min | 0.00000 | 0.00000 | 0.00000 | 0.00000 | |
| 25% | 2.00000 | 0.00000 | 0.00000 | 1.00000 | |
| 50% | 2.00000 | 0.00000 | 1.00000 | 2.00000 | |
| 75% | 2.00000 | 0.00000 | 2.00000 | 3.00000 | |
| max | 4.00000 | 10.00000 | 7.00000 | 17.00000 | |

## Univariate Analysis

In [20]:
```
def histogram_boxplot(data, feature, figsize=(15, 10), kde=False, bins=No
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (15,10))
    kde: whether to show the density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows=2,  # Number of rows of the subplot grid= 2
        sharex=True,  # x-axis will be shared among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize,
    )  # creating the 2 subplots
    sns.boxplot(
        data=data, x=feature, ax=ax_box2, showmeans=True, color="violet"
    )  # boxplot will be created and a triangle will indicate the mean va
    sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins
    ) if bins else sns.histplot(
        data=data, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram
    ax_hist2.axvline(
        data[feature].mean(), color="green", linestyle="--"
    )  # Add mean to the histogram
    ax_hist2.axvline(
        data[feature].median(), color="black", linestyle="-"
    )  # Add median to the histogram
```

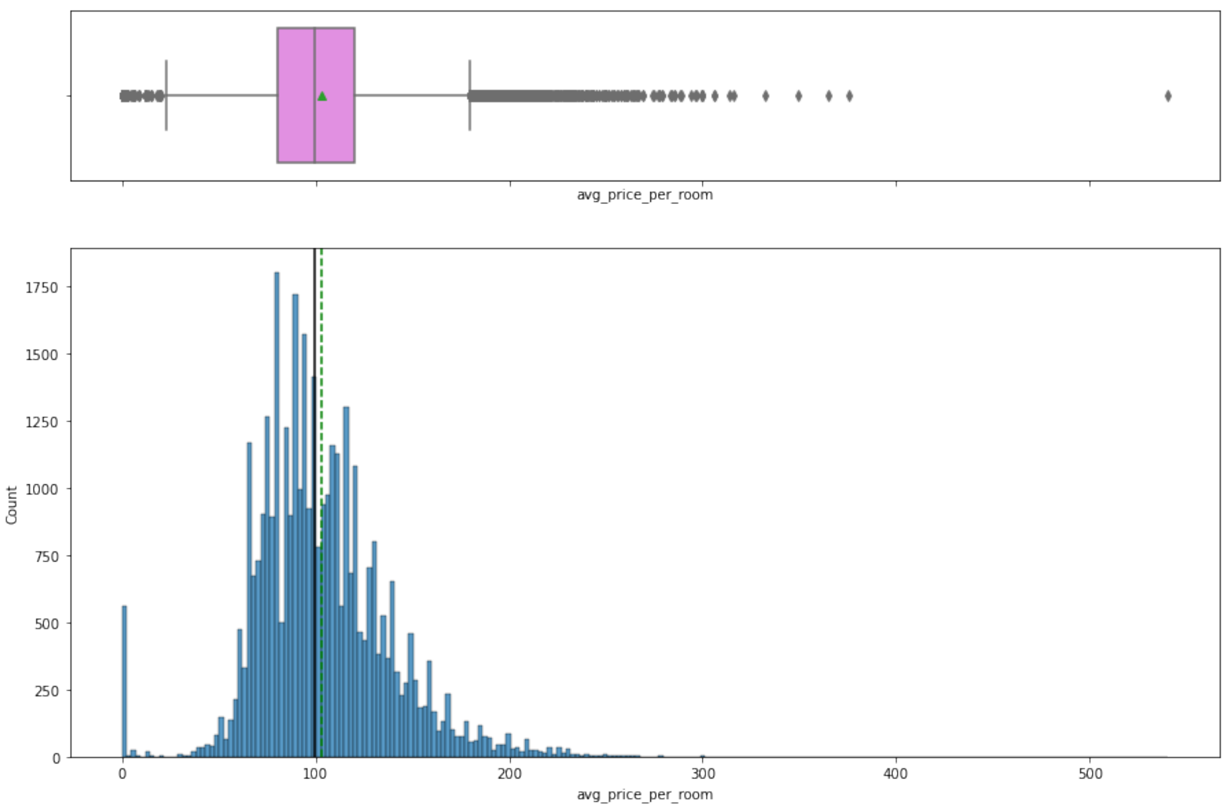
## Observations on lead time

In [21]:
```
histogram_boxplot(data, "lead_time")
```




## Observations on average price per room

In [22]:
```
histogram_boxplot(data,"avg_price_per_room")  ## Complete the code to cre
```



In [23]:
```
data[data["avg_price_per_room"] == 0]
```

Out[23]:

| | no_of_adults | no_of_children | no_of_weekend_nights | no_of_week_nights | type_of |
|---|---|---|---|---|---|
| **63** | 1 | 0 | 0 | 1 | |
| **145** | 1 | 0 | 0 | 2 | |
| **209** | 1 | 0 | 0 | 0 | |
| **266** | 1 | 0 | 0 | 2 | |
| **267** | 1 | 0 | 2 | 1 | |
| **...** | ... | ... | ... | ... | |
| **35983** | 1 | 0 | 0 | 1 | |
| **36080** | 1 | 0 | 1 | 1 | |
| **36114** | 1 | 0 | 0 | 1 | |
| **36217** | 2 | 0 | 2 | 1 | |
| **36250** | 1 | 0 | 0 | 2 | |

545 rows × 18 columns

In [24]:
```
data.loc[data["avg_price_per_room"] == 0, "market_segment_type"].value_co
```

Out[24]:
```
Complementary    354
Online           191
Name: market_segment_type, dtype: int64
```

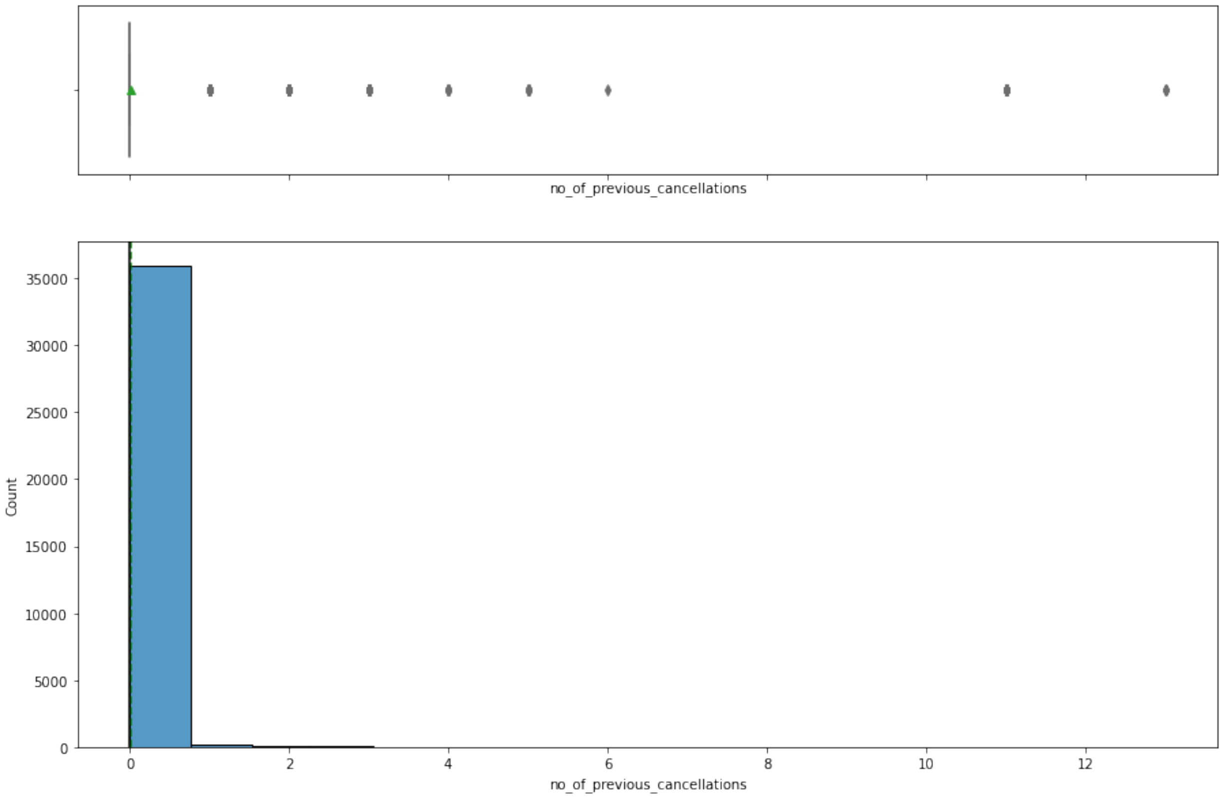
In [25]:
```
# Calculating the 25th quantile
Q1 = data["avg_price_per_room"].quantile(0.25)

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

# Calculating IQR
IQR = Q3 - Q1

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

Out[25]:
```
179.55
```

In [26]:
```
# assigning the outliers the value of upper whisker
data.loc[data["avg_price_per_room"] >= 500, "avg_price_per_room"] = Upper_
```

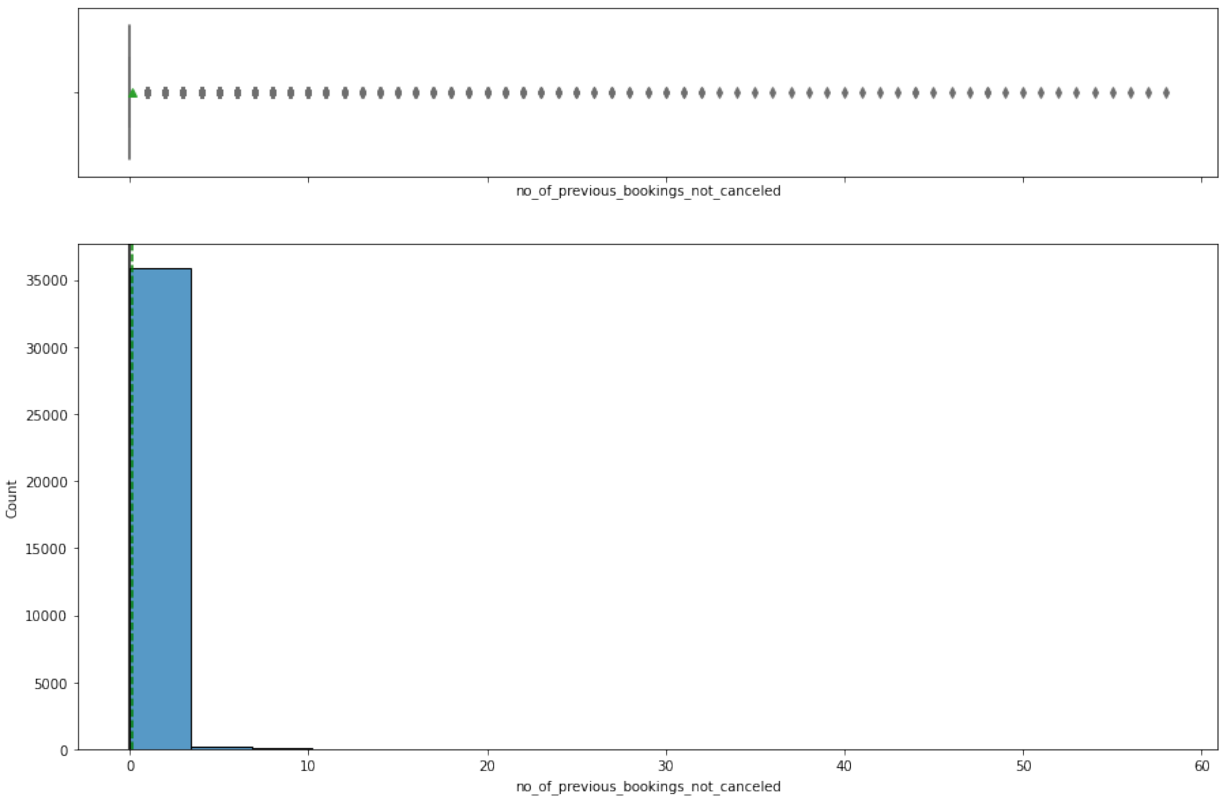
## Observations on number of previous booking cancellations

In [27]:
```
histogram_boxplot(data,'no_of_previous_cancellations')  ## Complete the c
```




## Observations on number of previous booking not canceled

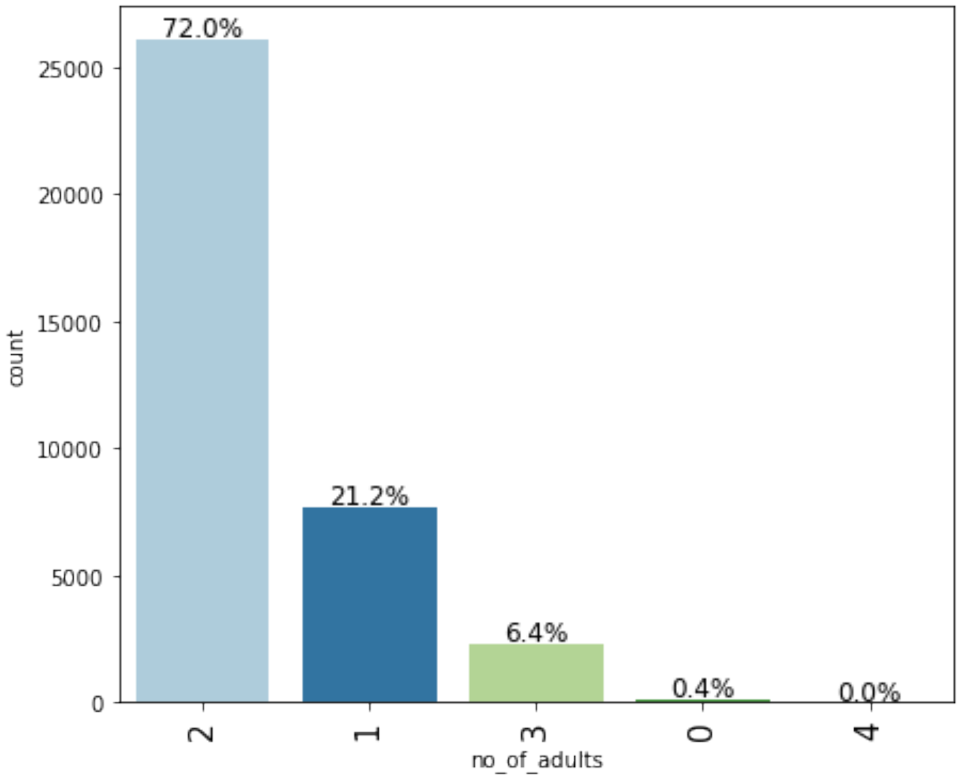
In [28]:
```
histogram_boxplot(data,'no_of_previous_bookings_not_canceled')  ## Comple
```



In [29]:
```
# function to create labeled barplots


def labeled_barplot(data, feature, perc=False, n=None):
    """
    Barplot with percentage at the top

    data: dataframe
    feature: dataframe column
    perc: whether to display percentages instead of count (default is Fal
    n: displays the top n category levels (default is None, i.e., display
    """

    total = len(data[feature])  # length of the column
    count = data[feature].nunique()
    if n is None:
        plt.figure(figsize=(count + 2, 6))
    else:
        plt.figure(figsize=(n + 2, 6))

    plt.xticks(rotation=90, fontsize=15)
    ax = sns.countplot(
        data=data,
        x=feature,
        palette="Paired",
        order=data[feature].value_counts().index[:n],
    )

    for p in ax.patches:
        if perc == True:
            label = "{:.1f}%".format(
                100 * p.get_height() / total
            )  # percentage of each class of the category
        else:
            label = p.get_height()  # count of each level of the category

        x = p.get_x() + p.get_width() / 2  # width of the plot
        y = p.get_height()  # height of the plot

        ax.annotate(
            label,
            (x, y),
            ha="center",
            va="center",
            size=12,
            xytext=(0, 5),
            textcoords="offset points",
        )  # annotate the percentage

    plt.show()  # show the plot
```

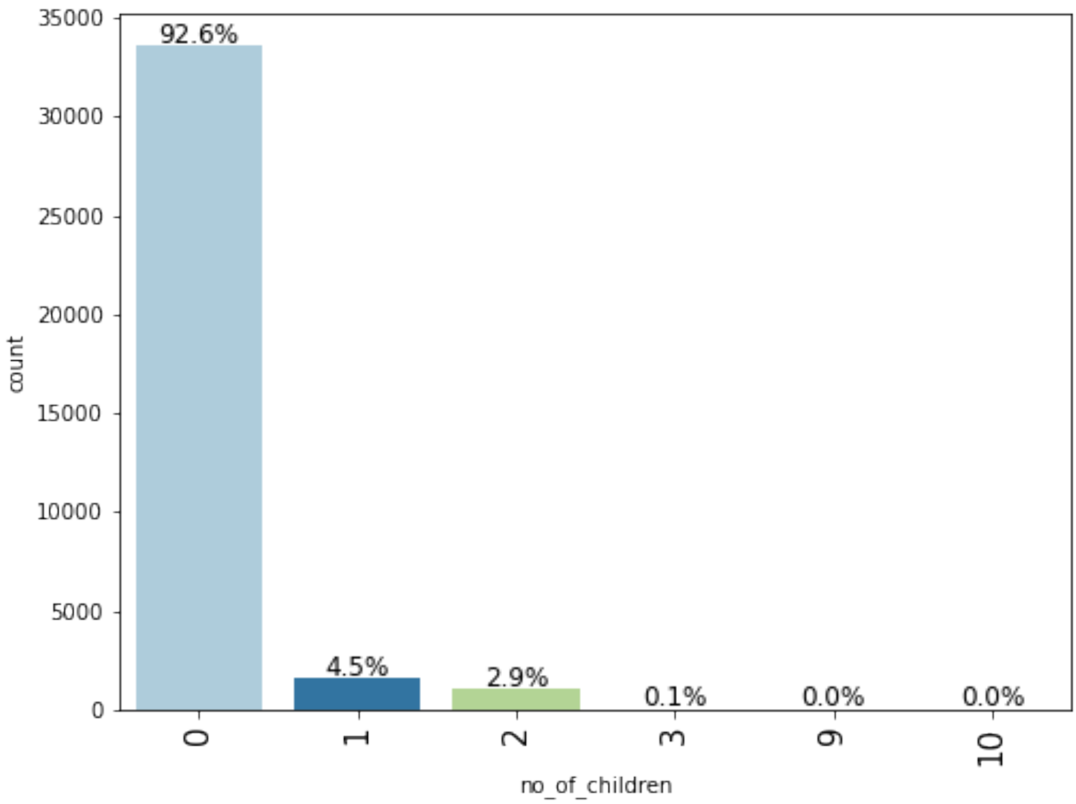
## Observations on number of adults

In [30]:
```
labeled_barplot(data, "no_of_adults", perc=True)
```




## Observations on number of children

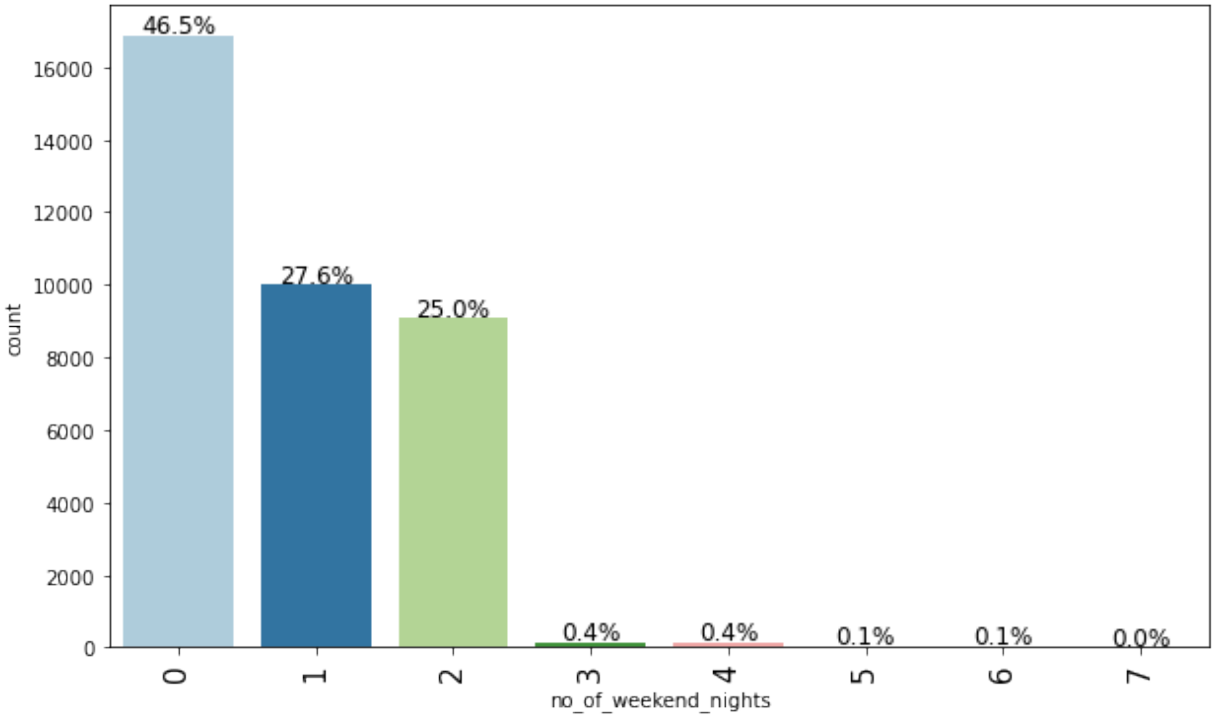
In [31]:
```
labeled_barplot(data,'no_of_children',perc=True)  ## Complete the code to
```




In [32]:
```
# replacing 9, and 10 children with 3
data["no_of_children"] = data["no_of_children"].replace([9, 10], 3)
```

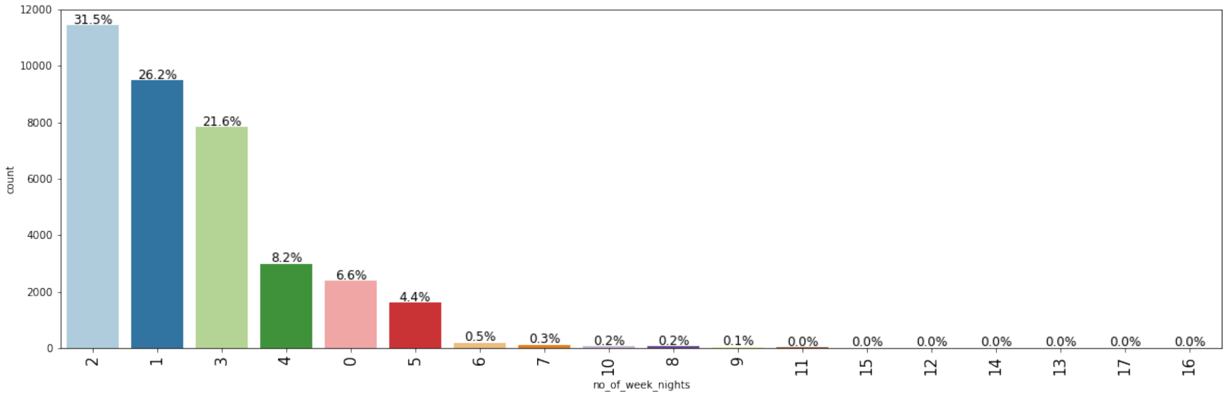
## Observations on number of week nights

In [33]:
```
labeled_barplot(data,'no_of_weekend_nights',perc=True)  ## Complete the c
```



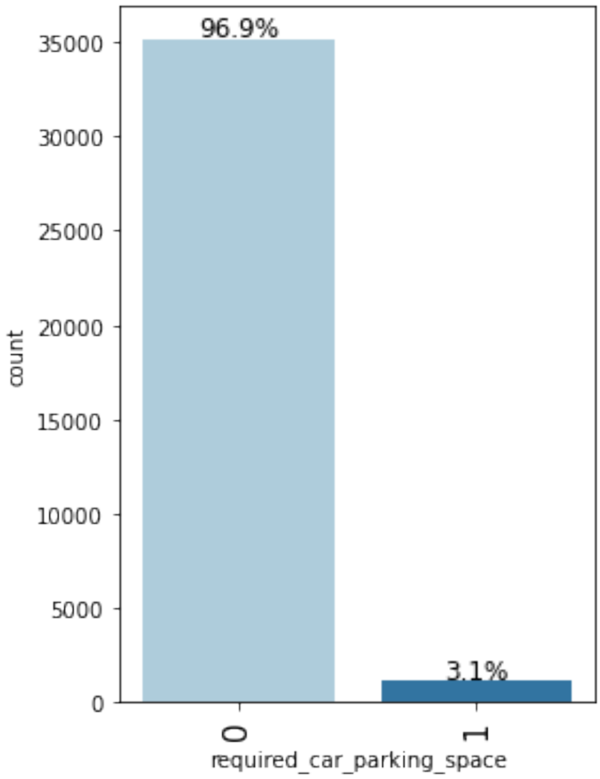

## Observations on number of weekend nights

In [34]:
```
labeled_barplot(data,'no_of_week_nights',perc=True)  ## Complete the code
```



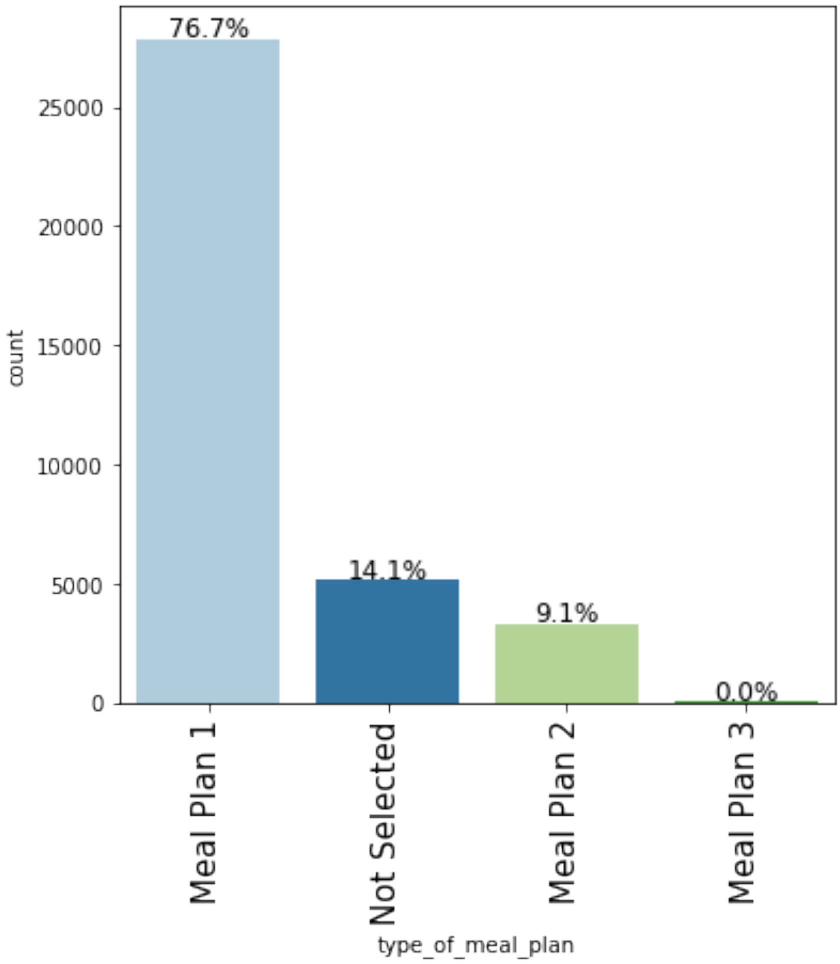

## Observations on required car parking space

In [35]:
```
labeled_barplot(data,'required_car_parking_space',perc=True)  ## Complete
```



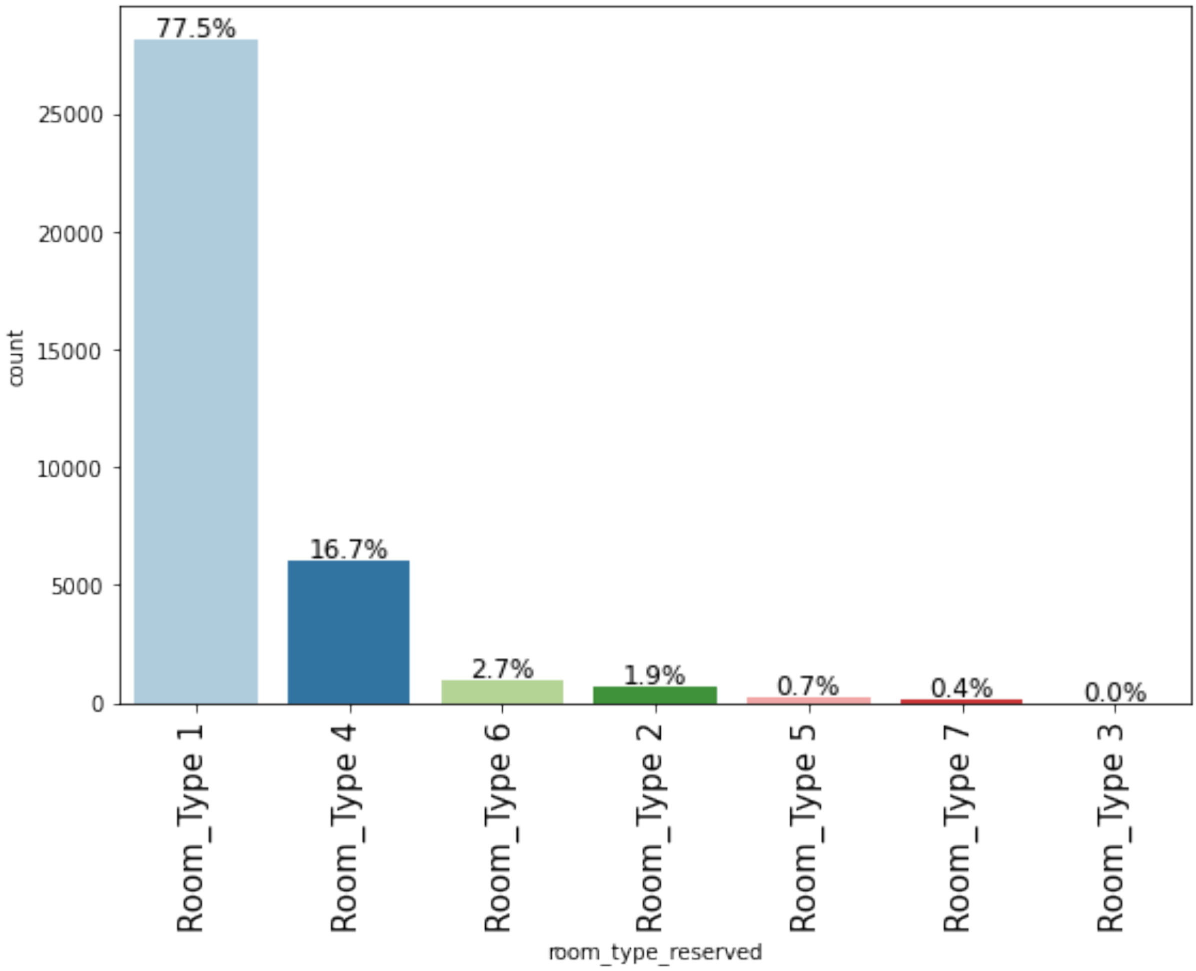

## Observations on type of meal plan

In [36]:
```
labeled_barplot(data,'type_of_meal_plan',perc=True)  ## Complete the code
```



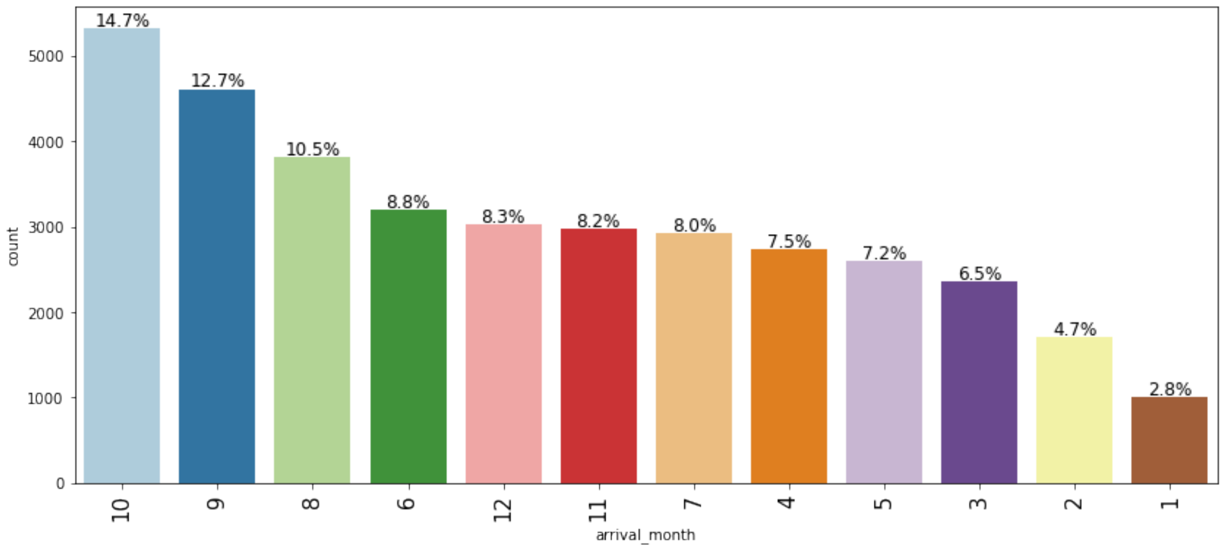

## Observations on room type reserved

In [37]:
```
labeled_barplot(data,'room_type_reserved',perc=True)   ## Complete the cod
```



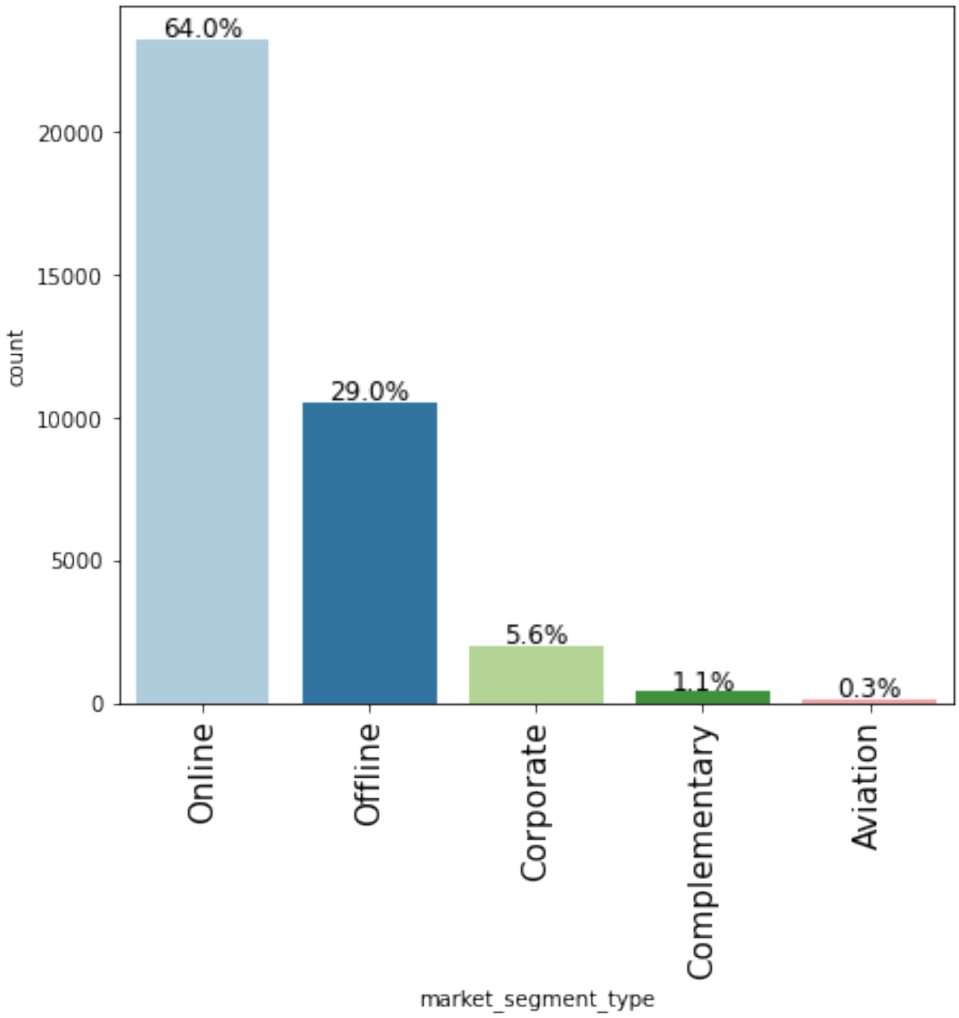

## Observations on arrival month

In [38]:
```
labeled_barplot(data,'arrival_month',perc=True)  ## Complete the code to
```



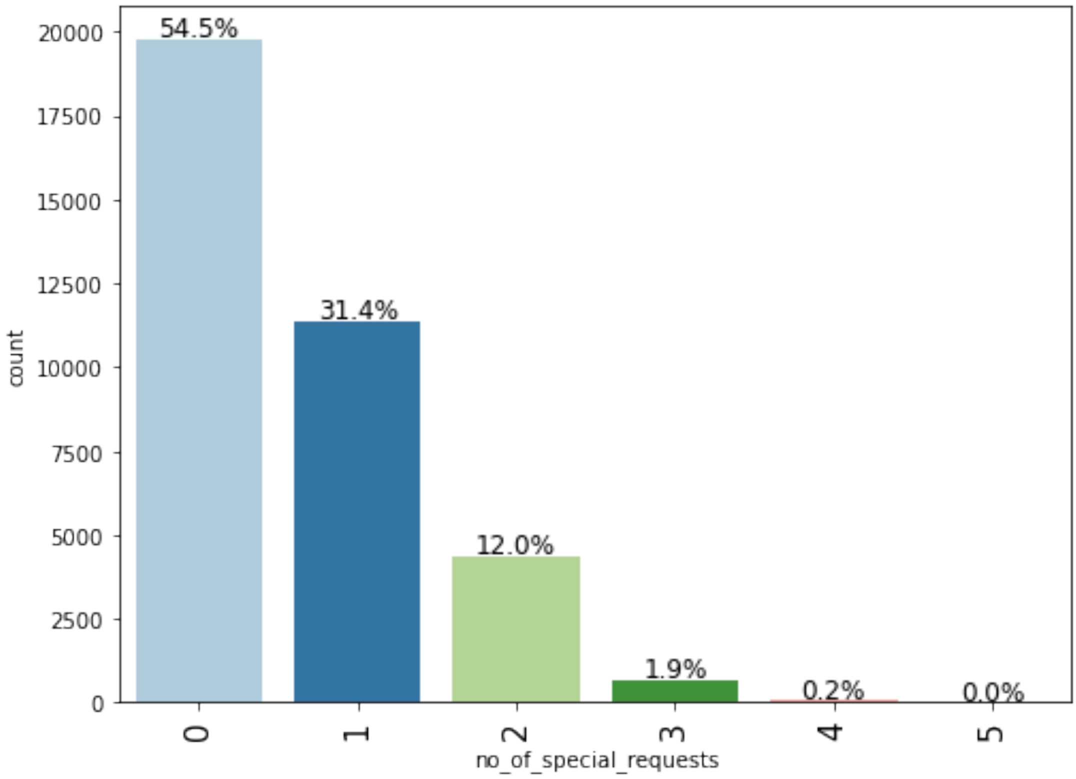

## Observations on market segment type

In [39]:
```
labeled_barplot(data,'market_segment_type',perc=True)  ## Complete the co
```



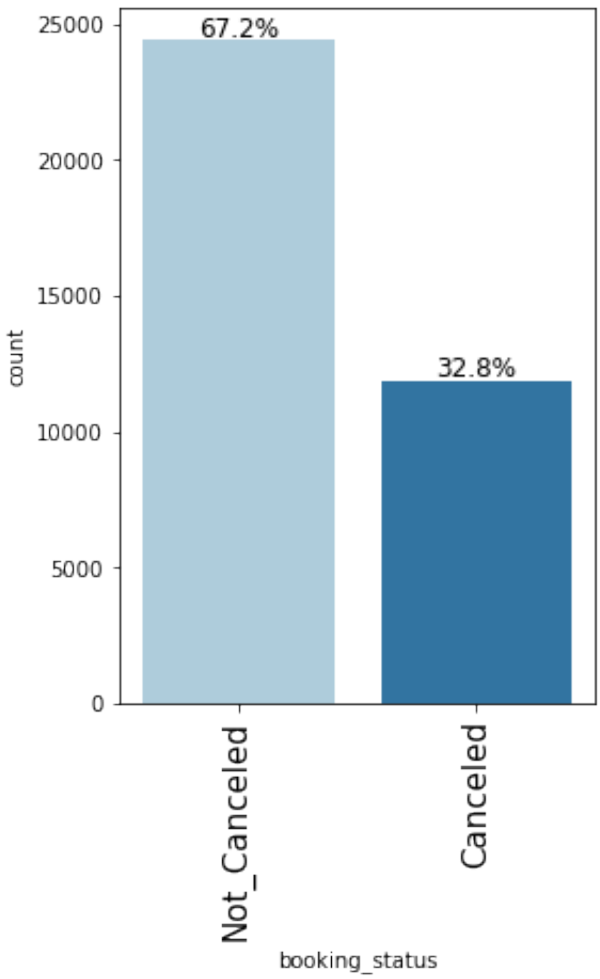

## Observations on number of special requests

```
In [40]: labeled_barplot(data,'no_of_special_requests',perc=True)   ## Complete the
```




## Observations on booking status

In [41]:
```
labeled_barplot(data,'booking_status',perc=True)  ## Complete the code to
```

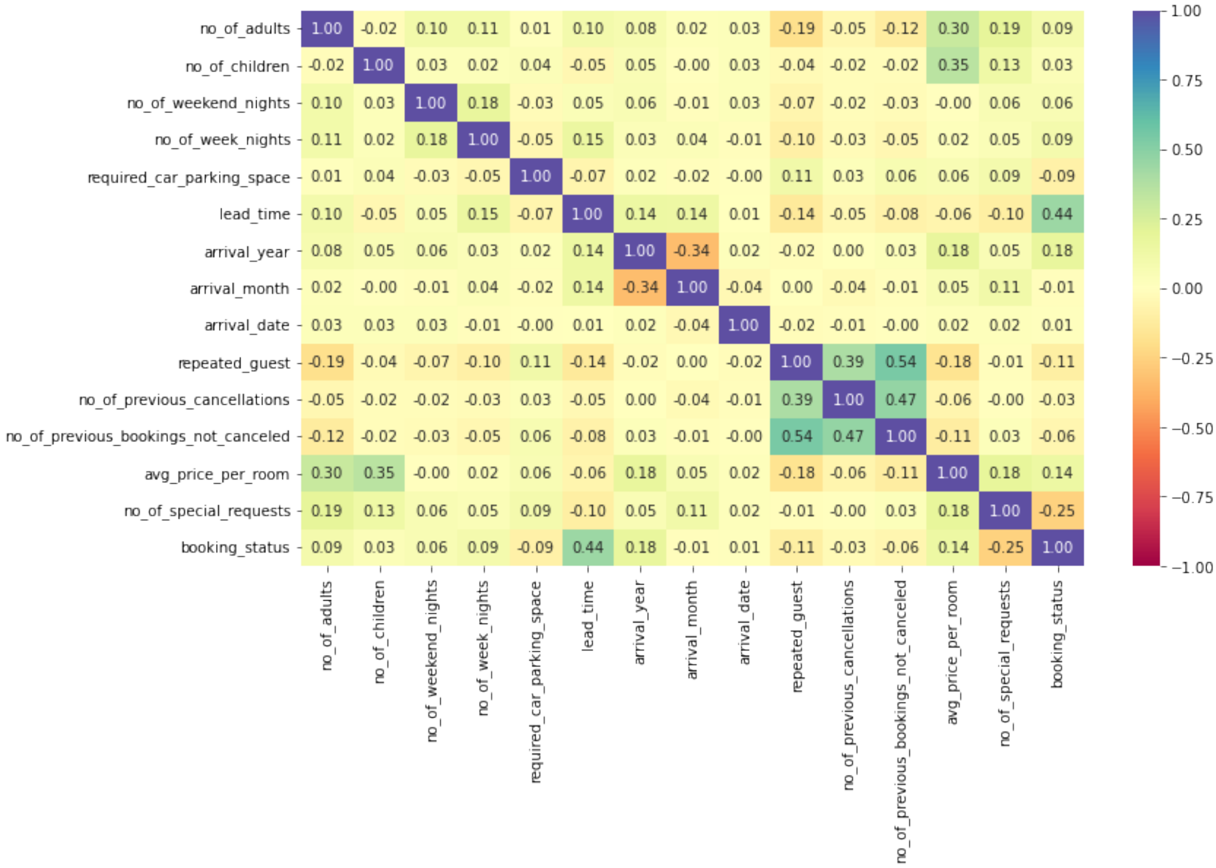



**Let's encode Canceled bookings to 1 and Not_Canceled as 0 for further analysis**

In [42]:
```
data["booking_status"] = data["booking_status"].apply(
    lambda x: 1 if x == "Canceled" else 0
)
```

## Bivariate Analysis

In [43]:
```
cols_list = data.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(12, 7))
sns.heatmap(
    data[cols_list].corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap=
)
plt.show()
```



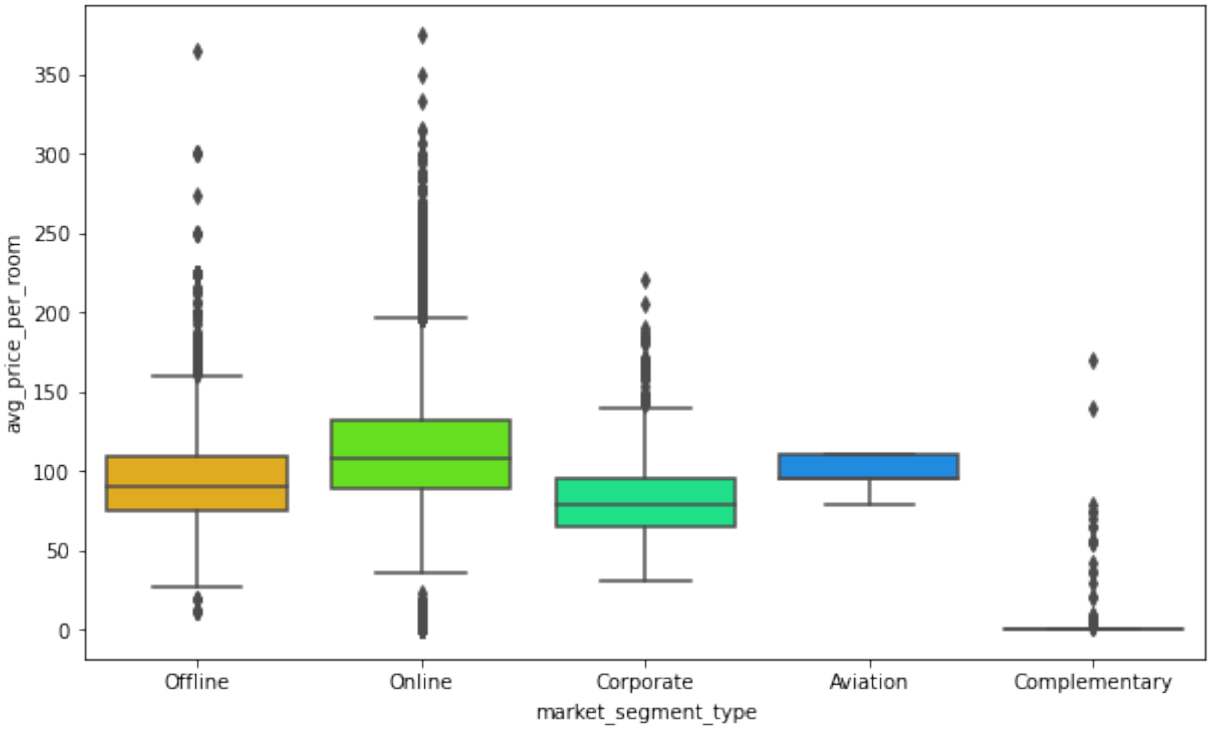
## Creating functions that will help us with further analysis.

In [44]:
```
### function to plot distributions wrt target


def distribution_plot_wrt_target(data, predictor, target):

    fig, axs = plt.subplots(2, 2, figsize=(12, 10))

    target_uniq = data[target].unique()

    axs[0, 0].set_title("Distribution of target for target=" + str(target_
    sns.histplot(
        data=data[data[target] == target_uniq[0]],
        x=predictor,
        kde=True,
        ax=axs[0, 0],
        color="teal",
        stat="density",
    )

    axs[0, 1].set_title("Distribution of target for target=" + str(target_
    sns.histplot(
        data=data[data[target] == target_uniq[1]],
        x=predictor,
        kde=True,
        ax=axs[0, 1],
        color="orange",
        stat="density",
    )

    axs[1, 0].set_title("Boxplot w.r.t target")
    sns.boxplot(data=data, x=target, y=predictor, ax=axs[1, 0], palette="

    axs[1, 1].set_title("Boxplot (without outliers) w.r.t target")
    sns.boxplot(
        data=data,
        x=target,
        y=predictor,
        ax=axs[1, 1],
        showfliers=False,
        palette="gist_rainbow",
    )

    plt.tight_layout()
    plt.show()
```

In [45]:
```
def stacked_barplot(data, predictor, target):
    """
    Print the category counts and plot a stacked bar chart

    data: dataframe
    predictor: independent variable
    target: target variable
    """
    count = data[predictor].nunique()
    sorter = data[target].value_counts().index[-1]
    tab1 = pd.crosstab(data[predictor], data[target], margins=True).sort_
        by=sorter, ascending=False
    )
    print(tab1)
    print("-" * 120)
    tab = pd.crosstab(data[predictor], data[target], normalize="index").s
        by=sorter, ascending=False
    )
    tab.plot(kind="bar", stacked=True, figsize=(count + 5, 5))
    plt.legend(
        loc="lower left", frameon=False,
    )
    plt.legend(loc="upper left", bbox_to_anchor=(1, 1))
    plt.show()
```

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

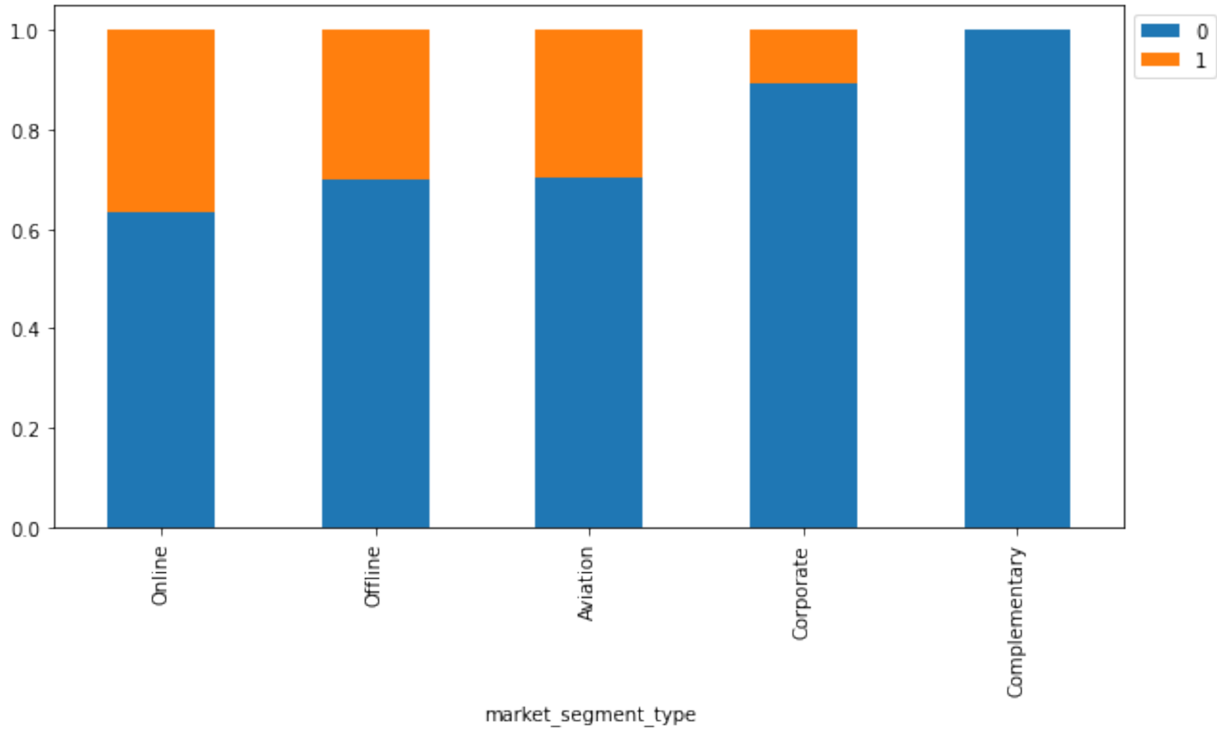
In [46]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=data, x="market_segment_type", y="avg_price_per_room", palette="
)
plt.show()
```




**Let's see how booking status varies across different market segments. Also, how average price per room impacts booking status**

In [47]:
```
stacked_barplot(data, "market_segment_type", "booking_status")
```

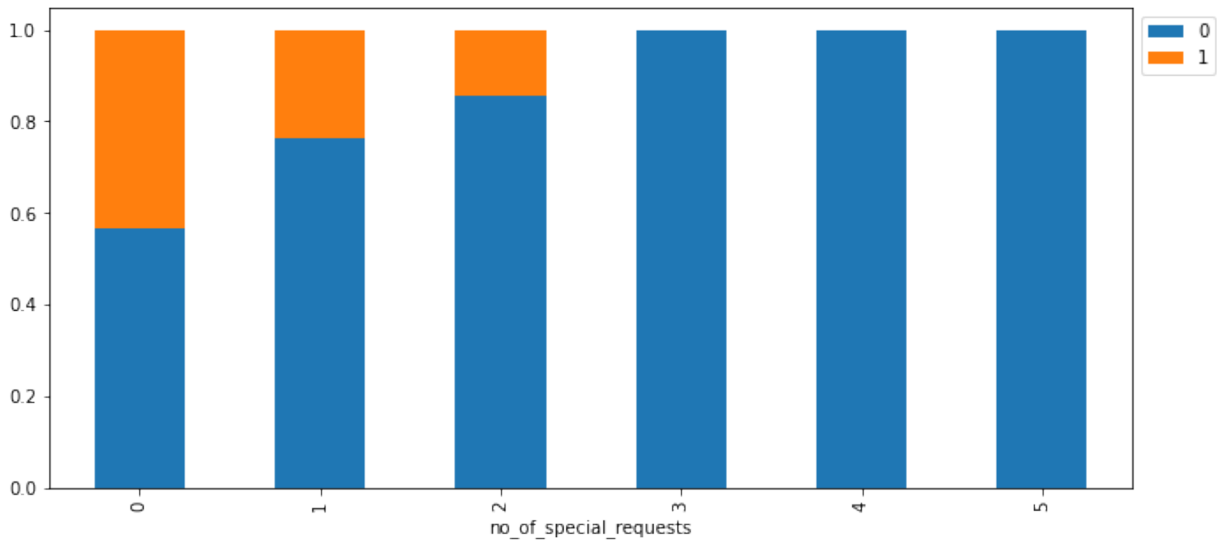
```
booking_status           0      1    All
market_segment_type
All                  24390  11885  36275
Online               14739   8475  23214
Offline               7375   3153  10528
Corporate             1797    220   2017
Aviation                88     37    125
Complementary          391      0    391
------------------------------------------------------------------------
------------------------------------------------
```




**Many guests have special requirements when booking a hotel room. Let's see how it impacts cancellations**

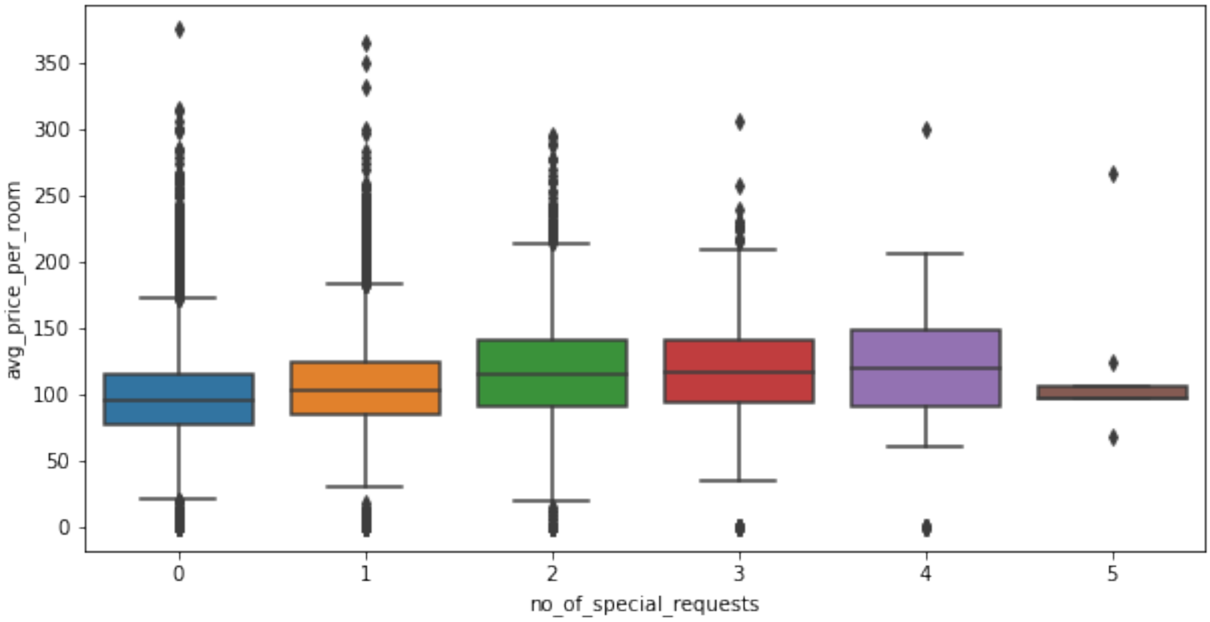
In [48]:
```
stacked_barplot(data,'no_of_special_requests','booking_status') ## Comple
```

```
booking_status              0      1    All
no_of_special_requests
All                     24390  11885  36275
0                       11232   8545  19777
1                        8670   2703  11373
2                        3727    637   4364
3                         675      0    675
4                          78      0     78
5                           8      0      8
------------------------------------------------------------------------
----------------------------------------------
```



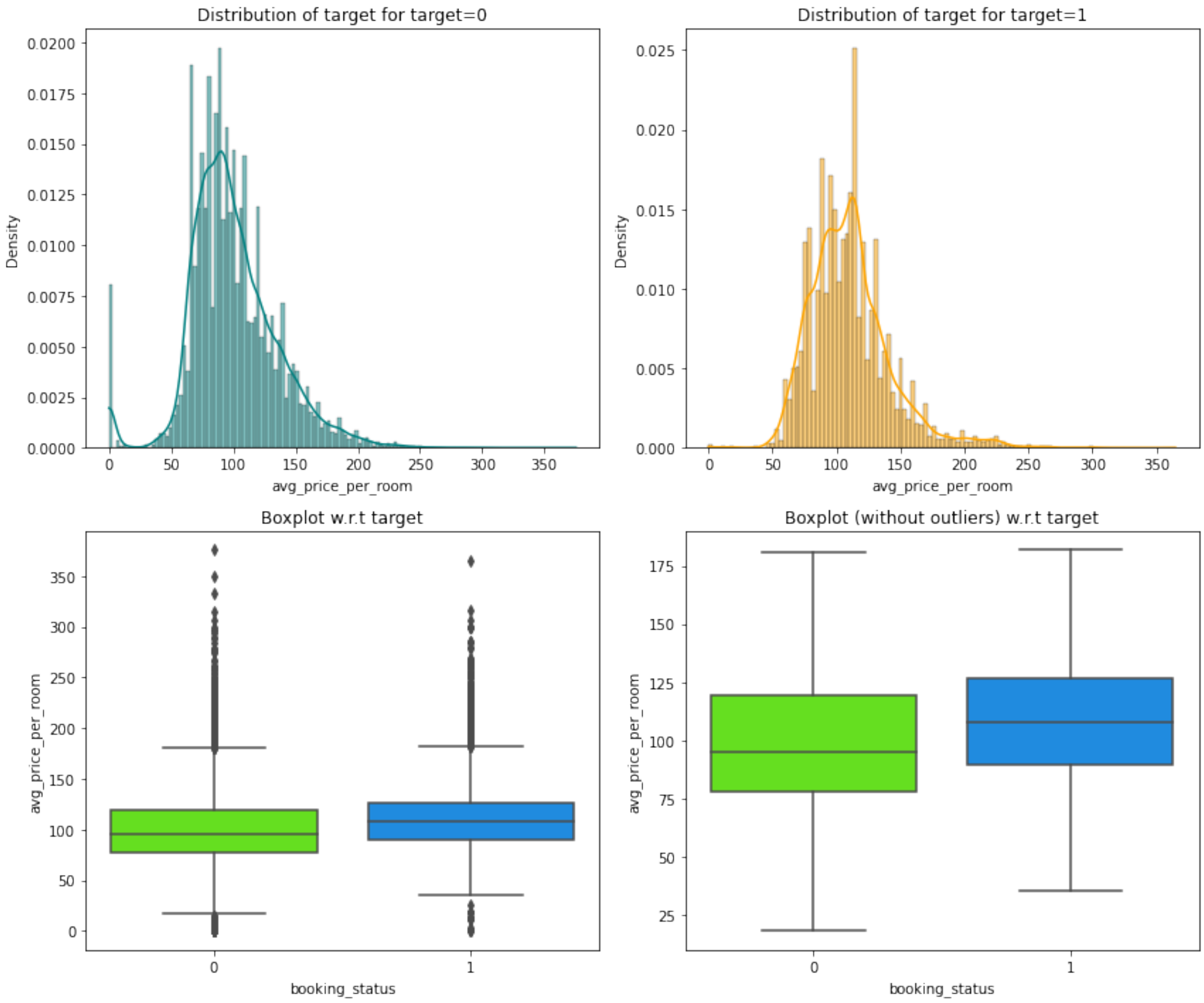
#### Let's see if the special requests made by the customers impacts the prices of a room

In [152…
```
plt.figure(figsize=(10, 5))
sns.boxplot(data=data, x="no_of_special_requests", y="avg_price_per_room"
plt.show()
```



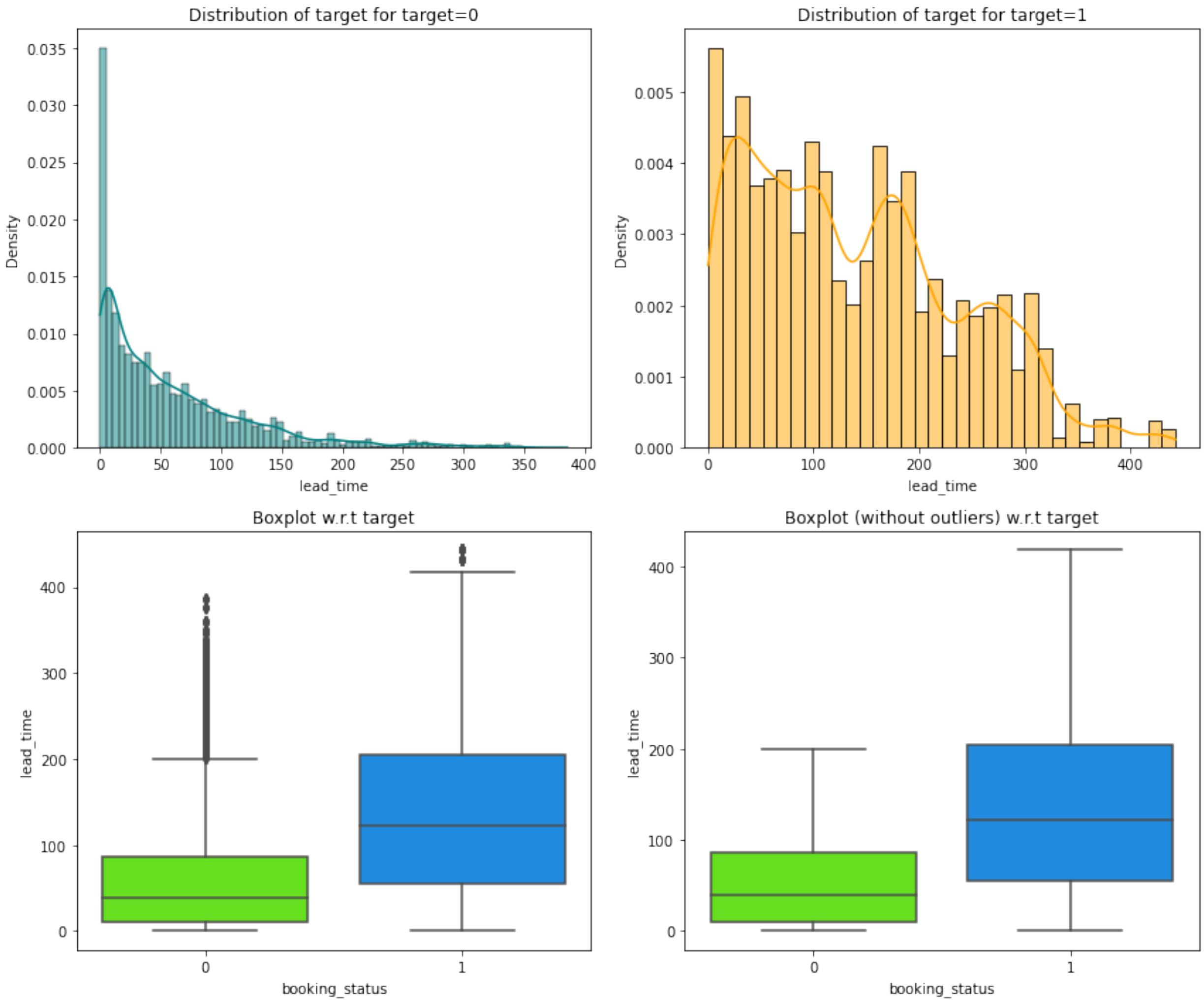

#### We saw earlier that there is a positive correlation between booking status and average price per room. Let's analyze it

In [50]:
```
distribution_plot_wrt_target(data, "avg_price_per_room", "booking_status"
```




### There is a positive correlation between booking status and lead time also. Let's analyze it further

In [51]:
```
distribution_plot_wrt_target(data, "lead_time", "booking_status") ## Comp
```




**Generally people travel with their spouse and children for vacations or other activities. Let's create a new dataframe of the customers who traveled with their families and analyze the impact on booking status.**

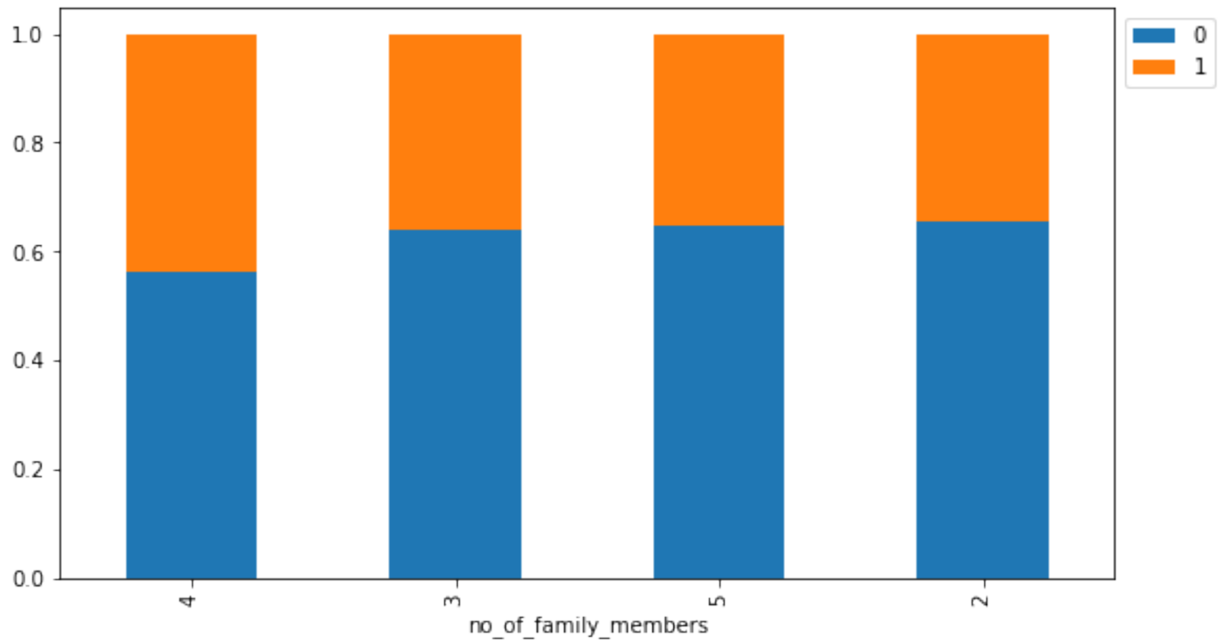
In [52]:
```
family_data = data[(data["no_of_children"] >= 0) & (data["no_of_adults"]
family_data.shape
```

Out[52]:
```
(28441, 18)
```

In [53]:
```
family_data["no_of_family_members"] = (
    family_data["no_of_adults"] + family_data["no_of_children"]
)
```

In [54]:
```
stacked_barplot(family_data,'no_of_family_members','booking_status') ## C
```

```
booking_status             0     1    All
no_of_family_members
All                    18456  9985  28441
2                      15506  8213  23719
3                       2425  1368   3793
4                        514   398    912
5                         11     6     17
------------------------------------------------------------------------
-----------------------------------------------
```




**Let's do a similar analysis for the customer who stay for at least a day at the hotel.**

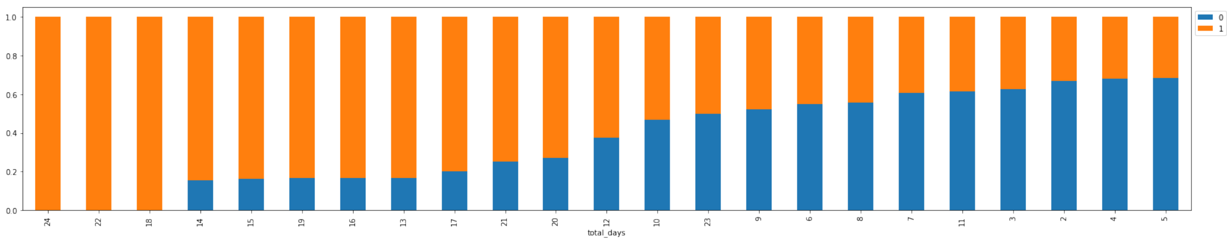
In [55]:
```
stay_data = data[(data["no_of_week_nights"] > 0) & (data["no_of_weekend_n
stay_data.shape
```

Out[55]: (17094, 18)

In [56]:
```
stay_data["total_days"] = (
    stay_data["no_of_week_nights"] + stay_data["no_of_weekend_nights"]
)
```

In [57]:
```
stacked_barplot(stay_data,"total_days","booking_status") ## Complete the
```

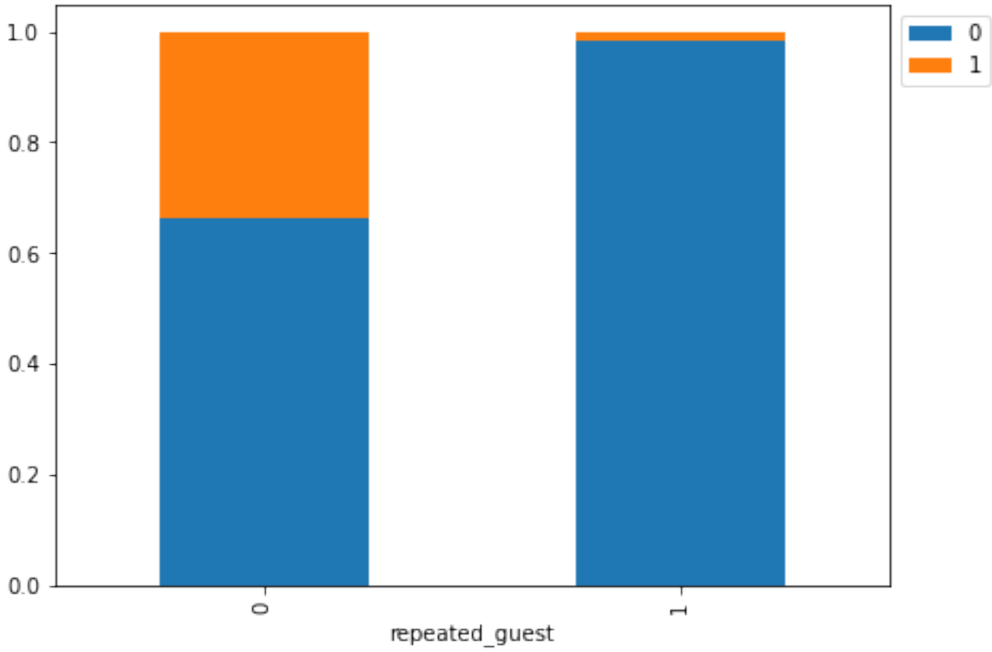
```
booking_status      0     1    All
total_days
All             10979  6115  17094
3                3689  2183   5872
4                2977  1387   4364
5                1593   738   2331
2                1301   639   1940
6                 566   465   1031
7                 590   383    973
8                 100    79    179
10                 51    58    109
9                  58    53    111
14                  5    27     32
15                  5    26     31
13                  3    15     18
12                  9    15     24
11                 24    15     39
20                  3     8     11
19                  1     5      6
16                  1     5      6
17                  1     4      5
18                  0     3      3
21                  1     3      4
22                  0     2      2
23                  1     1      2
24                  0     1      1
------------------------------------------------------------------------
----------------------------------------------
```




**Repeating guests are the guests who stay in the hotel often and are important to brand equity. Let's see what percentage of repeating guests cancel?**

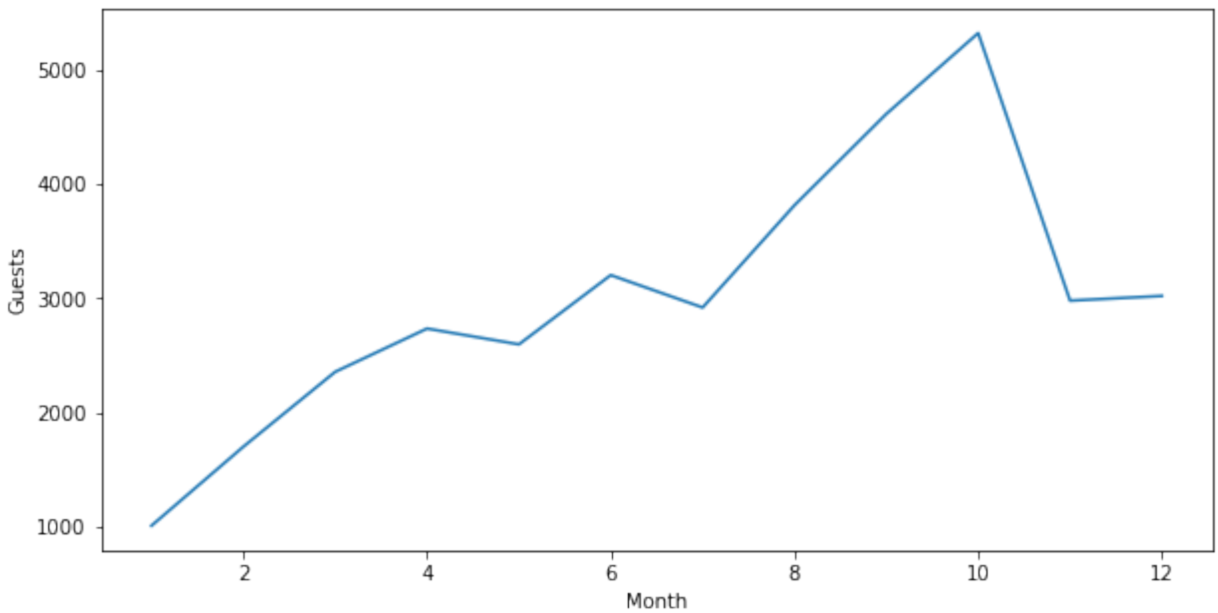
In [58]:
```
stacked_barplot(data,"repeated_guest","booking_status") ## Complete the c
```

```
booking_status      0      1    All
repeated_guest
All             24390  11885  36275
0               23476  11869  35345
1                 914     16    930
------------------------------------------------------------------------
----------------------------------------------
```




**Let's find out what are the busiest months in the hotel.**

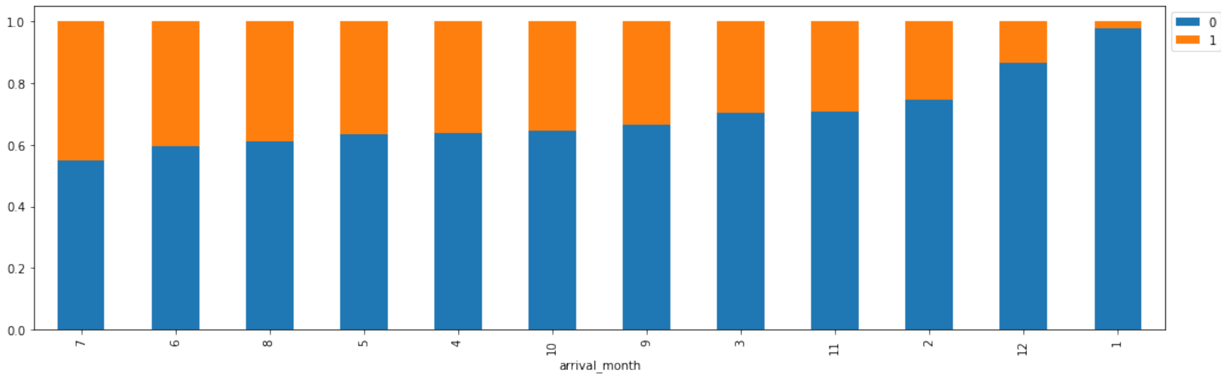
In [59]:
```
# grouping the data on arrival months and extracting the count of booking
monthly_data = data.groupby(["arrival_month"])["booking_status"].count()

# creating a dataframe with months and count of customers in each month
monthly_data = pd.DataFrame(
    {"Month": list(monthly_data.index), "Guests": list(monthly_data.value
)

# plotting the trend over different months
plt.figure(figsize=(10, 5))
sns.lineplot(data=monthly_data, x="Month", y="Guests")
plt.show()
```




**Let's check the percentage of bookings canceled in each month.**

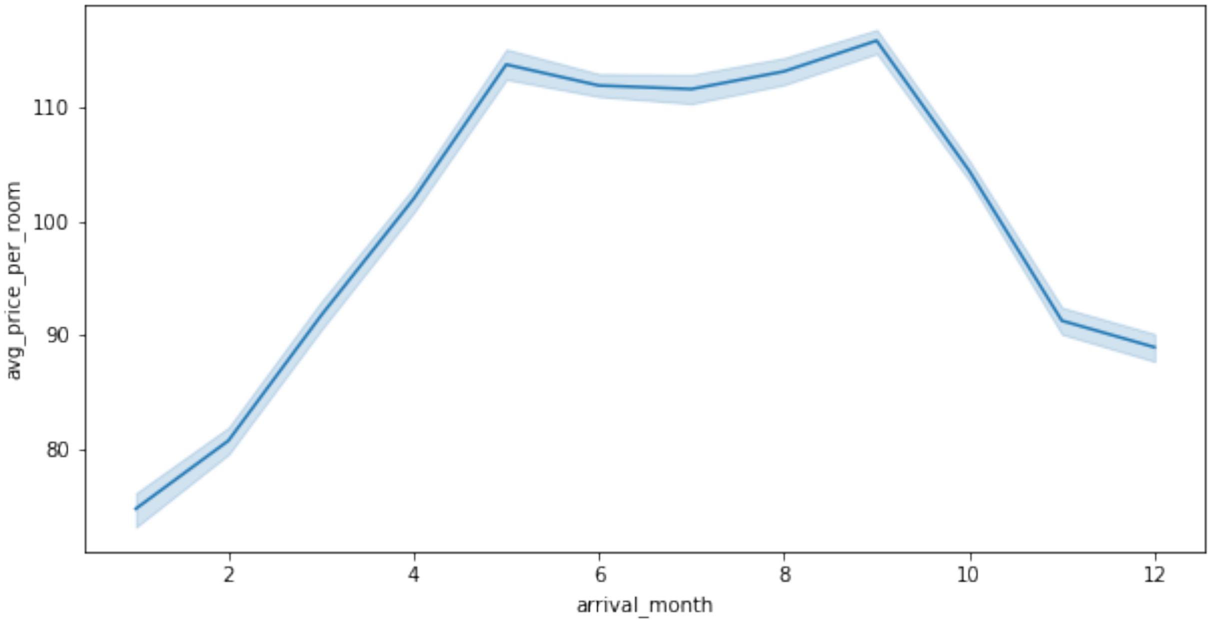
In [60]:
```
stacked_barplot(data,'arrival_month','booking_status') ## Complete the co
```

```
booking_status      0      1    All
arrival_month
All             24390  11885  36275
10               3437   1880   5317
9                3073   1538   4611
8                2325   1488   3813
7                1606   1314   2920
6                1912   1291   3203
4                1741    995   2736
5                1650    948   2598
11               2105    875   2980
3                1658    700   2358
2                1274    430   1704
12               2619    402   3021
1                 990     24   1014
------------------------------------------------------------------------
----------------------------------------------
```




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

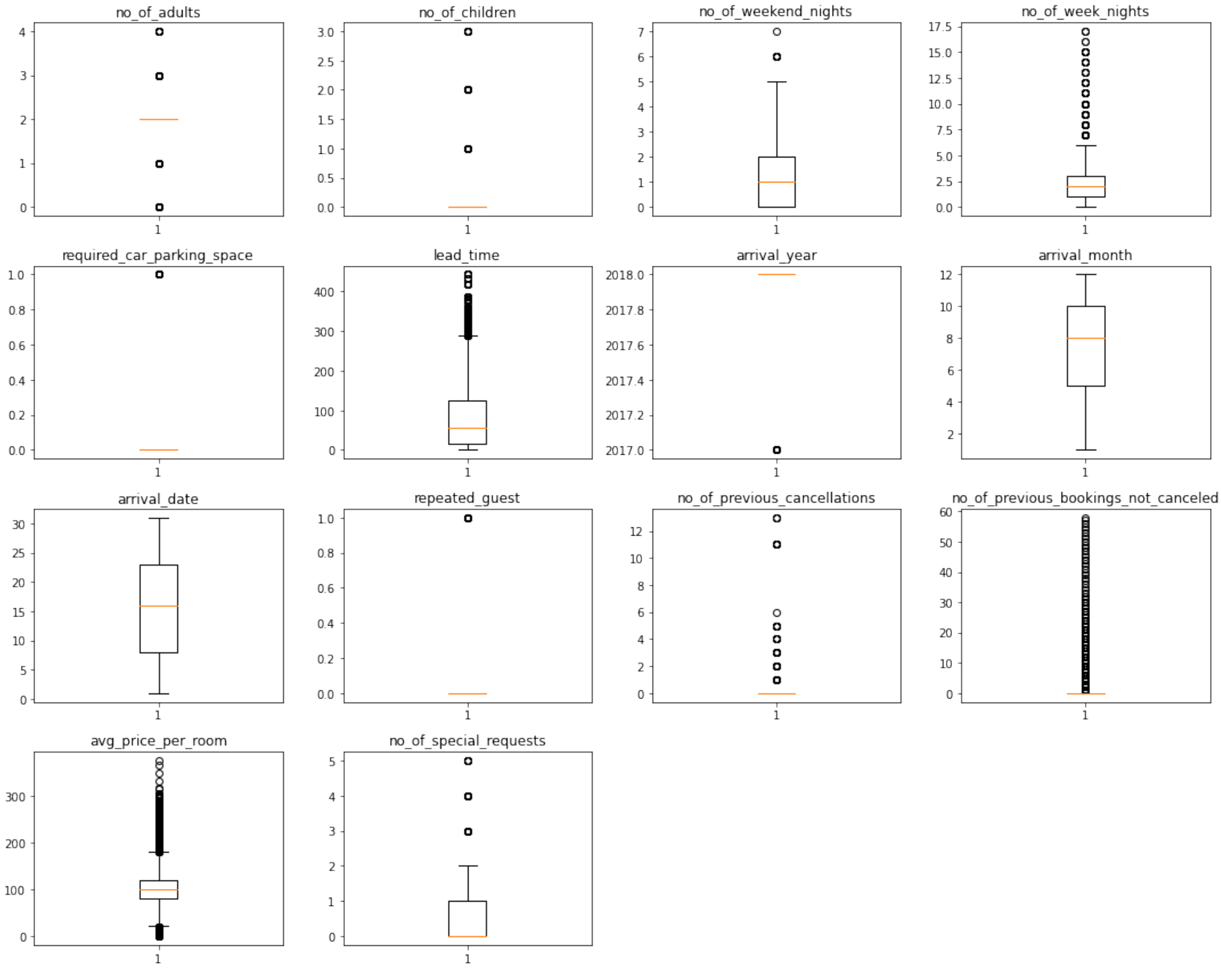
In [61]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(data=data, x='arrival_month', y='avg_price_per_room')
plt.show()
 ## Complete the code to create lineplot between average price per room a
```




## Outlier Check

- Let's check for outliers in the data.

In [62]:
```
# outlier detection using boxplot
numeric_columns = data.select_dtypes(include=np.number).columns.tolist()
# dropping booking_status
numeric_columns.remove("booking_status")

plt.figure(figsize=(15, 12))

for i, variable in enumerate(numeric_columns):
    plt.subplot(4, 4, i + 1)
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```




# Model Building

## 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 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.

### First, let's create functions to calculate different metrics and confusion matrix so that we don't have to use the same code repeatedly for each model.

- The model_performance_classification_statsmodels function will be used to check the model performance of models.
- The confusion_matrix_statsmodels function will be used to plot the confusion matrix.

In [63]:
```
# defining a function to compute different metrics to check performance o
def model_performance_classification_statsmodels(
    model, predictors, target, threshold=0.5
):
    """
    Function to compute different metrics to check classification model p

    model: classifier
    predictors: independent variables
    target: dependent variable
    threshold: threshold for classifying the observation as class 1
    """

    # checking which probabilities are greater than threshold
    pred_temp = model.predict(predictors) > threshold
    # rounding off the above values to get classes
    pred = np.round(pred_temp)

    acc = accuracy_score(target, pred)  # to compute Accuracy
    recall = recall_score(target, pred)  # to compute Recall
    precision = precision_score(target, pred)  # to compute Precision
    f1 = f1_score(target, pred)  # to compute F1-score

    # creating a dataframe of metrics
    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1":
        index=[0],
    )

    return df_perf
```

In [64]:
```
# defining a function to plot the confusion_matrix of a classification mo


def confusion_matrix_statsmodels(model, predictors, target, threshold=0.5
    """
    To plot the confusion_matrix with percentages

    model: classifier
    predictors: independent variables
    target: dependent variable
    threshold: threshold for classifying the observation as class 1
    """
    y_pred = model.predict(predictors) > threshold
    cm = confusion_matrix(target, y_pred)
    labels = np.asarray(
        [
            ["{0:0.0f}".format(item) + "\n{0:.2%}".format(item / cm.flatt
            for item in cm.flatten()
        ]
    ).reshape(2, 2)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=labels, fmt="")
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
```

## Logistic Regression (with statsmodels library)

### Data Preparation for modeling (Logistic Regression)

- 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.

In [65]:
```
X = data.drop(["booking_status"], axis=1)
Y = data["booking_status"]

# adding constant
X = sm.add_constant(X) ## Complete the code to add constant to X

X = pd.get_dummies(X,drop_first=True) ## Complete the code to create dumm

# Splitting data in train and test sets
X_train, X_test, y_train, y_test = train_test_split(
X, Y, test_size=0.3, random_state=1) ## Complete the code to split the da
```

In [66]:
```
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, 28)
Shape of test set :  (10883, 28)
Percentage of classes in training set:
0    0.67064
1    0.32936
Name: booking_status, dtype: float64
Percentage of classes in test set:
0    0.67638
1    0.32362
Name: booking_status, dtype: float64
```

## Building Logistic Regression Model

In [67]:
```
# fitting logistic regression model
logit = sm.Logit(y_train, X_train.astype(float))
lg = logit.fit() ## Complete the code to fit logistic regression

print(lg.summary()) ## Complete the code to print summary of the model
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.425090
         Iterations: 35
                           Logit Regression Results
==========================================================================
=====
Dep. Variable:          booking_status   No. Observations:
25392
Model:                           Logit   Df Residuals:
25364
Method:                            MLE   Df Model:
27
Date:                 Wed, 15 Mar 2023   Pseudo R-squ.:                   0
.3292
Time:                         18:57:50   Log-Likelihood:                 -1
0794.
converged:                       False   LL-Null:                        -1
6091.
Covariance Type:             nonrobust   LLR p-value:
0.000
==========================================================================
================================
                                           coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
--------------------------------
const                                 -922.8266    120.832     -7.637
0.000   -1159.653    -686.000
no_of_adults                             0.1137      0.038      3.019
0.003       0.040       0.188
```

```
no_of_children                          0.1580      0.062      2.544
0.011       0.036       0.280
no_of_weekend_nights                    0.1067      0.020      5.395
0.000       0.068       0.145
no_of_week_nights                       0.0397      0.012      3.235
0.001       0.016       0.064
required_car_parking_space             -1.5943      0.138    -11.565
0.000      -1.865      -1.324
lead_time                               0.0157      0.000     58.863
0.000       0.015       0.016
arrival_year                            0.4561      0.060      7.617
0.000       0.339       0.573
arrival_month                          -0.0417      0.006     -6.441
0.000      -0.054      -0.029
arrival_date                            0.0005      0.002      0.259
0.796      -0.003       0.004
repeated_guest                         -2.3472      0.617     -3.806
0.000      -3.556      -1.139
no_of_previous_cancellations            0.2664      0.086      3.108
0.002       0.098       0.434
no_of_previous_bookings_not_canceled   -0.1727      0.153     -1.131
0.258      -0.472       0.127
avg_price_per_room                      0.0188      0.001     25.396
0.000       0.017       0.020
no_of_special_requests                 -1.4689      0.030    -48.782
0.000      -1.528      -1.410
type_of_meal_plan_Meal Plan 2           0.1756      0.067      2.636
0.008       0.045       0.306
type_of_meal_plan_Meal Plan 3          17.3584   3987.836      0.004
0.997   -7798.656    7833.373
type_of_meal_plan_Not Selected          0.2784      0.053      5.247
0.000       0.174       0.382
room_type_reserved_Room_Type 2         -0.3605      0.131     -2.748
0.006      -0.618      -0.103
room_type_reserved_Room_Type 3         -0.0012      1.310     -0.001
0.999      -2.568       2.566
room_type_reserved_Room_Type 4         -0.2823      0.053     -5.304
0.000      -0.387      -0.178
room_type_reserved_Room_Type 5         -0.7189      0.209     -3.438
0.001      -1.129      -0.309
room_type_reserved_Room_Type 6         -0.9501      0.151     -6.274
0.000      -1.247      -0.653
room_type_reserved_Room_Type 7         -1.4003      0.294     -4.770
0.000      -1.976      -0.825
market_segment_type_Complementary     -40.5975   5.65e+05  -7.19e-05
1.000   -1.11e+06    1.11e+06
market_segment_type_Corporate          -1.1924      0.266     -4.483
0.000      -1.714      -0.671
market_segment_type_Offline            -2.1946      0.255     -8.621
0.000      -2.694      -1.696
market_segment_type_Online             -0.3995      0.251     -1.590
0.112      -0.892       0.093
==========================================================================
===============================
```

In [68]:
```
print("Training performance:")
model_performance_classification_statsmodels(lg, X_train, y_train)
```

```
Training performance:
```

Out[68]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.80600 | 0.63410 | 0.73971 | 0.68285 |

## Multicollinearity

In [69]:
```
# we will define a function to check VIF
def checking_vif(predictors):
    vif = pd.DataFrame()
    vif["feature"] = predictors.columns

    # calculating VIF for each feature
    vif["VIF"] = [
        variance_inflation_factor(predictors.values, i)
        for i in range(len(predictors.columns))
    ]
    return vif
```

In [70]:
```
checking_vif(X_train)
```

Out[70]:

| | feature | VIF |
|---|---|---|
| 0 | const | 39497686.20788 |
| 1 | no_of_adults | 1.35113 |
| 2 | no_of_children | 2.09358 |
| 3 | no_of_weekend_nights | 1.06948 |
| 4 | no_of_week_nights | 1.09571 |
| 5 | required_car_parking_space | 1.03997 |
| 6 | lead_time | 1.39517 |
| 7 | arrival_year | 1.43190 |
| 8 | arrival_month | 1.27633 |
| 9 | arrival_date | 1.00679 |
| 10 | repeated_guest | 1.78358 |
| 11 | no_of_previous_cancellations | 1.39569 |
| 12 | no_of_previous_bookings_not_canceled | 1.65200 |
| 13 | avg_price_per_room | 2.06860 |
| 14 | no_of_special_requests | 1.24798 |
| 15 | type_of_meal_plan_Meal Plan 2 | 1.27328 |
| 16 | type_of_meal_plan_Meal Plan 3 | 1.02526 |
| 17 | type_of_meal_plan_Not Selected | 1.27306 |
| 18 | room_type_reserved_Room_Type 2 | 1.10595 |
| 19 | room_type_reserved_Room_Type 3 | 1.00330 |
| 20 | room_type_reserved_Room_Type 4 | 1.36361 |
| 21 | room_type_reserved_Room_Type 5 | 1.02800 |
| 22 | room_type_reserved_Room_Type 6 | 2.05614 |
| 23 | room_type_reserved_Room_Type 7 | 1.11816 |
| 24 | market_segment_type_Complementary | 4.50276 |
| 25 | market_segment_type_Corporate | 16.92829 |
| 26 | market_segment_type_Offline | 64.11564 |
| 27 | market_segment_type_Online | 71.18026 |

### Dropping high p-value variables

- We will drop the predictor variables having a p-value greater than 0.05 as they do not significantly impact the target variable.
- But sometimes p-values change after dropping a variable. So, we'll not drop all variables at once.
- Instead, we will do the following:
  - Build a model, check the p-values of the variables, and drop the column with the highest p-value.
  - Create a new model without the dropped feature, check the p-values of the variables, and drop the column with the highest p-value.
  - Repeat the above two steps till there are no columns with p-value > 0.05.

The above process can also be done manually by picking one variable at a time that has a high p-value, dropping it, and building a model again. But that might be a little tedious and using a loop will be more efficient.

In [71]:
```
# initial list of columns
cols = X_train.columns.tolist()

# setting an initial max p-value
max_p_value = 1

while len(cols) > 0:
    # defining the train set
    x_train_aux = X_train[cols]

    # fitting the model
    model = sm.Logit(y_train, x_train_aux).fit(disp=False)

    # getting the p-values and the maximum p-value
    p_values = model.pvalues
    max_p_value = max(p_values)

    # name of the variable with maximum p-value
    feature_with_p_max = p_values.idxmax()

    if max_p_value > 0.05:
        cols.remove(feature_with_p_max)
    else:
        break

selected_features = cols
print(selected_features)
```

```
['const', 'no_of_adults', 'no_of_children', 'no_of_weekend_nights', 'no_o
f_week_nights', 'required_car_parking_space', 'lead_time', 'arrival_year'
, 'arrival_month', 'repeated_guest', 'no_of_previous_cancellations', 'avg
_price_per_room', 'no_of_special_requests', 'type_of_meal_plan_Meal Plan
2', 'type_of_meal_plan_Not Selected', 'room_type_reserved_Room_Type 2', '
room_type_reserved_Room_Type 4', 'room_type_reserved_Room_Type 5', 'room_
type_reserved_Room_Type 6', 'room_type_reserved_Room_Type 7', 'market_seg
ment_type_Corporate', 'market_segment_type_Offline']
```

In [72]:

```
X_train1 = X_train[selected_features]
X_test1 = X_test[selected_features]
```

In [73]:

```
logit1 = sm.Logit(y_train,X_train1) ## Complete the code to train logisti
lg1 = logit1.fit() ## Complete the code to fit logistic regression
print(lg1.summary()) ## Complete the code to print summary of the model
```

```
Optimization terminated successfully.
         Current function value: 0.425731
         Iterations 11
                           Logit Regression Results
==========================================================================
=====
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25370
Method:                           MLE   Df Model:
21
Date:                Wed, 15 Mar 2023   Pseudo R-squ.:                   0
.3282
Time:                        18:57:51   Log-Likelihood:                 -1
0810.
converged:                       True   LL-Null:                        -1
6091.
Covariance Type:            nonrobust   LLR p-value:
0.000
==========================================================================
=========================
                                   coef    std err          z      P>|z
|      [0.025      0.975]
--------------------------------------------------------------------------
-------------------------
const                         -915.6391    120.471     -7.600      0.00
0   -1151.758    -679.520
no_of_adults                     0.1088      0.037      2.914      0.00
4       0.036       0.182
no_of_children                   0.1531      0.062      2.470      0.01
4       0.032       0.275
no_of_weekend_nights             0.1086      0.020      5.498      0.00
0       0.070       0.147
no_of_week_nights                0.0417      0.012      3.399      0.00
1       0.018       0.066
required_car_parking_space      -1.5947      0.138    -11.564      0.00
0      -1.865      -1.324
lead_time                        0.0157      0.000     59.213      0.00
0       0.015       0.016
arrival_year                     0.4523      0.060      7.576      0.00
```

```
0        0.335        0.569
arrival_month                              -0.0425      0.006     -6.591      0.00
0       -0.055       -0.030
repeated_guest                             -2.7367      0.557     -4.916      0.00
0       -3.828       -1.646
no_of_previous_cancellations                0.2288      0.077      2.983      0.00
3        0.078        0.379
avg_price_per_room                          0.0192      0.001     26.336      0.00
0        0.018        0.021
no_of_special_requests                     -1.4698      0.030    -48.884      0.00
0       -1.529       -1.411
type_of_meal_plan_Meal Plan 2               0.1642      0.067      2.469      0.01
4        0.034        0.295
type_of_meal_plan_Not Selected              0.2860      0.053      5.406      0.00
0        0.182        0.390
room_type_reserved_Room_Type 2             -0.3552      0.131     -2.709      0.00
7       -0.612       -0.098
room_type_reserved_Room_Type 4             -0.2828      0.053     -5.330      0.00
0       -0.387       -0.179
room_type_reserved_Room_Type 5             -0.7364      0.208     -3.535      0.00
0       -1.145       -0.328
room_type_reserved_Room_Type 6             -0.9682      0.151     -6.403      0.00
0       -1.265       -0.672
room_type_reserved_Room_Type 7             -1.4343      0.293     -4.892      0.00
0       -2.009       -0.860
market_segment_type_Corporate              -0.7913      0.103     -7.692      0.00
0       -0.993       -0.590
market_segment_type_Offline                -1.7854      0.052    -34.363      0.00
0       -1.887       -1.684
==========================================================================
=========================
```

In [74]:

```
print("Training performance:")
model_performance_classification_statsmodels(lg1,X_train1,y_train) ## Com
```

Training performance:

Out[74]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.80545 | 0.63267 | 0.73907 | 0.68174 |

## Converting coefficients to odds

- The coefficients of the logistic regression model are in terms of log(odd), to find the odds we have to take the exponential of the coefficients.
- Therefore, **odds = exp(b)**
- The percentage change in odds is given as **odds = (exp(b) - 1) * 100**

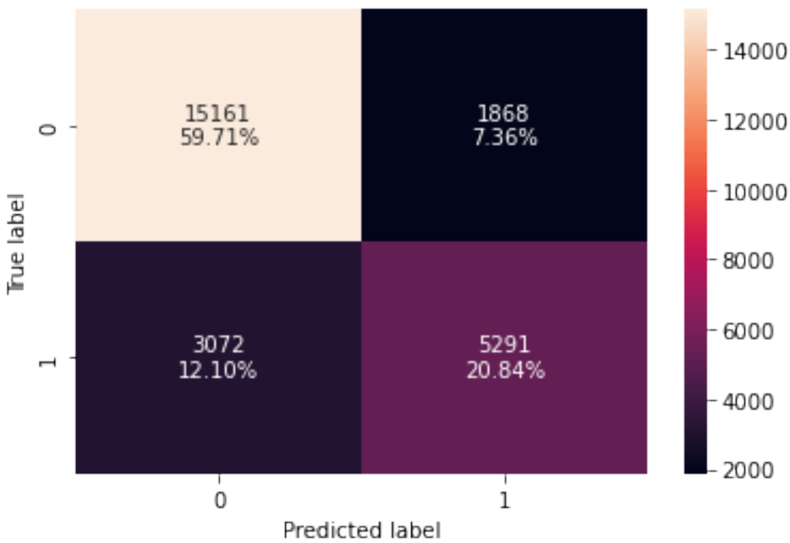
In [75]:
```
# converting coefficients to odds
odds = np.exp(lg1.params)

# finding the percentage change
perc_change_odds = (np.exp(lg1.params) - 1) * 100

# removing limit from number of columns to display
pd.set_option("display.max_columns", None)

# adding the odds to a dataframe
pd.DataFrame({"Odds": odds, "Change_odd%": perc_change_odds}, index=X_tra
```

Out[75]:

| | const | no_of_adults | no_of_children | no_of_weekend_nights | no_of_w |
|---|---|---|---|---|---|
| **Odds** | 0.00000 | 1.11491 | 1.16546 | 1.11470 | |
| **Change_odd%** | -100.00000 | 11.49096 | 16.54593 | 11.46966 | |

## Checking model performance on the training set

In [76]:
```
# creating confusion matrix
confusion_matrix_statsmodels(lg1, X_train1, y_train)
```




In [77]:
```
print("Training performance:")
log_reg_model_train_perf = model_performance_classification_statsmodels(l
log_reg_model_train_perf
```

```
Training performance:
```

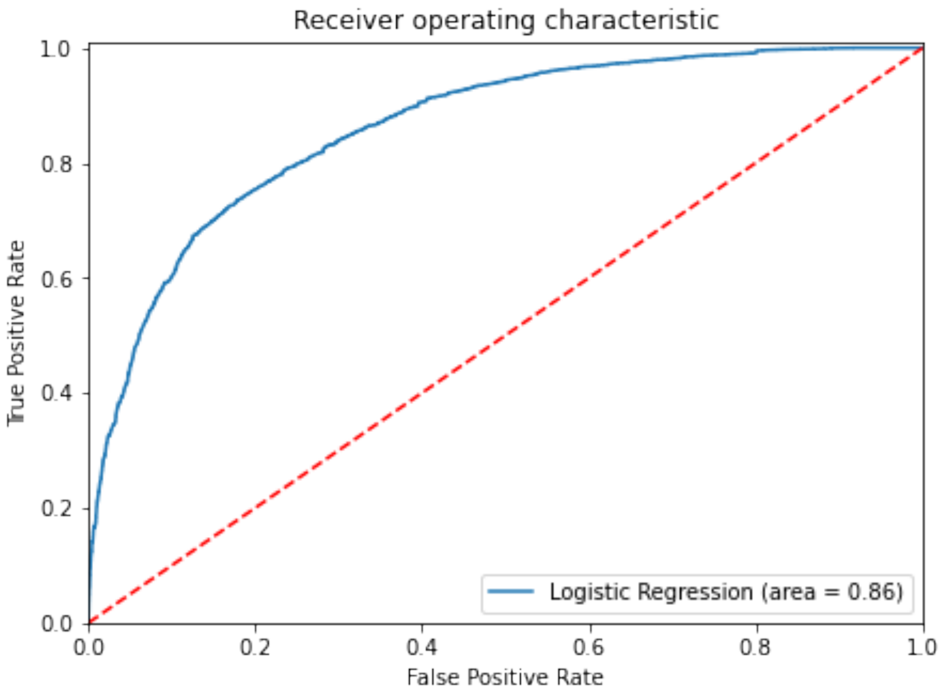
Out[77]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.80545 | 0.63267 | 0.73907 | 0.68174 |

## ROC-AUC

- ROC-AUC on training set

In [78]:
```
logit_roc_auc_train = roc_auc_score(y_train, lg1.predict(X_train1))
fpr, tpr, thresholds = roc_curve(y_train, lg1.predict(X_train1))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.01])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```




## Model Performance Improvement

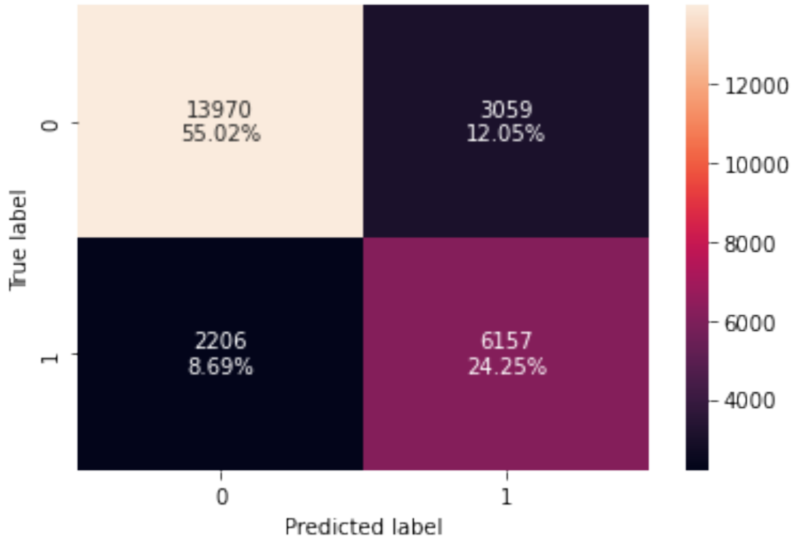
- Let's see if the recall score can be improved further, by changing the model threshold using AUC-ROC Curve.

## Optimal threshold using AUC-ROC curve

In [79]:
```
# Optimal threshold as per AUC-ROC curve
# The optimal cut off would be where tpr is high and fpr is low
fpr, tpr, thresholds = roc_curve(y_train, lg1.predict(X_train1))

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold_auc_roc = thresholds[optimal_idx]
print(optimal_threshold_auc_roc)
```

```
0.37005225587078305
```

In [80]:
```
# creating confusion matrix
confusion_matrix_statsmodels(
    lg1, X_train1,y_train, threshold=optimal_threshold_auc_roc
) ## Complete the code to create the confusion matrix for X_train1 and y_
```




In [81]:
```
# checking model performance for this model
log_reg_model_train_perf_threshold_auc_roc = model_performance_classifica
    lg1, X_train1, y_train, threshold=optimal_threshold_auc_roc
)
print("Training performance:")
log_reg_model_train_perf_threshold_auc_roc
```

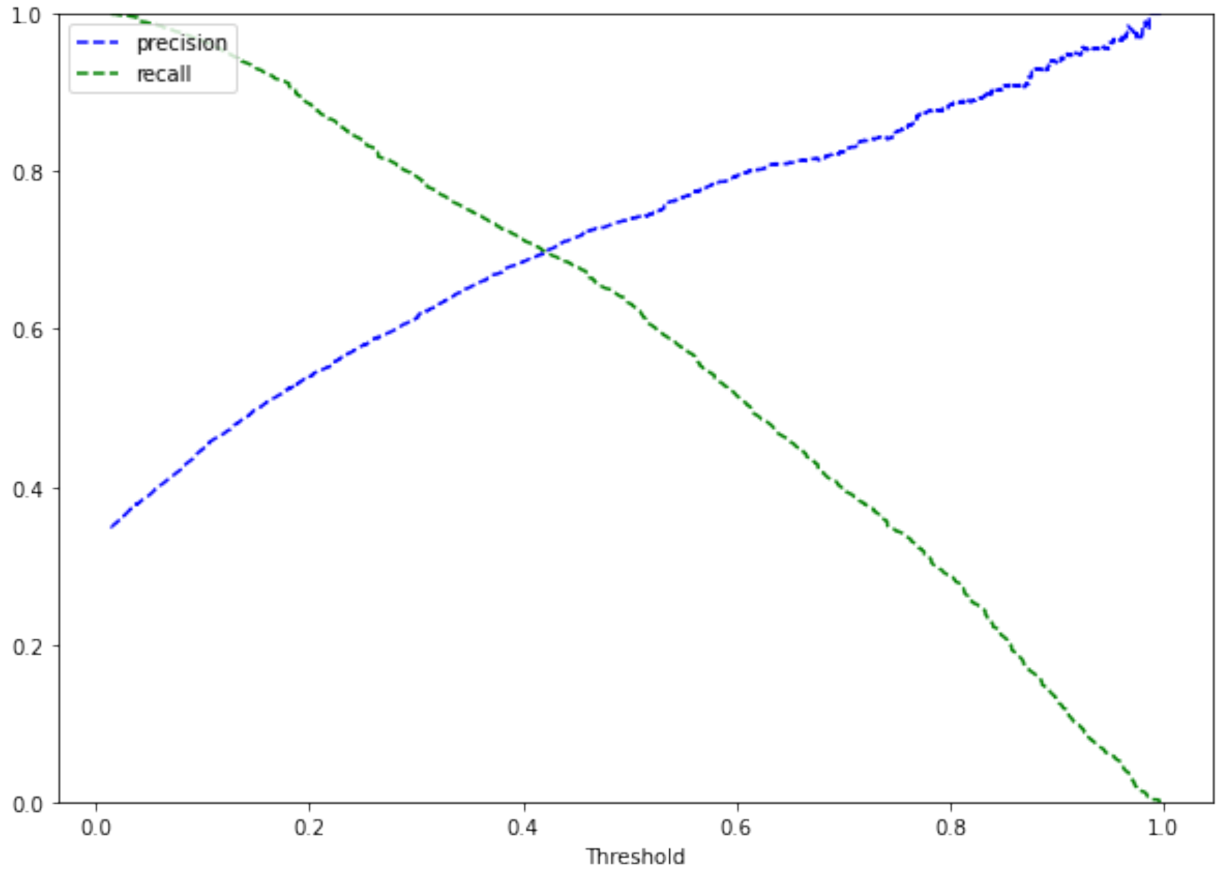
```
Training performance:
```

Out[81]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.79265 | 0.73622 | 0.66808 | 0.70049 |

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

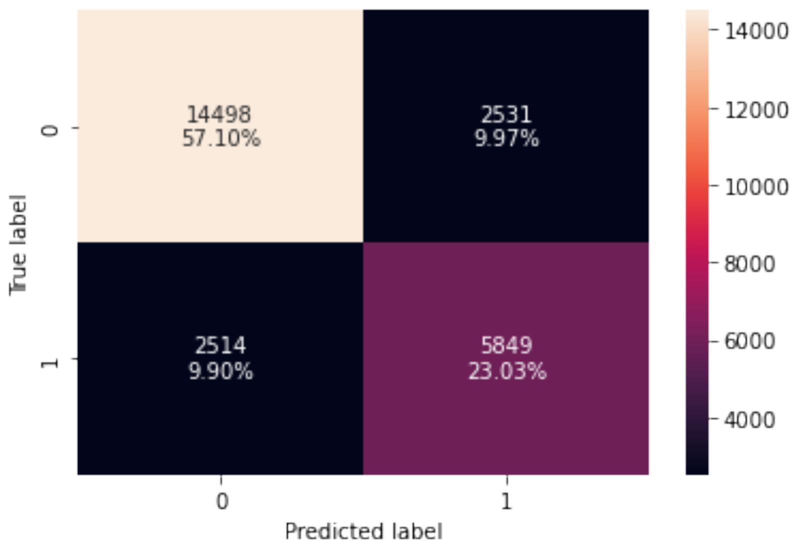
In [82]:
```
y_scores = lg1.predict(X_train1)
prec, rec, tre = precision_recall_curve(y_train, y_scores,)


def plot_prec_recall_vs_tresh(precisions, recalls, thresholds):
    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.figure(figsize=(10, 7))
plot_prec_recall_vs_tresh(prec, rec, tre)
plt.show()
```




In [83]:
```
# setting the threshold
optimal_threshold_curve = 0.42
```

### Checking model performance on training set

In [84]:
```
# creating confusion matrix
confusion_matrix_statsmodels(
    lg1, X_train1, y_train, threshold=optimal_threshold_curve)
 ## Complete the code to create the confusion matrix for X_train1 and y_t
```



In [85]:
```
log_reg_model_train_perf_threshold_curve = model_performance_classificati
    lg1, X_train1, y_train, threshold=optimal_threshold_curve
)
print("Training performance:")
log_reg_model_train_perf_threshold_curve
```

```
Training performance:
```

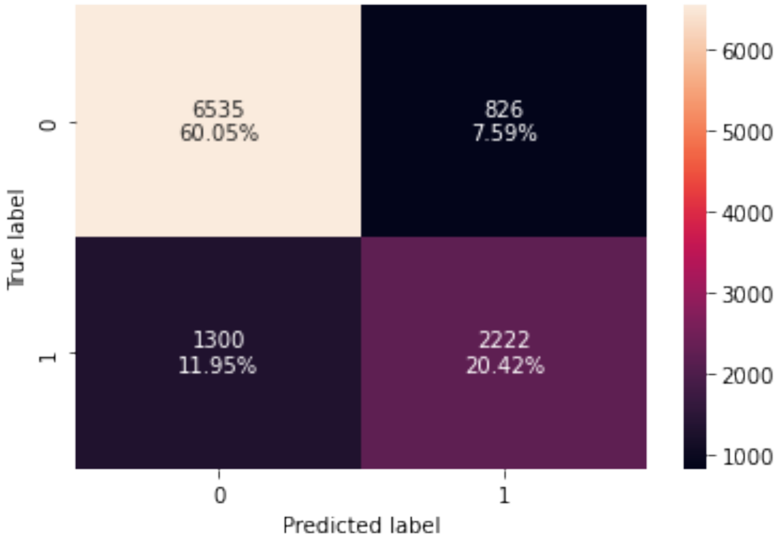
Out[85]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.80132 | 0.69939 | 0.69797 | 0.69868 |

## Let's check the performance on the test set

### Using model with default threshold

In [86]:
```
# creating confusion matrix
confusion_matrix_statsmodels(lg1, X_test1, y_test) ## Complete the code t
```




In [87]:
```
log_reg_model_test_perf = model_performance_classification_statsmodels(lg

print("Test performance:")
log_reg_model_test_perf
```

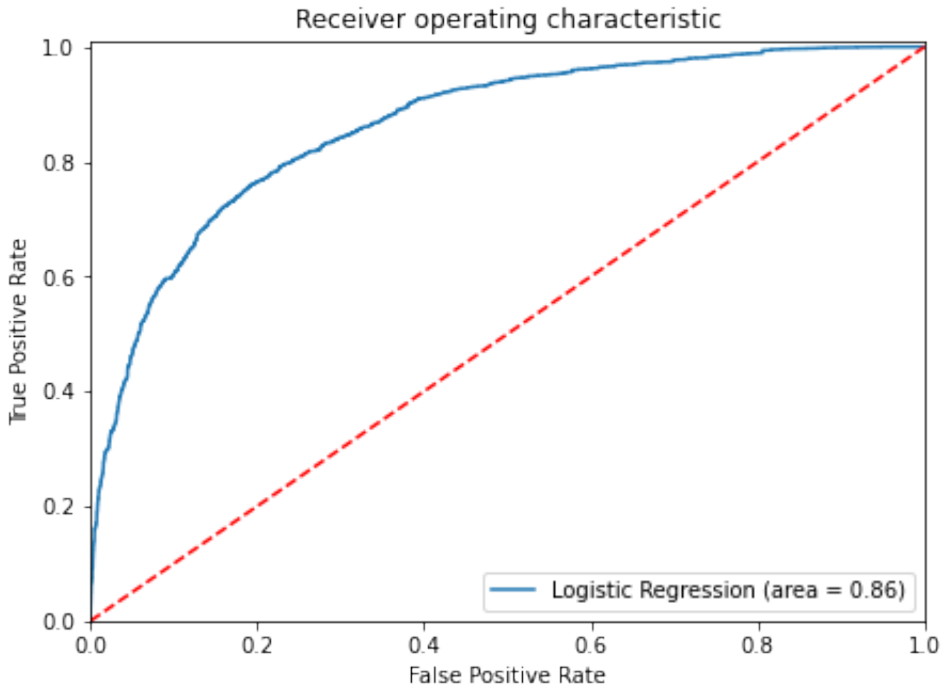
```
Test performance:
```

Out[87]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.80465 | 0.63089 | 0.72900 | 0.67641 |

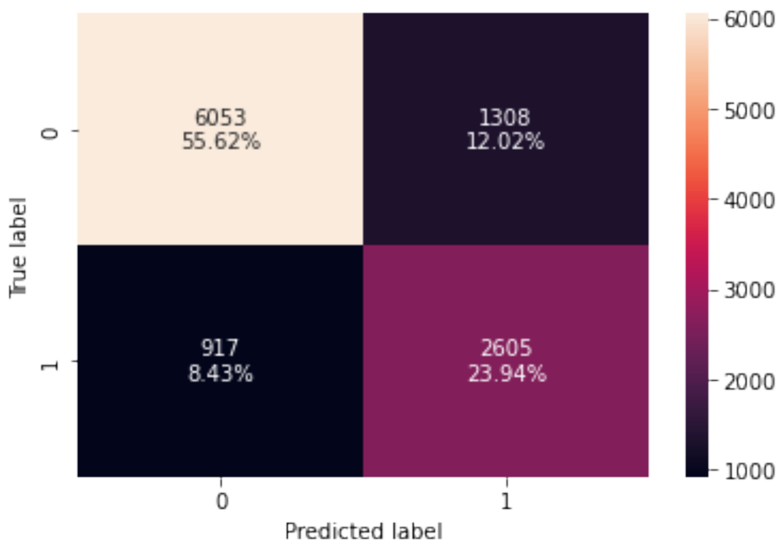
- ROC curve on test set

In [88]:
```
logit_roc_auc_train = roc_auc_score(y_test, lg1.predict(X_test1))
fpr, tpr, thresholds = roc_curve(y_test, lg1.predict(X_test1))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.01])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```




### Using model with threshold=0.37

In [89]:
```
# creating confusion matrix
confusion_matrix_statsmodels(lg1, X_test1, y_test, threshold=0.37) ## Com
```



In [90]:
```
# checking model performance for this model
log_reg_model_test_perf_threshold_auc_roc = model_performance_classificat
    lg1, X_test1, y_test, threshold=optimal_threshold_auc_roc
)
print("Test performance:")
log_reg_model_test_perf_threshold_auc_roc
```

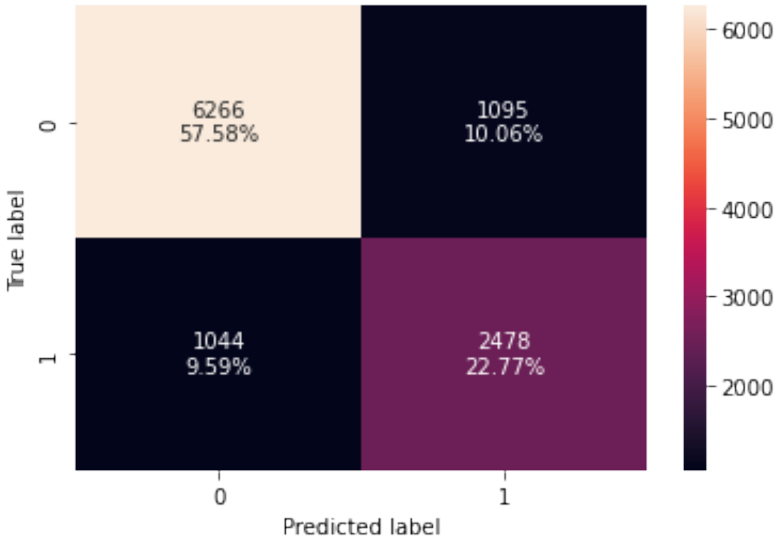
Test performance:

Out[90]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.79555 | 0.73964 | 0.66573 | 0.70074 |

### Using model with threshold = 0.42

In [91]:
```
# creating confusion matrix
confusion_matrix_statsmodels(lg1, X_test1, y_test, threshold=0.42) ## Com
```




In [92]:
```
log_reg_model_test_perf_threshold_curve = model_performance_classificatio
    lg1, X_test1, y_test, threshold=optimal_threshold_curve
)
print("Test performance:")
log_reg_model_test_perf_threshold_curve
```

Test performance:

Out[92]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.80345 | 0.70358 | 0.69353 | 0.69852 |

## Model performance summary

In [93]:
```
# training performance comparison

models_train_comp_df = pd.concat(
    [
        log_reg_model_train_perf.T,
        log_reg_model_train_perf_threshold_auc_roc.T,
        log_reg_model_train_perf_threshold_curve.T,
    ],
    axis=1,
)
models_train_comp_df.columns = [
    "Logistic Regression-default Threshold",
    "Logistic Regression-0.37 Threshold",
    "Logistic Regression-0.42 Threshold",
]

print("Training performance comparison:")
models_train_comp_df
```

Training performance comparison:

Out[93]:

| | Logistic Regression-default Threshold | Logistic Regression-0.37 Threshold | Logistic Regression-0.42 Threshold |
|---|---|---|---|
| **Accuracy** | 0.80545 | 0.79265 | 0.80132 |
| **Recall** | 0.63267 | 0.73622 | 0.69939 |
| **Precision** | 0.73907 | 0.66808 | 0.69797 |
| **F1** | 0.68174 | 0.70049 | 0.69868 |

In [94]:
```
# test performance comparison

models_test_comp_df = pd.concat(
    [
        log_reg_model_test_perf.T,
        log_reg_model_test_perf_threshold_auc_roc.T,
        log_reg_model_test_perf_threshold_curve.T,
    ],
    axis=1,
)
models_test_comp_df.columns = [     "Logistic Regression-default Threshold

print("Test performance comparison:")
models_test_comp_df## Complete the code to compare test performance
```

Test performance comparison:

Out[94]:

| | Logistic Regression-default Threshold | Logistic Regression-0.37 Threshold | Logistic Regression-0.42 Threshold |
|---|---|---|---|
| **Accuracy** | 0.80465 | 0.79555 | 0.80345 |
| **Recall** | 0.63089 | 0.73964 | 0.70358 |
| **Precision** | 0.72900 | 0.66573 | 0.69353 |
| **F1** | 0.67641 | 0.70074 | 0.69852 |

# Decision Tree

## Data Preparation for modeling (Decision Tree)

- 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.

In [95]:

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

X = pd.get_dummies(X, drop_first=True) ## Complete the code to create dum

# Splitting data in train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
```

In [96]:

```
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    0.67064
1    0.32936
Name: booking_status, dtype: float64
Percentage of classes in test set:
0    0.67638
1    0.32362
Name: booking_status, dtype: float64
```

## First, let's create functions to calculate different metrics and confusion matrix so that we don't have to use the same code repeatedly for each model.

- The model_performance_classification_sklearn function will be used to check the model performance of models.
- The confusion_matrix_sklearnfunction will be used to plot the confusion matrix.

In [97]:
```
# defining a function to compute different metrics to check performance o
def model_performance_classification_sklearn(model, predictors, target):
    """
    Function to compute different metrics to check classification model p

    model: classifier
    predictors: independent variables
    target: dependent variable
    """

    # predicting using the independent variables
    pred = model.predict(predictors)

    acc = accuracy_score(target, pred)  # to compute Accuracy
    recall = recall_score(target, pred)  # to compute Recall
    precision = precision_score(target, pred)  # to compute Precision
    f1 = f1_score(target, pred)  # to compute F1-score

    # creating a dataframe of metrics
    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision, "F1":
        index=[0],
    )

    return df_perf
```

In [98]:
```
def confusion_matrix_sklearn(model, predictors, target):
    """
    To plot the confusion_matrix with percentages

    model: classifier
    predictors: independent variables
    target: dependent variable
    """
    y_pred = model.predict(predictors)
    cm = confusion_matrix(target, y_pred)
    labels = np.asarray(
        [
            ["{0:0.0f}".format(item) + "\n{0:.2%}".format(item / cm.flatt
            for item in cm.flatten()
        ]
    ).reshape(2, 2)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=labels, fmt="")
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
```

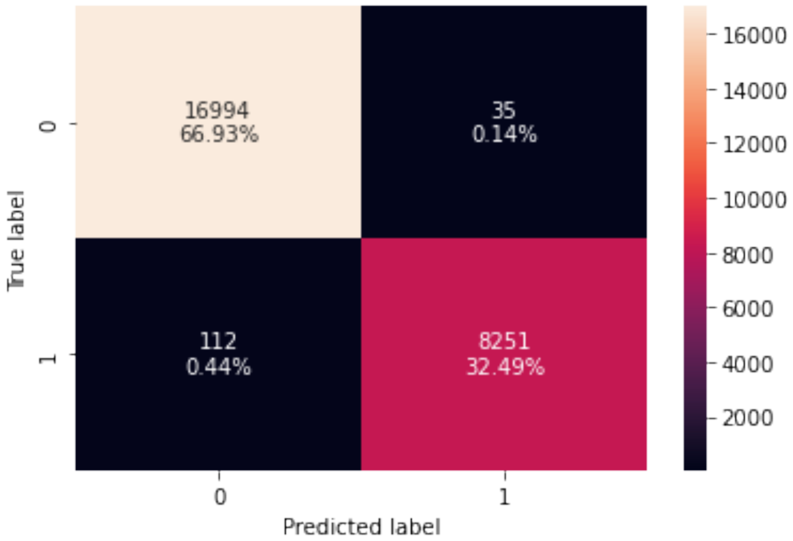
## Building Decision Tree Model

In [99]:
```
model = DecisionTreeClassifier(random_state=1)
model.fit(X_train,y_train) ## Complete the code to fit decision tree on t
```

Out[99]:
```
DecisionTreeClassifier(random_state=1)
```

## Checking model performance on training set

In [100…
```
confusion_matrix_sklearn(model, X_train, y_train) ## Complete the code to
```



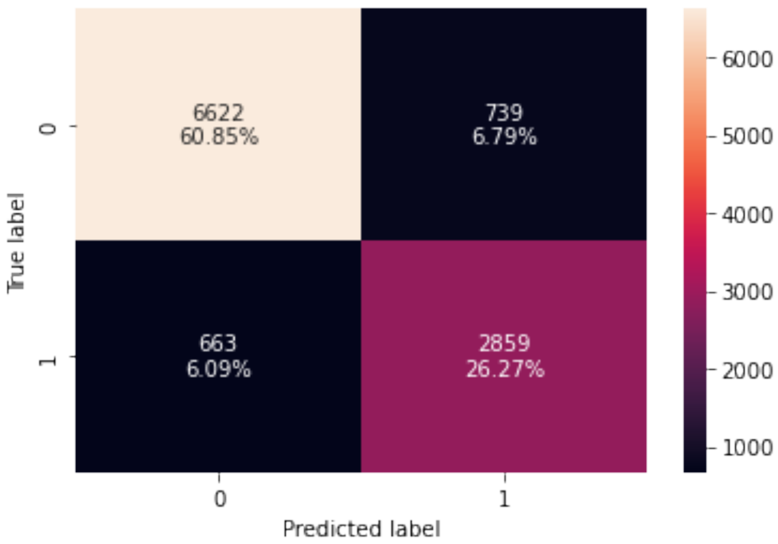
In [101…
```
decision_tree_perf_train = model_performance_classification_sklearn(
    model, X_train, y_train
)
decision_tree_perf_train
```

Out[101]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.99421 | 0.98661 | 0.99578 | 0.99117 |

## Checking model performance on test set

In [102…
```
confusion_matrix_sklearn( model,X_test, y_test) ## Complete the code to c
```



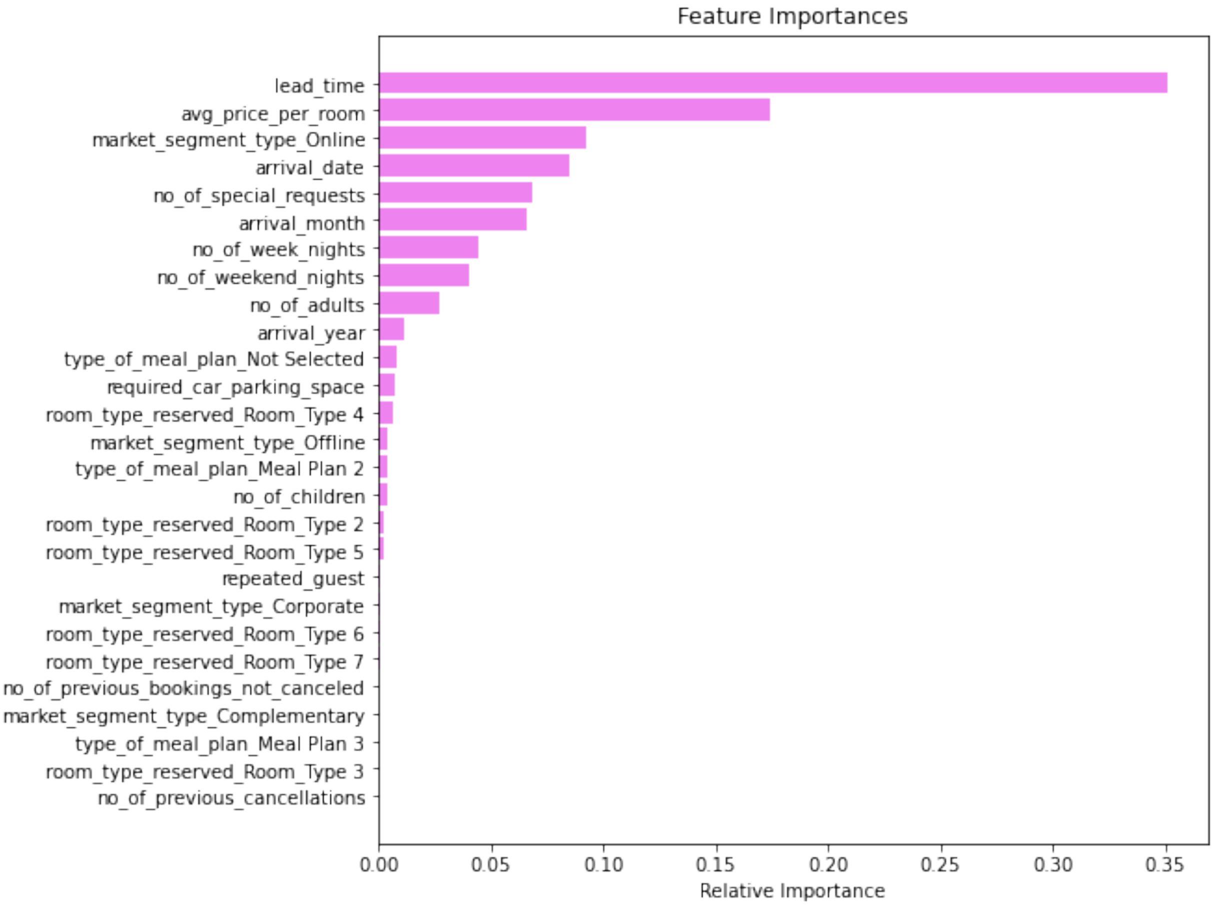

In [103…
```
decision_tree_perf_test = model_performance_classification_sklearn(model,
decision_tree_perf_test
```

Out[103]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.87118 | 0.81175 | 0.79461 | 0.80309 |

### Before pruning the tree let's check the important features.

In [104…
```
feature_names = list(X_train.columns)
importances = model.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
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```




## Pruning the tree

### Pre-Pruning

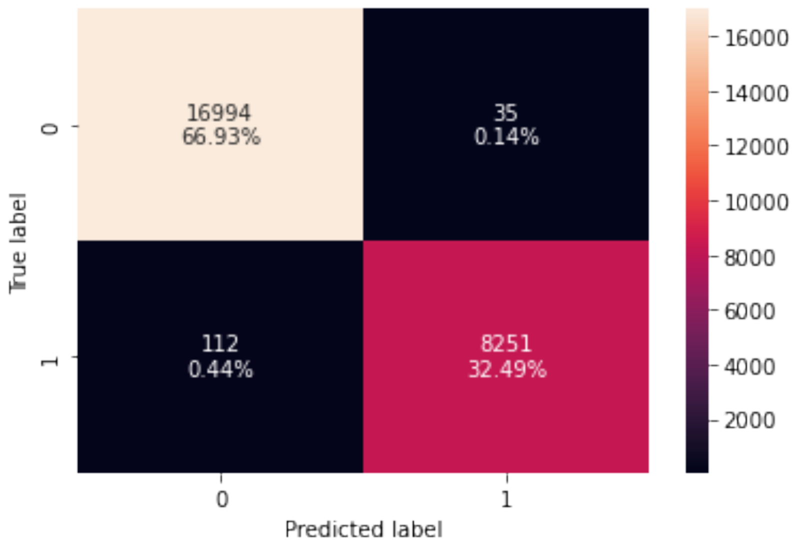
In [105…

```
# Choose the type of classifier.
estimator = DecisionTreeClassifier(random_state=1, class_weight="balanced

# 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],
}

# Type of scoring used to compare parameter combinations
acc_scorer = make_scorer(f1_score)

# Run the grid search
grid_obj = GridSearchCV(estimator, parameters, scoring=acc_scorer, cv=5)
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[105]:
```
DecisionTreeClassifier(class_weight='balanced', max_depth=6, max_leaf_no
des=50,
                       min_samples_split=10, random_state=1)
```

## Checking performance on training set

In [106…
```
confusion_matrix_sklearn(model, X_train, y_train) ## Complete the code to
```



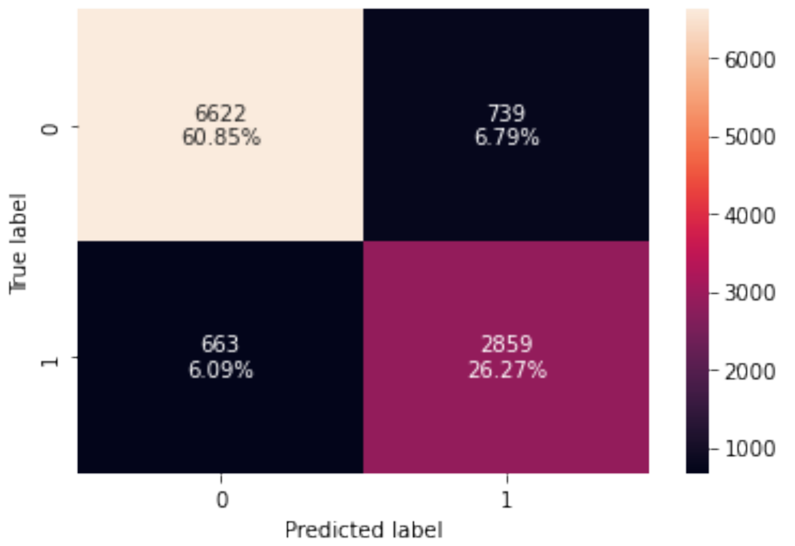

In [107…
```
decision_tree_tune_perf_train = model_performance_classification_sklearn(
decision_tree_tune_perf_train
```

Out[107]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| 0 | 0.99421 | 0.98661 | 0.99578 | 0.99117 |

## Checking performance on test set

In [108…
```
confusion_matrix_sklearn(model,X_test,y_test) ## Complete the code to cre
```



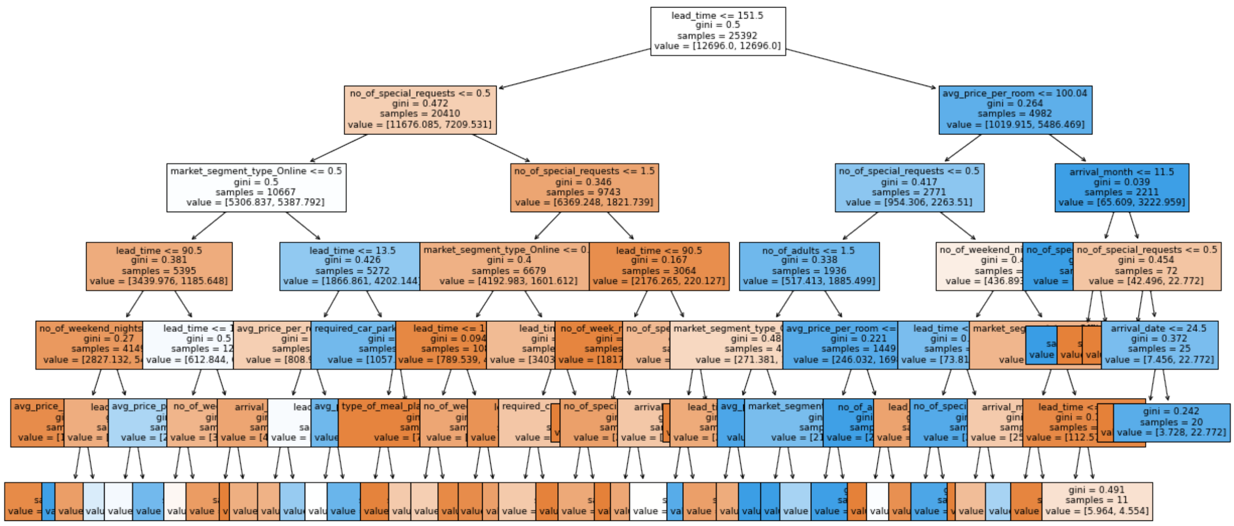
In [109…
```
decision_tree_tune_perf_test = model_performance_classification_sklearn(m
decision_tree_tune_perf_test
```

Out[109]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| 0 | 0.87118 | 0.81175 | 0.79461 | 0.80309 |

## Visualizing the Decision Tree

In [110…
```
plt.figure(figsize=(20, 10))
out = tree.plot_tree(
    estimator,
    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 missi
for o in out:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor("black")
        arrow.set_linewidth(1)
plt.show()
```




In [111…
```
# Text report showing the rules of a decision tree -
print(tree.export_text(estimator, feature_names=feature_names, show_weigh
```

```
|--- lead_time <= 151.50
|   |--- no_of_special_requests <= 0.50
|   |   |--- market_segment_type_Online <= 0.50
|   |   |   |--- lead_time <= 90.50
|   |   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |   |--- avg_price_per_room <= 196.50
|   |   |   |   |   |   |--- weights: [1736.39, 133.59] class: 0
|   |   |   |   |   |--- avg_price_per_room >  196.50
```

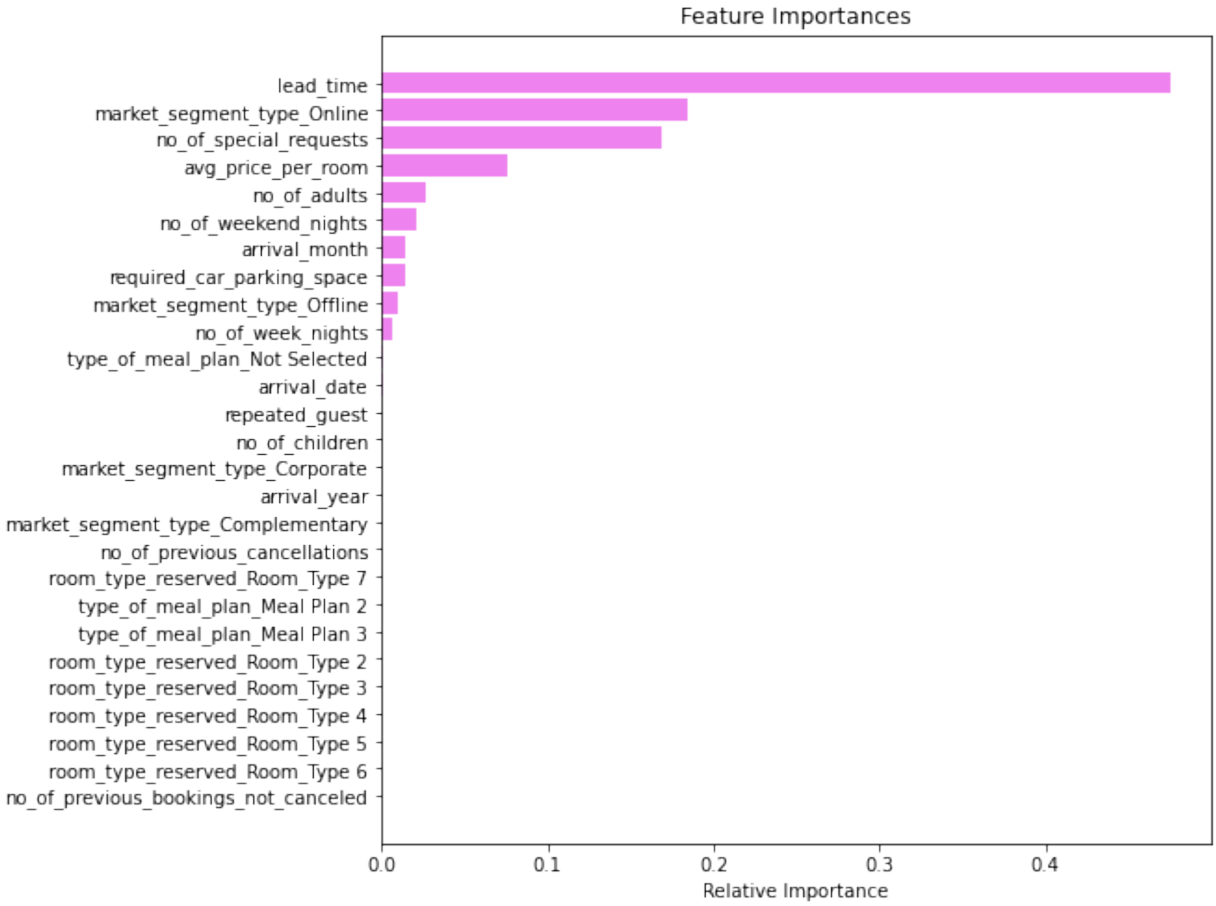
```
|   |   |   |   |   |   |--- weights: [0.75, 24.29] class: 1
|   |   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |   |--- lead_time <= 68.50
|   |   |   |   |   |   |--- weights: [960.27, 223.16] class: 0
|   |   |   |   |   |--- lead_time >  68.50
|   |   |   |   |   |   |--- weights: [129.73, 160.92] class: 1
|   |   |   |--- lead_time >  90.50
|   |   |   |   |--- lead_time <= 117.50
|   |   |   |   |   |--- avg_price_per_room <= 93.58
|   |   |   |   |   |   |--- weights: [214.72, 227.72] class: 1
|   |   |   |   |   |--- avg_price_per_room >  93.58
|   |   |   |   |   |   |--- weights: [82.76, 285.41] class: 1
|   |   |   |   |--- lead_time >  117.50
|   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |--- weights: [87.23, 81.98] class: 0
|   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |--- weights: [228.14, 48.58] class: 0
|   |   |--- market_segment_type_Online >  0.50
|   |   |   |--- lead_time <= 13.50
|   |   |   |   |--- avg_price_per_room <= 99.44
|   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |--- weights: [92.45, 0.00] class: 0
|   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |--- weights: [363.83, 132.08] class: 0
|   |   |   |   |--- avg_price_per_room >  99.44
|   |   |   |   |   |--- lead_time <= 3.50
|   |   |   |   |   |   |--- weights: [219.94, 85.01] class: 0
|   |   |   |   |   |--- lead_time >  3.50
|   |   |   |   |   |   |--- weights: [132.71, 280.85] class: 1
|   |   |   |--- lead_time >  13.50
|   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |--- avg_price_per_room <= 71.92
|   |   |   |   |   |   |--- weights: [158.80, 159.40] class: 1
|   |   |   |   |   |--- avg_price_per_room >  71.92
|   |   |   |   |   |   |--- weights: [850.67, 3543.28] class: 1
|   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |--- weights: [48.46, 1.52] class: 0
|   |--- no_of_special_requests >  0.50
|   |   |--- no_of_special_requests <= 1.50
|   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |--- lead_time <= 102.50
|   |   |   |   |   |--- type_of_meal_plan_Not Selected <= 0.50
|   |   |   |   |   |   |--- weights: [697.09, 9.11] class: 0
|   |   |   |   |   |--- type_of_meal_plan_Not Selected >  0.50
|   |   |   |   |   |   |--- weights: [15.66, 9.11] class: 0
|   |   |   |   |--- lead_time >  102.50
|   |   |   |   |   |--- no_of_week_nights <= 2.50
|   |   |   |   |   |   |--- weights: [32.06, 19.74] class: 0
|   |   |   |   |   |--- no_of_week_nights >  2.50
|   |   |   |   |   |   |--- weights: [44.73, 3.04] class: 0
|   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |--- lead_time <= 8.50
|   |   |   |   |   |--- lead_time <= 4.50
|   |   |   |   |   |   |--- weights: [498.03, 44.03] class: 0
|   |   |   |   |   |--- lead_time >  4.50
|   |   |   |   |   |   |--- weights: [258.71, 63.76] class: 0
|   |   |   |   |--- lead_time >  8.50
```

```
|   |   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |   |--- weights: [2512.51, 1451.32] class: 0
|   |   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |   |--- weights: [134.20, 1.52] class: 0
|   |   |--- no_of_special_requests >  1.50
|   |   |   |--- lead_time <= 90.50
|   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |--- weights: [1585.04, 0.00] class: 0
|   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |   |--- weights: [180.42, 57.69] class: 0
|   |   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |   |--- weights: [52.19, 0.00] class: 0
|   |   |   |--- lead_time >  90.50
|   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |--- weights: [184.90, 56.17] class: 0
|   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |--- weights: [106.61, 106.27] class: 0
|   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |--- weights: [67.10, 0.00] class: 0
|--- lead_time >  151.50
|   |--- avg_price_per_room <= 100.04
|   |   |--- no_of_special_requests <= 0.50
|   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |   |--- lead_time <= 163.50
|   |   |   |   |   |   |--- weights: [3.73, 24.29] class: 1
|   |   |   |   |   |--- lead_time >  163.50
|   |   |   |   |   |   |--- weights: [257.96, 62.24] class: 0
|   |   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |   |--- avg_price_per_room <= 2.50
|   |   |   |   |   |   |--- weights: [8.95, 3.04] class: 0
|   |   |   |   |   |--- avg_price_per_room >  2.50
|   |   |   |   |   |   |--- weights: [0.75, 97.16] class: 1
|   |   |   |--- no_of_adults >  1.50
|   |   |   |   |--- avg_price_per_room <= 82.47
|   |   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |   |--- weights: [2.98, 282.37] class: 1
|   |   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |   |--- weights: [213.97, 385.60] class: 1
|   |   |   |   |--- avg_price_per_room >  82.47
|   |   |   |   |   |--- no_of_adults <= 2.50
|   |   |   |   |   |   |--- weights: [23.86, 1030.80] class: 1
|   |   |   |   |   |--- no_of_adults >  2.50
|   |   |   |   |   |   |--- weights: [5.22, 0.00] class: 0
|   |   |--- no_of_special_requests >  0.50
|   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |--- lead_time <= 180.50
|   |   |   |   |   |--- lead_time <= 159.50
|   |   |   |   |   |   |--- weights: [7.46, 7.59] class: 1
|   |   |   |   |   |--- lead_time >  159.50
|   |   |   |   |   |   |--- weights: [37.28, 4.55] class: 0
|   |   |   |   |--- lead_time >  180.50
|   |   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |   |--- weights: [20.13, 212.54] class: 1
|   |   |   |   |   |--- no_of_special_requests >  2.50
```

```
|   |   |   |   |   |   |--- weights: [8.95, 0.00] class: 0
|   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |--- weights: [231.12, 110.82] class: 0
|   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |--- weights: [19.38, 34.92] class: 1
|   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |--- lead_time <= 348.50
|   |   |   |   |   |   |--- weights: [106.61, 3.04] class: 0
|   |   |   |   |   |--- lead_time >  348.50
|   |   |   |   |   |   |--- weights: [5.96, 4.55] class: 0
|   |--- avg_price_per_room >  100.04
|   |   |--- arrival_month <= 11.50
|   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |--- weights: [0.00, 3200.19] class: 1
|   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |--- weights: [23.11, 0.00] class: 0
|   |   |--- arrival_month >  11.50
|   |   |   |--- no_of_special_requests <= 0.50
|   |   |   |   |--- weights: [35.04, 0.00] class: 0
|   |   |   |--- no_of_special_requests >  0.50
|   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |--- weights: [3.73, 0.00] class: 0
|   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |--- weights: [3.73, 22.77] class: 1
```

In [112…

```python
# 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
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```




### Cost Complexity Pruning

In [113…

```
clf = DecisionTreeClassifier(random_state=1, class_weight="balanced")
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = abs(path.ccp_alphas), path.impurities
```

In [114…

```
pd.DataFrame(path)
```

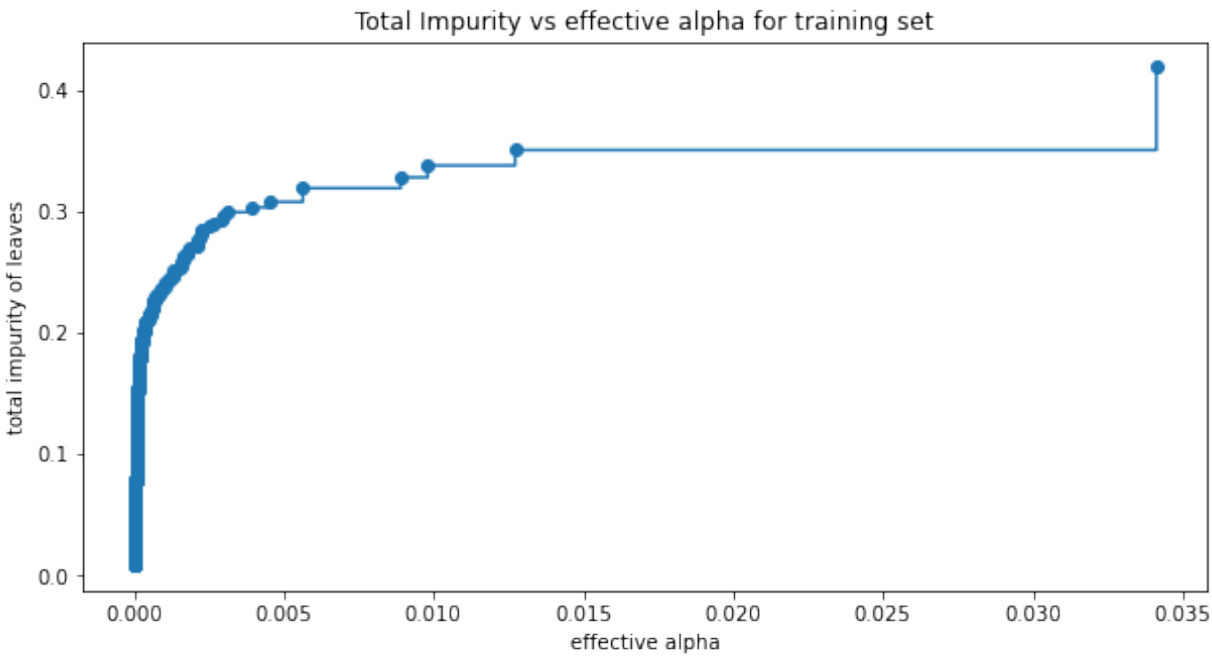
Out[114]:

| | ccp_alphas | impurities |
|---|---|---|
| 0 | 0.00000 | 0.00838 |
| 1 | 0.00000 | 0.00838 |
| 2 | 0.00000 | 0.00838 |
| 3 | 0.00000 | 0.00838 |
| 4 | 0.00000 | 0.00838 |
| ... | ... | ... |
| 1839 | 0.00890 | 0.32806 |
| 1840 | 0.00980 | 0.33786 |
| 1841 | 0.01272 | 0.35058 |
| 1842 | 0.03412 | 0.41882 |
| 1843 | 0.08118 | 0.50000 |

1844 rows × 2 columns

In [115…

```
fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(ccp_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-po
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
plt.show()
```




Next, we train a decision tree using effective alphas. The last value in `ccp_alphas` is the alpha value that prunes the whole tree, leaving the tree, `clfs[-1]`, with one node.

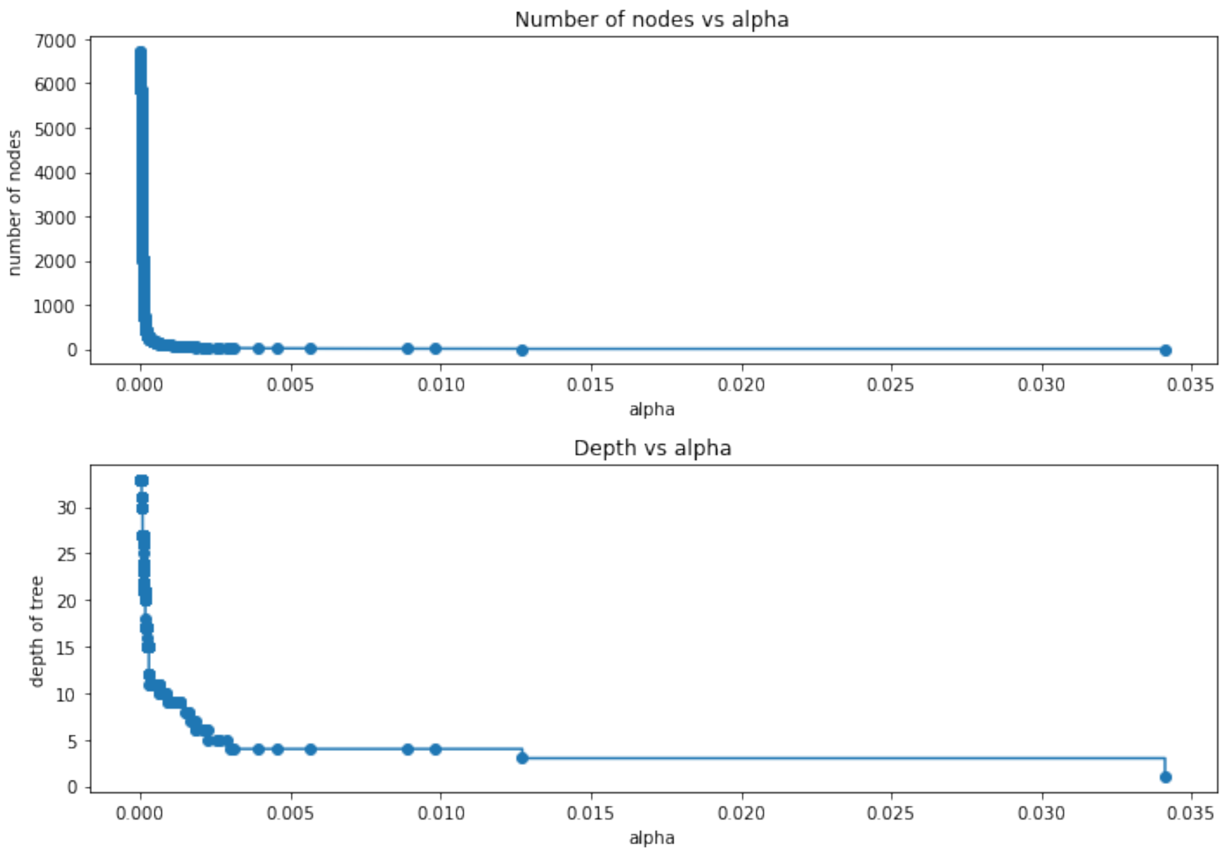
In [118...

```
clfs = []
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(
        random_state=1, ccp_alpha=ccp_alpha, class_weight="balanced"
    )
    clf.fit(X_train,y_train)## Complete the code to fit decision tree on
    clfs.append(clf)
print(
    "Number of nodes in the last tree is: {} with ccp_alpha: {}".format(
        clfs[-1].tree_.node_count, ccp_alphas[-1]
    )
)
```

```
Number of nodes in the last tree is: 1 with ccp_alpha: 0.0811791438913696
```

In [119...

```
clfs = clfs[:-1]
ccp_alphas = ccp_alphas[:-1]

node_counts = [clf.tree_.node_count for clf in clfs]
depth = [clf.tree_.max_depth for clf in clfs]
fig, ax = plt.subplots(2, 1, figsize=(10, 7))
ax[0].plot(ccp_alphas, node_counts, marker="o", drawstyle="steps-post")
ax[0].set_xlabel("alpha")
ax[0].set_ylabel("number of nodes")
ax[0].set_title("Number of nodes vs alpha")
ax[1].plot(ccp_alphas, depth, marker="o", drawstyle="steps-post")
ax[1].set_xlabel("alpha")
ax[1].set_ylabel("depth of tree")
ax[1].set_title("Depth vs alpha")
fig.tight_layout()
```



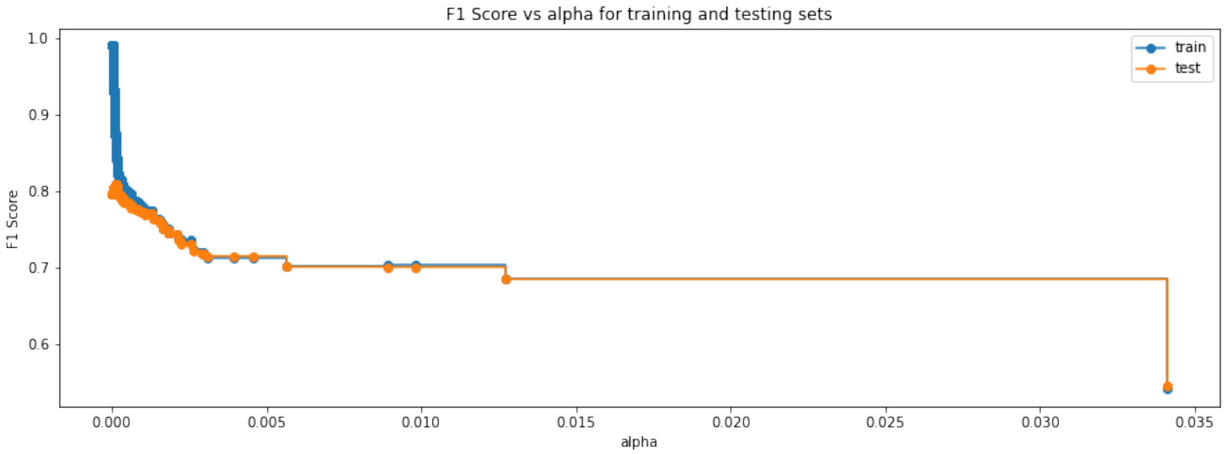
## F1 Score vs alpha for training and testing sets

In [128...

```
f1_train = []
for clf in clfs:
    pred_train = clf.predict(X_train)
    values_train = f1_score(y_train, pred_train)
    f1_train.append(values_train)

f1_test = []
for clf in clfs:
    pred_test = clf.predict(X_test)
    values_test = f1_score(y_test, pred_test)
    f1_test.append(values_test)
```

In [129...

```
fig, ax = plt.subplots(figsize=(15, 5))
ax.set_xlabel("alpha")
ax.set_ylabel("F1 Score")
ax.set_title("F1 Score vs alpha for training and testing sets")
ax.plot(ccp_alphas, f1_train, marker="o", label="train", drawstyle="steps
ax.plot(ccp_alphas, f1_test, marker="o", label="test", drawstyle="steps-p
ax.legend()
plt.show()
```




In [130...

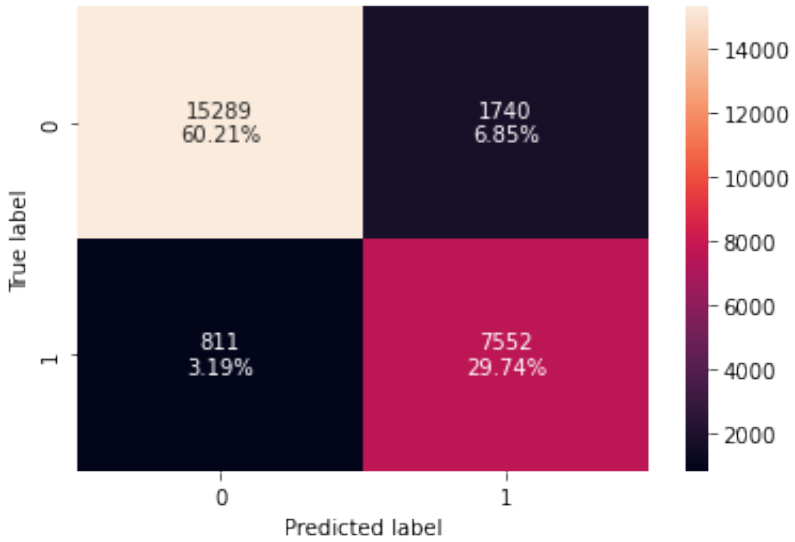
```
index_best_model = np.argmax(f1_test)
best_model = clfs[index_best_model]
print(best_model)
```

```
DecisionTreeClassifier(ccp_alpha=0.00012267633155167043,
                       class_weight='balanced', random_state=1)
```

## Checking performance on training set

In [131...

```
confusion_matrix_sklearn(best_model, X_train, y_train)
```




In [132…

```
decision_tree_post_perf_train = model_performance_classification_sklearn(
    best_model, X_train, y_train
)
decision_tree_post_perf_train
```

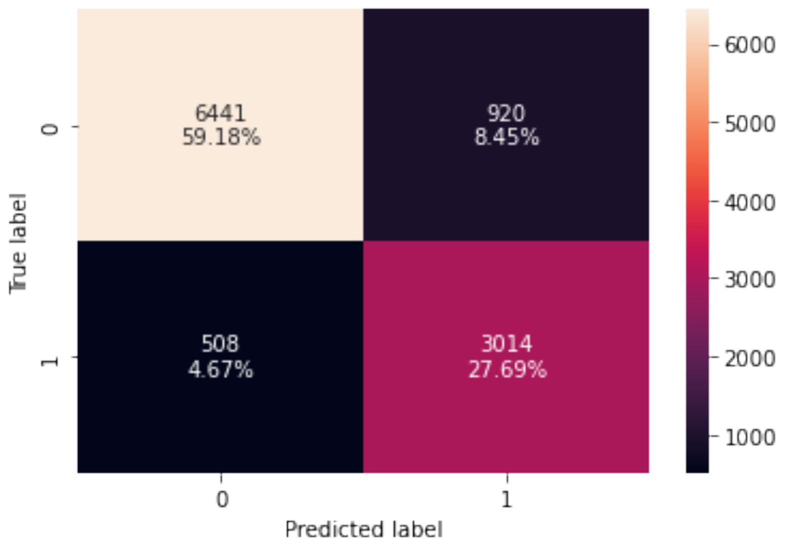
Out[132]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.89954 | 0.90303 | 0.81274 | 0.85551 |

## Checking performance on test set

In [134…

```
confusion_matrix_sklearn(best_model, X_test, y_test) ## Complete the code
```




In [148…

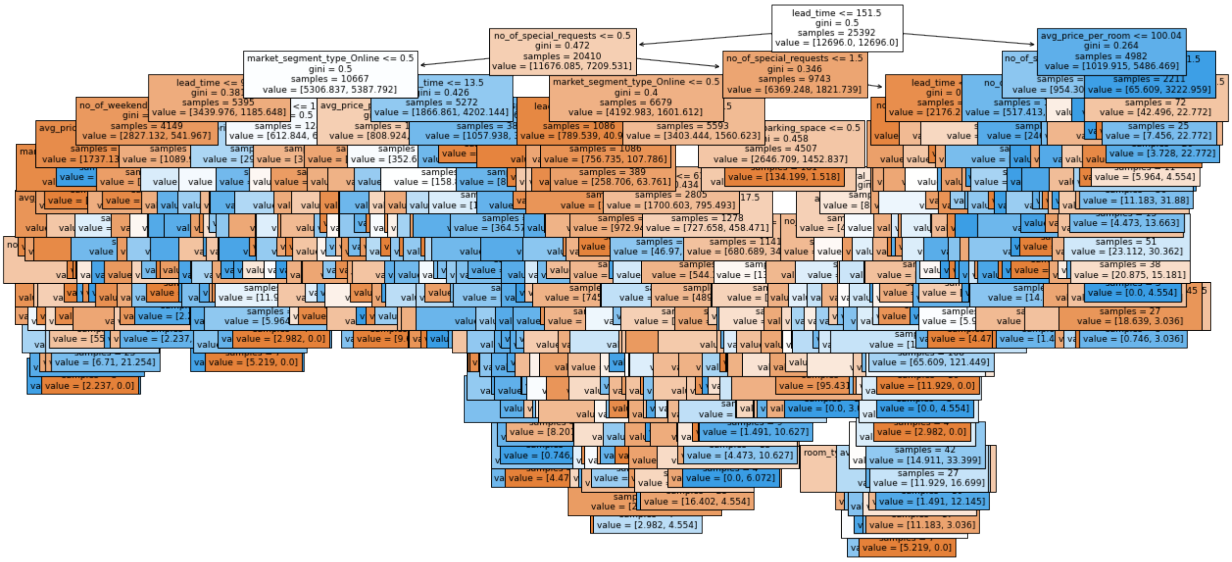
```
decision_tree_post_perf_test = model_performance_classification_sklearn(b
decision_tree_post_test
```

Out[148]:

| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.86879 | 0.85576 | 0.76614 | 0.80848 |

In [136...

```
plt.figure(figsize=(20, 10))

out = tree.plot_tree(
    best_model,
    feature_names=feature_names,
    filled=True,
    fontsize=9,
    node_ids=False,
    class_names=None,
)
for o in out:
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor("black")
        arrow.set_linewidth(1)
plt.show()
```




In [137...

```
# Text report showing the rules of a decision tree -

print(tree.export_text(best_model, feature_names=feature_names, show_weig
```

```
|--- lead_time <= 151.50
|   |--- no_of_special_requests <= 0.50
|   |   |--- market_segment_type_Online <= 0.50
|   |   |   |--- lead_time <= 90.50
|   |   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |   |--- avg_price_per_room <= 196.50
|   |   |   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |   |   |--- lead_time <= 16.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 68.50
|   |   |   |   |   |   |   |   |   |--- weights: [207.26, 10.63] class:
0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  68.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 29.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
```

```
h 5
|   |   |   |   |   |   |   |   |   |--- arrival_date >  29.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 7.59] class: 
1
|   |   |   |   |   |   |   |--- lead_time >  16.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 135.00
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_previous_bookings_not_
canceled <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 4
|   |   |   |   |   |   |   |   |   |   |--- no_of_previous_bookings_not_
canceled >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [11.18, 0.00] c
lass: 0
|   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [21.62, 0.00] class
: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  135.00
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 12.14] class: 1
|   |   |   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |   |   |--- weights: [1199.59, 1.52] class: 0
|   |   |   |   |   |--- avg_price_per_room >  196.50
|   |   |   |   |   |   |--- weights: [0.75, 24.29] class: 1
|   |   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |   |--- lead_time <= 68.50
|   |   |   |   |   |   |--- arrival_month <= 9.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 63.29
|   |   |   |   |   |   |   |   |--- arrival_date <= 20.50
|   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected <
= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [41.75, 0.00] class
: 0
|   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected >
0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 3.04] class: 
1
|   |   |   |   |   |   |   |   |--- arrival_date >  20.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 59.75
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 23.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [1.49, 12.14] c
lass: 1
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  23.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [14.91, 1.52] c
lass: 0
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  59.75
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 44.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 59.21] c
lass: 1
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  44.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |--- avg_price_per_room >  63.29
|   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 3.50
|   |   |   |   |   |   |   |   |   |--- lead_time <= 59.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 7.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
```

```
h 3
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  7.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |--- lead_time >  59.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 5.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  5.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [20.13, 0.00] c
lass: 0
|   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  3.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.75, 15.18] class: 1
|   |   |   |   |   |   |--- arrival_month >  9.50
|   |   |   |   |   |   |   |--- weights: [413.04, 27.33] class: 0
|   |   |   |   |   |--- lead_time >  68.50
|   |   |   |   |   |   |--- avg_price_per_room <= 99.98
|   |   |   |   |   |   |   |--- arrival_month <= 3.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 62.50
|   |   |   |   |   |   |   |   |   |--- weights: [15.66, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  62.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 80.38
|   |   |   |   |   |   |   |   |   |   |--- weights: [8.20, 25.81] class
: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  80.38
|   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 0.00] class:
0
|   |   |   |   |   |   |   |--- arrival_month >  3.50
|   |   |   |   |   |   |   |   |--- no_of_week_nights <= 2.50
|   |   |   |   |   |   |   |   |   |--- weights: [55.17, 3.04] class: 0
|   |   |   |   |   |   |   |   |--- no_of_week_nights >  2.50
|   |   |   |   |   |   |   |   |   |--- lead_time <= 73.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.55] class:
1
|   |   |   |   |   |   |   |   |   |--- lead_time >  73.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [21.62, 4.55] class
: 0
|   |   |   |   |   |   |--- avg_price_per_room >  99.98
|   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |--- weights: [8.95, 0.00] class: 0
|   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 132.43
|   |   |   |   |   |   |   |   |   |--- weights: [9.69, 122.97] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  132.43
|   |   |   |   |   |   |   |   |   |--- weights: [6.71, 0.00] class: 0
|   |   |   |--- lead_time >  90.50
|   |   |   |   |--- lead_time <= 117.50
|   |   |   |   |   |--- avg_price_per_room <= 93.58
|   |   |   |   |   |   |--- avg_price_per_room <= 75.07
|   |   |   |   |   |   |   |--- no_of_week_nights <= 2.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 58.75
|   |   |   |   |   |   |   |   |   |--- weights: [5.96, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  58.75
|   |   |   |   |   |   |   |   |   |--- market_segment_type_Offline <= 0
.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [4.47, 0.00] class:
0
```

```
|   |   |   |   |   |   |   |   |   |--- market_segment_type_Offline >  0
.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 4.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 118.41]
class: 1
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  4.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 4
|   |   |   |   |   |   |   |--- no_of_week_nights >  2.50
|   |   |   |   |   |   |   |   |--- arrival_date <= 11.50
|   |   |   |   |   |   |   |   |   |--- weights: [31.31, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- arrival_date >  11.50
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [23.11, 6.07] class
: 0
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [5.96, 9.11] class:
1
|   |   |   |   |   |   |--- avg_price_per_room >  75.07
|   |   |   |   |   |   |   |--- arrival_month <= 3.50
|   |   |   |   |   |   |   |   |--- weights: [59.64, 3.04] class: 0
|   |   |   |   |   |   |   |--- arrival_month >  3.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 4.50
|   |   |   |   |   |   |   |   |   |--- weights: [1.49, 16.70] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  4.50
|   |   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 86.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 16.70] c
lass: 1
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  86.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [8.95, 3.04] cl
ass: 0
|   |   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 22.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [44.73, 4.55] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  22.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |--- avg_price_per_room >  93.58
|   |   |   |   |   |   |--- arrival_date <= 11.50
|   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |--- weights: [16.40, 39.47] class: 1
|   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |--- weights: [20.13, 6.07] class: 0
|   |   |   |   |   |   |--- arrival_date >  11.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 102.09
|   |   |   |   |   |   |   |   |--- weights: [5.22, 144.22] class: 1
|   |   |   |   |   |   |   |--- avg_price_per_room >  102.09
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 109.50
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 16.70] class
: 1
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [33.55, 0.00] class
: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  109.50
```

```
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 124.25
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.98, 75.91] class
: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  124.25
|   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 3.04] class:
0
|   |   |   |   |--- lead_time >  117.50
|   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |--- arrival_date <= 7.50
|   |   |   |   |   |   |   |--- weights: [38.02, 0.00] class: 0
|   |   |   |   |   |   |--- arrival_date >  7.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 93.58
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 65.38
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.55] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  65.38
|   |   |   |   |   |   |   |   |   |--- weights: [24.60, 3.04] class: 0
|   |   |   |   |   |   |   |--- avg_price_per_room >  93.58
|   |   |   |   |   |   |   |   |--- arrival_date <= 28.00
|   |   |   |   |   |   |   |   |   |--- weights: [14.91, 72.87] class: 1
|   |   |   |   |   |   |   |   |--- arrival_date >  28.00
|   |   |   |   |   |   |   |   |   |--- weights: [9.69, 1.52] class: 0
|   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |--- weights: [84.25, 0.00] class: 0
|   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |--- lead_time <= 125.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 90.85
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 87.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [13.42, 13.66] clas
s: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  87.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 15.18] class
: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  90.85
|   |   |   |   |   |   |   |   |   |--- weights: [10.44, 0.00] class: 0
|   |   |   |   |   |   |   |--- lead_time >  125.50
|   |   |   |   |   |   |   |   |--- arrival_date <= 19.50
|   |   |   |   |   |   |   |   |   |--- weights: [58.15, 18.22] class: 0
|   |   |   |   |   |   |   |   |--- arrival_date >  19.50
|   |   |   |   |   |   |   |   |   |--- weights: [61.88, 1.52] class: 0
|   |   |--- market_segment_type_Online >  0.50
|   |   |   |--- lead_time <= 13.50
|   |   |   |   |--- avg_price_per_room <= 99.44
|   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |--- weights: [92.45, 0.00] class: 0
|   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 70.05
|   |   |   |   |   |   |   |   |   |--- weights: [31.31, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  70.05
|   |   |   |   |   |   |   |   |   |--- lead_time <= 5.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [38.77, 1.52] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
```

```
h 2
|   |   |   |   |   |   |   |   |   |--- lead_time >  5.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 3.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [6.71, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  3.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [34.30, 40.99]
class: 1
|   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 19.74] class: 1
|   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |--- lead_time <= 2.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 74.21
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 3.04] cl
ass: 1
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  74.21
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [9.69, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |   |   |--- lead_time >  2.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [4.47, 10.63] class
: 1
|   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |   |   |   |--- weights: [155.07, 6.07] class: 0
|   |   |   |   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- weights: [3.73, 10.63] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [7.46, 0.00] class: 0
|   |   |   |   |--- avg_price_per_room >  99.44
|   |   |   |   |   |--- lead_time <= 3.50
|   |   |   |   |   |   |--- avg_price_per_room <= 202.67
|   |   |   |   |   |   |   |--- no_of_week_nights <= 4.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 5.50
|   |   |   |   |   |   |   |   |   |--- weights: [63.37, 30.36] class: 0
|   |   |   |   |   |   |   |   |--- arrival_month >  5.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 20.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [115.56, 12.14] cla
ss: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  20.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [28.33, 3.04] c
lass: 0
|   |   |   |   |   |   |   |--- no_of_week_nights >  4.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 6.07] class: 1
|   |   |   |   |   |   |--- avg_price_per_room >  202.67
|   |   |   |   |   |   |   |--- weights: [0.75, 22.77] class: 1
|   |   |   |   |   |--- lead_time >  3.50
|   |   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 119.25
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 118.50
|   |   |   |   |   |   |   |   |   |--- weights: [18.64, 59.21] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  118.50
```

```
|   |   |   |   |   |   |   |   |   |--- weights: [8.20, 1.52] class: 0
|   |   |   |   |   |   |   |--- avg_price_per_room >  119.25
|   |   |   |   |   |   |   |   |--- weights: [34.30, 171.55] class: 1
|   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |--- weights: [26.09, 1.52] class: 0
|   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 14.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [9.69, 36.43] class
: 1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  14.00
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 208.67
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  208.67
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.55] cl
ass: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [15.66, 0.00] class: 0
|   |   |   |--- lead_time >  13.50
|   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |--- avg_price_per_room <= 71.92
|   |   |   |   |   |   |--- avg_price_per_room <= 59.43
|   |   |   |   |   |   |   |--- lead_time <= 84.50
|   |   |   |   |   |   |   |   |--- weights: [50.70, 7.59] class: 0
|   |   |   |   |   |   |   |--- lead_time >  84.50
|   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 27.00
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 131.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 15.18] c
lass: 1
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  131.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  27.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 0.00] class:
0
|   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |--- weights: [10.44, 0.00] class: 0
|   |   |   |   |   |   |--- avg_price_per_room >  59.43
|   |   |   |   |   |   |   |--- lead_time <= 25.50
|   |   |   |   |   |   |   |   |--- weights: [20.88, 6.07] class: 0
|   |   |   |   |   |   |   |--- lead_time >  25.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 71.34
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 3.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 68.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [15.66, 78.94]
class: 1
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  68.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |--- arrival_month >  3.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 102.00
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  102.00
```

```
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [12.67, 3.04] c
lass: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  71.34
|   |   |   |   |   |   |   |   |   |--- weights: [11.18, 0.00] class: 0
|   |   |   |   |   |--- avg_price_per_room >  71.92
|   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |--- lead_time <= 65.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 120.45
|   |   |   |   |   |   |   |   |   |--- weights: [79.77, 9.11] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  120.45
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 0.00] class: 
0
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 12.14] class
: 1
|   |   |   |   |   |   |   |--- lead_time >  65.50
|   |   |   |   |   |   |   |   |--- type_of_meal_plan_Meal Plan 2 <= 0.5
0
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 27.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [16.40, 47.06] clas
s: 1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  27.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [3.73, 0.00] class: 
0
|   |   |   |   |   |   |   |   |--- type_of_meal_plan_Meal Plan 2 >  0.5
0
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 63.76] class: 1
|   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 104.31
|   |   |   |   |   |   |   |   |--- lead_time <= 25.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [16.40, 0.00] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [38.77, 118.41]
class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [23.11, 0.00] class
: 0
|   |   |   |   |   |   |   |   |--- lead_time >  25.50
|   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected <
= 0.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [39.51, 185.21]
class: 1
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 6
|   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected >
0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [73.81, 411.41] cla
ss: 1
|   |   |   |   |   |   |   |--- avg_price_per_room >  104.31
|   |   |   |   |   |   |   |   |--- arrival_month <= 10.50
|   |   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 5 <
```

```
= 0.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 195.30
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 9
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  195.30
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 138.15]
class: 1
|   |   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 5 >
0.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 22.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [11.18, 6.07] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  22.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 9.11] cl
ass: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  10.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 168.06
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 22.00
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  22.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [17.15, 83.50]
class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  168.06
|   |   |   |   |   |   |   |   |   |   |--- weights: [12.67, 6.07] class
: 0
|   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |--- weights: [48.46, 1.52] class: 0
|   |--- no_of_special_requests >  0.50
|   |   |--- no_of_special_requests <= 1.50
|   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |--- lead_time <= 102.50
|   |   |   |   |   |--- type_of_meal_plan_Not Selected <= 0.50
|   |   |   |   |   |   |--- weights: [697.09, 9.11] class: 0
|   |   |   |   |   |--- type_of_meal_plan_Not Selected >  0.50
|   |   |   |   |   |   |--- lead_time <= 63.00
|   |   |   |   |   |   |   |--- weights: [15.66, 1.52] class: 0
|   |   |   |   |   |   |--- lead_time >  63.00
|   |   |   |   |   |   |   |--- weights: [0.00, 7.59] class: 1
|   |   |   |   |--- lead_time >  102.50
|   |   |   |   |   |--- no_of_week_nights <= 2.50
|   |   |   |   |   |   |--- lead_time <= 105.00
|   |   |   |   |   |   |   |--- weights: [0.75, 6.07] class: 1
|   |   |   |   |   |   |--- lead_time >  105.00
|   |   |   |   |   |   |   |--- weights: [31.31, 13.66] class: 0
|   |   |   |   |   |--- no_of_week_nights >  2.50
|   |   |   |   |   |   |--- weights: [44.73, 3.04] class: 0
|   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |--- lead_time <= 8.50
|   |   |   |   |   |--- lead_time <= 4.50
|   |   |   |   |   |   |--- no_of_week_nights <= 10.00
|   |   |   |   |   |   |   |--- weights: [498.03, 40.99] class: 0
|   |   |   |   |   |   |--- no_of_week_nights >  10.00
|   |   |   |   |   |   |   |--- weights: [0.00, 3.04] class: 1
|   |   |   |   |   |--- lead_time >  4.50
|   |   |   |   |   |   |--- arrival_date <= 13.50
|   |   |   |   |   |   |   |--- arrival_month <= 9.50
```

```
|   |   |   |   |   |   |   |   |--- weights: [58.90, 36.43] class: 0
|   |   |   |   |   |   |   |--- arrival_month >  9.50
|   |   |   |   |   |   |   |   |--- weights: [33.55, 1.52] class: 0
|   |   |   |   |   |   |--- arrival_date >  13.50
|   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected <= 0.50
|   |   |   |   |   |   |   |   |--- weights: [123.76, 9.11] class: 0
|   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected >  0.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 126.33
|   |   |   |   |   |   |   |   |   |--- weights: [32.80, 3.04] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  126.33
|   |   |   |   |   |   |   |   |   |--- weights: [9.69, 13.66] class: 1
|   |   |   |   |--- lead_time >  8.50
|   |   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |   |--- avg_price_per_room <= 118.55
|   |   |   |   |   |   |   |--- lead_time <= 61.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [70.08, 0.00] class
: 0
|   |   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 4.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 11
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  4.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 6
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [126.74, 1.52] class: 0
|   |   |   |   |   |   |   |--- lead_time >  61.50
|   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 7.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [4.47, 57.69] class
: 1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  7.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 66.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [5.22, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  66.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 5
|   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 9.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 71.93
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [54.43, 3.04] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  71.93
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 10
|   |   |   |   |   |   |   |   |   |--- arrival_month >  9.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 4
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 6
|   |   |   |   |   |   |--- avg_price_per_room >  118.55
|   |   |   |   |   |   |   |--- arrival_month <= 8.50
```

```
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 19.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 7.50
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 177.15
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 6
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  177.15
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  7.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 6.07] class:
1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  19.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 27.50
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 121.20
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [18.64, 6.07] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  121.20
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 4
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  27.50
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time <= 55.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time >  55.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 9.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [11.93, 10.63] clas
s: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  9.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [37.28, 0.00] class
: 0
|   |   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 119.20
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [9.69, 28.84] c
lass: 1
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  119.20
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 12
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time <= 100.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [49.95, 0.00] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time >  100.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.75, 18.22] c
lass: 1
|   |   |   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |   |   |--- weights: [134.20, 1.52] class: 0
|   |   |--- no_of_special_requests >  1.50
|   |   |   |--- lead_time <= 90.50
|   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |--- weights: [1585.04, 0.00] class: 0
|   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |--- no_of_special_requests <= 2.50
```

```
|   |   |   |   |   |   |--- no_of_week_nights <= 9.50
|   |   |   |   |   |   |   |--- lead_time <= 6.50
|   |   |   |   |   |   |   |   |--- weights: [32.06, 0.00] class: 0
|   |   |   |   |   |   |   |--- lead_time >  6.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 5.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [23.11, 1.52] class
: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  5.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 93.09
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 2
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  93.09
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [77.54, 27.33]
class: 0
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [19.38, 0.00] class: 0
|   |   |   |   |   |   |--- no_of_week_nights >  9.50
|   |   |   |   |   |   |   |--- weights: [0.00, 3.04] class: 1
|   |   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |   |--- weights: [52.19, 0.00] class: 0
|   |   |   |--- lead_time >  90.50
|   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |--- avg_price_per_room <= 202.95
|   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 7.50
|   |   |   |   |   |   |   |   |   |--- weights: [1.49, 9.11] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  7.50
|   |   |   |   |   |   |   |   |   |--- weights: [8.20, 3.04] class: 0
|   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |--- lead_time <= 150.50
|   |   |   |   |   |   |   |   |   |--- weights: [175.20, 28.84] class:
0
|   |   |   |   |   |   |   |   |--- lead_time >  150.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.55] class: 1
|   |   |   |   |   |   |--- avg_price_per_room >  202.95
|   |   |   |   |   |   |   |--- weights: [0.00, 10.63] class: 1
|   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |--- avg_price_per_room <= 153.15
|   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 2 <= 0.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 71.12
|   |   |   |   |   |   |   |   |   |--- weights: [3.73, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  71.12
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 90.42
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [12.67, 7.59] c
lass: 0
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  90.42
|   |   |   |   |   |   |   |   |   |   |--- weights: [64.12, 60.72] clas
s: 0
|   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 2 >  0.50
|   |   |   |   |   |   |   |   |--- weights: [5.96, 0.00] class: 0
|   |   |   |   |   |   |--- avg_price_per_room >  153.15
```

```
|   |   |   |   |   |   |   |--- weights: [12.67, 3.04] class: 0
|   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |--- weights: [67.10, 0.00] class: 0
|--- lead_time >  151.50
|   |--- avg_price_per_room <= 100.04
|   |   |--- no_of_special_requests <= 0.50
|   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |   |--- lead_time <= 163.50
|   |   |   |   |   |   |--- arrival_month <= 5.00
|   |   |   |   |   |   |   |--- weights: [2.98, 0.00] class: 0
|   |   |   |   |   |   |--- arrival_month >  5.00
|   |   |   |   |   |   |   |--- weights: [0.75, 24.29] class: 1
|   |   |   |   |   |--- lead_time >  163.50
|   |   |   |   |   |   |--- lead_time <= 341.00
|   |   |   |   |   |   |   |--- lead_time <= 173.00
|   |   |   |   |   |   |   |   |--- arrival_date <= 3.50
|   |   |   |   |   |   |   |   |   |--- weights: [46.97, 9.11] class: 0
|   |   |   |   |   |   |   |   |--- arrival_date >  3.50
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 13.66] class
: 1
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 0.00] class:
0
|   |   |   |   |   |   |   |--- lead_time >  173.00
|   |   |   |   |   |   |   |   |--- arrival_month <= 5.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 7.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.55] class:
1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  7.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [6.71, 0.00] class:
0
|   |   |   |   |   |   |   |   |--- arrival_month >  5.50
|   |   |   |   |   |   |   |   |   |--- weights: [188.62, 7.59] class: 0
|   |   |   |   |   |   |--- lead_time >  341.00
|   |   |   |   |   |   |   |--- weights: [13.42, 27.33] class: 1
|   |   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |   |--- avg_price_per_room <= 2.50
|   |   |   |   |   |   |--- lead_time <= 285.50
|   |   |   |   |   |   |   |--- weights: [8.20, 0.00] class: 0
|   |   |   |   |   |   |--- lead_time >  285.50
|   |   |   |   |   |   |   |--- weights: [0.75, 3.04] class: 1
|   |   |   |   |   |--- avg_price_per_room >  2.50
|   |   |   |   |   |   |--- weights: [0.75, 97.16] class: 1
|   |   |   |--- no_of_adults >  1.50
|   |   |   |   |--- avg_price_per_room <= 82.47
|   |   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |   |--- weights: [2.98, 282.37] class: 1
|   |   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |--- lead_time <= 244.00
|   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 166.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 0.00] cl
ass: 0
```

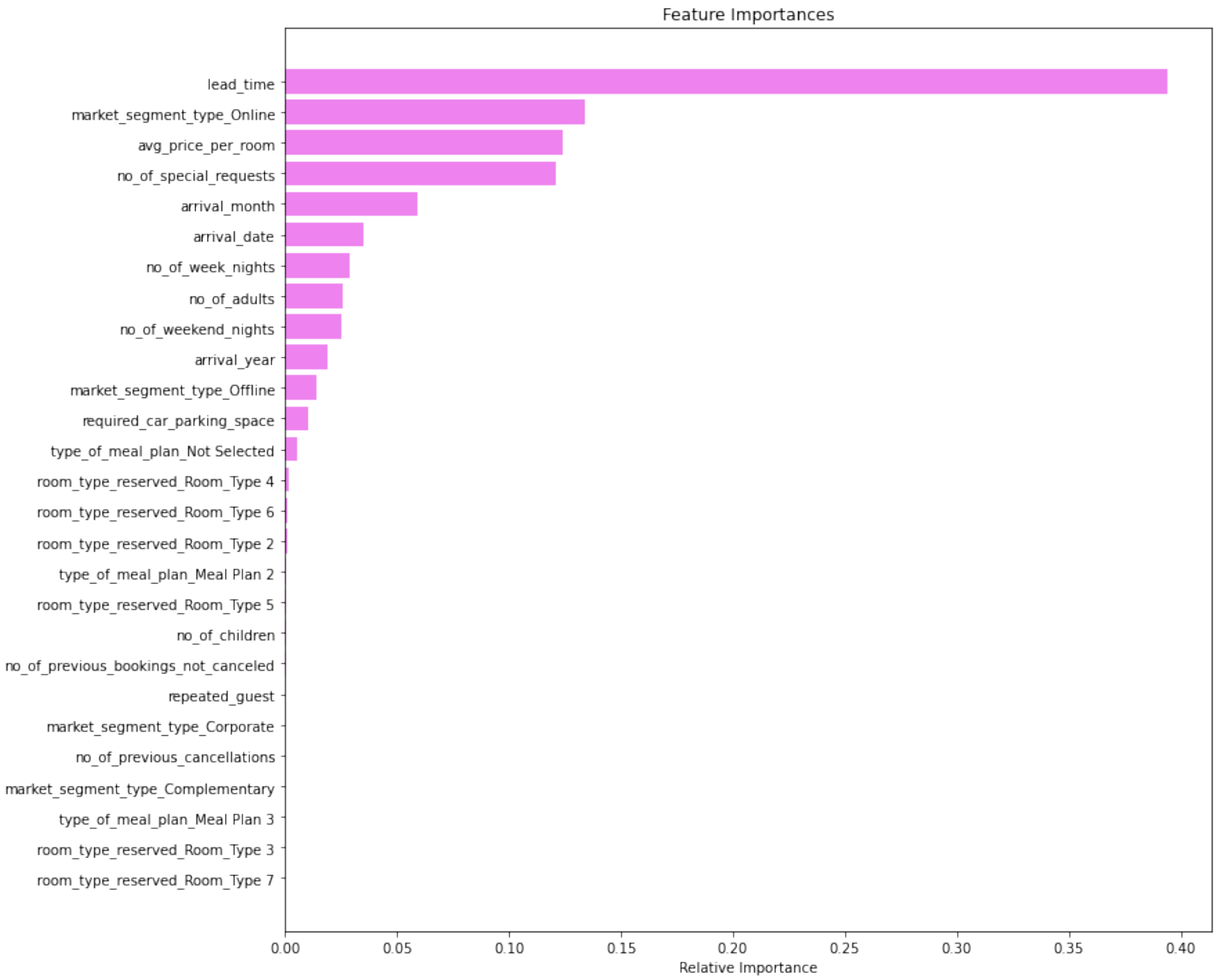
```
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  166.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 57.69] c
lass: 1
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [17.89, 0.00] class
: 0
|   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 9.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [11.18, 3.04] c
lass: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  9.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 12.14] c
lass: 1
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [75.30, 12.14] clas
s: 0
|   |   |   |   |   |   |   |--- lead_time >  244.00
|   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |--- weights: [25.35, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 80.38
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [11.18, 264.15]
class: 1
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  80.38
|   |   |   |   |   |   |   |   |   |   |--- weights: [7.46, 0.00] class:
0
|   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |--- weights: [46.22, 0.00] class: 0
|   |   |   |   |--- avg_price_per_room >  82.47
|   |   |   |   |   |--- no_of_adults <= 2.50
|   |   |   |   |   |   |--- lead_time <= 324.50
|   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 4 <= 0.
50
|   |   |   |   |   |   |   |   |   |--- weights: [7.46, 986.78] class: 1
|   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 4 >  0.
50
|   |   |   |   |   |   |   |   |   |--- market_segment_type_Offline <= 0
.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 10.63] class
: 1
|   |   |   |   |   |   |   |   |   |--- market_segment_type_Offline >  0
.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [4.47, 0.00] class:
0
|   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 19.74] class: 1
|   |   |   |   |   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |   |   |   |   |--- weights: [5.22, 0.00] class: 0
|   |   |   |   |   |   |--- lead_time >  324.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 89.00
```

```
|   |   |   |   |   |   |   |   |--- weights: [5.96, 0.00] class: 0
|   |   |   |   |   |   |   |--- avg_price_per_room >  89.00
|   |   |   |   |   |   |   |   |--- weights: [0.75, 13.66] class: 1
|   |   |   |   |   |--- no_of_adults >  2.50
|   |   |   |   |   |   |--- weights: [5.22, 0.00] class: 0
|   |   |--- no_of_special_requests >  0.50
|   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |--- lead_time <= 180.50
|   |   |   |   |   |--- lead_time <= 159.50
|   |   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |   |--- weights: [5.96, 0.00] class: 0
|   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |--- weights: [1.49, 7.59] class: 1
|   |   |   |   |   |--- lead_time >  159.50
|   |   |   |   |   |   |--- arrival_date <= 1.50
|   |   |   |   |   |   |   |--- weights: [1.49, 3.04] class: 1
|   |   |   |   |   |   |--- arrival_date >  1.50
|   |   |   |   |   |   |   |--- weights: [35.79, 1.52] class: 0
|   |   |   |   |--- lead_time >  180.50
|   |   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |   |   |   |--- no_of_adults <= 2.50
|   |   |   |   |   |   |   |   |--- weights: [12.67, 3.04] class: 0
|   |   |   |   |   |   |   |--- no_of_adults >  2.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 3.04] class: 1
|   |   |   |   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |   |   |   |--- weights: [7.46, 206.46] class: 1
|   |   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |   |--- weights: [8.95, 0.00] class: 0
|   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |--- avg_price_per_room <= 76.48
|   |   |   |   |   |   |   |--- weights: [46.97, 4.55] class: 0
|   |   |   |   |   |   |--- avg_price_per_room >  76.48
|   |   |   |   |   |   |   |--- no_of_week_nights <= 6.50
|   |   |   |   |   |   |   |   |--- arrival_date <= 27.50
|   |   |   |   |   |   |   |   |   |--- lead_time <= 233.00
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 152.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [1.49, 4.55] cl
ass: 1
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  152.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |--- lead_time >  233.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [23.11, 19.74] clas
s: 0
|   |   |   |   |   |   |   |   |--- arrival_date >  27.50
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.24, 15.18] class
: 1
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 269.00
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of dept
h 3
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  269.00
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.55] cl
```

```
ass: 1
|   |   |   |   |   |   |   |--- no_of_week_nights >  6.50
|   |   |   |   |   |   |   |   |--- weights: [4.47, 13.66] class: 1
|   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |--- arrival_date <= 14.50
|   |   |   |   |   |   |   |--- weights: [8.20, 3.04] class: 0
|   |   |   |   |   |   |--- arrival_date >  14.50
|   |   |   |   |   |   |   |--- weights: [11.18, 31.88] class: 1
|   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |--- lead_time <= 348.50
|   |   |   |   |   |   |--- weights: [106.61, 3.04] class: 0
|   |   |   |   |   |--- lead_time >  348.50
|   |   |   |   |   |   |--- weights: [5.96, 4.55] class: 0
|   |--- avg_price_per_room >  100.04
|   |   |--- arrival_month <= 11.50
|   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |--- weights: [0.00, 3200.19] class: 1
|   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |--- weights: [23.11, 0.00] class: 0
|   |   |--- arrival_month >  11.50
|   |   |   |--- no_of_special_requests <= 0.50
|   |   |   |   |--- weights: [35.04, 0.00] class: 0
|   |   |   |--- no_of_special_requests >  0.50
|   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |--- weights: [3.73, 0.00] class: 0
|   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |--- weights: [3.73, 22.77] class: 1
```

In [138…

```
importances = best_model.feature_importances_
indices = np.argsort(importances)

plt.figure(figsize=(12, 12))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], color="violet", align
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```




## Comparing Decision Tree models

In [144…

```
# training performance comparison

models_train_comp_df = pd.concat(
    [
        decision_tree_perf_train.T,
        decision_tree_tune_perf_train.T,
        decision_tree_post_perf_train.T,
    ],
    axis=1,
)
models_train_comp_df.columns = [
    "Decision Tree sklearn",
    "Decision Tree (Pre-Pruning)",
    "Decision Tree (Post-Pruning)",
]
print("Training performance comparison:")
models_train_comp_df
```

```
Training performance comparison:
```

Out[144]:

| | Decision Tree sklearn | Decision Tree (Pre-Pruning) | Decision Tree (Post-Pruning) |
|---|---|---|---|
| **Accuracy** | 0.99421 | 0.99421 | 0.89954 |
| **Recall** | 0.98661 | 0.98661 | 0.90303 |
| **Precision** | 0.99578 | 0.99578 | 0.81274 |
| **F1** | 0.99117 | 0.99117 | 0.85551 |

In [150…

```
# testing performance comparison
models_test_comp_df = pd.concat(
    [
        decision_tree_perf_test.T,
        decision_tree_tune_perf_test.T,
        decision_tree_post_perf_test.T,
    ],
    axis=1,
)
models_test_comp_df.columns = [
    "Decision Tree sklearn",
    "Decision Tree (Pre-Pruning)",
    "Decision Tree (Post-Pruning)",
]
print("Testing performance comparison:")
models_test_comp_df

## Complete the code to compare performance of test set
```

```
Testing performance comparison:
```

Out[150]:

| | Decision Tree sklearn | Decision Tree (Pre-Pruning) | Decision Tree (Post-Pruning) |
|---|---|---|---|
| **Accuracy** | 0.87118 | 0.87118 | 0.86879 |
| **Recall** | 0.81175 | 0.81175 | 0.85576 |
| **Precision** | 0.79461 | 0.79461 | 0.76614 |
| **F1** | 0.80309 | 0.80309 | 0.80848 |

# Business Recommendations