INN Hotels Project

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:

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

Importing necessary libraries and data

In [1]:
```
# Installing the libraries with the specified version.
!pip install pandas==1.5.3 numpy==1.25.2 matplotlib==3.7.1 seaborn==0.13.1 sci
```

```
[notice] A new release of pip is available: 24.2 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Note: *After running the above cell, kindly restart the notebook kernel and run all cells sequentially from the start again.*

IMPORTING THE NECESSARY LIBRARIES

In [562…

```
# this will help in making the Python code more structured automatically (help
#%load_ext nb_black

import warnings
warnings.filterwarnings("ignore")

# 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

# 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)


# To build model for prediction
import statsmodels.stats.api as sms
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
from statsmodels.tools.tools import add_constant
from sklearn.linear_model import LogisticRegression

# To build model for prediction
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,
    make_scorer,
)
```

Data Overview

- Observations
- Sanity checks

In [565…
```
# importing the dataset
data = pd.read_csv("INNHotelsGroup.csv")
```

Observing the dataset

In [568…
```
# looking at the head of the dataset
data.head()
```

Out[568…

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_
0	INN00001	2	0	1	
1	INN00002	2	0	2	
2	INN00003	1	0	2	
3	INN00004	2	0	0	
4	INN00005	2	0	1	

In [570…
```
# looking at the last 5 rows
data.tail()
```

Out[570…

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_w
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	

In [572…
```
# looking at the shape if the dataset
data.shape
```

Out[572… (36275, 19)

In [574…
```
# looking at the different the data types
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 [576…
```
# looking at the statistical summary of the numerical columns of the dataset
data.describe().T
```

Out[576…

	count	mean	std	
no_of_adults	36275.00000	1.84496	0.51871	0.0
no_of_children	36275.00000	0.10528	0.40265	0.0
no_of_weekend_nights	36275.00000	0.81072	0.87064	0.0
no_of_week_nights	36275.00000	2.20430	1.41090	0.0
required_car_parking_space	36275.00000	0.03099	0.17328	0.0
lead_time	36275.00000	85.23256	85.93082	0.0
arrival_year	36275.00000	2017.82043	0.38384	2017.0
arrival_month	36275.00000	7.42365	3.06989	1.0
arrival_date	36275.00000	15.59700	8.74045	1.0
repeated_guest	36275.00000	0.02564	0.15805	0.0
no_of_previous_cancellations	36275.00000	0.02335	0.36833	0.0
no_of_previous_bookings_not_canceled	36275.00000	0.15341	1.75417	0.0
avg_price_per_room	36275.00000	103.42354	35.08942	0.0
no_of_special_requests	36275.00000	0.61966	0.78624	0.0

In [578…

```
# checking missing values
missing_data = data.isnull().sum().any()
print(missing_data)
```

```
False
```

Q.1 OBSERVATION

- After looking at the dataset, the columns had no missing values.

Exploratory Data Analysis (EDA)

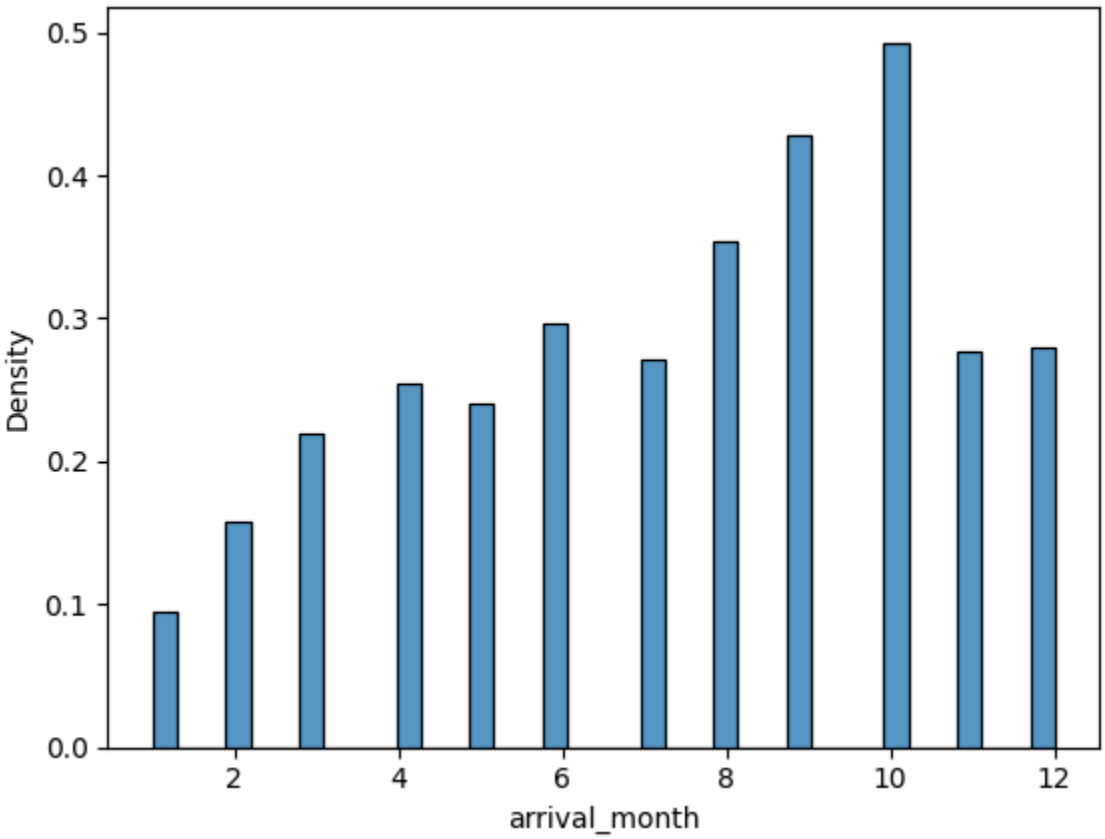
- EDA is an important part of any project involving data.
- It is important to investigate and understand the data better before building a model with it.
- A few questions have been mentioned below which will help you approach the analysis in the right manner and generate insights from the data.
- A thorough analysis of the data, in addition to the questions mentioned below, should be done.

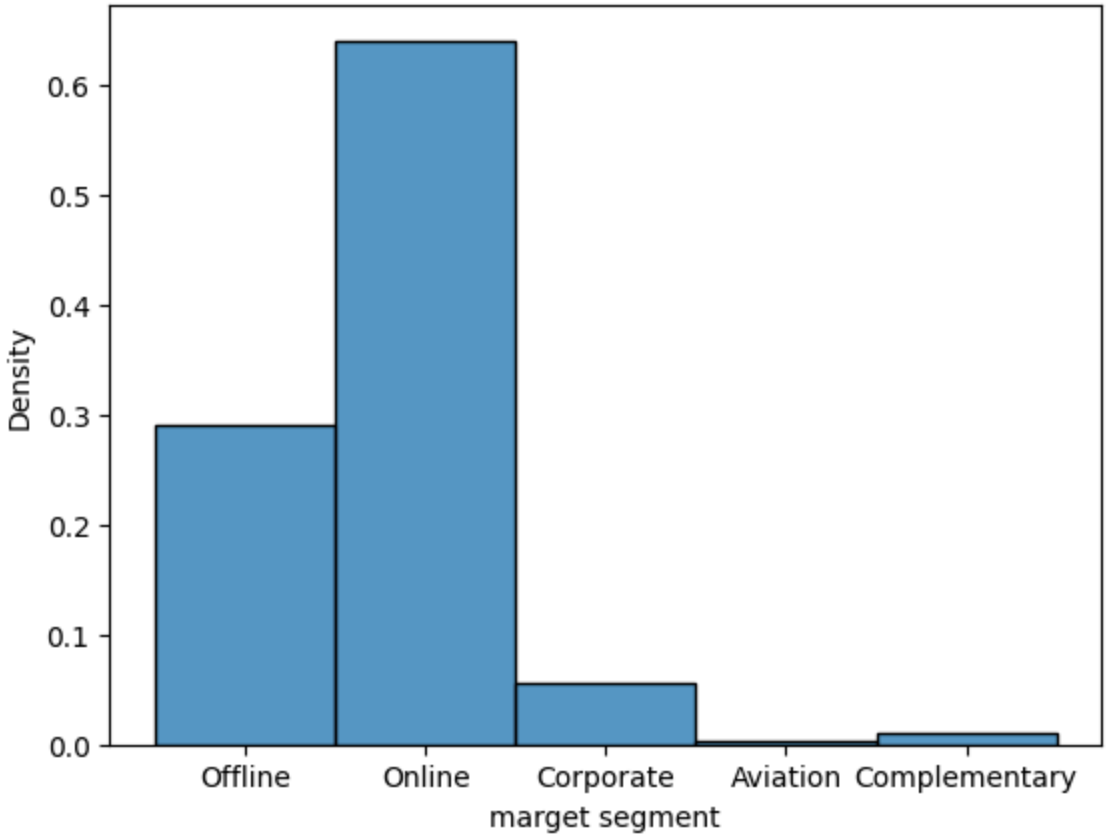
Leading Questions:

1. What are the busiest months in the hotel?
2. Which market segment do most of the guests come from?
3. Hotel rates are dynamic and change according to demand and customer demographics. What are the differences in room prices in different market segments?
4. What percentage of bookings are canceled?
5. Repeating guests are the guests who stay in the hotel often and are important to brand equity. What percentage of repeating guests cancel?
6. Many guests have special requirements when booking a hotel room. Do these requirements affect booking cancellation?

In [583…

```
# looking at the busiest month
sns.histplot(data = data, x= 'arrival_month', stat = "density")
plt.xlabel("arrival_month")
plt.show()

sns.histplot(data=data, x = 'market_segment_type', stat = 'density')
plt.xlabel("marget segment")
plt.show()
```





Q.2 OBSERVATION

- From the histoplot, it shows that the busiest month is October, followed by September and later August.
- Also, most guest come from the online market segment, which is about twice the next market segment which is offline.

In [586…

```
## calculating the percentage of booking that where canceled
tot_booking = len(data["booking_status"])
canceled_bookings = len(data[data["booking_status"] == 'Canceled'])
canceled_percent = (canceled_bookings / tot_booking) * 100
print(f" Total Bookings: {tot_booking}")
print(f"Canceled Bookings: {canceled_bookings}")
print(f"Cancellation Percentage: {canceled_percent:.2f}%")
```

```
 Total Bookings: 36275
Canceled Bookings: 11885
Cancellation Percentage: 32.76%
```

Q3 OBSERVATION

The percentage of booking cancellation is estimated to be 32.76%

In [588…
```
import scipy.stats as stats
crosstab = pd.crosstab(
    data["no_of_special_requests"], data["booking_status"]
)  # Contingency table of repeated_guest and booking_status attributes

Ho = "Number of special requests have no effect on booking cancellation"  # St
Ha = "Number of special requests have a significant effect on booking cancella

chi, p_value, dof, expected = stats.chi2_contingency(crosstab)

if p_value < 0.05:  # Setting our significance level at 5%
    print(f"{Ha} as the p_value ({p_value.round(3)}) < 0.05")
else:
    print(f"{Ho} as the p_value ({p_value.round(3)}) > 0.05")
```

```
Number of special requests have a significant effect on booking cancellation as
the p_value (0.0) < 0.05
```

Q4 OBSERVATION

- Since the P-value is less than 0.05, it shows that we have enough information to deny the null hypothesis.

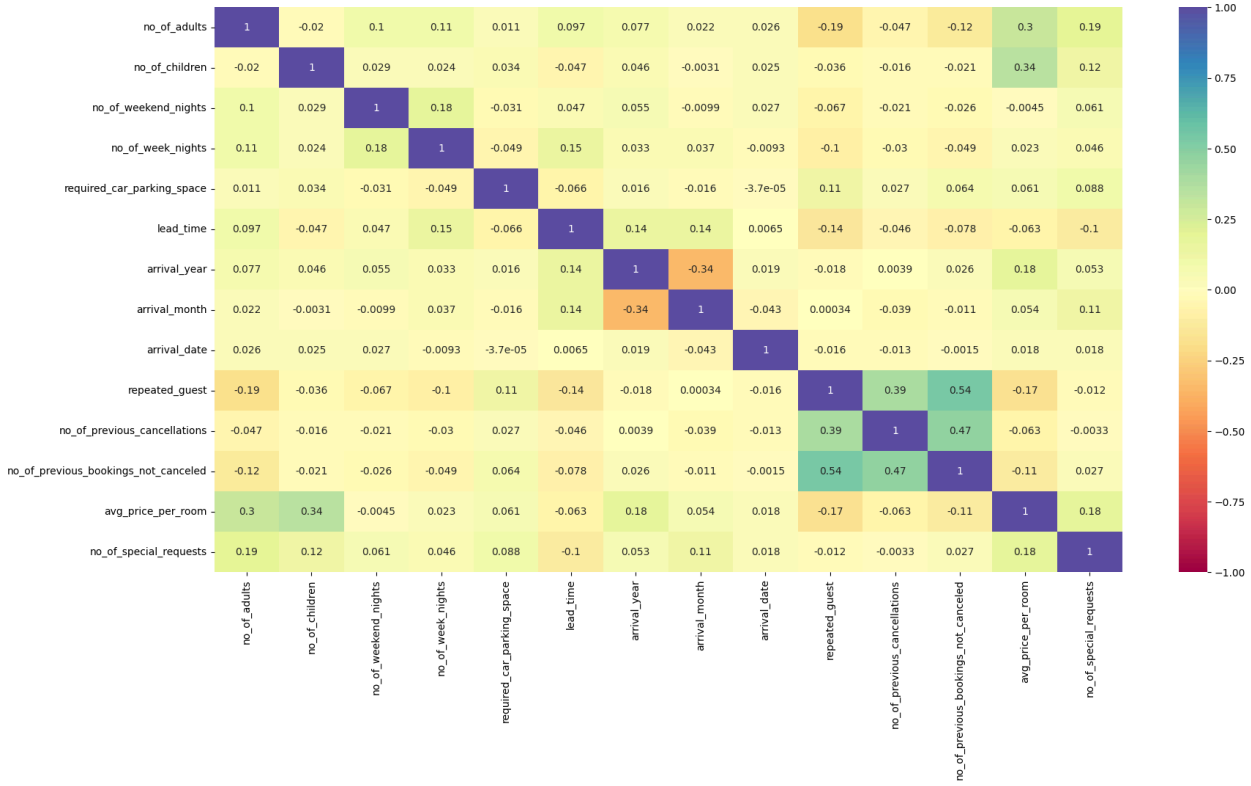
In [590…
```
## checking the percentage of repeated guest that canceled
total_repeated_guest = len(data[data["repeated_guest"] >= 1])
canceled_bookings = len(data[data["booking_status"] == 'Canceled'])
repeated_guest_canceled_percent = (total_repeated_guest/canceled_bookings)*100
print(f"Total Repeated Guest: {total_repeated_guest}")
print(f"Canceled Bookings: {canceled_bookings}")
print(f"Cancellation Percentage: {repeated_guest_canceled_percent:.2f}%")
```

```
Total Repeated Guest: 930
Canceled Bookings: 11885
Cancellation Percentage: 7.82%
```

Q5 OBSERVATION

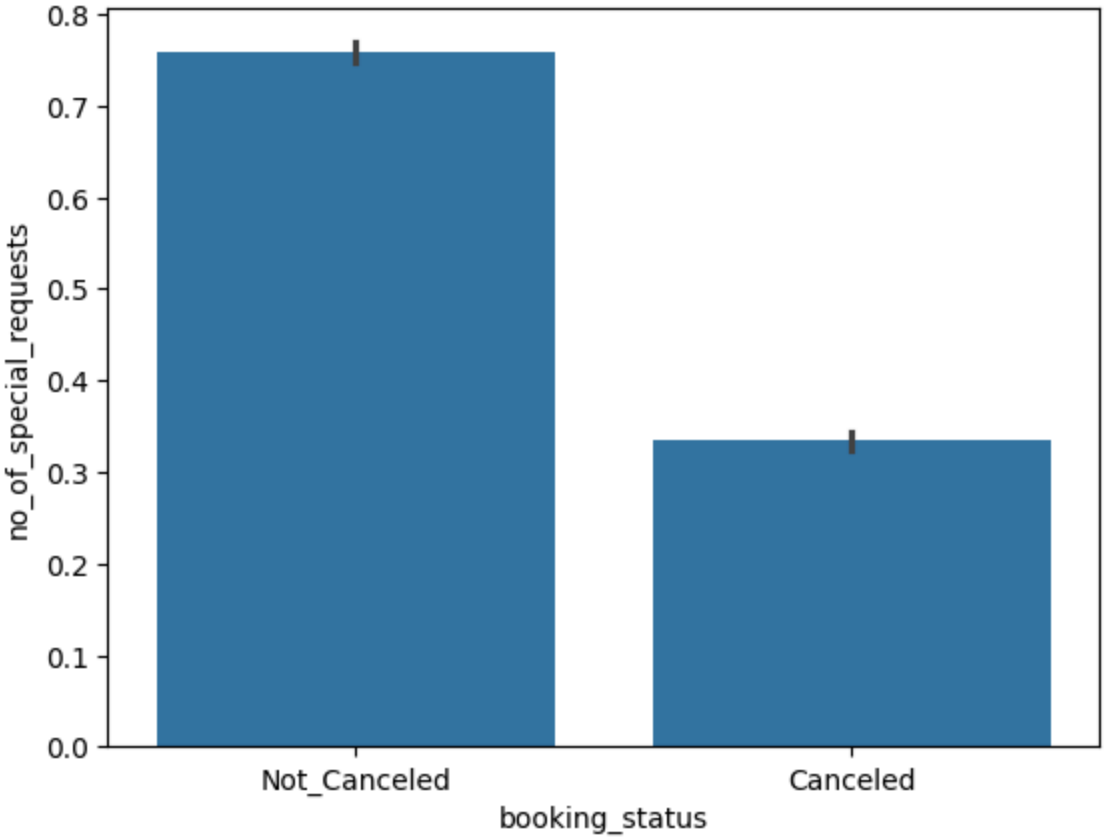
- Out of 930 repeated guests, 7.8% of them cancelled their booking.

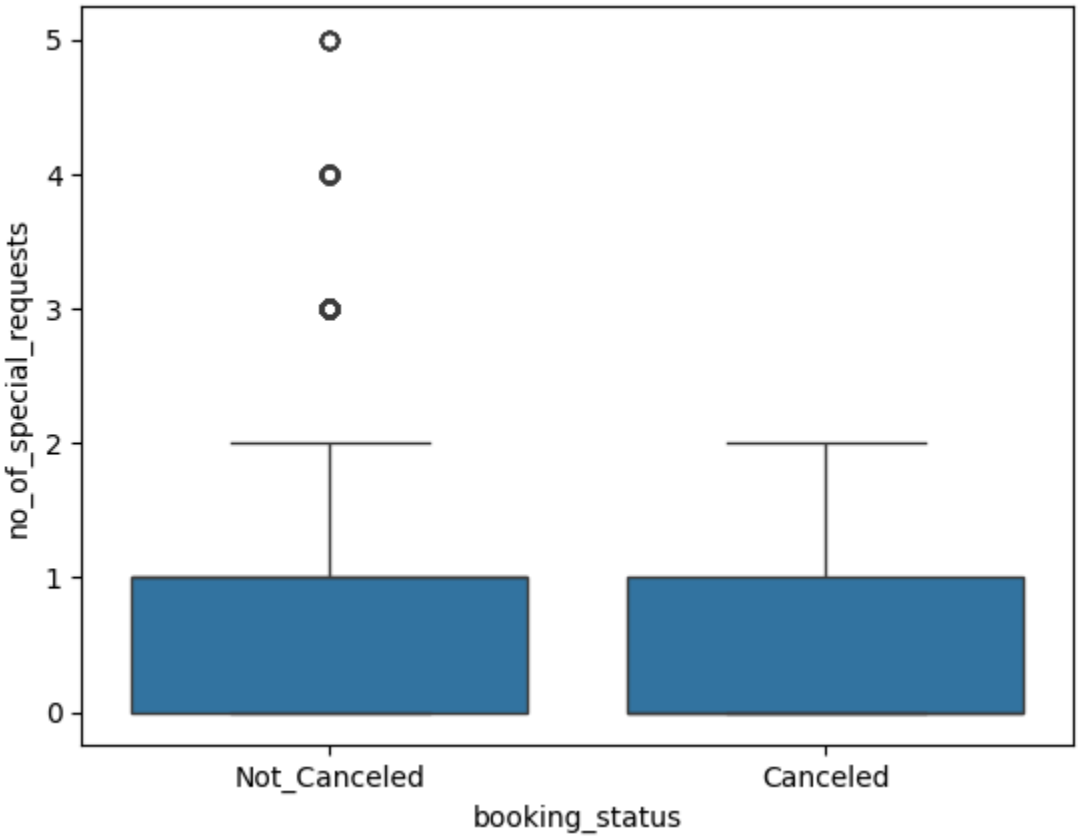
In [592…
```
plt.figure(figsize=(20,10))
sns.heatmap(data.corr(),annot=True,cmap='Spectral',vmin=-1,vmax=1)
plt.show()
```

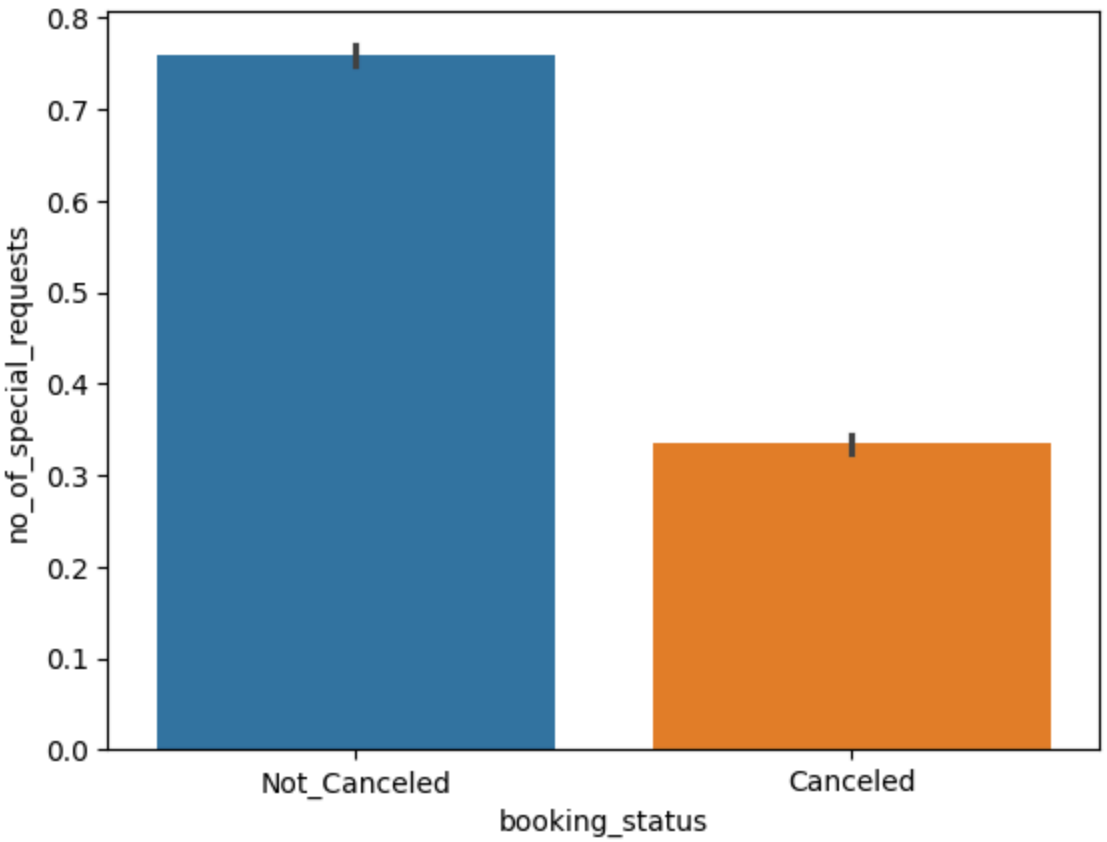


In [593…

```
sns.barplot(x='booking_status', y='no_of_special_requests', data=data)
plt.show()

sns.boxplot(x='booking_status', y='no_of_special_requests', data=data)
plt.show()

sns.barplot(x='booking_status', y='no_of_special_requests', hue='booking_statu
plt.show()
```


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
no_of_special_requests
Not_Canceled
Canceled
booking_status


5
4
3
2
1
0
no_of_special_requests
Not_Canceled
Canceled
booking_status



In [ ]:

In [596…

```
data
```

Out[596…

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_w
0	INN00001	2	0	1	
1	INN00002	2	0	2	
2	INN00003	1	0	2	
3	INN00004	2	0	0	
4	INN00005	2	0	1	
...	...	...	...	...	
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	

36275 rows × 19 columns

Data Preprocessing

- Missing value treatment (if needed)
- Feature engineering (if needed)
- Outlier detection and treatment (if needed)
- Preparing data for modeling
- Any other preprocessing steps (if needed)

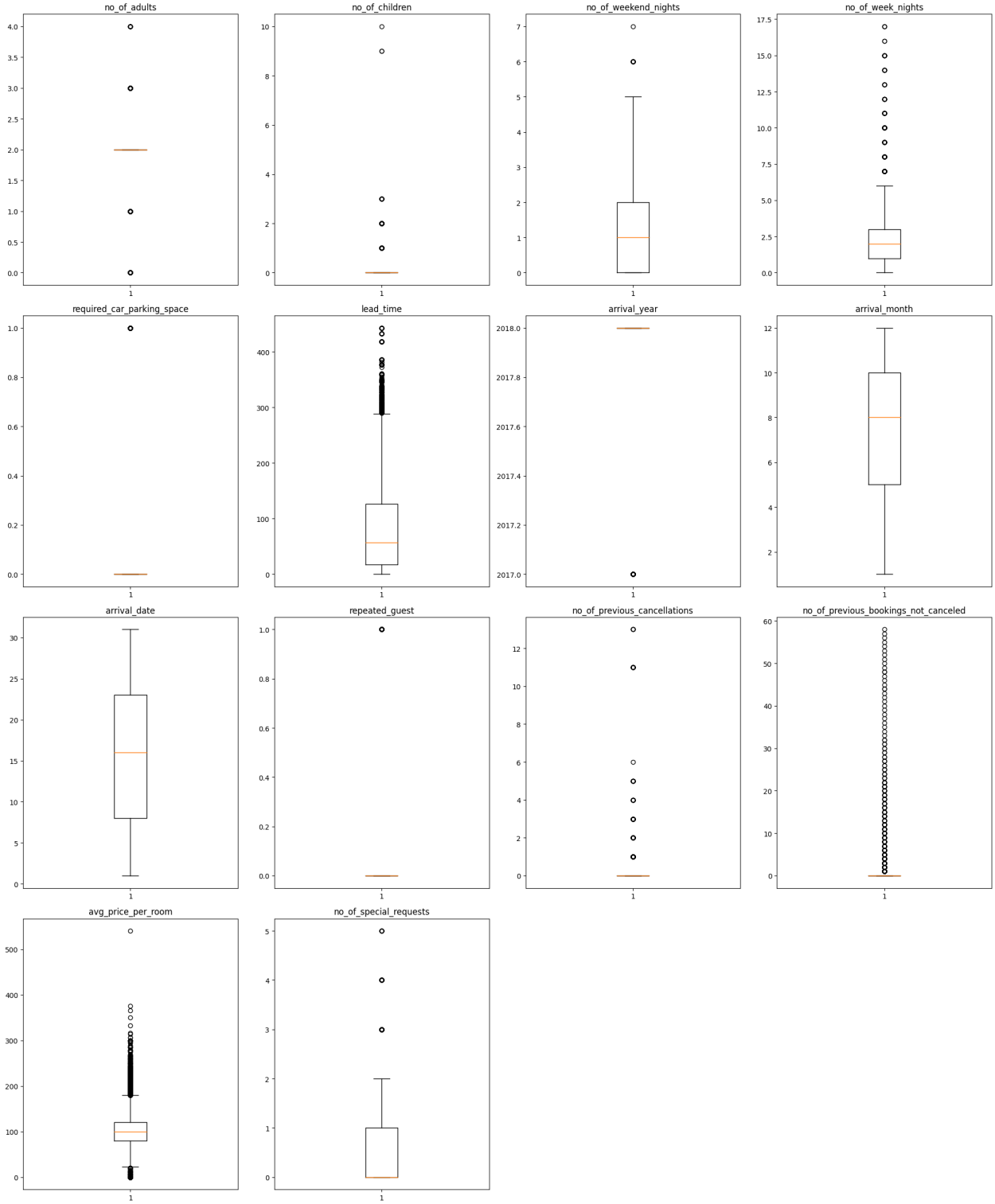
In [600…
```
## checking missing values
data.isnull().sum()
```

Out[600…
```
Booking_ID                            0
no_of_adults                          0
no_of_children                        0
no_of_weekend_nights                  0
no_of_week_nights                     0
type_of_meal_plan                     0
required_car_parking_space            0
room_type_reserved                    0
lead_time                             0
arrival_year                          0
arrival_month                         0
arrival_date                          0
market_segment_type                   0
repeated_guest                        0
no_of_previous_cancellations          0
no_of_previous_bookings_not_canceled  0
avg_price_per_room                    0
no_of_special_requests                0
booking_status                        0
dtype: int64
```

In [ ]:
```
```

OUTLIER DETECTION

In [604…
```
numerical_col = data.select_dtypes(include=np.number).columns.tolist()
plt.figure(figsize=(20, 30))

for i, variable in enumerate(numerical_col):
    plt.subplot(5, 4, i + 1)
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```



Q6 OBSERVATION

Majority of the columns have outliers and need to be treated.

OUTLIER TREATMENT

In [606…

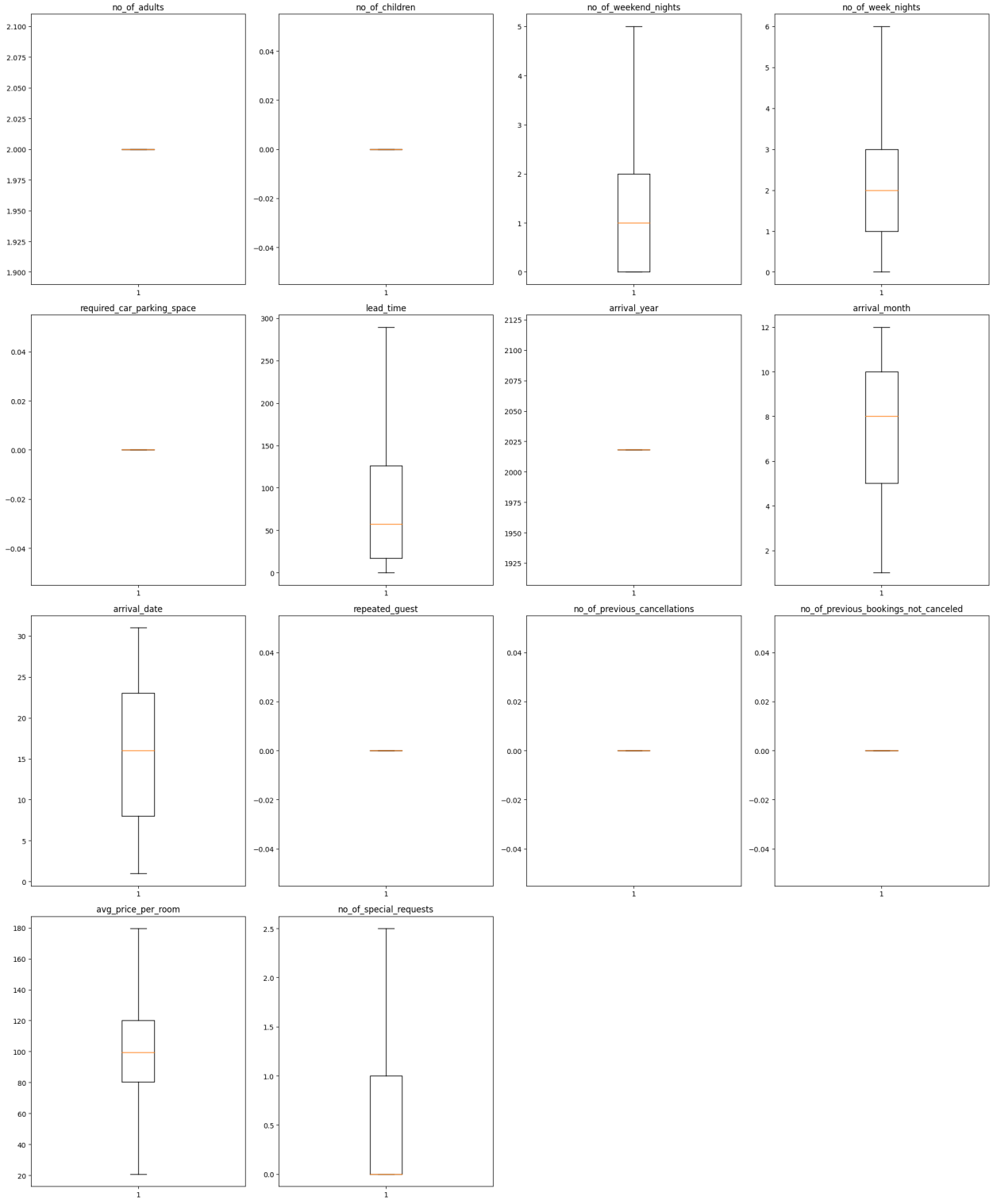
```
# functions to treat outliers by flooring and capping


def treat_outliers(df, col):
    """
    Treats outliers in a variable

    df: dataframe
    col: dataframe column
    """
    Q1 = df[col].quantile(0.25)  # 25th quantile
    Q3 = df[col].quantile(0.75)  # 75th quantile
    IQR = Q3 - Q1
    Lower_Whisker = Q1 - 1.5 * IQR
    Upper_Whisker = Q3 + 1.5 * IQR

    # all the values smaller than Lower_Whisker will be assigned the value of
    # all the values greater than Upper_Whisker will be assigned the value of
    df[col] = np.clip(df[col], Lower_Whisker, Upper_Whisker)

    return df


def treat_outliers_all(df, col_list):
    """
    Treat outliers in a list of variables

    df: dataframe
    col_list: list of dataframe columns
    """
    for c in col_list:
        df = treat_outliers(df, c)

    return df
```

In [609…

```
numerical_col = data.select_dtypes(include=np.number).columns.tolist()
data = treat_outliers_all(data, numerical_col)
```

In [611…

```
# let's look at box plot to see if outliers have been treated or not
plt.figure(figsize=(20, 30))

for i, variable in enumerate(numerical_col):
    plt.subplot(5, 4, i + 1)
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```



Q7 OBSERVATION

All outliers were well treated.

In [ ]:

EDA

- It is a good idea to explore the data once again after manipulating it.

In [629…

```
def histogram_boxplot(data, feature, figsize=(15, 10), kde=False, bins=None):
    """
    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 value c
    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
```

In [631…

```
# 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 False)
    n: displays the top n category levels (default is None, i.e., display all
    """

    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
```

In [633…

```
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_value
        by=sorter, ascending=False
    )
    print(tab1)
    print("-" * 120)
    tab = pd.crosstab(data[predictor], data[target], normalize="index").sort_v
        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()
```

In [635…

```
### 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_unic
    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_unic
    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="gist_

    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 [637…

```
data.head()
```

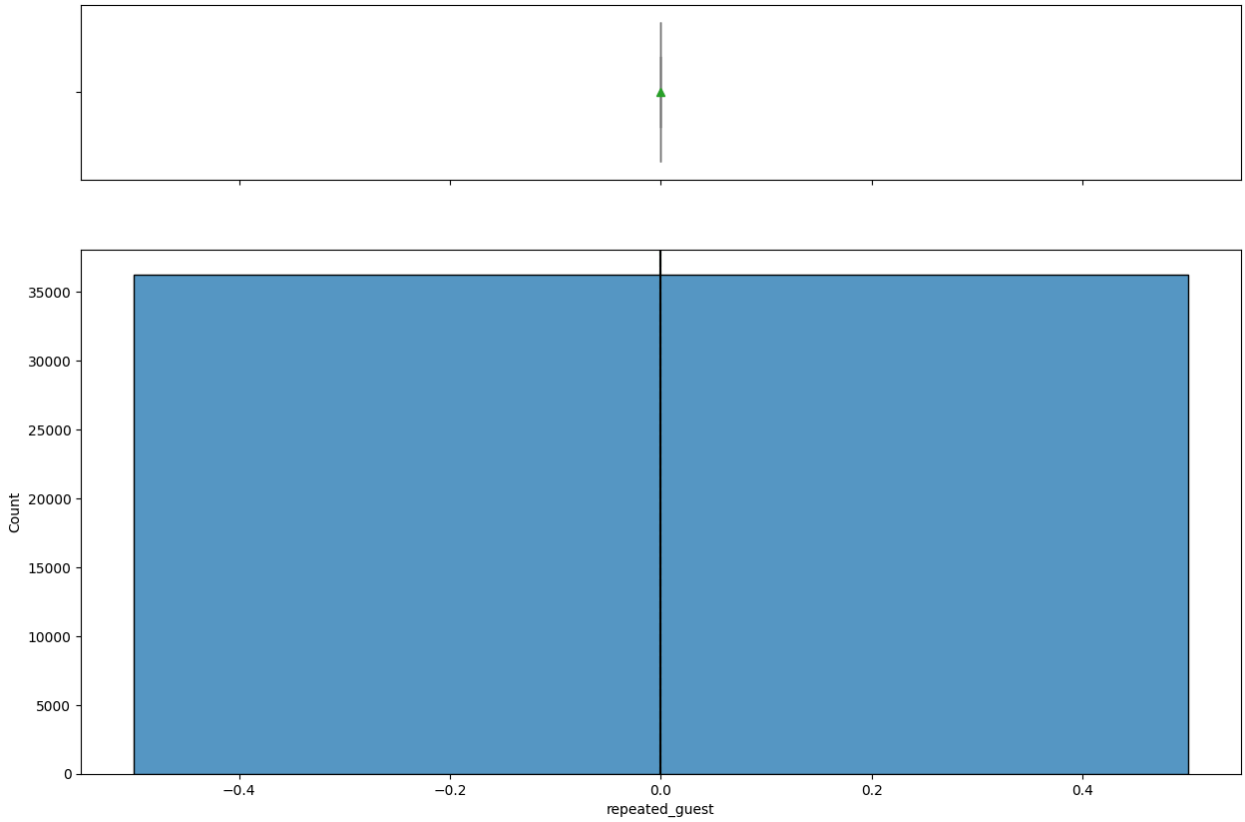
Out[637…

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_
0	INN00001	2	0	1	
1	INN00002	2	0	2	
2	INN00003	2	0	2	
3	INN00004	2	0	0	
4	INN00005	2	0	1	

univariate analysis

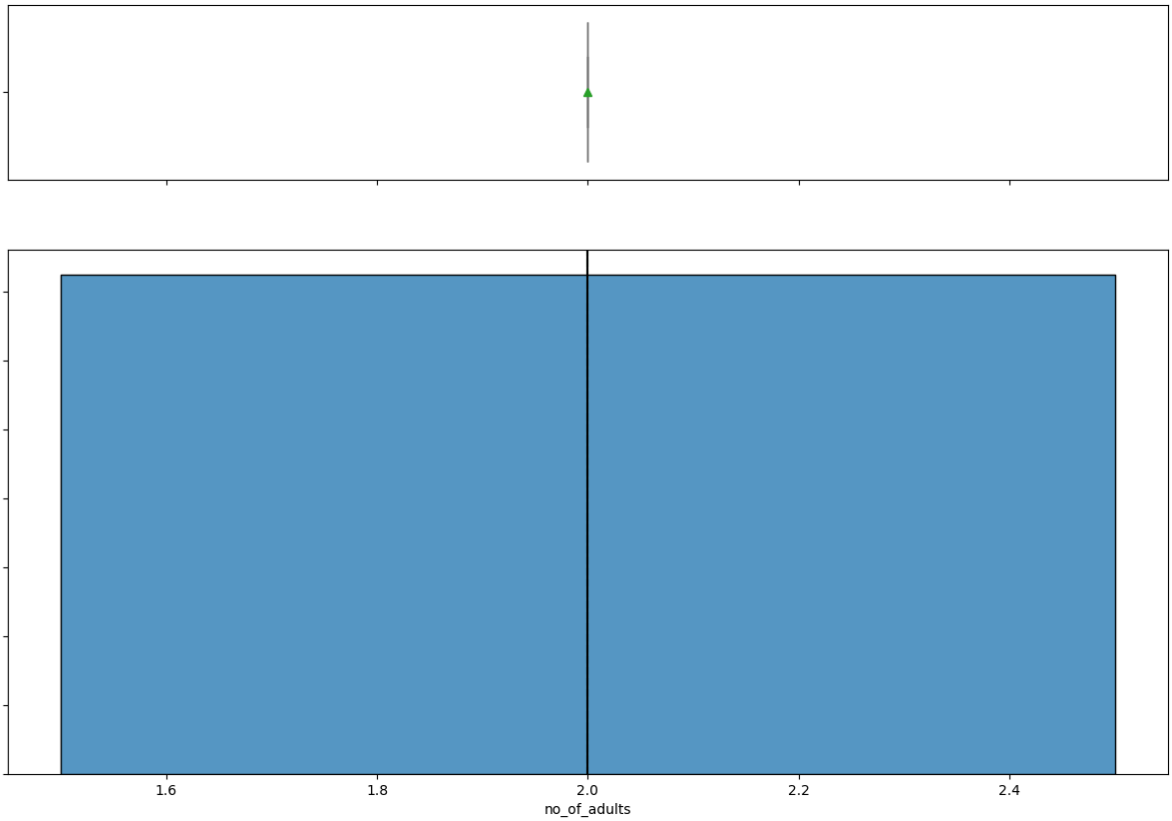
In [640…

```
histogram_boxplot(data, "repeated_guest")
```

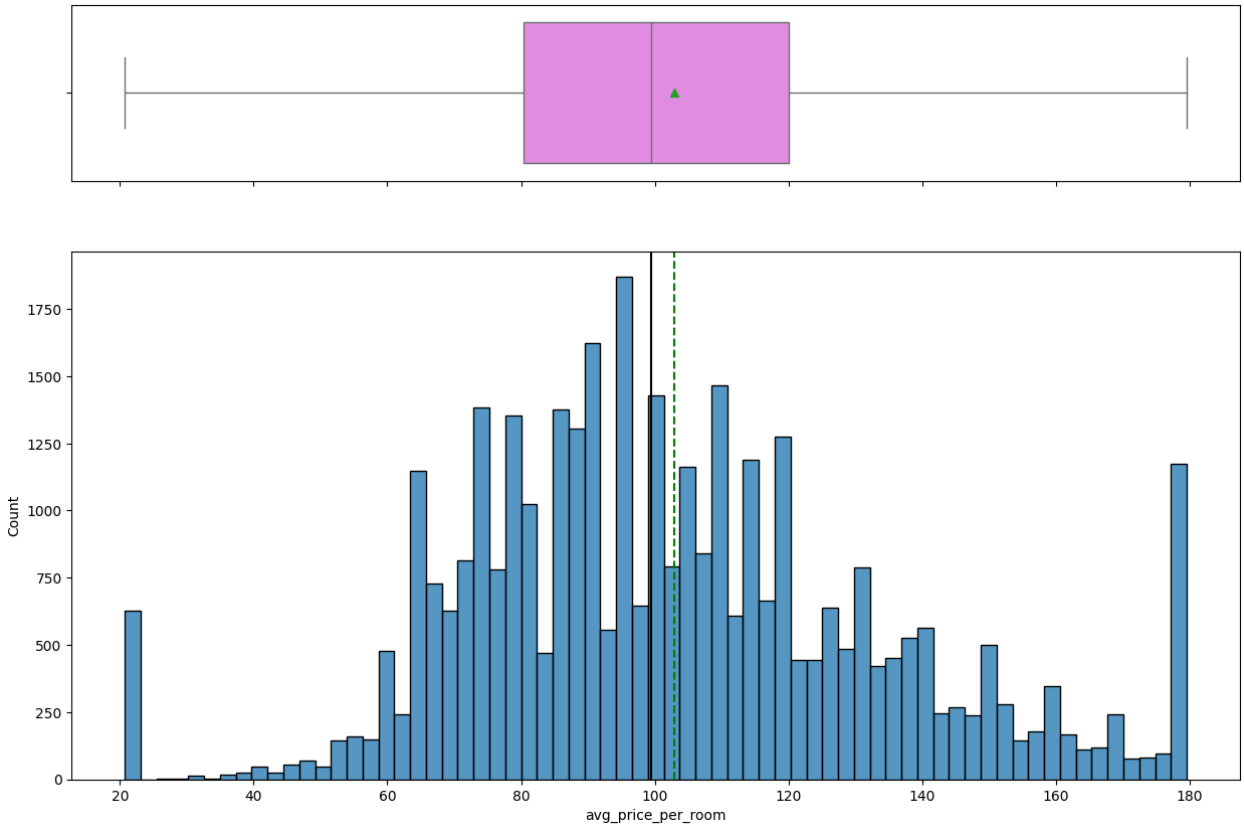


In [641…

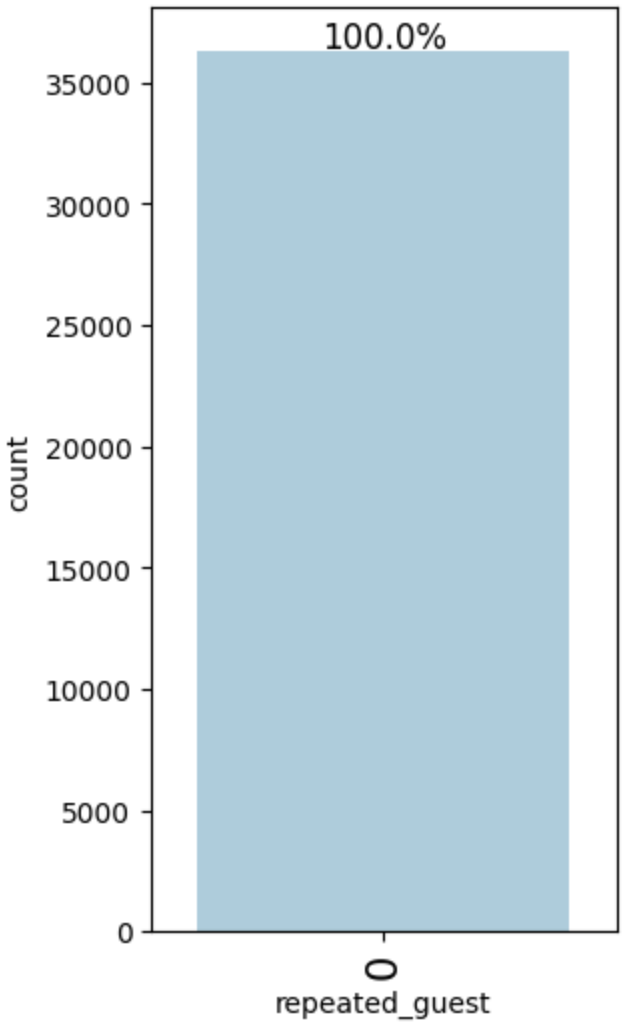
```
histogram_boxplot(data, "no_of_adults")
```



In [642…
```
histogram_boxplot(data, "avg_price_per_room")
```

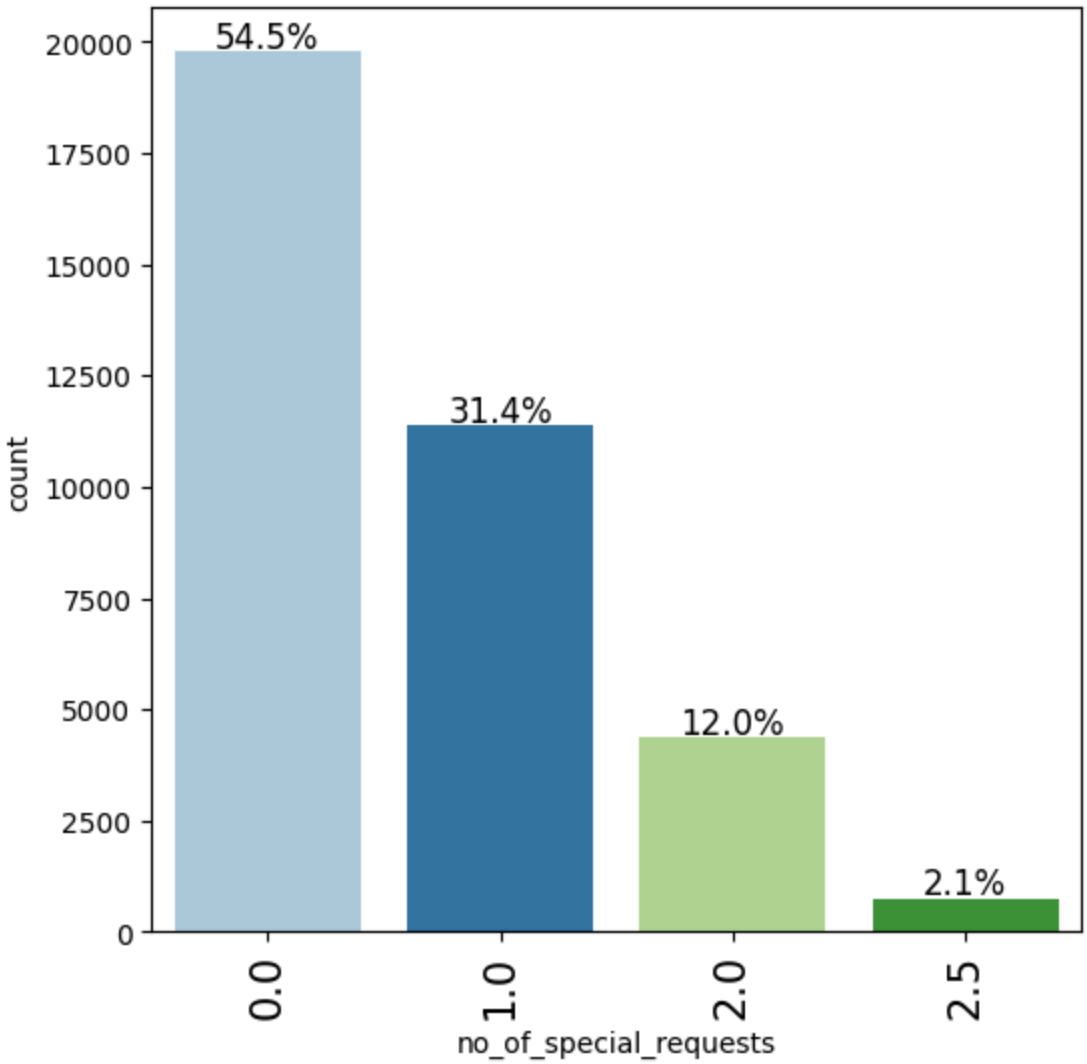


In [643…
```
labeled_barplot(data, "repeated_guest", perc=True)
```



In [644…

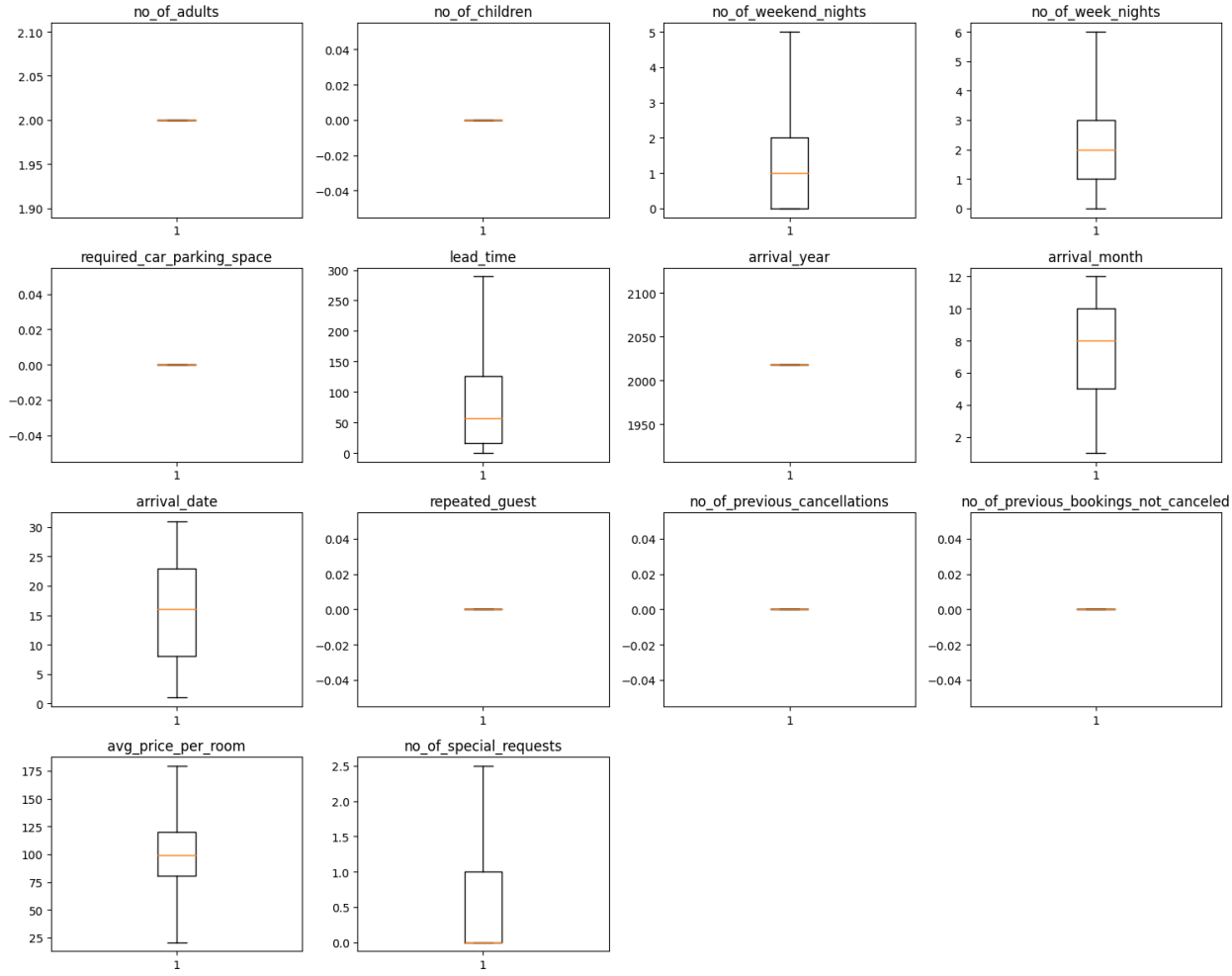
```
labeled_barplot(data, "no_of_special_requests", perc=True)
```



Bivariate analysis

In [646…

```
# outlier detection using boxplot
numeric_columns = data.select_dtypes(include=np.number).columns.tolist()

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()
```



In [653…

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

Out[653…

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights
0	2	0	1	2
1	2	0	2	3
2	2	0	2	1
3	2	0	0	2
4	2	0	1	1
...	...	...	...	...
36270	2	0	2	6
36271	2	0	1	3
36272	2	0	2	6
36273	2	0	0	3
36274	2	0	1	2

36275 rows × 18 columns

In [655…
```
X = df.drop(["booking_status"], axis = 1)
y = df["booking_status"]
```

In [657…
```
# encoding the categorical variables
X = pd.get_dummies(X, drop_first=True)
y = pd.get_dummies(y, drop_first=True)
X.head()
```

Out[657…

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	requi
0	2	0	1	2	
1	2	0	2	3	
2	2	0	2	1	
3	2	0	0	2	
4	2	0	1	1	

In [659…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, randc
```

In [ ]:
```
```

Building a Logistic Regression model

- In order to make statistical inferences from a logistic regression model,

it is important to ensure that there is no multicollinearity present in the data.

In [662…
```
# defining a function to compute different metrics to check performance of a c
def model_performance_classification_statsmodels(
    model, predictors, target, threshold=0.5
):
    """
    Function to compute different metrics to check classification model perfor

    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": f1,}
        index=[0],
    )

    return df_perf
```

In [664…
```
print("Number of rows in train data =", X_train.shape[0])
print("Number of rows in test data =", X_test.shape[0])
```

```
Number of rows in train data = 21765
Number of rows in test data = 14510
```

In [666…
```
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))
```

```
Percentage of classes in training set:
Not_Canceled
1              0.66855
0              0.33145
dtype: float64
Percentage of classes in test set:
Not_Canceled
1              0.67808
0              0.32192
dtype: float64
```

In [ ]:

Building a Logistic Regression model

In [750…

```
# fitting logistic regression model
logit = sm.Logit(y_train.astype(float), X_train.astype(float))
lg = logit.fit(disp=False)

print(lg.summary())
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
Cell In[750], line 3
      1 # fitting logistic regression model
      2 logit = sm.Logit(y_train.astype(float), X_train.astype(float))
----> 3 lg = logit.fit(disp=False)
      5 print(lg.summary())

File ~\AppData\Roaming\Python\Python311\site-packages\statsmodels\discrete\discrete_model.py:2601, in Logit.fit(self, start_params, method, maxiter, full_output, disp, callback, **kwargs)
   2598 @Appender(DiscreteModel.fit.__doc__)
   2599 def fit(self, start_params=None, method='newton', maxiter=35,
   2600         full_output=1, disp=1, callback=None, **kwargs):
-> 2601     bnryfit = super().fit(start_params=start_params,
   2602                           method=method,
   2603                           maxiter=maxiter,
   2604                           full_output=full_output,
   2605                           disp=disp,
   2606                           callback=callback,
   2607                           **kwargs)
   2609     discretefit = LogitResults(self, bnryfit)
   2610     return BinaryResultsWrapper(discretefit)

File ~\AppData\Roaming\Python\Python311\site-packages\statsmodels\discrete\discrete_model.py:243, in DiscreteModel.fit(self, start_params, method, maxiter, full_output, disp, callback, **kwargs)
    240 else:
    241     pass  # TODO: make a function factory to have multiple call-backs
--> 243 mlefit = super().fit(start_params=start_params,
    244                       method=method,
    245                       maxiter=maxiter,
    246                       full_output=full_output,
    247                       disp=disp,
    248                       callback=callback,
    249                       **kwargs)
    251 return mlefit

File ~\AppData\Roaming\Python\Python311\site-packages\statsmodels\base\model.py:582, in LikelihoodModel.fit(self, start_params, method, maxiter, full_output, disp, fargs, callback, retall, skip_hessian, **kwargs)
    580     Hinv = cov_params_func(self, xopt, retvals)
    581 elif method == 'newton' and full_output:
--> 582     Hinv = np.linalg.inv(-retvals['Hessian']) / nobs
    583 elif not skip_hessian:
    584     H = -1 * self.hessian(xopt)

File ~\AppData\Roaming\Python\Python311\site-packages\numpy\linalg\linalg.py:561, in inv(a)
    559 signature = 'D->D' if isComplexType(t) else 'd->d'
    560 extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 561 ainv = _umath_linalg.inv(a, signature=signature, extobj=extobj)
    562 return wrap(ainv.astype(result_t, copy=False))
```

```
File ~\AppData\Roaming\Python\Python311\site-packages\numpy\linalg\linalg.py:11
2, in _raise_linalgerror_singular(err, flag)
    111 def _raise_linalgerror_singular(err, flag):
--> 112     raise LinAlgError("Singular matrix")

LinAlgError: Singular matrix
```

checking and dealing with multicollinearity

In [762…

```
vif_series = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape
    index=X_train.columns,
    dtype=float,
)
print("Series before feature selection: \n\n{}\n".format(vif_series))
```

```
Series before feature selection:

no_of_adults                            0.00000
no_of_children                              NaN
no_of_weekend_nights                    1.04619
no_of_week_nights                       1.08977
required_car_parking_space                  NaN
lead_time                               1.22898
arrival_year                            0.00000
arrival_month                           1.04772
arrival_date                            1.00611
repeated_guest                              NaN
no_of_previous_cancellations                NaN
no_of_previous_bookings_not_canceled        NaN
avg_price_per_room                      1.75889
no_of_special_requests                  1.22395
type_of_meal_plan_Meal Plan 2           1.19848
type_of_meal_plan_Meal Plan 3           1.02849
type_of_meal_plan_Not Selected          1.23056
room_type_reserved_Room_Type 2          1.03211
room_type_reserved_Room_Type 3          1.00349
room_type_reserved_Room_Type 4          1.30057
room_type_reserved_Room_Type 5          1.02629
room_type_reserved_Room_Type 6          1.21977
room_type_reserved_Room_Type 7          1.06401
market_segment_type_Complementary       4.24466
market_segment_type_Corporate          15.49222
market_segment_type_Offline            58.33690
market_segment_type_Online             64.76543
dtype: float64
```

Q8 OBSERVATION

The market segment type showed the height multicollinearity amongst all the

other columns.

Model performance evaluation

In [684…
```
logit2 = sm.Logit(y_train, X_train2.astype(float))
lg2 = logit2.fit(disp=False)

print("Training performance:")
model_performance_classification_statsmodels(lg2, X_train2, y_train)
```

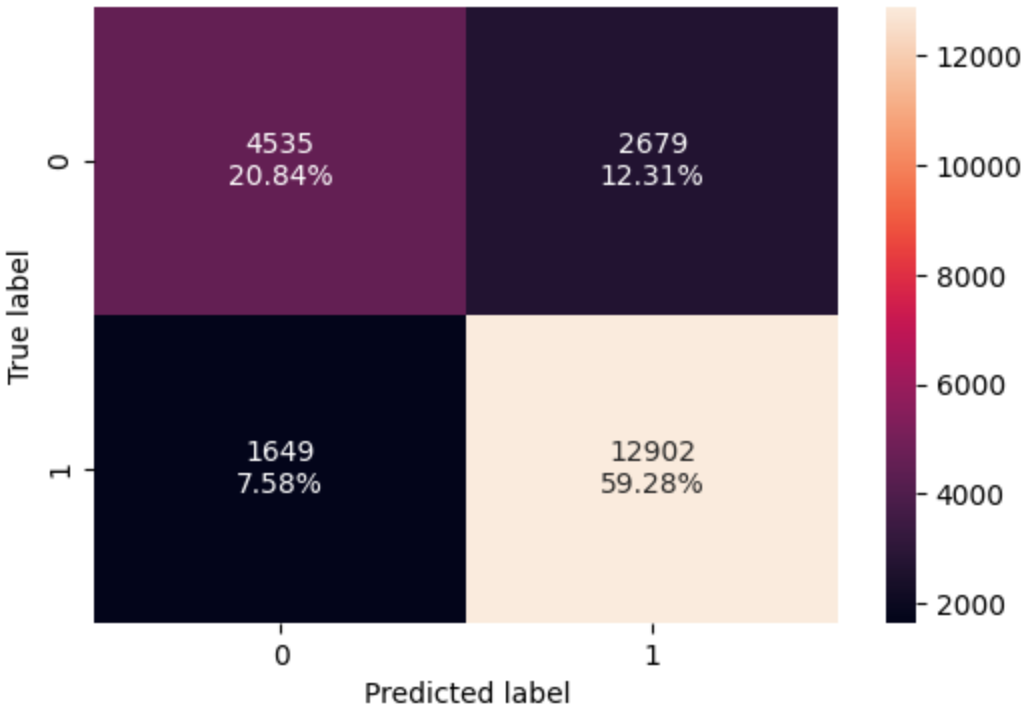
```
Training performance:
```

Out[684…

	Accuracy	Recall	Precision	F1
0	0.80391	0.88929	0.82965	0.85843

In [686…
```
# defining a function to plot the confusion_matrix of a classification model


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.flatten().
            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")
```

In [688…
```
confusion_matrix_statsmodels(lg, X_train, y_train)
```



In [690…
```
print("Training performance:")
model_performance_classification_statsmodels(lg, X_train, y_train)
```

Training performance:

Out[690…

	Accuracy	Recall	Precision	F1
0	0.80115	0.88667	0.82806	0.85637

In [692…
```
print(lg2.summary())
```

Logit Regression Results

Dep. Variable:	Not_Canceled	No. Observations:	21765
Model:	Logit	Df Residuals:	21738
Method:	MLE	Df Model:	26
Date:	Tue, 03 Jun 2025	Pseudo R-squ.:	inf
Time:	19:37:56	Log-Likelihood:	-inf
converged:	False	LL-Null:	0.0000
Covariance Type:	nonrobust	LLR p-value:	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
no_of_adults	-0.1139	0.041	-2.810	0.005	-0.193	-0.034
no_of_children	-0.1587	0.061	-2.601	0.009	-0.278	-0.039
no_of_weekend_nights	-0.1066	0.021	-4.997	0.000	-0.148	-0.065
no_of_week_nights	-0.0376	0.013	-2.832	0.005	-0.064	-0.012
required_car_parking_space	1.6514	0.151	10.919	0.000	1.355	1.948
lead_time	-0.0163	0.000	-58.510	0.000	-0.017	-0.016
arrival_year	0.0012	0.000	8.663	0.000	0.001	0.001
arrival_month	0.0654	0.006	10.085	0.000	0.053	0.078
arrival_date	-0.0001	0.002	-0.063	0.950	-0.004	0.004
repeated_guest	2.2913	0.611	3.752	0.000	1.094	3.488
no_of_previous_cancellations	-0.2803	0.091	-3.089	0.002	-0.458	-0.102
no_of_previous_bookings_not_canceled	0.1216	0.125	0.971	0.332	-0.124	0.367
avg_price_per_room	-0.0206	0.001	-26.484	0.000	-0.022	-0.019
no_of_special_requests	1.4662	0.032	45.234	0.000	1.403	1.530
type_of_meal_plan_Meal Plan 2	-0.0239	0.069	-0.345	0.730	-0.160	0.112
type_of_meal_plan_Meal Plan 3	-15.6124	2116.683	-0.007	0.994	-4164.235	4133.010
type_of_meal_plan_Not Selected	-0.3385	0.057	-5.988	0.000	-0.449	-0.228
room_type_reserved_Room_Type 2	0.3008	0.140	2.147	0.032	0.026	0.575
room_type_reserved_Room_Type 3	-0.0260	1.301	-0.020	0.984	-2.575	2.523
room_type_reserved_Room_Type 4	0.2983	0.057	5.206	0.000		

```
0       0.186       0.411
room_type_reserved_Room_Type 5           0.7453      0.224      3.325      0.00
1       0.306       1.185
room_type_reserved_Room_Type 6           1.0724      0.158      6.809      0.00
0       0.764       1.381
room_type_reserved_Room_Type 7           1.4806      0.322      4.596      0.00
0       0.849       2.112
market_segment_type_Complementary       24.1323   2149.454      0.011      0.99
1   -4188.720    4236.985
market_segment_type_Corporate            1.3625      0.276      4.938      0.00
0       0.822       1.903
market_segment_type_Offline              2.3997      0.263      9.108      0.00
0       1.883       2.916
market_segment_type_Online               0.5722      0.260      2.201      0.02
8       0.063       1.082
==============================================================================
==========================
```

Q9 OBSERVATION

Holding any other variable constant,

- Repeated guest will increase the booking status by 2.2913
- Number of special requests will increase the booking status by 1.4662
- Average price per month will decrease booking status by 0.0206
- Number of previous booking cancellation will decrease booking status by 0.2803

In [693…

```
# initial list of columns
cols = X_train2.columns.tolist()

# setting an initial max p-value
max_p_value = 1

while len(cols) > 0:
    # defining the train set
    X_train_aux = X_train2[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)
```

```
['no_of_adults', 'no_of_children', 'no_of_weekend_nights', 'no_of_week_nights',
'required_car_parking_space', 'lead_time', 'arrival_year', 'arrival_month', 're
peated_guest', 'no_of_previous_cancellations', 'avg_price_per_room', 'no_of_spe
cial_requests', '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_t
ype_reserved_Room_Type 6', 'room_type_reserved_Room_Type 7', 'market_segment_ty
pe_Corporate', 'market_segment_type_Offline']
```

In [695…

```
X_train3 = X_train2[selected_features]
```

In [696…

```
logit3 = sm.Logit(y_train, X_train3.astype(float))
lg3 = logit3.fit(disp=False)

print(lg3.summary())
```

Logit Regression Results

Dep. Variable:	Not_Canceled	No. Observations:	21765
Model:	Logit	Df Residuals:	21745
Method:	MLE	Df Model:	19
Date:	Tue, 03 Jun 2025	Pseudo R-squ.:	inf
Time:	19:38:00	Log-Likelihood:	-inf
converged:	True	LL-Null:	0.0000
Covariance Type:	nonrobust	LLR p-value:	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
no_of_adults	-0.1059	0.040	-2.640	0.008	-0.184	-0.027
no_of_children	-0.1536	0.061	-2.518	0.012	-0.273	-0.034
no_of_weekend_nights	-0.1086	0.021	-5.098	0.000	-0.150	-0.067
no_of_week_nights	-0.0401	0.013	-3.024	0.002	-0.066	-0.014
required_car_parking_space	1.6487	0.151	10.899	0.000	1.352	1.945
lead_time	-0.0163	0.000	-59.084	0.000	-0.017	-0.016
arrival_year	0.0015	5.75e-05	25.611	0.000	0.001	0.002
arrival_month	0.0659	0.006	10.204	0.000	0.053	0.079
repeated_guest	2.5547	0.567	4.505	0.000	1.443	3.666
no_of_previous_cancellations	-0.2467	0.081	-3.046	0.002	-0.405	-0.088
avg_price_per_room	-0.0210	0.001	-28.008	0.000	-0.022	-0.020
no_of_special_requests	1.4682	0.032	45.368	0.000	1.405	1.532
type_of_meal_plan_Not Selected	-0.3428	0.056	-6.080	0.000	-0.453	-0.232
room_type_reserved_Room_Type 2	0.2979	0.140	2.129	0.033	0.024	0.572
room_type_reserved_Room_Type 4	0.2973	0.057	5.222	0.000	0.186	0.409
room_type_reserved_Room_Type 5	0.7640	0.223	3.423	0.001	0.327	1.201
room_type_reserved_Room_Type 6	1.0905	0.157	6.944	0.000	0.783	1.398
room_type_reserved_Room_Type 7	1.5153	0.321	4.720	0.000	0.886	2.145
market_segment_type_Corporate	0.7935	0.110	7.240	0.000	0.579	1.008
market_segment_type_Offline	1.8159	0.054	33.620	0.000		

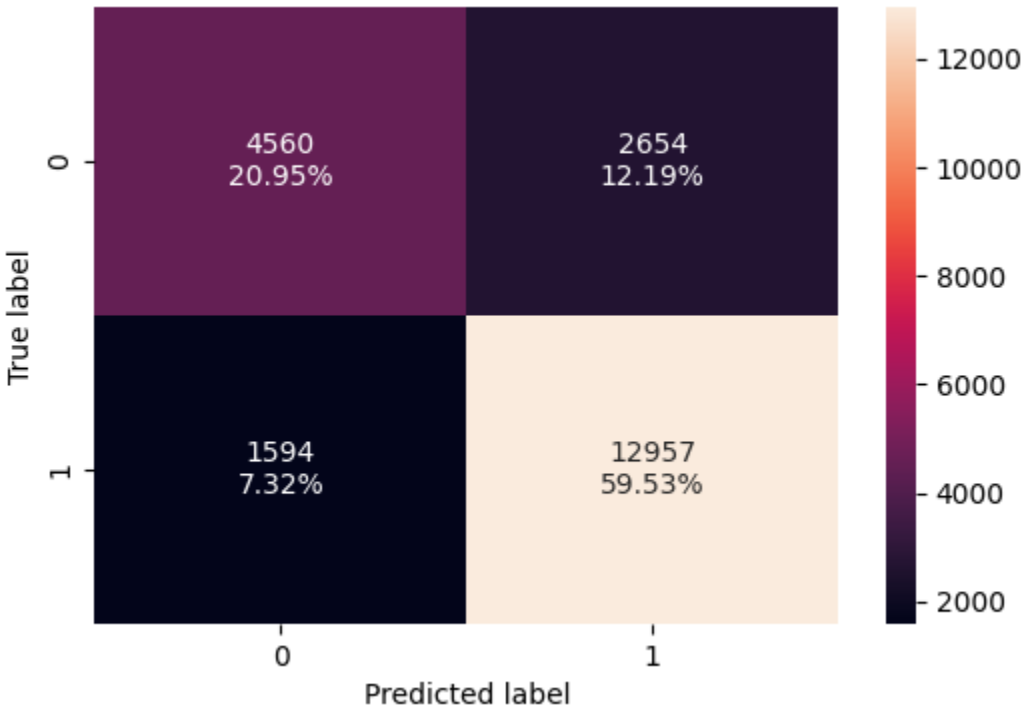
```
1.710       1.922
==============================================================================
==================
```

In [700…
```
# converting coefficients to odds
odds = np.exp(lg3.params)

# finding the percentage change
perc_change_odds = (np.exp(lg3.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_train3.c
```

Out[700…

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_
Odds	0.89956	0.85759	0.89705	0
Change_odd%	-10.04437	-14.24082	-10.29482	-3

In [702…
```
# creating confusion matrix
confusion_matrix_statsmodels(lg3, X_train3, y_train)
```



Q10 OBSERVATION

After treating the P-values and from the conclusion matrix,

- The model was able to predict 1257 making it 59% accurate in predicting true values and a total of 4560 true negative values prediction making it 20.95% accurate in predicting true negative values.
- It predicted that 1594 values are negative which normally are positive and 2654 values to be positive which are supposed to be negative.
- This model ended up with an accuracy of 80.5%, Recall of 89.1% precision of 83.0%, F1-score of 86%.

In [703…
```
log_reg_model_train_perf = model_performance_classification_statsmodels(
    lg3, X_train3, y_train
)

print("Training performance:")
log_reg_model_train_perf
```

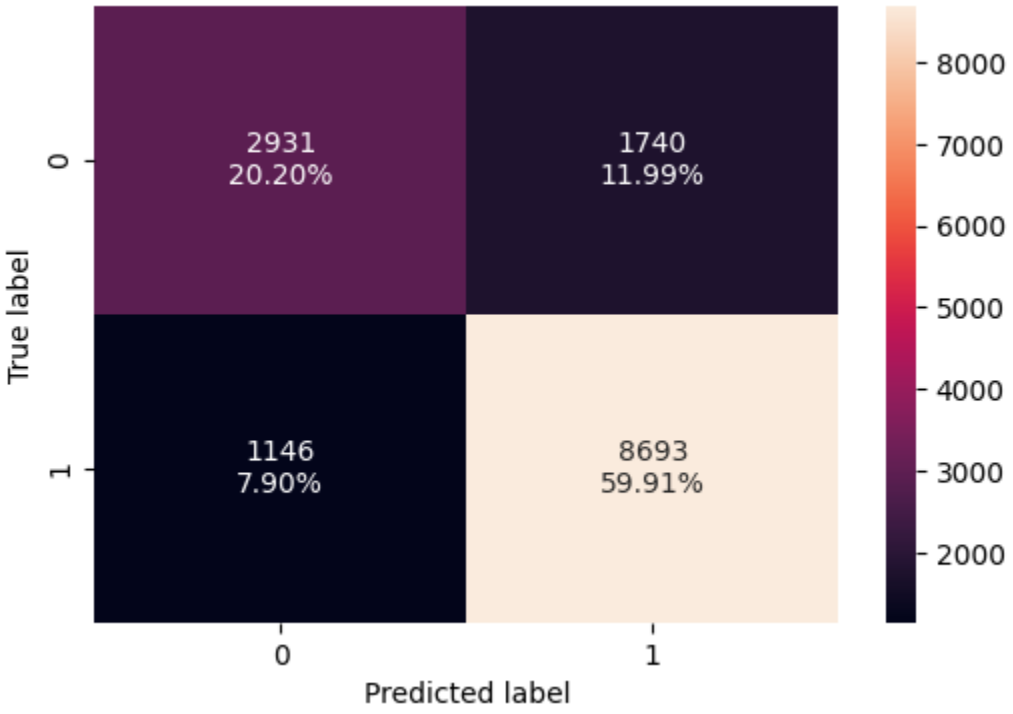
```
Training performance:
```

Out[703…

	Accuracy	Recall	Precision	F1
0	0.80482	0.89045	0.82999	0.85916

In [706…
```
X_test3 = X_test[list(X_train3.columns)]
```

In [708…
```
# creating confusion matrix
confusion_matrix_statsmodels(lg3, X_test3, y_test)
```



In [710…
```
log_reg_model_test_perf = model_performance_classification_statsmodels(
    lg3, X_test3, y_test
)

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

```
Test performance:
```

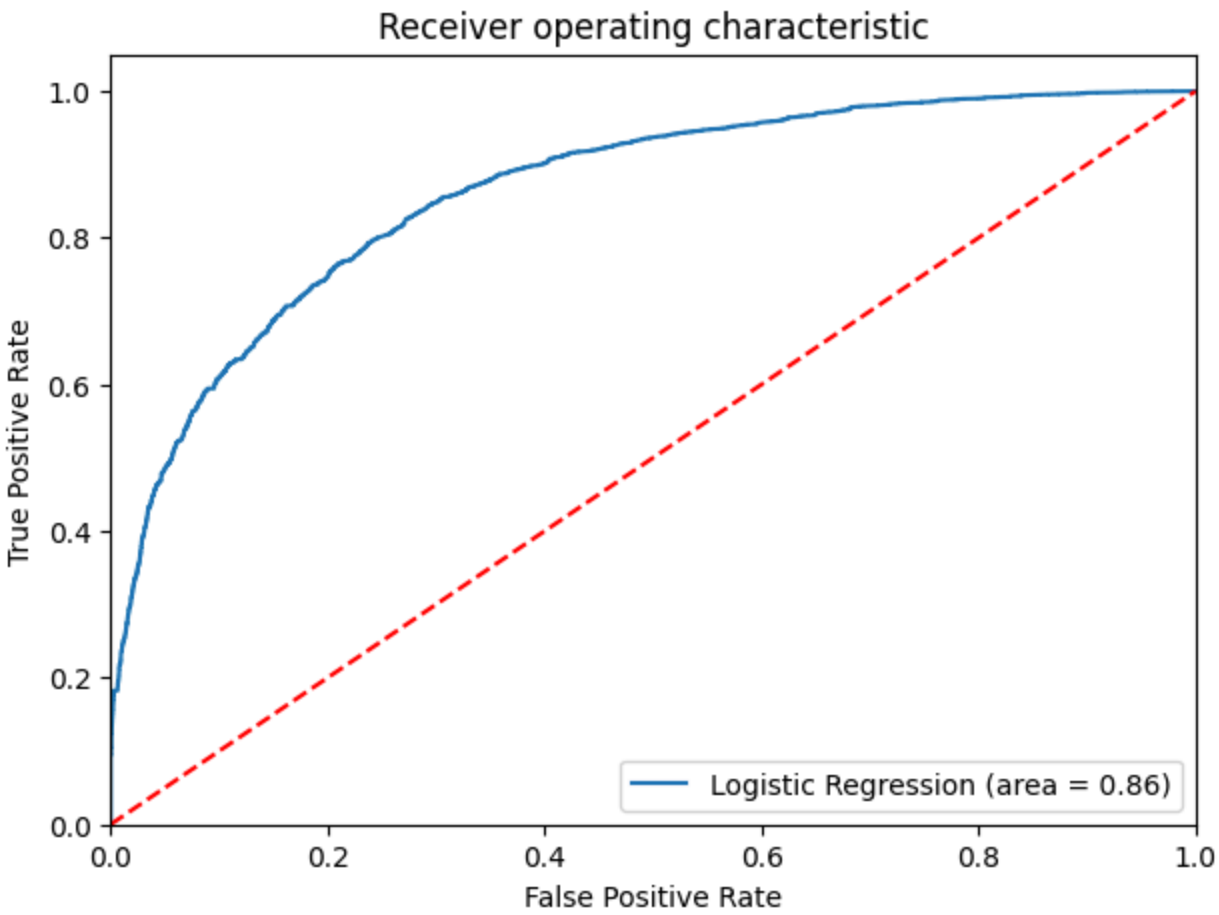
Out[710…

	Accuracy	Recall	Precision	F1
0	0.80110	0.88352	0.83322	0.85764

Q11 OBSERVATION

After testng the model with another test set, the model improved by just 1% in its precision and a decrease in Recall, Accuracy and F1.

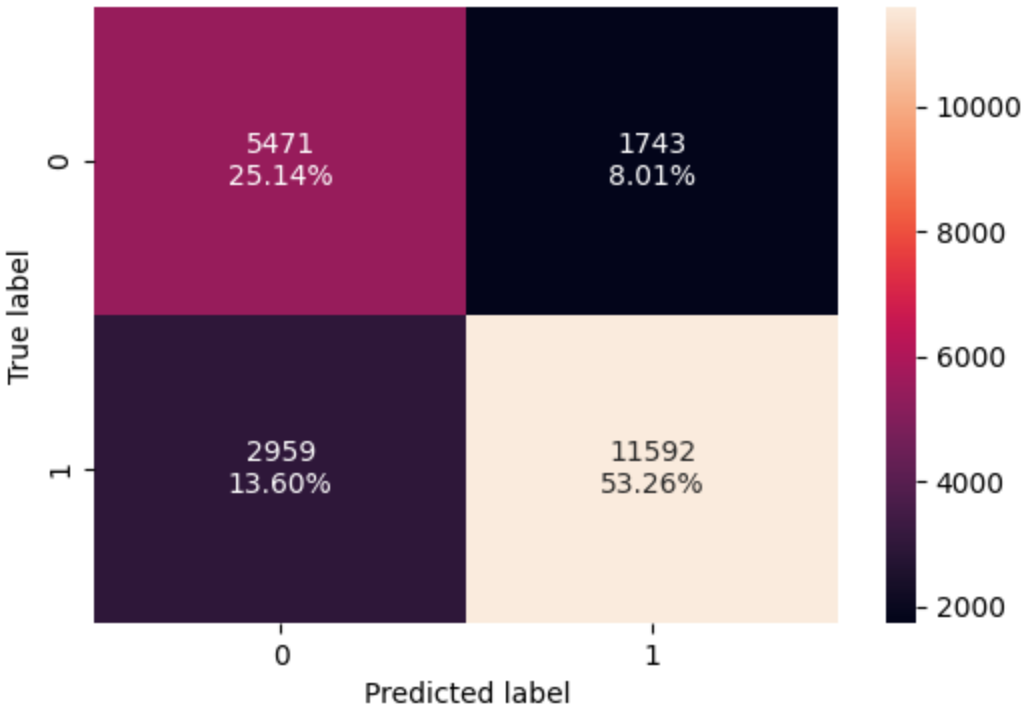
In [712…
```
logit_roc_auc_train = roc_auc_score(y_train, lg3.predict(X_train3))
fpr, tpr, thresholds = roc_curve(y_train, lg3.predict(X_train3))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc_auc_
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



In [714…
```
# 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, lg3.predict(X_train3))

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

```
0.6519068743848913
```

In [716…
```
# creating confusion matrix
confusion_matrix_statsmodels(
    lg3, X_train3, y_train, threshold=optimal_threshold_auc_roc
)
```



In [718…
```
# checking model performance for this model
log_reg_model_train_perf_threshold_auc_roc = model_performance_classification_
    lg3, X_train3, y_train, threshold=optimal_threshold_auc_roc
)
print("Training performance:")
log_reg_model_train_perf_threshold_auc_roc
```

```
Training performance:
```

Out[718…

	Accuracy	Recall	Precision	F1
0	0.78397	0.79665	0.86929	0.83138

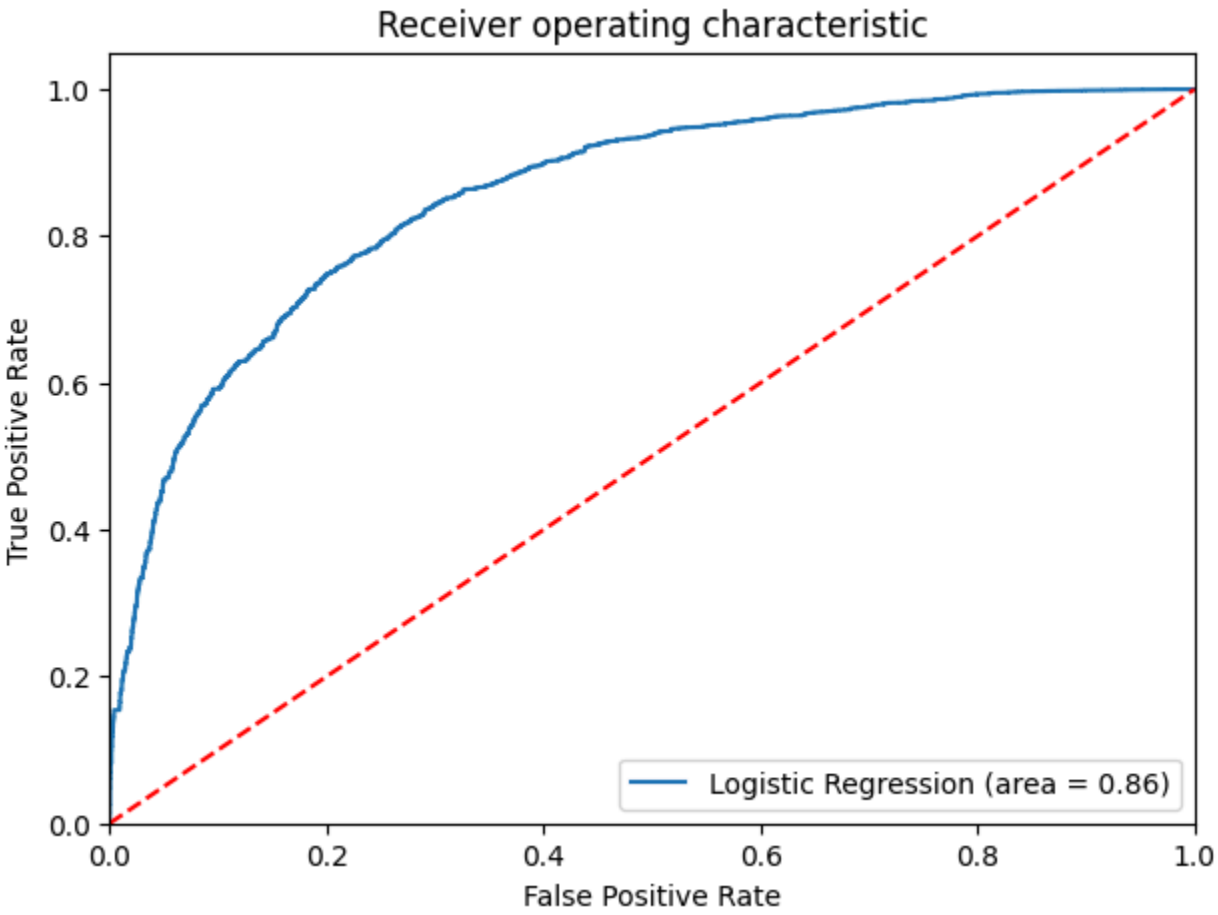
Q12 OBSERVATION

With the ROC,

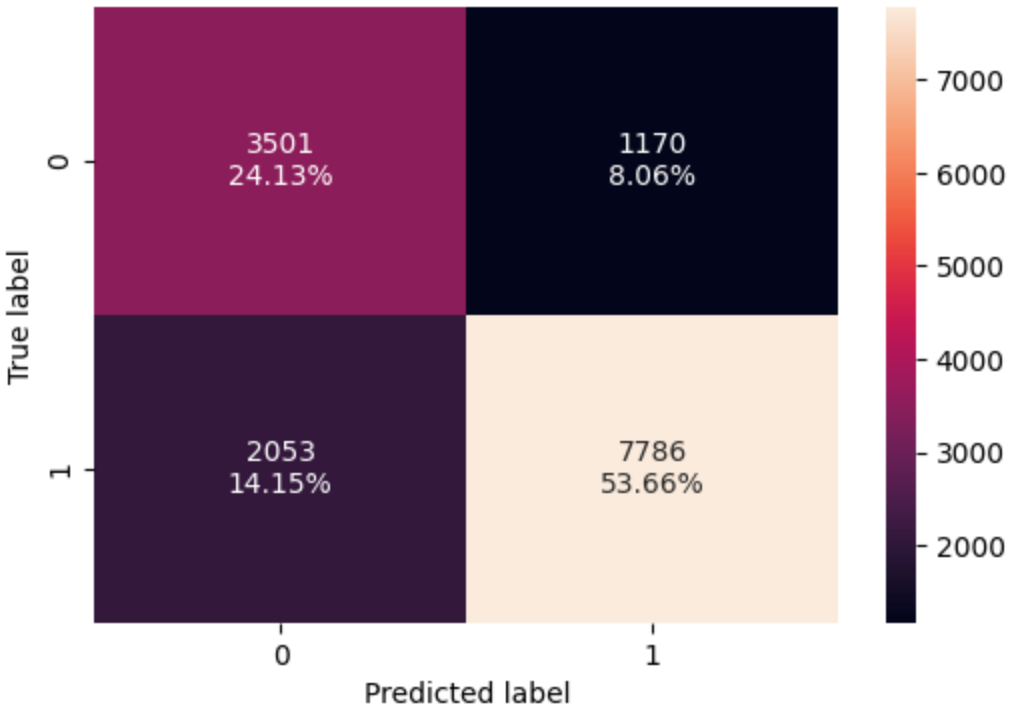
- the model predicted 11592 true positives 5471 true negatives 2943 false positives 1743 false negatives
- This gave it an Accuracy of 78.4% Recall of 79.7% Precision of 87.0% F1 score of 83%

In [720…
```
logit_roc_auc_train = roc_auc_score(y_test, lg3.predict(X_test3))
fpr, tpr, thresholds = roc_curve(y_test, lg3.predict(X_test3))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc_auc_
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



In [722…
```
# creating confusion matrix
confusion_matrix_statsmodels(lg3, X_test3, y_test, threshold=optimal_threshold
```



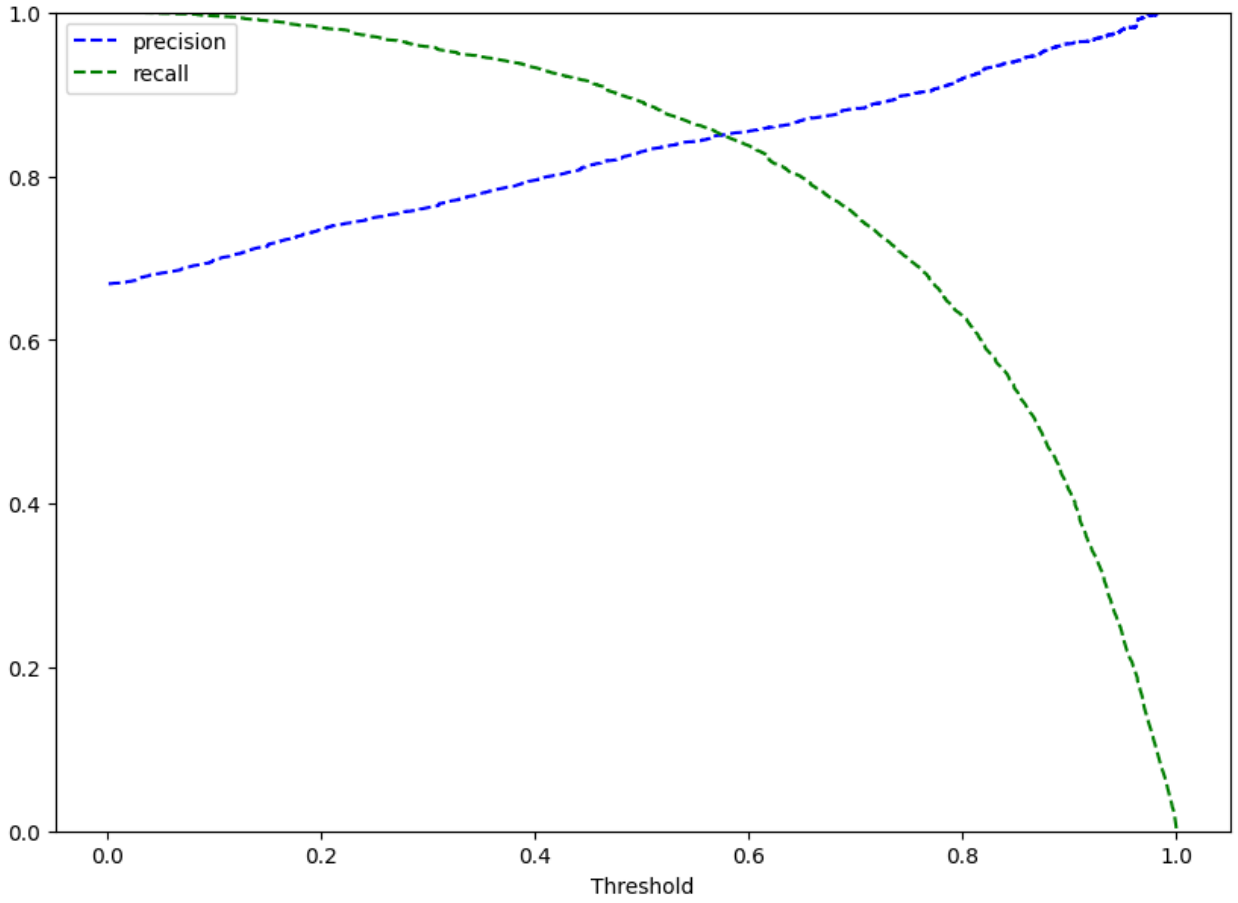
In [724…
```
# checking model performance for this model
log_reg_model_test_perf_threshold_auc_roc = model_performance_classification_s
    lg3, X_test3, y_test, threshold=optimal_threshold_auc_roc
)
print("Test performance:")
log_reg_model_test_perf_threshold_auc_roc
```

Test performance:

Out[724…

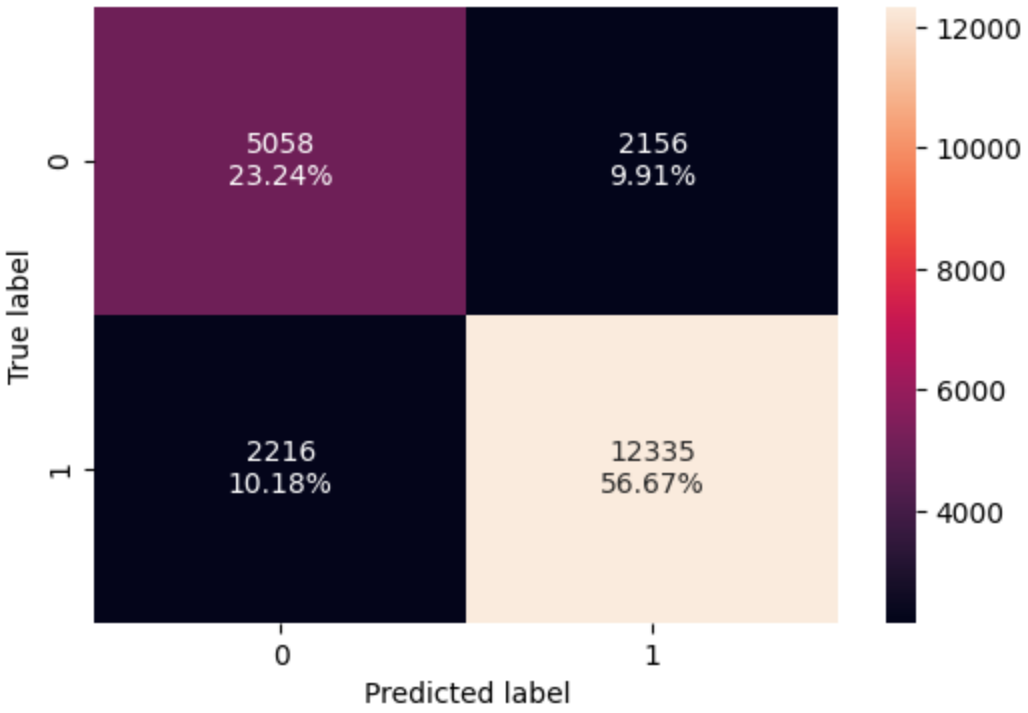
	Accuracy	Recall	Precision	F1
0	0.77788	0.79134	0.86936	0.82852

In [726…
```
y_scores = lg3.predict(X_train3)
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 [281…
```
# setting the threshold
optimal_threshold_curve = 0.58
```

In [283…
```
# creating confusion matrix
confusion_matrix_statsmodels(lg3, X_train3, y_train, threshold=optimal_threshc
```



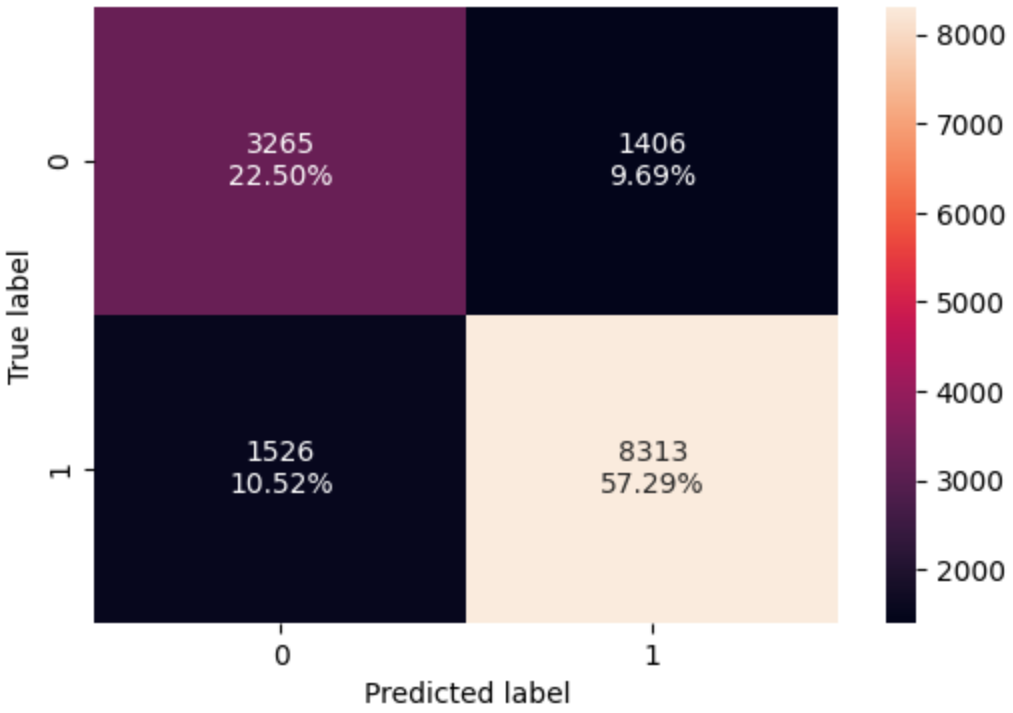
In [285…
```
log_reg_model_train_perf_threshold_curve = model_performance_classification_st
    lg3, X_train3, y_train, threshold=optimal_threshold_curve
)
print("Training performance:")
log_reg_model_train_perf_threshold_curve
```

```
Training performance:
```

Out[285…

	Accuracy	Recall	Precision	F1
0	0.799127	0.847708	0.851218	0.849459

In [287…
```
# creating confusion matrix
confusion_matrix_statsmodels(lg3, X_test3, y_test, threshold=optimal_threshold
```



In [289…
```
log_reg_model_test_perf_threshold_curve = model_performance_classification_sta
    lg3, X_test3, y_test, threshold=optimal_threshold_curve
)
print("Test performance:")
log_reg_model_test_perf_threshold_curve
```

```
Test performance:
```

Out[289…

	Accuracy	Recall	Precision	F1
0	0.797932	0.844903	0.855335	0.850087

In [ ]:
```
```

Final Model Summary

In [291…
```
# 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 (0.5)",
    "Logistic Regression-0.76 Threshold",
```

```
    "Logistic Regression-0.58 Threshold",
]

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

```
Training performance comparison:
```

Out[291…

	Logistic Regression-default Threshold (0.5)	Logistic Regression-0.76 Threshold	Logistic Regression-0.58 Threshold
Accuracy	0.804824	0.783965	0.799127
Recall	0.890454	0.796646	0.847708
Precision	0.829992	0.869291	0.851218
F1	0.859161	0.831385	0.849459

Q13 OBSERVATION

After careful observation on the training set, A recall of 89.1% with a threshold of 0.5 or default shows the highest performance followed by precision with a threshold of 0.76 shows a performance of 86.9%.

In [293…

```
# testing 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 (0.5)",
    "Logistic Regression-0.76 Threshold",
    "Logistic Regression-0.58 Threshold",
]

print("Test set performance comparison:")
models_test_comp_df
```

```
Test set performance comparison:
```

Out[293…

	Logistic Regression-default Threshold (0.5)	Logistic Regression-0.76 Threshold	Logistic Regression-0.58 Threshold
Accuracy	0.803239	0.784425	0.797932
Recall	0.886371	0.798963	0.844903
Precision	0.833907	0.872378	0.855335
F1	0.859339	0.834058	0.850087

Q14 OBSERVATION

On the other hand looking at the test set, Recall had the best performance and will be the best fit with a default threshold (0.5) showing average performance of 88.6% followed by precision with a performance of 87.2% using a threshold of 0.76.

In [ ]:

In [ ]:

In [297…
```
# Libraries to build decision tree classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# To tune different models
from sklearn.model_selection import GridSearchCV
```

Building a Decision Tree model

In [728…
```
model = DecisionTreeClassifier(criterion="gini", random_state=1)
model.fit(X_train, y_train)
```

Out[728…

▾ DecisionTreeClassifier

DecisionTreeClassifier(random_state=1)

In [730…
```
# defining a function to compute different metrics to check performance of a c
def model_performance_classification_sklearn(model, predictors, target):
    """
    Function to compute different metrics to check classification model perfor

    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": f1,}
        index=[0],
    )

    return df_perf
```

In [732…

```
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.flatten().
            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")
```

In [734…

```
decision_tree_perf_train = model_performance_classification_sklearn(
    model, X_train, y_train
)
decision_tree_perf_train
```

Out[734…

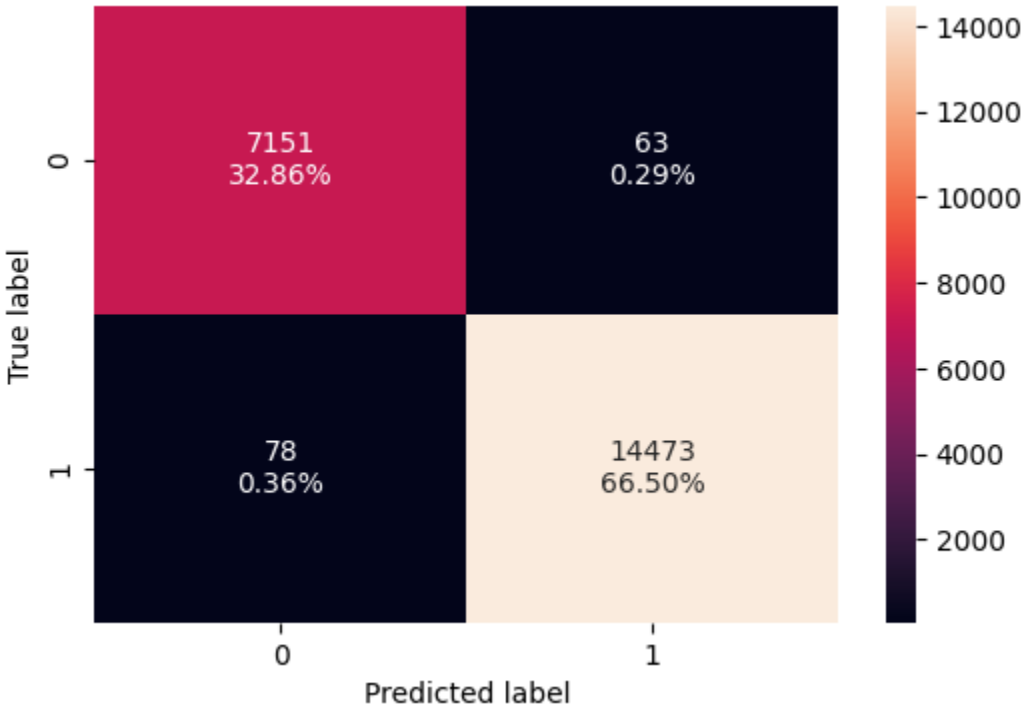
	Accuracy	Recall	Precision	F1
0	0.99352	0.99464	0.99567	0.99515

Q15 OBSERVATION

From the decision tree classifier, It can be observed that the model has a

performance of approximately 99.7% which shows that it is overfitted and needs to be treated.

In [736…
```
confusion_matrix_sklearn(model, X_train, y_train)
```

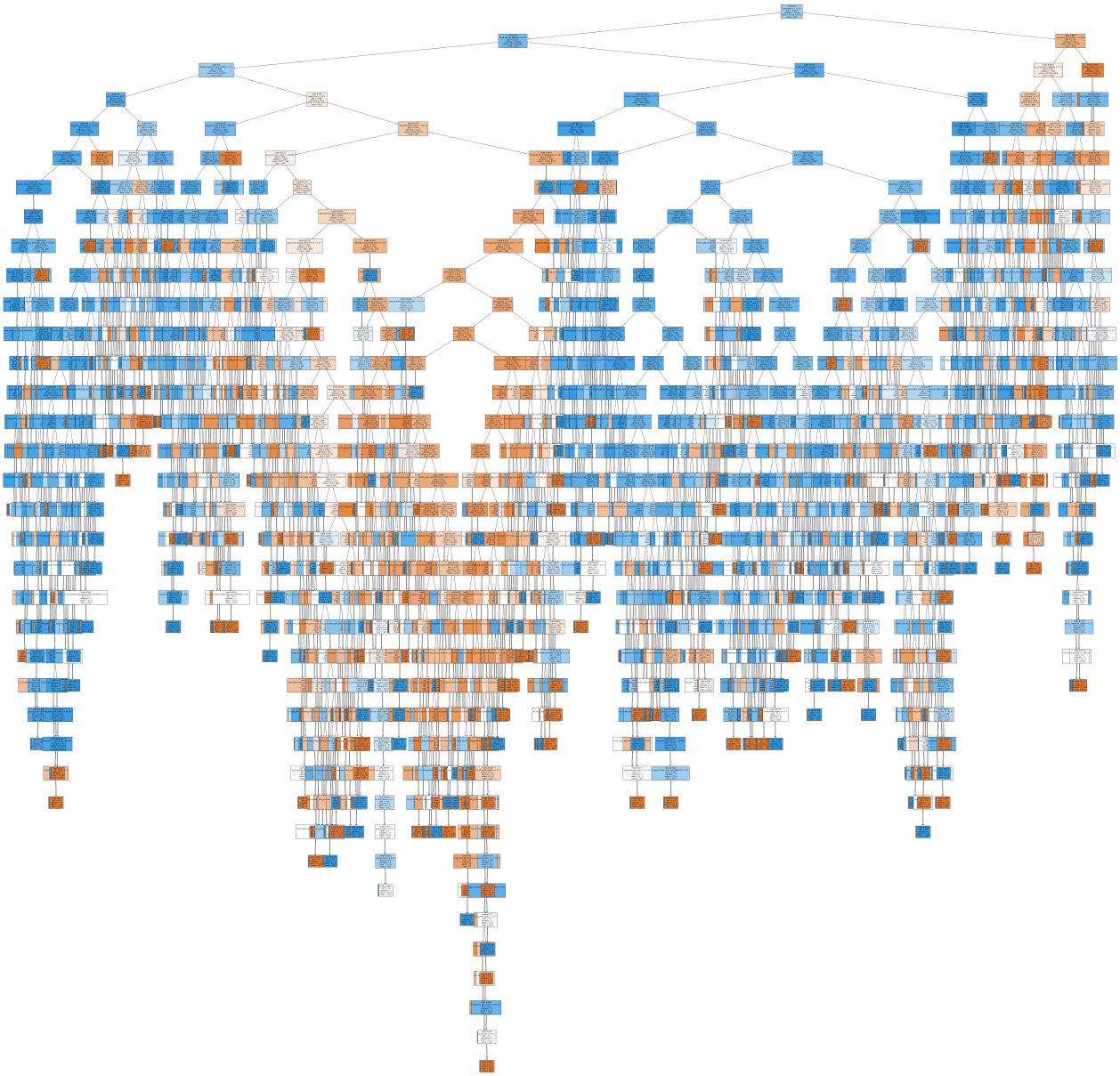


In [314…
```
column_names = list(X.columns)
feature_names = column_names
print(feature_names)
```

```
['no_of_adults', 'no_of_children', 'no_of_weekend_nights', 'no_of_week_nights',
'required_car_parking_space', 'lead_time', 'arrival_year', 'arrival_month', 'ar
rival_date', 'repeated_guest', 'no_of_previous_cancellations', 'no_of_previou
s_bookings_not_canceled', 'avg_price_per_room', 'no_of_special_requests', 'typ
e_of_meal_plan_Meal Plan 2', 'type_of_meal_plan_Meal Plan 3', 'type_of_meal_pla
n_Not Selected', 'room_type_reserved_Room_Type 2', 'room_type_reserved_Room_Typ
e 3', 'room_type_reserved_Room_Type 4', 'room_type_reserved_Room_Type 5', 'roo
m_type_reserved_Room_Type 6', 'room_type_reserved_Room_Type 7', 'market_segmen
t_type_Complementary', 'market_segment_type_Corporate', 'market_segment_type_Of
fline', 'market_segment_type_Online']
```

In [374…
```
plt.figure(figsize=(100, 100))

out = tree.plot_tree(
    model,
    feature_names=feature_names,
    filled=True,
    fontsize=9,
    node_ids=True,
    class_names=True,
)
for o in out:
```

```
    arrow = o.arrow_patch
    if arrow is not None:
        arrow.set_edgecolor("black")
        arrow.set_linewidth(1)
plt.show()
```



In [376…
```
clf = DecisionTreeClassifier(random_state=1)
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
```

In [378…
```
pd.DataFrame(path)
```

Out[378…

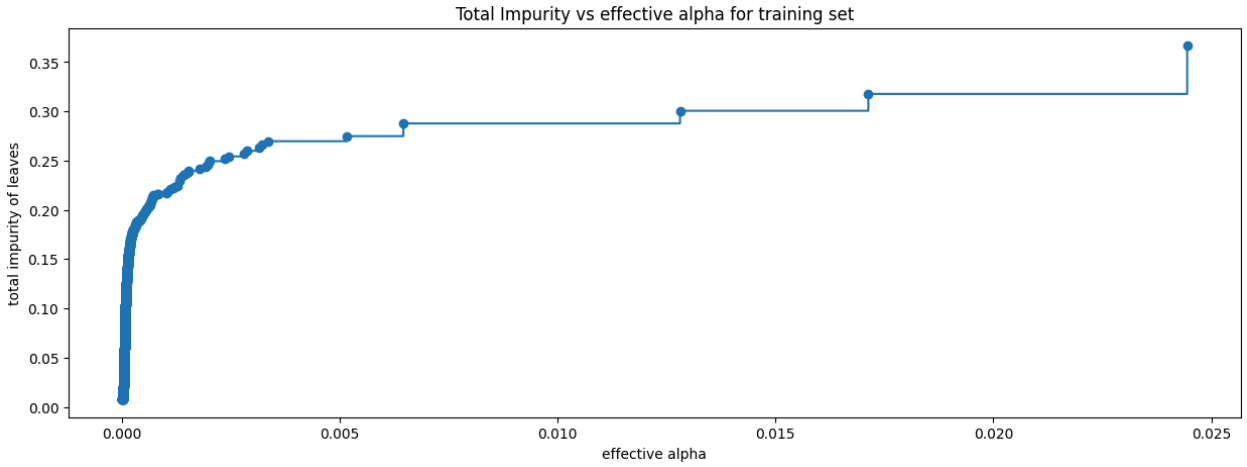
	ccp_alphas	impurities
0	0.000000e+00	0.007551
1	5.543931e-08	0.007552
2	8.204522e-07	0.007552
3	1.055204e-06	0.007554
4	1.747920e-06	0.007556
...	...	...
1199	6.460295e-03	0.287793
1200	1.280944e-02	0.300603
1201	1.713152e-02	0.317734
1202	2.444876e-02	0.366632
1203	7.654983e-02	0.443182

1204 rows × 2 columns

Do we need to prune the tree?

In [381…

```
fig, ax = plt.subplots(figsize=(15, 5))
ax.plot(ccp_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-post")
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()
```



In [382…

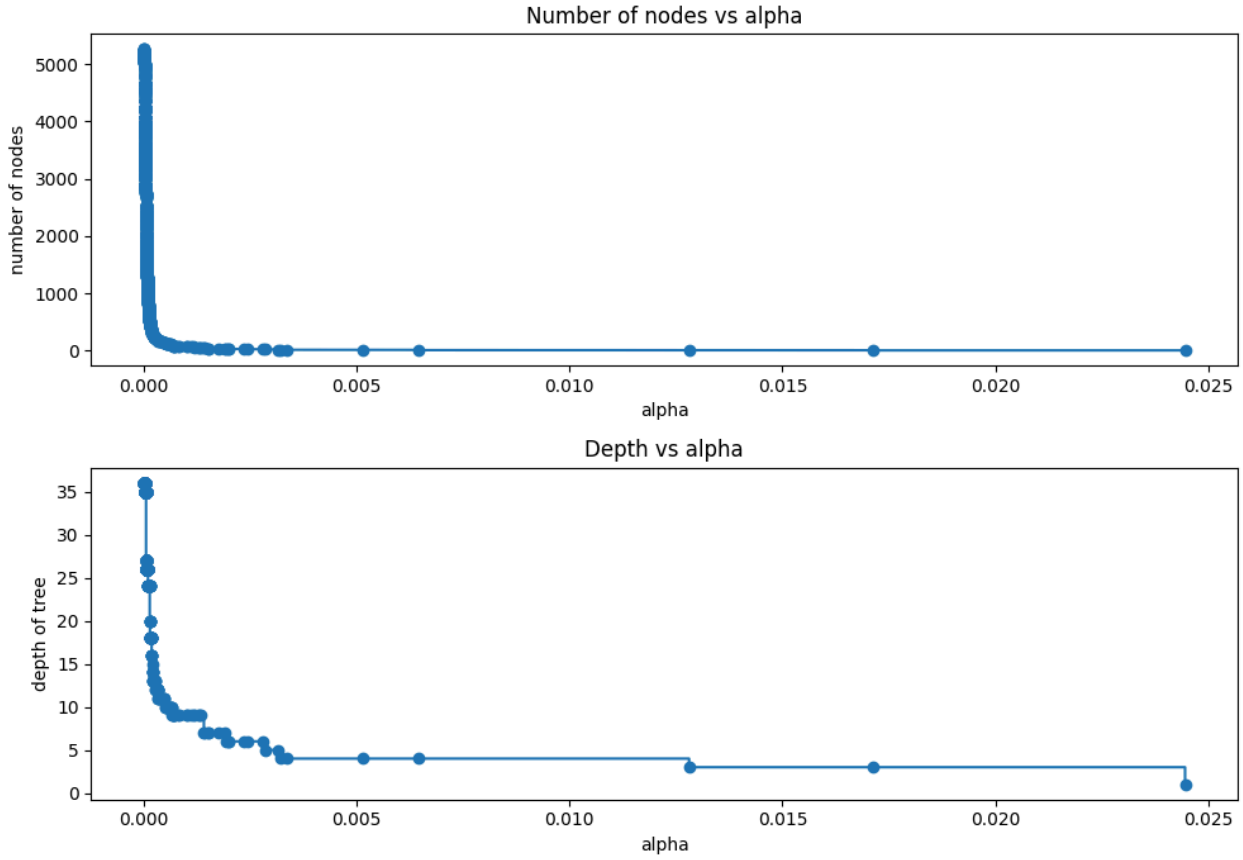
```
clfs = []
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=1, ccp_alpha=ccp_alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)
```

```
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: 0.07654983444812336
```

In [384…

```
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()
```



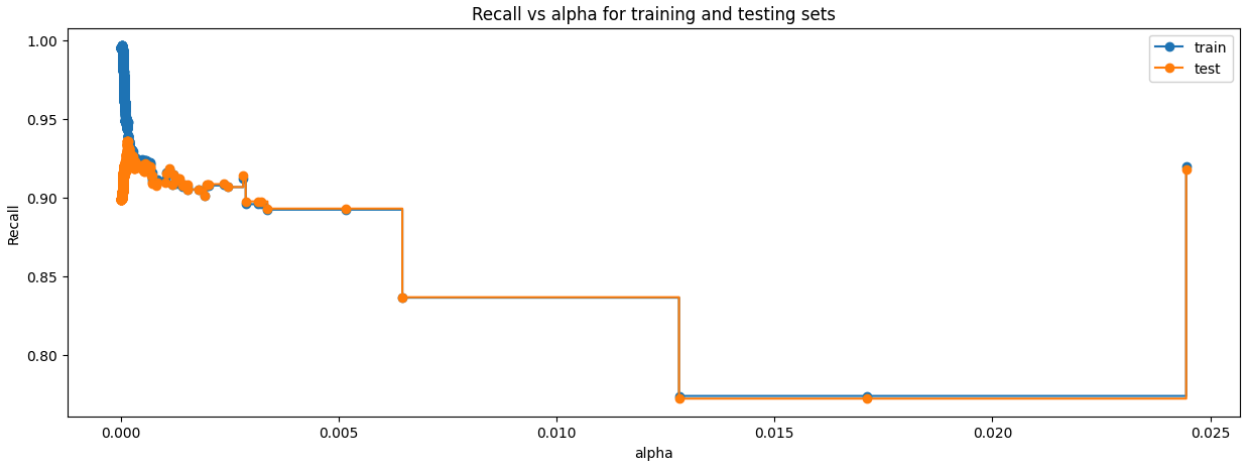
In [385…

```
recall_train = []
for clf in clfs:
    pred_train = clf.predict(X_train)
```

```
    values_train = recall_score(y_train, pred_train)
    recall_train.append(values_train)
```

In [386…
```
recall_test = []
for clf in clfs:
    pred_test = clf.predict(X_test)
    values_test = recall_score(y_test, pred_test)
    recall_test.append(values_test)
```

In [387…
```
fig, ax = plt.subplots(figsize=(15, 5))
ax.set_xlabel("alpha")
ax.set_ylabel("Recall")
ax.set_title("Recall vs alpha for training and testing sets")
ax.plot(ccp_alphas, recall_train, marker="o", label="train", drawstyle="steps-
ax.plot(ccp_alphas, recall_test, marker="o", label="test", drawstyle="steps-po
ax.legend()
plt.show()
```



In [388…
```
# creating the model where we get highest train and test recall
index_best_model = np.argmax(recall_test)
best_model = clfs[index_best_model]
print(best_model)
```

```
DecisionTreeClassifier(ccp_alpha=0.00015171650666790764, random_state=1)
```

In [ ]:

Model Performance Comparison and Conclusions

In [391…
```
decision_tree_postpruned_perf_train = model_performance_classification_sklearn
    best_model, X_train, y_train
)
decision_tree_postpruned_perf_train
```

Out[391…

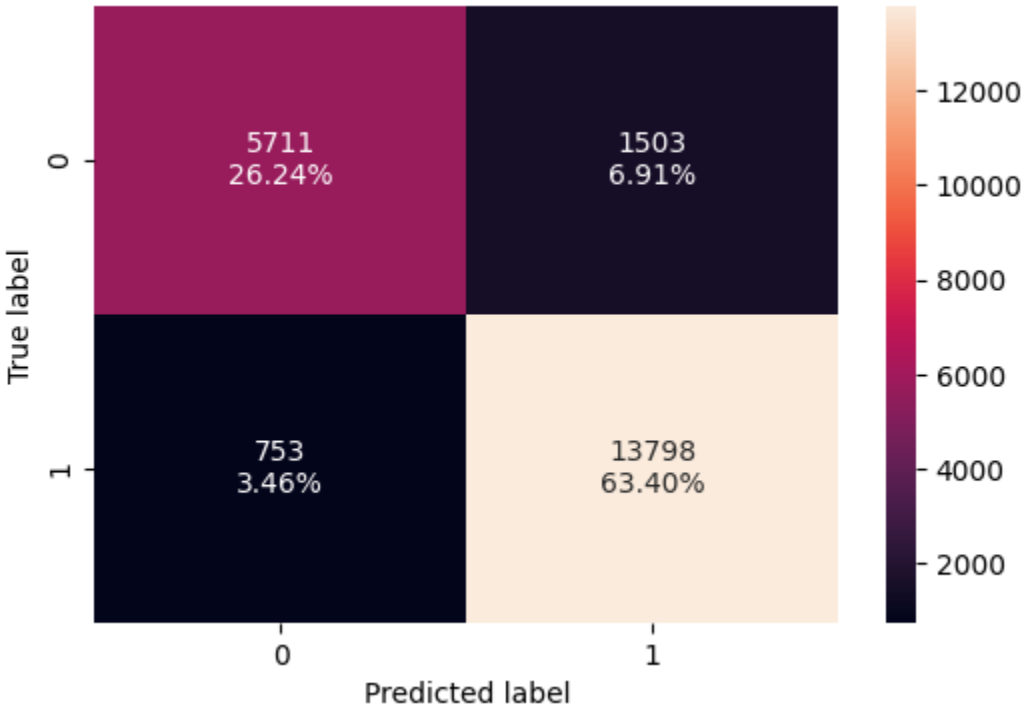
	Accuracy	Recall	Precision	F1
0	0.896347	0.948251	0.901771	0.924427

Q16

- On the training set after pruning, it can be observed that the performance of the model decreased significantly for all the different performance test

In [392…

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



In [ ]:

```
```

In [394…

```
decision_tree_postpruned_perf_test = model_performance_classification_sklearn(
    best_model, X_test, y_test
)
decision_tree_postpruned_perf_test
```

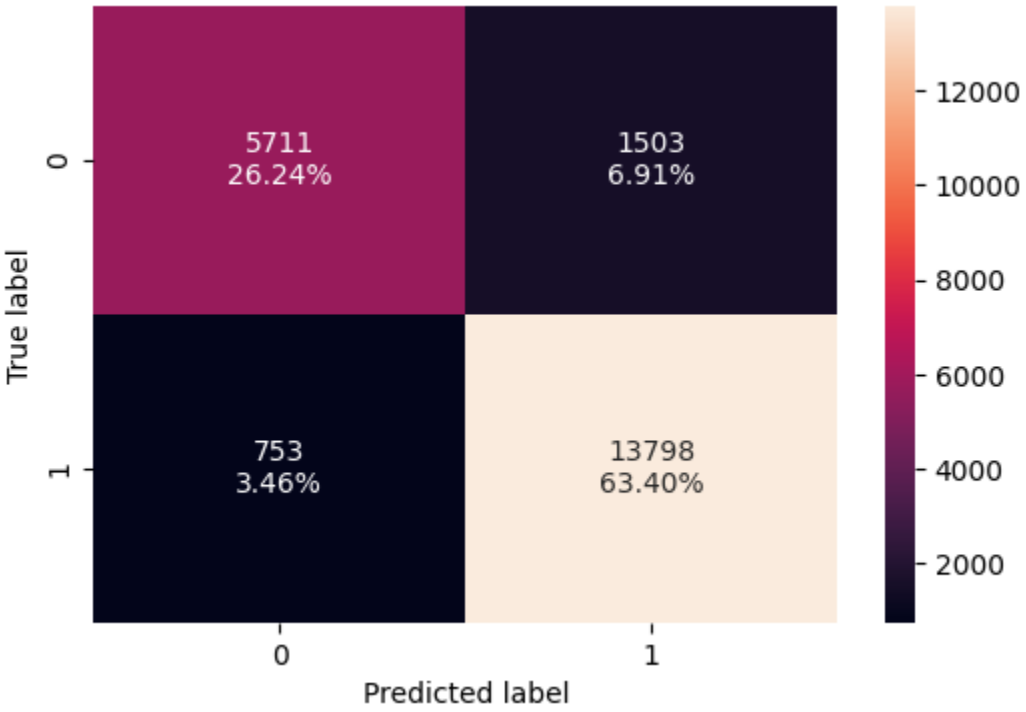
Out[394…

	Accuracy	Recall	Precision	F1
0	0.880152	0.935969	0.892518	0.913727

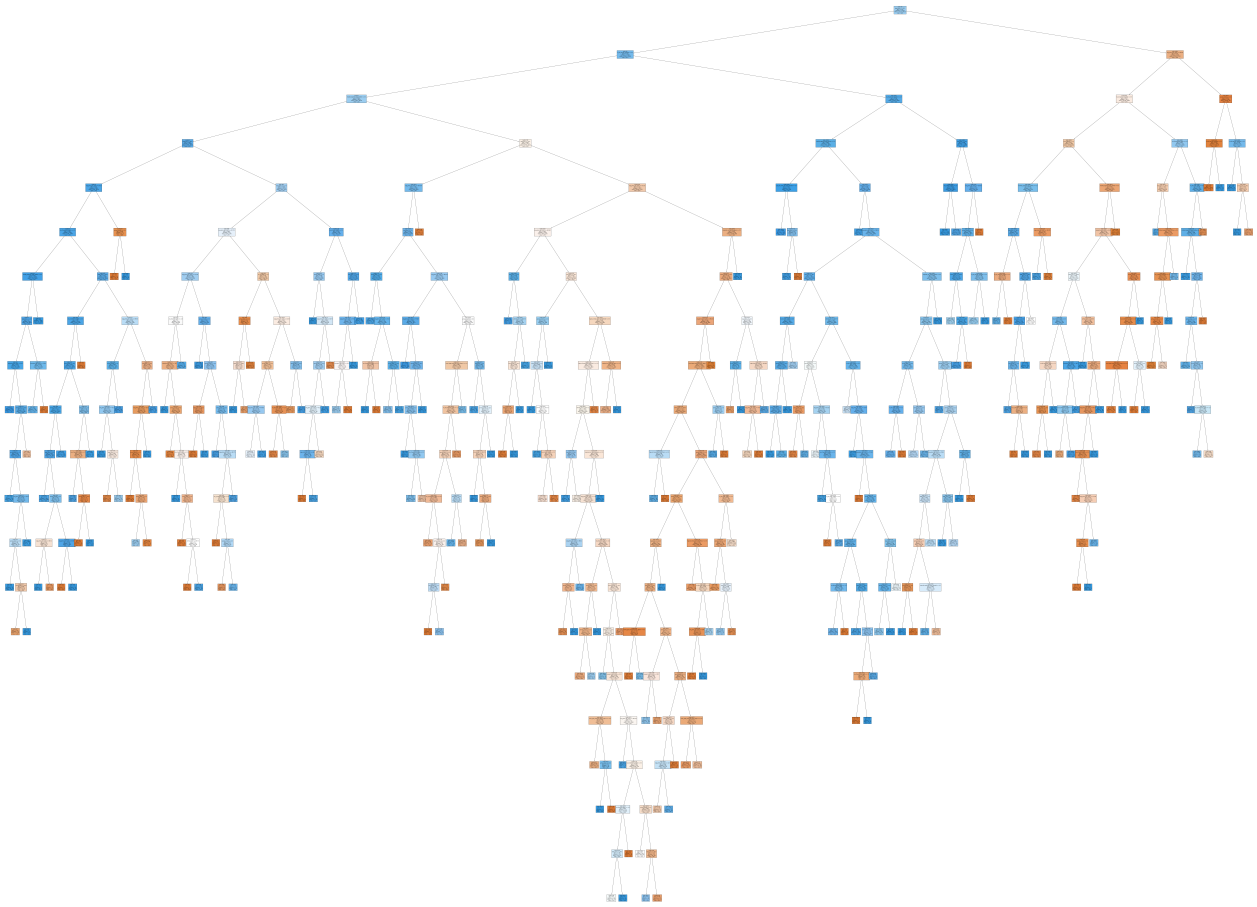
Q17

- after prunning, the model performance droped. indicating it is less overfitted as compared to the previous model that was overfitted.

In [395…
```
confusion_matrix_sklearn(best_model, X_train, y_train)
```



In [738…
```
plt.figure(figsize=(200, 150))

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



In [419…

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

print(tree.export_text(best_model, feature_names=feature_names, show_weights=T
```

```
|--- lead_time <= 151.50
|   |--- no_of_special_requests <= 0.50
|   |   |--- market_segment_type_Online <= 0.50
|   |   |   |--- lead_time <= 90.50
|   |   |   |   |--- avg_price_per_room <= 201.50
|   |   |   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |   |   |--- lead_time <= 16.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 68.50
|   |   |   |   |   |   |   |   |   |--- weights: [6.00, 237.00] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  68.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 29.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [15.00, 194.00] clas
s: 1
|   |   |   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |--- arrival_date >  29.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [4.00, 2.00] class: 0
|   |   |   |   |   |   |   |   |--- lead_time >  16.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 135.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [31.00, 134.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  135.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [7.00, 0.00] class: 0
|   |   |   |   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 1385.00] class: 1
|   |   |   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |   |   |--- lead_time <= 68.50
|   |   |   |   |   |   |   |--- no_of_weekend_nights <= 4.50
|   |   |   |   |   |   |   |   |--- arrival_date <= 27.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 9.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 59.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [58.00, 548.00] clas
s: 1
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  59.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |--- arrival_month >  9.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [17.00, 442.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_date >  27.50
|   |   |   |   |   |   |   |   |   |--- lead_time <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [1.00, 8.00] class: 1
|   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |--- lead_time >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 72.00] class: 1
|   |   |   |   |   |   |   |--- no_of_weekend_nights >  4.50
|   |   |   |   |   |   |   |   |--- weights: [8.00, 0.00] class: 0
|   |   |   |   |   |   |--- lead_time >  68.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 99.98
|   |   |   |   |   |   |   |   |--- arrival_month <= 3.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 62.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 20.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  62.50
```

```
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time <= 77.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [9.00, 1.00] class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time >  77.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [6.00, 12.00] class:
1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  3.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [6.00, 82.00] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  99.98
|   |   |   |   |   |   |   |   |   |--- lead_time <= 81.00
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 123.25
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [50.00, 0.00] class:
0
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  123.25
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 8.00] class: 1
|   |   |   |   |   |   |   |   |   |--- lead_time >  81.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [1.00, 14.00] class: 1
|   |   |   |   |   |--- avg_price_per_room >  201.50
|   |   |   |   |   |   |--- arrival_month <= 10.50
|   |   |   |   |   |   |   |--- weights: [15.00, 0.00] class: 0
|   |   |   |   |   |   |--- arrival_month >  10.50
|   |   |   |   |   |   |   |--- weights: [0.00, 2.00] class: 1
|   |   |   |   |--- lead_time >  90.50
|   |   |   |   |   |--- lead_time <= 117.50
|   |   |   |   |   |   |--- avg_price_per_room <= 92.72
|   |   |   |   |   |   |   |--- avg_price_per_room <= 75.38
|   |   |   |   |   |   |   |   |--- no_of_week_nights <= 2.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 58.75
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 8.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  58.75
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 4.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [61.00, 3.00] class: 0
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  4.50
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 11.00] class:
1
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |--- no_of_week_nights >  2.50
|   |   |   |   |   |   |   |   |   |--- weights: [10.00, 64.00] class: 1
|   |   |   |   |   |   |   |--- avg_price_per_room >  75.38
|   |   |   |   |   |   |   |   |--- arrival_month <= 3.50
|   |   |   |   |   |   |   |   |   |--- weights: [1.00, 69.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  3.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 4.50
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 102.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [10.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  102.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 2.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  4.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 15.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [3.00, 45.00] class: 1
```

```
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  15.50
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 85.38
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  85.38
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 16.00] class:
1
|   |   |   |   |   |--- avg_price_per_room >  92.72
|   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |--- arrival_date <= 12.00
|   |   |   |   |   |   |   |   |--- lead_time <= 108.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.00] class: 1
|   |   |   |   |   |   |   |   |--- lead_time >  108.50
|   |   |   |   |   |   |   |   |   |--- weights: [8.00, 2.00] class: 0
|   |   |   |   |   |   |   |--- arrival_date >  12.00
|   |   |   |   |   |   |   |   |--- weights: [57.00, 1.00] class: 0
|   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 108.50
|   |   |   |   |   |   |   |   |--- arrival_date <= 14.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 96.61
|   |   |   |   |   |   |   |   |   |   |--- weights: [14.00, 16.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  96.61
|   |   |   |   |   |   |   |   |   |   |--- weights: [1.00, 16.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_date >  14.50
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 2.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [69.00, 5.00] class: 0
|   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  2.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 5.00] class: 1
|   |   |   |   |   |   |   |--- avg_price_per_room >  108.50
|   |   |   |   |   |   |   |   |--- arrival_date <= 12.50
|   |   |   |   |   |   |   |   |   |--- weights: [14.00, 3.00] class: 0
|   |   |   |   |   |   |   |   |--- arrival_date >  12.50
|   |   |   |   |   |   |   |   |   |--- weights: [3.00, 43.00] class: 1
|   |   |   |   |   |--- lead_time >  117.50
|   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |--- arrival_date <= 7.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 39.00] class: 1
|   |   |   |   |   |   |   |--- arrival_date >  7.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 94.75
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 22.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 92.25
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 68.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [3.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  68.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 18.00] class:
1
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  92.25
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [26.00, 18.00] class: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  22.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [1.00, 25.00] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  94.75
|   |   |   |   |   |   |   |   |   |--- weights: [17.00, 2.00] class: 0
|   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |--- lead_time <= 125.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 90.85
```

```
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 87.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [9.00, 18.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  87.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [10.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  90.85
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 50.00] class: 1
|   |   |   |   |   |   |   |--- lead_time >  125.50
|   |   |   |   |   |   |   |   |--- weights: [12.00, 197.00] class: 1
|   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |--- lead_time <= 13.50
|   |   |   |   |   |--- avg_price_per_room <= 200.38
|   |   |   |   |   |   |--- lead_time <= 3.50
|   |   |   |   |   |   |   |--- arrival_date <= 20.50
|   |   |   |   |   |   |   |   |--- weights: [33.00, 366.00] class: 1
|   |   |   |   |   |   |   |--- arrival_date >  20.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 61.60
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.00] class: 1
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [9.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  61.60
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [20.00, 53.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [8.00, 97.00] class: 1
|   |   |   |   |   |   |--- lead_time >  3.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 99.38
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 77.51
|   |   |   |   |   |   |   |   |   |--- weights: [2.00, 100.00] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  77.51
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 19.00] class: 1
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [28.00, 81.00] class:
1
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [11.00, 6.00] class:
0
|   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 35.00] class: 1
|   |   |   |   |   |   |   |--- avg_price_per_room >  99.38
|   |   |   |   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 6 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [35.00, 8.00] class: 0
|   |   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 6 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 6.00] class: 1
```

```
|   |   |   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |   |   |--- lead_time <= 9.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [6.00, 59.00] class: 1
|   |   |   |   |   |   |   |   |   |--- lead_time >  9.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.00] class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 9.00] class: 1
|   |   |   |   |   |--- avg_price_per_room >  200.38
|   |   |   |   |   |   |--- weights: [28.00, 1.00] class: 0
|   |   |   |   |--- lead_time >  13.50
|   |   |   |   |   |--- avg_price_per_room <= 105.27
|   |   |   |   |   |   |--- avg_price_per_room <= 60.07
|   |   |   |   |   |   |   |--- lead_time <= 84.50
|   |   |   |   |   |   |   |   |--- weights: [4.00, 61.00] class: 1
|   |   |   |   |   |   |   |--- lead_time >  84.50
|   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 27.00
|   |   |   |   |   |   |   |   |   |   |--- lead_time <= 131.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [10.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |   |--- lead_time >  131.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 3.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  27.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 5.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |--- weights: [1.00, 12.00] class: 1
|   |   |   |   |   |   |--- avg_price_per_room >  60.07
|   |   |   |   |   |   |   |--- lead_time <= 25.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 27.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 26.00] class: 1
|   |   |   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [58.00, 46.00] class:
0
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [13.00, 1.00] class:
0
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 49.00] class: 1
|   |   |   |   |   |   |   |--- lead_time >  25.50
|   |   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected <= 0.50
|   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Meal Plan 2 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time <= 60.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 46.00] class:
1
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time >  60.50
```

```
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [35.00, 31.00] class:
0
|   |   |   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 10
|   |   |   |   |   |   |   |   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 12.00] class:
1
|   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Meal Plan 2 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [28.00, 2.00] class: 0
|   |   |   |   |   |   |   |   |--- type_of_meal_plan_Not Selected >  0.50
|   |   |   |   |   |   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [271.00, 108.00] class: 0
|   |   |   |   |   |   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 4.00] class: 1
|   |   |   |   |--- avg_price_per_room >  105.27
|   |   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |   |--- arrival_month <= 10.50
|   |   |   |   |   |   |   |--- avg_price_per_room <= 195.43
|   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 5 <= 0.50
|   |   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Meal Plan 2 <=
0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [3.00, 15.00] class:
1
|   |   |   |   |   |   |   |   |   |   |--- type_of_meal_plan_Meal Plan 2 >
0.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [7.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 8
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 5 >  0.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 27.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 14.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_date >  27.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [5.00, 0.00] class: 0
|   |   |   |   |   |   |   |--- avg_price_per_room >  195.43
|   |   |   |   |   |   |   |   |--- weights: [83.00, 1.00] class: 0
|   |   |   |   |   |   |--- arrival_month >  10.50
|   |   |   |   |   |   |   |--- lead_time <= 22.50
|   |   |   |   |   |   |   |   |--- no_of_adults <= 1.50
|   |   |   |   |   |   |   |   |   |--- weights: [4.00, 1.00] class: 0
|   |   |   |   |   |   |   |   |--- no_of_adults >  1.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 21.00] class: 1
|   |   |   |   |   |   |   |--- lead_time >  22.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 168.06
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 146.90
|   |   |   |   |   |   |   |   |   |   |--- weights: [32.00, 22.00] class: 0
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  146.90
|   |   |   |   |   |   |   |   |   |   |--- weights: [14.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  168.06
|   |   |   |   |   |   |   |   |   |--- weights: [2.00, 12.00] class: 1
```

```
|   |   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |   |--- weights: [0.00, 31.00] class: 1
|   |--- no_of_special_requests >  0.50
|   |   |--- no_of_special_requests <= 1.50
|   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |--- type_of_meal_plan_Not Selected <= 0.50
|   |   |   |   |   |--- weights: [18.00, 900.00] class: 1
|   |   |   |   |--- type_of_meal_plan_Not Selected >  0.50
|   |   |   |   |   |--- lead_time <= 63.00
|   |   |   |   |   |   |--- weights: [1.00, 19.00] class: 1
|   |   |   |   |   |--- lead_time >  63.00
|   |   |   |   |   |   |--- weights: [6.00, 0.00] class: 0
|   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |--- lead_time <= 6.50
|   |   |   |   |   |--- weights: [40.00, 753.00] class: 1
|   |   |   |   |--- lead_time >  6.50
|   |   |   |   |   |--- avg_price_per_room <= 118.54
|   |   |   |   |   |   |--- lead_time <= 61.50
|   |   |   |   |   |   |   |--- no_of_week_nights <= 4.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 81.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [172.00, 950.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 154.00] class: 1
|   |   |   |   |   |   |   |--- no_of_week_nights >  4.50
|   |   |   |   |   |   |   |   |--- weights: [25.00, 47.00] class: 1
|   |   |   |   |   |   |--- lead_time >  61.50
|   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 7.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 25.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [29.00, 2.00] class: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  25.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [1.00, 5.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  7.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 85.25
|   |   |   |   |   |   |   |   |   |   |--- weights: [22.00, 24.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  85.25
|   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 25.00] class:
1
|   |   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |--- no_of_week_nights <= 0.50
|   |   |   |   |   |   |   |   |   |--- weights: [21.00, 29.00] class: 1
|   |   |   |   |   |   |   |   |--- no_of_week_nights >  0.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 71.87
|   |   |   |   |   |   |   |   |   |   |--- weights: [2.00, 74.00] class: 1
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  71.87
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 72.19
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [3.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  72.19
```

```
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |--- avg_price_per_room >  118.54
|   |   |   |   |   |   |   |--- required_car_parking_space <= 0.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 19.50
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 7.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [80.00, 367.00] class: 1
|   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  7.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [3.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  19.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 27.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [72.00, 114.00] class: 1
|   |   |   |   |   |   |   |   |   |   |--- arrival_date >  27.50
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [22.00, 83.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [6.00, 52.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 159.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  159.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time <= 100.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 62.00] class:
1
|   |   |   |   |   |   |   |   |   |   |   |--- lead_time >  100.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [12.00, 0.00] class:
0
|   |   |   |   |   |   |   |--- required_car_parking_space >  0.50
|   |   |   |   |   |   |   |   |--- weights: [1.00, 89.00] class: 1
|   |   |   |--- no_of_special_requests >  1.50
|   |   |   |   |--- lead_time <= 90.50
|   |   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |   |--- weights: [0.00, 1799.00] class: 1
|   |   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |   |--- weights: [31.00, 262.00] class: 1
|   |   |   |   |--- lead_time >  90.50
|   |   |   |   |   |--- avg_price_per_room <= 202.95
|   |   |   |   |   |   |--- arrival_month <= 8.50
|   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |--- weights: [7.00, 13.00] class: 1
|   |   |   |   |   |   |   |--- arrival_year >  2017.50
|   |   |   |   |   |   |   |   |--- lead_time <= 150.50
|   |   |   |   |   |   |   |   |   |--- weights: [16.00, 233.00] class: 1
|   |   |   |   |   |   |   |   |--- lead_time >  150.50
|   |   |   |   |   |   |   |   |   |--- weights: [3.00, 0.00] class: 0
|   |   |   |   |   |   |--- arrival_month >  8.50
|   |   |   |   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |   |   |   |--- weights: [57.00, 116.00] class: 1
|   |   |   |   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 45.00] class: 1
|   |   |   |   |--- avg_price_per_room >  202.95
```

```
|   |   |   |   |   |--- weights: [7.00, 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
|   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |--- weights: [1.00, 5.00] class: 1
|   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |--- weights: [13.00, 0.00] class: 0
|   |   |   |   |   |--- lead_time >  163.50
|   |   |   |   |   |   |--- lead_time <= 341.00
|   |   |   |   |   |   |   |--- lead_time <= 173.00
|   |   |   |   |   |   |   |   |--- arrival_date <= 3.50
|   |   |   |   |   |   |   |   |   |--- weights: [6.00, 51.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_date >  3.50
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [8.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 3.00] class: 1
|   |   |   |   |   |   |   |--- lead_time >  173.00
|   |   |   |   |   |   |   |   |--- weights: [7.00, 227.00] class: 1
|   |   |   |   |   |   |--- lead_time >  341.00
|   |   |   |   |   |   |   |--- weights: [16.00, 16.00] class: 0
|   |   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |   |--- avg_price_per_room <= 30.53
|   |   |   |   |   |   |--- weights: [1.00, 11.00] class: 1
|   |   |   |   |   |--- avg_price_per_room >  30.53
|   |   |   |   |   |   |--- weights: [54.00, 1.00] class: 0
|   |   |   |--- no_of_adults >  1.50
|   |   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |   |--- avg_price_per_room <= 84.58
|   |   |   |   |   |   |--- lead_time <= 244.00
|   |   |   |   |   |   |   |--- no_of_week_nights <= 1.50
|   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |--- arrival_date <= 19.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [30.00, 2.00] class: 0
|   |   |   |   |   |   |   |   |   |--- arrival_date >  19.00
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 2.00] class: 1
|   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |--- weights: [0.00, 17.00] class: 1
|   |   |   |   |   |   |   |--- no_of_week_nights >  1.50
|   |   |   |   |   |   |   |   |--- avg_price_per_room <= 66.50
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [6.00, 2.00] class: 0
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [1.00, 14.00] class: 1
|   |   |   |   |   |   |   |   |--- avg_price_per_room >  66.50
|   |   |   |   |   |   |   |   |   |--- weights: [7.00, 113.00] class: 1
|   |   |   |   |   |   |--- lead_time >  244.00
|   |   |   |   |   |   |   |--- arrival_year <= 2017.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 30.00] class: 1
|   |   |   |   |   |   |   |--- arrival_year >  2017.50
```

```
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 80.38
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights <= 3.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- weights: [156.00, 11.00] clas
s: 0
|   |   |   |   |   |   |   |   |   |   |   |--- no_of_week_nights >  3.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  80.38
|   |   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 10.00] class: 1
|   |   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 26.00] class: 1
|   |   |   |   |   |   |--- avg_price_per_room >  84.58
|   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 4 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [257.00, 8.00] class: 0
|   |   |   |   |   |   |   |   |   |--- room_type_reserved_Room_Type 4 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 3.00] class: 1
|   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |--- arrival_month <= 6.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [11.00, 0.00] class: 0
|   |   |   |   |   |   |   |   |   |--- arrival_month >  6.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [0.00, 13.00] class: 1
|   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 8.00] class: 1
|   |   |   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |   |   |--- weights: [459.00, 4.00] class: 0
|   |   |   |--- no_of_special_requests >  0.50
|   |   |   |   |--- no_of_weekend_nights <= 0.50
|   |   |   |   |   |--- lead_time <= 180.50
|   |   |   |   |   |   |--- weights: [8.00, 54.00] class: 1
|   |   |   |   |   |--- lead_time >  180.50
|   |   |   |   |   |   |--- market_segment_type_Offline <= 0.50
|   |   |   |   |   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 11.50
|   |   |   |   |   |   |   |   |   |--- weights: [114.00, 1.00] class: 0
|   |   |   |   |   |   |   |   |--- arrival_month >  11.50
|   |   |   |   |   |   |   |   |   |--- weights: [10.00, 7.00] class: 0
|   |   |   |   |   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |   |   |   |   |--- weights: [0.00, 8.00] class: 1
|   |   |   |   |   |   |--- market_segment_type_Offline >  0.50
|   |   |   |   |   |   |   |--- weights: [4.00, 13.00] class: 1
|   |   |   |   |--- no_of_weekend_nights >  0.50
|   |   |   |   |   |--- no_of_week_nights <= 9.00
|   |   |   |   |   |   |--- market_segment_type_Online <= 0.50
|   |   |   |   |   |   |   |--- weights: [4.00, 131.00] class: 1
|   |   |   |   |   |   |--- market_segment_type_Online >  0.50
|   |   |   |   |   |   |   |--- lead_time <= 366.00
|   |   |   |   |   |   |   |   |--- arrival_date <= 23.50
|   |   |   |   |   |   |   |   |   |--- weights: [41.00, 214.00] class: 1
|   |   |   |   |   |   |   |--- arrival_date >  23.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 7.50
|   |   |   |   |   |   |   |   |   |--- weights: [4.00, 33.00] class: 1
|   |   |   |   |   |   |   |   |--- arrival_month >  7.50
```

```
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [7.00, 23.00] class: 1
|   |   |   |   |   |   |   |   |   |--- no_of_weekend_nights >  1.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [22.00, 16.00] class: 0
|   |   |   |   |   |   |   |--- lead_time >  366.00
|   |   |   |   |   |   |   |   |--- weights: [3.00, 0.00] class: 0
|   |   |   |   |   |--- no_of_week_nights >  9.00
|   |   |   |   |   |   |--- weights: [7.00, 1.00] class: 0
|   |--- avg_price_per_room >  100.04
|   |   |--- arrival_month <= 11.50
|   |   |   |--- no_of_special_requests <= 2.50
|   |   |   |   |--- weights: [1818.00, 0.00] class: 0
|   |   |   |--- no_of_special_requests >  2.50
|   |   |   |   |--- weights: [0.00, 27.00] class: 1
|   |   |--- arrival_month >  11.50
|   |   |   |--- no_of_special_requests <= 0.50
|   |   |   |   |--- weights: [0.00, 42.00] class: 1
|   |   |   |--- no_of_special_requests >  0.50
|   |   |   |   |--- arrival_date <= 24.50
|   |   |   |   |   |--- weights: [0.00, 5.00] class: 1
|   |   |   |   |--- arrival_date >  24.50
|   |   |   |   |   |--- weights: [15.00, 4.00] class: 0
```

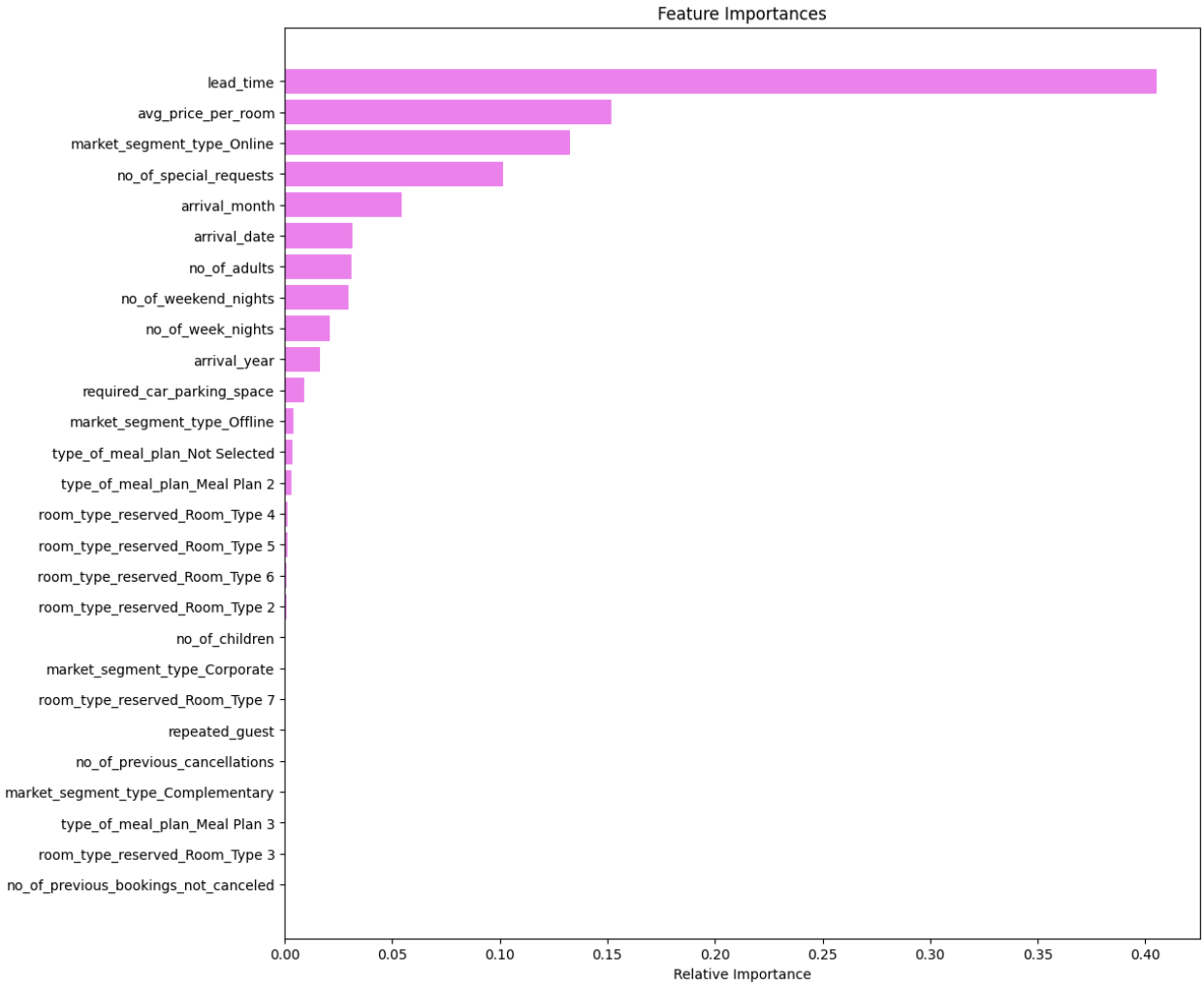
In [421…

```
# importance of features in the tree building ( The importance of a feature is
# (normalized) total reduction of the 'criterion' brought by that feature. It

print(
    pd.DataFrame(
        best_model.feature_importances_, columns=["Imp"], index=X_train.columr
    ).sort_values(by="Imp", ascending=False)
)
```

	Imp
lead_time	0.405119
avg_price_per_room	0.151754
market_segment_type_Online	0.132834
no_of_special_requests	0.101380
arrival_month	0.054396
arrival_date	0.031578
no_of_adults	0.031248
no_of_weekend_nights	0.029732
no_of_week_nights	0.021120
arrival_year	0.016613
required_car_parking_space	0.009186
market_segment_type_Offline	0.003902
type_of_meal_plan_Not Selected	0.003478
type_of_meal_plan_Meal Plan 2	0.003167
room_type_reserved_Room_Type 4	0.001515
room_type_reserved_Room_Type 5	0.001158
room_type_reserved_Room_Type 6	0.001047
room_type_reserved_Room_Type 2	0.000773
no_of_previous_bookings_not_canceled	0.000000
no_of_previous_cancellations	0.000000
no_of_children	0.000000
repeated_guest	0.000000
room_type_reserved_Room_Type 3	0.000000
room_type_reserved_Room_Type 7	0.000000
market_segment_type_Complementary	0.000000
market_segment_type_Corporate	0.000000
type_of_meal_plan_Meal Plan 3	0.000000

In [423…

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



Q18

-According to importance, it can be seen that lead time is the most important feature followed by the average price per room.

In [748…

```
# test performance comparison

models_train_comp_df = pd.concat(
    [
        decision_tree_perf_train.T,
        decision_tree_postpruned_perf_test.T,
        decision_tree_postpruned_perf_train.T,
    ],
    axis=1,
)
models_train_comp_df.columns = [
    "Decision Tree sklearn",
    "Decision Tree (Pre-pruning)",
    "Decision Tree (Post-Pruning)",
]
```

```
print("Test set performance comparison:")
models_train_comp_df
```

```
Test set performance comparison:
```

Out[748…

	Decision Tree sklearn	Decision Tree (Pre-pruning)	Decision Tree (Post-Pruning)
Accuracy	0.99352	0.88015	0.89635
Recall	0.99464	0.93597	0.94825
Precision	0.99567	0.89252	0.90177
F1	0.99515	0.91373	0.92443

Q19

- From the decision tree, *recall after postprunning* is best model to use for our predictons.

In [ ]:

Actionable Insights and Recommendations

- What profitable policies for cancellations and refunds can the hotel adopt?
- What other recommedations would you suggest to the hotel?

Recommendations

- Guest should pay for reservations of their rooms upon booking
- If a guest wants to cancel his or her booking, they should know that 20% of their reservation payments will be deducted then the balance will be refunded.
- they should consider more on their online platforms with charming ads to attract more customers.

references

- additonal_casestudy_loan_Delinquent
- incomeGroupClassifier_casestudy

In [ ]: