In [1]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
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 import metrics
from sklearn.metrics import f1_score, accuracy_score, recall_score, precis
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import warnings
from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter('ignore', ConvergenceWarning)
```

Here im importing all needed packages

In [2]:
```
df = pd.read_csv('/Users/conne/Downloads/INNHotelsGroup.csv')
```

Here im loading my data

The next few lines im getting an overview of the code

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df.tail()
```

Out[4]:

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights
36270	INN36271	3	0	2	6
36271	INN36272	2	0	1	3
36272	INN36273	2	0	2	6
36273	INN36274	2	0	0	3
36274	INN36275	2	0	1	2

In [5]:
```
df.info()
df.shape
```

```
<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
```

In [6]:
```
print(df.value_counts(subset='type_of_meal_plan'))
print(df.value_counts(subset='room_type_reserved'))
print(df.value_counts(subset='market_segment_type'))
print(df.value_counts(subset='booking_status'))
print(df.value_counts(subset='repeated_guest'))
```

```
type_of_meal_plan
Meal Plan 1     27835
Not Selected     5130
Meal Plan 2      3305
Meal Plan 3         5
dtype: int64
room_type_reserved
Room_Type 1    28130
Room_Type 4     6057
Room_Type 6      966
Room_Type 2      692
Room_Type 5      265
Room_Type 7      158
Room_Type 3        7
dtype: int64
market_segment_type
Online           23214
Offline          10528
Corporate         2017
Complementary      391
Aviation           125
dtype: int64
booking_status
Not_Canceled    24390
Canceled        11885
dtype: int64
repeated_guest
0    35345
1      930
dtype: int64
```

In [7]:
```
print(df["type_of_meal_plan"].value_counts(1))
print(df["room_type_reserved"].value_counts(1))
print(df["market_segment_type"].value_counts(1))
print(df["booking_status"].value_counts(1))
```

```
Meal Plan 1     0.767333
Not Selected    0.141420
Meal Plan 2     0.091110
Meal Plan 3     0.000138
Name: type_of_meal_plan, dtype: float64
Room_Type 1    0.775465
Room_Type 4    0.166975
Room_Type 6    0.026630
Room_Type 2    0.019076
Room_Type 5    0.007305
Room_Type 7    0.004356
Room_Type 3    0.000193
Name: room_type_reserved, dtype: float64
Online           0.639945
Offline          0.290227
Corporate        0.055603
Complementary    0.010779
Aviation         0.003446
Name: market_segment_type, dtype: float64
```

77.55% of people get room type 1 only 14.14% of people dont get a meal plan

In [8]:
```
df.describe()
```

Out[8]:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	required_ca
count	36275.000000	36275.000000	36275.000000	36275.000000	
mean	1.844962	0.105279	0.810724	2.204300	
std	0.518715	0.402648	0.870644	1.410905	
min	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	
50%	2.000000	0.000000	1.000000	2.000000	
75%	2.000000	0.000000	2.000000	3.000000	
max	4.000000	10.000000	7.000000	17.000000	

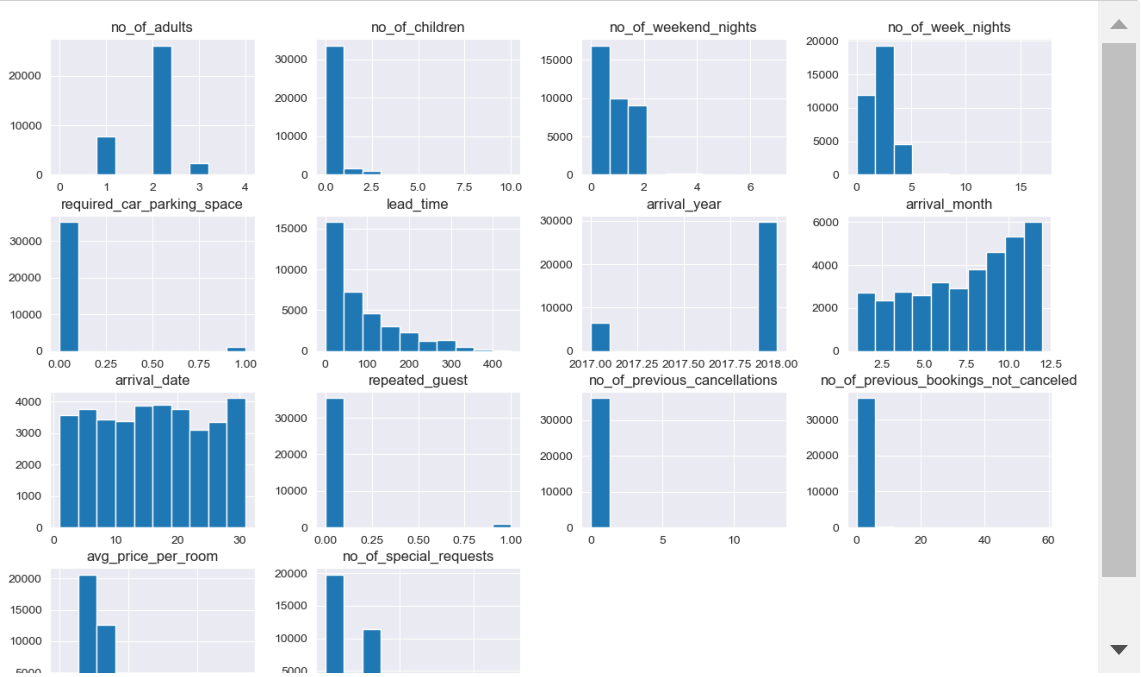
Most people dont reqiure a place to park so this probably has little impact on cancellations

In [9]:
```
df.isna().values.any()
```

Out[9]:
```
False
```

i have no missing values

In [10]:
```
sns.set_style("darkgrid")
df.hist(figsize=(15, 10))
plt.show()
```



Most people are getting rooms for 2 adults Most people came in 2018 Bussiness increases in the last 4 months of the year Almost everyone who booked a room didn't need a parking spot

In [11]:
```
cols = ["required_car_parking_space", "repeated_guest"]
df[cols] = df[cols].replace(0, "no")
df[cols] = df[cols].replace(1, "yes")
```

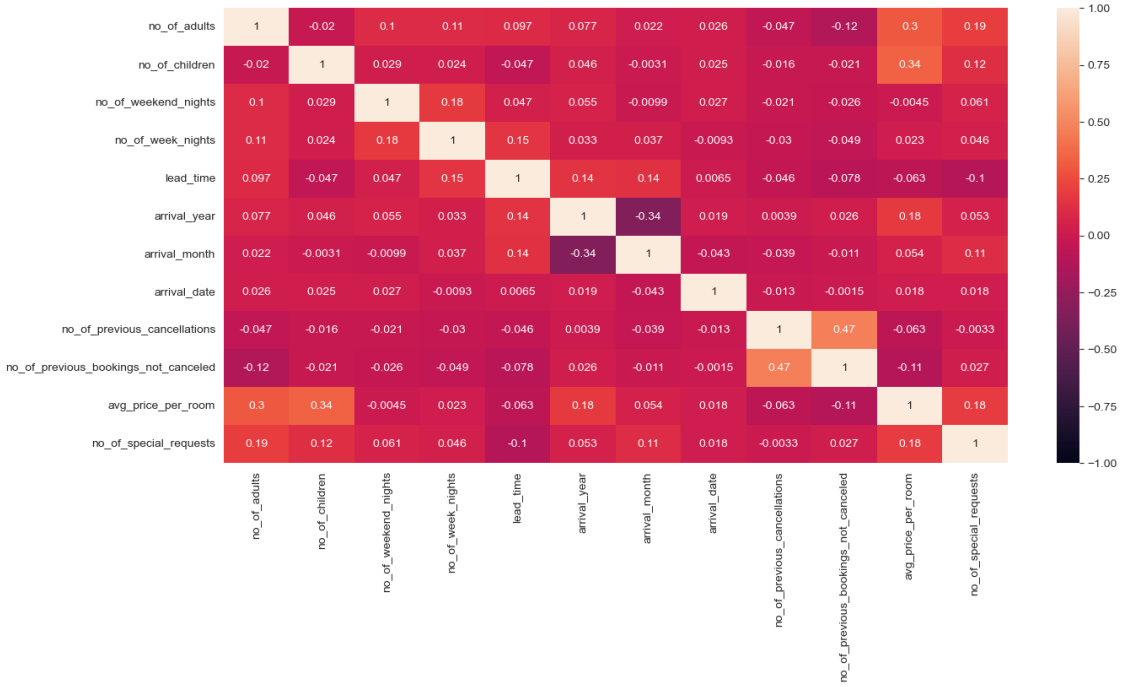
In [12]:
```
cols = ["required_car_parking_space", "repeated_guest"]
df[cols] = df[cols].replace(0, "no")
df[cols] = df[cols].replace(1, "yes")
df['required_car_parking_space'].value_counts()
```

Out[12]:
```
no     35151
yes     1124
Name: required_car_parking_space, dtype: int64
```

In [13]:
```
df['repeated_guest'].value_counts()
```

Out[13]:
```
no     35345
yes      930
Name: repeated_guest, dtype: int64
```

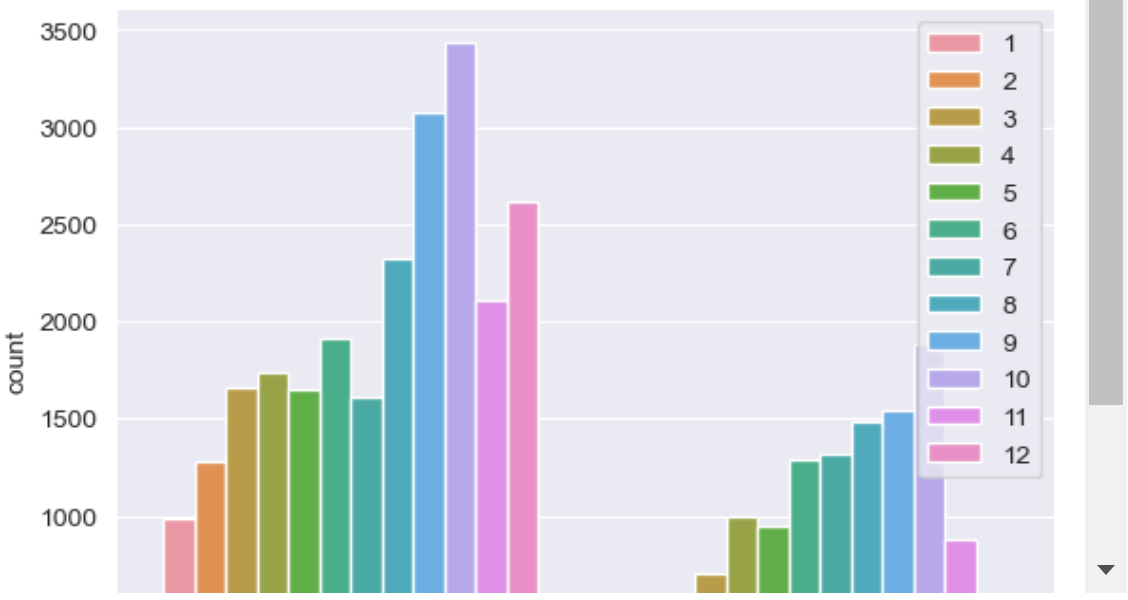
In [14]:
```
plt.figure(figsize=(15, 7))
sns.heatmap(df.corr(), annot=True, vmin=-1, vmax=1)
plt.show()
```



There are not highly correlated values

In [15]:
```
sns.countplot(data=df, x='booking_status', hue='arrival_month')
plt.legend(loc='upper right')
```

Out[15]:
```
<matplotlib.legend.Legend at 0x24138a25ad0>
```

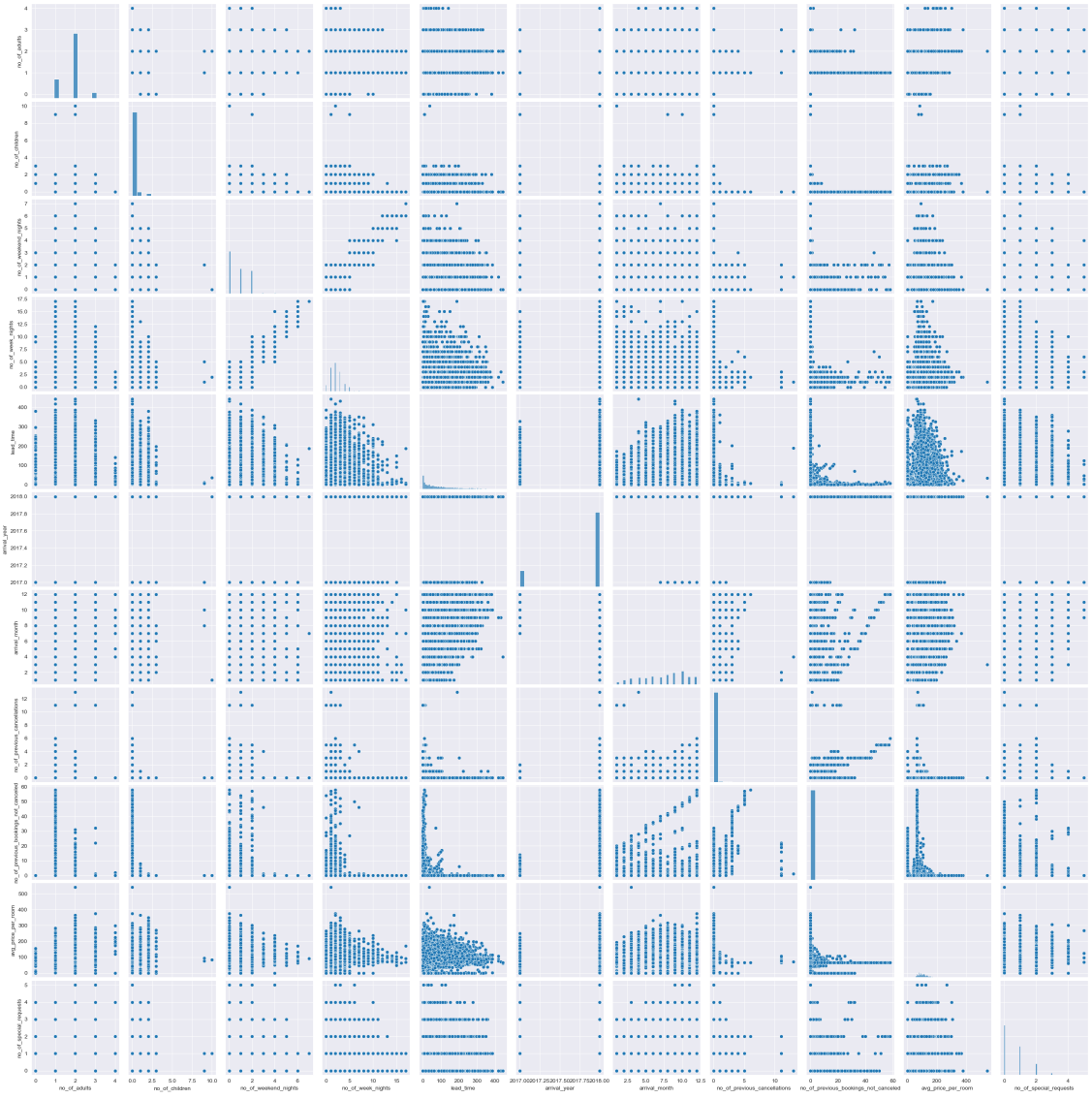


Question 1: the ratio of not canceled to canceled increase as you get later in to the year

In [16]:
```
df.columns
```

Out[16]:
```
Index(['Booking_ID', 'no_of_adults', 'no_of_children', 'no_of_weekend_nig
hts',
       'no_of_week_nights', 'type_of_meal_plan', 'required_car_parking_sp
ace',
       'room_type_reserved', 'lead_time', 'arrival_year', 'arrival_mont
h',
       'arrival_date', 'market_segment_type', 'repeated_guest',
       'no_of_previous_cancellations', 'no_of_previous_bookings_not_cance
led',
       'avg_price_per_room', 'no_of_special_requests', 'booking_status'],
      dtype='object')
```

In [17]:
```
sns.pairplot(df[['no_of_adults', 'no_of_children', 'no_of_weekend_nights',
       'arrival_month', 'no_of_previous_cancellations', 'no_of_previous_bo
plt.show()
```



The pair plot shows us are data columns aren't very correlated with eachother

In [18]:
```
df["arrival_month"].value_counts(1)
```

Out[18]:
```
10    0.146575
9     0.127112
8     0.105114
6     0.088298
12    0.083280
11    0.082150
7     0.080496
4     0.075424
5     0.071620
3     0.065003
2     0.046975
1     0.027953
Name: arrival_month, dtype: float64
```

October is the busiest month of the year as 14% of people visit in that month.

In [19]:
```
df["market_segment_type"].value_counts()
```

Out[19]:
```
Online           23214
Offline          10528
Corporate         2017
Complementary      391
Aviation           125
Name: market_segment_type, dtype: int64
```

In [20]:
```
df["market_segment_type"].value_counts(1)
```

Out[20]:
```
Online           0.639945
Offline          0.290227
Corporate        0.055603
Complementary    0.010779
Aviation         0.003446
Name: market_segment_type, dtype: float64
```

Question 2: most guests come from online

In [21]:
```
df_online = df.copy()
df_online['market_segment_type'] = df_online['market_segment_type'].replac
df_online = df_online.dropna()
df_offline = df.copy()
df_offline['market_segment_type'] = df_offline['market_segment_type'].repl
df_offline = df_offline.dropna()
df_corporate = df.copy()
df_corporate['market_segment_type'] = df_corporate['market_segment_type'].
df_corporate = df_corporate.dropna()
df_complementary = df.copy()
df_complementary['market_segment_type'] = df_complementary['market_segment
df_complementary = df_complementary.dropna()
df_aviation = df.copy()
df_aviation['market_segment_type'] = df_aviation['market_segment_type'].re
df_aviation = df_aviation.dropna()
```

In [22]:
```
df_online.isna().sum()
```

Out[22]:
```
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 [23]:
```
df_online.shape
```

Out[23]:
```
(23214, 19)
```

In [24]:
```
df_online.mean()
```

Out[24]:
```
no_of_adults                            1.939476
no_of_children                          0.151977
no_of_weekend_nights                    0.886577
no_of_week_nights                       2.289911
lead_time                              75.334238
arrival_year                         2017.872878
arrival_month                           7.380417
arrival_date                           15.690618
no_of_previous_cancellations            0.013225
no_of_previous_bookings_not_canceled    0.012105
avg_price_per_room                    112.256855
no_of_special_requests                  0.842250
dtype: float64
```

In [25]:
```
df_offline.shape
```

Out[25]:
```
(10528, 19)
```

In [26]:
```
df_offline.mean()
```

Out[26]:
```
no_of_adults                            1.777641
no_of_children                          0.021087
no_of_weekend_nights                    0.730528
no_of_week_nights                       2.180661
lead_time                             122.872625
arrival_year                         2017.722074
arrival_month                           7.572758
arrival_date                           15.396087
no_of_previous_cancellations            0.011113
no_of_previous_bookings_not_canceled    0.010828
avg_price_per_room                     91.632679
no_of_special_requests                  0.202603
dtype: float64
```

In [27]:
```
df_corporate.shape
```

Out[27]:
```
(2017, 19)
```

In [28]:
```
df_corporate.mean()
```

Out[28]:
```
no_of_adults                            1.230045
no_of_children                          0.009916
no_of_weekend_nights                    0.427863
no_of_week_nights                       1.488845
lead_time                              21.818047
arrival_year                         2017.753099
arrival_month                           7.103619
arrival_date                           15.695092
no_of_previous_cancellations            0.166584
no_of_previous_bookings_not_canceled    2.070402
avg_price_per_room                     82.911740
no_of_special_requests                  0.222112
dtype: float64
```

In [29]:
```
df_complementary.shape
```

Out[29]:
```
(391, 19)
```

In [30]:
```
df_complementary.mean()
```

Out[30]:
```
no_of_adults                            1.483376
no_of_children                          0.125320
no_of_weekend_nights                    0.329923
no_of_week_nights                       1.240409
lead_time                              12.035806
arrival_year                         2017.644501
arrival_month                           7.723785
arrival_date                           15.017903
no_of_previous_cancellations            0.209719
no_of_previous_bookings_not_canceled    2.475703
avg_price_per_room                      3.141765
no_of_special_requests                  0.882353
dtype: float64
```

In [31]:
```
df_aviation.shape
```

Out[31]:
```
(125, 19)
```

In [32]:
```
df_aviation.mean()
```

Out[32]:
```
no_of_adults                            1.016
no_of_children                          0.000
no_of_weekend_nights                    1.160
no_of_week_nights                       2.856
lead_time                               5.488
arrival_year                         2018.000
arrival_month                           7.120
arrival_date                           15.360
no_of_previous_cancellations            0.040
no_of_previous_bookings_not_canceled    0.208
avg_price_per_room                    100.704
no_of_special_requests                  0.000
dtype: float64
```

Question 3: The average price for an online room is 112.25 The average price for an offline room is 91.63 The average price for an corporate room is 82.91 The average price for an complementary room is 3.14 The average price for an aviation room is 100.7

In [33]:
```
df["booking_status"].value_counts()
```

Out[33]:
```
Not_Canceled    24390
Canceled        11885
Name: booking_status, dtype: int64
```

In [34]:
```
df["booking_status"].value_counts(1)
```

Out[34]:
```
Not_Canceled    0.672364
Canceled        0.327636
Name: booking_status, dtype: float64
```

Question 4: 32.76% of bookings are conceled

In [35]:
```
df_repeat = df.copy()
df_repeat['repeated_guest'] = df_repeat['repeated_guest'].replace(['no'],
df_repeat = df_repeat.dropna()
df_repeat.shape
```

Out[35]:
```
(930, 19)
```

In [116]:
```
df_repeat["booking_status"].value_counts(1)
```

Out[116]:
```
Not_Canceled    0.982796
Canceled        0.017204
Name: booking_status, dtype: float64
```

Question 5: 1.72% of repeat customers canceled.

In [37]:
```
df["no_of_special_requests"].value_counts()
```

Out[37]:
```
0    19777
1    11373
2     4364
3      675
4       78
5        8
Name: no_of_special_requests, dtype: int64
```

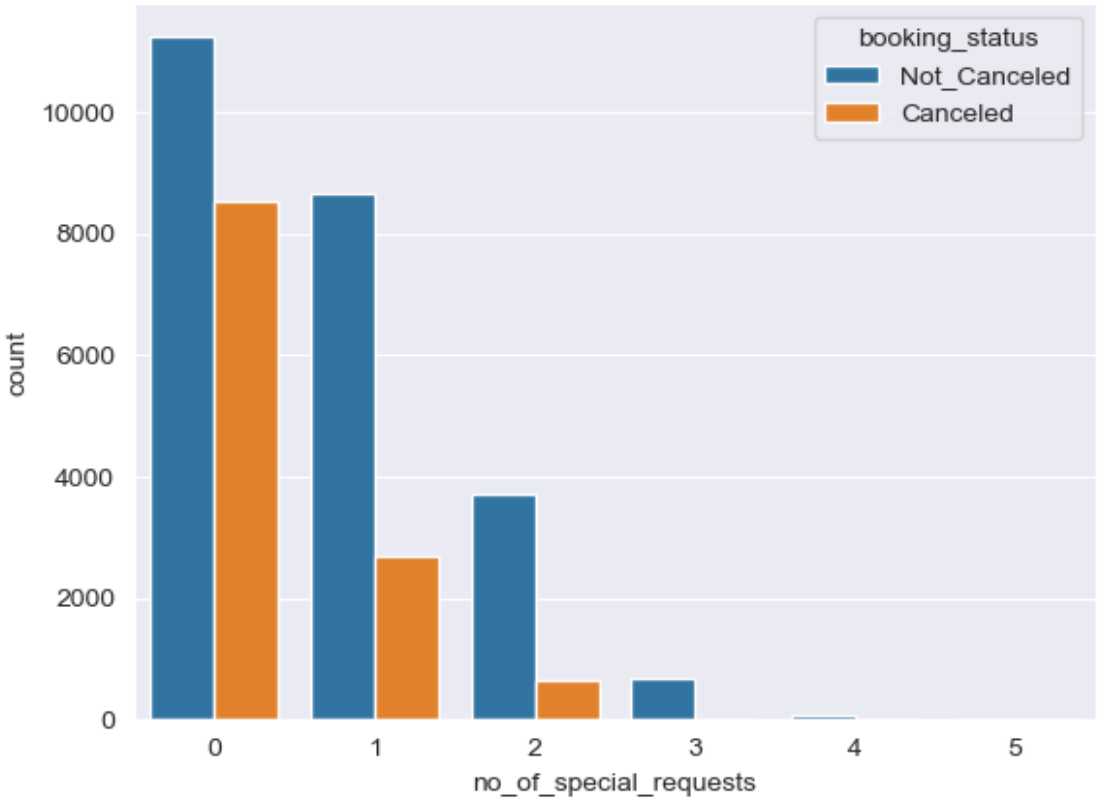
In [38]:
```
print("There have been", 36275-19777,"people to have a special request.")
```

```
There have been 16498 people to have a special request.
```

Question 6: Yes, when people make special requests the cancelation rate goes down. There have been 16498 people to have at least one special request.

In [39]:
```
sns.countplot(data=df, x='no_of_special_requests', hue='booking_status')
```

Out[39]:
```
<Axes: xlabel='no_of_special_requests', ylabel='count'>
```



In [40]:
```
df = df.drop(columns=['Booking_ID'])
```

In [41]:
```
df['booking_status'] = df['booking_status'].replace(['Not_Canceled'], 0)
df['booking_status'] = df['booking_status'].replace(['Canceled'], 1)
df["booking_status"].value_counts(1)
```

Out[41]:
```
0    0.672364
1    0.327636
Name: booking_status, dtype: float64
```

In [42]:
```
X = df.drop(["booking_status"], axis=1)
Y = df["booking_status"]
X = add_constant(X)
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.30, random_state=1, stratify=Y)
```

Here im creating my vars and splitting my data

In [43]:
```
logit = sm.Logit(y_train, X_train.astype(float))
#lg = logit.fit(max_iter=500)
```

In [44]:
```
lg = logit.fit(max_iter=500)
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.422351
         Iterations: 35
```

In [45]:
```
print(lg.summary())
```

```
                           Logit Regression Results
==============================================================================
=====
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25364
Method:                           MLE   Df Model:
27
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3322
Time:                        01:57:00   Log-Likelihood:                -1
0724.
converged:                      False   LL-Null:                       -1
6060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
==============================
                                            coef    std err          z
P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------
const                                 -886.9563    121.332     -7.310
0.000   -1124.762    -649.150
no_of_adults                            0.0333      0.038      0.883
0.377      -0.041       0.107
no_of_children                          0.0833      0.061      1.371
0.170      -0.036       0.202
no_of_weekend_nights                    0.1461      0.020      7.368
0.000       0.107       0.185
no_of_week_nights                       0.0354      0.012      2.881
0.004       0.011       0.059
lead_time                               0.0158      0.000     58.946
0.000       0.015       0.016
arrival_year                            0.4382      0.060      7.288
0.000       0.320       0.556
arrival_month                          -0.0475      0.006     -7.315
0.000      -0.060      -0.035
arrival_date                            0.0030      0.002      1.528
0.127      -0.001       0.007
no_of_previous_cancellations            0.3476      0.102      3.413
0.001       0.148       0.547
no_of_previous_bookings_not_canceled   -1.3821      0.906     -1.526
0.127      -3.157       0.393
avg_price_per_room                      0.0184      0.001     24.903
0.000       0.017       0.020
no_of_special_requests                 -1.4904      0.030    -48.963
0.000      -1.550      -1.431
type_of_meal_plan_Meal Plan 2           0.1746      0.067      2.607
0.009       0.043       0.306
type_of_meal_plan_Meal Plan 3          10.1526    107.971      0.094
0.925    -201.466     221.771
type_of_meal_plan_Not Selected          0.1995      0.053      3.741
0.000       0.095       0.304
required_car_parking_space_yes         -1.6146      0.137    -11.769
0.000      -1.883      -1.346
```

```
room_type_reserved_Room_Type 2          -0.4172      0.133     -3.129
0.002      -0.679      -0.156
room_type_reserved_Room_Type 3           1.1881      1.891      0.628
0.530      -2.518       4.895
room_type_reserved_Room_Type 4          -0.2679      0.053     -5.019
0.000      -0.373      -0.163
room_type_reserved_Room_Type 5          -0.6822      0.215     -3.173
0.002      -1.104      -0.261
room_type_reserved_Room_Type 6          -0.8460      0.153     -5.534
0.000      -1.146      -0.546
room_type_reserved_Room_Type 7          -1.3601      0.298     -4.566
0.000      -1.944      -0.776
market_segment_type_Complementary      -24.5570   2.24e+04     -0.001
0.999   -4.39e+04    4.39e+04
market_segment_type_Corporate           -0.8514      0.276     -3.087
0.002      -1.392      -0.311
market_segment_type_Offline             -1.7645      0.264     -6.688
0.000      -2.282      -1.247
market_segment_type_Online               0.0070      0.261      0.027
0.979      -0.504       0.518
repeated_guest_yes                      -1.9186      0.767     -2.502
0.012      -3.421      -0.416
==============================================================================
==============================
```

In [46]:
```
pred_train = lg.predict(X_train) > 0.5
pred_train = np.round(pred_train)
```

In [47]:
```
cm = confusion_matrix(y_train, pred_train)
plt.figure(figsize=(7, 5))
sns.heatmap(cm, annot=True, fmt="g")
plt.xlabel("Predicted Values")
plt.ylabel("Actual Values")
plt.show()
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [48]:
```
print("Accuracy on training set : ", accuracy_score(y_train, pred_train))
```

```
Accuracy on training set :  0.8068683049779458
```

The model is preforming fairly well

In [49]:
```
vif_series = pd.Series ([variance_inflation_factor(X_train.values, i) for
print("VIF values: \n\n{}\n".format(vif_series))
```

```
VIF values:

const                                  3.959545e+07
no_of_adults                           1.345059e+00
no_of_children                         2.007213e+00
no_of_weekend_nights                   1.067255e+00
no_of_week_nights                      1.094373e+00
lead_time                              1.401920e+00
arrival_year                           1.433261e+00
arrival_month                          1.277398e+00
arrival_date                           1.007629e+00
no_of_previous_cancellations           1.322011e+00
no_of_previous_bookings_not_canceled   1.570863e+00
avg_price_per_room                     2.032631e+00
no_of_special_requests                 1.247233e+00
type_of_meal_plan_Meal Plan 2          1.261818e+00
type_of_meal_plan_Meal Plan 3          1.007964e+00
type_of_meal_plan_Not Selected         1.279209e+00
required_car_parking_space_yes         1.034943e+00
room_type_reserved_Room_Type 2         1.094581e+00
```

I will now drop all p-values greater than .05

In [50]:
```
X_train1 = X_train.drop("market_segment_type_Complementary", axis=1)
logit1 = sm.Logit(y_train, X_train1.astype(float))
lg1 = logit1.fit()
pred_train1 = lg1.predict(X_train1)
pred_train1 = np.round(pred_train1)
print("Accuracy on training set : ", accuracy_score(y_train, pred_train1))
```

```
Optimization terminated successfully.
         Current function value: 0.422630
         Iterations 16
Accuracy on training set :  0.8063563327032136
```

In [51]:
```
print(lg1.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25365
Method:                           MLE   Df Model:
26
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3318
Time:                        01:57:01   Log-Likelihood:
-10731.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
==================================
```

In [52]:
```
X_train2 = X_train1.drop("room_type_reserved_Room_Type 3", axis=1)
logit2 = sm.Logit(y_train, X_train2.astype(float))
lg2 = logit2.fit()

pred_train2 = lg2.predict(X_train2)
pred_train2 = np.round(pred_train2)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train2))
```

```
Optimization terminated successfully.
         Current function value: 0.422635
         Iterations 16
Accuracy on training set :  0.8063957151858853
```

In [53]:
```
print(lg2.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25366
Method:                           MLE   Df Model:
25
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3318
Time:                        01:57:01   Log-Likelihood:
-10732.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
==================================
```

In [54]:
```
X_train3 = X_train2.drop("no_of_adults", axis=1)
logit3 = sm.Logit(y_train, X_train3.astype(float))
lg3 = logit3.fit()

pred_train3 = lg3.predict(X_train3)
pred_train3 = np.round(pred_train3)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train3))
```

```
Optimization terminated successfully.
         Current function value: 0.422646
         Iterations 16
Accuracy on training set :  0.8065926275992439
```

In [55]:
```
print(lg3.summary())
```

Logit Regression Results

Dep. Variable:	booking_status	No. Observations:	25392
Model:	Logit	Df Residuals:	25367
Method:	MLE	Df Model:	24
Date:	Sat, 30 Sep 2023	Pseudo R-squ.:	0.3318
Time:	01:57:02	Log-Likelihood:	-10732.
converged:	True	LL-Null:	-16060.
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-885.9115	121.142	-7.313	0.000	-1123.346	-648.477
no_of_children	0.0737	0.060	1.220	0.222	-0.045	0.192
no_of_weekend_nights	0.1489	0.020	7.520	0.000	0.110	0.188
no_of_week_nights	0.0369	0.012	3.007	0.003	0.013	0.061
lead_time	0.0158	0.000	59.388	0.000	0.015	0.016
arrival_year	0.4375	0.060	7.288	0.000	0.320	0.555
arrival_month	-0.0478	0.006	-7.369	0.000	-0.061	-0.035
arrival_date	0.0030	0.002	1.534	0.125	-0.001	0.007
no_of_previous_cancellations	0.3479	0.102	3.415	0.001	0.148	0.548
no_of_previous_bookings_not_canceled	-1.3704	0.899	-1.525	0.127	-3.132	0.391
avg_price_per_room	0.0187	0.001	25.747	0.000	0.017	0.020
no_of_special_requests	-1.4898	0.030	-49.240	0.000	-1.549	-1.431
type_of_meal_plan_Meal Plan 2	0.1723	0.067	2.573	0.010	0.041	0.304
type_of_meal_plan_Meal Plan 3	3.4727	2.912	1.193	0.233	-2.235	9.180
type_of_meal_plan_Not Selected	0.2056	0.053	3.865	0.000	0.101	0.310
required_car_parking_space_yes	-1.6143	0.137	-11.767	0.000	-1.883	-1.345
room_type_reserved_Room Type 2	-0.4193	0.133	-3.146	0.002	-0.681	-0.158

```
room_type_reserved_Room_Type 4          -0.2586      0.052     -4.969
0.000      -0.361      -0.157
room_type_reserved_Room_Type 5          -0.6914      0.215     -3.219
0.001      -1.112      -0.270
room_type_reserved_Room_Type 6          -0.8466      0.153     -5.546
0.000      -1.146      -0.547
room_type_reserved_Room_Type 7          -1.3589      0.297     -4.576
0.000      -1.941      -0.777
market_segment_type_Corporate           -0.4549      0.259     -1.758
0.079      -0.962       0.052
market_segment_type_Offline             -1.3574      0.245     -5.541
0.000      -1.837      -0.877
market_segment_type_Online               0.4148      0.242      1.715
0.086      -0.059       0.889
repeated_guest_yes                      -1.9179      0.768     -2.498
0.012      -3.423      -0.413
==========================================================================
==============================
```

In [56]:

```
X_train4 = X_train3.drop("no_of_children", axis=1)
logit4 = sm.Logit(y_train, X_train4.astype(float))
lg4 = logit4.fit()

pred_train4 = lg4.predict(X_train4)
pred_train4 = np.round(pred_train4)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train4))
```

```
Optimization terminated successfully.
         Current function value: 0.422675
         Iterations 16
Accuracy on training set :  0.806710775047259
```

In [57]:

```
print(lg4.summary())
```

```
                           Logit Regression Results
==========================================================================
========
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25368
Method:                           MLE   Df Model:
23
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3317
Time:                        01:57:02   Log-Likelihood:
-10733.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==========================================================================
================================
```

In [58]:
```
X_train5 = X_train4.drop("type_of_meal_plan_Meal Plan 3", axis=1)
logit5 = sm.Logit(y_train, X_train5.astype(float))
lg5 = logit5.fit()

pred_train5 = lg5.predict(X_train5)
pred_train5 = np.round(pred_train5)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train5))
```

```
Optimization terminated successfully.
         Current function value: 0.422715
         Iterations 16
Accuracy on training set :  0.8066713925645873
```

In [59]:
```
print(lg5.summary())
```

```
                           Logit Regression Results
==============================================================================
=====
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25369
Method:                           MLE   Df Model:
22
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3317
Time:                        01:57:02   Log-Likelihood:                -1
0734.
converged:                       True   LL-Null:                       -1
6060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
==============================
                                            coef    std err          z
P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------
const                                 -885.5409    121.030     -7.317
0.000   -1122.755    -648.326
no_of_weekend_nights                     0.1491      0.020      7.529
0.000       0.110       0.188
no_of_week_nights                        0.0370      0.012      3.014
0.003       0.013       0.061
lead_time                                0.0158      0.000     59.376
0.000       0.015       0.016
arrival_year                             0.4373      0.060      7.291
0.000       0.320       0.555
arrival_month                           -0.0479      0.006     -7.381
0.000      -0.061      -0.035
arrival_date                             0.0030      0.002      1.561
0.119      -0.001       0.007
no_of_previous_cancellations             0.3481      0.102      3.416
0.001       0.148       0.548
no_of_previous_bookings_not_canceled    -1.3695      0.899     -1.524
0.128      -3.131       0.392
avg_price_per_room                       0.0189      0.001     26.112
0.000       0.017       0.020
no_of_special_requests                  -1.4882      0.030    -49.243
0.000      -1.547      -1.429
type_of_meal_plan_Meal Plan 2            0.1695      0.067      2.531
0.011       0.038       0.301
type_of_meal_plan_Not Selected           0.2005      0.053      3.785
0.000       0.097       0.304
required_car_parking_space_yes          -1.6142      0.137    -11.769
0.000      -1.883      -1.345
room_type_reserved_Room_Type 2          -0.3794      0.129     -2.937
0.003      -0.633      -0.126
room_type_reserved_Room_Type 4          -0.2666      0.052     -5.156
0.000      -0.368      -0.165
room_type_reserved_Room_Type 5          -0.6848      0.215     -3.190
0.001      -1.106      -0.264
```

```
room_type_reserved_Room_Type 6          -0.7349      0.120     -6.119
0.000      -0.970      -0.500
room_type_reserved_Room_Type 7          -1.3066      0.292     -4.476
0.000      -1.879      -0.734
market_segment_type_Corporate           -0.4650      0.259     -1.797
0.072      -0.972       0.042
market_segment_type_Offline             -1.3650      0.245     -5.571
0.000      -1.845      -0.885
market_segment_type_Online               0.4103      0.242      1.696
0.090      -0.064       0.885
repeated_guest_yes                      -1.9187      0.768     -2.499
0.012      -3.424      -0.414
==============================================================================
==============================
```

In [60]:
```
X_train6 = X_train5.drop("no_of_previous_bookings_not_canceled", axis=1)
logit6 = sm.Logit(y_train, X_train6.astype(float))
lg6 = logit6.fit()

pred_train6 = lg6.predict(X_train6)
pred_train6 = np.round(pred_train6)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train6))
```

```
Optimization terminated successfully.
         Current function value: 0.422882
         Iterations 11
Accuracy on training set :  0.806710775047259
```

In [61]:
```
print(lg6.summary())
```

```
                           Logit Regression Results
==============================================================================
========
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25370
Method:                           MLE   Df Model:
21
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3314
Time:                        01:57:02   Log-Likelihood:
-10738.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
============================
```

In [62]:
```
X_train7 = X_train6.drop("arrival_date", axis=1)
logit7 = sm.Logit(y_train, X_train7.astype(float))
lg7 = logit7.fit()

pred_train7 = lg7.predict(X_train7)
pred_train7 = np.round(pred_train7)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train7))
```

```
Optimization terminated successfully.
         Current function value: 0.422928
         Iterations 11
Accuracy on training set :  0.8061988027725268
```

In [63]:
```
print(lg7.summary())
```

```
                           Logit Regression Results
==============================================================================
========
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25371
Method:                           MLE   Df Model:
20
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3313
Time:                        01:57:02   Log-Likelihood:
-10739.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
============================
```

In [64]:
```
X_train8 = X_train7.drop("market_segment_type_Online", axis=1)
logit8 = sm.Logit(y_train, X_train8.astype(float))
lg8 = logit8.fit()

pred_train8 = lg8.predict(X_train8)
pred_train8 = np.round(pred_train8)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train8))
```

```
Optimization terminated successfully.
         Current function value: 0.422990
         Iterations 11
Accuracy on training set :  0.8065926275992439
```

In [65]:
```
print(lg8.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25372
Method:                           MLE   Df Model:
19
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3312
Time:                        01:57:02   Log-Likelihood:
-10741.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
============================
```

In [66]:
```
X_train9 = X_train8.drop("type_of_meal_plan_Meal Plan 2", axis=1)
logit9 = sm.Logit(y_train, X_train9.astype(float))
lg9 = logit9.fit()

pred_train9 = lg9.predict(X_train9)
pred_train9 = np.round(pred_train9)

print("Accuracy on training set : ", accuracy_score(y_train, pred_train9))
```

```
Optimization terminated successfully.
         Current function value: 0.423112
         Iterations 11
Accuracy on training set :  0.806710775047259
```

In [67]:
```
print(lg9.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:         booking_status   No. Observations:
25392
Model:                          Logit   Df Residuals:
25373
Method:                           MLE   Df Model:
18
Date:                Sat, 30 Sep 2023   Pseudo R-squ.:
0.3310
Time:                        01:57:03   Log-Likelihood:
-10744.
converged:                       True   LL-Null:
-16060.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
```

In [68]:
```
odds = np.exp(lg9.params)

# adding the odds to a dataframe
pd.DataFrame(odds, X_train9.columns, columns=["odds"]).T
```

Out[68]:

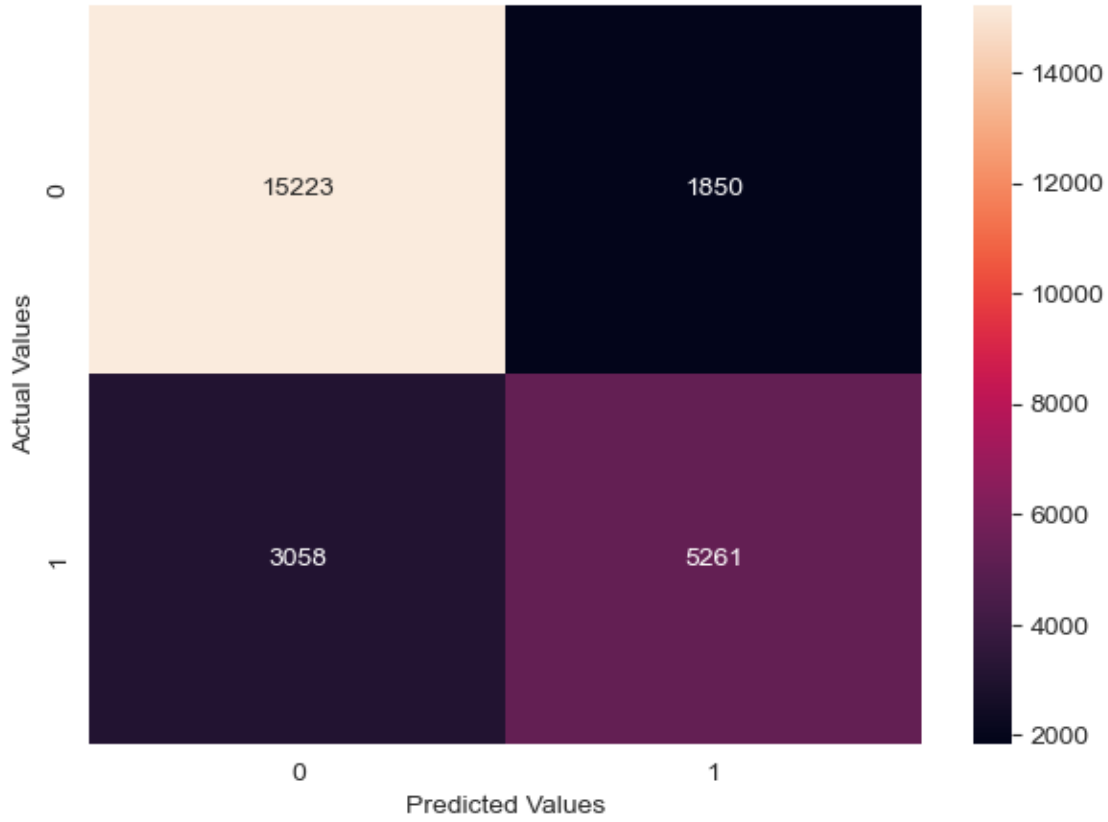
	const	no_of_weekend_nights	no_of_week_nights	lead_time	arrival_year	arrival_mor
odds	0.0	1.162288	1.035666	1.016125	1.482135	0.9501

In [69]:
```
# finding the percentage change
perc_change_odds = (np.exp(lg9.params) - 1) * 100

# adding the change_odds% to a dataframe
pd.DataFrame(perc_change_odds, X_train9.columns, columns=["change_odds%"])
```

Out[69]:

	change_odds%
const	-100.000000
no_of_weekend_nights	16.228751
no_of_week_nights	3.566640
lead_time	1.612464
arrival_year	48.213533
arrival_month	-4.981121
no_of_previous_cancellations	33.320739
avg_price_per_room	1.969566
no_of_special_requests	-77.288387
type_of_meal_plan_Not Selected	23.532481
required_car_parking_space_yes	-80.208023
room_type_reserved_Room_Type 2	-31.422024

In [70]:
```
cm = confusion_matrix(y_train, pred_train9)
plt.figure(figsize=(7, 5))
sns.heatmap(cm, annot=True, fmt="g")
plt.xlabel("Predicted Values")
plt.ylabel("Actual Values")
plt.show()
```

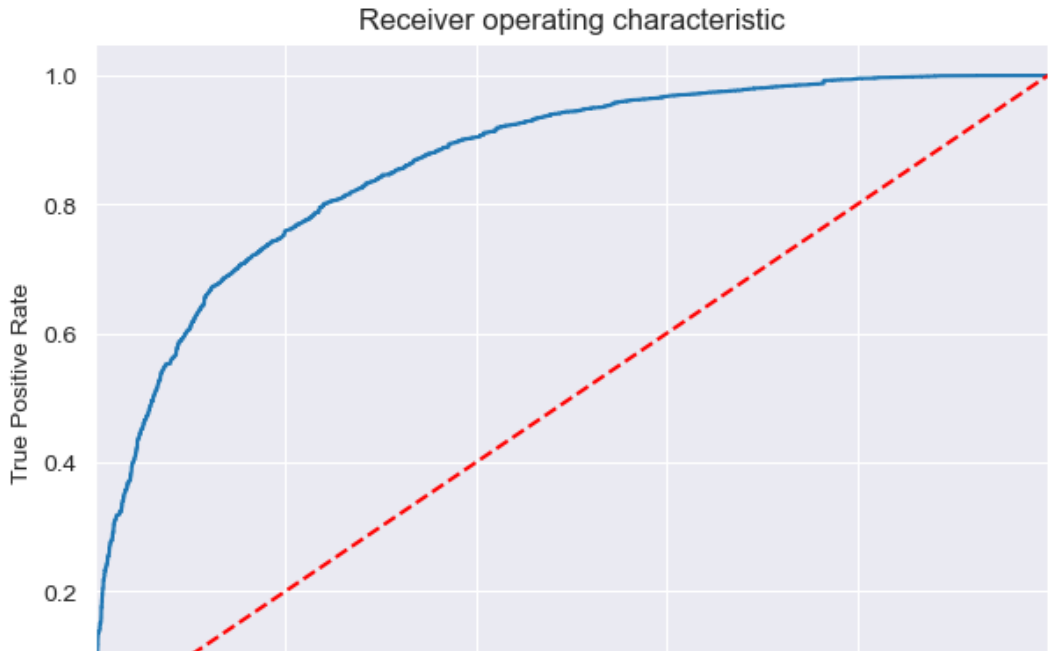


It seems the Predicted value seems twice as likely to incorrectly guess people wont be there when they actually are

In [71]:
```
print("Accuracy on training set : ", accuracy_score(y_train, pred_train9))
```

```
Accuracy on training set :  0.806710775047259
```

In [72]:
```
logit_roc_auc_train = roc_auc_score(y_train, lg9.predict(X_train9))
fpr, tpr, thresholds = roc_curve(y_train, lg9.predict(X_train9))
plt.figure(figsize=(7, 5))
plt.plot(fpr, tpr, label="Logistic Regression (area = %0.2f)" % logit_roc_
plt.plot([0, 1], [0, 1], "r--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



In [73]:
```
X_test = X_test.drop(["market_segment_type_Complementary", "room_type_rese
                      "no_of_adults", "no_of_children","type_of_meal_plan_
                      "no_of_previous_bookings_not_canceled","arrival_date
                      'type_of_meal_plan_Meal Plan 2'], axis=1)
```

In [74]:
```
pred_test = lg9.predict(X_test) > 0.5
pred_test = np.round(pred_test)
print("Accuracy on training set : ", accuracy_score(y_train, pred_train4))
print("Accuracy on test set : ", accuracy_score(y_test, pred_test))
```

```
Accuracy on training set :  0.806710775047259
Accuracy on test set :  0.8020766332812643
```

The final model preformed well with an 80% accuracy

In [75]:
```
odds = np.exp(lg9.params)
perc_change_odds = (np.exp(lg9.params) - 1) * 100
pd.set_option("display.max_columns", None)
pd.DataFrame({"Odds": odds, "Change_odd%": perc_change_odds}, index=X_trai
```

Out[75]:

	const	no_of_children	no_of_weekend_nights	no_of_week_nights	lead_time
Odds	0.0	NaN	1.162288	1.035666	1.016125
Change_odd%	-100.0	NaN	16.228751	3.566640	1.612464

In [76]:
```
df['repeated_guest'].value_counts()
```

Out[76]:
```
no     35345
yes      930
Name: repeated_guest, dtype: int64
```

In [77]:
```
col = ["required_car_parking_space"]
df[col] = df[col].replace("no", 0)
df[col] = df[col].replace("yes", 1)
col = ["type_of_meal_plan"]
df[col] = df[col].replace("Not Selected", 0)
df[col] = df[col].replace("Meal Plan 1", 1)
df[col] = df[col].replace("Meal Plan 2", 2)
df[col] = df[col].replace("Meal Plan 3", 3)
col = ["room_type_reserved"]
df[col] = df[col].replace("Room_Type 1", 1)
df[col] = df[col].replace("Room_Type 2", 2)
df[col] = df[col].replace("Room_Type 3", 3)
df[col] = df[col].replace("Room_Type 4", 4)
df[col] = df[col].replace("Room_Type 5", 5)
df[col] = df[col].replace("Room_Type 6", 6)
df[col] = df[col].replace("Room_Type 7", 7)
col = ["market_segment_type"]
df[col] = df[col].replace("Online", 1)
df[col] = df[col].replace("Offline", 2)
df[col] = df[col].replace("Corporate", 3)
df[col] = df[col].replace("Complementary", 4)
df[col] = df[col].replace("Aviation", 5)
col = ["repeated_guest"]
df[col] = df[col].replace("no", 0)
df[col] = df[col].replace("yes", 2)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 18 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   no_of_adults                          36275 non-null  int64
 1   no_of_children                        36275 non-null  int64
 2   no_of_weekend_nights                  36275 non-null  int64
 3   no_of_week_nights                     36275 non-null  int64
 4   type_of_meal_plan                     36275 non-null  int64
 5   required_car_parking_space            36275 non-null  int64
 6   room_type_reserved                    36275 non-null  int64
 7   lead_time                             36275 non-null  int64
 8   arrival_year                          36275 non-null  int64
 9   arrival_month                         36275 non-null  int64
 10  arrival_date                          36275 non-null  int64
 11  market_segment_type                   36275 non-null  int64
 12  repeated_guest                        36275 non-null  int64
 13  no_of_previous_cancellations          36275 non-null  int64
```

In [78]:
```
X = df.drop("booking_status" , axis=1)
y = df["booking_status"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.30, r
```

In [79]:
```
X = pd.get_dummies(X, drop_first=True)
X.head()
```

Out[79]:

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

In [80]:
```
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:
0    0.670644
1    0.329356
Name: booking_status, dtype: float64
Percentage of classes in test set:
0    0.676376
1    0.323624
Name: booking_status, dtype: float64
```

In [81]:
```
model = DecisionTreeClassifier(criterion="gini", random_state=1)
model.fit(X_train, y_train)
```

Out[81]:
```
DecisionTreeClassifier(random_state=1)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [82]:
```
print("Accuracy on training set : ",model.score(X_train, y_train))
print("Accuracy on test set : ",model.score(X_test, y_test))
```

```
Accuracy on training set :  0.994210775047259
Accuracy on test set :  0.8712671138472847
```

In [83]:
```
y.sum(axis = 0)
```

Out[83]:
```
11885
```

In [84]:
```
def model_performance_classification_sklearn(model, predictors, target):
    """
    Function to compute different metrics to check classification model pe

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

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

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

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

    return df_perf
```

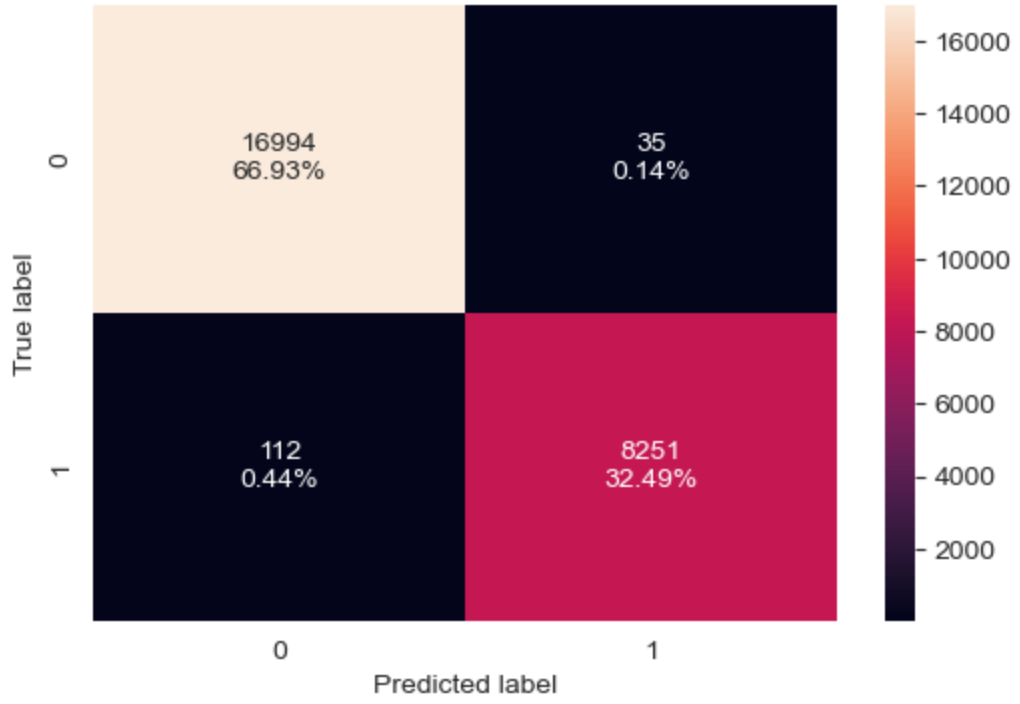
In [85]:
```
def confusion_matrix_sklearn(model, predictors, target):
    """
    To plot the confusion_matrix with percentages

    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.flatte
            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 [86]:
```
decision_tree_perf_train = model_performance_classification_sklearn(
    model, X_train, y_train)
decision_tree_perf_train
```

Out[86]:

	Accuracy	Recall	Precision	F1
0	0.994211	0.986608	0.995776	0.991171

In [87]:
```
confusion_matrix_sklearn(model, X_train, y_train)
```

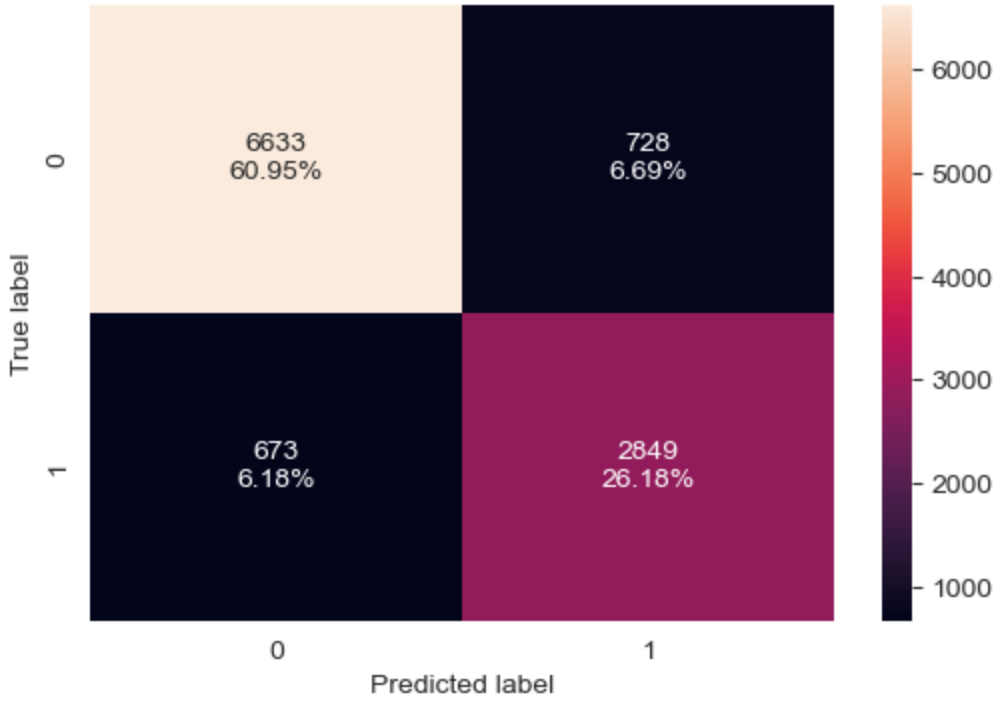


In [88]:
```
decision_tree_perf_test = model_performance_classification_sklearn(
    model, X_test, y_test
)
decision_tree_perf_test
```

Out[88]:

	Accuracy	Recall	Precision	F1
0	0.871267	0.808915	0.796477	0.802648

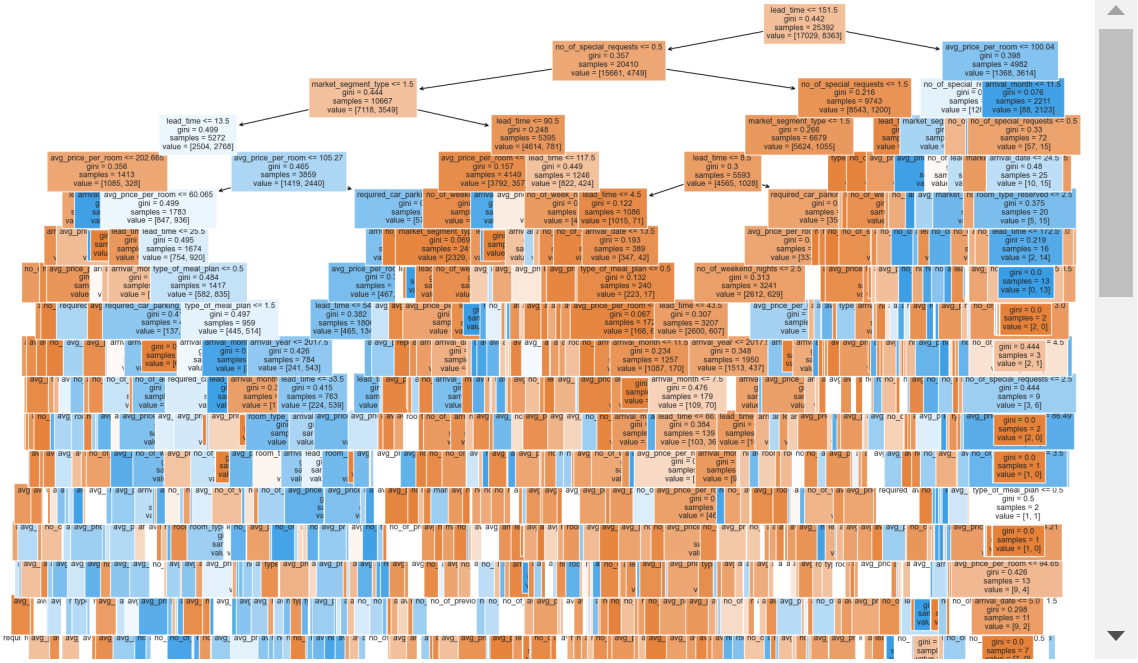
In [89]:
```
confusion_matrix_sklearn(model, X_test, y_test)
```



In [90]:
```
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_ni
ghts', 'type_of_meal_plan', 'required_car_parking_space', 'room_type_rese
rved', 'lead_time', 'arrival_year', 'arrival_month', 'arrival_date', 'mar
ket_segment_type', 'repeated_guest', 'no_of_previous_cancellations', 'no_
of_previous_bookings_not_canceled', 'avg_price_per_room', 'no_of_special_
requests']
```

In [91]:
```
names = list(X.columns)
plt.figure(figsize=(20,30))
out = tree.plot_tree(model,feature_names=feature_names,filled=True,fontsiz
for o in out:
      arrow = o.arrow_patch
      if arrow is not None:
         arrow.set_edgecolor('black')
         arrow.set_linewidth(1)
plt.show()
```

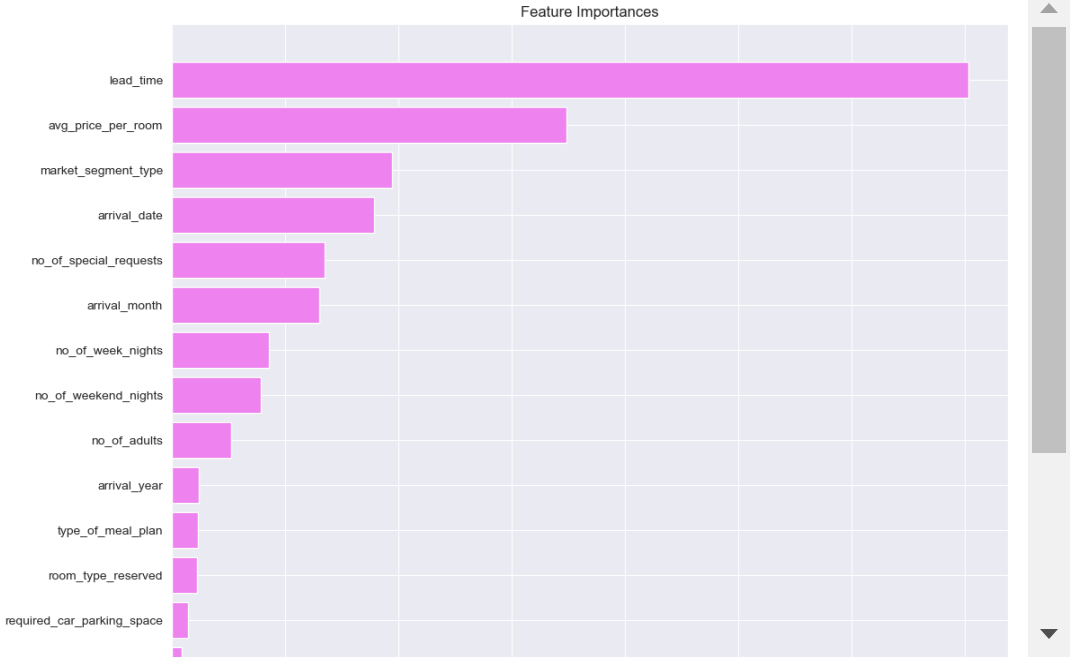


In [92]:
```
print(tree.export_text(model, feature_names=feature_names, show_weights=Tr
```

```
|--- lead_time <= 151.50
|   |--- no_of_special_requests <= 0.50
|   |   |--- market_segment_type <= 1.50
|   |   |   |--- lead_time <= 13.50
|   |   |   |   |--- avg_price_per_room <= 202.67
|   |   |   |   |   |--- lead_time <= 3.50
|   |   |   |   |   |   |--- arrival_month <= 5.50
|   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |--- weights: [56.00, 0.00] class:
0
|   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 77.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [24.00, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  77.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 26.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of d
epth 14
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [93]:
```
importances = model.feature_importances_
indices = np.argsort(importances)

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



In [94]:
```
estimator = DecisionTreeClassifier(random_state=1)

# Grid of parameters to choose from

parameters = {
    "max_depth": [np.arange(2, 50, 5), None],
    "criterion": ["entropy", "gini"],
    "splitter": ["best", "random"],
    "min_impurity_decrease": [0.000001, 0.00001, 0.0001],
}

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

# Run the grid search
grid_obj = GridSearchCV(estimator, parameters, scoring=acc_scorer, cv=5)
grid_obj = grid_obj.fit(X_train, y_train)

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

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

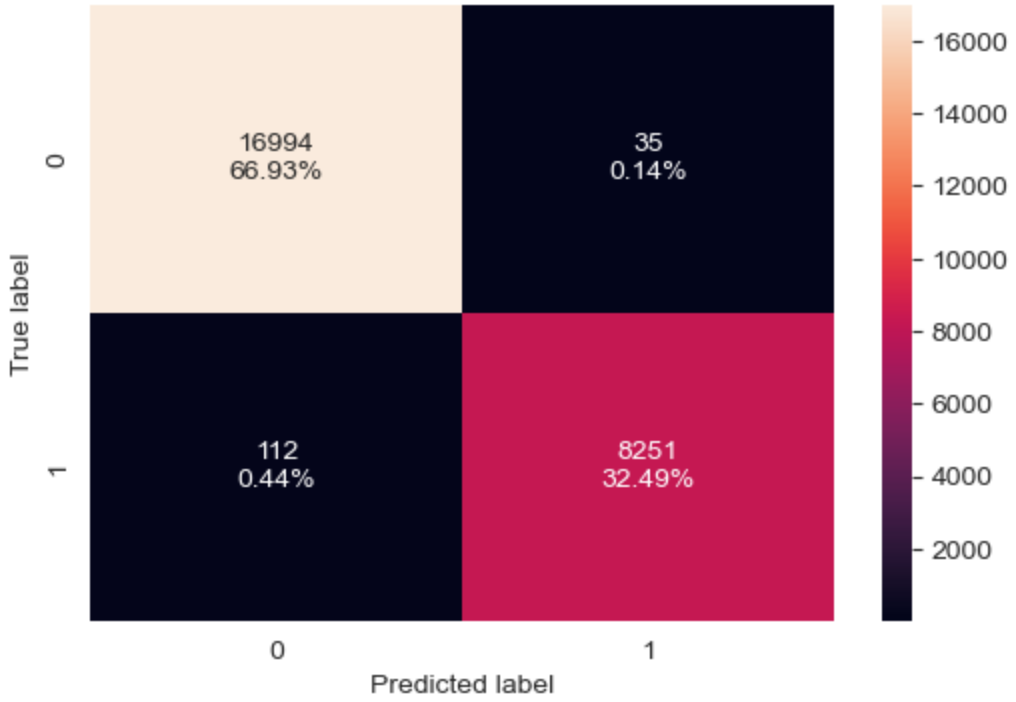
Out[94]:
```
DecisionTreeClassifier(criterion='entropy', min_impurity_decrease=1e-05,
                       random_state=1)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [95]:
```
decision_tree_tune_perf_train = model_performance_classification_sklearn(
    estimator, X_train, y_train
)
decision_tree_tune_perf_train
```

Out[95]:

	Accuracy	Recall	Precision	F1
0	0.994211	0.986608	0.995776	0.991171

In [96]:
```
confusion_matrix_sklearn(estimator, X_train, y_train)
```

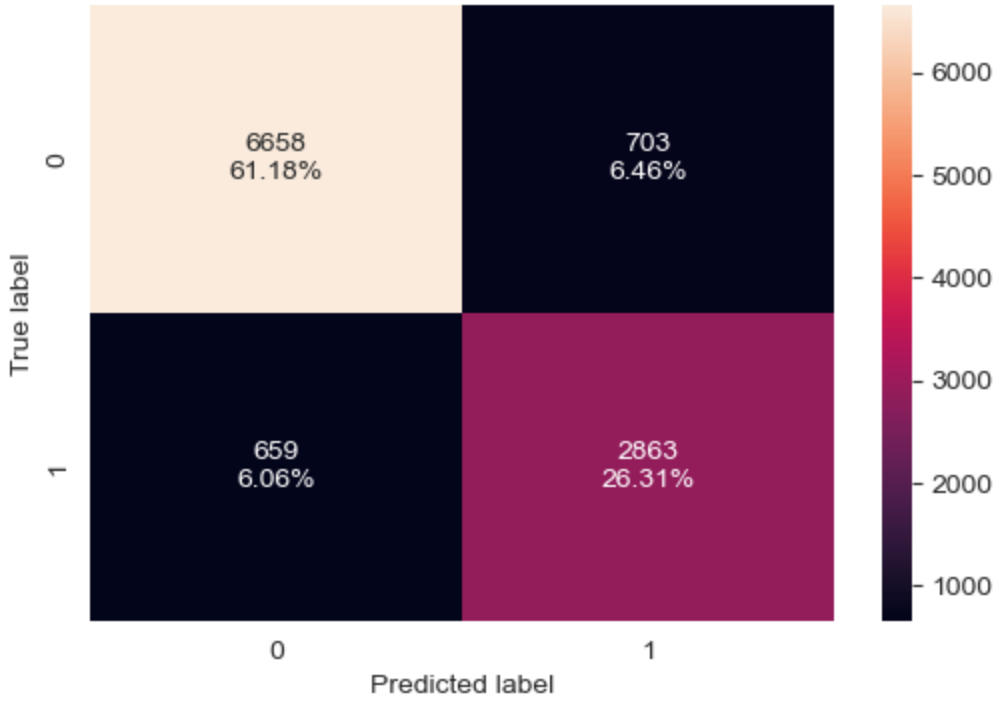


In [97]:
```
decision_tree_tune_perf_test = model_performance_classification_sklearn(
    estimator, X_test, y_test
)

decision_tree_tune_perf_test
```

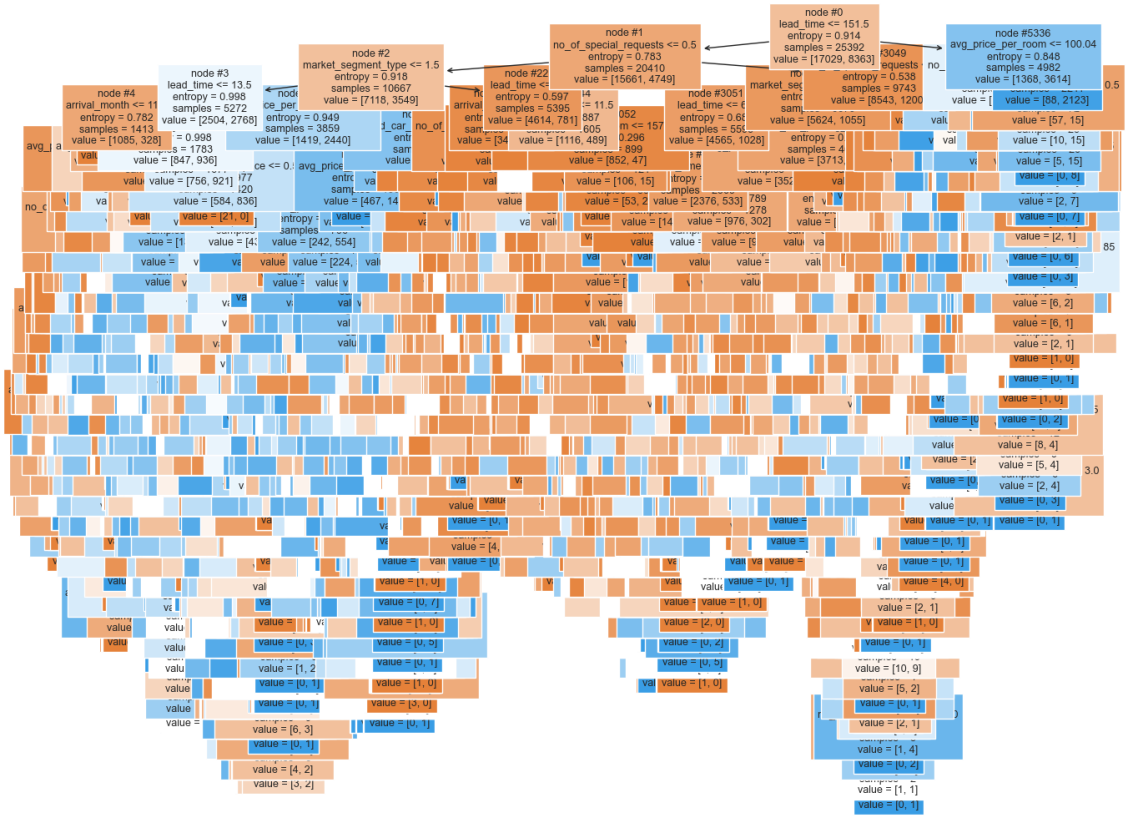
Out[97]:

	Accuracy	Recall	Precision	F1
0	0.874851	0.81289	0.80286	0.807844

In [98]:
```
confusion_matrix_sklearn(estimator, X_test, y_test)
```



In [99]:
```
plt.figure(figsize=(15, 12))

tree.plot_tree(
    estimator,
    feature_names=feature_names,
    filled=True,
    fontsize=9,
    node_ids=True,
    class_names=None,
)
plt.show()
```



I will now pron the desion tree

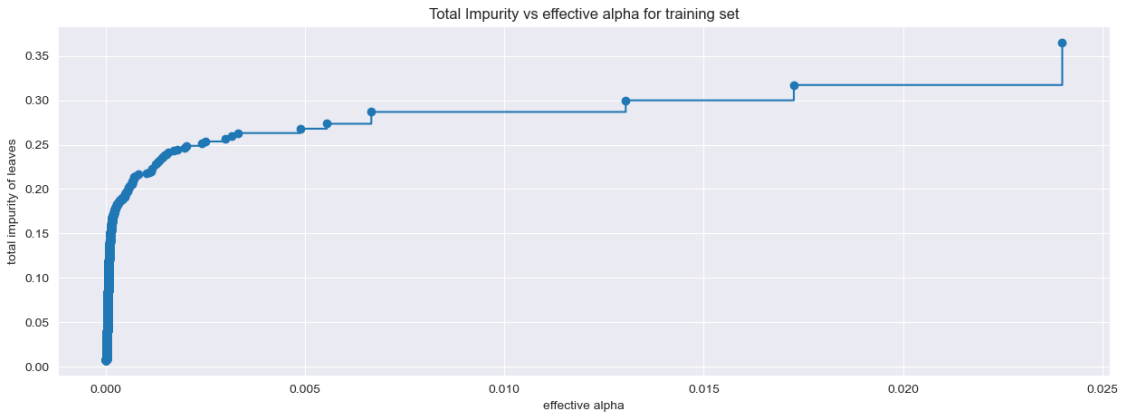
In [100]:
```
clf = DecisionTreeClassifier(random_state=1)
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
pd.DataFrame(path)
```

Out[100]:

	ccp_alphas	impurities
0	0.000000e+00	0.007572
1	4.327745e-07	0.007573
2	4.688391e-07	0.007573
3	5.329960e-07	0.007574
4	6.133547e-07	0.007575
...	...	...
1337	6.665684e-03	0.286897
1338	1.304480e-02	0.299942
1339	1.725993e-02	0.317202
1340	2.399048e-02	0.365183
1341	7.657789e-02	0.441761

1342 rows × 2 columns

In [101]:
```
fig, ax = plt.subplots(figsize=(15, 5))
ax.plot(ccp_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-pos
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()
```

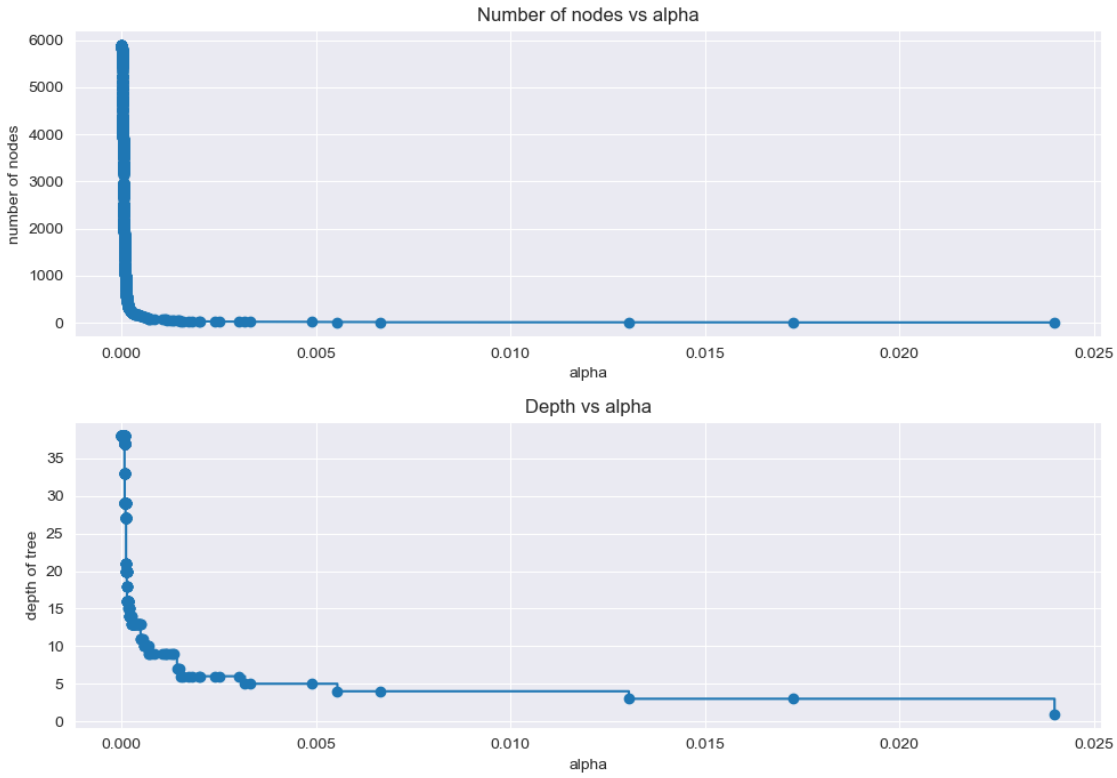


Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [102]:
```
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: 0.07657789477371357
```

In [103]:
```
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 [104]:
```
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)
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)
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="st
ax.plot(ccp_alphas, recall_test, marker="o", label="test", drawstyle="step
ax.legend()
plt.show()
```

In [105]:
```
index_best_model = np.argmax(recall_test)
best_model = clfs[index_best_model]
print(best_model)
```

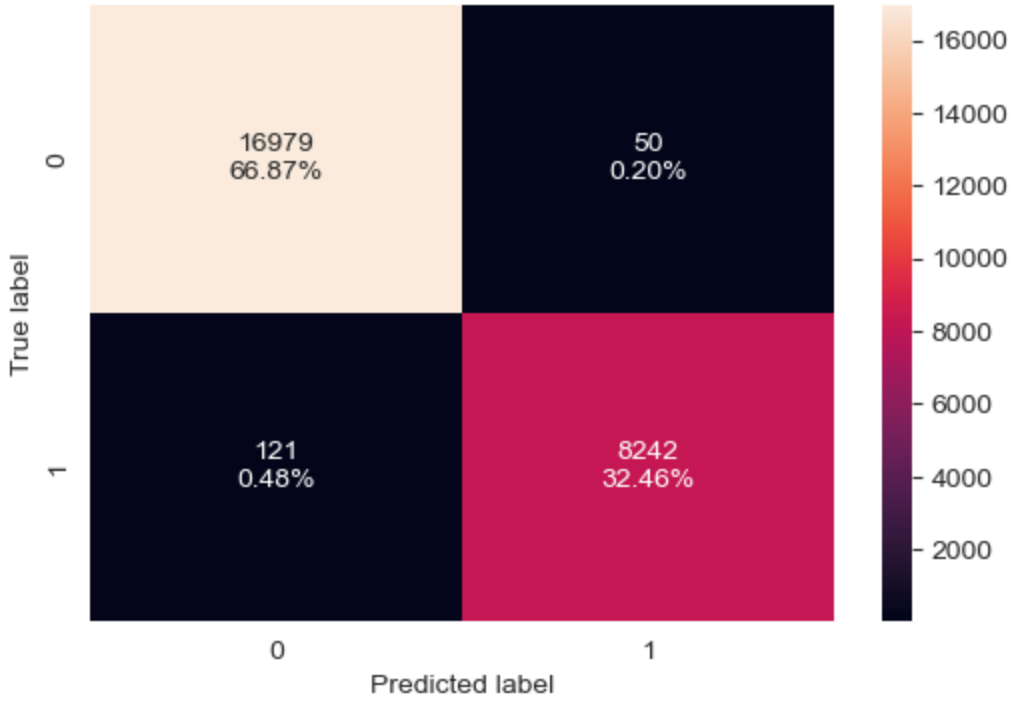
```
DecisionTreeClassifier(ccp_alpha=2.4891092944023008e-05, random_state=1)
```

In [106]:
```
decision_tree_postpruned_perf_train = model_performance_classification_skl
    best_model, X_train, y_train
)
decision_tree_postpruned_perf_train
```

Out[106]:

	Accuracy	Recall	Precision	F1
0	0.993266	0.985532	0.99397	0.989733

In [107]:
```
confusion_matrix_sklearn(best_model, X_train, y_train)
```

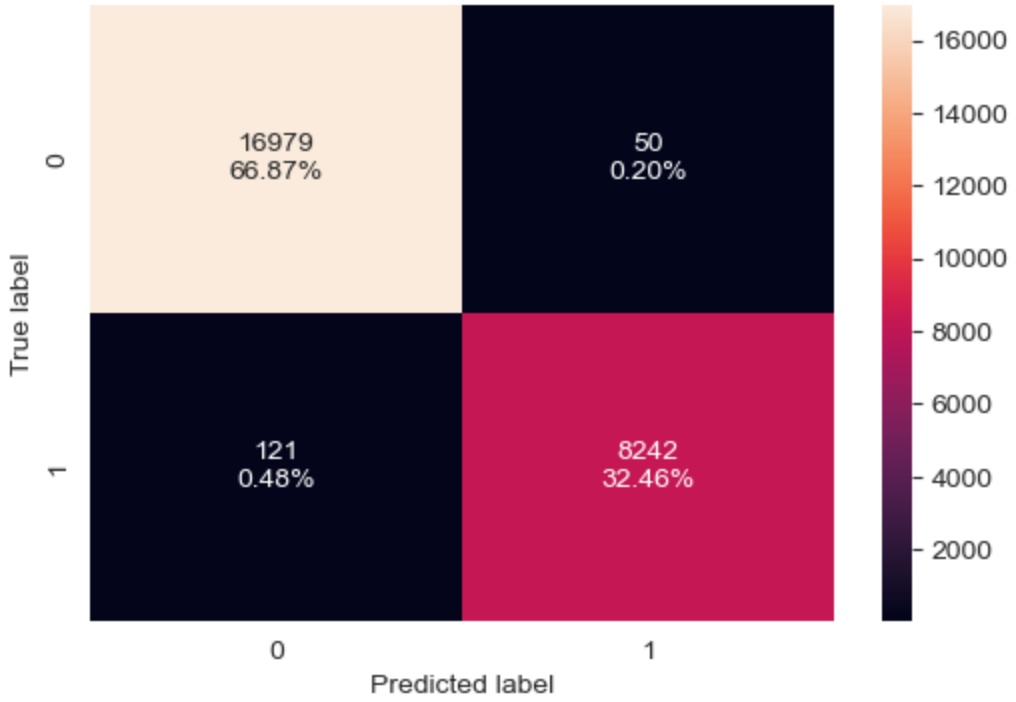


In [108]:
```
decision_tree_postpruned_perf_test = model_performance_classification_skle
    best_model, X_test, y_test
)
decision_tree_postpruned_perf_test
```

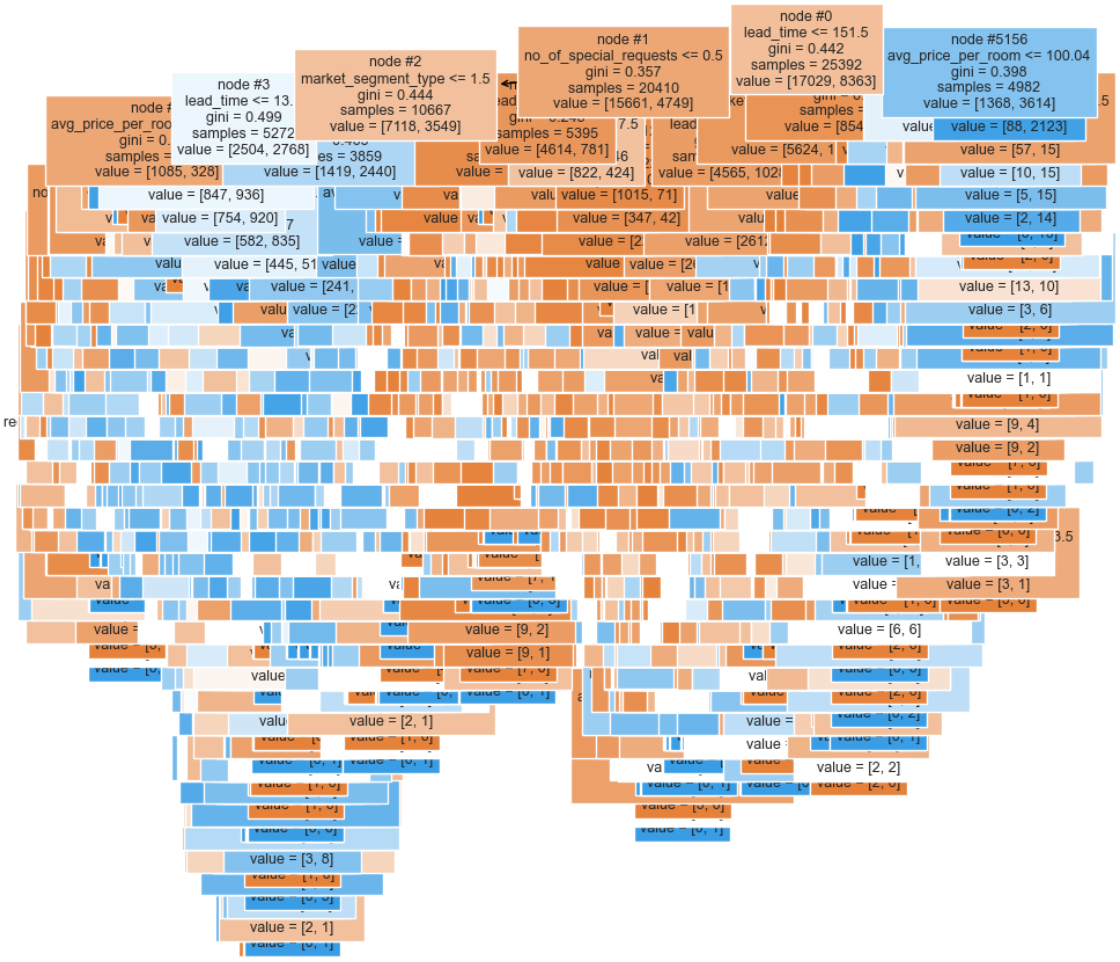
Out[108]:

	Accuracy	Recall	Precision	F1
0	0.871451	0.809483	0.796591	0.802985

In [109]:
```
confusion_matrix_sklearn(best_model, X_train, y_train)
```



In [110]:
```
plt.figure(figsize=(10, 10))

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



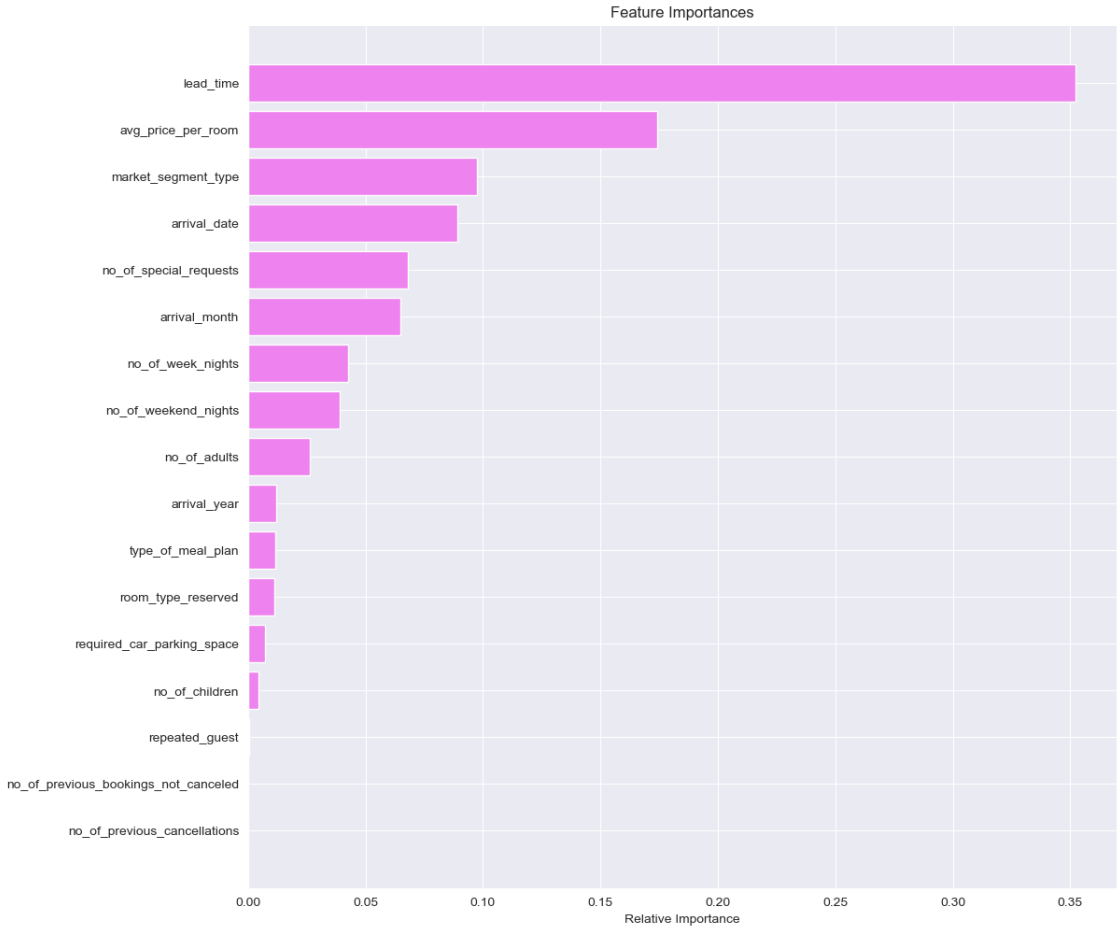
In [111]:
```
print(tree.export_text(best_model, feature_names=feature_names, show_weigh
```

```
|--- lead_time <= 151.50
|   |--- no_of_special_requests <= 0.50
|   |   |--- market_segment_type <= 1.50
|   |   |   |--- lead_time <= 13.50
|   |   |   |   |--- avg_price_per_room <= 202.67
|   |   |   |   |   |--- lead_time <= 3.50
|   |   |   |   |   |   |--- arrival_month <= 5.50
|   |   |   |   |   |   |   |--- no_of_weekend_nights <= 1.50
|   |   |   |   |   |   |   |   |--- arrival_month <= 1.50
|   |   |   |   |   |   |   |   |   |--- weights: [56.00, 0.00] class:
0
|   |   |   |   |   |   |   |   |--- arrival_month >  1.50
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room <= 77.50
|   |   |   |   |   |   |   |   |   |   |--- weights: [24.00, 0.00] cl
ass: 0
|   |   |   |   |   |   |   |   |   |--- avg_price_per_room >  77.50
|   |   |   |   |   |   |   |   |   |   |--- arrival_date <= 26.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of d
epth 14
```

In [112]:
```
print(pd.DataFrame(best_model.feature_importances_, columns=["Imp"], index
```

	Imp
lead_time	0.352079
avg_price_per_room	0.174046
market_segment_type	0.097631
arrival_date	0.088920
no_of_special_requests	0.068086
arrival_month	0.064782
no_of_week_nights	0.042594
no_of_weekend_nights	0.039044
no_of_adults	0.026149
arrival_year	0.011808
type_of_meal_plan	0.011506
room_type_reserved	0.011142
required_car_parking_space	0.007128
no_of_children	0.004282
repeated_guest	0.000591
no_of_previous_bookings_not_canceled	0.000213
no_of_previous_cancellations	0.000000

In [113]:
```
importances = best_model.feature_importances_
indices = np.argsort(importances)

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



We see the top the final factors are lead time, average price of room, and the market segment type.

In [114]:
```
models_train_comp_df = pd.concat(
    [
        decision_tree_perf_train.T,
        decision_tree_tune_perf_train.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("Training performance comparison:")
models_train_comp_df
```

```
Training performance comparison:
```

Out[114]:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)	Decision Tree (Post-Pruning)
Accuracy	0.994211	0.994211	0.993266
Recall	0.986608	0.986608	0.985532
Precision	0.995776	0.995776	0.993970
F1	0.991171	0.991171	0.989733

In [115]:
```
models_train_comp_df = pd.concat(
    [
        decision_tree_perf_test.T,
        decision_tree_tune_perf_test.T,
        decision_tree_postpruned_perf_test.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[115]:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)	Decision Tree (Post-Pruning)
Accuracy	0.871267	0.874851	0.871451
Recall	0.808915	0.812890	0.809483
Precision	0.796477	0.802860	0.796591
F1	0.802648	0.807844	0.802985

The final model true is desplaying high accuracy

Both the Logistical regression model and the desion tree were very accurate in there predictions Lead time is by far the biggest driving factor in cancellations My reccomendation would be to try and reduce the average number of days between booking and arriving. You could also slightly over sell rooms far in advanced and then wait for cancellations leaving you with less empty rooms.