Hotel Booking Demand Dataset Analysis

The dataset used was downloaded from Kaggle:

https://www.kaggle.com/jessemostipak/hotel-booking-demand (https://www.kaggle.com/jessemostipak/hotel-booking-demand)

The related article explaining the dataset and use is at link below:

https://www.sciencedirect.com/science/article/pii/S2352340918315191 (https://www.sciencedirect.com/science/article/pii/S2352340918315191)

The article and initial data exploration shows that over 40% of bookings made are canceled.

The main objective is to develop a cancellation prediction model using the dataset.

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report,confusion_matrix, mean_squared_error,
accuracy_score
from math import sqrt
from sklearn.preprocessing import StandardScaler
```

In [3]:

```
hotel_bookings = pd.read_csv('data/hotel_bookings.csv')
```

In [4]:

```
hotel_bookings
```

Out[4]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_we
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	
3	Resort Hotel	0	13	2015	July	
4	Resort Hotel	0	14	2015	July	
...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	
119386	City Hotel	0	102	2017	August	
119387	City Hotel	0	34	2017	August	
119388	City Hotel	0	109	2017	August	
119389	City Hotel	0	205	2017	August	

119390 rows × 32 columns

There are two hotels in the dataset, one is resort hotel and other is a city hotel,

both located in Portugal as explained in the article

In [5]:

```
hotel_bookings['hotel'].value_counts()
```

Out[5]:

```
City Hotel      79330
Resort Hotel    40060
Name: hotel, dtype: int64
```

We will select City Hotel for analysis to keep it simple.

In [4]:

```
city_hotel_bookings = hotel_bookings[hotel_bookings['hotel'] == 'City Hotel']
```

In [7]:

```
city_hotel_bookings
```

Out[7]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_weel
40060	City Hotel	0	6	2015	July	
40061	City Hotel	1	88	2015	July	
40062	City Hotel	1	65	2015	July	
40063	City Hotel	1	92	2015	July	
40064	City Hotel	1	100	2015	July	
...	...	...	...	...	...	
119385	City Hotel	0	23	2017	August	
119386	City Hotel	0	102	2017	August	
119387	City Hotel	0	34	2017	August	
119388	City Hotel	0	109	2017	August	
119389	City Hotel	0	205	2017	August	

79330 rows × 32 columns

In [8]:

```
# we will check the different column variables for any null values
city_hotel_bookings.isnull().sum()
```

Out[8]:

```
hotel                               0
is_canceled                         0
lead_time                           0
arrival_date_year                   0
arrival_date_month                  0
arrival_date_week_number            0
arrival_date_day_of_month           0
stays_in_weekend_nights             0
stays_in_week_nights                0
adults                              0
children                            4
babies                              0
meal                                0
country                            24
market_segment                      0
distribution_channel                0
is_repeated_guest                   0
previous_cancellations              0
previous_bookings_not_canceled      0
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
agent                            8131
company                         75641
days_in_waiting_list                0
customer_type                       0
adr                                 0
required_car_parking_spaces         0
total_of_special_requests           0
reservation_status                  0
reservation_status_date             0
dtype: int64
```

In [12]:

```
# as seen, the column for 'company' is almost empty, therefore we will just drop this c
olumn
# we will drop the 'hotel' column as we are using city hotel data only
# and for column 'agent', we will drop the rows where we have null values, as these are
about 10% of total rows
# for the children and country column, we will fill these with most common value as the
se are very low in numbers
```

In [5]:

```
city_hotel_bookings = city_hotel_bookings.drop(columns = ['company', 'hotel'])
```

In [10]:

```
city_hotel_bookings['children'].mode()
```

Out[10]:

```
0    0.0
dtype: float64
```

In [11]:

```
city_hotel_bookings['country'].mode()
# most frequent country of the customers is Portugal
```

Out[11]:

```
0    PRT
dtype: object
```

In [6]:

```
city_hotel_bookings['children'] = city_hotel_bookings['children'].fillna(value = 0)
```

In [7]:

```
city_hotel_bookings['country'] = city_hotel_bookings['country'].fillna(value = 'PRT')
```

In [8]:

```
city_hotel_bookings.dropna(inplace = True)
```

In [15]:

```
# check again for nulls
city_hotel_bookings.isnull().sum()
```

Out[15]:

```
is_canceled                       0
lead_time                         0
arrival_date_year                 0
arrival_date_month                0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          0
babies                            0
meal                              0
country                           0
market_segment                    0
distribution_channel              0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
assigned_room_type                0
booking_changes                   0
deposit_type                      0
agent                             0
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
reservation_status                0
reservation_status_date           0
dtype: int64
```

In [16]:

```
city_hotel_bookings.shape
```

Out[16]:

```
(71199, 30)
```

In [9]:

```
city_hotel_bookings.reset_index(inplace = True)
```

In [10]:

```
city_hotel_bookings.drop(columns = 'index', inplace = True)
```

In [19]:

```
# dataframe now has no null values
city_hotel_bookings
```

Out[19]:

	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_numbe
0	0	6	2015	July	2
1	1	88	2015	July	2
2	1	65	2015	July	2
3	1	92	2015	July	2
4	1	100	2015	July	2
...	...	...	...	...	.
71194	0	23	2017	August	3
71195	0	102	2017	August	3
71196	0	34	2017	August	3
71197	0	109	2017	August	3
71198	0	205	2017	August	3

71199 rows × 30 columns

In [20]:

```
# let's look at basic statistics of the dataset
city_hotel_bookings.describe()
```

Out[20]:

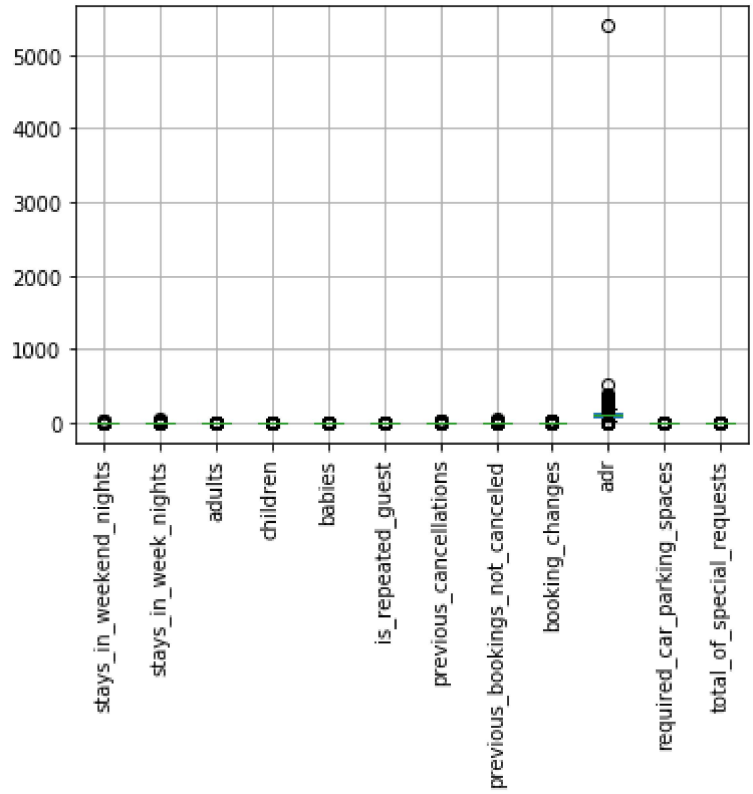
	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_d
count	71199.000000	71199.000000	71199.000000	71199.000000	
mean	0.428278	116.841725	2016.167966	27.333137	
std	0.494833	112.535690	0.698185	13.197539	
min	0.000000	0.000000	2015.000000	1.000000	
25%	0.000000	29.000000	2016.000000	17.000000	
50%	0.000000	82.000000	2016.000000	28.000000	
75%	1.000000	172.000000	2017.000000	38.000000	
max	1.000000	629.000000	2017.000000	53.000000	

In [21]:

```
# let's check for outliers of important numerical variables
city_hotel_bookings.drop(columns = ['is_canceled','agent','lead_time','days_in_waiting_
list', 'arrival_date_year',
                                    'arrival_date_week_number', 'arrival_date_day_of_mon
th'], axis = 1).boxplot(rot = 90)
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x250d56bb588>
```



In [ ]:

```
# let's drop the rows where the count of 'adr' is above 1000, as it seems to be anomaly
```

In [22]:

```
city_hotel_bookings.drop(city_hotel_bookings[city_hotel_bookings['adr'] > 1000].index,
inplace = True)
```

In [23]:

```
city_hotel_bookings.reset_index(inplace = True)
```

In [24]:

```
city_hotel_bookings.drop(columns= 'index', inplace = True)
```

In [13]:

```
# there seem to be no outliers in the data now
# let's add a new column for booking revenue total
city_hotel_bookings['booking_revenue'] = (city_hotel_bookings['stays_in_week_nights'] +
city_hotel_bookings['stays_in_weekend_nights'])* (city_hotel_bookings['adults'] + city_
hotel_bookings['children']) * city_hotel_bookings['adr']
```

In [28]:

```
# how many bookings have adr = 0?
city_hotel_bookings['adr'][city_hotel_bookings['adr'] == 0].count()
```

Out[28]:

```
600
```

In [29]:

```
# total potential revenue for entire year of 2016
city_hotel_bookings[city_hotel_bookings['arrival_date_year'] == 2016]['booking_revenue'
].sum()
```

Out[29]:

```
23411785.910000004
```

In [31]:

```
# let's set the categorical variables
city_hotel_bookings[['arrival_date_month', 'meal', 'country', 'market_segment', 'distri
bution_channel', 'reserved_room_type',
                     'assigned_room_type', 'deposit_type','agent', 'customer_type', 'rese
rvation_status']] = city_hotel_bookings[['arrival_date_month', 'meal', 'country', 'mark
et_segment', 'distribution_channel', 'reserved_room_type',
                     'assigned_room_type', 'deposit_type','agent', 'customer_type', 'rese
rvation_status']].astype('category')
city_hotel_bookings.dtypes
```

Out[31]:

```
is_canceled                        int64
lead_time                          int64
arrival_date_year                  int64
arrival_date_month              category
arrival_date_week_number           int64
arrival_date_day_of_month          int64
stays_in_weekend_nights            int64
stays_in_week_nights               int64
adults                             int64
children                         float64
babies                             int64
meal                            category
country                         category
market_segment                  category
distribution_channel            category
is_repeated_guest                  int64
previous_cancellations             int64
previous_bookings_not_canceled     int64
reserved_room_type              category
assigned_room_type              category
booking_changes                    int64
deposit_type                    category
agent                           category
days_in_waiting_list               int64
customer_type                   category
adr                              float64
required_car_parking_spaces        int64
total_of_special_requests          int64
reservation_status              category
reservation_status_date           object
booking_revenue                  float64
dtype: object
```

In [32]:

```
# let's set the date as datetime
city_hotel_bookings['reservation_status_date'] = city_hotel_bookings['reservation_statu
s_date'].astype('datetime64[ns]')
```

In [33]:

```
# number of canceled bookings
city_hotel_bookings.is_canceled.value_counts()
```

Out[33]:

```
0    40706
1    30492
Name: is_canceled, dtype: int64
```

In [34]:

```
# % of bookings canceled
30492 / (40706 + 30492) * 100
```

Out[34]:

```
42.827045703531
```

In [ ]:

```
# whopping 43% of bookings are canceled
# assuming the hotel is not overbooking already, it is important to predict cancellatio
ns based on available data
```

In [35]:

```
# let's use pivot table to see any obvious information using numerical features that ca
use cancellation
city_hotel_bookings.pivot_table(index = 'is_canceled', values = ['lead_time', 'stays_in
_weekend_nights',
 'stays_in_week_nights', 'adults', 'children','babies', 'previous_cancellations','previ
ous_bookings_not_canceled',
    'booking_changes', 'days_in_waiting_list', 'adr', 'required_car_parking_spaces','to
tal_of_special_requests'],
                           aggfunc = 'mean')
```

Out[35]:

is_canceled	adr	adults	babies	booking_changes	children	days_in_waiting_list
0	108.174479	1.879526	0.006510	0.248637	0.103670	2.432639
1	104.594111	1.909255	0.001902	0.081005	0.083924	5.044733

In [ ]:

```
# it looks like cancellations could be caused by higher number of days in waiting list,
higher lead time and previous cancellations
```

In [37]:

```
# average values for all numerical features
city_hotel_bookings[['lead_time', 'booking_changes', 'stays_in_weekend_nights', 'stays_
in_week_nights', 'adults', 'children',
 'babies', 'previous_cancellations', 'previous_bookings_not_canceled', 'days_in_waiting
_list', 'adr',
 'required_car_parking_spaces', 'total_of_special_requests']][city_hotel_bookings['is_c
anceled'] == 0].apply(np.mean)
```

Out[37]:

```
lead_time                         88.254066
booking_changes                    0.248637
stays_in_weekend_nights            0.839508
stays_in_week_nights               2.195106
adults                             1.879526
children                           0.103670
babies                             0.006510
previous_cancellations             0.007837
previous_bookings_not_canceled     0.022577
days_in_waiting_list               2.432639
adr                              108.174479
required_car_parking_spaces        0.036457
total_of_special_requests          0.775021
dtype: float64
```

In [38]:

```
# average values for all numerical features
city_hotel_bookings[['lead_time', 'booking_changes', 'stays_in_weekend_nights', 'stays_
in_week_nights', 'adults', 'children',
 'babies', 'previous_cancellations', 'previous_bookings_not_canceled', 'days_in_waiting
_list', 'adr',
 'required_car_parking_spaces', 'total_of_special_requests']][city_hotel_bookings['is_c
anceled'] == 1].apply(np.mean)
```

Out[38]:

```
lead_time                        155.008166
booking_changes                    0.081005
stays_in_weekend_nights            0.800013
stays_in_week_nights               2.285026
adults                             1.909255
children                           0.083924
babies                             0.001902
previous_cancellations             0.160501
previous_bookings_not_canceled     0.001017
days_in_waiting_list               5.044733
adr                              104.594111
required_car_parking_spaces        0.000000
total_of_special_requests          0.288994
dtype: float64
```

aggregate and visualize the clusters as they exist across the time range

In [40]:

```
# let's plot important features to understand correlations to bookings not canceled vs. those canceled
sns.pairplot(data = city_hotel_bookings[['lead_time',
 'stays_in_weekend_nights', 'stays_in_week_nights', 'previous_cancellations', 'previous_bookings_not_canceled',
 'booking_changes', 'days_in_waiting_list','is_canceled']], hue = 'is_canceled');
# x axis scale is used for self-correlation, e.g. lower lead time correlates with less cancellations
# lead_time vs. booking_changes: higher changes relates to less cancellations irrespective of lead time
# if it depended on lead time also, it should have less cancellations in both axis,
# higher the previous bookings not canceled and higher booking changes, lower the cancellations,
# booking changes vs. lead time: higher the changes and lower the time, less cancellations
# higher the previous bookings not canceled and lower the stays, less cancellations
# higher booking changes and higher stays, lower cancellations
# higher wait time and higher lead time, higher cancellation
# but correlation doesn't mean causation
```

```
C:\Users\vaibh\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmo
dels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered i
n true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\vaibh\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmo
dels\nonparametric\kdetools.py:34: RuntimeWarning: invalid value encounter
ed in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```


lead_time
stays_in_weekend_nights
stays_in_week_nights
previous_cancellations
previous_bookings_not_canceled
booking_changes
days_in_waiting_list
is_canceled
is_canceled
0
1

In [46]:

```
city_hotel_bookings[['booking_revenue']] = city_hotel_bookings[['booking_revenue']].astype('int64')
```

In [47]:

```
# let's look at adr statistics by customer type
city_hotel_bookings.groupby('customer_type')['adr'].describe()
```

Out[47]:

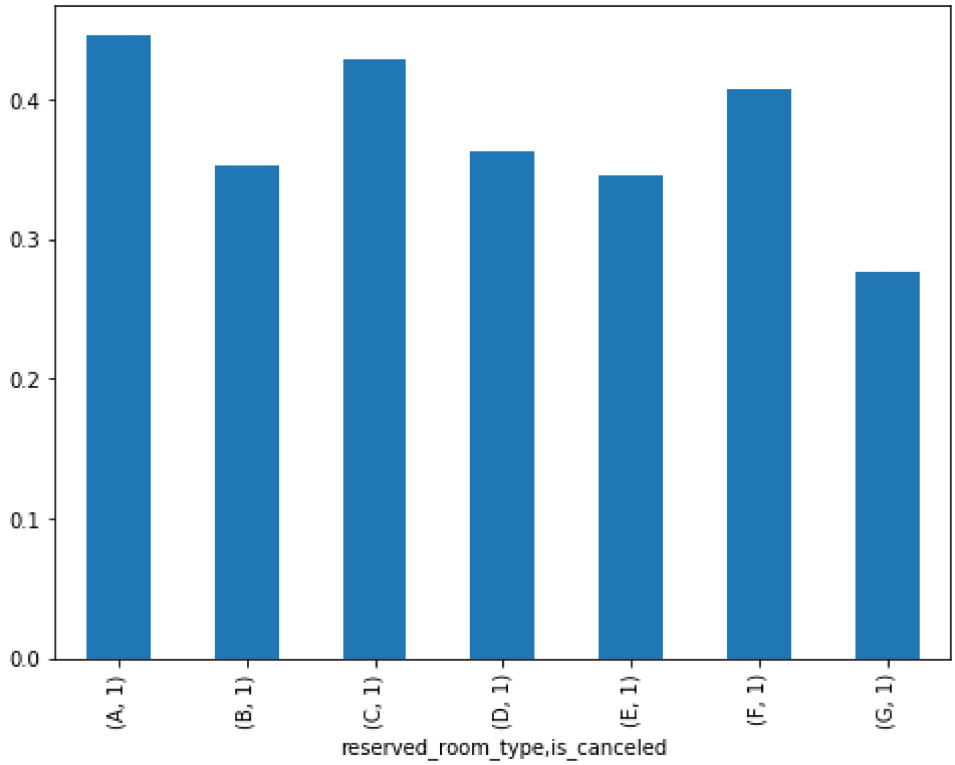
customer_type	count	mean	std	min	25%	50%	75%	max
Contract	2293.0	94.199538	32.619262	0.0	62.00	90.95	113.33	250.0
Group	262.0	90.636641	38.921407	0.0	64.85	85.00	109.00	290.0
Transient	53218.0	111.768649	39.572711	0.0	85.00	105.60	132.30	510.0
Transient-Party	15425.0	91.071876	30.258466	0.0	70.00	90.00	109.00	365.0

In [ ]:

```
# customer type 'transient' is most common that has higher mean adr, which is expected
```

In [49]:

```
# percentage of cancellations by reserved room type
cancellation_by_reserved_room_type = city_hotel_bookings[city_hotel_bookings['is_cancel
ed'] == 1].groupby('reserved_room_type')['is_canceled'].value_counts()
bookings_by_reserved_room_type = city_hotel_bookings.groupby('reserved_room_type')['is_
canceled'].count()
cancellation_by_reserved_room_type.div(bookings_by_reserved_room_type).plot.bar(rot= 90
, figsize=(8,6))
plt.show()
```

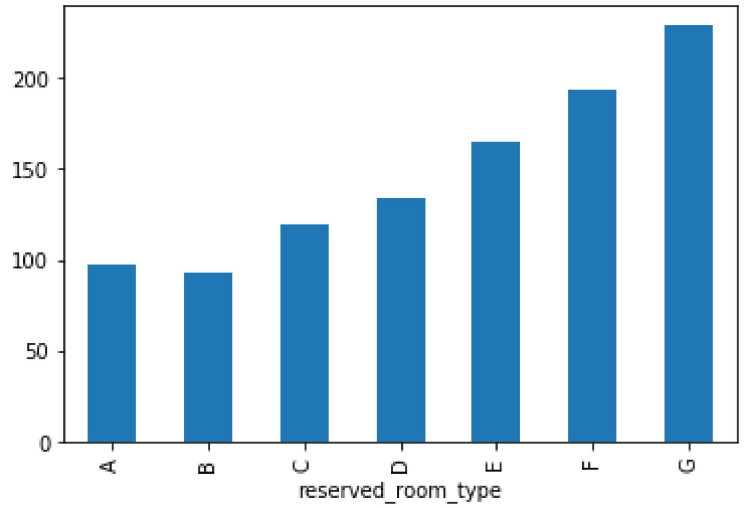


In [50]:

```
# adr by reserved room type
city_hotel_bookings.groupby('reserved_room_type')['adr'].mean().plot.bar()
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x250de209bc8>
```



In [ ]:

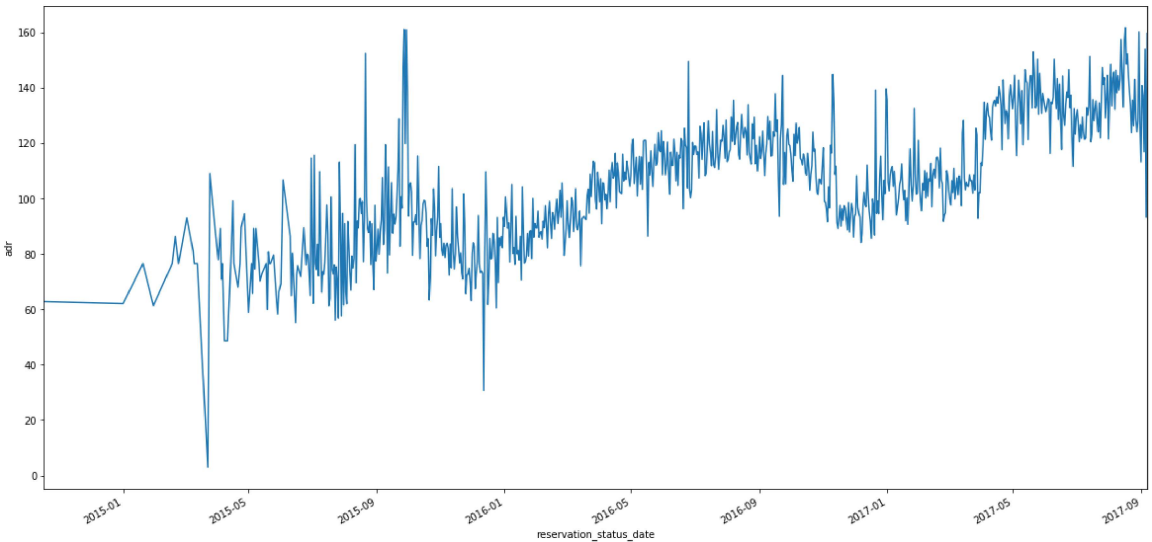
```
# looks like higher the adr by room type, lower the percentage of cancellations when ab
ove two graphs are compared
# thus combination of room type and adr does seem to have effect on cancellations
```

In [51]:

```
# adr over the year
city_hotel_bookings.groupby('reservation_status_date')['adr'].mean().plot(figsize=(20,1
0))
plt.ylabel('adr')
# adr shows seasonality, being higher around september each year, and lower during wint
er months
```

Out[51]:

```
Text(0, 0.5, 'adr')
```

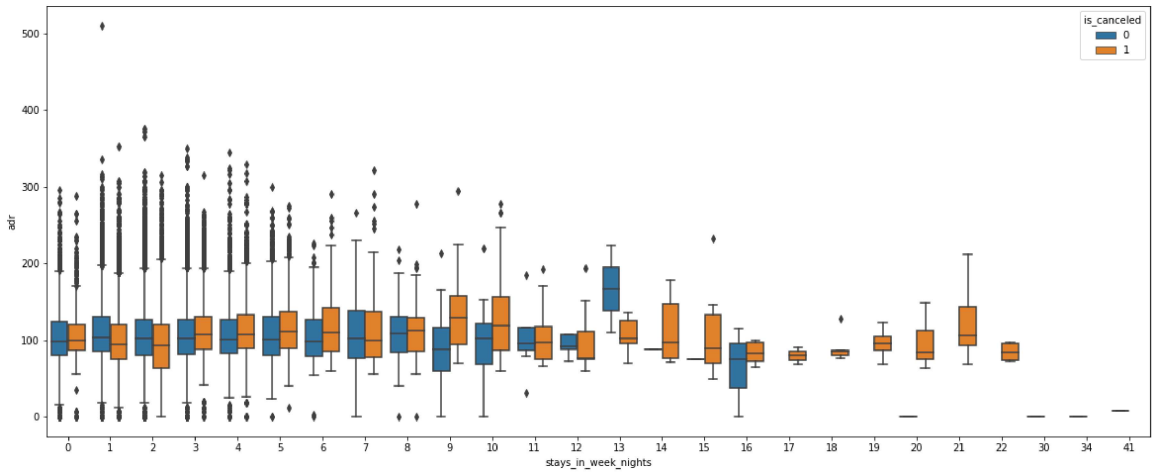


In [59]:

```
fig = plt.figure(figsize=(20,8))
sns.boxplot(x = 'stays_in_week_nights', y = 'adr', data= city_hotel_bookings, hue = 'is
_canceled')
# for more than two weeks of stays, higher the days of stays in week nights, higher the
cancellations
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x250d5da8c08>
```

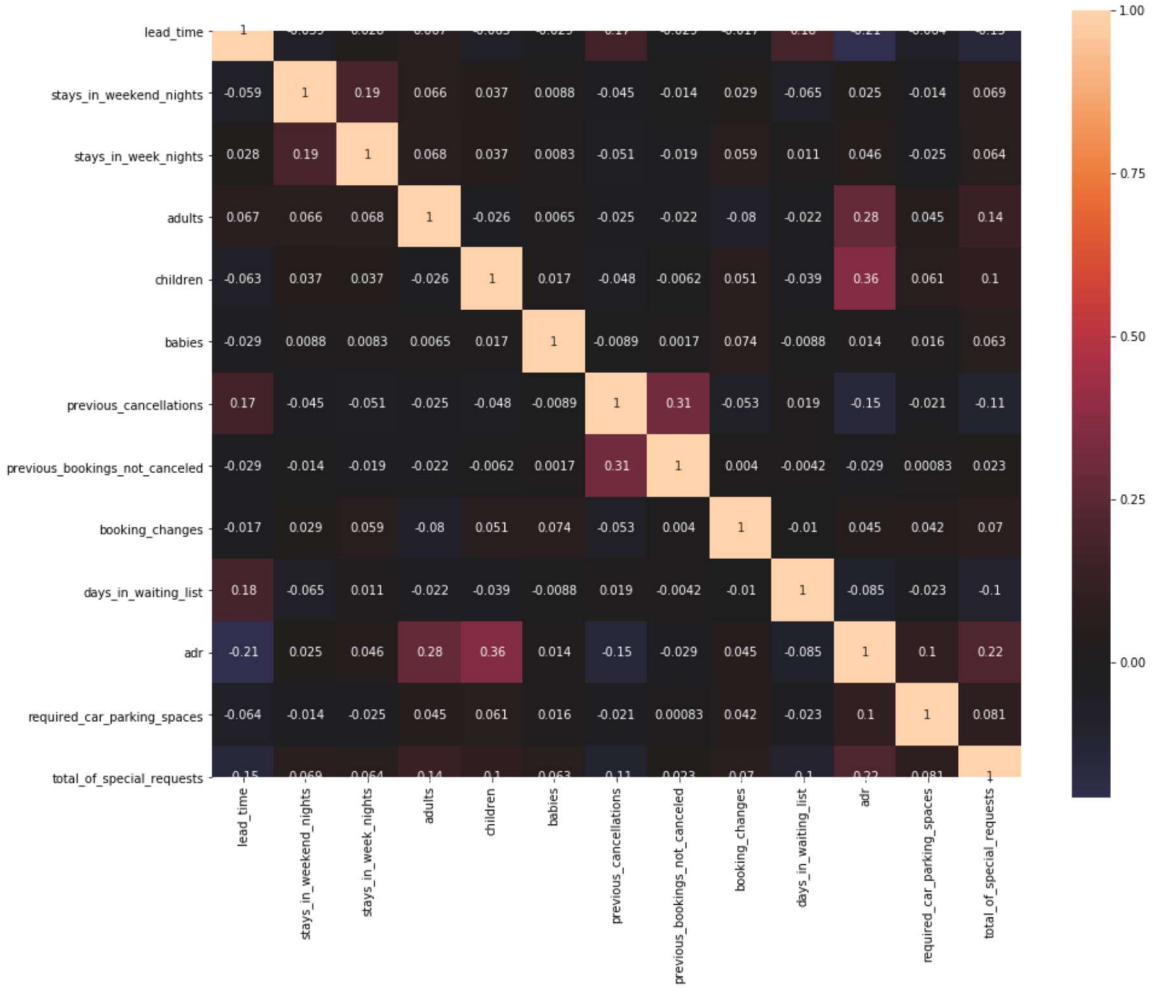


In [103]:

```
# checking for correlated numerical features
fig = plt.figure(figsize=(15,12))
sns.heatmap(city_hotel_bookings[['lead_time', 'stays_in_weekend_nights', 'stays_in_week
_nights', 'adults', 'children',
 'babies', 'previous_cancellations', 'previous_bookings_not_canceled', 'booking_change
s', 'days_in_waiting_list', 'adr',
 'required_car_parking_spaces', 'total_of_special_requests']].corr(), center = 0, squar
e = True, annot = True)
# none of the features are highly correlated with each other, thus all features are imp
ortant for prediction
```

Out[103]:

<matplotlib.axes._subplots.AxesSubplot at 0x260a80602c8>



In [11]:

```
# Classification (data is already labeled whether canceled or not)
# label encoding important and relevant categorical features for classification
OHE_city_hotel_bookings = pd.get_dummies(data = city_hotel_bookings, columns = ['arriva
l_date_month', 'arrival_date_week_number',
                                'meal', 'market_segment', 'distribution_channel',
                               'previous_bookings_not_canceled','reserved_room_type', 'dep
osit_type',
                               'customer_type'])
```

In [69]:

```
X = OHE_city_hotel_bookings.drop(columns=['is_canceled', 'arrival_date_year', 'arrival_
date_day_of_month',
                                          'country', 'assigned_room_type','agent', 'res
ervation_status',
                                         'reservation_status_date', 'booking_revenue'])
.copy()
X
```

Out[69]:

	lead_time	stays_in_weekend_nights	stays_in_week_nights	adults	children	babies	i
0	6	0	2	1	0.0	0	
1	88	0	4	2	0.0	0	
2	65	0	4	1	0.0	0	
3	92	2	4	2	0.0	0	
4	100	0	2	2	0.0	0	
...	...	...	...	...	...	...	
71193	23	2	5	2	0.0	0	
71194	102	2	5	3	0.0	0	
71195	34	2	5	2	0.0	0	
71196	109	2	5	2	0.0	0	
71197	205	2	7	2	0.0	0	

71198 rows × 131 columns

In [70]:

```
y = OHE_city_hotel_bookings['is_canceled']
```

In [71]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
324)
```

In [72]:

```
X_train.shape[0]
```

Out[72]:

```
47702
```

In [73]:

```
X_test.shape[0]
```

Out[73]:

```
23496
```

Method: DecisionTreeClassifier

In [74]:

```
# using decision tree classifier to begin with:
canceled_classifier = DecisionTreeClassifier(max_leaf_nodes=10, random_state=0)
canceled_classifier.fit(X_train, y_train)
```

Out[74]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=10,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [75]:

```
predictions = canceled_classifier.predict(X_test)
```

In [76]:

```
predictions
```

Out[76]:

```
array([1, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [77]:

```
y_test[:10]
```

Out[77]:

```
7577     1
69297    0
6348     0
3999     1
17468    1
13757    0
33014    1
53349    0
45229    0
51637    0
Name: is_canceled, dtype: int64
```

In [78]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.78      0.90      0.84     13360
           1       0.84      0.67      0.74     10136

    accuracy                           0.80     23496
   macro avg       0.81      0.79      0.79     23496
weighted avg       0.81      0.80      0.80     23496
```

In [79]:

```
print(confusion_matrix(y_test, predictions))
```

```
[[12049  1311]
 [ 3342  6794]]
```

In [80]:

```
accuracy_score(y_true = y_test, y_pred = predictions)
# the model predicts if a booking will be canceled correctly 80% of the time, which is
 pretty good
```

Out[80]:

```
0.8019662921348315
```

In [81]:

```
# sensitivity or true recall rate:
12049 / (12049 + 3342) * 100
```

Out[81]:

```
78.2860113053083
```

In [82]:

```
# precision:
12049 / (12049 + 1311) * 100
```

Out[82]:

```
90.187125748503
```

In [83]:

```
#specificity (showing again that classifier improves in correctly classifying negative
s)
6794 / (6794 + 1311) * 100
```

Out[83]:

```
83.82479950647749
```

Method: DecisionTreeClassifier without numeric features

In [16]:

```
# Classification without numerical features
U = OHE_city_hotel_bookings.drop(columns=['is_canceled','lead_time', 'arrival_date_yea
r', 'arrival_date_day_of_month',
                                          'stays_in_weekend_nights','stays_in_week_nigh
ts', 'adults', 'children', 'babies',
                                          'previous_cancellations', 'booking_changes',
'days_in_waiting_list', 'adr',
                                          'required_car_parking_spaces','total_of_speci
al_requests', 'country',
                                          'assigned_room_type','agent', 'reservation_st
atus', 'reservation_status_date']).copy()
```

In [85]:

```
v = OHE_city_hotel_bookings['is_canceled']
```

In [86]:

```
U_train, U_test, v_train, v_test = train_test_split(U, v, test_size=0.33, random_state=
324)
```

In [87]:

```
U_train.shape[0]
```

Out[87]:

```
47702
```

In [88]:

```
U_test.shape[0]
```

Out[88]:

```
23496
```

In [89]:

```
canceled_cat_classifier = DecisionTreeClassifier(max_leaf_nodes=10, random_state=0)
canceled_cat_classifier.fit(U_train, v_train)
```

Out[89]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=10,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [90]:

```
predictions = canceled_cat_classifier.predict(U_test)
```

In [91]:

```
predictions
```

Out[91]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [92]:

```
v_test[:10]
```

Out[92]:

```
7577     1
69297    0
6348     0
3999     1
17468    1
13757    0
33014    1
53349    0
45229    0
51637    0
Name: is_canceled, dtype: int64
```

In [93]:

```
print(classification_report(v_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.68      1.00      0.81     13360
           1       0.99      0.39      0.56     10136

    accuracy                           0.74     23496
   macro avg       0.84      0.69      0.69     23496
weighted avg       0.82      0.74      0.70     23496
```

In [94]:

```
print(confusion_matrix(v_test, predictions))
```

```
[[13322    38]
 [ 6185  3951]]
```

In [95]:

```
accuracy_score(y_true = v_test, y_pred = predictions)
# it shows that excluding numerical features does reduce the model accuracy,
# however, only with categorical features, the model predicts 73% correctly,
# which confirms that categorical features play more important role
# this model also has very less false positives and very high false negatives,
```

Out[95]:

```
0.7351464078992169
```

In [96]:

```
# sensitivity or true recall rate: has reduced compared to using both numeric and categ
orical features
# the classifier incorrectly predicts no cancellations more number of times, compared t
o previous model
13322 / (13322 + 6185) * 100
```

Out[96]:

```
68.29343312656995
```

In [97]:

```
# precision (showing classifier improves in correctly classifying positives)
13322 / (13322 + 38) * 100
```

Out[97]:

```
99.71556886227545
```

In [98]:

```
#specificity (showing again that classifier improves in correctly classifying negative
s)
3951 / (3951 + 38)
```

Out[98]:

```
0.9904738029581349
```

In [ ]:

```
# thus, the classifier improves on precision and specificity, however overall accuracy
 is reduced
```

Method: LinearRegression

In [99]:

```
# Linear Regression
canceled_LR_regressor = LinearRegression()
canceled_LR_regressor.fit(X_train, y_train)
```

Out[99]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [100]:

```
y_prediction = canceled_LR_regressor.predict(X_test)
y_prediction[:10]
```

Out[100]:

```
array([0.48393622, 0.25384015, 0.40764482, 0.11247647, 1.04280734,
       0.60046294, 0.54397789, 0.26796192, 0.25183568, 0.25340713])
```

In [101]:

```
y_test[:10]
```

Out[101]:

```
7577     1
69297    0
6348     0
3999     1
17468    1
13757    0
33014    1
53349    0
45229    0
51637    0
Name: is_canceled, dtype: int64
```

In [102]:

```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
RMSE
```

Out[102]:

```
0.39433607489549755
```

In [104]:

```
from sklearn.metrics import explained_variance_score
explained_variance_score(y_test, y_prediction)
```

Out[104]:

```
0.3661623525987189
```

Method: DecisionTreeRegressor

In [105]:

```
# Regression using Decision Tree
canceled_DT_regressor = DecisionTreeRegressor(max_depth=20)
canceled_DT_regressor.fit(X_train, y_train)
```

Out[105]:

```
DecisionTreeRegressor(criterion='mse', max_depth=20, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

In [106]:

```
prediction = canceled_DT_regressor.predict(X_test)
prediction[:20]
```

Out[106]:

```
array([0.6776699 , 1.        , 0.07894737, 1.        , 1.        ,
       0.7752809 , 1.        , 0.19492294, 0.01549587, 0.19492294,
       0.6776699 , 1.        , 0.57407407, 0.01549587, 0.34615385,
       1.        , 0.05590062, 0.3902439 , 0.01549587, 0.41666667])
```

In [107]:

```
y_test[:10]
```

Out[107]:

```
7577     1
69297    0
6348     0
3999     1
17468    1
13757    0
33014    1
53349    0
45229    0
51637    0
Name: is_canceled, dtype: int64
```

In [108]:

```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = prediction))
RMSE
```

Out[108]:

```
0.379002390858811
```

In [109]:

```
explained_variance_score(y_test, prediction)
# not good accuracy, need to improve the model, best score could be 1.0
```

Out[109]:

```
0.4144283994890825
```

Method: LogisticRegression

In [110]:

```
canceled_Logit_regressor = LogisticRegression(random_state = 0, max_iter=1000)
canceled_Logit_regressor.fit(X_train, y_train)
```

```
C:\Users\vaibh\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
\linear_model\logistic.py:432: FutureWarning: Default solver will be chang
ed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[110]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [111]:

```
y_pred = canceled_Logit_regressor.predict(X_test)
```

In [114]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.91      0.84     13360
           1       0.84      0.65      0.73     10136

    accuracy                           0.80     23496
   macro avg       0.81      0.78      0.79     23496
weighted avg       0.80      0.80      0.79     23496
```

In [112]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[12152  1208]
 [ 3554  6582]]
```

In [113]:

```
accuracy_score(y_true = y_test, y_pred = y_pred)
```

Out[113]:

```
0.7973272046305754
```

In [115]:

```
# sensitivity
12152 / (12152 + 3554) * 100
```

Out[115]:

```
77.37170508086082
```

In [116]:

```
# precision
12152 / (12152 + 1208) * 100
```

Out[116]:

```
90.95808383233533
```

In [119]:

```
6582 / (6582 + 1208) * 100
```

Out[119]:

```
84.49293966623877
```