Project Phase II: Data Preparation, Cleaning and Feature Engineering

Farida Simaika - 900201753
Katia Gabriel - 900202272

In [121…
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from scipy.stats import chi2_contingency
import plotly.figure_factory as ff
import seaborn as sns
from scipy.stats import f_oneway
import matplotlib.pyplot as plt
```

Reading the data set

In [121…
```
df=pd.read_csv("hotel_bookings.csv")
```

Dividing the data set into numerical and categorical to facilitate primary analysis

In [121…
```
categ = df.loc[:, (df.dtypes != int) & (df.dtypes != float)].columns.tolist(
num=df.loc[:, (df.dtypes == int) | (df.dtypes == float)].columns.tolist()
```

In [121…
```
df[num].describe()
```

Out[1213]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arriva
count	119390.000000	119390.000000	119390.000000	119390.000000	
mean	0.370416	104.011416	2016.156554	27.165173	
std	0.482918	106.863097	0.707476	13.605138	
min	0.000000	0.000000	2015.000000	1.000000	
25%	0.000000	18.000000	2016.000000	16.000000	
50%	0.000000	69.000000	2016.000000	28.000000	
75%	1.000000	160.000000	2017.000000	38.000000	
max	1.000000	737.000000	2017.000000	53.000000	

In [121…
```
df[categ].describe()
```

Out[1214]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_chann
count	119390	119390	119390	118902	119390	11939
unique	2	12	5	177	8	
top	City Hotel	August	BB	PRT	Online TA	TA/T
freq	79330	13877	92310	48590	56477	978

In [121…

```
#unique values of categorical variables and their counts

for col in categ:
    print("----------------------")
    print(df[col].value_counts())
```

```
----------------------
City Hotel      79330
Resort Hotel    40060
Name: hotel, dtype: int64
----------------------
August       13877
July         12661
May          11791
October      11160
April        11089
June         10939
September    10508
March         9794
February      8068
November      6794
December      6780
January       5929
Name: arrival_date_month, dtype: int64
----------------------
BB           92310
HB           14463
SC           10650
Undefined     1169
FB             798
Name: meal, dtype: int64
----------------------
PRT    48590
GBR    12129
FRA    10415
ESP     8568
DEU     7287
       ...
DJI        1
BWA        1
HND        1
VGB        1
NAM        1
Name: country, Length: 177, dtype: int64
----------------------
Online TA        56477
Offline TA/TO    24219
Groups           19811
Direct           12606
Corporate         5295
Complementary      743
Aviation           237
Undefined            2
Name: market_segment, dtype: int64
----------------------
TA/TO        97870
Direct       14645
Corporate     6677
GDS            193
Undefined        5
Name: distribution_channel, dtype: int64
----------------------
A    85994
D    19201
E     6535
F     2897
G     2094
B     1118
C      932
H      601
```

```
P       12
L        6
Name: reserved_room_type, dtype: int64
----------------------
A    74053
D    25322
E     7806
F     3751
G     2553
C     2375
B     2163
H      712
I      363
K      279
P       12
L        1
Name: assigned_room_type, dtype: int64
----------------------
No Deposit    104641
Non Refund     14587
Refundable       162
Name: deposit_type, dtype: int64
----------------------
Transient          89613
Transient-Party    25124
Contract            4076
Group                577
Name: customer_type, dtype: int64
----------------------
Check-Out    75166
Canceled     43017
No-Show       1207
Name: reservation_status, dtype: int64
----------------------
2015-10-21    1461
2015-07-06     805
2016-11-25     790
2015-01-01     763
2016-01-18     625
              ...
2015-02-27       1
2015-04-25       1
2015-03-11       1
2015-06-14       1
2015-02-12       1
Name: reservation_status_date, Length: 926, dtype: int64
```

Checking for NAs

In [121…

```
df.isnull().sum()
```

Out[1216]:

```
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                           488
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                           16340
company                        112593
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 [121…

```
#dropping 4 NAs of variable children
df.dropna(subset=["children"],inplace=True)
```

94% of the column company is missing, it is obvious that it has to be removed

In [121…

```
print(len(df[(df['stays_in_weekend_nights']==0) & (df['stays_in_week_nights

df.drop(df[(df['stays_in_weekend_nights']==0) & (df['stays_in_week_nights']=
```

715

715 bookings don't have both weekday or weekend nights which is not possible so dropped

Adjusting Data Types

In [121…

```
df['children'] = df['children'].astype(int)

df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date']
```

Feature Engineering

In [122…
```
df['kids'] = df['children'] + df['babies'] #most booking websites do not di
df["total_number_of_stays"]=df["stays_in_weekend_nights"]+df["stays_in_week_
df["total_guests"]=df["adults"]+df["children"]+df["babies"]
df["total_bookings"]=df["previous_cancellations"]+df["previous_bookings_not_
```

Stays in weekends and weeknights Feature Analysis

In [122…
```
df.corr(method='pearson')["stays_in_weekend_nights"]["is_canceled"]
```

Out[1221]:
```
-0.0055337983815443815
```

In [122…
```
df.corr(method='pearson')["stays_in_week_nights"]["is_canceled"]
```

Out[1222]:
```
0.019656436530706718
```

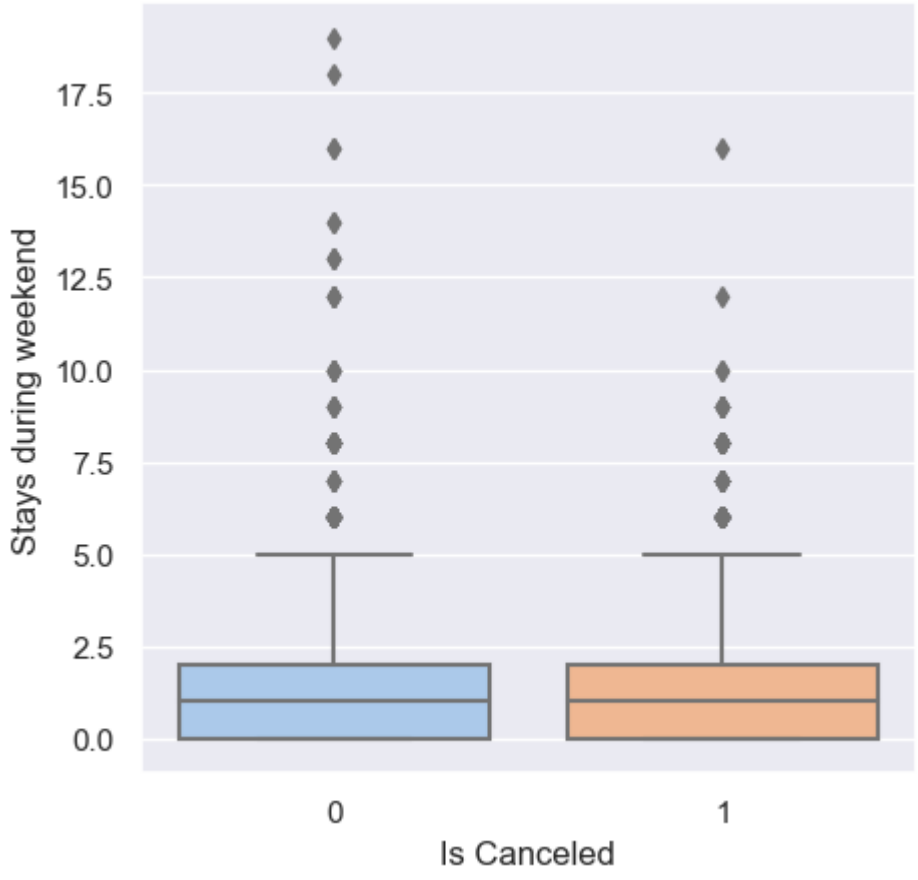
In [122…
```
df.corr(method='pearson')["stays_in_week_nights"]["stays_in_weekend_nights"]
```

Out[1223]:
```
0.49548639831576335
```

Low correlation with label and collinearity problem between the variables

In [122…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x="is_canceled", y="stays_in_weekend_nights",data=df, palett
plt.ylabel('Stays during weekend', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
```

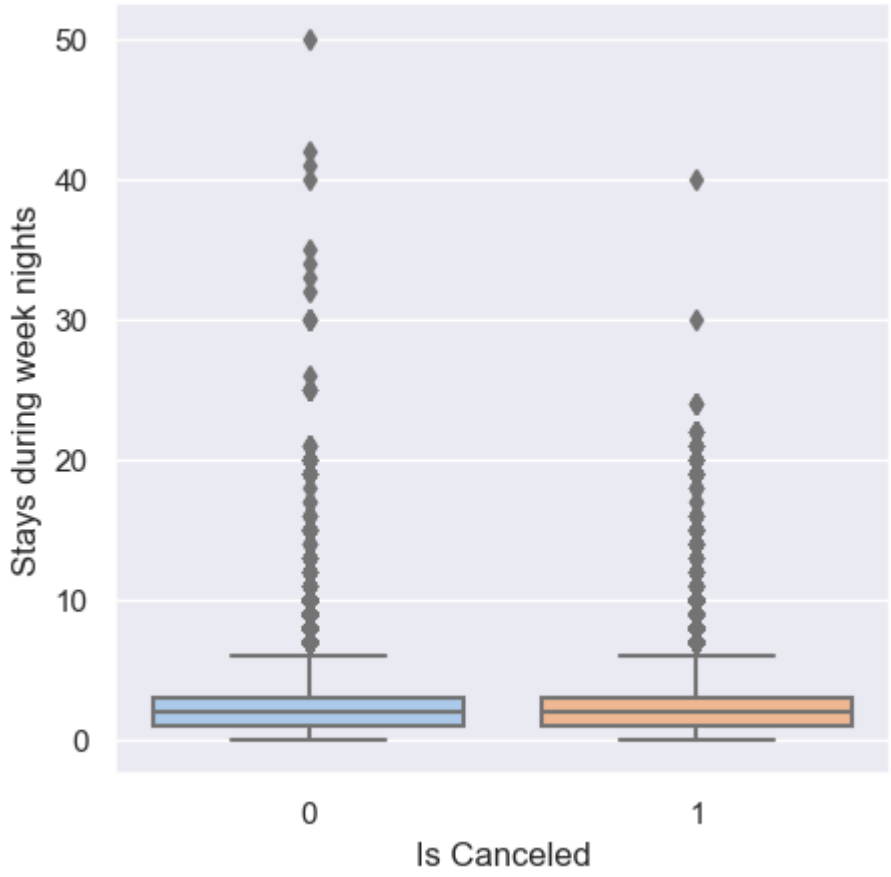
Out[1224]:
```
Text(0.5, 0, 'Is Canceled')
```



In [122…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x="is_canceled", y="stays_in_week_nights",data=df, palette=
```

```
plt.ylabel('Stays during week nights', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
```

Out[1225]:
```
Text(0.5, 0, 'Is Canceled')
```



According to the above boxplots, the number of stay during weekends and weekdays has nothing to do with a booking cancellation. The two boxplots have a very similar distribution regardless of whether or not the booking was canceled. These columns can be eliminated. A new feature which is the sum of the two variables will be added to the data set.

In [122…
```
df["total_number_of_stays"]=df["stays_in_weekend_nights"]+df["stays_in_week_
```

In [122…
```
df=df.drop(["stays_in_weekend_nights"],axis=1)
df=df.drop(["stays_in_week_nights"],axis=1)
```

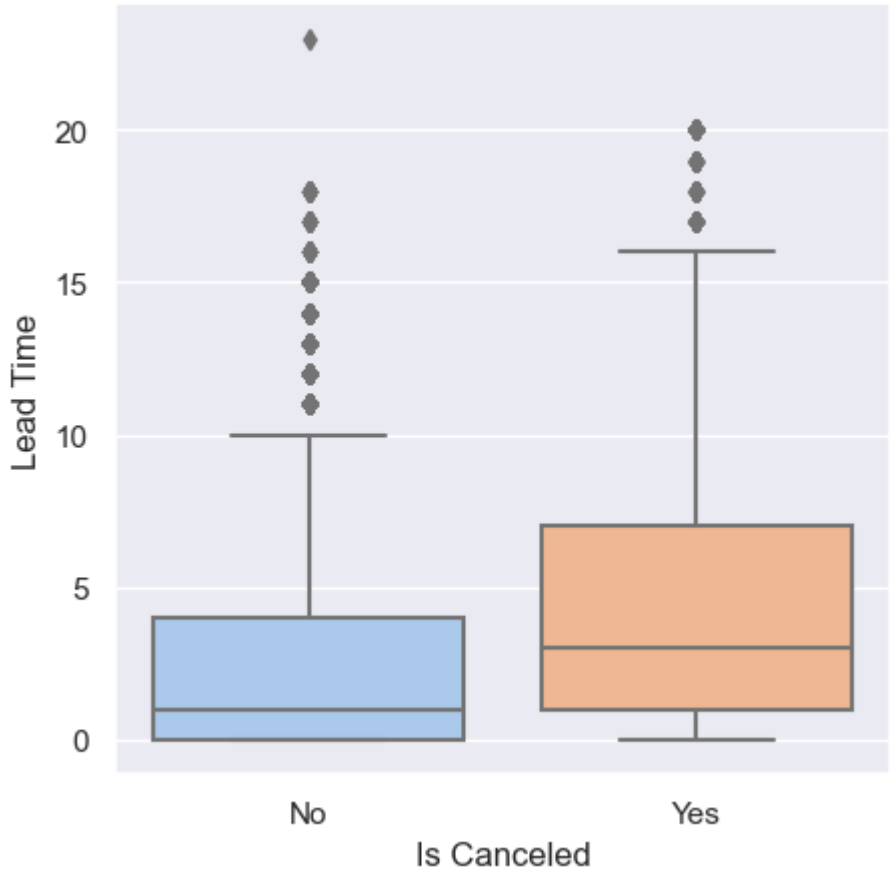
Lead time Feature Analysis

In [122…
```
df.corr(method='pearson')['is_canceled']["lead_time"]
```

Out[1228]:
```
0.29172405336589574
```

In [122…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x='is_canceled', y='lead_time_month', data=df_lt, palette='
plt.ylabel('Lead Time', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
fig.set_xticklabels(['No', 'Yes'])
```

Out[1229]:
```
[Text(0, 0, 'No'), Text(1, 0, 'Yes')]
```



In [123…

```
df_lead_time = df.copy()
df_lead_time['lead_time_month'] = df_lead_time['lead_time'] // 30 #convertir
df_lead_time = df_lead_time.groupby(['lead_time_month','is_canceled']).agg(
df_lead_time = df_lead_time.rename(columns={'hotel':'total'})
df_lead_time['total_guests'] = df_lead_time.groupby(['lead_time_month'])['to
df_lead_time['cancel_percentage'] = (df_lead_time['total'] / df_lead_time['t
df_lead_time = df_lead_time.sort_values('cancel_percentage', ascending=False
df_lead_time.pivot_table(index='lead_time_month', columns="is_canceled", val
```

Out[1230]:

is_canceled	0	1
lead_time_month		
0	81.573543	18.426457
1	63.533944	36.466056
2	60.179641	39.820359
3	55.886009	44.113991
4	56.363185	43.636815
5	53.739086	46.260914
6	55.203778	44.796222
7	53.003210	46.996790
8	44.806517	55.193483
9	36.065097	63.934903
10	30.626366	69.373634
11	29.362416	70.637584
12	42.008197	57.991803
13	37.500000	62.500000
14	27.255639	72.744361
15	35.159011	64.840989
16	17.073171	82.926829
17	18.852459	81.147541
18	25.274725	74.725275
19	NaN	100.000000
20	NaN	100.000000
23	100.000000	NaN

The longer the lead time, the more likely the cancellation of the booking. Bookings with a lead time of more than 7 months have more than 50% chance of cancellation.

Hotel Feature Analysis

In [123…

```
df["hotel"].value_counts()
```

Out[1231]:

```
City Hotel      78995
Resort Hotel    39676
Name: hotel, dtype: int64
```

In [123…

```
fig=plotly.histogram(df, x="hotel", color="is_canceled",barmode="group",colo
fig.update_layout(
    title_text='Cancellations based on hotel types', # title of plot
    title_x=0.5,
    xaxis_title_text='Hotel Type', # xaxis label
    yaxis_title_text='Count', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
```

```
fig.show()
```

Cancellations are more frequent in City Hotels. City Hotels also receive more booking than resort hotels. This is why the cancellation percentages should be calculated.

In [123…

```
fig=plotly.histogram(df, x="hotel", color="hotel", barmode="group",color_dis
fig.update_layout(
    title_text='Number of Bookings by Hotel Types', # title of plot
    title_x=0.5,
    xaxis_title_text='Hotel Type', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)

fig.show()
```

As expected,City Hotels receive the majority of the bookings. This explains the higher number of cancellations

Label Encoding (one-hot encoding) for variable hotel

In [123…
```
class_names = df["hotel"].unique()
class_names
```

Out[1234]:
```
array(['Resort Hotel', 'City Hotel'], dtype=object)
```

In [123…
```
df.loc[ df["hotel"] == class_names[0],"hotel"]=0 #resort =0
df
```

Out[1235]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date
2	0	0	7	2015	July	
3	0	0	13	2015	July	
4	0	0	14	2015	July	
5	0	0	14	2015	July	
6	0	0	0	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	

118671 rows × 34 columns

In [123…
```
df.loc[ df["hotel"] == class_names[1],"hotel"]=1 #city =1
df
```

Out[1236]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date
2	0	0	7	2015	July	
3	0	0	13	2015	July	
4	0	0	14	2015	July	
5	0	0	14	2015	July	
6	0	0	0	2015	July	
...	...	...	...	...	...	
119385	1	0	23	2017	August	
119386	1	0	102	2017	August	
119387	1	0	34	2017	August	
119388	1	0	109	2017	August	
119389	1	0	205	2017	August	

118671 rows × 34 columns

In [123…
```
df["hotel"].isnull().sum()
```

Out[1237]:
```
0
```

Cancellation percentages by Hotel Type

In [123…
```
cancel_year = df.groupby(['hotel', 'is_canceled']).size().to_frame(name='cou
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
```

```
print(cancel_percentage)
```

```
                        count
hotel is_canceled
0     0            71.998185
      1            28.001815
1     0            58.130261
      1            41.869739
```

Chi-Square Test

H0: no relationship between the variables

H1:there is a relationship between the variables

In [123…
```
chi_square_args = pd.crosstab(df['hotel'], df['is_canceled']).values

_, p_value, _, _ = scipy.stats.chi2_contingency(chi_square_args)
print("The p-value is:",p_value)

if p_values<0.05:
    print("We reject null hypothesis, there is a significant relationship be
else:
    print("accept null hypothesis")
```

```
The p-value is: 0.0
We reject null hypothesis, there is a significant relationship between meal
type and booking cancellation
```

At significance level of 5%, we reject the null hypothesis

Arrival Years and Months

In [124…
```
df["arrival_date_year"].value_counts()
```

Out[1240]:
```
2016    56290
2017    40543
2015    21838
Name: arrival_date_year, dtype: int64
```

In [124…
```
df["arrival_date_month"].value_counts()
```

Out[1241]:
```
August       13822
July         12601
May          11712
October      11074
April        11051
June         10897
September    10485
March         9728
February      8007
November      6723
December      6695
January       5876
Name: arrival_date_month, dtype: int64
```

In [124…
```
fig=plotly.histogram(df, x="arrival_date_year", color="is_canceled",barmode=
fig

#cancellations more frequent in city hotels
fig=plotly.histogram(df, x="arrival_date_year", color="is_canceled",barmode=
fig.update_layout(
    title_text='Cancellations based on Year of Booking', # title of plot
```

```
    title_x=0.5,
    xaxis_title_text='Arrival Year', # xaxis label
    yaxis_title_text='Count', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)

fig.show()
```

The year 2016 appears to be the year with the most cancellations. However, it is also the year with the most bookings. This why cancellation percentages should be calculated.

In [124…

```
fig=plotly.histogram(df, x="arrival_date_month", color="is_canceled",barmode
fig.update_layout(
    title_text='Cancellations based on Month of Booking', # title of plot
    title_x=0.5,
    xaxis_title_text='Arrival Month', # xaxis label
    yaxis_title_text='Count', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)

fig.show()
```

July and August appear to be the months with the most cancellations. They are also the months with the most bookings. Cancellation percentages should be calculated.

Cancellation Percentages by year and month

In [124…
```
cancel_year = df.groupby(['arrival_date_year', 'is_canceled']).size().to_fra
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                                  count
arrival_date_year is_canceled
2015              0            62.739262
                  1            37.260738
2016              0            63.911885
                  1            36.088115
2017              0            61.191821
                  1            38.808179
```

In [124…
```
cancel_month = df.groupby(['arrival_date_month', 'is_canceled']).size().to_
cancel_percentage = cancel_month.groupby(level=0).apply(lambda x:100 * x / 
print(cancel_percentage)
```

```
                                          count
arrival_date_month is_canceled
April              0            59.134920
                   1            40.865080
August             0            62.140067
                   1            37.859933
December           0            64.645258
                   1            35.354742
February           0            66.429374
                   1            33.570626
January            0            69.298843
                   1            30.701157
July               0            62.376002
                   1            37.623998
June               0            58.401395
                   1            41.598605
March              0            67.650082
                   1            32.349918
May                0            60.083675
                   1            39.916325
November           0            68.451584
                   1            31.548416
October            0            61.666968
                   1            38.333032
September          0            60.753457
                   1            39.246543
```

Primary assumotion was erroneous: 2017 is the year with the highest cancellation rates
The months of April and June have the highest cancellation rates as well.

Encoding the months

In [124…
```
class_names=df["arrival_date_month"].unique()
class_names
```

Out[1246]:
```
array(['July', 'August', 'September', 'October', 'November', 'December',
       'January', 'February', 'March', 'April', 'May', 'June'],
      dtype=object)
```

In [124…
```
df.loc[ df["arrival_date_month"] == class_names[0],"arrival_date_month"]=7
df.loc[ df["arrival_date_month"] == class_names[1],"arrival_date_month"]=8
df.loc[ df["arrival_date_month"] == class_names[2],"arrival_date_month"]=9
df.loc[ df["arrival_date_month"] == class_names[3],"arrival_date_month"]=10
df.loc[ df["arrival_date_month"] == class_names[4],"arrival_date_month"]=11
df.loc[ df["arrival_date_month"] == class_names[5],"arrival_date_month"]=12
df.loc[ df["arrival_date_month"] == class_names[6],"arrival_date_month"]=1
df.loc[ df["arrival_date_month"] == class_names[7],"arrival_date_month"]=2
df.loc[ df["arrival_date_month"] == class_names[8],"arrival_date_month"]=3
df.loc[ df["arrival_date_month"] == class_names[9],"arrival_date_month"]=4
df.loc[ df["arrival_date_month"] == class_names[10],"arrival_date_month"]=5
df.loc[ df["arrival_date_month"] == class_names[11],"arrival_date_month"]=6
```

In [124…
```
df["arrival_date_month"] = df["arrival_date_month"].astype(int)
```

Arrival Week and Day Feature Analysis

In [124…
```
df.corr(method='pearson')['is_canceled']["arrival_date_day_of_month"]
```

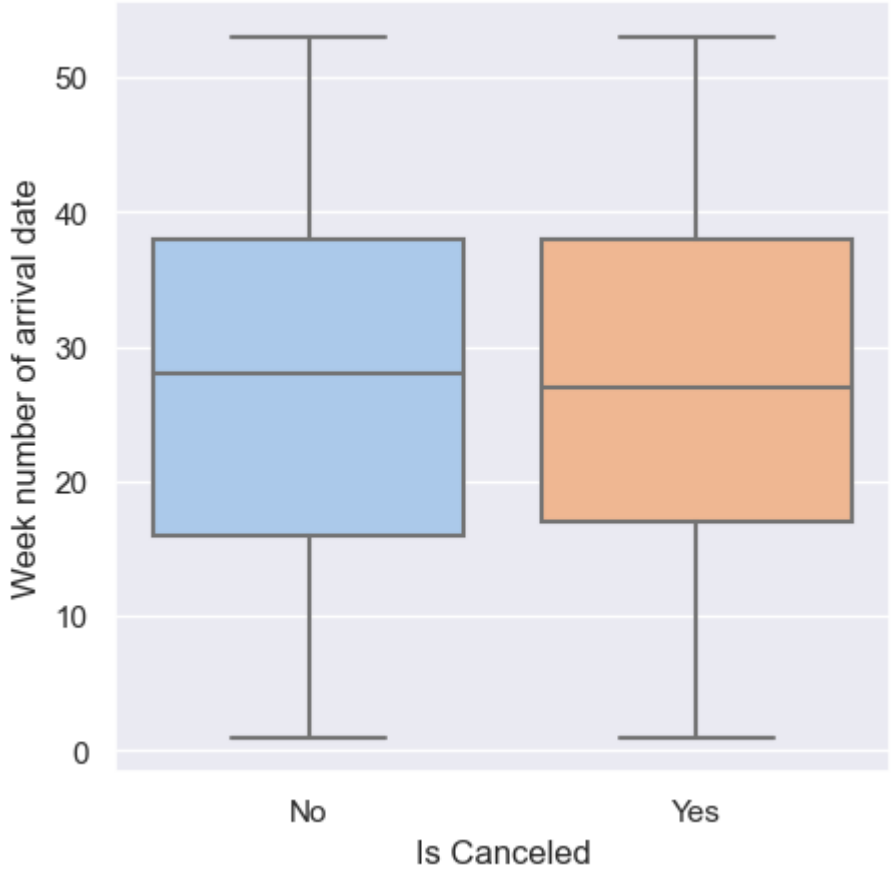
Out[1249]:
```
-0.006157910570479181
```

In [125…
```
df.corr(method='pearson')['is_canceled']["arrival_date_week_number"]
```

Out[1250]:
```
0.008876331356995967
```

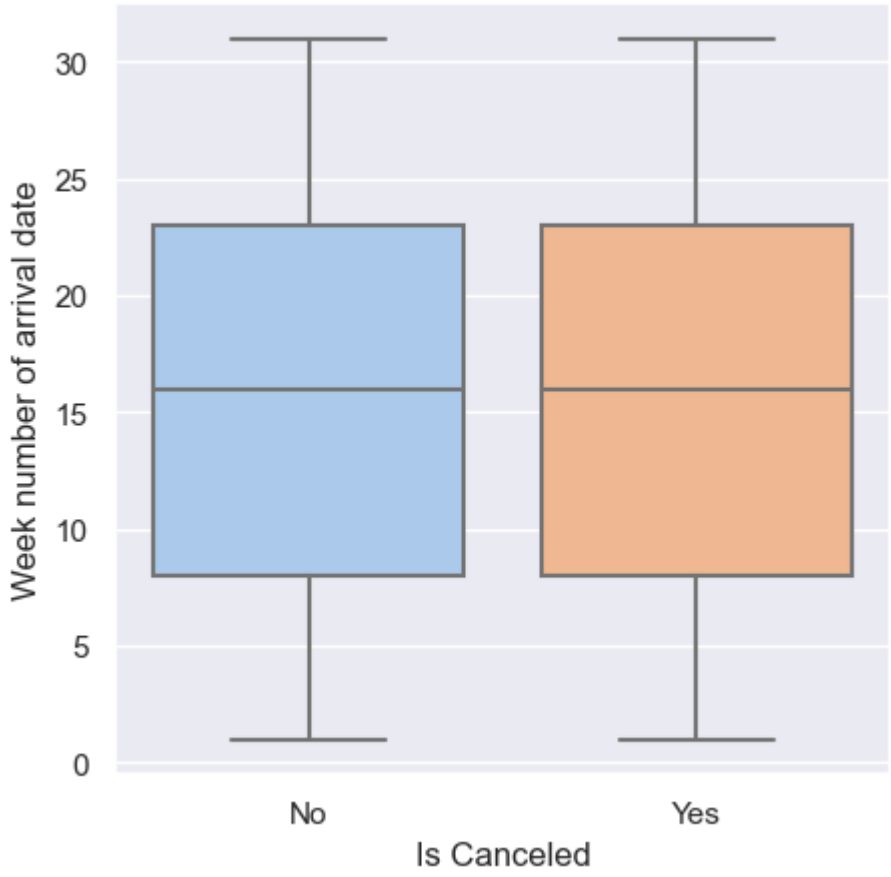
In [125…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x='is_canceled', y='arrival_date_week_number', data=df_lt, p
plt.ylabel('Week number of arrival date', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
fig.set_xticklabels(['No', 'Yes'])
```

Out[1251]:
```
[Text(0, 0, 'No'), Text(1, 0, 'Yes')]
```



In [125…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x='is_canceled', y='arrival_date_day_of_month', data=df_lt,
plt.ylabel('Week number of arrival date', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
fig.set_xticklabels(['No', 'Yes'])
```

Out[1252]:
```
[Text(0, 0, 'No'), Text(1, 0, 'Yes')]
```



According to the above plots, the week number and the day of the month of arrival date do not play a role in determining a cancellation. The two boxplots have a very similar distribution regardless of whether or not the booking was canceled. These columns can be eliminated. We can infer that the arrival year and month will be enough to predict booking cancellation.

In [125…
```
df=df.drop(["arrival_date_day_of_month"],axis=1)
```

In [125…
```
df=df.drop(["arrival_date_week_number"],axis=1)
```

Adults Feature Analysis

In [125…
```
df["adults"].value_counts()
```

Out[1255]:
```
2     89250
1     22825
3      6185
0       333
4        62
26        5
27        2
20        2
5         2
40        1
50        1
55        1
6         1
10        1
Name: adults, dtype: int64
```

Observation with adults <1 should be dropped, a minor cannot book room alone without an adult.

In [125…
```
df.drop(df[df["adults"]==0].index,inplace=True)
```

In [125…
```
df.corr(method='pearson')['is_canceled']["adults"]
```

Out[1257]:
```
0.058229923116148455
```

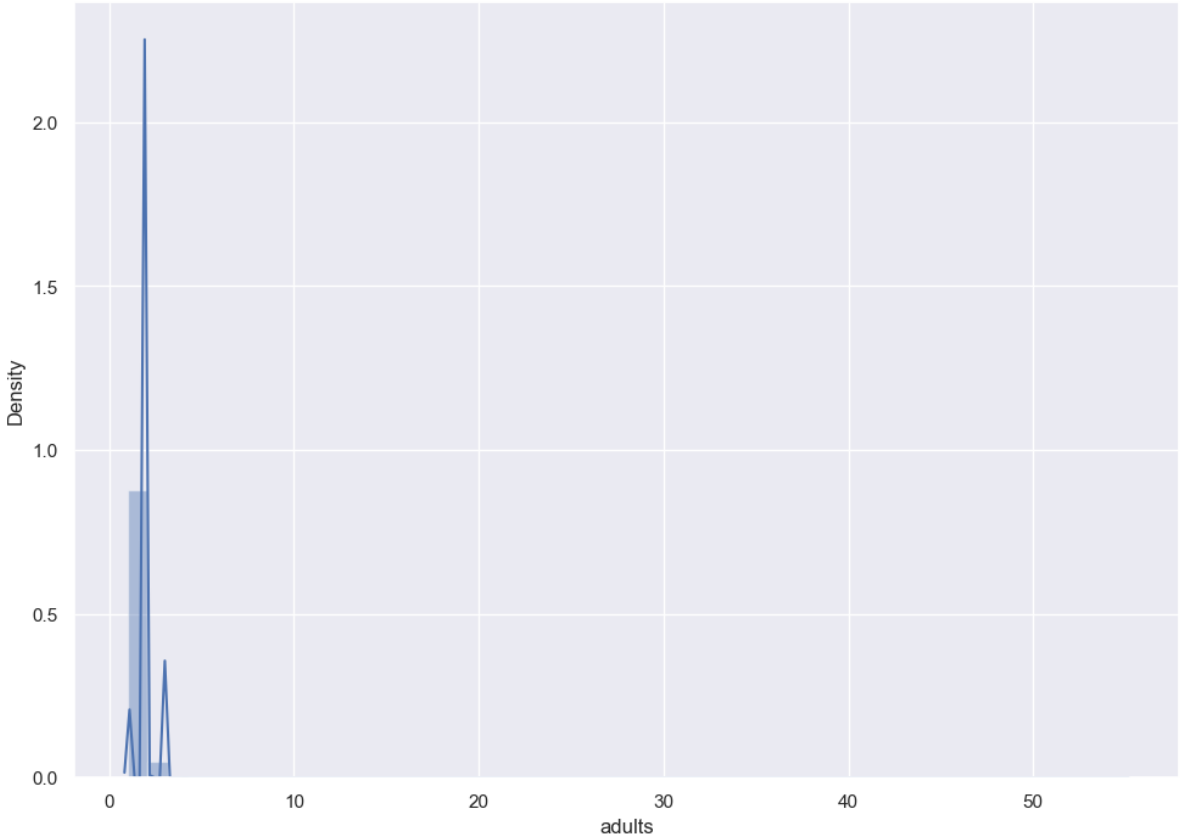
In [125…
```
cancel_year = df.groupby(['adults', 'is_canceled']).size().to_frame(name='c
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                          count
adults is_canceled
1      0              70.808324
       1              29.191676
2      0              60.512045
       1              39.487955
3      0              65.238480
       1              34.761520
4      0              74.193548
       1              25.806452
5      1             100.000000
6      1             100.000000
10     1             100.000000
20     1             100.000000
26     1             100.000000
27     1             100.000000
40     1             100.000000
50     1             100.000000
55     1             100.000000
```

The number of adults does not seem to impact the cancellation of the booking.
Bookings with many adults are as likely to be canceled as reservations with only 1 adult.

In [125…
```
sns.distplot(df["adults"])
```

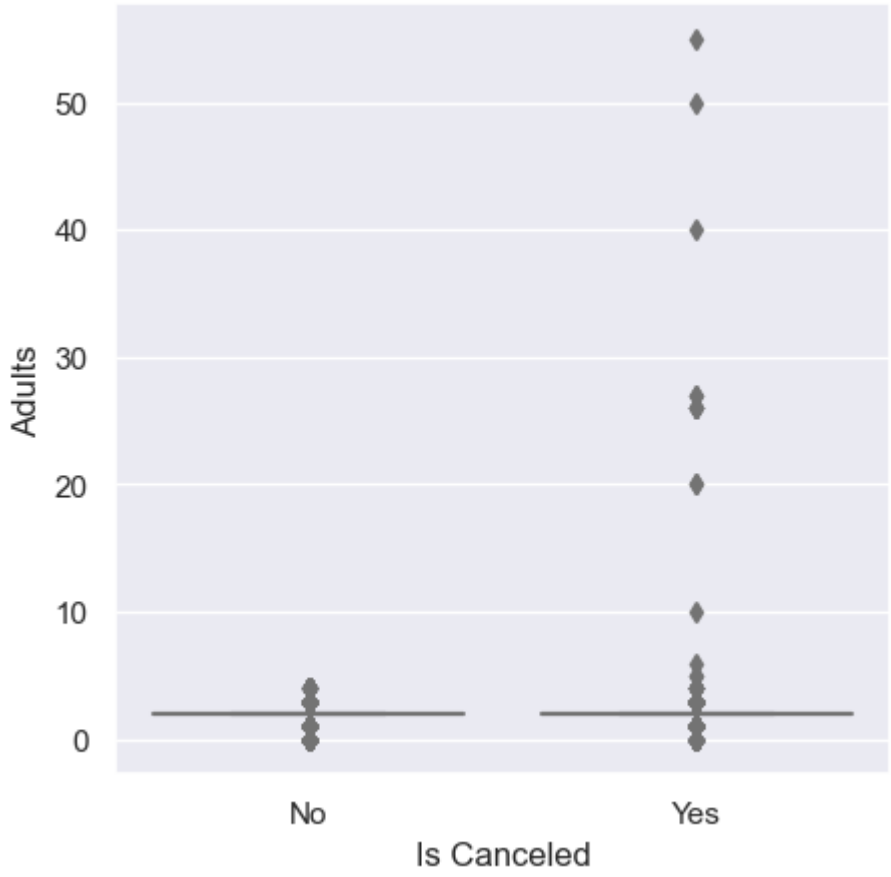
Out[1259]:
```
<AxesSubplot:xlabel='adults', ylabel='Density'>
```



The distribution is skewed. Most of the bookings have between 1 or 2 adults.

In [126…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x='is_canceled', y='adults', data=df_lt, palette='pastel')
plt.ylabel('Adults', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
fig.set_xticklabels(['No', 'Yes'])
```

Out[1260]:
```
[Text(0, 0, 'No'), Text(1, 0, 'Yes')]
```



According to the above plots, the number of adults does not determine a cancellation. The two boxplots have a very similar distribution regardless of whether or not the booking was canceled.

Children and Babies Feature Analysis

In [126…
```
df["children"].value_counts()
```

Out[1261]:
```
0     110005
1       4838
2       3429
3         65
10         1
Name: children, dtype: int64
```

In [126…
```
df["children"].isnull().sum()
```

Out[1262]:
```
0
```

In [126…
```
df.corr(method='pearson')['is_canceled']["children"]
```

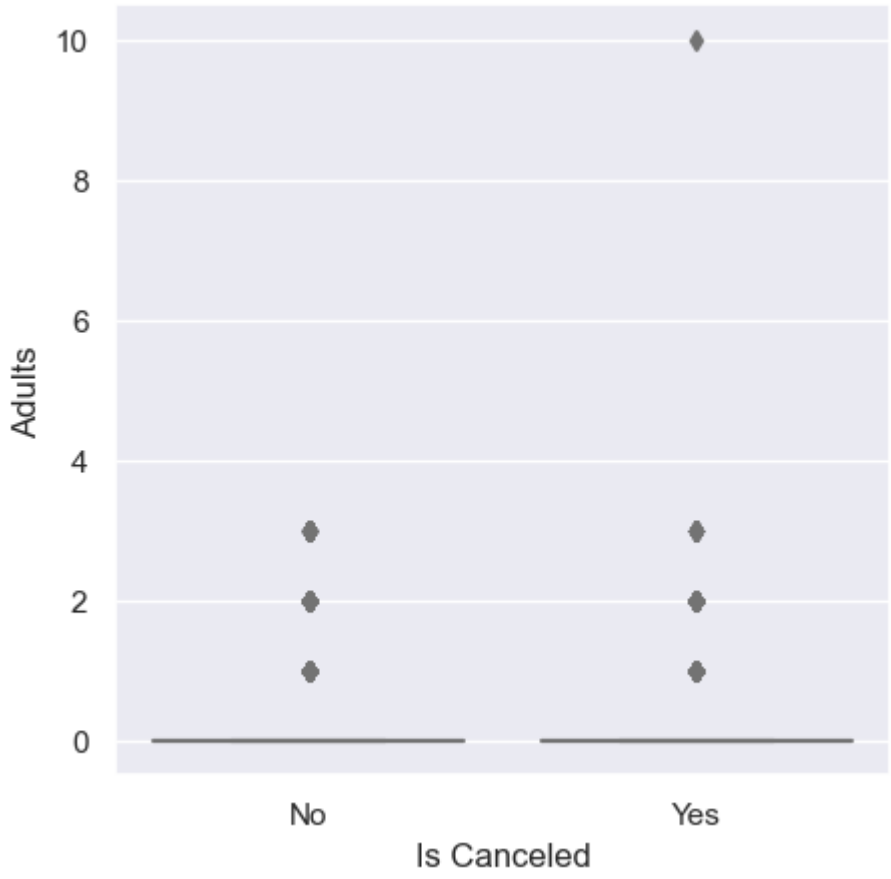
Out[1263]:
```
0.004707351352995382
```

In [126…
```
df["children"].groupby(df["is_canceled"]).value_counts()
```

Out[1264]:
```
is_canceled  children
0            0           68959
             1            3271
             2            1969
             3              51
1            0           41046
             1            1567
             2            1460
             3              14
             10              1
Name: children, dtype: int64
```

In [126…
```
plt.figure(figsize=(5, 5))
fig=sns.boxplot(x='is_canceled', y='children', data=df_lt, palette='pastel')
plt.ylabel('Adults', fontsize=12)
plt.xlabel('Is Canceled', fontsize=12)
fig.set_xticklabels(['No', 'Yes'])
```

Out[1265]:
```
[Text(0, 0, 'No'), Text(1, 0, 'Yes')]
```



According to the above plots, the number of children does not play a role in determining a cancellation. The two boxplots have a very similar distribution regardless of whether or not the booking was canceled. These columns can be eliminated and will be combined with the babies feature under one variable called kids.

In [126…
```
df=df.drop(["children"],axis=1)
df=df.drop(["babies"],axis=1)
```

Meal Feature Analysis

In [126…
```
df["meal"].value_counts()
```

Out[1267]:
```
BB           91501
HB           14380
SC           10500
Undefined     1160
FB             797
Name: meal, dtype: int64
```

According to data source Undefined meal type is the same as SC

In [126…
```
df=df.replace("Undefined","SC")
```

Chi-Square Test

In [126…
```
#H0: no relationship between the variables
#H1:there is a relationship between the variables

chi_square_args = pd.crosstab(df['meal'], df['is_canceled']).values

_, p_value, _, _ = scipy.stats.chi2_contingency(chi_square_args)
print("The p-value is:",p_value)

if p_values<0.05:
    print("We reject null hypothesis, there is a significant relationship be
else:
    print("accept null hypothesis")
```

```
The p-value is: 6.1984337227451404e-49
We reject null hypothesis, there is a significant relationship between meal
s and booking cancellation
```

In [127…
```
fig=plotly.histogram(df, x="meal", color="is_canceled", barmode="group",colo
fig.update_layout(
    title_text='Booking by Meals Types', # title of plot
    title_x=0.5,
    xaxis_title_text='Meal Type', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
fig.show()
```

The most frequent meal type is the BB. It also has the highest cancellations numbers.

Meal types and Cancellation percentages

In [127…
```
cancel_year = df.groupby(['meal', 'is_canceled']).size().to_frame(name='cou
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                     count
meal is_canceled
BB   0           62.406968
     1           37.593032
FB   0           40.025094
     1           59.974906
HB   0           65.347705
     1           34.652295
SC   0           63.730703
     1           36.269297
```

Hotel booking with meal of type FB have the highest cancellation rate compared to other meal types.

In [127…
```
fig=plotly.histogram(df, x="meal", color="deposit_type", barmode="stack",co
fig.update_layout(
    title_text='Booking by Meal and Deposit Types', # title of plot
    title_x=0.5,
    xaxis_title_text='Meal Type', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
```

```
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
fig.show()
```

Meal type and Deposit cancellation percentages

In [127…
```
df.columns
```

Out[1273]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'adults', 'meal', 'country', 'market_segmen
t',
       'distribution_channel', 'is_repeated_guest', 'previous_cancellatio
ns',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'kids',
       'total_number_of_stays', 'total_guests', 'total_bookings'],
      dtype='object')
```

In [127…
```
cancel_year = df.groupby(['meal', 'deposit_type']).size().to_frame(name='cou
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                          count
meal deposit_type
BB   No Deposit    86.242773
     Non Refund    13.587830
     Refundable     0.169397
FB   No Deposit    63.111669
     Non Refund    36.888331
HB   No Deposit    88.115438
     Non Refund    11.863700
     Refundable     0.020862
SC   No Deposit    98.644940
     Non Refund     1.320755
     Refundable     0.034305
```

FB has the highest non-refund deposit ratio (usually associated with higher cancellation).

Encoding the meal categories

In [127…
```
df["BB"] = df['meal'].apply(lambda x: 1 if x == 'BB' else 0) #the hotels in

df["FB"] = df['meal'].apply(lambda x: 1 if x == 'FB' else 0)

df["HB"] = df['meal'].apply(lambda x: 1 if x == 'HB' else 0)

df["SC"] = df['meal'].apply(lambda x: 1 if x == 'SC' else 0)
```

In [127…
```
df=df.drop(["meal"],axis=1)
```

Distribution Channel Feature Analysis

In [127…
```
df["distribution_channel"].value_counts()
```

Out[1277]:
```
TA/TO        97145
Direct       14418
Corporate     6584
GDS            190
SC               1
Name: distribution_channel, dtype: int64
```

Chi-Square Test

H0: no relationship between the variables
H1:there is a relationship between the variables

In [127…
```
chi_square_args = pd.crosstab(df['distribution_channel'], df['is_canceled'])

_, p_value, _, _ = scipy.stats.chi2_contingency(chi_square_args)
print("The p-value is:",p_value)

if p_values<0.05:
    print("We reject null hypothesis, there is a significant relationship be
else:
    print("accept null hypothesis")
```

```
The p-value is: 0.0
We reject null hypothesis, there is a significant relationship between dist
ribution channels and booking cancellation
```

In [127…
```
fig=plotly.histogram(df, x="distribution_channel", color="is_canceled", barr
fig.update_layout(
    title_text='Bookings by Distribution Channels', # title of plot
    title_x=0.5,
    xaxis_title_text='Distribution Channels', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
fig.show()
```

Bookings made through tour operators/tour agents have the highest booking and cancellation rates. The reasons behind this will be explored.

Distribution channels and Cancellation rates

In [128…
```
cancel_year = df.groupby(['distribution_channel', 'is_canceled']).size().to_
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

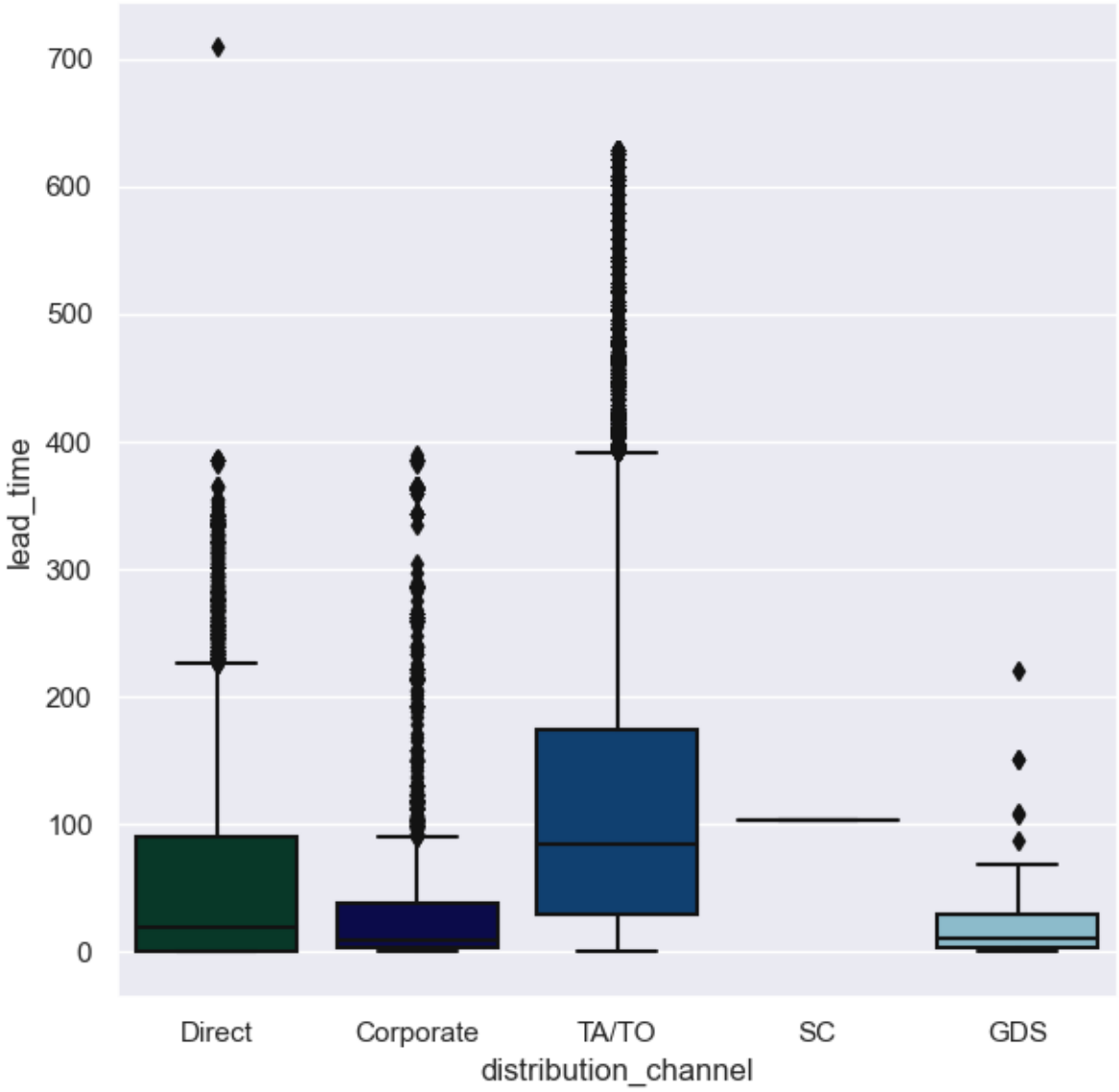
distribution_channel	is_canceled	count
Corporate	0	77.794654
	1	22.205346
Direct	0	82.369261
	1	17.630739
GDS	0	80.526316
	1	19.473684
SC	0	100.000000
TA/TO	0	58.776056
	1	41.223944

Direct Channels have the smallest cancellation rate. Travel agents have the biggest cancellation rate

Lead Time and Distribution Channels

In [128…
```
plt.figure(figsize=(7, 7))
sns.boxplot(x='distribution_channel', y='lead_time', data=df, palette='ocear
```

Out[1281]: <AxesSubplot:xlabel='distribution_channel', ylabel='lead_time'>



From the above boxplots, we can infer that the high cancellation rates among tour agents are mainly due to high lead time. As previously explained, higher lead times increase the probability of cancellations by 50%.

Label encoding

In [128…
```
df["TA/TO"] = df['distribution_channel'].apply(lambda x: 1 if x == 'TA/TO' e
df["Direct"] = df['distribution_channel'].apply(lambda x: 1 if x == 'Direct
df["Corporate"]= df['distribution_channel'].apply(lambda x: 1 if x == 'Corpo
df["GDS"]=df['distribution_channel'].apply(lambda x: 1 if x == 'GDS' else 0)
df["SC"]=df['distribution_channel'].apply(lambda x: 1 if x == 'SC' else 0)
```

In [128…
```
df=df.drop(["distribution_channel"],axis=1)
```

Country Feature Analysis

In [128…
```
df["country"].value_counts()
```

Out[1284]:
```
PRT    47896
GBR    12080
FRA    10364
ESP     8530
DEU     7271
       ...
DJI        1
HND        1
BWA        1
VGB        1
NAM        1
Name: country, Length: 177, dtype: int64
```

In [128…
```
df["country"].isnull().sum()
```

Out[1285]:
```
473
```

Filling NAs with the mode which is Portugal

In [128…
```
df["country"].fillna(df["country"].mode()[0], inplace=True) #filling NAs wi
```

Encoding countries by creating two new binary features: Portugal and International

In [128…
```
df["Portugal"] = df['country'].apply(lambda x: 1 if x == 'PRT' else 0) #the
df["International"]=df['country'].apply(lambda x: 0 if x == 'PRT' else 1)
```

In [128…
```
df=df.drop(["country"],axis=1)
```

Portugal and International Feature Analysis

In [128…
```
df.corr(method='pearson')['is_canceled']["Portugal"]
```

Out[1289]: 0.33827672176175844

In [129…
```
df.corr(method='pearson')['is_canceled']["International"]
```

Out[1290]: -0.3382767217617613

In [129…
```
fig=plotly.histogram(df, x="Portugal", color="is_canceled", barmode="stack",
fig.update_layout(
    title_text='Cancellations from Portuguese guests', # title of plot
    title_x=0.5,
    xaxis_title_text='Portuguese Guests', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
fig.show()
```

In [129…
```
fig=plotly.histogram(df, x="International", color="is_canceled", barmode="st
fig.update_layout(
    title_text='Cancellations from International guests', # title of plot
    title_x=0.5,
    xaxis_title_text='International Guests', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
fig.show()
```

Portuguese guests have a higher cancellation rate than International guests. However, Portuguse guests also make up the majority of customers in this data set.

Cancellation rates among nationalities

In [129…
```
cancel_year = df.groupby(['Portugal','is_canceled']).size().to_frame(name='
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                          count
Portugal is_canceled
0        0            76.342380
         1            23.657620
1        0            43.073043
         1            56.926957
```

In [129…
```
cancel_year = df.groupby(['International','is_canceled']).size().to_frame(na
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                              count
International is_canceled
0             0            43.073043
              1            56.926957
1             0            76.342380
              1            23.657620
```

Deposit types and Nationalities

In [129…
```
cancel_year = df.groupby(['Portugal','deposit_type']).size().to_frame(name=
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                           count
Portugal deposit_type
0        No Deposit    99.269677
         Non Refund     0.567394
         Refundable     0.162929
1        No Deposit    70.563791
         Non Refund    29.336972
         Refundable     0.099237
```

In [129…
```
cancel_year = df.groupby(['International','deposit_type']).size().to_frame(
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                                 count
International deposit_type
0             No Deposit    70.563791
              Non Refund    29.336972
              Refundable     0.099237
1             No Deposit    99.269677
              Non Refund     0.567394
              Refundable     0.162929
```

Conclusion:

Local portuguese customers have the highest cancellation rate. One can notice that the no-deposit type is the highest among Portuguese customers. This might explain the high cancellation rate. Portuguese guests have nothing to lose if they decide to cancel their bookings.

Market Segments Feature Analysis

In [129…
```
df["market_segment"].value_counts()
```

Out[1297]:
```
Online TA        55941
Offline TA/TO    24046
Groups           19759
Direct           12419
Corporate         5231
Complementary      711
Aviation           231
Name: market_segment, dtype: int64
```

Chi-Square Test

H0: no relationship between the variables

H1: there is a relationship between the variables

In [129…
```
chi_square_args = pd.crosstab(df['market_segment'], df['is_canceled']).value

_, p_value, _, _ = scipy.stats.chi2_contingency(chi_square_args)
print("The p-value is:",p_value)

if p_values<0.05:
    print("We reject null hypothesis, there is a significant relationship be
else:
    print("accept null hypothesis")
```

```
The p-value is: 0.0
We reject null hypothesis, there is a significant relationship between mark
et segments and booking cancellation
```

In [129…
```
fig=plotly.histogram(df, x="market_segment", color="is_canceled", barmode="s
fig.update_layout(
    title_text='Booking by Market segment', # title of plot
    title_x=0.5,
    xaxis_title_text=' Market segment', # xaxis label
    yaxis_title_text='Bookings', # yaxis label
    bargap=0.2, # gap between bars of adjacent location coordinates
    bargroupgap=0.1 # gap between bars of the same location coordinates
)
fig.show()
```

Online Tour agents have the highest cancellation rates and booking numbers

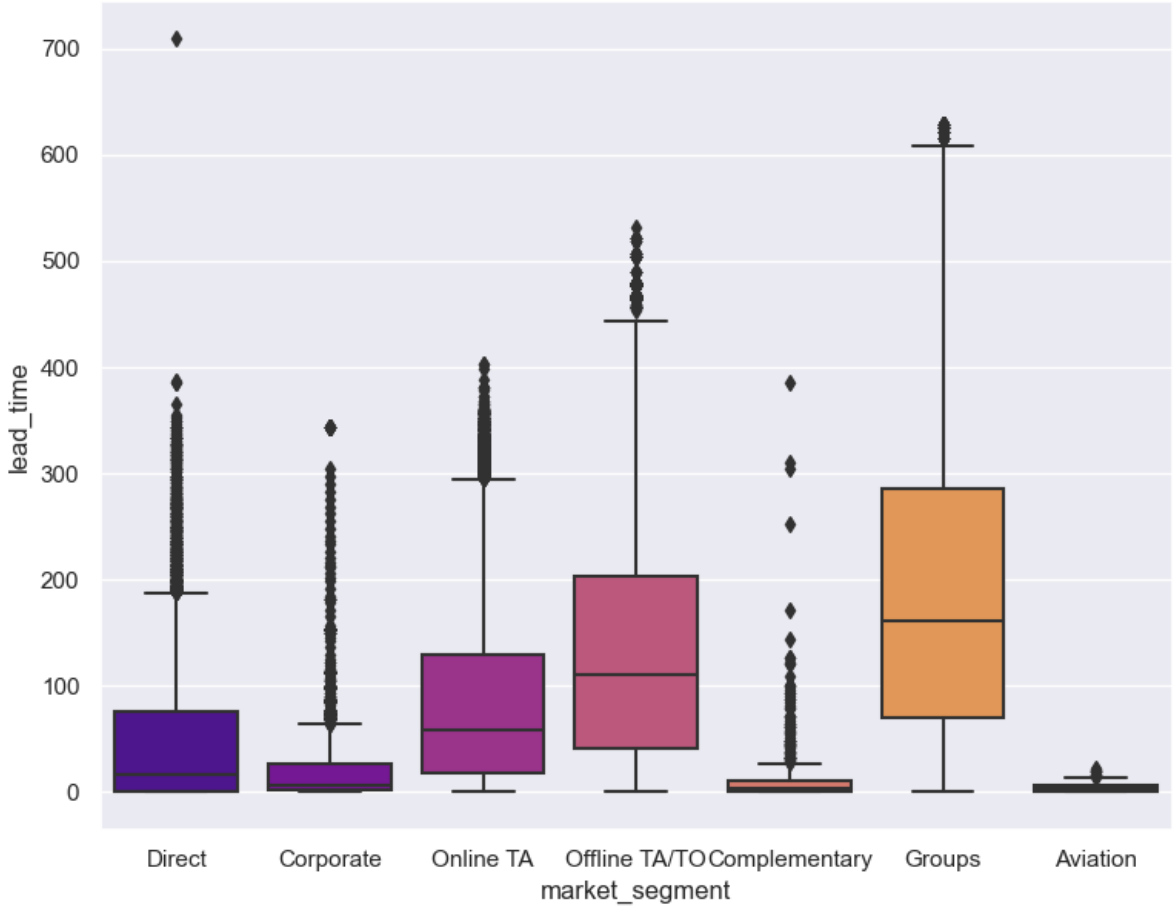
Cancelllation rates and Market segments

In [130…
```
cancel_year = df.groupby(['market_segment','is_canceled']).size().to_frame(n
cancel_percentage = cancel_year.groupby(level=0).apply(lambda x:100 * x / f
print(cancel_percentage)
```

```
                                   count
market_segment is_canceled
Aviation       0            77.922078
               1            22.077922
Complementary  0            87.904360
               1            12.095640
Corporate      0            81.093481
               1            18.906519
Direct         0            84.523714
               1            15.476286
Groups         0            38.797510
               1            61.202490
Offline TA/TO  0            65.482825
               1            34.517175
Online TA      0            63.091471
               1            36.908529
```

In [130…
```
plt.figure(figsize=(9, 7))
sns.boxplot(x='market_segment', y='lead_time', data=df, palette='plasma')
```

Out[1301]:
```
<AxesSubplot:xlabel='market_segment', ylabel='lead_time'>
```



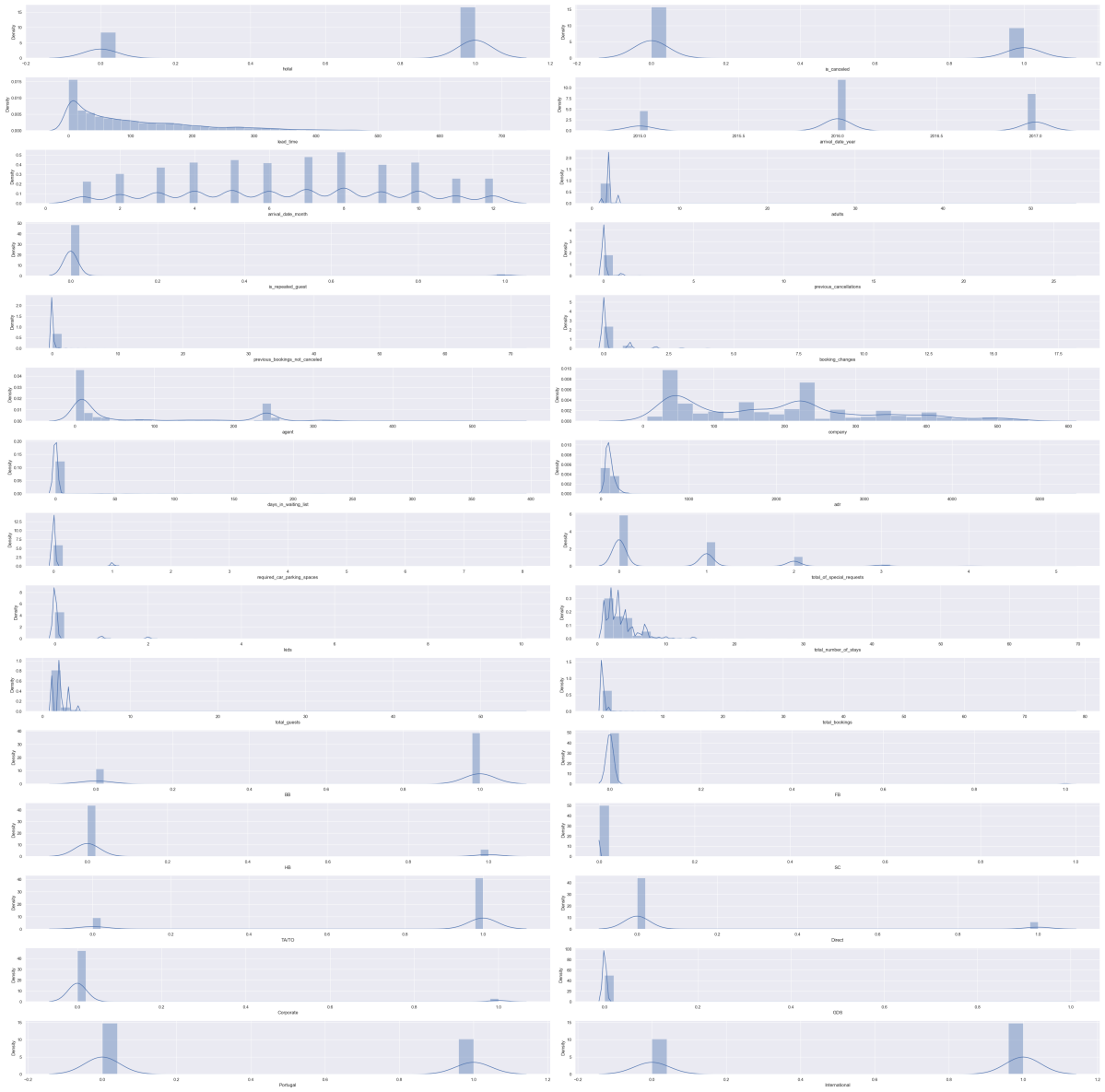
Market segments of type group have the highest cancellation rates. It can also be observe that groups have the highest lead time. As previously explained, higher lead times increase the probability of cancellations by 50%.

Distributions of Numerical Variables

In [130…
```
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
num = df.loc[:, (df.dtypes == int) | (df.dtypes == float)].columns.tolist()
#36 numerical variables

plt.figure(figsize=(40, 60))
for i in range(0, len(num)):
    plt.subplot(23, 2,i+1)
    sns.distplot(df[num[i]])
    plt.tight_layout()
```



In [130…

```
df_num=df.select_dtypes(include=np.number)

df_num
```

Out[1303]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	adults	is_r
2	0	0	7	2015	7	1	
3	0	0	13	2015	7	1	
4	0	0	14	2015	7	2	
5	0	0	14	2015	7	2	
6	0	0	0	2015	7	2	
...	...	...	...	...	...	...	
119385	1	0	23	2017	8	2	
119386	1	0	102	2017	8	3	
119387	1	0	34	2017	8	2	
119388	1	0	109	2017	8	2	
119389	1	0	205	2017	8	2	

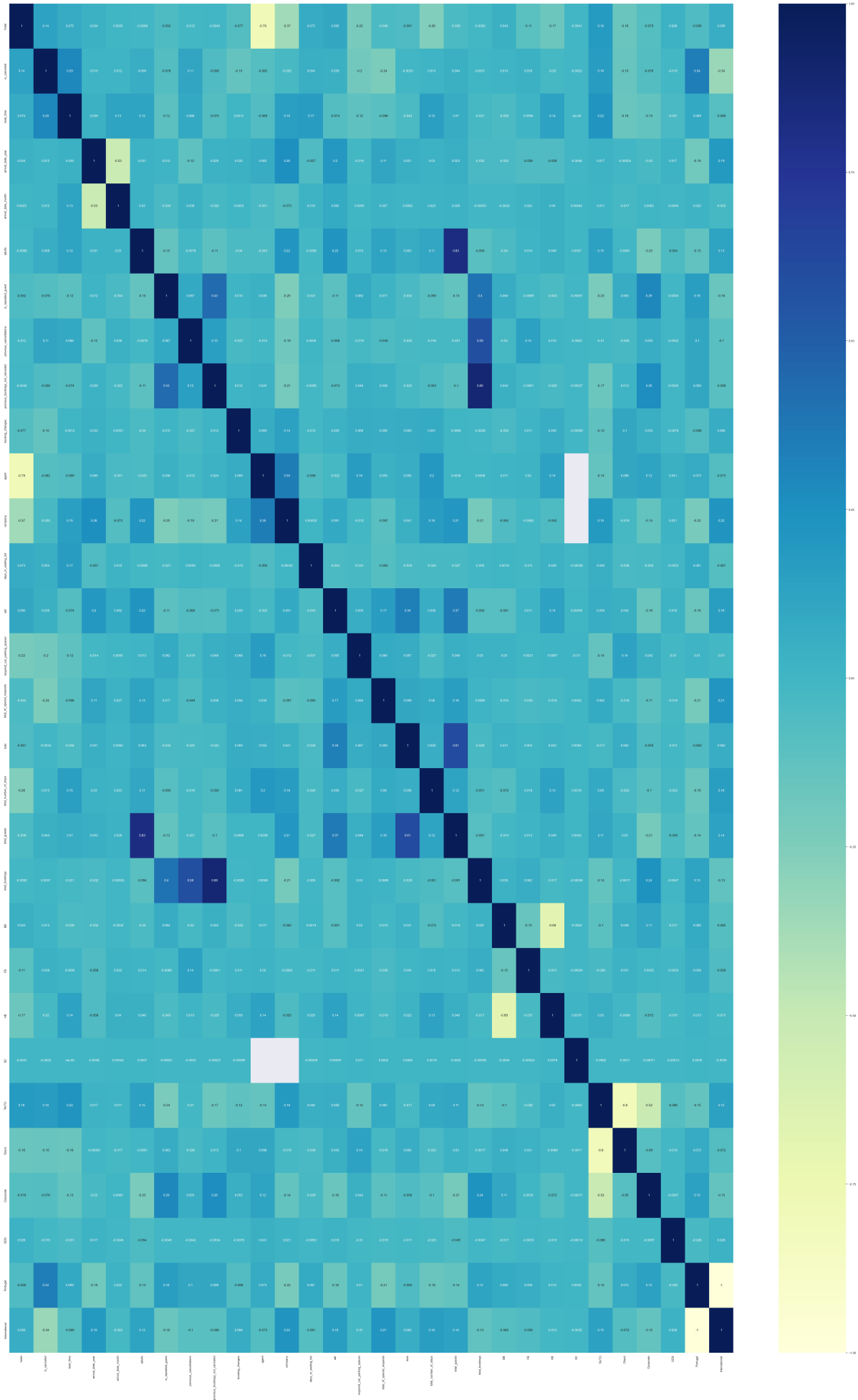
118338 rows × 30 columns

In [130…

```
plt.figure(figsize=(60, 90))

sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)

# displaying heatmap
plt.show()
```



Reservation Status Feature Analysis

Checking for missing values

In [130…
```
df["reservation_status"].isnull().sum()
```

Out[1305]:
```
0
```

Checking the unique values of the variables

In [130…
```
df["reservation_status"].value_counts()
```

Out[1306]:
```
Check-Out    74250
Canceled     42898
No-Show       1190
Name: reservation_status, dtype: int64
```

Check-Out: the guest has fulfilled their stay at the hotel
Canceled: the guest has canceled their reservation prior to the date of arrival
No-Show: the guest neither fulfilled nor canceled their reservation and did not show up on the day of check in

It is evident that there are guests that do not show up on the day of the bookings, which have a huge impact on the revenue management system. It is important to discover the motives behind the customer's no show decision.

In [130…
```
#grouping the datasets based on deposit type and reservation status
df.groupby(['deposit_type', 'reservation_status']).size()
```

Out[1307]:
```
deposit_type  reservation_status
No Deposit    Canceled              28403
              Check-Out             74031
              No-Show                1155
Non Refund    Canceled              14460
              Check-Out                93
              No-Show                  34
Refundable    Canceled                 35
              Check-Out               126
              No-Show                   1
dtype: int64
```

It is evident that the highest no-shows occur in reservations done without any deposit requirements.

In [130…
```
df["reservation_status"].groupby(df["is_canceled"]).value_counts()
```

Out[1308]:
```
is_canceled  reservation_status
0            Check-Out             74250
1            Canceled              42898
             No-Show                1190
Name: reservation_status, dtype: int64
```

Both the "canceled" and the "No-Show" classes are classified in the label variable "is_canceled" as canceled

Label Encoding of the variable "reservation_status"

In [130…
```
#storing the unique values in a variable called class_names
class_names = df['reservation_status'].unique()
```

```
class_names
```

Out[1309]:
```
array(['Check-Out', 'Canceled', 'No-Show'], dtype=object)
```

In [131…
```
#performing a for-loop to replace each class with its corresponding label (r
for i in range(len(class_names)):
    df.loc[ df['reservation_status'] == class_names[i],'reservation_status']

#mapping the "No-Show" varaible with the "Canceled" variable by setting its
df["reservation_status"]=df["reservation_status"].replace(2,1)
```

In [131…
```
#displaying the value of both calsses after their label encoding
df["reservation_status"].value_counts()
```

Out[1311]:
```
0    74250
1    44088
Name: reservation_status, dtype: int64
```

Checking the correlation between this feature variable and the label variabel

In [131…
```
tablers = pd.crosstab(index=df["is_canceled"],columns=df["reservation_status
tablers
```

Out[1312]:

reservation_status	0	1
is_canceled		
0	74250	0
1	0	44088

In [131…
```
chi2_contingency(tablers)[1]
```

Out[1313]:
```
0.0
```

By choosing a p-value of 0.05, it is evident that both variables are perfectly correlated, since the value of the chi2 test is 0<0.05. Hence, this feature is redundant and will be therefore droped from the dataset.

In [131…
```
df.drop("reservation_status",axis=1,inplace=True)
```

Special Requests Feature Analysis

In [131…
```
df["total_of_special_requests"].isna().sum()
```

Out[1315]:
```
0
```

Statistical Distribution of the Variable

In [131…
```
colors = ['#A56CC1', '#A6ACEC']
fig = px.histogram(df, x="total_of_special_requests", color="is_canceled",
                   marginal="box",
                   hover_data=df.columns,
                   color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
                   template="plotly_white")
fig.update_layout(plot_bgcolor="white", title="Distribution of the Total Nur
fig.show()
```

If we compare the distribution of the total number of special requests based on the reservation status, it becomes clear that the distributions are almost identical, which means that this feature does not have a huge role in determining any potential cancellations

In [131…
```
df["total_of_special_requests"].describe()
```

Out[1317]:
```
count    118338.000000
mean          0.571042
std           0.792804
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max           5.000000
Name: total_of_special_requests, dtype: float64
```

In [131…
```
df.corr()['is_canceled']["total_of_special_requests"]
```

Out[1318]:
```
-0.2365990365876268
```

It is evident that the distribution of the variable is left skewed. The majority of the reservations did not ask for any special requests. Moreover, there are 3 outliers shown by the boxplots of reservations having 3, 4 and 5 special requests.

In [131…
```
corr=df.groupby('total_of_special_requests')['is_canceled'].apply(list)
Anova = f_oneway(*corr)
```

```
print('P-Value for Anova is: ', Anova[1])
```

```
P-Value for Anova is:  0.0
```

There exists a weak negative relationship between the total number of special requests and the cancelation status

Required Car Parking Spaces Feature Analysis

Checking for missing values

In [132…
```
df["required_car_parking_spaces"].isnull().sum()
```

Out[1320]:
```
0
```

Checking for unique values

In [132…
```
df["required_car_parking_spaces"].value_counts()
```

Out[1321]:
```
0    110947
1      7358
2        28
3         3
8         2
Name: required_car_parking_spaces, dtype: int64
```

Statistical Distribution of the Variable

In [132…
```
colors = ['#A56CC1', '#A6ACEC']
fig = px.histogram(df, x="required_car_parking_spaces", color="is_canceled",
                   marginal="box",
                   hover_data=df.columns,
                   color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
                   template="plotly_white")
fig.update_layout(plot_bgcolor="white", title="Distribution of the Totel Nur
fig.show()
```

As can be seen from the graph above, the distribution of the variable is left skewed, with the majority of observations, requesting no car parking spaces regardless of them canceling or not. It is evident that there exists two major outliers which represent two reservations that was fullfilled and that requested around 8 car parking spaces. This will be further looked into to try and discover the reason behind it.

In [132…

```
fig = px.box(df, y="required_car_parking_spaces", color="is_canceled",
             template="plotly_white",
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

In [132…
```
df[df["required_car_parking_spaces"]==8]
```

Out[1324]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	adults	mark
29045	0	0	26	2017	3	2	
29046	0	0	138	2017	3	2	

2 rows × 36 columns

It can be seen that the guests who requested 8 car parking spaces were of type "Transient-Party". Transient-Party, transient guests that are associated with at least another transient booking and could viewed as a group that require a short stay at a hotel. Moreover, the above boxplot assumes that as the number of car parking spots required increases, the booking is more likely to be fullfilled

In [132…
```
df.corr()['is_canceled']["required_car_parking_spaces"]
```

Out[1325]:
```
-0.19696441224308822
```

In [132…
```
cars=df.groupby('required_car_parking_spaces')['is_canceled'].apply(list)
Anova = f_oneway(*cars)
print('P-Value for Anova is: ', Anova[1])
```

```
P-Value for Anova is:  0.0
```

Average Daily Rate Feature Analysis

Checking for missing values

In [132…
```
df["adr"].isnull().sum()
```

Out[1327]:
```
0
```

Statistical Distribution of the Variable

In [132…
```
colors = ['#A56CC1', '#A6ACEC']
fig = px.histogram(df, x="adr", color="is_canceled",
                   marginal="box",
                   hover_data=df.columns,
                   color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
                   template="plotly_white")
fig.update_layout(plot_bgcolor="white", title="Distribution of the Average D
fig.show()
```

In [132…
```
df[df["adr"]>500]
```

Out[1329]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	adults	mar
15083	0	0	1	2015	7	2	
48515	1	1	35	2016	3	2	
111403	1	0	0	2017	5	1	

3 rows × 36 columns

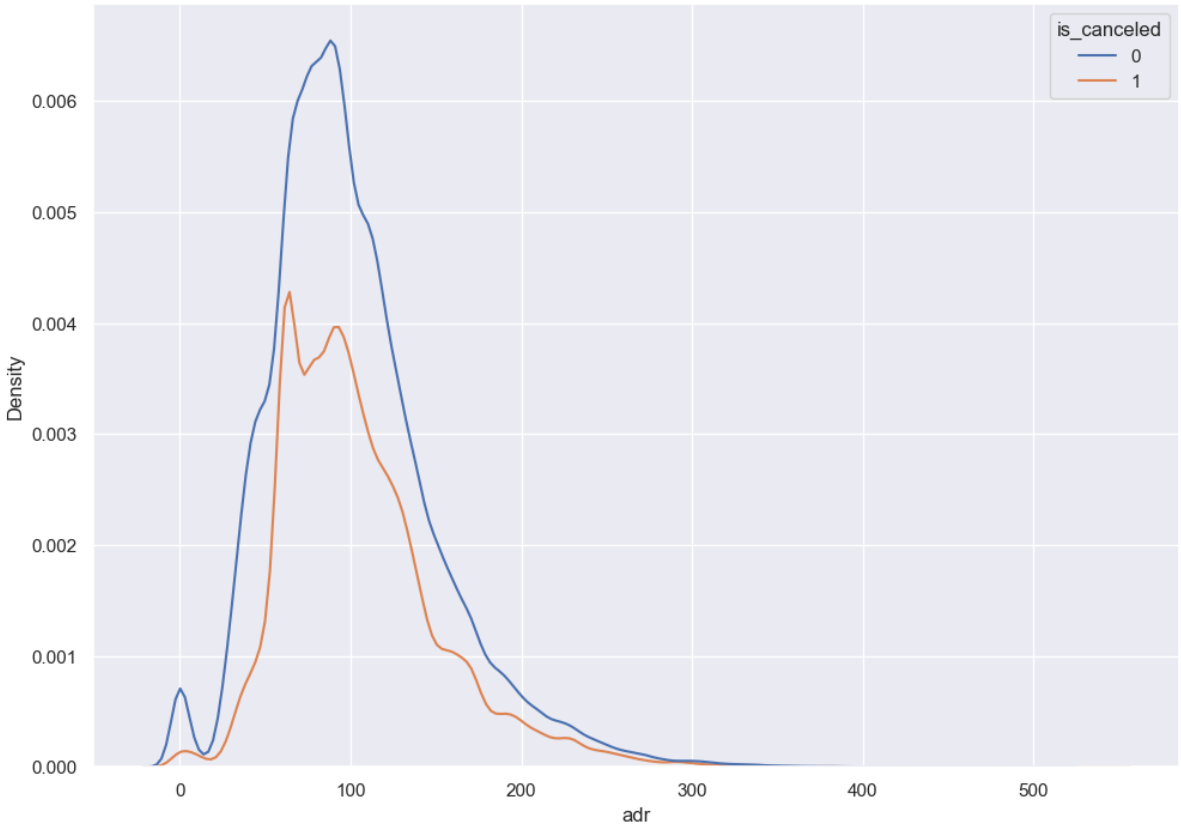
There is a value of the average daily rate that is 5400, which is represented as a major outlier in the above plot. The only explanation for this outlier could be a typo. Accordingly 5400 will be changed to be 540 and the plots will be reploted

In [133…
```
df["adr"]=df["adr"].replace(5400.0,540)
```

In [133…
```
colors = ['#A56CC1', '#A6ACEC']
fig = px.histogram(df, x="adr", color="is_canceled",
                   marginal="box",
                   hover_data=df.columns,
                   color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
                   template="plotly_white")
fig.update_layout(plot_bgcolor="white", title="Distribution of the Average D
fig.show()
```

In [133…
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.kdeplot(x='adr', hue='is_canceled', data=df)
```

Out[1332]: <AxesSubplot:xlabel='adr', ylabel='Density'>



The distributions above assume that the average daily rate of reservations that were canceled are higher than those who were fullfilled. The median of the average daily rate of canceled reservations lies around 96.2, while the average daily rate of fullfilled reservation lies around 92.5

In [133…
```
df["adr"].groupby(df["arrival_date_month"]).mean()
```

Out[1333]:
```
arrival_date_month
1      71.121509
2      74.228054
3      80.824289
4     100.793782
5     109.470897
6     117.188230
7     127.664760
8     140.918852
9     105.341845
10     88.679549
11     74.615753
12     82.223800
Name: adr, dtype: float64
```

In [133…
```
df["adr"].groupby(df["arrival_date_year"]).mean()
```

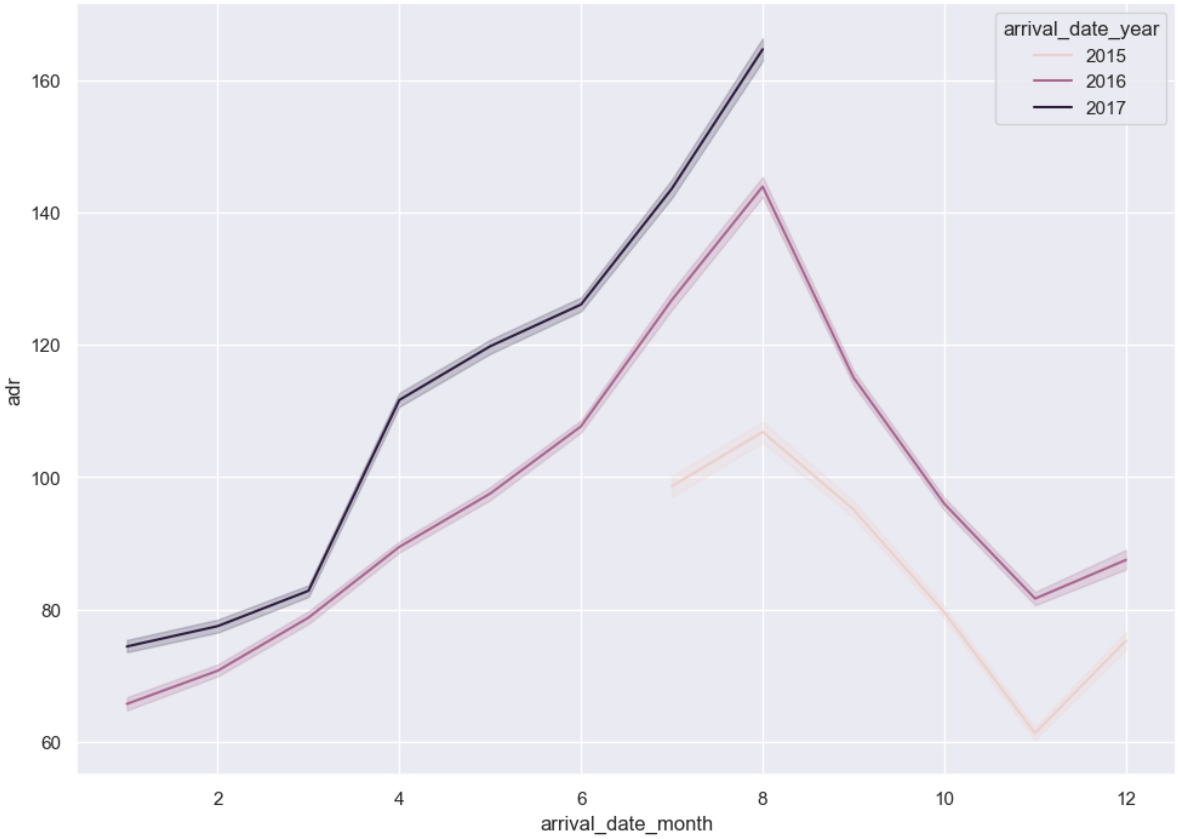
Out[1334]:
```
arrival_date_year
2015     87.924321
2016     99.089431
2017    115.160385
Name: adr, dtype: float64
```

In [133…
```
df["adr"].groupby(df["is_canceled"]).mean()
```

Out[1335]:
```
is_canceled
0    101.048399
1    105.012991
Name: adr, dtype: float64
```

In [133…
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.lineplot(data=df, x="arrival_date_month", y="adr", hue="arrival_date_yea
```

Out[1336]:
```
<AxesSubplot:xlabel='arrival_date_month', ylabel='adr'>
```



Moreover, the above graph shows that the ADR increases over the years. Bookings made during the months during summer(April, May, June, July, and August) experience the highest rate relative to the whole year, which makes them high season. The ADR experiences a huge drop in January.

In [133…
```
df.corr()['is_canceled']["adr"]
```

Out[1337]:
```
0.04026497057962397
```

In [133…
```
adr=df.groupby('adr')['is_canceled'].apply(list)
Anova = f_oneway(*adr)
print('P-Value for Anova is: ', Anova[1])
```

```
P-Value for Anova is:  0.0
```

Customer Type Feature Analysis

Categorical variable that indicates the type of customers that are booking rooms in the hotel. The variable has 4 levels.
Transient: individuals that need a short, mostly urgent stay at a hotel,
Transient-party: individuals that need a short, mostly urgent stay at a hotel but are associated with at least another transient booking,

Group: guests that request an accommodation at a hotel as a group, e.g. families, tourist groups, school groups, etc.,
Contract: employees that request a reservation at a hotel through a business to business contract, which offers them rooms at discounted rates).

Checking for missing values

In [133…
```
df["customer_type"].isnull().sum()
```

Out[1339]:
```
0
```

Checking the unique values of the variable

In [134…
```
df["customer_type"].value_counts()
```

Out[1340]:
```
Transient          88796
Transient-Party    24919
Contract            4055
Group                568
Name: customer_type, dtype: int64
```

The most common customer type in this dataset are of type transient, which are individuals that need a short stay in a hotel

In [134…
```
df["customer_type"].groupby(df["is_canceled"]).value_counts()
```

Out[1341]:
```
is_canceled  customer_type
0            Transient          52383
             Transient-Party    18564
             Contract            2793
             Group                510
1            Transient          36413
             Transient-Party     6355
             Contract            1262
             Group                 58
Name: customer_type, dtype: int64
```

In [134…
```
fig = px.histogram(df, x="customer_type",
             color='is_canceled', barmode='group',
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
             height=400,
              template="plotly_white")
fig.show()
```

One-Hot Encoding of the variable "customer_type"

In [134…
```
df["Transient"]=df["customer_type"].apply(lambda x:1 if x=="Transient" else
df["Contract"]=df["customer_type"].apply(lambda x:1 if x=="Contract" else 0
df["Transient-Party"]=df["customer_type"].apply(lambda x:1 if x=="Transient-
df["Group"]=df["customer_type"].apply(lambda x:1 if x=="Group" else 0)
```

In [134…
```
df["customer_type"].value_counts()
```

Out[1344]:
```
Transient          88796
Transient-Party    24919
Contract            4055
Group                568
Name: customer_type, dtype: int64
```

In [134…
```
tablect = pd.crosstab(index=df["is_canceled"],columns=df["customer_type"])
tablect
```

Out[1345]:

is_canceled	Contract	Group	Transient	Transient-Party
0	2793	510	52383	18564
1	1262	58	36413	6355

In [134…
```
chi2_contingency(tablect)[1]
```

Out[1346]:
```
0.0
```

In [134…
```
df.drop("customer_type",axis=1,inplace=True)
```

Days in Waiting List Feature Analysis

Checking for missing values

In [134…
```
df["days_in_waiting_list"].isnull().sum()
```

Out[1348]:
```
0
```

Statistical Distribution of the Variable

In [134…
```
df["days_in_waiting_list"].groupby([df["is_canceled"]]).describe()
```

Out[1349]:

is_canceled	count	mean	std	min	25%	50%	75%	max
0	74250.0	1.602640	14.851689	0.0	0.0	0.0	0.0	379.0
1	44088.0	3.575077	21.520974	0.0	0.0	0.0	0.0	391.0

In [135…
```
colors = ['#A56CC1', '#A6ACEC']
fig = px.histogram(df, x="days_in_waiting_list", color="is_canceled",
                   marginal="box",
                   hover_data=df.columns,
                   color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
                   template="plotly_white")
fig.update_layout(plot_bgcolor="white", title="Distribution of the Total Day
fig.show()
```

In [135…
```
fig = px.box(df, y="days_in_waiting_list", color="is_canceled",
             template="plotly_white",
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'})
```

```
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

The average number of days spent in the waiting list is higher than canceled bookings than in fullfilled bookings. One might infer that as the number of days on the waiting list increase, the reservation is more likely to be cancelled. However, looking at the distributions above, such conclusion is not applicable because both distributions are almost identical.

In [135…
```
df.corr()['is_canceled']["days_in_waiting_list"]
```

Out[1352]:
```
0.05400243059183626
```

In [135…
```
df["days_in_waiting_list"].groupby([df["arrival_date_month"]]).size()
```

Out[1353]:
```
arrival_date_month
1      5853
2      7980
3      9699
4     11024
5     11688
6     10879
7     12553
8     13776
9     10466
10    11048
11     6706
12     6666
Name: days_in_waiting_list, dtype: int64
```

The high-season months stated above are the months with the highest number of days in the waiting list

Company Feature Analysis

Cheking for missing values

In [135…
```
df["company"].isnull().sum()
```

Out[1354]:
```
111645
```

Calculating the percentage of missing values

In [135…
```
(df["company"].isnull().sum()/len(df))*100
```

Out[1355]:
```
94.34416670891852
```

Since more thatn 94% of the data in this column is missing, it won't be of any benefit in our predicition as it does not capture any information that is generalizable. Accordingly, this feature variable will be dropped.

In [135…
```
df.drop("company",axis=1,inplace=True)
```

Agent Feature Analysis

Numeric variable that indicates the hotel agent sitting on the front desk of a hotel who is responsible for booking the rooms for the customers, verifying the reservation, and collecting payments.

Checking for missing values

In [135…
```
df["agent"].isnull().sum()
```

Out[1357]:
```
16060
```

In [135…
```
(df["agent"].isnull().sum()/len(df))*100
```

Out[1358]:
```
13.571295779884737
```

In [135…
```
df["agent"].value_counts()
```

Out[1359]:
```
9.0      31675
240.0    13795
1.0       7185
14.0      3603
7.0       3515
         ...
93.0         1
54.0         1
497.0        1
337.0        1
59.0         1
Name: agent, Length: 333, dtype: int64
```

Deposit Type Feature Analysis

Categorical variable that indicates the deposit type. This variable has 3 levels.
no deposit: during room reservation the guest is not required to pay a deposit,
No refund: in case of a cancellation made by the guest the amount of money paid in advance is non-refundable,
Refundable: in case of a cancellation made by the guest the amount of money paid is refundable

Checking for missing values

In [136…
```
df["deposit_type"].isnull().sum()
```

Out[1360]:
```
0
```

Checking the unique values of the variables

In [136…
```
df["deposit_type"].value_counts()
```

Out[1361]:
```
No Deposit    103589
Non Refund     14587
Refundable       162
Name: deposit_type, dtype: int64
```

The most common deposit type in this dataset is the "no deposit" type, followed by the "non refundable" and the "refundable" type.

In [136…
```
df["deposit_type"].groupby(df["is_canceled"]).value_counts()
```

Out[1362]:
```
is_canceled  deposit_type
0            No Deposit      74031
             Refundable        126
             Non Refund         93
1            No Deposit      29558
             Non Refund      14494
             Refundable         36
Name: deposit_type, dtype: int64
```

In [136…
```
fig = px.histogram(df, x="deposit_type",
             color='is_canceled', barmode='group',
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
             height=400,
```

```
                  template="plotly_white")
fig.show()
```

The above table shows that even non-refundable reservations are also canceled

In [136…
```
tabledt = pd.crosstab(index=df["is_canceled"],columns=df["deposit_type"])
tabledt
```

Out[1367]:

deposit_type	No Deposit	Non Refund	Refundable
is_canceled			
0	74031	93	126
1	29558	14494	36

In [136…
```
chi2_contingency(tabledt)[1]
```

Out[1366]:
```
0.0
```

Booking Changes Feature Analysis

In [136…
```
df["booking_changes"].isnull().sum()
```

Out[1369]:
```
0
```

Statistical Distribution of the variables

In [137…
```
fig = px.box(df, y="booking_changes", color="is_canceled",
             template="plotly_white",
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

In [137…
```
df["booking_changes"].describe()
```

Out[1371]:
```
count    118338.000000
mean          0.218096
std           0.637217
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          18.000000
Name: booking_changes, dtype: float64
```

The distributions above are left skewed, as the majority of reservations are done without any booking changes. Since both distributions are almost identical, one might infer that there is no direct impact of the booking changes on the customer's behaviour to cancel or not

In [137…
```
df.corr()['is_canceled']["booking_changes"]
```

Out[1372]:
```
-0.14521883916427447
```

There is a negative correlation between the reservation status and the number of booking changes undertaken by a reserving guest.

Assigned Room Type Feature Analysis

Categorical variable that indicates if a hotel guest got assigned a different room type than the one reserved. This variable has 10 levels , with each room type assigned a letter from the alphabet (A, B, C, D, E, F, G, H, L, and P).

In [137…
```
df["assigned_room_type"].isnull().sum()
```

Out[1374]:
```
0
```

In [137…
```
df["assigned_room_type"].value_counts()
```

Out[1375]:
```
A    73752
D    25202
E     7763
F     3728
G     2531
C     2350
B     1957
H      708
I      219
K      127
L        1
Name: assigned_room_type, dtype: int64
```

In [137…
```
df["assigned_room_type"].groupby(df["is_canceled"]).value_counts()
```

Out[1376]:
```
is_canceled  assigned_room_type
0            A                    40826
             D                    18844
             E                     5802
             F                     2801
             C                     1904
             G                     1752
             B                     1532
             H                      457
             I                      215
             K                      117
1            A                    32926
             D                     6358
             E                     1961
             F                      927
             G                      779
             C                      446
             B                      425
             H                      251
             K                       10
             I                        4
             L                        1
Name: assigned_room_type, dtype: int64
```

The room type A is the most booked room accross all reservations. The rooms of type P and L are always cancelled after booking.

In [137…
```
df["adr"].groupby(df["assigned_room_type"]).mean()
```

Out[1377]:
```
assigned_room_type
A     93.414257
B     96.674732
C    114.630217
D    107.956754
E    118.356199
F    152.827020
G    167.898807
H    172.349025
I     67.699954
K    106.464724
L      8.000000
Name: adr, dtype: float64
```

In [137…
```
fig = px.histogram(df, x="assigned_room_type",
             color='is_canceled', barmode='group',
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
             height=400,
              template="plotly_white")
fig.show()
```

In [137…
```
fig = px.box(df,
          x = 'assigned_room_type',
          y = 'adr',
          color = 'is_canceled',
          title = "Price Distribution According to Room Type"
        );
fig.show()
```

In [138…
```
tableart = pd.crosstab(index=df["is_canceled"],columns=df["assigned_room_typ
tableart
```

Out[1380]:

assigned_room_type	A	B	C	D	E	F	G	H	I	K	L
is_canceled											
0	40826	1532	1904	18844	5802	2801	1752	457	215	117	0
1	32926	425	446	6358	1961	927	779	251	4	10	1

In [138…
```
chi2_contingency(tableart)[1]
```

Out[1381]:
```
0.0
```

Label Encoding of the variable "assigned room type"

In [138…
```
#storing the unique values in a variable called class_names
class_names = df['assigned_room_type'].unique()
class_names
```

Out[1382]:
```
array(['C', 'A', 'D', 'E', 'G', 'F', 'I', 'B', 'H', 'L', 'K'],
      dtype=object)
```

In [138…
```
#performing a for-loop to replace each class with its corresponding label (
for i in range(len(class_names)):
    df.loc[ df['assigned_room_type'] == class_names[i],'assigned_room_type'
```

Reserved Room Type Feature Analysis

Categorical variable that indicates the various room types that were reserved by hotel guests. This variable has 10 levels, with each room type assigned a letter from the alphabet (A, B, C, D, E, F, G, H, L, and P).

Checking for missing values

In [138…
```
df["reserved_room_type"].isnull().sum()
```

Out[1384]:
```
0
```

In [138…
```
df["reserved_room_type"].value_counts()
```

Out[1385]:
```
A    85388
D    19095
E     6481
F     2877
G     2073
C      923
B      898
H      597
L        6
Name: reserved_room_type, dtype: int64
```

In [138…
```
fig = px.histogram(df, x="reserved_room_type",
             color='is_canceled', barmode='group',
             color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
             height=400,
              template="plotly_white")
fig.show()
```

The most room type that experiences reservations is room type A. However, room type I does not get reserved at all and whenever it gets assigned, the booking results in

cancelation.

In [138…
```
subset=df[df["reserved_room_type"]!=df["assigned_room_type"]]
len(subset)
```

Out[1387]:
```
118338
```

In [138…
```
subset["is_canceled"].sum()
```

Out[1388]:
```
44088
```

Out of the 14917 of reserving guests that were assigned different rooms than the one they booked, 802 of them canceled their bookings. Being assigned a different room than the one reserved could be a reason why guests cancel their booking.

In [138…
```
tablerrt = pd.crosstab(index=df["is_canceled"],columns=df["reserved_room_typ
tablerrt
```

Out[1389]:

reserved_room_type	A	B	C	D	E	F	G	H	L
is_canceled									
0	51782	617	616	12997	4574	1997	1311	352	4
1	33606	281	307	6098	1907	880	762	245	2

In [139…
```
chi2_contingency(tablerrt)[1]
```

Out[1390]:
```
3.3141155717282356e-133
```

Label Encoding of the variable "reserved room type"

In [139…
```
#storing the unique values in a variable called class_names
class_names = df['reserved_room_type'].unique()
class_names
```

Out[1391]:
```
array(['A', 'C', 'D', 'E', 'G', 'F', 'H', 'L', 'B'], dtype=object)
```

In [139…
```
#performing a for-loop to replace each class with its corresponding label (
for i in range(len(class_names)):
    df.loc[ df['reserved_room_type'] == class_names[i],'reserved_room_type']
```

Previous Booking Not Canceled

In [139…
```
df["previous_bookings_not_canceled"].isnull().sum()
```

Out[1393]:
```
0
```

In [139…
```
df["previous_bookings_not_canceled"].value_counts()
```

Out[1394]:
```
0     114756
1       1519
2        576
3        331
4        226
       ...
47         1
49         1
50         1
51         1
72         1
Name: previous_bookings_not_canceled, Length: 73, dtype: int64
```

In [139…
```
colors = ['#A56CC1', '#A6ACEC']
fig = px.histogram(df, x="previous_bookings_not_canceled", color="is_cancele
                   marginal="box",
                   hover_data=df.columns,
                   color_discrete_map = {0:'#7FA6EE',1:'#835AF1'},
                   template="plotly_white")
fig.update_layout(plot_bgcolor="white", title="Distribution of the Total Day
fig.show()
```

From the distribution above, it is evident that the majority of guests who reserved in the respective hotels from 2015 to 2017 are new guests that haven't booked before. However, out of the returning guests, the majority of them did not cancel their reservation and only a minority canceled thier bookings.

In [139…
```
df.corr()['is_canceled']["previous_bookings_not_canceled"]
```

Out[1396]:
```
-0.05758810775734415
```

Previous Cancellation Feature Analysis

In [139…
```
df["previous_cancellations"].isnull().sum()
```

Out[1397]:
```
0
```

In [139…
```
df["previous_cancellations"].value_counts()
```

Out[1398]:
```
0     111865
1       6042
2        114
3         65
24        48
11        35
4         31
26        26
25        25
6         22
19        19
5         19
14        14
13        12
21         1
Name: previous_cancellations, dtype: int64
```

In [140…
```
df.corr()['is_canceled']["previous_cancellations"]
```

Out[1401]:
```
0.11006340783788891
```

Is Repeated Guest Feature Analysis

In [140…
```
df["is_repeated_guest"].isnull().sum()
```

Out[1402]:
```
0
```

In [140…
```
df["is_repeated_guest"].value_counts()
```

Out[1403]:
```
0    114840
1      3498
Name: is_repeated_guest, dtype: int64
```

In [140…
```
tablerg = pd.crosstab(index=df["is_canceled"],columns=df["is_repeated_guest"
tablerg
```

Out[1404]:

is_repeated_guest	0	1
is_canceled		
0	71301	2949
1	43539	549

It is evident from the contingency table above, that the majority of returning guests fullfill their reservation, while only a very small group of returning guests cancel their booking. Newly customers could cancel or not cancel, however the majority do not cancel

In [140…
```
chi2_contingency(tablerg)[1]
```

Out[1405]:
```
1.0389591595185729e-157
```

In [140…
```
df.head(5)
```

Out[1406]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	adults	market_se
2	0	0	7	2015	7	1	
3	0	0	13	2015	7	1	Co
4	0	0	14	2015	7	2	Or
5	0	0	14	2015	7	2	Or
6	0	0	0	2015	7	2	

5 rows × 38 columns

In [141…
```
df["No_Deposit"]=df["deposit_type"].apply(lambda x:1 if x=="No Deposit" else
df["Non_Refund"]=df["deposit_type"].apply(lambda x:1 if x=="Non Refund" else
df["Refundable"]=df["deposit_type"].apply(lambda x:1 if x=="Refundable" else
```

In [141…
```
df=df.drop(["market_segment"],axis=1)
df=df.drop(["deposit_type"],axis=1)
df=df.drop(["agent"],axis=1)
df=df.drop(["reservation_status_date"],axis=1)
```

In [141…
```
df.columns
```

Out[1416]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'adults', 'is_repeated_guest',
       'previous_cancellations', 'previous_bookings_not_canceled',
       'reserved_room_type', 'assigned_room_type', 'booking_changes',
       'days_in_waiting_list', 'adr', 'required_car_parking_spaces',
       'total_of_special_requests', 'kids', 'total_number_of_stays',
       'total_guests', 'total_bookings', 'BB', 'FB', 'HB', 'SC', 'TA/TO',
       'Direct', 'Corporate', 'GDS', 'Portugal', 'International', 'Transi
ent',
       'Contract', 'Transient-Party', 'Group', 'No_Deposit', 'Non_Refun
d',
       'Refundable'],
      dtype='object')
```

In [141…
```
df.to_csv("final_data.csv")
```

In [ ]:
```
```