Import data from google drive

```
# Code to read csv file into Colaboratory:
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
id = "1bfKxTD7CU9q_lvEHpOCqHJhiU1pZ6aHV"
file = drive.CreateFile({'id':id})
file.GetContentFile('hotel_bookings.csv')
```

```
import pandas as pd
import numpy as np
```

```
hb=pd.read_csv("hotel_bookings.csv")
hb.head()
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_dat
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	

Data Preprocessing

Data Preprocessing

```
hb.isnull().sum().sort_values(ascending=False)/hb.shape[0]
```

```
company                         0.943069
agent                           0.136862
country                         0.004087
children                        0.000034
lead_time                       0.000000
arrival_date_year               0.000000
arrival_date_month              0.000000
arrival_date_week_number        0.000000
is_canceled                     0.000000
market_segment                  0.000000
arrival_date_day_of_month       0.000000
stays_in_weekend_nights         0.000000
stays_in_week_nights            0.000000
adults                          0.000000
babies                          0.000000
meal                            0.000000
reservation_status_date         0.000000
distribution_channel            0.000000
reservation_status              0.000000
is_repeated_guest               0.000000
previous_cancellations          0.000000
previous_bookings_not_canceled  0.000000
reserved_room_type              0.000000
assigned_room_type              0.000000
booking_changes                 0.000000
deposit_type                    0.000000
days_in_waiting_list            0.000000
customer_type                   0.000000
adr                             0.000000
required_car_parking_spaces     0.000000
total_of_special_requests       0.000000
hotel                           0.000000
dtype: float64
```

```
hb_new = hb.copy()
```

```
hb_new.drop("company",axis=1,inplace=True)
```

```
hb_new.agent.fillna(0, inplace=True)
```

```
hb_new.country.fillna(hb_new.country.mode()[0],inplace=True)
```

```
hb_new.children.fillna(hb_new.children.median(), inplace=True)
```

```
hb_new.isnull().sum().sort_values(ascending=False)
```

```
reservation_status_date         0
market_segment                  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                          0
babies                            0
meal                              0
country                           0
distribution_channel              0
reservation_status                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
hotel                             0
dtype: int64
```

```
hb_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 31 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   hotel                           119390 non-null  object 
 1   is_canceled                     119390 non-null  int64  
 2   lead_time                       119390 non-null  int64  
 3   arrival_date_year               119390 non-null  int64  
 4   arrival_date_month              119390 non-null  object 
 5   arrival_date_week_number        119390 non-null  int64  
 6   arrival_date_day_of_month       119390 non-null  int64  
 7   stays_in_weekend_nights         119390 non-null  int64  
 8   stays_in_week_nights            119390 non-null  int64  
 9   adults                          119390 non-null  int64  
 10  children                        119390 non-null  float64
 11  babies                          119390 non-null  int64  
 12  meal                            119390 non-null  object 
 13  country                         119390 non-null  object 
 14  market_segment                  119390 non-null  object 
 15  distribution_channel            119390 non-null  object 
 16  is_repeated_guest               119390 non-null  int64  
 17  previous_cancellations          119390 non-null  int64  
 18  previous_bookings_not_canceled  119390 non-null  int64  
 19  reserved_room_type              119390 non-null  object 
 20  assigned_room_type              119390 non-null  object 
 21  booking_changes                 119390 non-null  int64  
 22  deposit_type                    119390 non-null  object 
 23  agent                           119390 non-null  float64
 24  days_in_waiting_list            119390 non-null  int64  
 25  customer_type                   119390 non-null  object 
 26  adr                             119390 non-null  float64
 27  required_car_parking_spaces     119390 non-null  int64  
```

```
 28  total_of_special_requests       119390 non-null  int64
 29  reservation_status              119390 non-null  object
 30  reservation_status_date         119390 non-null  object
dtypes: float64(3), int64(16), object(12)
memory usage: 28.2+ MB
```

```
hb_new.children = hb_new.children.astype(int)
hb_new.agent = hb_new.agent.astype(int)
```

```
hb_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 31 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119390 non-null  int64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         119390 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           119390 non-null  int64
 24  days_in_waiting_list            119390 non-null  int64
 25  customer_type                   119390 non-null  object
 26  adr                             119390 non-null  float64
 27  required_car_parking_spaces     119390 non-null  int64
 28  total_of_special_requests       119390 non-null  int64
 29  reservation_status              119390 non-null  object
 30  reservation_status_date         119390 non-null  object
dtypes: float64(1), int64(18), object(12)
memory usage: 28.2+ MB
```

```
hb_new.describe().T
```

	count	mean	std	min	25%	
is_canceled	119390.0	0.370416	0.482918	0.00	0.00	
lead_time	119390.0	104.011416	106.863097	0.00	18.00	
arrival_date_year	119390.0	2016.156554	0.707476	2015.00	2016.00	20
arrival_date_week_number	119390.0	27.165173	13.605138	1.00	16.00	
arrival_date_day_of_month	119390.0	15.798241	8.780829	1.00	8.00	
stays_in_weekend_nights	119390.0	0.927599	0.998613	0.00	0.00	
stays_in_week_nights	119390.0	2.500302	1.908286	0.00	1.00	
adults	119390.0	1.856403	0.579261	0.00	2.00	
children	119390.0	0.103886	0.398555	0.00	0.00	
babies	119390.0	0.007949	0.097436	0.00	0.00	
is_repeated_guest	119390.0	0.031912	0.175767	0.00	0.00	
previous_cancellations	119390.0	0.087118	0.844336	0.00	0.00	
previous_bookings_not_canceled	119390.0	0.137097	1.497437	0.00	0.00	
booking_changes	119390.0	0.221124	0.652306	0.00	0.00	
agent	119390.0	74.828319	107.141953	0.00	7.00	

```
guests_0 = list(hb_new["adults"] + hb_new["children"] + hb_new["babies"] == 0)
hb_new.drop(hb_new.index[guests_0],inplace=True)
```

```
hb_new[hb_new['adr']<0]
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival
14969	Resort Hotel	0	195	2017	March	

```
hb_new = hb_new.drop(hb_new[hb_new.adr >2000].index)
```

```
pd.concat([hb_new[hb_new.babies>3], hb_new[hb_new.children>3]]).T
```

	46619	78656	328
hotel	City Hotel	City Hotel	Resort Hotel
is_canceled	0	0	1
lead_time	37	11	55
arrival_date_year	2016	2015	2015
arrival_date_month	January	October	July
arrival_date_week_number	3	42	29
arrival_date_day_of_month	12	11	12
stays_in_weekend_nights	0	2	4
stays_in_week_nights	2	1	10
adults	2	1	2
children	0	0	10
babies	10	9	0
meal	BB	BB	BB
country	PRT	GBR	PRT
market_segment	Online TA	Corporate	Offline TA/TO
distribution_channel	TA/TO	Corporate	TA/TO
is_repeated_guest	0	0	0
previous_cancellations	0	0	0
previous_bookings_not_canceled	0	0	0
reserved_room_type	D	A	D
assigned_room_type	D	B	D
booking_changes	1	1	2
deposit_type	No Deposit	No Deposit	No Deposit
agent	0	95	8

```
hb_new.drop([14969, 46619, 78656, 328], inplace=True)
```

```
hb_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119205 entries, 0 to 119389
Data columns (total 31 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119205 non-null  object
 1   is_canceled                     119205 non-null  int64
 2   lead_time                       119205 non-null  int64
 3   arrival_date_year               119205 non-null  int64
 4   arrival_date_month              119205 non-null  object
```

```
 5   arrival_date_week_number        119205 non-null  int64
 6   arrival_date_day_of_month       119205 non-null  int64
 7   stays_in_weekend_nights         119205 non-null  int64
 8   stays_in_week_nights            119205 non-null  int64
 9   adults                          119205 non-null  int64
 10  children                        119205 non-null  int64
 11  babies                          119205 non-null  int64
 12  meal                            119205 non-null  object
 13  country                         119205 non-null  object
 14  market_segment                  119205 non-null  object
 15  distribution_channel            119205 non-null  object
 16  is_repeated_guest               119205 non-null  int64
 17  previous_cancellations          119205 non-null  int64
 18  previous_bookings_not_canceled  119205 non-null  int64
 19  reserved_room_type              119205 non-null  object
 20  assigned_room_type              119205 non-null  object
 21  booking_changes                 119205 non-null  int64
 22  deposit_type                    119205 non-null  object
 23  agent                           119205 non-null  int64
 24  days_in_waiting_list            119205 non-null  int64
 25  customer_type                   119205 non-null  object
 26  adr                             119205 non-null  float64
 27  required_car_parking_spaces     119205 non-null  int64
 28  total_of_special_requests       119205 non-null  int64
 29  reservation_status              119205 non-null  object
 30  reservation_status_date         119205 non-null  object
dtypes: float64(1), int64(18), object(12)
memory usage: 29.1+ MB
```

```
hb_new.describe().T
```

	count	mean	std	min	25%	5
is_canceled	119205.0	0.370765	0.483012	0.0	0.0	0
lead_time	119205.0	104.110801	106.876568	0.0	18.0	69

Descriptive analysis

Comment: This section is a supplement to the descriptive analysis in data dictionary.xlsx.

```
import matplotlib.pyplot as plt
%matplotlib inline
```

2.1 Load dataset

```
# hb=pd.read_csv("hb.csv")
```

previous_bookings_not_canceled	119205.0	0.137083	1.498159	0.0	0.0	0

2.2 Data Overview

```
hb_new
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arriva
0	Resort Hotel	0	342	2015	July	

2.3 Descriptive Analysis

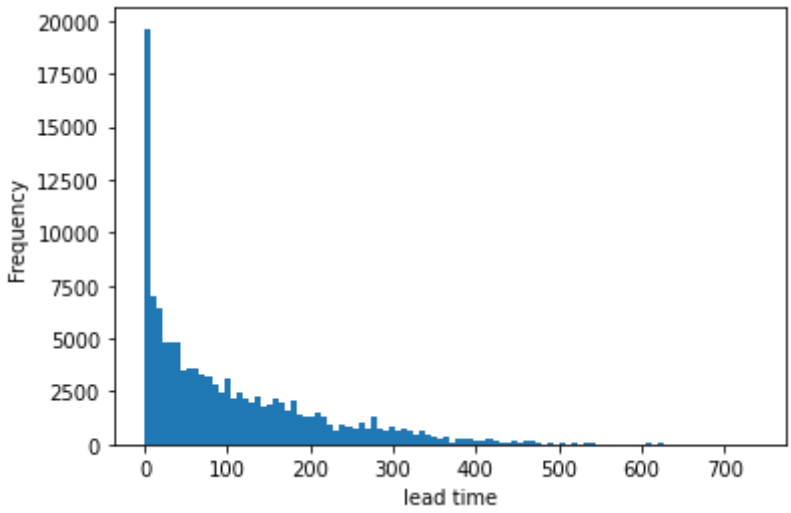
lead time

```
(pd.DataFrame(hb_new.lead_time)).describe()
```

	lead_time
count	119205.000000
mean	104.110801
std	106.876568
min	0.000000
25%	18.000000
50%	69.000000
75%	161.000000
max	737.000000

119389	Hotel	0	205	2017	August

```
from matplotlib import pyplot
lead_time=hb_new.lead_time
def drawHist(lead_time):
    pyplot.hist(hb_new.lead_time, 100)
    pyplot.xlabel('lead time')
    pyplot.ylabel('Frequency')
    pyplot.show()
drawHist(lead_time)
```

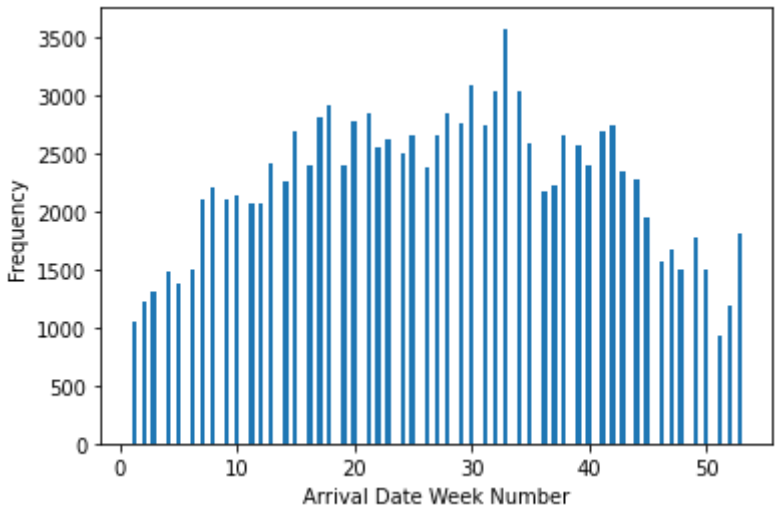


arrival date week number

```
(pd.DataFrame(hb_new.arrival_date_week_number)).describe()
```

	arrival_date_week_number
count	119205.000000
mean	27.163701
std	13.600991
min	1.000000
25%	16.000000
50%	28.000000
75%	38.000000
max	53.000000

```
from matplotlib import pyplot
arrival_date_week_number=hb_new.arrival_date_week_number
def drawHist(arrival_date_week_number):
    pyplot.hist(hb_new.arrival_date_week_number, 125)
    pyplot.xlabel('Arrival Date Week Number')
    pyplot.ylabel('Frequency')
    pyplot.show()
drawHist(arrival_date_week_number)
```

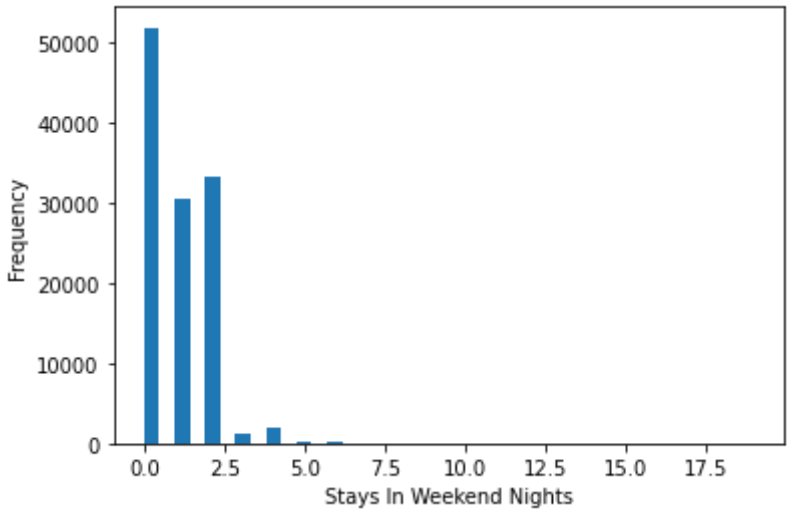


stays in weekend nights

```
(pd.DataFrame(hb_new.stays_in_weekend_nights)).describe()
```

	stays_in_weekend_nights
count	119205.000000
mean	0.927008
std	0.995046
min	0.000000

```
from matplotlib import pyplot
stays_in_weekend_nights=hb_new.stays_in_weekend_nights
def drawHist(stays_in_weekend_nights):
    pyplot.hist(hb_new.stays_in_weekend_nights, 40)
    pyplot.xlabel('Stays In Weekend Nights')
    pyplot.ylabel('Frequency')
    pyplot.show()
drawHist(stays_in_weekend_nights)
```



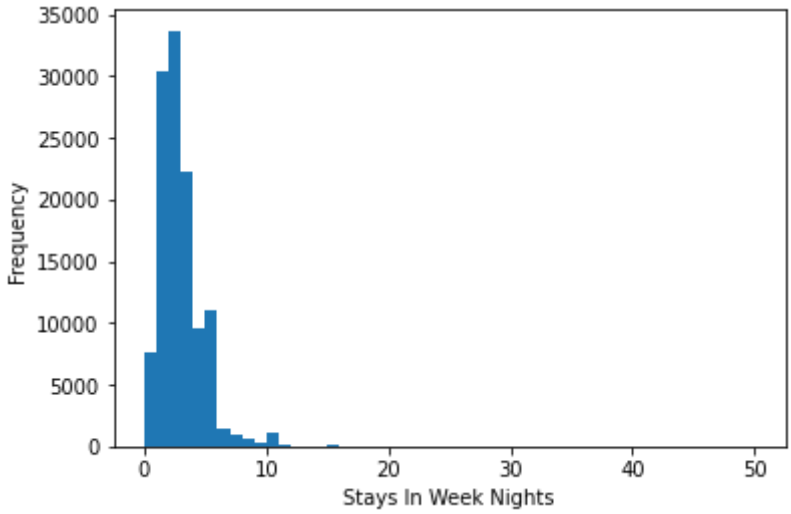
stays in week nights

```
(pd.DataFrame(hb_new.stays_in_week_nights)).describe()
```

	stays_in_week_nights
count	119205.000000
mean	2.499132
std	1.896984
min	0.000000
25%	1.000000
50%	2.000000
75%	3.000000
max	50.000000

```
from matplotlib import pyplot
stays_in_week_nights=hb_new.stays_in_week_nights
```

```
stays_in_week_nights=hb_new.stays_in_week_nights
def drawHist(stays_in_week_nights):
    pyplot.hist(hb_new.stays_in_week_nights, 50)
    pyplot.xlabel('Stays In Week Nights')
    pyplot.ylabel('Frequency')
    pyplot.show()
drawHist(stays_in_week_nights)
```

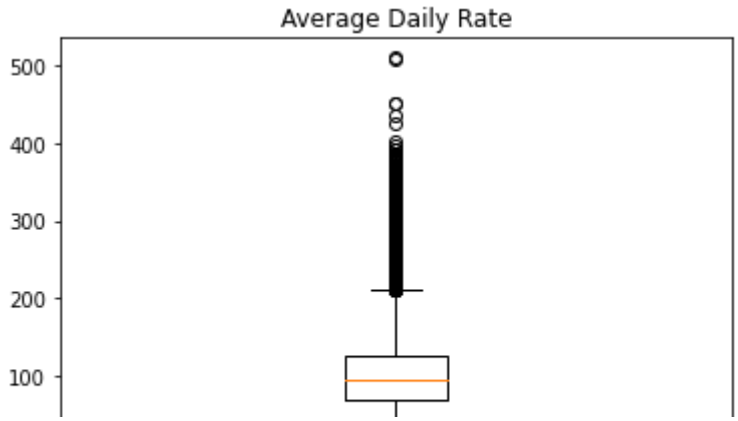


Average Daily Rate

```
(pd.DataFrame(hb.adr)).describe()
```

	adr
count	119390.000000
mean	101.831122
std	50.535790
min	-6.380000
25%	69.290000
50%	94.575000
75%	126.000000
max	5400.000000

```
from matplotlib import pyplot
adr=hb.adr
def drawBox(adr):
    pyplot.boxplot([hb_new.adr], labels=['Rating'])
    pyplot.title('Average Daily Rate')
    pyplot.show()
drawBox(adr)
```

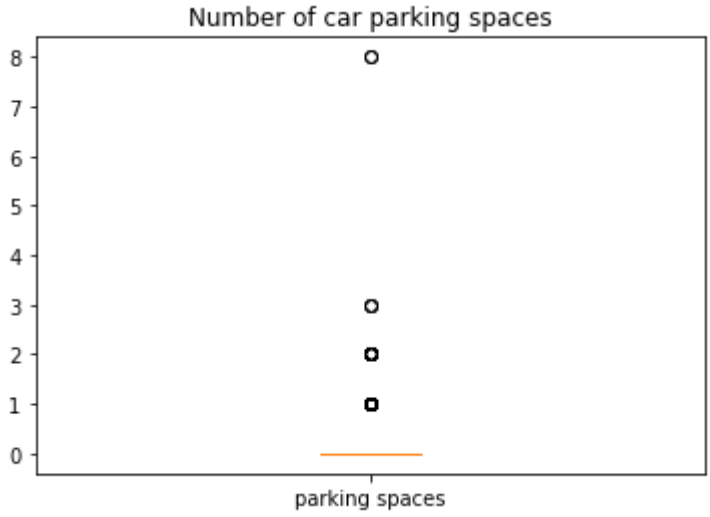


Number of car parking spaces required by the customer

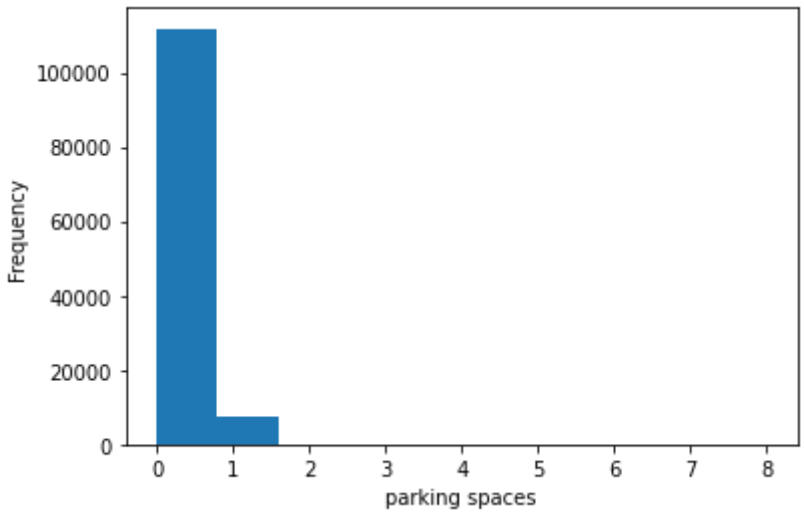
```
(pd.DataFrame(hb_new.required_car_parking_spaces)).describe()
```

	required_car_parking_spaces
count	119205.000000
mean	0.062556
std	0.245364
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	8.000000

```
from matplotlib import pyplot
required_car_parking_spaces=hb_new.required_car_parking_spaces
def drawBox(required_car_parking_spaces):
    pyplot.boxplot([hb_new.required_car_parking_spaces], labels=['parking spaces'])
    pyplot.title('Number of car parking spaces')
    pyplot.show()
drawBox(required_car_parking_spaces)
```



```
from matplotlib import pyplot
required_car_parking_spaces=hb_new.required_car_parking_spaces
def drawHist(required_car_parking_spaces):
    pyplot.hist(hb_new.required_car_parking_spaces, 10)
    pyplot.xlabel('parking spaces')
    pyplot.ylabel('Frequency')
    pyplot.show()
drawHist(required_car_parking_spaces)
```

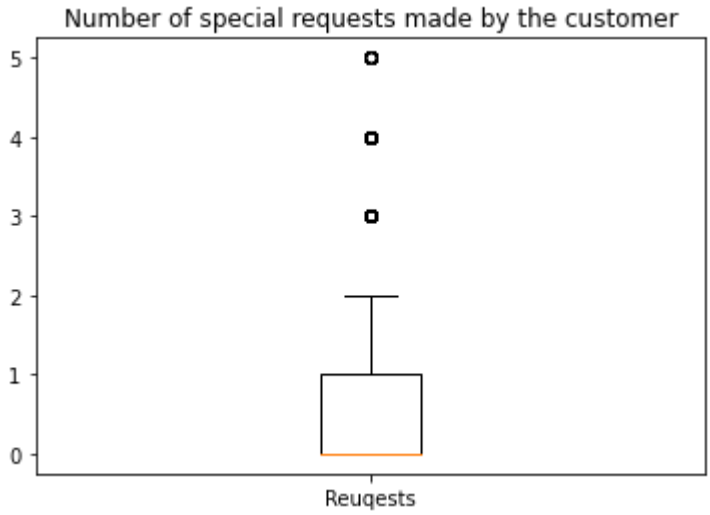


Number of special requests made by the customer

```
(pd.DataFrame(hb_new.total_of_special_requests)).describe()
```

	total_of_special_requests
count	119205.000000
mean	0.571511
std	0.792885
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	5.000000

```
from matplotlib import pyplot
total_of_special_requests=hb_new.total_of_special_requests
def drawBox(total_of_special_requests):
    pyplot.boxplot([hb_new.total_of_special_requests], labels=['Reuqests'])
    pyplot.title('Number of special requests made by the customer')
    pyplot.show()
drawBox(total_of_special_requests)
```



Visualization

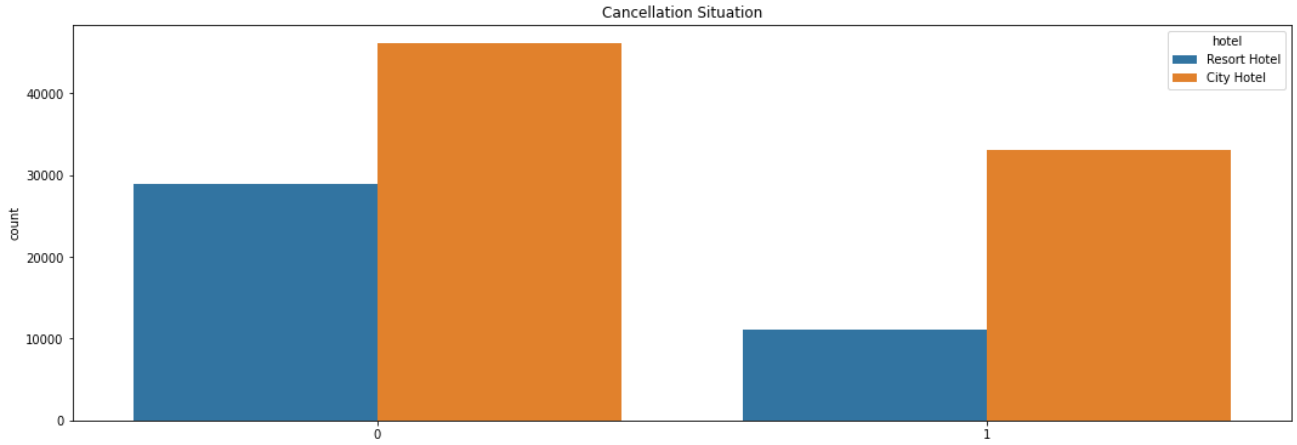
```
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
import plotly
import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import iplot, init_notebook_mode
import cufflinks as cf
```

Cancellation Situation

```
hotel_eda = hb_new.copy()
hotel_eda['hotel'].value_counts()
```

```
City Hotel      79160
Resort Hotel    40045
Name: hotel, dtype: int64
```

```
plt.figure(figsize=(18,6))
sns.countplot(x='is_canceled', hue = 'hotel', data= hotel_eda)
plt.title('Cancellation Situation')
plt.show()
```




```
hotel_eda['is_canceled'].value_counts()/hb_new.shape[0]*100
```

```
0    62.923535
1    37.076465
Name: is_canceled, dtype: float64
```

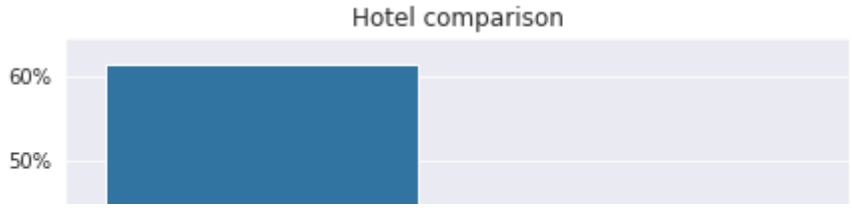
The cancel rate of the hotel is 37.04%, the rate of not cancel is 62.96%.

Booking rate

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mtick


def get_count(series, limit=None):

    '''
    INPUT:
        series: Pandas Series (Single Column from DataFrame)
        limit:  If value given, limit the output value to first limit samples.
    OUTPUT:
        x = Unique values
        y = Count of unique values
    '''

    if limit != None:
        series = series.value_counts()[:limit]
    else:
        series = series.value_counts()

    x = series.index
    y = series/series.sum()*100

    return x.values,y.values



def plot(x, y, x_label=None,y_label=None, title=None, figsize=(7,5), type='bar'):

    '''
```

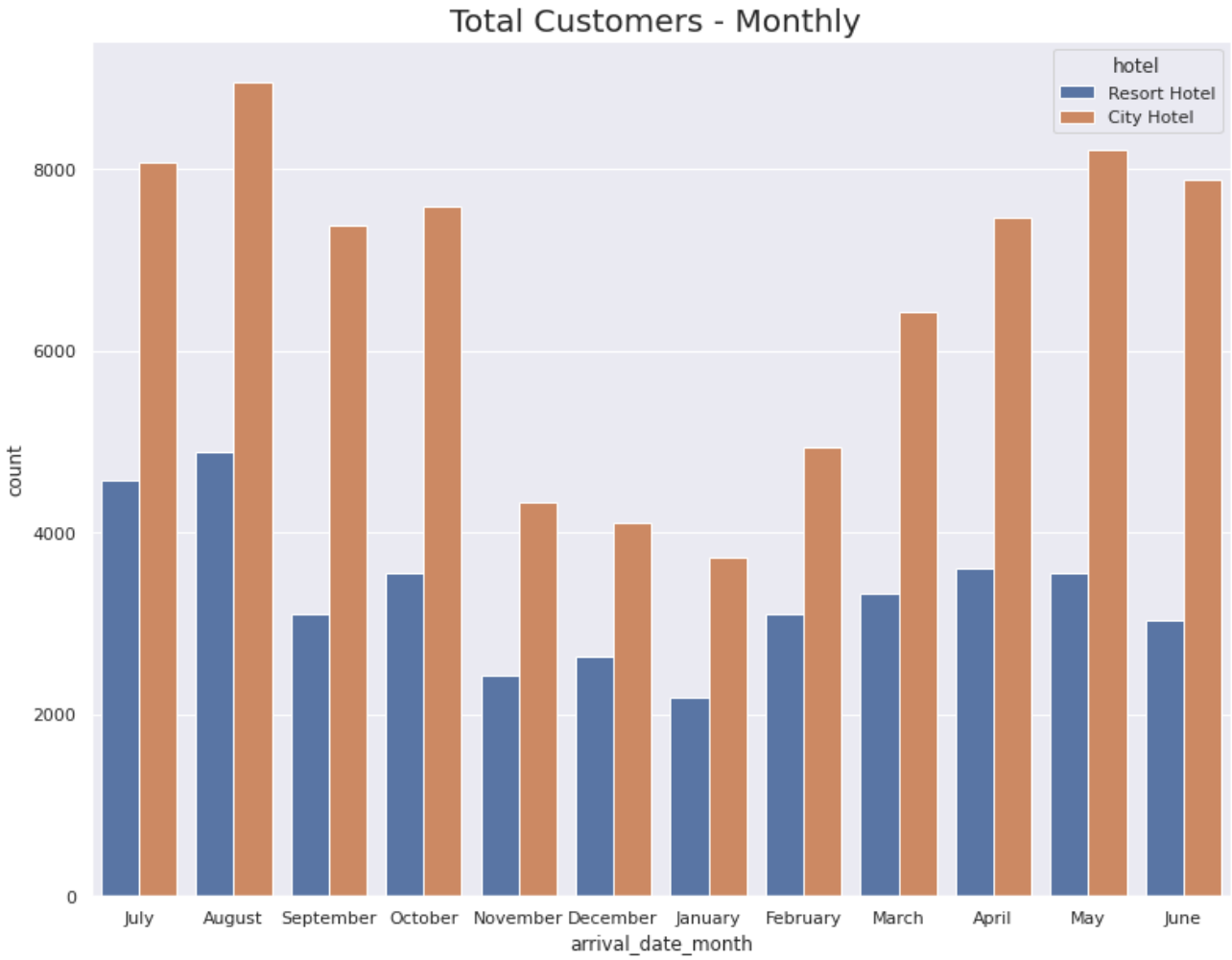
```
    '''
    INPUT:
        x:        Array containing values for x-axis
        y:        Array containing values for y-axis
        x_lable:  String value for x-axis label
        y_lable:  String value for y-axis label
        title:    String value for plot title
        figsize:  tuple value, for figure size
        type:     type of plot (default is bar plot)

    OUTPUT:
        Display the plot
    '''


    sns.set_style('darkgrid')

    fig, ax = plt.subplots(figsize=figsize)

    ax.yaxis.set_major_formatter(mtick.PercentFormatter())

    if x_label != None:
        ax.set_xlabel(x_label)

    if y_label != None:
        ax.set_ylabel(y_label)

    if title != None:
        ax.set_title(title)

    if type == 'bar':
        sns.barplot(x,y, ax = ax)
    elif type == 'line':
        sns.lineplot(x,y, ax = ax, sort=False)


    plt.show()


df_not_canceled = hb_new[hb_new['is_canceled'] == 0]
x,y = get_count(df_not_canceled['hotel'])
plot(x,y, x_label='Hotels', y_label='Total Booking (%)', title='Hotel comparison')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valic
```



Monthly cancellations and customer by hotel types

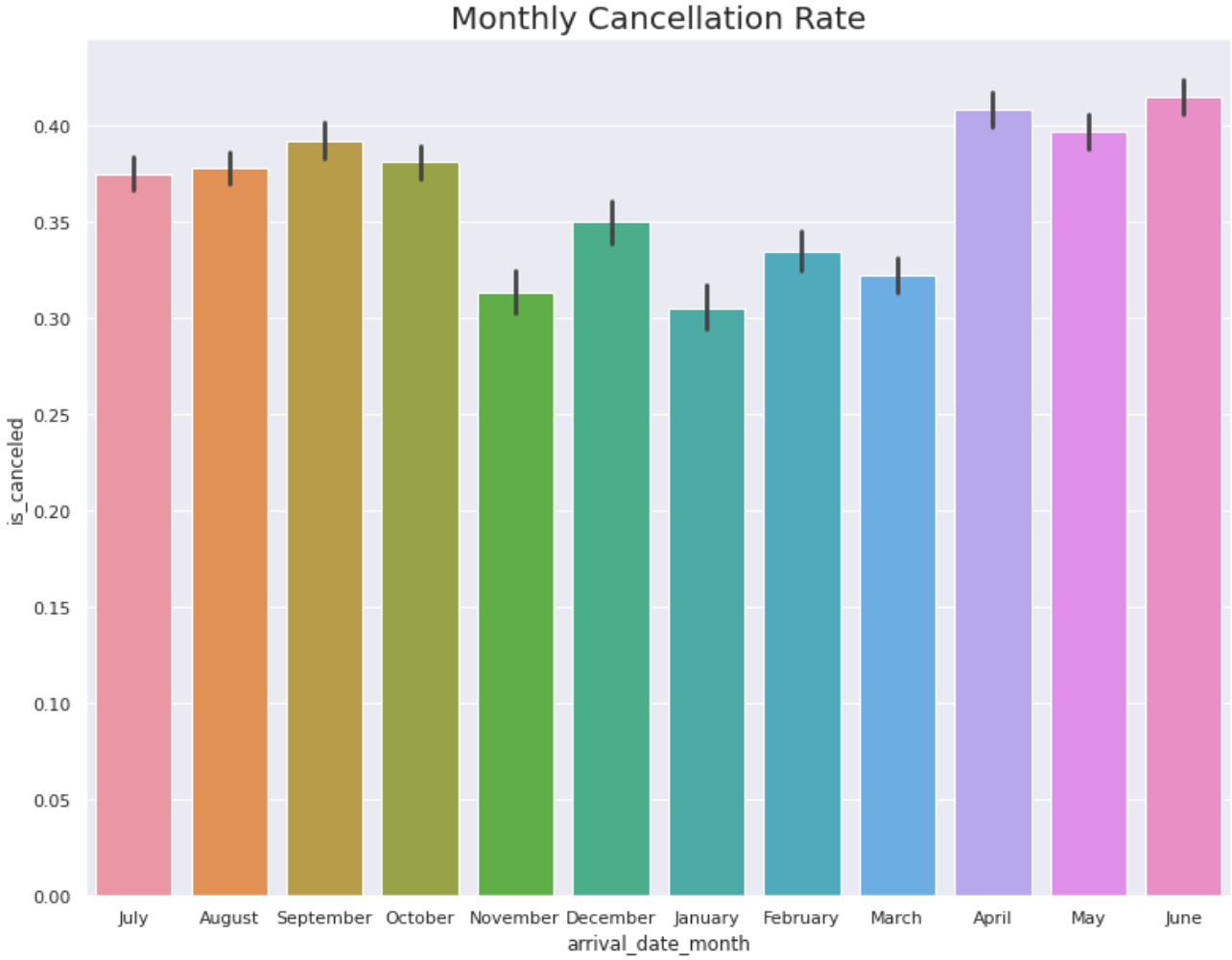

```
plt.figure(figsize =(13,10))
sns.set(style="darkgrid")
plt.title("Total Customers - Monthly ", fontdict={'fontsize': 20})
ax = sns.countplot(x = "arrival_date_month", hue = 'hotel', data = hb_new)
```




```
plt.figure(figsize = (13,10))
sns.barplot(x = 'arrival_date_month', y = 'is_canceled', data = hb_new);
plt.title('Monthly Cancellation Rate', fontdict={'fontsize': 20})
```

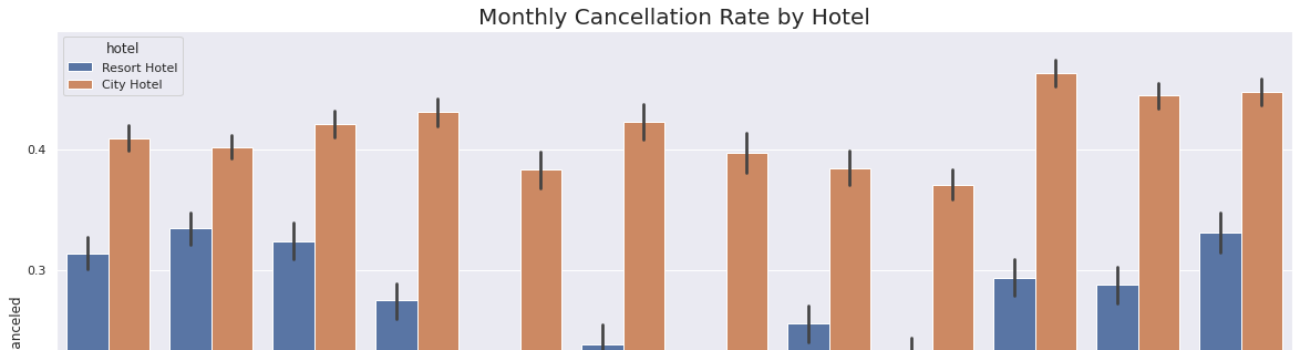
```
plt.title( Monthly Cancellation Rate , fontdict={ fontsize : 20})
```

```
Text(0.5, 1.0, 'Monthly Cancellation Rate')
```




```
plt.figure(figsize = (20,10))
sns.barplot(x = 'arrival_date_month', y = 'is_canceled', hue = 'hotel', data = hb_new);
plt.title('Monthly Cancellation Rate by Hotel', fontdict={'fontsize': 20})
```

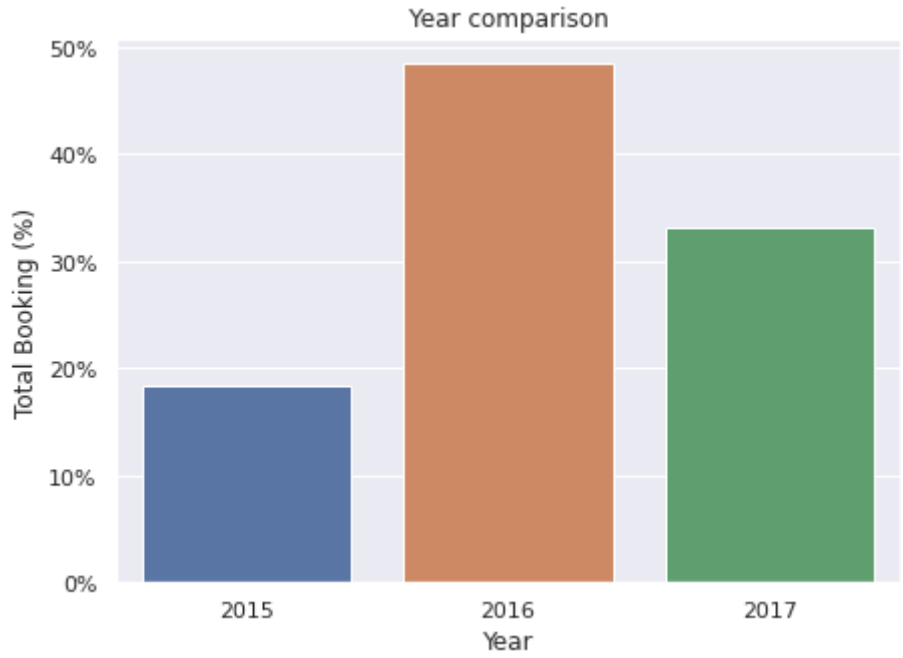
```
Text(0.5, 1.0, 'Monthly Cancellation Rate by Hotel')
```



Percentage of booking per year

```
x,y = get_count(df_not_canceled['arrival_date_year'])
plot(x,y, x_label='Year', y_label='Total Booking (%)', title='Year comparison')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valic
```



Busiest month

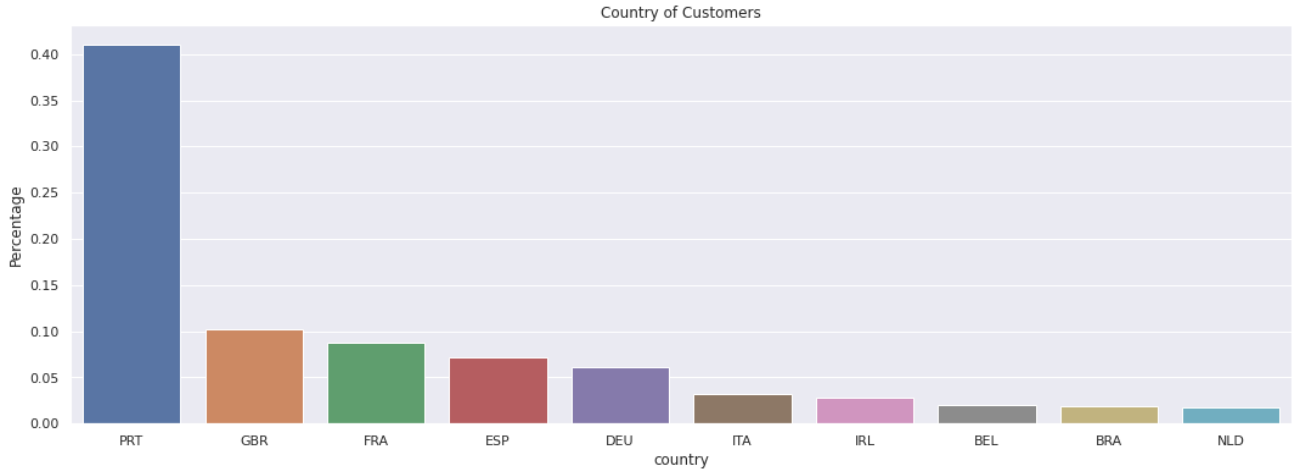
```
## Order of months
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'Se
             'November', 'December']


## Select only City Hotel
sorted_months = df_not_canceled.loc[hb_new.hotel=='City Hotel' ,'arrival_date_month'].valu

x1 = sorted_months.index
y1 = sorted_months/sorted_months.sum()*100
```

```
## Select only Resort Hotel
sorted_months = df_not_canceled.loc[hb_new.hotel=='Resort Hotel' ,'arrival_date_month'].va

x2 = sorted_months.index
y2 = sorted_months/sorted_months.sum()*100
```

```
## Draw the line plot

fig, ax = plt.subplots(figsize=(18,6))

ax.set_xlabel('Months')
ax.set_ylabel('Booking (%)')
ax.set_title('Booking Trend (Monthly)')


sns.lineplot(x1, y1.values, label='City Hotel', sort=False)
sns.lineplot(x1, y2.values, label='Resort Hotel', sort=False)

plt.show()
```

Country of customers

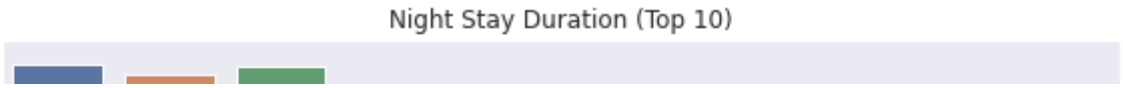
```
plt.figure(figsize=(18,6))
country_booking = hb_new['country'].value_counts(normalize=True).rename_axis('country').re
sns.barplot(x='country', y='Percentage', data=country_booking.head(10))
plt.title('Country of Customers')
plt.show()
```



How long people stay

```
total_nights = df_not_canceled['stays_in_weekend_nights']+ df_not_canceled['stays_in_week_
x,y = get_count(total_nights, limit=10)

plot(x,y, x_label='Number of Nights', y_label='Booking Percentage (%)', title='Night Stay
```

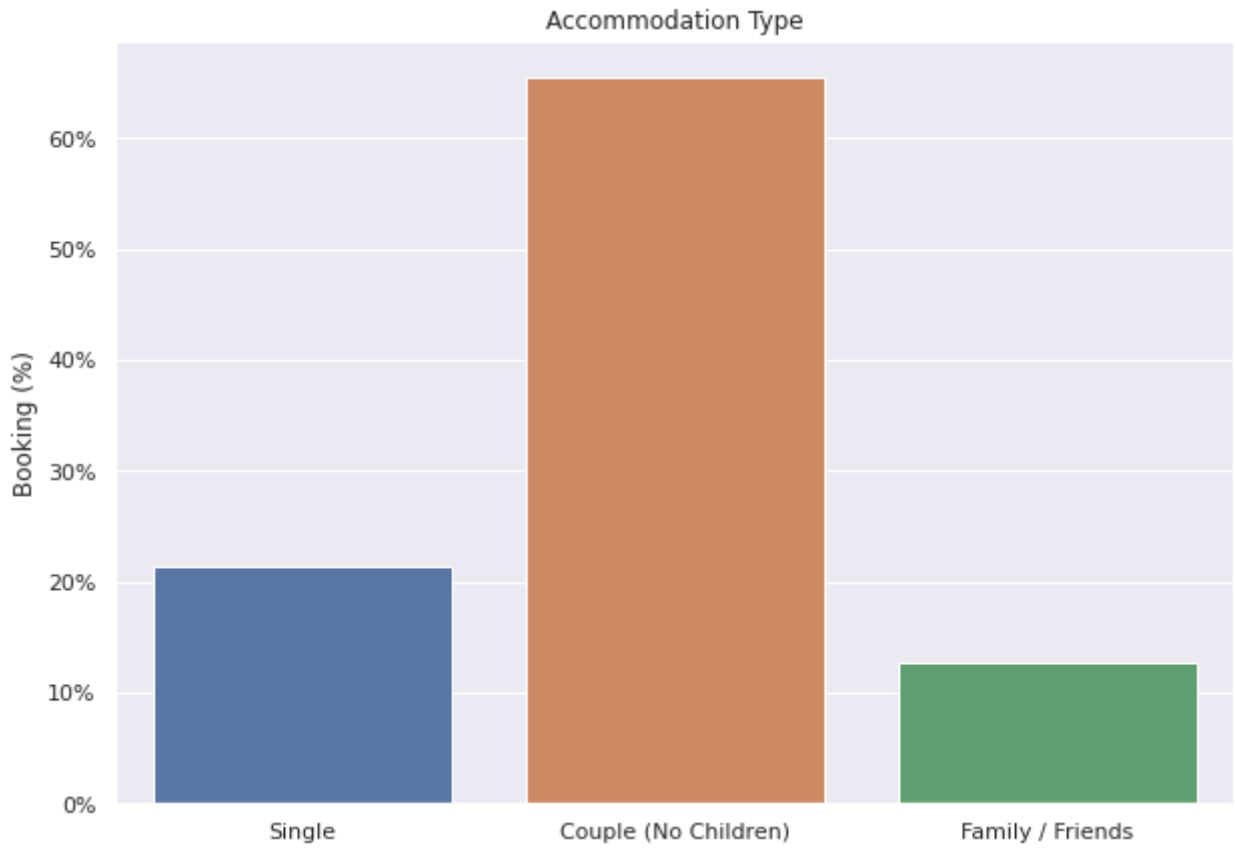
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valic
```



Accomodation type

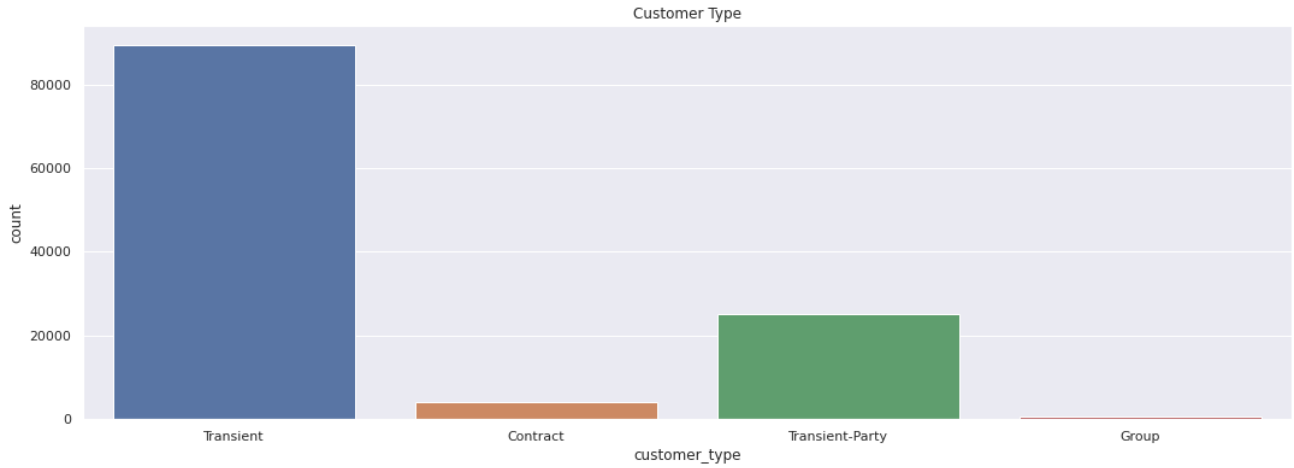
```
## Select single, couple, multiple adults and family
single   = df_not_canceled[(df_not_canceled.adults==1) & (df_not_canceled.children==0) & (
couple   = df_not_canceled[(df_not_canceled.adults==2) & (df_not_canceled.children==0) & (
family   = df_not_canceled[df_not_canceled.adults + df_not_canceled.children + df_not_canc


## Make the list of Category names, and their total percentage
names = ['Single', 'Couple (No Children)', 'Family / Friends']
count = [single.shape[0],couple.shape[0], family.shape[0]]
count_percent = [x/df_not_canceled.shape[0]*100 for x in count]


## Draw the curve
plot(names,count_percent,  y_label='Booking (%)', title='Accommodation Type', figsize=(10,
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valic
```



Customer type

```
plt.figure(figsize=(18,6))
sns.countplot(x='customer_type', data= hb_new)
plt.title('Customer Type')
plt.show()
```



Type of booking, assuming one of four categories:

Contract, when the booking has an allotment or other type of contract associated to it.

Group, when the booking is associated to a group.

Transient, when the booking is not part of a group or contract, and is not associated to other transient booking.

Transient-party, when the booking is transient, but is associated to at least other transient booking.

From the graph:

Transient as much as 75.05%.

Transient-party as much as 21.04%.

Contract as much as 3.41%.

Group as much as 0.48%.

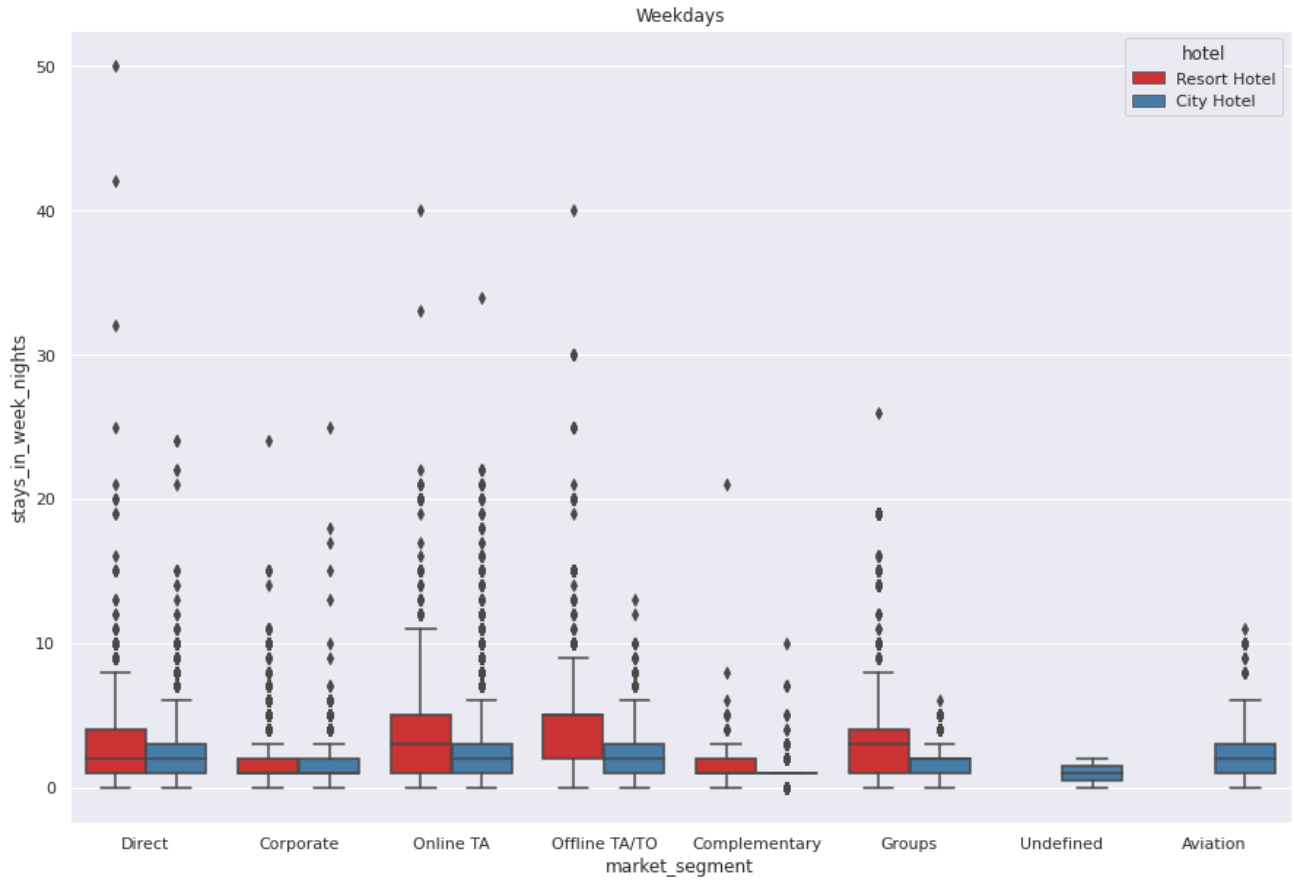
Most of customers is Transient, means they are walk-in guests, last-minute or bookers, or simply people that require a very short-term stay in your facility. Transient customers are one of the major market segments consist of individuals or groups

Distribution of nights spent at hotels by market segment and hotel type

weekdays

```
plt.figure(figsize = (15,10))
sns.boxplot(x = "market_segment", y = "stays_in_week_nights", data = hb_new, hue = "hotel"
plt.title('Weekdays')
```

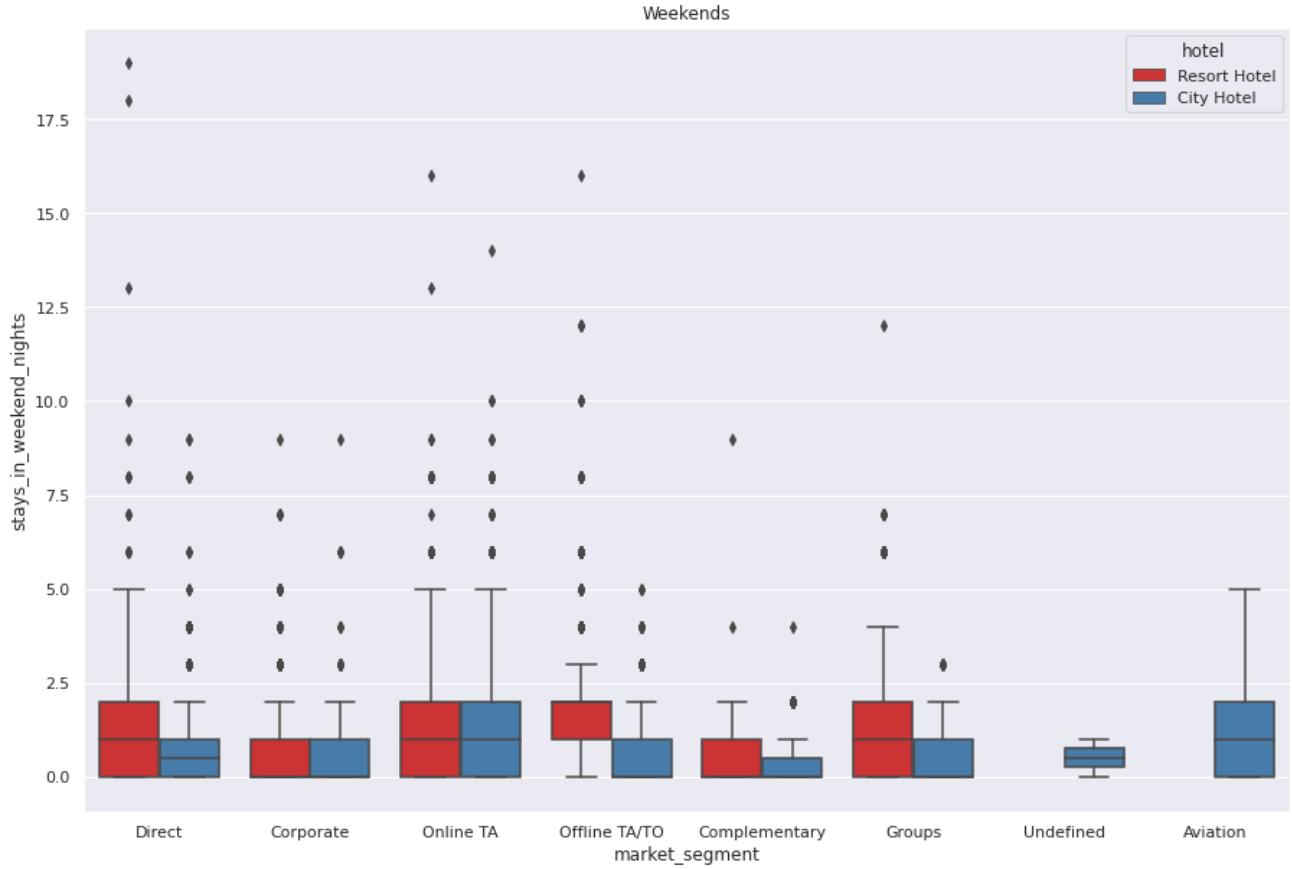
```
Text(0.5, 1.0, 'Weekdays')
```



weekends

```
plt.figure(figsize=(15,10))
sns.boxplot(x = "market_segment", y = "stays_in_weekend_nights", data = hb_new, hue = "hot
plt.title('Weekends')
```

```
Text(0.5, 1.0, 'Weekends')
```



Feature engineering

```
hb_new['hotel'] = hb_new['hotel'].map({'Resort Hotel':0, 'City Hotel':1})

hb_new['arrival_date_month'] = hb_new['arrival_date_month'].map({'January':1, 'February':
                                                                 'August':8, 'September':9, 'Oc


def family(hb_new):
    if ((hb_new['adults'] > 0) & (hb_new['children'] > 0)):
        val = 1
    elif ((hb_new['adults'] > 0) & (hb_new['babies'] > 0)):
        val = 1
    else:
        val = 0
    return val

def deposit(hb_new):
    if ((hb_new['deposit_type'] == 'No Deposit') | (hb_new['deposit_type'] == 'Refundable'
        return 0
    else:
        return 1
```

```
def feature(hb_new):
    hb_new["is_family"] = hb_new.apply(family, axis = 1)
    hb_new["total_customer"] = hb_new["adults"] + hb_new["children"] + hb_new["babies"]
    hb_new["deposit_given"] = hb_new.apply(deposit, axis=1)
    hb_new["total_nights"] = hb_new["stays_in_weekend_nights"]+ hb_new["stays_in_week_nigh
    return hb_new

hb_new = feature(hb_new)


hb_new = hb_new.drop(columns = ['adults', 'babies', 'children', 'deposit_type', 'reservati


cor_data = hb_new.copy()


from sklearn.preprocessing import LabelEncoder, StandardScaler
le = LabelEncoder()
cor_data['meal'] = le.fit_transform(cor_data['meal'])
cor_data['distribution_channel'] = le.fit_transform(cor_data['distribution_channel'])
cor_data['reserved_room_type'] = le.fit_transform(cor_data['reserved_room_type'])
cor_data['assigned_room_type'] = le.fit_transform(cor_data['assigned_room_type'])
cor_data['agent'] = le.fit_transform(cor_data['agent'])
cor_data['customer_type'] = le.fit_transform(cor_data['customer_type'])
cor_data['reservation_status'] = le.fit_transform(cor_data['reservation_status'])
cor_data['market_segment'] = le.fit_transform(cor_data['market_segment'])


cor_data.corr()
```

	hotel	is_canceled	lead_time	arrival_date_year
hotel	1.000000	0.137096	0.075990	0.035185
is_canceled	0.137096	1.000000	0.292893	0.016638
lead_time	0.075990	0.292893	1.000000	0.040307
arrival_date_year	0.035185	0.016638	0.040307	1.000000
arrival_date_month	0.001770	0.011179	0.131609	-0.527545
arrival_date_week_number	0.001239	0.008313	0.127053	-0.540381
arrival_date_day_of_month	-0.001698	-0.005969	0.002313	-0.000121
stays_in_weekend_nights	-0.187761	-0.001326	0.085982	0.021719
stays_in_week_nights	-0.235901	0.025516	0.166899	0.031230
meal	0.007276	-0.017226	0.001413	0.065639
market_segment	0.084117	0.059407	0.013271	0.107908
distribution_channel	0.174980	0.167682	0.220225	0.022643
is_repeated_guest	-0.052468	-0.083721	-0.123270	0.010222
previous_cancellations	-0.012262	0.110142	0.086025	-0.119911
previous_bookings_not_canceled	-0.004452	-0.057358	-0.073612	0.029220

```
cor_data.corr()["is_canceled"].sort_values(ascending = False)
```

```
is_canceled                      1.000000
deposit_given                    0.481501
lead_time                        0.292893
distribution_channel             0.167682
hotel                            0.137096
previous_cancellations           0.110142
market_segment                   0.059407
days_in_waiting_list             0.054303
adr                              0.047578
total_customer                   0.044933
stays_in_week_nights             0.025516
total_nights                     0.018533
arrival_date_year                0.016638
arrival_date_month               0.011179
arrival_date_week_number         0.008313
stays_in_weekend_nights         -0.001326
arrival_date_day_of_month       -0.005969
is_family                       -0.013220
meal                            -0.017226
agent                           -0.046994
previous_bookings_not_canceled  -0.057358
reserved_room_type              -0.062225
customer_type                   -0.068152
is_repeated_guest               -0.083721
booking_changes                 -0.144857
assigned_room_type              -0.175833
required_car_parking_spaces     -0.195705
total_of_special_requests       -0.234888
```

```
reservation_status               -0.917239
Name: is_canceled, dtype: float64
```

Correlation heatmap

```
sns.set(rc={'figure.figsize':(20,15)})
corr = cor_data.corr()# plot the heatmap
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cmap=sns.diverging_p
```

```
cor_data = cor_data.drop(columns = ['total_nights', 'arrival_date_week_number', 'stays_in_
```

```
indices = cor_data.loc[pd.isna(cor_data["country"]), :].index
cor_data = cor_data.drop(cor_data.index[indices])
cor_data.isnull().sum()
```

```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_day_of_month       0
stays_in_week_nights            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
days_in_waiting_list            0
customer_type                   0
adr                             0
required_car_parking_spaces     0
total_of_special_requests       0
reservation_status              0
is_family                       0
total_customer                  0
deposit_given                   0
dtype: int64
```

```
indices = hb_new.loc[pd.isna(hb_new["country"]), :].index
hb_new = hb_new.drop(hb_new.index[indices])
hb_new = hb_new.drop(columns = ['arrival_date_week_number', 'stays_in_weekend_nights', 'ar
```

One-hot encoding

```
df1 = hb_new.copy()
```

```
df1 = pd.get_dummies(data = df1, columns = ['meal', 'market_segment', 'distribution_channe
                                            'reserved_room_type', 'assigned_room_type', 'c
```

```
df1['country'] = le.fit_transform(df1['country'])
```

Logistic regression

```
df2 = df1.drop(columns = ['reservation_status_Canceled', 'reservation_status_Check-Out', '
```

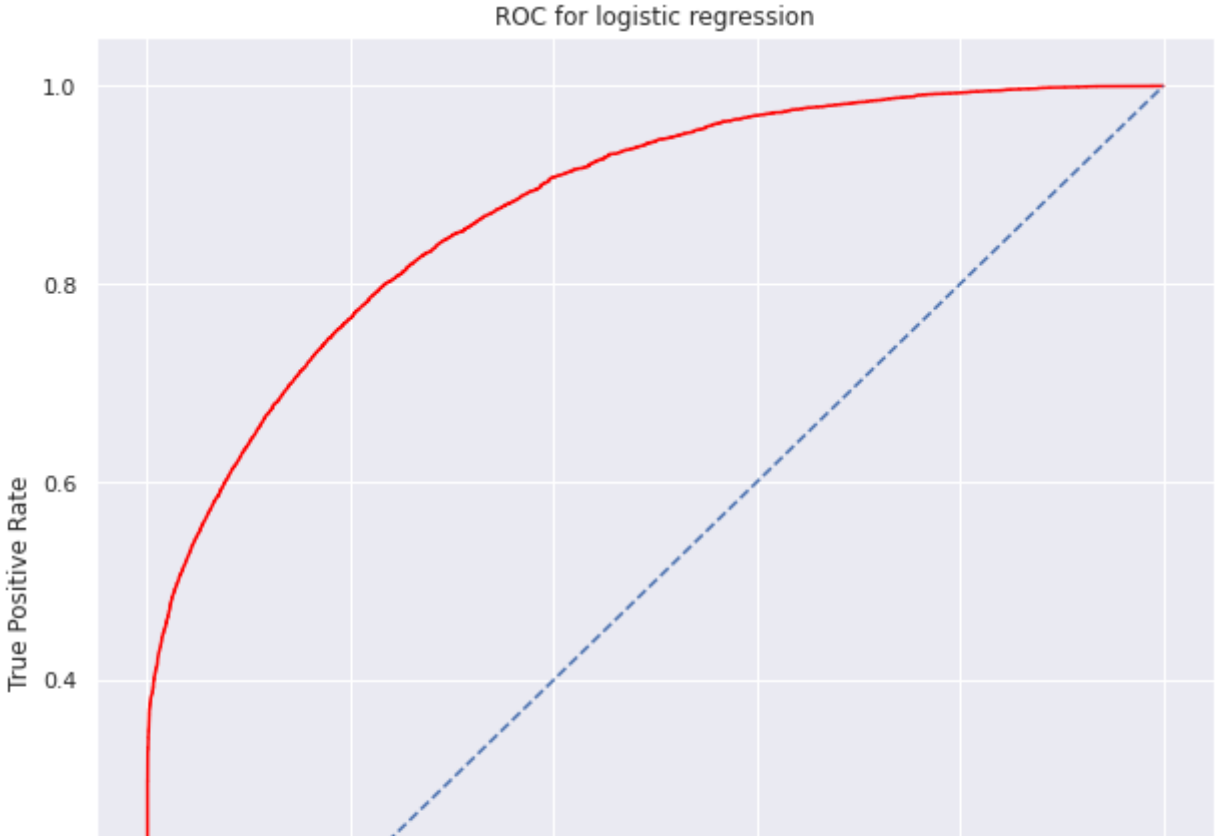
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, confusion_matrix, au
y = df2["is_canceled"]
X = df2.drop(["is_canceled"], axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state =


def model(algorithm, X_train, X_test, y_train, y_test):
    alg = algorithm
    alg_model = alg.fit(X_train, y_train)
    global y_prob, y_pred
    y_prob = alg.predict_proba(X_test)[:,1]
    y_pred = alg_model.predict(X_test)

    print('Accuracy Score: {}\n\nConfusion Matrix:\n {}'
      .format(accuracy_score(y_test,y_pred), confusion_matrix(y_test,y_pred)))


def ROC(y_test, y_prob):

    false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_prob)
    roc_auc = auc(false_positive_rate, true_positive_rate)

    plt.figure(figsize = (10,10))
    plt.title('Receiver Operating Characteristic')
    plt.plot(false_positive_rate, true_positive_rate, color = 'red', label = 'AUC = %0.2f'
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1], linestyle = '--')
    plt.axis('tight')
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')


from sklearn.linear_model import LogisticRegression

print('Model: Logistic Regression\n')
model(LogisticRegression(solver = "liblinear"), X_train, X_test, y_train, y_test)
```

```
Model: Logistic Regression

Accuracy Score: 0.8024998601867904

Confusion Matrix:
 [[20537  1841]
 [ 5222  8162]]
```

```
ROC(y_test, y_prob)
plt.title('ROC for logistic regression')
```

```
Text(0.5, 1.0, 'ROC for logistic regression')
```



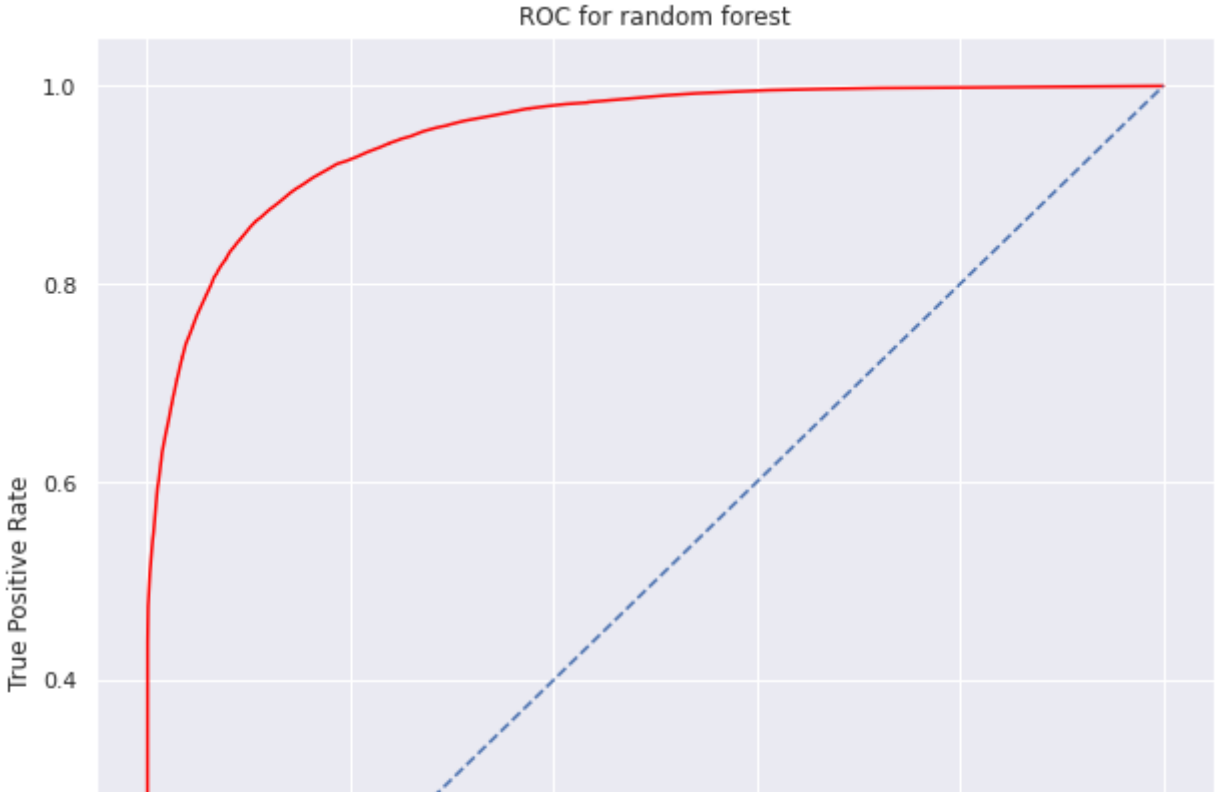
Random forest

```
from sklearn.ensemble import RandomForestClassifier
print('Model: Random Forest\n')
model(RandomForestClassifier(), X_train, X_test, y_train, y_test)
```

```
Model: Random Forest

Accuracy Score: 0.8855489066607013

Confusion Matrix:
 [[20962  1416]
 [ 2677 10707]]
```

```
ROC(y_test, y_prob)
plt.title('ROC for random forest')
```

```
Text(0.5, 1.0, 'ROC for random forest')
```



Feature Importances

```
randomf = RandomForestClassifier()
rf_model1 = randomf.fit(X_train, y_train)

pd.DataFrame(data = rf_model1.feature_importances_*100,
                   columns = ["Importances"],
                   index = X_train.columns).sort_values("Importances", ascending = False)[

plt.xlabel("Feature Importances (%)")
```

```
Text(0.5, 0, 'Feature Importances (%)')
```