HOTEL BOOKING CANCELLATION PREDICTION

GOAL OF THE PROJECT

In this project, we aim to build a predictive model to determine whether a hotel booking would be canceled or not, which is crucial for hotels as cancellations affect revenue and operational planning. The dataset contains a high number of features related to booking, such as lead time, deposit type, and different countries, which adds to the complexity of the model. The challenge lies in the data preprocessing steps, which include feature selection and engineering, handling missing values, and noise in the data. Additionally, we are going to train different models, evaluate their performance using the right metrics, and interpret the model by analyzing the most important features in the context of hotel booking cancellations. This is classification problem.

IMPORT LIBRARIES

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly
import chart_studio.plotly as py
from plotly.offline import download_plotlyjs ,init_notebook_mode ,plot ,ipl
init_notebook_mode(connected=True) #visualize in offline mode or in noteboo
import plotly.express as px
import sort_dataframeby_monthorweek as sd

import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

IMPORT DATASET

In [2]:
```
df=pd.read_csv(r"D:\IMARTICUS\ml project\hotel_booking_mlpro\hotel_bookings
```

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

Out[3]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nu
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	

3 rows × 32 columns

In [4]:
```
df.shape
```

Out[4]:
```
(119390, 32)
```

In [5]:
```
df.is_canceled.value_counts()
```

Out[5]:
```
0    75166
1    44224
Name: is_canceled, dtype: int64
```

DATA CLEANING

In [6]:
```
df.isnull().sum()[df.isnull().sum()>0]
```

Out[6]:
```
children         4
country        488
agent        16340
company     112593
dtype: int64
```

In [7]:
```
df.drop(['agent','company'],axis=1,inplace=True)

# agent feture is id of travel agency that have made booking
# company is id on entity who did booking
```

In [8]:
```
df.country.value_counts()
```

Out[8]:
```
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
```

In [9]:
```
df.country.fillna('PRT', inplace=True)
df.fillna(0,inplace=True)
```

In [10]:
```
df.isnull().sum()[df.isnull().sum()>0]
```

Out[10]:
```
Series([], dtype: int64)
```

In [11]:
```
### seems to have some dirtiness in data as Adults,babies & children cant b

### bcz if 3 entities are 0 ,then how can a booking be possible ??
```

In [12]:
```
### Visualise Entire Dataframe where adult,children & babies are 0

filter1=(df['children']==0) & (df['adults']==0) & (df['babies']==0)
```

In [13]:
```
df[filter1]
```

Out[13]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_we
2224	Resort Hotel	0	1	2015	October	
2409	Resort Hotel	0	0	2015	October	
3181	Resort Hotel	0	36	2015	November	
3684	Resort Hotel	0	165	2015	December	
3708	Resort Hotel	0	165	2015	December	
...	...	...	...	...	...	
115029	City Hotel	0	107	2017	June	
115091	City Hotel	0	1	2017	June	
116251	City Hotel	0	44	2017	July	
116534	City Hotel	0	2	2017	July	
117087	City Hotel	0	170	2017	July	

180 rows × 30 columns

In [14]:
```
#data=df[filter1== False]
data=df[~filter1] # ~ is nagation
#it return all rows upper condition is not true
```

In [15]:
```
data.shape
```

Out[15]:
```
(119210, 30)
```

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

Out[16]:
```
(119390, 30)
```

In [17]:
```
data.arrival_date_year.unique()
```

Out[17]:
```
array([2015, 2016, 2017], dtype=int64)
```

EXPLORATORY DATA ANALYSIS

1. Where do the guests come from ?

In [18]:
```
## Lets perform Spatial Analysis
```

In [19]:
```
data['is_canceled'].unique()
# 0-- Booking Doesnt Cancelled
# 1--Customer is cancelled booking
```

Out[19]:
```
array([0, 1], dtype=int64)
```

In [20]:
```
data[data['is_canceled']==0]['country'].value_counts()/75011
# there is apperend diff booking location
```

Out[20]:
```
PRT    0.285265
GBR    0.128888
FRA    0.112890
ESP    0.085094
DEU    0.080881
         ...
BHR    0.000013
DJI    0.000013
MLI    0.000013
NPL    0.000013
FRO    0.000013
Name: country, Length: 165, dtype: float64
```

In [21]:
```
len(data[data['is_canceled']==0])
```

Out[21]:
```
75011
```

In [22]:
```
country_wise_data=data[data['is_canceled']==0]['country'].value_counts().re
country_wise_data.columns=['country','no_of_guests']
country_wise_data
```

Out[22]:

	country	no_of_guests
0	PRT	21398
1	GBR	9668
2	FRA	8468
3	ESP	6383
4	DEU	6067
...	...	...
160	BHR	1
161	DJI	1
162	MLI	1
163	NPL	1
164	FRO	1

165 rows × 2 columns

In [23]:
```
#!pip install plotly
```

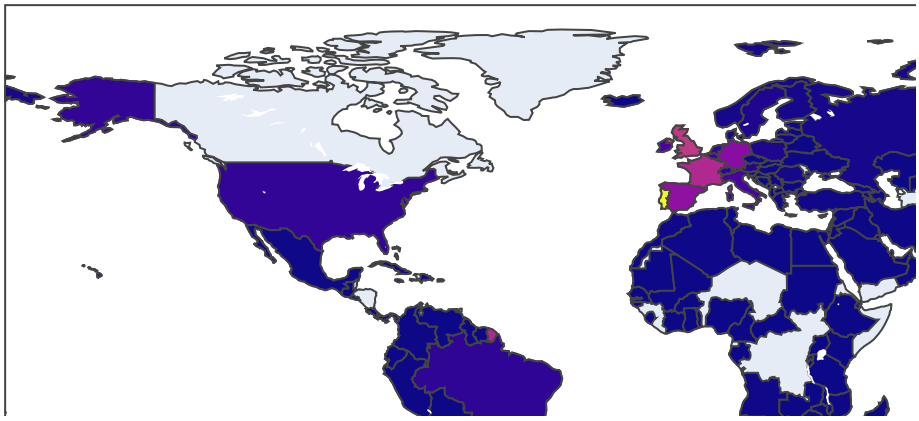
In [24]:
```
#!pip install chart_studio
```

In [25]:
```
import plotly
import chart_studio.plotly as py
from plotly.offline import download_plotlyjs ,init_notebook_mode ,plot ,ipl
init_notebook_mode(connected=True) #visualize in offline mode or in noteboo
import plotly.express as px
```

In [26]:
```
map_guest=px.choropleth(country_wise_data,
             locations=country_wise_data['country'],
             color=country_wise_data['no_of_guests'],
              hover_name=country_wise_data['country'],
              title='Home country of guests'
)
```

In [27]:
```
map_guest.show()
```

Home country of guests



```
Conclusion : People from all over the world are staying in these two
hotels. Most guests are from Portugal and other countries in Europe
```

2. How much do guests pay for a room per night ?

```
Both hotels have different room types and different meal arrangements.
Seasonal factors are also important. So the prices vary a lot. Since no
currency information is given, but Portugal is part of the European
Monetary Union, I assume that all prices are in EUR.
```

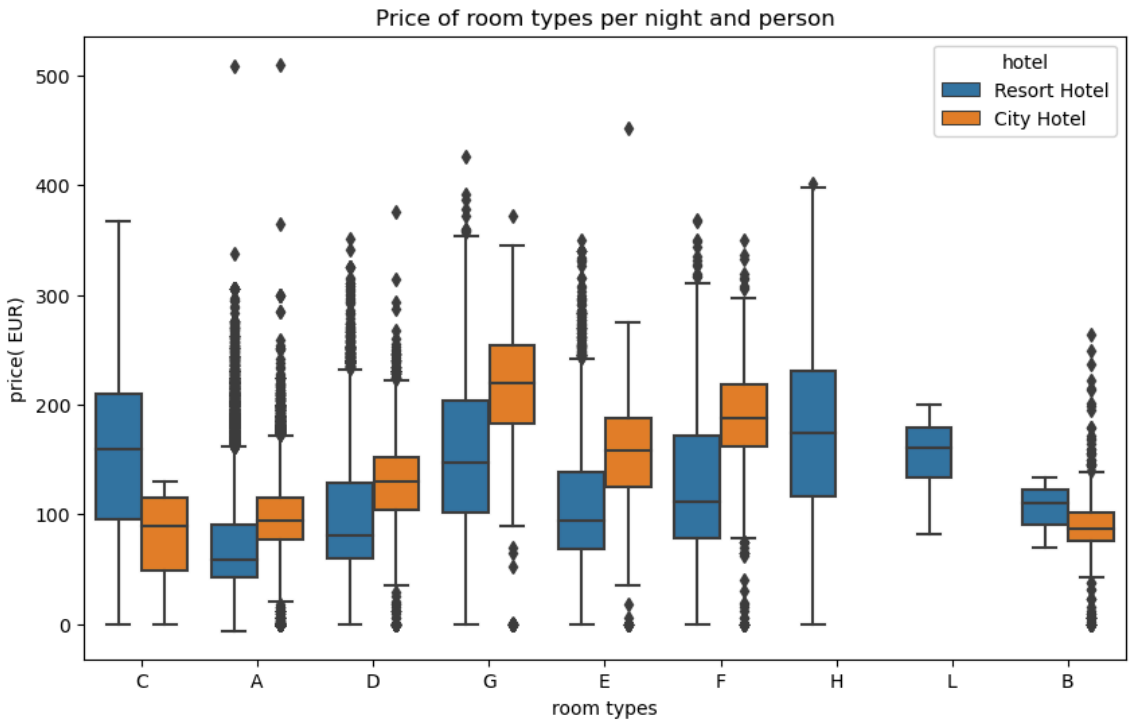
In [28]:
```
data2=data[data['is_canceled']==0]
```

In [29]:
```
data2.columns
```

Out[29]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type',
       'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

In [30]:
```
# seaborn boxplot:

plt.figure(figsize=(10,6))
sns.boxplot(x='reserved_room_type',y='adr' ,hue='hotel',data=data2)

plt.title('Price of room types per night and person')
plt.xlabel('room types')
plt.ylabel('price( EUR)')
```

Out[30]:
```
Text(0, 0.5, 'price( EUR)')
```



In case of City Hotel 'G' room type category performing well because median value is extremly high wheras in case of Resort Hotel 'H' room category is performing well .

3. Which are the most busy month ?

In [31]:
```
data['hotel'].unique()
```

Out[31]:
```
array(['Resort Hotel', 'City Hotel'], dtype=object)
```

In [32]:
```
data_resort=data[(data['hotel']=='Resort Hotel') & (data['is_canceled']==0)
data_city = data[(data['hotel']=='City Hotel') & (data['is_canceled']==0)]
```

In [33]:
```
data_resort.head(3)
```

Out[33]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nu
0	Resort Hotel	0	342	2015	July	
1	Resort Hotel	0	737	2015	July	
2	Resort Hotel	0	7	2015	July	

3 rows × 30 columns

In [34]:
```
rush_resort=data_resort['arrival_date_month'].value_counts().reset_index()
rush_resort.columns=['month','no_of_guests']
rush_resort
```

Out[34]:

	month	no_of_guests
0	August	3257
1	July	3137
2	October	2575
3	March	2571
4	April	2550
5	May	2535
6	February	2308
7	September	2102
8	June	2037
9	December	2014
10	November	1975
11	January	1866

In [35]:
```
rush_city=data_city['arrival_date_month'].value_counts().reset_index()
rush_city.columns=['month','no_of_guests']
rush_city
```

Out[35]:

	month	no_of_guests
0	August	5367
1	July	4770
2	May	4568
3	June	4358
4	October	4326
5	September	4283
6	March	4049
7	April	4010
8	February	3051
9	November	2676
10	December	2377
11	January	2249

In [36]:
```
final_rush=rush_resort.merge(rush_city,on='month')
```

In [37]:
```
final_rush.columns=['month','no_of_guests_in_resort','no_of_guests_city']
```

In [38]:
```
final_rush
```

Out[38]:

	month	no_of_guests_in_resort	no_of_guests_city
0	August	3257	5367
1	July	3137	4770
2	October	2575	4326
3	March	2571	4049
4	April	2550	4010
5	May	2535	4568
6	February	2308	3051
7	September	2102	4283
8	June	2037	4358
9	December	2014	2377
10	November	1975	2676
11	January	1866	2249

```
now we will observe over here is month column is not in order, & if we
will visualise we will get improper conclusion
```

```
so very first we have to provide right hierarchy to the month column
```

In [39]:
```
# !pip install sorted-months-weekdays

# ## Dependency package to be installed
# !pip install sort_dataframeby_monthorweek
```

In [40]:
```
import sort_dataframeby_monthorweek as sd
```

In [41]:
```
final_rush=sd.Sort_Dataframeby_Month(final_rush,'month')
```

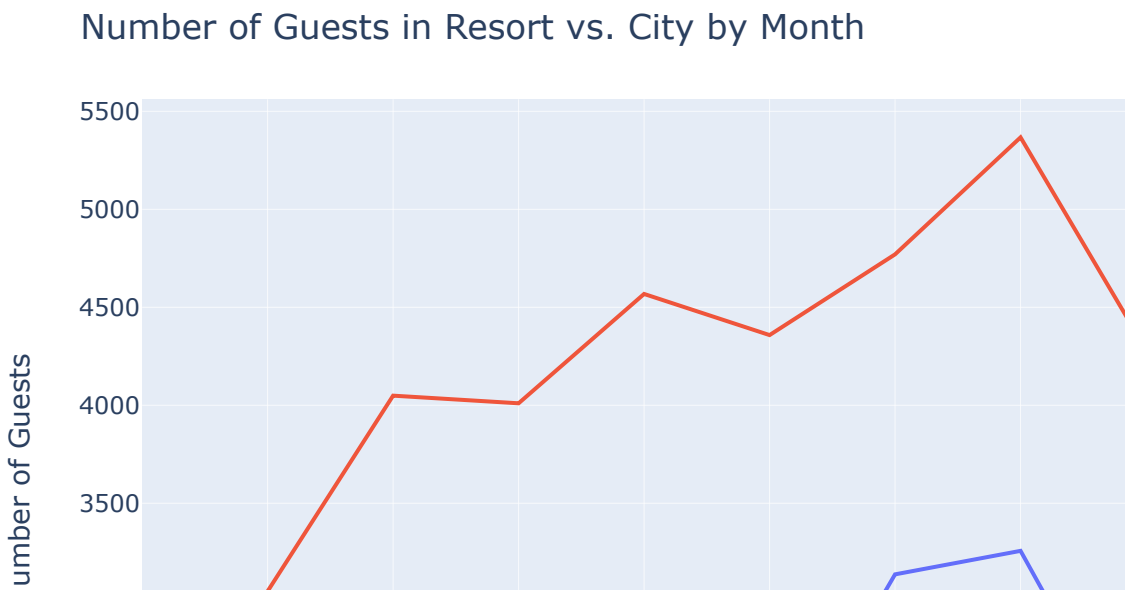
In [42]:
```
final_rush
```

Out[42]:

	month	no_of_guests_in_resort	no_of_guests_city
0	January	1866	2249
1	February	2308	3051
2	March	2571	4049
3	April	2550	4010
4	May	2535	4568
5	June	2037	4358
6	July	3137	4770
7	August	3257	5367
8	September	2102	4283
9	October	2575	4326
10	November	1975	2676
11	December	2014	2377

In [43]:

```
fig = px.line(data_frame=final_rush, x='month', y=['no_of_guests_in_resort'


fig.update_layout(
    title='Number of Guests in Resort vs. City by Month',
    xaxis_title='Month',
    yaxis_title='Number of Guests'
)

fig.show()
```



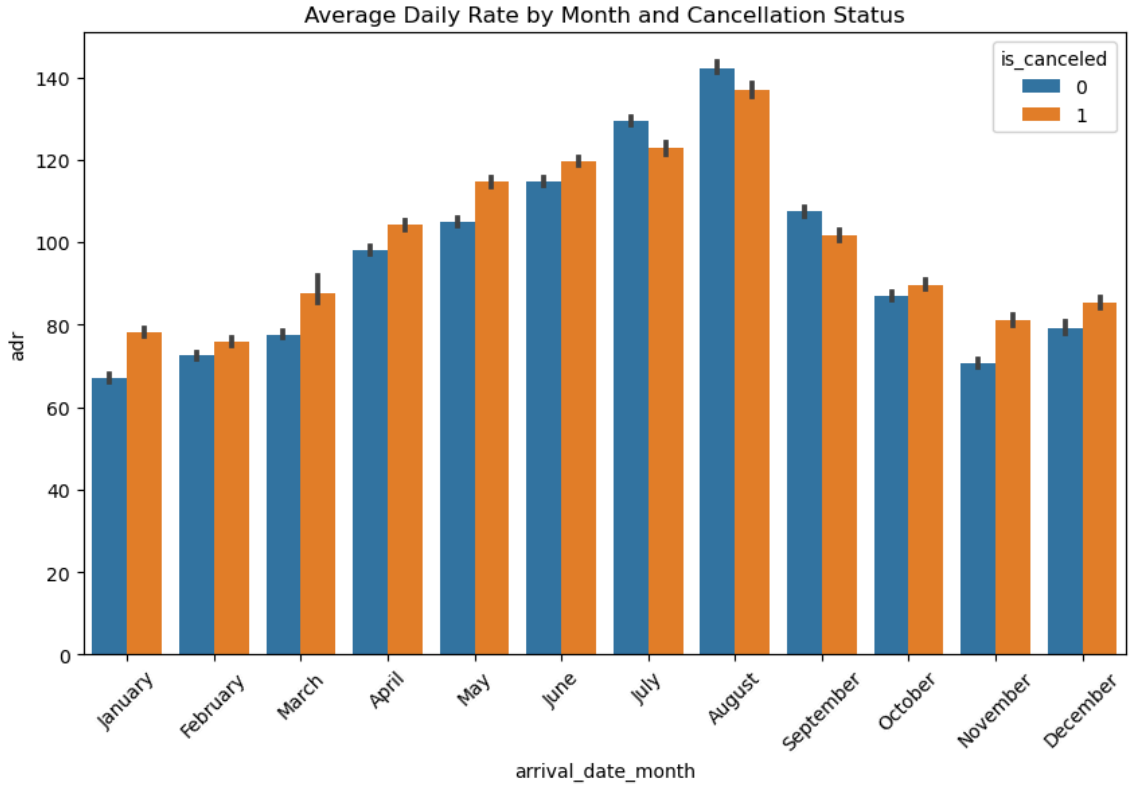

```
Conclusion-->> This clearly shows that the prices in the Resort hotel are
much higher during the summer (no surprise here)., The price of the city
hotel varies less and is most expensive during spring and autumn.
```

```
August is most intense month of booking , and rush of booking are in
july, september, october months
```

4. which month has highest adr ?

In [44]:
```
data=sd.Sort_Dataframeby_Month(data,'arrival_date_month')
```

In [45]:
```
#barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='arrival_date_month', y='adr', data=data, hue='is_canceled')

plt.xticks(rotation=45)
plt.title('Average Daily Rate by Month and Cancellation Status')

plt.show()
```




```
Conclusion: Most of the months cancelled booking have higher daily rate
than not cancelled booking
The adr feature is main reasone of cancelled booking.
```

5. Lets analyse whether bookings were made only for weekdays or for weekends or for both ?

In [46]:
```
data.columns
```

Out[46]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type',
       'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

In [47]:
```
### Lets create a relationship table..
pd.crosstab(index=data['stays_in_weekend_nights'],columns=data['stays_in_we
```

Out[47]:

stays_in_week_nights	0	1	2	3	4	5	6	7	8	9	...	24
stays_in_weekend_nights												
0	645	16436	17949	11557	4478	830	0	0	0	0	...	0
1	4569	7325	8976	6150	2407	1188	0	0	0	0	...	0
2	2358	6531	6745	4534	2658	8648	847	446	391	81	...	0
3	0	0	0	0	0	308	300	397	131	61	...	0
4	0	0	0	0	0	94	347	181	132	86	...	0
5	0	0	0	0	0	0	0	0	0	0	...	0
6	0	0	0	0	0	0	0	0	0	0	...	0
7	0	0	0	0	0	0	0	0	0	0	...	0
8	0	0	0	0	0	0	0	0	0	0	...	0
9	0	0	0	0	0	0	0	0	0	0	...	3
10	0	0	0	0	0	0	0	0	0	0	...	0
12	0	0	0	0	0	0	0	0	0	0	...	0
13	0	0	0	0	0	0	0	0	0	0	...	0
14	0	0	0	0	0	0	0	0	0	0	...	0
16	0	0	0	0	0	0	0	0	0	0	...	0
18	0	0	0	0	0	0	0	0	0	0	...	0
19	0	0	0	0	0	0	0	0	0	0	...	0

17 rows × 33 columns

In [48]:
```
## Lets define our own function :

def week_function(row):
    feature1='stays_in_weekend_nights'
    feature2='stays_in_week_nights'

    if row[feature2]==0 and row[feature1] >0 :
        return 'stay_just_weekend'

    elif row[feature2]>0 and row[feature1] ==0 :
        return 'stay_just_weekdays'

    elif row[feature2]>0 and row[feature1] >0 :
        return 'stay_both_weekdays_weekends'

    else:
        return 'undefined_data'
```

In [49]:
```
data2['weekend_or_weekday']=data2.apply(week_function,axis=1)
```

In [50]:
```
data2.head(5)
```

Out[50]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nu
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	

5 rows × 31 columns

In [51]:
```
data2['weekend_or_weekday'].value_counts()
```

Out[51]:
```
stay_both_weekdays_weekends    37551
stay_just_weekdays             31788
stay_just_weekend               5050
undefined_data                   622
Name: weekend_or_weekday, dtype: int64
```

In [52]:
```
data2=sd.Sort_Dataframeby_Month(data2,'arrival_date_month')
```

In [53]:
```
data2.groupby(['arrival_date_month','weekend_or_weekday']).size()
```

Out[53]:
```
arrival_date_month  weekend_or_weekday
April               stay_both_weekdays_weekends    3627
                    stay_just_weekdays             2559
                    stay_just_weekend               344
                    undefined_data                   30
August              stay_both_weekdays_weekends    4929
                    stay_just_weekdays             3137
                    stay_just_weekend               509
                    undefined_data                   49
December            stay_both_weekdays_weekends    1901
                    stay_just_weekdays             2123
                    stay_just_weekend               298
                    undefined_data                   69
February            stay_both_weekdays_weekends    2438
                    stay_just_weekdays             2514
                    stay_just_weekend               360
                    undefined_data                   47
January             stay_both_weekdays_weekends    1550
                    stay_just_weekdays             2125
                    stay_just_weekend               393
                    undefined_data                   47
July                stay_both_weekdays_weekends    4570
                    stay_just_weekdays             2818
                    stay_just_weekend               462
                    undefined_data                   57
June                stay_both_weekdays_weekends    3241
                    stay_just_weekdays             2685
                    stay_just_weekend               433
                    undefined_data                   36
March               stay_both_weekdays_weekends    3151
                    stay_just_weekdays             3060
                    stay_just_weekend               359
                    undefined_data                   50
May                 stay_both_weekdays_weekends    3442
                    stay_just_weekdays             3017
                    stay_just_weekend               570
                    undefined_data                   74
November            stay_both_weekdays_weekends    2117
                    stay_just_weekdays             2214
                    stay_just_weekend               261
                    undefined_data                   59
October             stay_both_weekdays_weekends    3393
                    stay_just_weekdays             2844
                    stay_just_weekend               582
                    undefined_data                   82
September           stay_both_weekdays_weekends    3192
                    stay_just_weekdays             2692
                    stay_just_weekend               479
                    undefined_data                   22
dtype: int64
```

In [54]:
```
group_data=data2.groupby(['arrival_date_month','weekend_or_weekday']).size(
group_data
```

Out[54]:

weekend_or_weekday	arrival_date_month	stay_both_weekdays_weekends	stay_just_weekdays
0	April	3627	2559
1	August	4929	3137
2	December	1901	2123
3	February	2438	2514
4	January	1550	2125
5	July	4570	2818
6	June	3241	2685
7	March	3151	3060
8	May	3442	3017
9	November	2117	2214
10	October	3393	2844
11	September	3192	2692

In [55]:
```
sorted_data=sd.Sort_Dataframeby_Month(group_data,'arrival_date_month')
sorted_data
```

Out[55]:

	arrival_date_month	stay_both_weekdays_weekends	stay_just_weekdays	stay_just_weeken
0	January	1550	2125	39
1	February	2438	2514	36
2	March	3151	3060	35
3	April	3627	2559	34
4	May	3442	3017	57
5	June	3241	2685	43
6	July	4570	2818	46
7	August	4929	3137	50
8	September	3192	2692	47
9	October	3393	2844	58
10	November	2117	2214	26
11	December	1901	2123	29

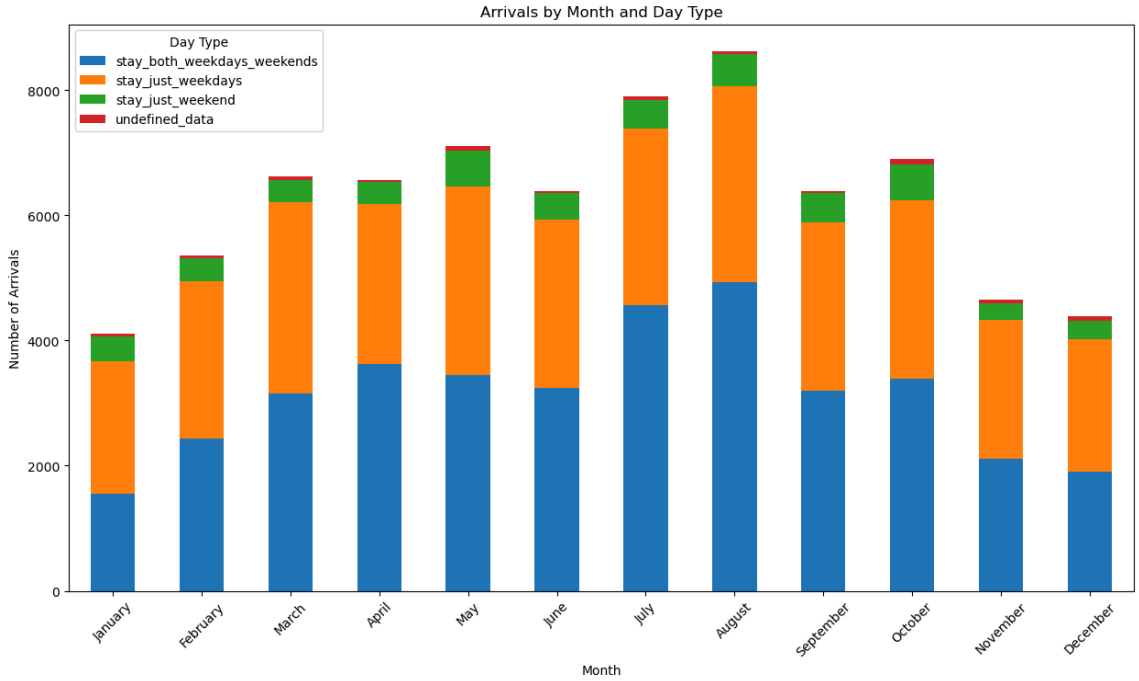
In [56]:
```
sorted_data.set_index('arrival_date_month',inplace=True)
```

In [57]:
```
sorted_data
```

Out[57]:

arrival_date_month	stay_both_weekdays_weekends	stay_just_weekdays	stay_just_weekend	u
January	1550	2125	393	
February	2438	2514	360	
March	3151	3060	359	
April	3627	2559	344	
May	3442	3017	570	
June	3241	2685	433	
July	4570	2818	462	
August	4929	3137	509	
September	3192	2692	479	
October	3393	2844	582	
November	2117	2214	261	
December	1901	2123	298	

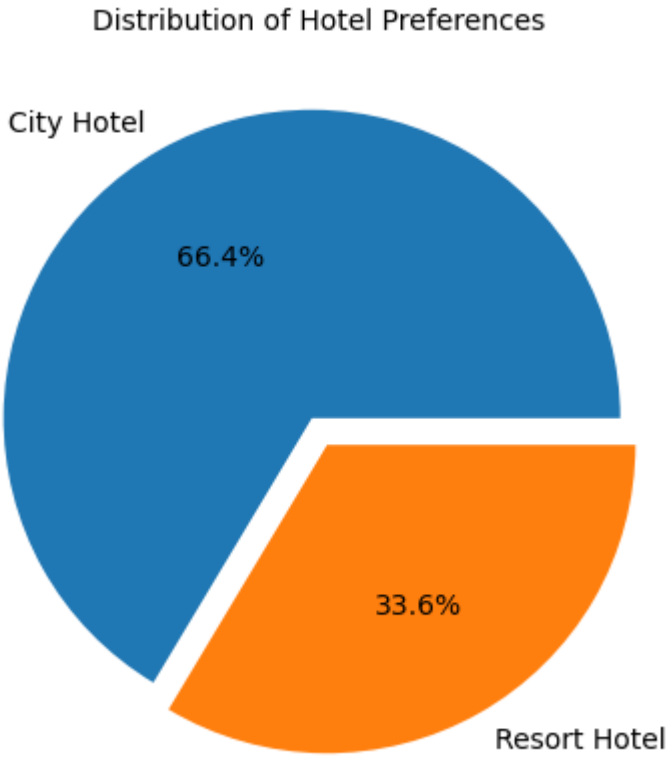
In [58]:
```
ax = sorted_data.plot(kind='bar', stacked=True, figsize=(15, 8))

plt.title('Arrivals by Month and Day Type')

plt.xlabel('Month')
plt.ylabel('Number of Arrivals')

plt.xticks(rotation=45)

plt.legend(title='Day Type', loc='upper left')
plt.show()
```



```
Conclusion : Most booking over stay only for the weekdays or for both
weekday_weekend.
```

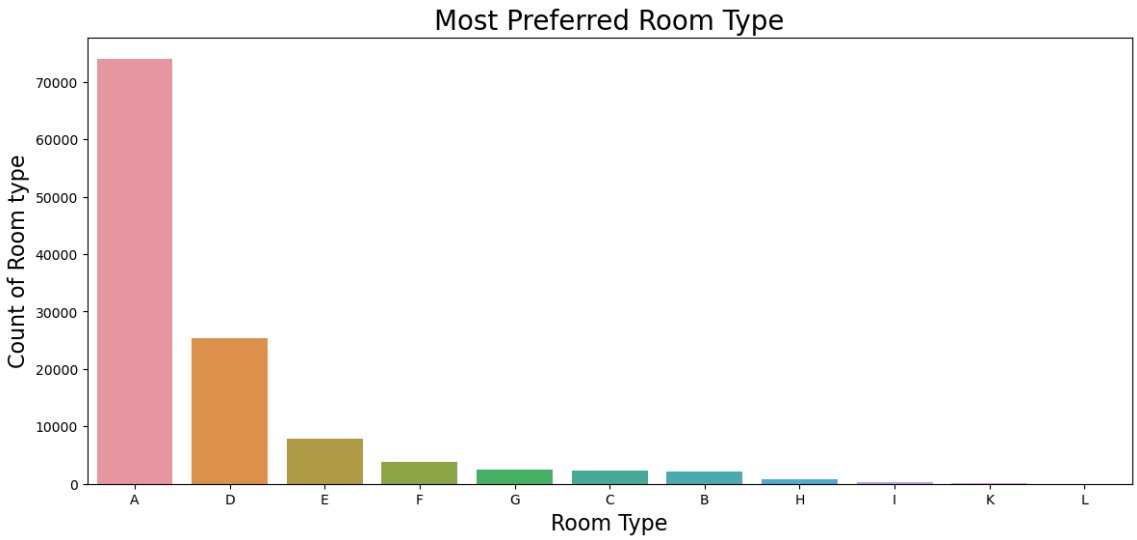
In [59]:

```
data['hotel'].value_counts().plot.pie(explode=[0.05, 0.05], figsize=(8, 5),


plt.title('Distribution of Hotel Preferences', fontsize=10)
plt.ylabel('')

plt.show()
```



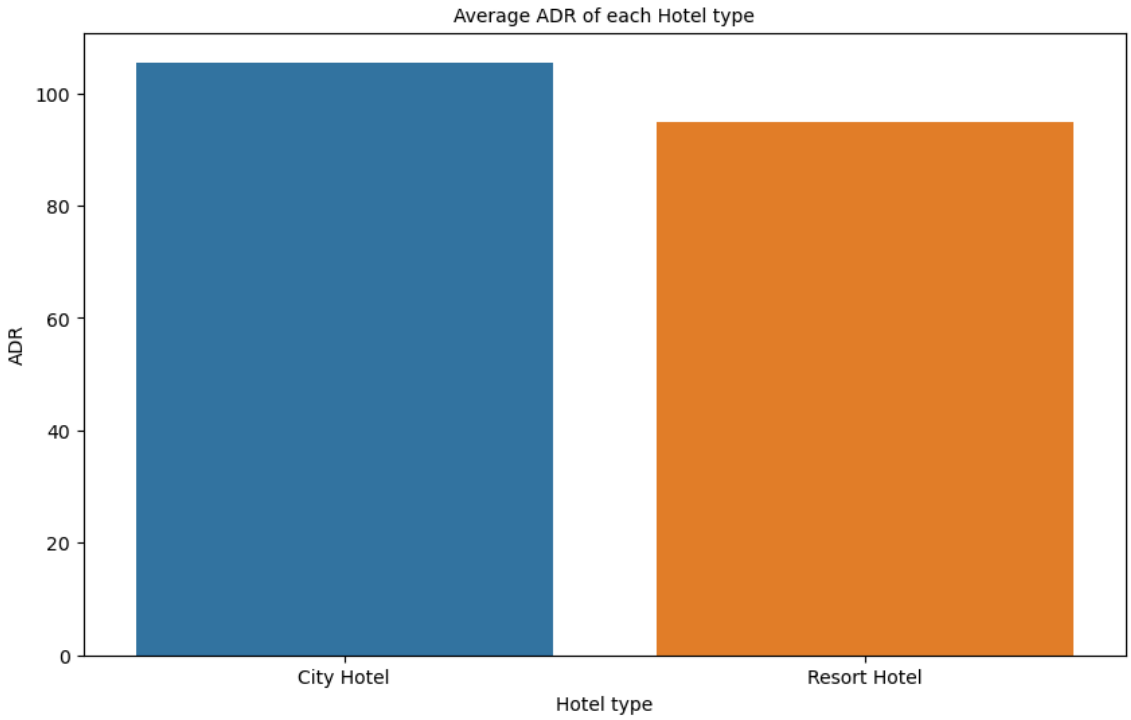

```
From the chart, we got to know that City Hotel is most preferred hotel by
the guests. Thus City Hotel has maximum bookings. 66.4% guests are
preferred City Hotel, while only 33.6% guests have shown interest in
Resort Hotel.
```

In [60]:
```
#countplot
plt.figure(figsize=(14, 6))

sns.countplot(x=data['assigned_room_type'], order=data['assigned_room_type'

plt.xlabel('Room Type', fontsize=16)
plt.ylabel('Count of Room type', fontsize=16)
plt.title('Most Preferred Room Type', fontsize=20)

plt.show()
```




```
Conclusion : It is found that the most preferred Room type is 'A'. So,
majority of the guests have shown interest in this room type.
There are positive impacts because 'A', 'D', 'E' is more preferred by
guest due to better services offered in room type.
```

In [61]:
```
# AVERAGE DAILY RATE OF HOTELS

group_by_hotel = data.groupby('hotel')

highest_adr = group_by_hotel['adr'].mean().reset_index()

plt.figure(figsize = (10,6))

ax = sns.barplot(x= highest_adr['hotel'], y= highest_adr['adr'])

ax.set_xlabel("Hotel type", fontsize = 10)
ax.set_ylabel("ADR", fontsize = 10)
ax.set_xticklabels(['City Hotel', 'Resort Hotel'], fontsize = 10)
ax.set_title('Average ADR of each Hotel type', fontsize = 10)

plt.show(ax)
```




```
Conclusion : City Hotels are generating more revenues than the Resort
Hotels, because City hotel has the highest ADR. More the ADR, more will
be the revenue.
```

Create some more features.

In [62]:
```
data.columns
```

Out[62]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type',
       'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

In [63]:
```
def family(row):
    if (row['adults'] > 0) & (row['children'] >0 or row['babies']>0):
        return 1
    else :
        return 0
```

In [64]:
```
data['is_family']= data.apply(family,axis=1)
```

In [65]:
```
data['total_customer']= data['adults'] + data['babies'] + data['children']
```

In [66]:
```
data['total_nights']= data['stays_in_week_nights'] + data['stays_in_weekend
```

In [67]:
```
data['deposit_type'].unique()
```

Out[67]:
```
array(['No Deposit', 'Non Refund', 'Refundable'], dtype=object)
```

In [68]:
```
dict1 = {'No Deposit':0, 'Non Refund':1, 'Refundable':0}
```

In [69]:
```
data['deposite_given']=data['deposit_type'].map(dict1)
```

In [70]:
```
data.columns
```

Out[70]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type',
       'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'is_family',
       'total_customer', 'total_nights', 'deposite_given'],
      dtype='object')
```

In [71]:
```
data.drop(columns=['adults', 'children', 'babies', 'deposit_type' ], axis=1
```

In [72]:
```
data.head(3)
```

Out[72]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nu
0	Resort Hotel	0	109	2016	January	
1	Resort Hotel	0	109	2016	January	
2	Resort Hotel	1	2	2016	January	

3 rows × 30 columns

FEATURE ENCODING

In [73]:
```
data.head(6)
```

Out[73]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_nu
0	Resort Hotel	0	109	2016	January	
1	Resort Hotel	0	109	2016	January	
2	Resort Hotel	1	2	2016	January	
3	Resort Hotel	0	88	2016	January	
4	Resort Hotel	1	20	2016	January	
5	Resort Hotel	1	76	2016	January	

6 rows × 30 columns

In [74]:
```
data.dtypes
```

Out[74]:
```
hotel                             object
is_canceled                        int64
lead_time                          int64
arrival_date_year                  int64
arrival_date_month                object
arrival_date_week_number           int64
arrival_date_day_of_month          int64
stays_in_weekend_nights            int64
stays_in_week_nights               int64
meal                              object
country                           object
market_segment                    object
distribution_channel              object
is_repeated_guest                  int64
previous_cancellations             int64
previous_bookings_not_canceled     int64
reserved_room_type                object
assigned_room_type                object
booking_changes                    int64
days_in_waiting_list               int64
customer_type                     object
adr                              float64
required_car_parking_spaces        int64
total_of_special_requests          int64
reservation_status                object
reservation_status_date           object
is_family                          int64
total_customer                   float64
total_nights                       int64
deposite_given                     int64
dtype: object
```

In [75]:
```
data.columns
```

Out[75]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'meal', 'country', 'market_segment',
       'distribution_channel', 'is_repeated_guest', 'previous_cancellation
s',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'days_in_waiting_list',
       'customer_type', 'adr', 'required_car_parking_spaces',
       'total_of_special_requests', 'reservation_status',
       'reservation_status_date', 'is_family', 'total_customer',
       'total_nights', 'deposite_given'],
      dtype='object')
```

In [76]:
```
cate_features=[col for col in data.columns if data[col].dtype=='object']
```

In [77]:
```
num_features=[col for col in data.columns if data[col].dtype!='object']
```

In [78]:
```
num_features
```

Out[78]:
```
['is_canceled',
 'lead_time',
 'arrival_date_year',
 'arrival_date_week_number',
 'arrival_date_day_of_month',
 'stays_in_weekend_nights',
 'stays_in_week_nights',
 'is_repeated_guest',
 'previous_cancellations',
 'previous_bookings_not_canceled',
 'booking_changes',
 'days_in_waiting_list',
 'adr',
 'required_car_parking_spaces',
 'total_of_special_requests',
 'is_family',
 'total_customer',
 'total_nights',
 'deposite_given']
```

In [79]:
```
cate_features
```

Out[79]:
```
['hotel',
 'arrival_date_month',
 'meal',
 'country',
 'market_segment',
 'distribution_channel',
 'reserved_room_type',
 'assigned_room_type',
 'customer_type',
 'reservation_status',
 'reservation_status_date']
```

In [80]:
```
data_cat=data[cate_features]
```

In [81]:
```
data.groupby(['hotel'])['is_canceled'].mean().to_dict()
```

Out[81]:
```
{'City Hotel': 0.4178593534858457, 'Resort Hotel': 0.27767373336329815}
```

In [82]:
```
import warnings
from warnings import filterwarnings
filterwarnings('ignore')
```

In [83]:
```
data_cat['cancellation']=data['is_canceled']
```

In [84]:
```
data_cat.head()
```

Out[84]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	reserved
0	Resort Hotel	January	BB	RUS	Online TA	TA/TO	
1	Resort Hotel	January	BB	RUS	Online TA	TA/TO	
2	Resort Hotel	January	BB	PRT	Online TA	TA/TO	
3	Resort Hotel	January	HB	ARG	Online TA	TA/TO	
4	Resort Hotel	January	BB	PRT	Online TA	TA/TO	

In [85]:
```
cols=data_cat.columns
```

In [86]:
```
cols=cols[0:-1]
```

In [87]:
```
cols
```

Out[87]:
```
Index(['hotel', 'arrival_date_month', 'meal', 'country', 'market_segment',
       'distribution_channel', 'reserved_room_type', 'assigned_room_type',
       'customer_type', 'reservation_status', 'reservation_status_date'],
      dtype='object')
```

In [88]:
```
### Perform Mean Encoding Technique

for col in cols:
    dict2=data_cat.groupby([col])['cancellation'].mean().to_dict()
    data_cat[col]=data_cat[col].map(dict2)
```

In [89]:
```
data_cat.head(3)
```

Out[89]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	r
0	0.277674	0.305016	0.374106	0.379365	0.36759	0.410598	
1	0.277674	0.305016	0.374106	0.379365	0.36759	0.410598	
2	0.277674	0.305016	0.374106	0.562958	0.36759	0.410598	

HANDLING OUTLIERS

In [90]:
```
data[num_features]
```

Out[90]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_
0	0	109	2016	1	
1	0	109	2016	1	
2	1	2	2016	1	
3	0	88	2016	1	
4	1	20	2016	1	
...	...	...	...	...	
119205	0	173	2016	53	
119206	0	17	2016	53	
119207	0	107	2016	53	
119208	0	310	2016	53	
119209	0	310	2016	53	

119210 rows × 19 columns

In [91]:
```
dataframe=pd.concat([data_cat,data[num_features]],axis=1)
```

In [92]:
```
dataframe.columns
```

Out[92]:
```
Index(['hotel', 'arrival_date_month', 'meal', 'country', 'market_segment',
       'distribution_channel', 'reserved_room_type', 'assigned_room_type',
       'customer_type', 'reservation_status', 'reservation_status_date',
       'cancellation', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_week_number', 'arrival_date_day_of_month',
       'stays_in_weekend_nights', 'stays_in_week_nights', 'is_repeated_gue
st',
       'previous_cancellations', 'previous_bookings_not_canceled',
       'booking_changes', 'days_in_waiting_list', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests', 'is_fam
ily',
       'total_customer', 'total_nights', 'deposite_given'],
      dtype='object')
```

In [93]:
```
dataframe.drop(['cancellation'],axis=1,inplace=True)
```

In [94]:
```
dataframe.head(3)
```

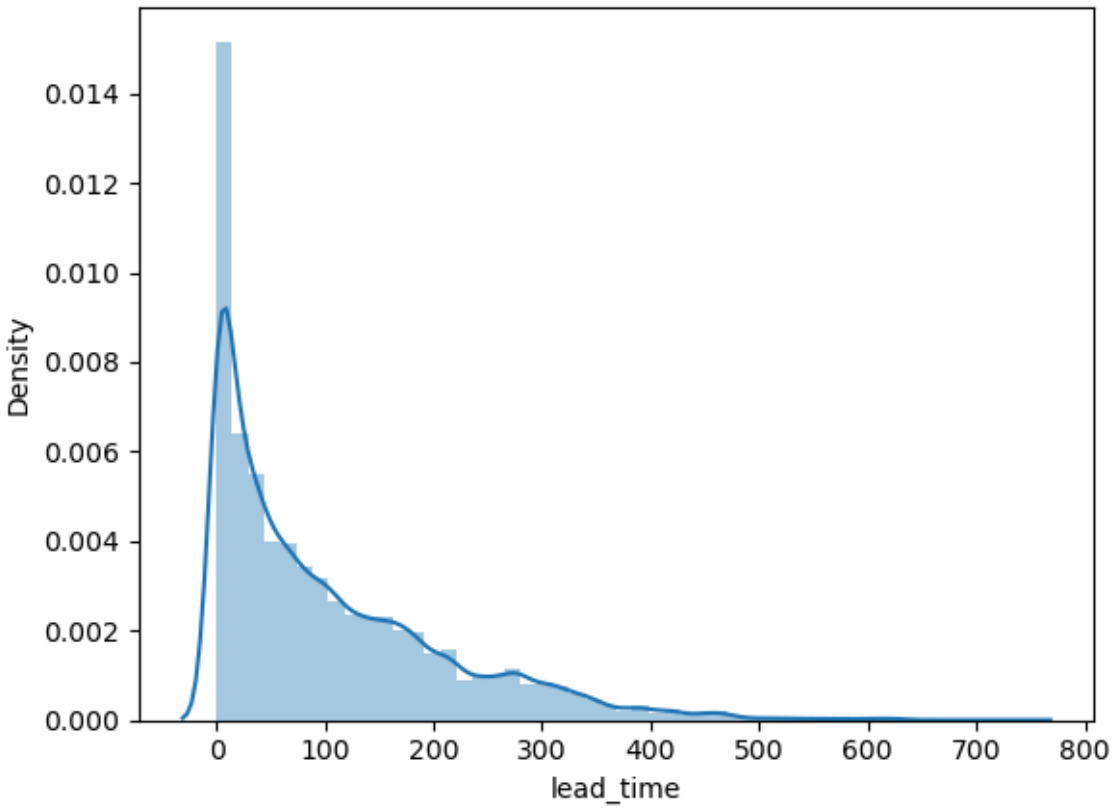
Out[94]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	r
0	0.277674	0.305016	0.374106	0.379365	0.36759	0.410598	
1	0.277674	0.305016	0.374106	0.379365	0.36759	0.410598	
2	0.277674	0.305016	0.374106	0.562958	0.36759	0.410598	

3 rows × 30 columns

In [95]:
```
sns.distplot(dataframe['lead_time'])
```

Out[95]:
```
<Axes: xlabel='lead_time', ylabel='Density'>
```

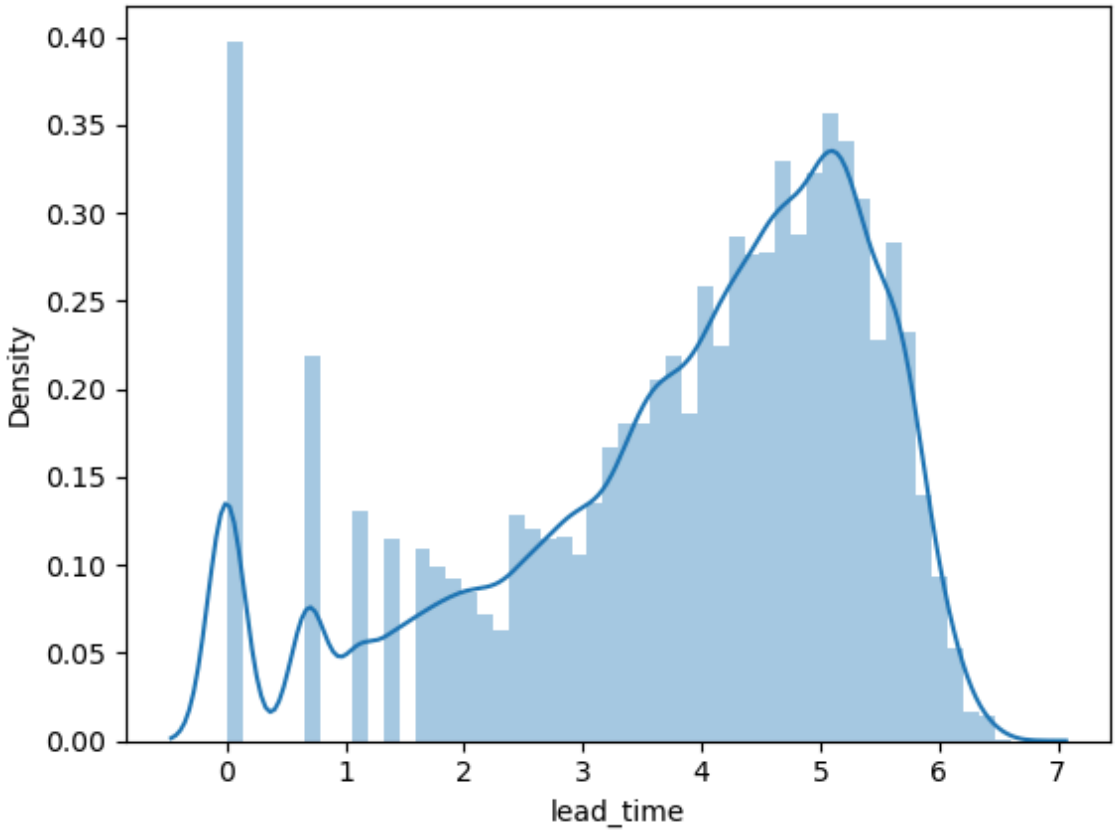


In [96]:
```
def handle_outlier(col):
    dataframe[col]=np.log1p(dataframe[col])
```

In [97]:
```
handle_outlier('lead_time')
```

In [98]:
```
sns.distplot(dataframe['lead_time'])
```

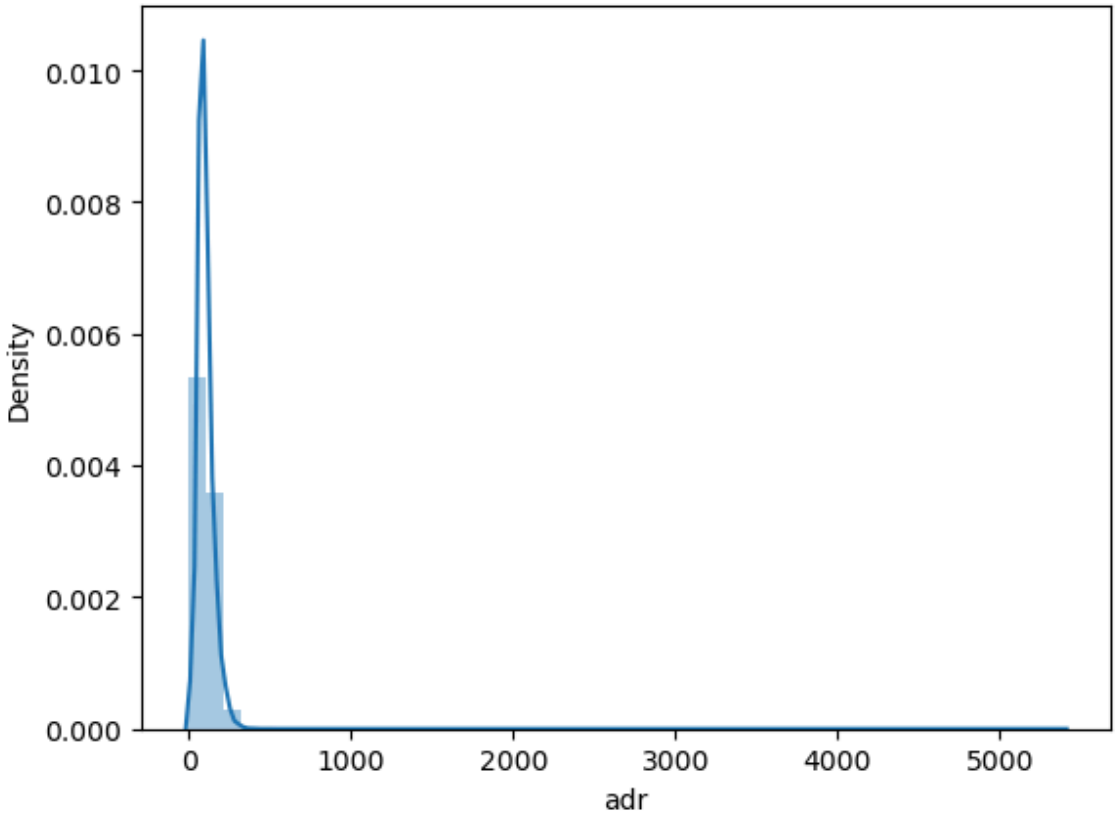
Out[98]:
```
<Axes: xlabel='lead_time', ylabel='Density'>
```



In [99]:
```
## adr
```

In [100]:
```
sns.distplot(dataframe['adr'])
```

Out[100]:
```
<Axes: xlabel='adr', ylabel='Density'>
```



In [101]:
```
dataframe[dataframe['adr']<0]
```

Out[101]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel
14989	0.277674	0.322277	0.374106	0.20231	0.611086	0.174868

1 rows × 30 columns

In [102]:
```
handle_outlier('adr')
```

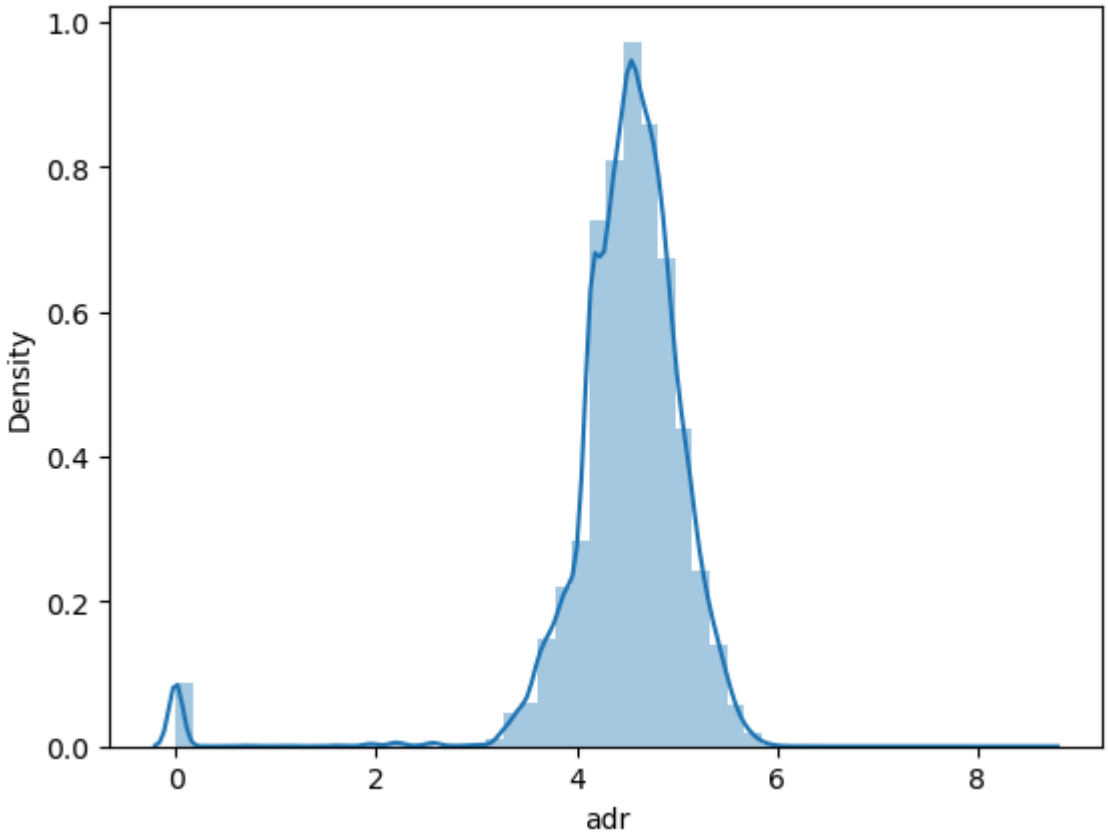
In [103]:
```
dataframe['adr'].isnull().sum()
```

Out[103]:
```
1
```

In [104]:
```
### now why this missing value , as we have already deal with the missing v
### bcz we have negative value in 'adr' feature as '-6.38'  ,& if we apply
## bcz log wont take negative values..
```

In [105]:
```
sns.distplot(dataframe['adr'].dropna())
```

Out[105]:
```
<Axes: xlabel='adr', ylabel='Density'>
```

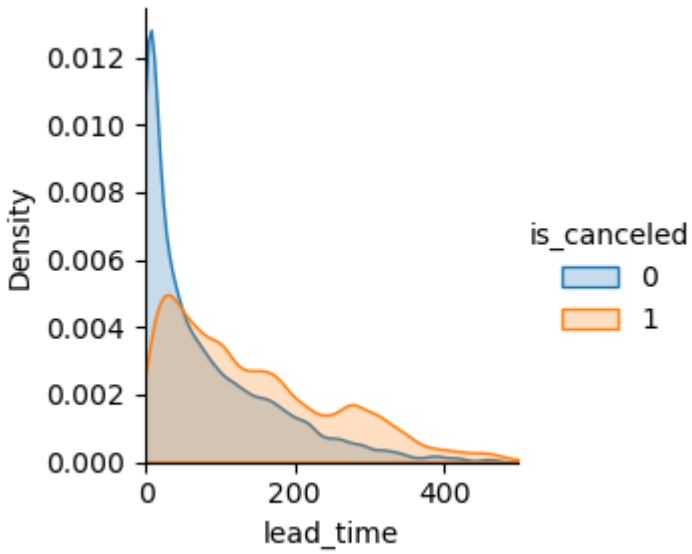


UNIVARIATE ANALYSIS

In [106]:
```
###UNIVARIATE ANALYSIS --One Varible is useful for ml model
```

In [107]:
```
sns.FacetGrid(data,hue='is_canceled',xlim=(0,500)).map(sns.kdeplot,'lead_ti
#kde--Kernel Density Estimation
```

Out[107]:
```
<seaborn.axisgrid.FacetGrid at 0x221b4678c50>
```



In [108]:
```
# upto some extend this feature plays important role bcoz it dosnt has comp
```

IMPORTANT FEATURE USING CO-RELATION

In [109]:
```
corr=dataframe.corr()
```

In [110]:
```
corr
# 2 feature
# High corr--1,.95,.9 --> better to drop cause situtation of overfitting
# low corr--- 0.01, 0.02--> interpretating low accuracy for model
```

Out[110]:

	hotel	arrival_date_month	meal	country	market_
hotel	1.000000	0.051197	0.061782	-0.040609	
arrival_date_month	0.051197	1.000000	-0.010208	-0.052405	
meal	0.061782	-0.010208	1.000000	0.022476	-
country	-0.040609	-0.052405	0.022476	1.000000	
market_segment	0.102592	0.047902	-0.026007	0.078982	
distribution_channel	0.182266	0.077075	-0.025486	-0.129774	
reserved_room_type	0.198171	-0.038801	0.010035	0.153177	
assigned_room_type	0.280884	0.015456	0.013913	0.125415	
customer_type	-0.008378	-0.029636	0.105767	-0.029320	-
reservation_status	0.137082	0.069886	0.050584	0.357232	
reservation_status_date	0.107481	-0.089512	0.038298	0.340400	
is_canceled	0.137082	0.069886	0.050584	0.357232	
lead_time	0.109869	0.241413	-0.040315	-0.071919	
arrival_date_year	0.035176	0.015509	-0.024568	-0.180684	-
arrival_date_week_number	0.001241	0.155653	0.015308	0.017322	
arrival_date_day_of_month	-0.001678	0.002248	-0.005737	0.001898	
stays_in_weekend_nights	-0.187816	0.046618	-0.040114	-0.146467	
stays_in_week_nights	-0.235955	0.036681	-0.053540	-0.138467	-
is_repeated_guest	-0.052526	-0.057691	0.009827	0.153305	-
previous_cancellations	-0.012261	0.008800	0.103606	0.093266	
previous_bookings_not_canceled	-0.004467	-0.021971	0.011109	0.091795	-
booking_changes	-0.076598	-0.011049	-0.027189	-0.061513	-
days_in_waiting_list	0.072725	0.029164	-0.031562	0.067054	
adr	0.154041	0.228179	-0.019908	-0.160520	-
required_car_parking_spaces	-0.218961	-0.021910	0.003455	0.007613	-
total_of_special_requests	-0.043478	0.002763	0.006875	-0.200822	-
is_family	-0.058094	0.020491	0.001466	-0.044139	-
total_customer	-0.038762	0.079090	-0.003447	-0.117351	-
total_nights	-0.249747	0.045648	-0.055689	-0.160759	-
deposite_given	0.172415	0.057361	0.047597	0.395685	

30 rows × 30 columns

In [111]:
```
corr['is_canceled'].sort_values(ascending=False)
```

Out[111]:
```
reservation_status                1.000000
is_canceled                       1.000000
reservation_status_date           0.488307
deposite_given                    0.481507
country                           0.357232
lead_time                         0.320075
market_segment                    0.267006
assigned_room_type                0.201570
distribution_channel              0.177167
hotel                             0.137082
customer_type                     0.136617
previous_cancellations            0.110139
adr                               0.081660
reserved_room_type                0.072769
arrival_date_month                0.069886
days_in_waiting_list              0.054301
meal                              0.050584
total_customer                    0.044826
stays_in_week_nights              0.025542
total_nights                      0.018554
arrival_date_year                 0.016622
arrival_date_week_number          0.008315
stays_in_weekend_nights          -0.001323
arrival_date_day_of_month        -0.005948
is_family                        -0.013226
previous_bookings_not_canceled   -0.057365
is_repeated_guest                -0.083745
booking_changes                  -0.144832
required_car_parking_spaces      -0.195701
total_of_special_requests        -0.234877
Name: is_canceled, dtype: float64
```

In [112]:
```
corr['is_canceled'].sort_values(ascending=False).index
```

Out[112]:
```
Index(['reservation_status', 'is_canceled', 'reservation_status_date',
       'deposite_given', 'country', 'lead_time', 'market_segment',
       'assigned_room_type', 'distribution_channel', 'hotel', 'customer_ty
pe',
       'previous_cancellations', 'adr', 'reserved_room_type',
       'arrival_date_month', 'days_in_waiting_list', 'meal', 'total_custom
er',
       'stays_in_week_nights', 'total_nights', 'arrival_date_year',
       'arrival_date_week_number', 'stays_in_weekend_nights',
       'arrival_date_day_of_month', 'is_family',
       'previous_bookings_not_canceled', 'is_repeated_guest',
       'booking_changes', 'required_car_parking_spaces',
       'total_of_special_requests'],
      dtype='object')
```

In [113]:
```
features_to_drop=['reservation_status', 'reservation_status_date','arrival_
       'arrival_date_week_number', 'stays_in_weekend_nights',
       'arrival_date_day_of_month']
```

In [114]:
```
dataframe.drop(features_to_drop,axis=1,inplace=True)
```

In [115]:
```
dataframe.shape
```

Out[115]:
```
(119210, 24)
```

FEATURE IMPORTANCE

In [116]:
```
dataframe.head(2)
```

Out[116]:

	hotel	arrival_date_month	meal	country	market_segment	distribution_channel	r
0	0.277674	0.305016	0.374106	0.379365	0.36759	0.410598	
1	0.277674	0.305016	0.374106	0.379365	0.36759	0.410598	

2 rows × 24 columns

In [117]:
```
dataframe.isnull().sum()
```

Out[117]:
```
hotel                             0
arrival_date_month                0
meal                              0
country                           0
market_segment                    0
distribution_channel              0
reserved_room_type                0
assigned_room_type                0
customer_type                     0
is_canceled                       0
lead_time                         0
stays_in_week_nights              0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
booking_changes                   0
days_in_waiting_list              0
adr                               1
required_car_parking_spaces       0
total_of_special_requests         0
is_family                         0
total_customer                    0
total_nights                      0
deposite_given                    0
dtype: int64
```

In [118]:
```
dataframe.dropna(inplace=True)
```

In [119]:
```
## separate dependent & independent features
```

In [120]:
```
x=dataframe.drop('is_canceled',axis=1)
```

In [121]:
```
y=dataframe['is_canceled']
```

In [122]:
```
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
```

In [123]:
```
##Lasso(alpha=0.005)
# select a suitable alpha (equivalent of penalty).
# The bigger the alpha the less features that will be selected.
```

In [124]:
```
feature_sel_model=SelectFromModel(Lasso(alpha=0.005))
```

In [125]:
```
feature_sel_model.fit(x,y)
```

Out[125]:
```
▸ SelectFromModel i ? (https://scikit-learn.org/1.4/modules/generated/sklearn.feature_selection.Sele
  ▸ estimator: Lasso
    ▸ Lasso ? (https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Lasso.html)
```

In [126]:
```
feature_sel_model.get_support()
```

Out[126]:
```
array([False, False, False,  True, False, False, False, False, False,
       True, False, False,  True,  True,  True,  True,  True,  True,
       True, False,  True,  True,  True])
```

In [127]:
```
cols=x.columns
```

In [128]:
```
cols.shape
```

Out[128]:
```
(23,)
```

In [129]:
```
# let's print the number of selected features

selected_feature=cols[feature_sel_model.get_support()]
```

In [130]:
```
selected_feature
```

Out[130]:
```
Index(['country', 'lead_time', 'previous_cancellations',
       'previous_bookings_not_canceled', 'booking_changes',
       'days_in_waiting_list', 'adr', 'required_car_parking_spaces',
       'total_of_special_requests', 'total_customer', 'total_nights',
       'deposite_given'],
      dtype='object')
```

In [131]:
```
selected_feature.shape
```

Out[131]:
```
(12,)
```

In [132]:
```
x=x[selected_feature]
```

In [133]:
```
y
```

Out[133]:
```
0         0
1         0
2         1
3         0
4         1
         ..
119205    0
119206    0
119207    0
119208    0
119209    0
Name: is_canceled, Length: 119209, dtype: int64
```

MODEL BUILDING

In [134]:
```
from sklearn.model_selection import train_test_split
```

In [135]:
```
X_train, X_test, y_train, y_test = train_test_split( x, y, test_size=0.25)
```

In [136]:
```
X_train.shape
```

Out[136]:
```
(89406, 12)
```

In [137]:
```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
```

In [138]:
```
logreg.fit(X_train,y_train)
```

Out[138]:
```
▾ LogisticRegression (https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Logist
LogisticRegression()
```

In [139]:
```
pred=logreg.predict(X_test)
```

In [140]:
```
pred
```

Out[140]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [141]:
```
from sklearn.metrics import confusion_matrix
```

In [142]:
```
confusion_matrix(y_test,pred)
```

Out[142]:
```
array([[17313,  1405],
       [ 4614,  6471]], dtype=int64)
```

In [143]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

Out[143]:
```
0.798040465724927
```

In [144]:
```
pred_prob_test = logreg.predict_proba(X_test)
```

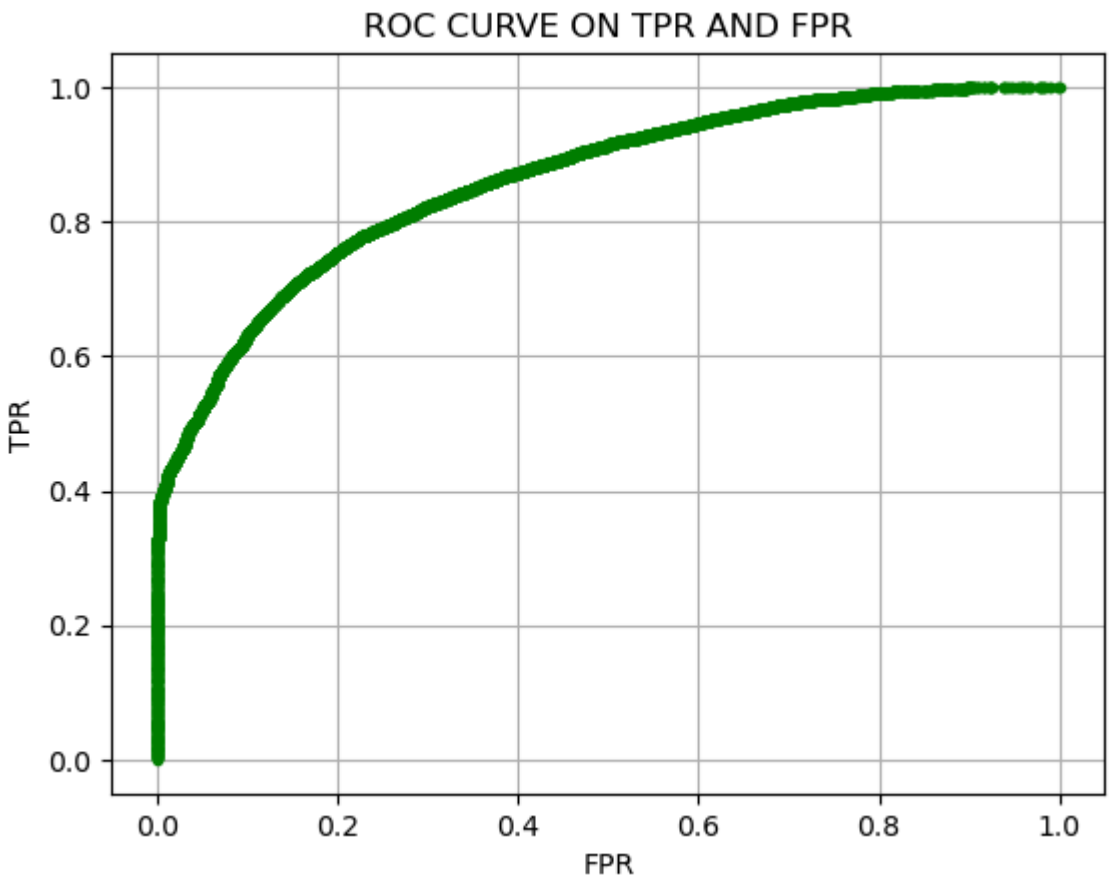
In [145]:
```
from sklearn.metrics import roc_auc_score, roc_curve
```

In [146]:
```
roc_auc_score(y_test ,pred_prob_test[:, 1])
```

Out[146]:
```
0.8612905703978664
```

In [147]:
```
fpr , tpr, threshold = roc_curve(y_test ,pred_prob_test[:, 1])
```

In [148]:
```
plt.plot(fpr, tpr , color='green' , marker='.')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title("ROC CURVE ON TPR AND FPR")
plt.grid()
```



APPLY ON MULTIPLE ALGORITHMS

In [149]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
```

In [150]:
```
models=[]

models.append(('LogisticRegression',LogisticRegression()))
models.append(('Decision_tree',DecisionTreeClassifier()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('Naive_bayes',GaussianNB()))
models.append(('Random Forest',RandomForestClassifier()))
```

In [151]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, classificatio

for name, model in models:
    print(name)
    model.fit(X_train, y_train)

    predictions = model.predict(X_test)

    cm = confusion_matrix(predictions, y_test)
    print("Confusion Matrix:")
    print(cm)

    acc = accuracy_score(predictions, y_test)
    print("Accuracy Score:", acc)

    report = classification_report(predictions, y_test)
    print("Classification Report:")
    print(report)

    print('\n')
```

```
LogisticRegression
Confusion Matrix:
[[17313  4614]
 [ 1405  6471]]
Accuracy Score: 0.798040465724927
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.79      0.85     21927
           1       0.58      0.82      0.68      7876

    accuracy                           0.80     29803
   macro avg       0.75      0.81      0.77     29803
weighted avg       0.83      0.80      0.81     29803



Decision_tree
Confusion Matrix:
[[16040  2625]
 [ 2678  8460]]
Accuracy Score: 0.8220648927960272
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86     18665
           1       0.76      0.76      0.76     11138

    accuracy                           0.82     29803
   macro avg       0.81      0.81      0.81     29803
weighted avg       0.82      0.82      0.82     29803



KNN
Confusion Matrix:
[[16696  3374]
 [ 2022  7711]]
Accuracy Score: 0.8189444015703117
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.83      0.86     20070
           1       0.70      0.79      0.74      9733

    accuracy                           0.82     29803
   macro avg       0.79      0.81      0.80     29803
weighted avg       0.83      0.82      0.82     29803



Naive_bayes
Confusion Matrix:
[[ 6953   660]
 [11765 10425]]
Accuracy Score: 0.5830956615105862
Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.91      0.53      7613
           1       0.94      0.47      0.63     22190
```

```
    accuracy                           0.58     29803
   macro avg       0.66      0.69      0.58     29803
weighted avg       0.80      0.58      0.60     29803



Random Forest
Confusion Matrix:
[[17164  2713]
 [ 1554  8372]]
Accuracy Score: 0.8568264939771164
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.86      0.89     19877
           1       0.76      0.84      0.80      9926

    accuracy                           0.86     29803
   macro avg       0.84      0.85      0.84     29803
weighted avg       0.86      0.86      0.86     29803
```

RANDOM FOREST GIVES GOOD RESULT WITH 85 % ACCURACY

In [152]:

```
evaluation_metrics = []

for name, model in models:
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    cm = confusion_matrix(predictions, y_test)
    acc = accuracy_score(predictions, y_test)
    report = classification_report(predictions, y_test, output_dict=True)

    precision = report['weighted avg']['precision']
    recall = report['weighted avg']['recall']
    f1_score = report['weighted avg']['f1-score']

    # Append metrics to the list
    evaluation_metrics.append([name, acc, precision, recall, f1_score])

# Create a DataFrame from the list of evaluation metrics
df = pd.DataFrame(evaluation_metrics, columns=['Model', 'Accuracy', 'Precis

df
```

Out[152]:

	Model	Accuracy	Precision	Recall	F1-Score
0	LogisticRegression	0.798040	0.834776	0.798040	0.807158
1	Decision_tree	0.823038	0.822905	0.823038	0.822970
2	KNN	0.818944	0.827852	0.818944	0.821668
3	Naive_bayes	0.583096	0.795113	0.583096	0.601442
4	Random Forest	0.856659	0.863462	0.856659	0.858570

In [ ]: