Bookings_Exploratory_Data_Analysis

November 22, 2022

[ ]:
```
# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import ExtraTreesClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import VotingClassifier

import folium
from folium.plugins import HeatMap
import plotly.express as px

plt.style.use('fivethirtyeight')
%matplotlib inline
pd.set_option('display.max_columns', 32)
```

[ ]:
```
df = pd.read_csv("hotel_booking.csv")
df.head()
```

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

[ ]:
```
df.info()
```

[ ]:
```
## Replace NaN in `agent` and `company` as 0
cols = ['agent', 'company']
for col in cols:
  df[col].fillna(0, inplace=True)
## Check leaves no NaN
print(pd.DataFrame({'#NaN': df[cols].isnull().sum(),
                    '%NaN': round(df[cols].isnull().mean() * 100, 2)}))
```

[ ]:
```
df.fillna(0,inplace=True)
```

[ ]:
```
#Understanding the correlation for each variable with is_canceled
corr = df.corr()['is_canceled']
corr.abs().sort_values(ascending=False)[1:]
```

[ ]:
```
# adults, babies and children cant be zero at same time, so dropping the rows␣
↪having all these zero at same time

filter = (df.children == 0) & (df.adults == 0) & (df.babies == 0)
df[filter]
```

[ ]:
```
df = df[~filter]
df
```

[ ]:
```
df=df.drop(['name','email','phone-number','credit_card'],axis=1)
```

[ ]:
```
df['No_of_Nights'] = df['stays_in_weekend_nights']+df['stays_in_week_nights']
```

[ ]:
```
## The home country of guests
country_dist = pd.DataFrame(df.loc[df["is_canceled"]==0]["country"].
↪value_counts())
country_dist.rename(columns={"country": "Number of Guests"}, inplace=True)
total_guests = country_dist["Number of Guests"].sum()
country_dist["Guest Percentage"] = round(country_dist["Number of Guests"] /␣
↪total_guests * 100, 2)
country_dist["Country"] = country_dist.index
country_dist
```

[ ]:
```
## Pie Plot
fig = px.pie(country_dist,
             values="Number of Guests",
             names="Country",
             title="Home country of guests")
fig.update_traces(textposition="inside", textinfo="value+percent+label")
fig.show()
```

[ ]:
```
## total bookings per market segment (incl. canceled)
segments=df["market_segment"].value_counts()

## pie plot
fig = px.pie(segments,
             values=segments.values,
             names=segments.index,
             title="Bookings per market segment")
fig.update_traces(rotation=-90, textinfo="percent+label")
fig.show()
```

[ ]:
```
df
```

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

df['year'] = df['reservation_status_date'].dt.year
df['month'] = df['reservation_status_date'].dt.month
df['day'] = df['reservation_status_date'].dt.day
```

[ ]:
```
## Reconstruct `adults`, `children`, `babies` as
## a new binary feature `is_family` and a new numerical feature␣
↪`customer_number`
## Reconstruct `stays_in_week_nights` and `stays_in_weekend_nights`
## as a new numerical feature `night_number`
def is_family(df):
  if ((df['adults'] > 0) & (df['children'] + df['babies'] > 0)):
    return 1
  else:
    return 0

df['is_family'] = df.apply(is_family, axis=1)
df['Number_of_guests'] = df['adults'] + df['children'] + df['babies']
```

[ ]:
```
## The home country of guests
country_cancel = pd.DataFrame(df.loc[df["is_canceled"]==1]["country"].
↪value_counts())
country_cancel.rename(columns={"country": "Number of Guests"}, inplace=True)
total_guests = country_cancel["Number of Guests"].sum()
country_cancel["Guest Percentage"] = round(country_cancel["Number of Guests"] /␣
↪total_guests * 100, 2)
country_cancel["Country"] = country_cancel.index
country_cancel
```

```
[ ]: ## Pie Plot
     fig = px.pie(country_cancel,
                  values="Number of Guests",
                  names="Country",
                  title="Home country of guests")
     fig.update_traces(textposition="inside", textinfo="value+percent+label")
     fig.show()
```

```
[ ]: # **prices in the Resort Hotel are much higher during the
     #summer and city hotel is most expensive during spring and autumn**
```

```
[ ]: plt.figure(figsize = (8, 4))

     px.line(df, x = 'arrival_date_month', y = ['adr'],
             title = 'Room price per night over the Months', template =␣
      ↪'plotly_dark')
```

```
[ ]: px.line(df, x = 'arrival_date_month', y = ['Number_of_guests'],
             title='Total No of Guests per Months', template = 'plotly_dark')
```

```
[ ]: plt.figure(figsize=(15, 8))
     plt.subplot(1, 2, 1)
     sns.countplot(x="market_segment", hue='is_canceled', data=df);
     plt.title('Types of market segment',fontweight="bold", size=20)

     plt.subplot(1, 2, 2)
     sns.countplot(x="distribution_channel", hue='is_canceled', data=df);
     plt.title('Types of distribution channels',fontweight="bold", size=20)
     plt.subplots_adjust(right=1.7)


     plt.show()
```

```
[ ]: plt.figure(figsize=(12, 6))

     sns.countplot(data = df, x = 'deposit_type',hue='hotel')
     plt.title('Types of Deposit type',fontweight="bold", size=20)


     plt.show()
```

```
[ ]: d1 = pd.DataFrame(df['agent'].value_counts()).reset_index().rename(
         columns = {'index':'agent','agent':'num_of_bookings'}).sort_values(
             by = 'num_of_bookings', ascending = False)
     d1.drop(d1[d1['agent'] == 0].index, inplace = True)
     # 0 represents that booking is not made by an agent
     d1 = d1[:10]
```

```
# Selecting top 10 performing agents
plt.figure(figsize = (10,5))
sns.barplot(x = 'agent', y = 'num_of_bookings', data = d1, order =
            d1.sort_values('num_of_bookings', ascending = False).agent)
```

```
[ ]: not_canceled = df[df['is_canceled'] == 0]
s1 = not_canceled[not_canceled['No_of_Nights'] < 15]
plt.figure(figsize = (10,5))
sns.countplot(x = s1['No_of_Nights'], hue = s1['hotel'])
plt.show()
#Most common stay length is less than 4 days and generally people prefer
#City hotel for short stay, but for long stays, Resort Hotel is preferred.
```

```
[ ]: # Selecting and counting number of cancelled bookings for each hotel.
cancelled_data = df[df['is_canceled'] == 1]
cancel_grp = cancelled_data.groupby('hotel')
D1 = pd.DataFrame(cancel_grp.size()).rename(
    columns = {0:'total_cancelled_bookings'})

# Counting total number of bookings for each type of hotel
grouped_by_hotel = df.groupby('hotel')
total_booking = grouped_by_hotel.size()
D2 = pd.DataFrame(total_booking).rename(columns = {0: 'total_bookings'})
D3 = pd.concat([D1,D2], axis = 1)

# Calculating cancel percentage
D3['cancel_%'] = round((D3['total_cancelled_bookings']/
                        D3['total_bookings'])*100,2)
D3
```

```
[ ]: group_by_dc_hotel = df.groupby(['distribution_channel', 'hotel'])

d5 = pd.DataFrame(round((group_by_dc_hotel['adr']).agg(
    np.mean),2)).reset_index().rename(columns = {'adr': 'avg_adr'})

plt.figure(figsize = (7,5))
sns.barplot(x = d5['distribution_channel'], y = d5['avg_adr'], hue =␣
 ↪d5['hotel'])
plt.ylim(40,140)
plt.show()
```

GDS channel brings higher revenue generating deals for City hotel, in contrast to that most bookings come via TA/TO. City Hotel can work to increase outreach on GDS channels to get more higher revenue generating deals.

Resort hotel has more revnue generating deals by direct and TA/TO channel. Resort Hotel need to increase outreach on GDS channel to increase revenue.

[ ]:
```
d_month = df['arrival_date_month'].value_counts().reset_index()
d_month.columns=['months','Number_of_guests']
d_month
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
          'August', 'September', 'October', 'November', 'December']
d_month['months'] = pd.Categorical(d_month['months'], categories=months,
                                   ordered=True)
d_month.sort_values('months').reset_index()


data_resort = df[(df['hotel'] == 'Resort Hotel') & (df['is_canceled'] == 0)]
data_city = df[(df['hotel'] == 'City Hotel') & (df['is_canceled'] == 0)]
resort_hotel = data_resort.groupby(['arrival_date_month'])['adr'
].mean().reset_index()
city_hotel=data_city.groupby(['arrival_date_month'])['adr'].mean().reset_index()
final_hotel = resort_hotel.merge(city_hotel, on = 'arrival_date_month')
final_hotel.columns = ['month', 'price_for_resort', 'price_for_city_hotel']
final_hotel

resort_guest = data_resort['arrival_date_month'].value_counts().reset_index()
resort_guest.columns=['month','no of guests']
resort_guest

city_guest = data_city['arrival_date_month'].value_counts().reset_index()
city_guest.columns=['month','no of guests']
city_guest

final_guest=resort_guest.merge(city_guest, on = 'month')
final_guest.columns=['month','no of guests in resort',
                     'no of guest in city hotel']
final_guest
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
          'August', 'September', 'October', 'November', 'December']
final_guest['month'] = pd.Categorical(final_guest['month'], categories=months,
                                      ordered=True)
final_guest = final_guest.sort_values('month').reset_index()

#Which month get most visitors?
sns.lineplot(data=final_guest, x='month', y='no of guests in resort')
sns.lineplot(data=final_guest, x='month', y='no of guest in city hotel')
plt.legend(['Resort','City Hotel'])
plt.ylabel('Number of guest')
fig = plt.gcf()
fig.set_size_inches(15,10)
```

[ ]:
```
```