

Agence Nationale de Réglementation des Télécommunications
(A.N.R.T)

L'Institut National des Postes et Télécommunications (INPT)

Rapport du :
Projet Data Mining

Réalisé par
TAMIR Taha
ALALOU Ayoub

Encadré par:
Mme. EL ASRI Ikram

Filière: INE2 DATA ENGINEER

02 Mai 2021

I. Dataset:

1. Data description :

This data describes two datasets with hotel demand data. One of the hotels (H1) is a resort hotel and the other is a city hotel (H2). Both datasets share the same structure, with 31 variables describing the 40,060 observations of H1 and 79,330 observations of H2. Each observation represents a hotel booking. Since this is hotel real data, all data elements pertaining hotel or costumer identification were deleted.

2. Data Dictionary :

variable	class	description
hotel	character	Hotel (H1 = Resort Hotel or H2 = City Hotel)
is_canceled	double	Value indicating if the booking was canceled (1) or not (0)
lead_time	double	Number of days that elapsed between the entering date of the booking into the PMS and the arrival date
arrival_date_year	double	Year of arrival date
arrival_date_month	character	Month of arrival date
arrival_date_week_number	double	Week number of year for arrival date
arrival_date_day_of_month	double	Day of arrival date
stays_in_weekend_nights	double	Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
stays_in_week_nights	double	Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
adults	double	Number of adults
children	double	Number of children
babies	double	Number of babies
meal	character	Type of meal booked. Categories are presented in standard hospitality meal packages: Undefined/SC – no meal package; BB – Bed & Breakfast;

variable	class	description
		HB – Half board (breakfast and one other meal – usually dinner); FB – Full board (breakfast, lunch and dinner)
country	character	Country of origin. Categories are represented in the ISO 3155–3:2013 format
market_segment	character	Market segment designation. In categories, the term “TA” means “Travel Agents” and “TO” means “Tour Operators”
distribution_channel	character	Booking distribution channel. The term “TA” means “Travel Agents” and “TO” means “Tour Operators”
is_repeated_guest	double	Value indicating if the booking name was from a repeated guest (1) or not (0)
previous_cancellations	double	Number of previous bookings that were cancelled by the customer prior to the current booking
previous_bookings_not_canceled	double	Number of previous bookings not cancelled by the customer prior to the current booking
reserved_room_type	character	Code of room type reserved. Code is presented instead of designation for anonymity reasons
assigned_room_type	character	Code for the type of room assigned to the booking. Sometimes the assigned room type differs from the reserved room type due to hotel operation reasons (e.g. overbooking) or by customer request. Code is presented instead of designation for anonymity reasons
booking_changes	double	Number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation
deposit_type	character	Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories: No Deposit – no deposit was made; Non Refund – a deposit was made in the value of the total stay cost; Refundable – a deposit was made with a value under the total cost of stay.
agent	character	ID of the travel agency that made the booking
company	character	ID of the company/entity that made the booking or responsible for paying the booking. ID is presented instead of designation for anonymity reasons
days_in_waiting_list	double	Number of days the booking was in the waiting list before it was confirmed to the customer

variable	class	description
customer_type	character	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
adr	double	Average Daily Rate as defined by dividing the sum of all lodging transactions by the total number of staying nights
required_car_parking_spaces	double	Number of car parking spaces required by the customer
total_of_special_requests	double	Number of special requests made by the customer (e.g. twin bed or high floor)
reservation_status	character	Reservation last status, assuming one of three categories: Canceled - booking was canceled by the customer; Check-Out - customer has checked in but already departed; No-Show - customer did not check-in and did inform the hotel of the reason why
reservation_status_date	double	Date at which the last status was set. This variable can be used in conjunction with the ReservationStatus to understand when was the booking canceled or when did the customer checked-out of the hotel

II. Data preprocessing

On commence par l'importation des bibliothèques nécessaires à notre travail, puis on montre le nombre totale des données et afficher les cinq premiers lignes de nos données :

I.Libraries

```

Entrée [ ]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

Entrée [2]: pd.set_option('display.max_columns', 34)

Entrée [74]: data = pd.read_csv('hotelsBookingDemand.csv')

Entrée [4]: len(data)
Out[4]: 119390

Entrée [5]: data.head()
Out[5]:

```

	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
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	

1. Verify missing values

Pour la phase Prétraitement des données, on commence par la vérification des valeurs manquantes :

```

II. Data preprocessing
1. Verify missing values

Entrée [6]: data.isna().sum()

Out[6]: hotel                0
is_canceled                0
lead_time                 0
arrival_date_year          0
arrival_date_month         0
arrival_date_week_number   0
arrival_date_day_of_month  0
stays_in_weekend_nights    0
stays_in_week_nights       0
adults                    0
children                   4
babies                    0
meal                      0
country                   488
market_segment             0
distribution_channel        0
is_repeated_guest          0
previous_cancellations      0
previous_bookings_not_canceled 0
reserved_room_type         0
assigned_room_type         0
booking_changes            0
deposit_type               0
agent                     16340
company                   112593
days_in_waiting_list       0
customer_type              0
adr                        0
required_car_parking_spaces 0
total_of_special_requests   0
reservation_status         0
reservation_status_date     0
dtype: int64

```

On remarque que 4 colonnes qui ont des valeurs manquantes, à savoir : **Children**, **Country**, **Agent** et **Company**.

2. Drop rows having missing values

On essaie d'éliminer les ligne ayant les valeurs manquantes pour la colonne Country et Children, mais on laisse ces ligne pour les colonnes Agent et Company , car elles présentent l'une des catégories.

2. Drop rows having missing values except for variables like Agent or Company. "NULL" is presented as one of the categories

Entrée [7]: `data.dropna(axis=0,subset=['country','children'],inplace=True)`

Entrée [8]: `data.isna().sum(),len(data)`

```
Out[8]: (hotel                0
is_canceled                0
lead_time                  0
arrival_date_year           0
arrival_date_month          0
arrival_date_week_number    0
arrival_date_day_of_month   0
stays_in_weekend_nights     0
stays_in_week_nights        0
adults                      0
children                    0
babies                      0
meal                        0
country                     0
market_segment              0
distribution_channel         0
is_repeated_guest           0
previous_cancellations       0
previous_bookings_not_canceled 0
reserved_room_type           0
assigned_room_type           0
booking_changes              0
deposit_type                 0
agent                       16004
company                     112275
days_in_waiting_list        0
customer_type                0
adr                          0
required_car_parking_spaces  0
total_of_special_requests    0
reservation_status           0
reservation_status_date      0
dtype: int64,
118898)
```

- Pour les valeurs manquantes pour les colonnes Agent et Company seront traitées dans la suite.

3. Change arrival year, month and day feature to datetime format called arrival_date.

On trouve que la date d'arrivée est présentée par 3 colonnes: « arrival_date_year », « arrival_date_month » et « arrival_date_day_of_month » :

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights
0	Resort Hotel	0	342	2015	July	27	1	0
1	Resort Hotel	0	737	2015	July	27	1	0
2	Resort Hotel	0	7	2015	July	27	1	0
3	Resort Hotel	0	13	2015	July	27	1	0
4	Resort Hotel	0	14	2015	July	27	1	0
...
119385	City Hotel	0	23	2017	August	35	30	2
119386	City Hotel	0	102	2017	August	35	31	2
119387	City Hotel	0	34	2017	August	35	31	2
119388	City Hotel	0	109	2017	August	35	31	2
119389	City Hotel	0	205	2017	August	35	29	2

118896 rows x 33 columns

1
2
3

Alors on essaie de la présenter par une seule colonne « **année-mois-jour** » :

```
3. Change arrival year, month and day feature to datetime format called arrival_date.

Entrée [9]: data['arrival_date']=data['arrival_date_day_of_month'].astype(str)+'th of '+data['arrival_date_month'].astype(str)+'', '+data['arri
Entrée [10]: data['arrival_date']=pd.to_datetime(data['arrival_date'])
Entrée [11]: data
```

Entrée [11]: data

Out[11]:

any	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	reservation_status	reservation_status_date	arrival_date
NaN	0	Transient	0.00	0	0	Check-Out	2015-07-01	2015-07-01
NaN	0	Transient	0.00	0	0	Check-Out	2015-07-01	2015-07-01
NaN	0	Transient	75.00	0	0	Check-Out	2015-07-02	2015-07-01
NaN	0	Transient	75.00	0	0	Check-Out	2015-07-02	2015-07-01
NaN	0	Transient	98.00	0	1	Check-Out	2015-07-03	2015-07-01
...
NaN	0	Transient	96.14	0	0	Check-Out	2017-09-06	2017-08-30
NaN	0	Transient	225.43	0	2	Check-Out	2017-09-07	2017-08-31
NaN	0	Transient	157.71	0	4	Check-Out	2017-09-07	2017-08-31
NaN	0	Transient	104.40	0	0	Check-Out	2017-09-07	2017-08-31
NaN	0	Transient	151.20	0	2	Check-Out	2017-09-07	2017-08-29

4. Verify that the timestamp of the variable `reservation_status_date` must occur after or at the same date as the input variable `arrival_date`

Ensuite, on vérifie que l'horodatage de la variable `reservation_status_date` doit apparaître après ou à la même date que la variable d'entrée `arrival_date` :

```
4. Verify that the timestamp of the variable reservation_status_date must occur after or at the same date as the input variable arrival_date

Entrée [12]: data["reservation_status_date"]=pd.to_datetime(data["reservation_status_date"])
Entrée [13]: test = data["reservation_status_date"]<=data['arrival_date']
Entrée [14]: test2=test.loc[test==True].index
```

On affiche notre résultat :

```
Entrée [15]: len(test2)
Out[15]: 44854
```

On remarque que 44854 réservations ne respectent pas cette condition.

5. Propose a preprocessing to be made on this dataset.

Pour le premier prétraitement qu'on a proposé est de donner à ces 44854 réservations la date de changement de statique correspondant au jour d'arrivée, comme la figure ci-dessous :

```
5. Propose a preprocessing to be made on this dataset.

Première méthode

Entrée [16]: sta.loc[data["reservation_status_date"]<=data['arrival_date'], 'reservation_status_date']=list(data['arrival_date'])[test2].values)

Entrée [17]: test = data["reservation_status_date"]<data['arrival_date']
             test2=test.loc[test==True].index
             len(test2)

Out[17]: 0
```

D'autres prétraitements seront faits sur ces données par la suite, car on a besoin de l'intégrité des données pour faire des analyses statistiques pour la partie suivante.

III. Exploratory data analysis:

Cette partie est particulière pour faire des analyses statistiques sur les données.

On commence par diviser ces colonnes en 3 types :

- Colonnes qui présentent des dates ;
- Colonnes qui présentent des valeurs catégoriques;
- Colonnes qui présentent des valeurs numériques ou continues.

III. Exploratory data analysis:

```

Entrée [18]: categorical=['hotel','is_canceled','meal','country','market_segment',
                        'distribution_channel','is_repeated_guest','reserved_room_type','assigned_room_type',
                        'deposit_type','customer_type','reservation_status']
len(categorical)

Out[18]: 12

Entrée [19]: cat=[]
for i in categorical:
    cat.append(len(data[i].unique()))

Entrée [20]: num=list(set(data.columns) - set(categorical))
num=list(set(num)-set(['arrival_date_year','arrival_date_month','arrival_date_week_number','arrival_date_day_of_month','reservat
num

Out[20]: ['lead_time',
          'babies',
          'required_car_parking_spaces',
          'agent',
          'stays_in_week_nights',
          'children',
          'booking_changes',
          'days_in_waiting_list',
          'stays_in_weekend_nights',
          'adults',
          'company',
          'total_of_special_requests',
          'previous_cancellations',
          'previous_bookings_not_canceled',
          'adr']

Entrée [21]: date = list(set(data.columns) - set(categorical)-set(num))

```

1. . Create dataset summary statistics – Date variables.

On crée des statistiques récapitulatives de jeu de données sur les variables de date, alors on essaie de calculer le nombre d'arrivées correspondante à chaque date :

1. Create dataset summary statistics – Date variables.

```

Entrée [22]: data['count']=1
data.head()

Out[22]:
   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_
0  Resort Hotel      0       342            2015           July                27                    1                    0
1  Resort Hotel      0       737            2015           July                27                    1                    0
2  Resort Hotel      0         7            2015           July                27                    1                    0
3  Resort Hotel      0        13            2015           July                27                    1                    0
4  Resort Hotel      0        14            2015           July                27                    1                    0

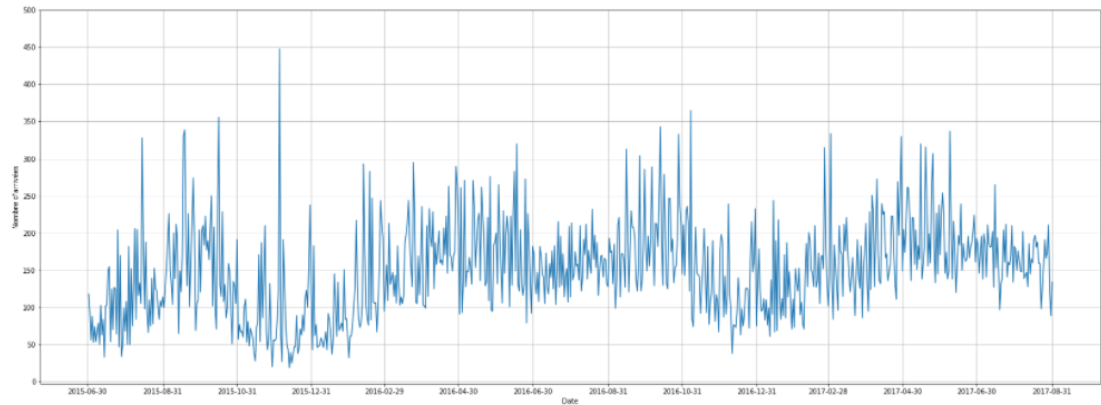
Entrée [23]: date_t = data['arrival_date'].unique()

```

```
Entrée [24]: sum_t=[]
             for i in date_t:
                 sum_t.append(sum(data.loc[data['arrival_date']==i,'count']))
```

```
Entrée [25]: plt.rcParams["figure.figsize"] = (28,10)

             plt.plot(date_t[:784],sum_t[:784])
             plt.xticks(pd.date_range('2015-06','2017-10',freq='2M'))
             plt.yticks(np.arange(0,550,50))
             plt.grid()
             plt.xlabel('Date')
             plt.ylabel("Nombre d'arrivées")
             plt.show()
```

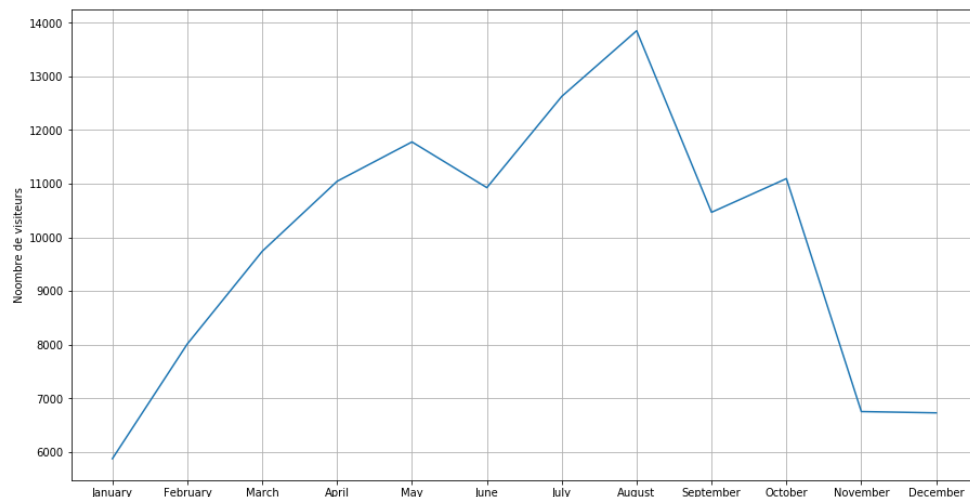


On remarque certaines saisonnalités dans les données, car on trouve des pics chaque année dans la période entre le mois Juillet et Septembre, et il est centré plus précisément au mois Aout.

- Ce qui on peut interpréter par l'augmentation au niveau de demandes dans la période d'été.

```
Entrée [26]: months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
                      'August', 'September', 'October', 'November', 'December']
             sum_t1=[]
             for i in months:
                 sum_t1.append(sum(data.loc[data['arrival_date_month']==i,'count']))
```

```
Entrée [27]: plt.rcParams["figure.figsize"] = (15,8)
             plt.plot(months,sum_t1)
             plt.grid()
             plt.xlabel('Mois')
             plt.ylabel('Noombre de visiteurs')
             plt.show()
```



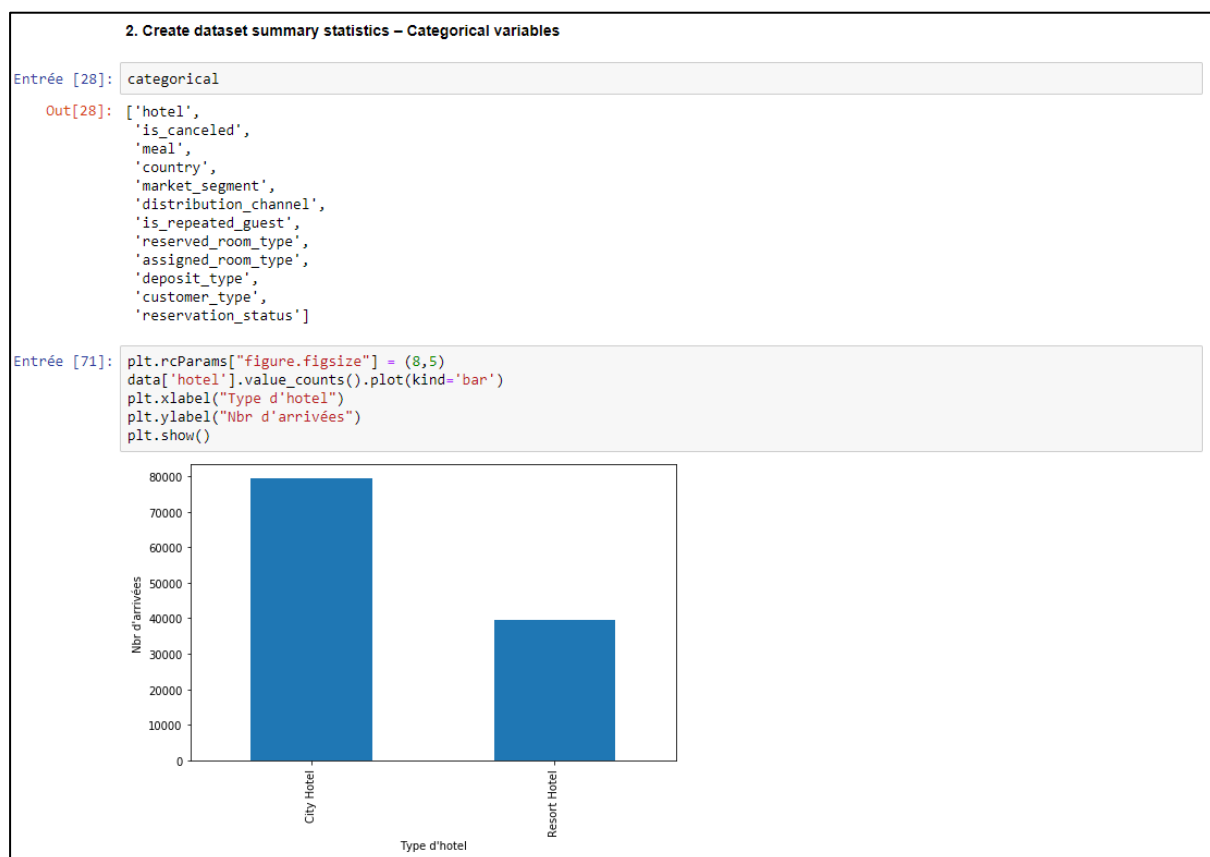
Pour confirmer cette remarque, on essaie de présenter l'ensemble des arrivées sur les trois années de 2005 à 2007 par rapport au mois comme la montre la figure ci-dessus.

➤ Ce qui nous montre que le mois Aout a le plus haut pic d'arrivées.

2. Create dataset summary statistics – Categorical variables.

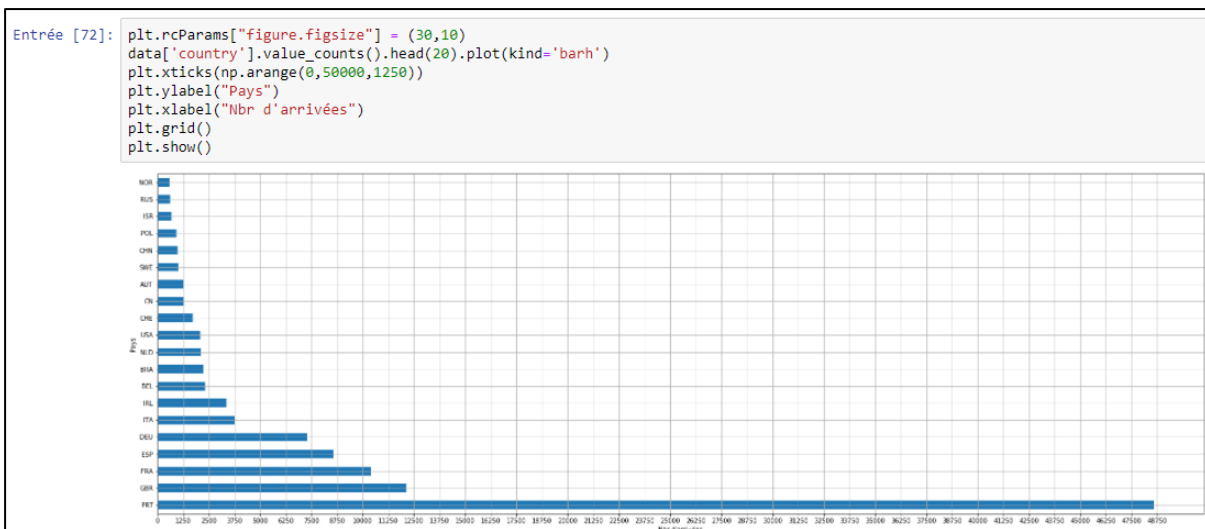
On passe à créer des statistiques récapitulatives de jeu de données - Variables catégorielles. On commence par :

■ Distribution des types d'hôtels :



On remarque que la plupart des réservations sont faites sur **City Hotel**.

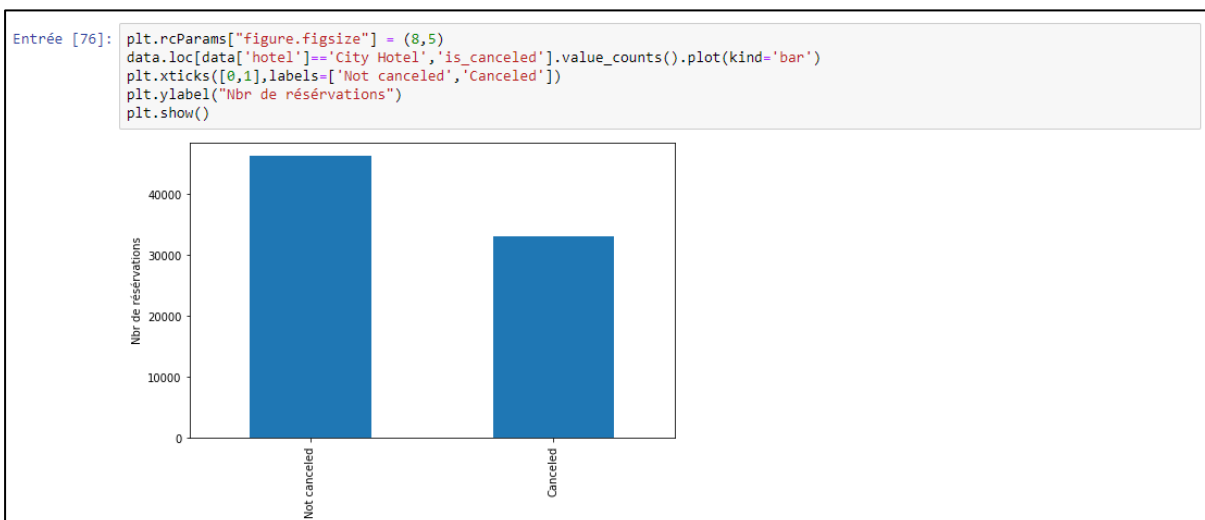
- Distribution des visites selon les pays :



On trouve que le Portugal a la majorité des réservations suivie du Royaume-Uni de Grande-Bretagne et d'Irlande du Nord. Mais le Portugal présente la quasi-totalité des clients de ces deux hôtels

- Annulation des réservations selon les types d'hôtels :

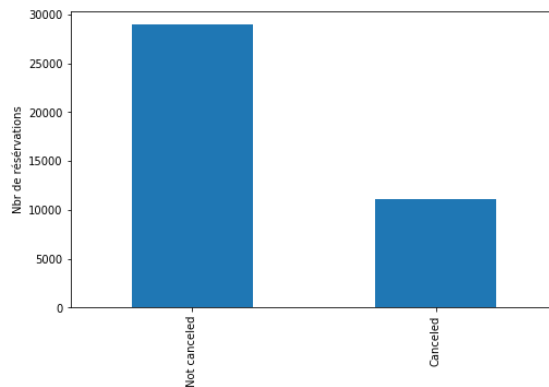
- a. *Pour le City-Hotel :*



Malgré que cet hôtel a connu un grand nombre des client, mais il connaît un grand nombre d'annulation des réservations qui arrive presque à 45000 annulations.

b. Pour Resort Hotel :

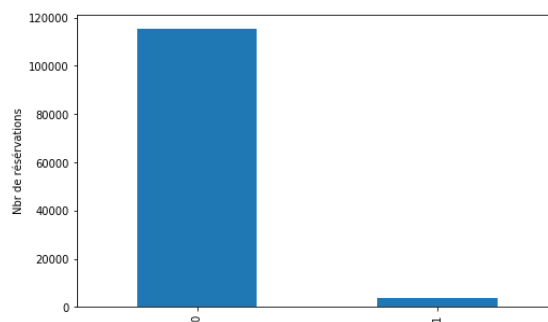
```
Entrée [75]: plt.rcParams["figure.figsize"] = (8,5)
data.loc[data['hotel']=='Resort Hotel','is_canceled'].value_counts().plot(kind='bar')
plt.xticks([0,1],labels=['Not canceled','Canceled'])
plt.ylabel("Nbr de réservations")
plt.show()
```



Cet hôtel a connu presque 30000 annulations.

▪ Clients qui visitent deux fois un hôtel :

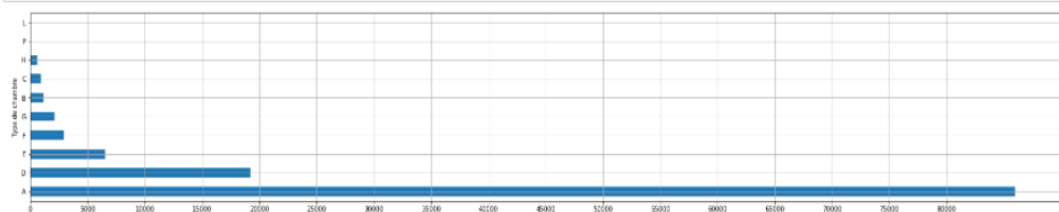
```
Entrée [77]: plt.rcParams["figure.figsize"] = (8,5)
data['is_repeated_guest'].value_counts().plot(kind='bar')
plt.ylabel("Nbr de réservations")
plt.show()
```



On trouve que la totalité des clients ayant leur première visite dans les deux hôtels.

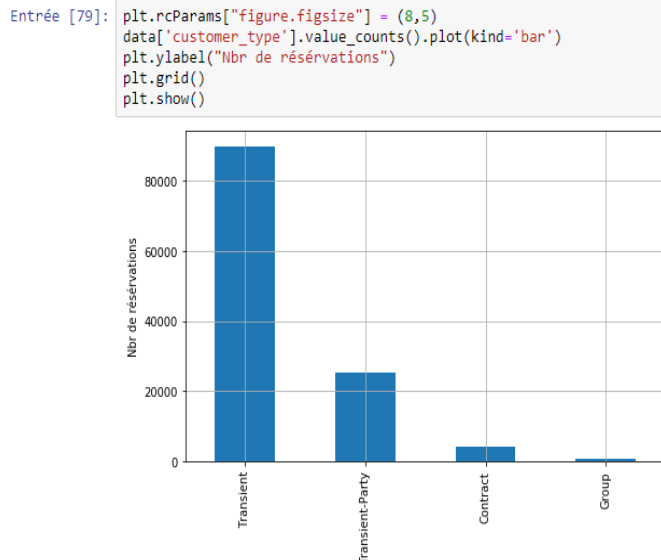
▪ Le nombre des réservations par rapport aux types des chambres :

```
Entrée [78]: plt.rcParams["figure.figsize"] = (30,5)
data['reserved_room_type'].value_counts().plot(kind='barh')
plt.xticks(np.arange(0,85000,5000))
plt.ylabel("Type de chambre")
plt.xlabel("Nbr de réservations")
plt.grid()
plt.show()
plt.show()
```



On trouve que les chambres de type A qui connaît une augmentation au niveau des demandes.

▪ Le nombre des réservations par rapport aux types des clients :



3. Create dataset summary statistics – Integer and numeric variables.

On passe maintenant à créer des statistiques récapitulatives de jeu de données - variables entières et numériques :

▪ Nombre d'enfants présente dans chaque réservation :

3. Create dataset summary statistics – Integer and numeric variables.

Entrée [36]:

```
data['children'].value_counts()
```

Out[36]:

0.0	110319
1.0	4852
2.0	3650
3.0	76
10.0	1

Name: children, dtype: int64

On remarque que la majorité des réservations ne présente aucun enfant.

▪ Nombre des clients qui ont annulé une réservation auparavant :

Entrée [37]:

```
data['previous_cancellations'].value_counts()
```

Out[37]:

0	112451
1	6017
2	113
3	65
24	48
11	35
4	31
26	26
25	25
6	22
19	19
5	19
14	14
13	12
21	1

Name: previous_cancellations, dtype: int64

On trouve que la majorité aussi n'a pas annulé une réservation.

- Nombre de jour entre la réservation et le jour d'arrivée :

```
Entrée [38]: data['lead_time'].describe()
#Nombre de jour entre La réservation et Le jour d'arrivée

Out[38]: count    118898.000000
         mean      104.311435
         std       106.903309
         min        0.000000
         25%       18.000000
         50%       69.000000
         75%      161.000000
         max       737.000000
         Name: lead_time, dtype: float64
```

- Moyennes des transactions faites sur les nombres des nuits dans l'hôtel :

```
Entrée [39]: data['adr'].describe()
#La moyennes des transactions faites sur les nombres des nuits dans L'hôtel

Out[39]: count    118898.000000
         mean      102.003243
         std       50.485862
         min       -6.380000
         25%       70.000000
         50%      95.000000
         75%     126.000000
         max     5400.000000
         Name: adr, dtype: float64
```

On trouve une valeur négative, on a supposé que cette valeur est due a une faute dans les données, alors on essaie de la supprimer :

```
Entrée [40]: data.loc[data['adr']<0]

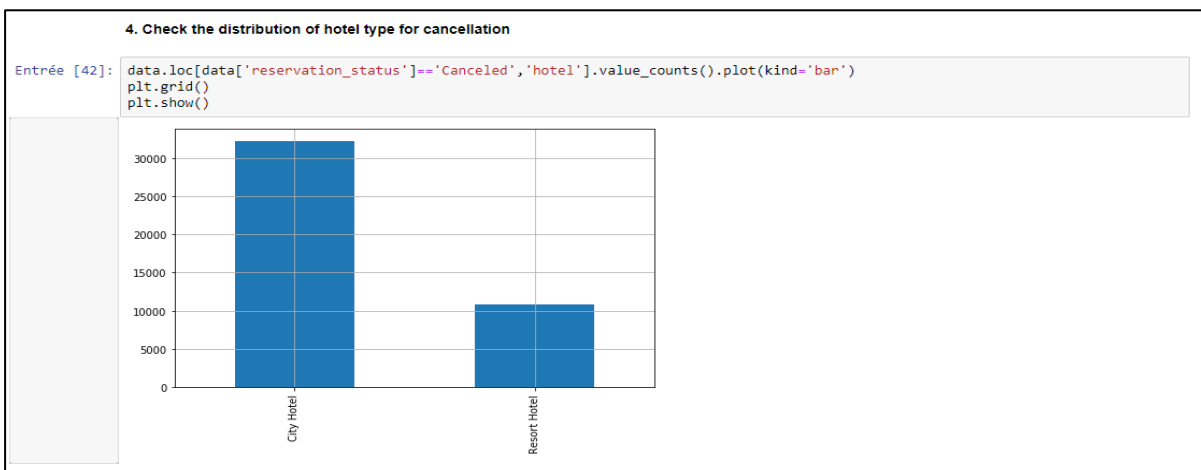
Out[40]:
```

	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
14969	Resort Hotel	0	195	2017	March	10	5	4	4

```
Entrée [41]: data.drop(14969,0,inplace=True)
```

4. Check the distribution of hotel type for cancellation

On travaille maintenant sur la vérification de la répartition du type d'hôtel pour l'annulation :



On observe que le City Hotel a connu la plupart des annulations.

5. Plot distribution of cancellation and Number of Adults

a. Le nombre d'annulation par rapport au nombre d'adultes :

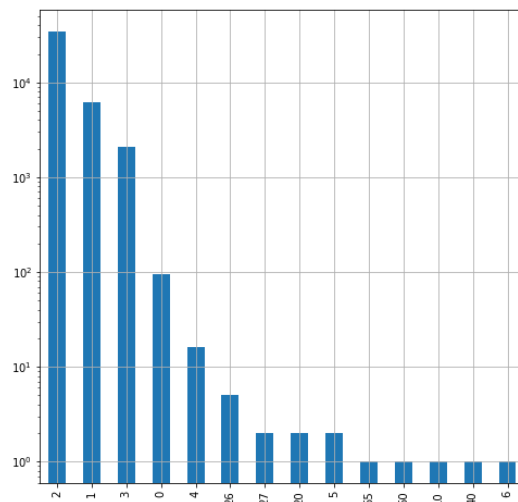
5. Plot distribution of cancellation and Number of Adults

```
Entrée [80]: data.loc[data['reservation_status']=='Canceled', 'adults'].value_counts()

Out[80]:
2    34548
1     6228
3     2104
0       105
4         16
26         5
27         2
20         2
5          2
55         1
50         1
10         1
40         1
6          1
Name: adults, dtype: int64
```

b. Le Plot de la répartition de l'annulation et le nombre d'adultes :

```
Entrée [44]: plt.rcParams["figure.figsize"] = (8,8)
data.loc[data['reservation_status']=='Canceled', 'adults'].value_counts().plot(kind='bar')
plt.yscale("log")
plt.grid()
plt.show()
```



On essaie d'utiliser l'échelle logarithmique a cause de la grande marge entre les valeurs (34548 >> 1).

6. Taking in consideration the characteristics of the variables included in this dataset propose two possible modeling this dataset can have an important role for research and education in revenue management (i.e define two possible target variables and the purpose of each analysis)

Les deux modélisations possibles, qu'on peut proposer pour que cet ensemble de données peut avoir un rôle important pour la recherche et l'éducation dans la gestion des revenus sont :

- a. D'abord, on peut utiliser ses données pour essayer de prédire si un client va annuler sa réservation ou pas, ce qui va permettre aux hôtels de minimiser leurs risques.
- b. Deuxièmement, on peut essayer de comprendre pourquoi les clients revient à notre établissement(en utilisant is_repeated_guest), cela permettra aux hôtels d'améliorer leurs services.

Remarque :

Après qu'on a terminé les analyses statistiques sur les données, on revient à la phase de prétraitement, plus précisément à «**II.5. Propose à preprocessing to be made on this dataset.** » Pour ajouter une autre proposition avant d'aborder la modélisation.

Alors, on commence par la suppression des valeurs manquantes qui restent dans les colonnes **Agent** et **Company** :

II. Data preprocessing

5. Propose a preprocessing to be made on this dataset.

Deuxième méthode

```
Entrée [45]: data.head(1)
```


	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
0	Resort Hotel	0	342	2015	July	27	1	0	0

```

Entrée [46]: data.isna().sum()

Out[46]: hotel 0
is_canceled 0
lead_time 0
arrival_date_year 0
arrival_date_month 0
arrival_date_week_number 0
arrival_date_day_of_month 0
stays_in_weekend_nights 0
stays_in_week_nights 0
adults 0
children 0
babies 0
meal 0
country 0
market_segment 0
distribution_channel 0
is_repeated_guest 0
previous_cancellations 0
previous_bookings_not_canceled 0
reserved_room_type 0
assigned_room_type 0
booking_changes 0
deposit_type 0
agent 16004
company 112274
days_in_waiting_list 0
customer_type 0
adr 0
required_car_parking_spaces 0
total_of_special_requests 0
reservation_status 0
reservation_status_date 0
arrival_date 0
count 0
dtype: int64

```



On essaie de le remplacer par des -1 et des 0 comme la montre la figure ci-dessous :

```
Entrée [47]: data['company'].fillna(-1,inplace=True)

Entrée [48]: data['agent'].fillna(0,inplace=True)

Entrée [49]: data.isna().sum()

Out[49]: hotel 0
is_canceled 0
lead_time 0
arrival_date_year 0
arrival_date_month 0
arrival_date_week_number 0
arrival_date_day_of_month 0
stays_in_weekend_nights 0
stays_in_week_nights 0
adults 0
children 0
babies 0
meal 0
country 0
market_segment 0
distribution_channel 0
is_repeated_guest 0
previous_cancellations 0
previous_bookings_not_canceled 0
reserved_room_type 0
assigned_room_type 0
booking_changes 0
deposit_type 0
agent 0
company 0
days_in_waiting_list 0
customer_type 0
adr 0
required_car_parking_spaces 0
total_of_special_requests 0
reservation_status 0
reservation_status_date 0
arrival_date 0
count 0
dtype: int64

```

Ensuite, pour les colonnes qui représentent des dates, vu que les méthodes de machine Learning n'acceptent pas les date comme input de cette format « année-mois-jour » :

Entrée [51]: `data.head(1)`

Out[51]:

mpany	days_in_waiting_list	customer_type	adr	required_car_parking_spaces	total_of_special_requests	reservation_status	reservation_status_date	arrival_date
-1.0	0	Transient	0.0	0	0	Check-Out	2015-07-01	2015-07-01

Alors on essaie de les diviser comme l'énoncé a présenté:

Entrée [52]:

```
mon=data['arrival_date_month'].unique()
import datetime
month=[]
for i in mon:
    month.append(datetime.datetime.strptime(i[:3], "%b").month)
months={}
for i in range(len(mon)):
    months[mon[i]]=month[i]
```

Entrée [53]:

```
months={}
for i in range(len(mon)):
    months[mon[i]]=month[i]
```

Entrée [54]: `data['arrival_date_month'] = data['arrival_date_month'].map(months)`
`data.head(1)`

Out[54]:

	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
0	Resort Hotel	0	342	2015	7	27	1	0	

Entrée [55]:

```
data['reservation_status_date_day'] = pd.DatetimeIndex(data['reservation_status_date']).day
data['reservation_status_date_year'] = pd.DatetimeIndex(data['reservation_status_date']).year
data['reservation_status_date_month'] = pd.DatetimeIndex(data['reservation_status_date']).month
```

Entrée [56]: `data.head(1)`

Out[56]:

	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
0	Resort Hotel	0	342	2015	7	27	1	0	

1 rows x 36 columns

Entrée [57]: `data.drop(['arrival_date', 'reservation_status_date'],1,inplace=True)`

Et voilà maintenant notre résultat :

Entrée [58]: `data.head(1)`

Out[58]:

	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
0	Resort Hotel	0	342	2015	7	27	1	0	

Entrée [59]: `get_d = list(set(categorical)-set(['is_canceled', 'is_repeated_guest', 'reservation_status']))`
`data[get_d].dtypes`

Out[59]:

country	object
assigned_room_type	object
distribution_channel	object
deposit_type	object
customer_type	object
hotel	object
meal	object
reserved_room_type	object
market_segment	object
dtype:	object

IV. Modeling

Comme toute entreprise, les hôtels cherchent également à réaliser des bénéfices. Alors on va proposer un modèle qui prédit si la réservation est susceptible d'être annulée, ce qui pourrait être une bonne indication pour les hôtels, car ils préfèrent peut-être accepter en premier les réservations à faible risque.

➤ On commence à ajouter une colonne, nommée Y, qui présente si un client va ou pas annuler sa réservation, ainsi qu'on va supprimer la colonne « reservation_statut » car grâce à la matrice de corrélation, on trouve qu'elle contient sur la même information que la nouvelle colonne Y:

IV. Modeling

Entrée [60]:

```
pd.get_dummies(data[['is_canceled', 'reservation_status']], columns=['reservation_status']).corr()
#On supprime les deux colonnes, vu que reservation_status contient la réponse
```

Out[60]:

	is_canceled	reservation_status_Canceled	reservation_status_Check-Out	reservation_status_No-Show
is_canceled	1.000000	0.978440	-1.000000	0.131542
reservation_status_Canceled	0.978440	1.000000	-0.978440	-0.076029
reservation_status_Check-Out	-1.000000	-0.978440	1.000000	-0.131542
reservation_status_No-Show	0.131542	-0.076029	-0.131542	1.000000

Entrée [61]:

```
Y=data['is_canceled']
data.drop(['is_canceled', 'reservation_status'],1,inplace=True)
```

Entrée [62]:

```
X=pd.get_dummies(data,columns=get_d,drop_first=True)
```

➤ On a utilisé deux modélisations :

- La régression logistique :

Entrée [63]:

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import xgboost as xgb
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

Entrée [65]:

```
from sklearn.linear_model import LogisticRegression
```

Entrée [66]:

```
clf = LogisticRegression(random_state=0,C=1,solver='liblinear',max_iter=700,tol=1e-9)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[66]:

```
0.8542472666105971
```

On a eu 85% des réponses correctes. Pour les choix des paramètres n'est pas montré dans notre Notebook pour des raisons de visibilité.

ii. Xgboost :

Ce modèle est plus robuste que le premier :

```
Entrée [67]: import xgboost as xgb

Entrée [68]: clf_xgb=xgb.XGBClassifier( learning_rate =0.05, n_estimators=800, max_depth=7,
    min_child_weight=5, gamma=0.3, subsample=0.8, colsample_bytree=0.7,
    objective= 'binary:logistic', scale_pos_weight=0.6,seed=27,n_jobs=-1)
clf_xgb.fit(X_train, y_train)
y_pred=clf_xgb.predict(X_test)
accuracy_score(y_test,y_pred)

C:\Users\Taha\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_classes - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[14:27:20] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

Out[68]: 0.9994953742640875
```

Pour ce modèle, on a eu un **accuracy_score** de 99%, qui présente un score très élevé.



On a essayé d'additionner les deux modèles pour avoir un modèle plus stable, avec moins de variance que présente le modèle Xgboost :

```
Entrée [69]: y_pred=clf_xgb.predict_proba(X_test)*0.8+clf.predict_proba(X_test)*0.2
y_pred = y_pred[:,0]
for i in range(len(y_pred)):
    if y_pred[i]<=0.5:
        y_pred[i]=1
    else:
        y_pred[i]=0
accuracy_score(y_test,y_pred)

Out[69]: 0.9991589571068125
```

A la fin, on a un modèle qui prédit si un client va ou pas annuler une réservation avec un score de 99%