INSTITUT NATIONAL DES POSTES ET TÉLÉCOMMUNICATION

Demandes de réservation d'hôtel
Etude de cas

Réalisé par : ARROUD YASSIR

21er mai 2021

Table des matières

1 Traitement des données :

1.1 Prétraitement de données : 4

1.2 Exploration des données : 7

1.3 Analyse de données : 10

2) Machine Learning :

2.1 Is cancelled ciblé : . . . . . 15. .

2.2 ADR ciblé : . . . . . 18.

Chapitre 1

Traitement des données :

1.1 Prétraitement de données :

On commence par l'importation des libraires et des modules, ensuite on importe notre jeu de données « hotel_booking.csv » , qui contient 119 390 lignes et 32 colonnes.

```
: df = pd.read_csv('hotel_bookings.csv')
df.head()
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nigh
0	Resort Hotel	0	342	2015	July	27	1	
1	Resort Hotel	0	737	2015	July	27	1	
2	Resort Hotel	0	7	2015	July	27	1	
3	Resort Hotel	0	13	2015	July	27	1	
4	Resort Hotel	0	14	2015	July	27	1	

5 rows × 32 columns

```
: df = pd.read_csv('hotel_bookings.csv')
df.shape
```

```
(119390, 32)
```

1.1 Prétraitement de données :

On vérifie les valeurs manquantes chez chaque variable de notre jeux de données.

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

```
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
```

1.1 Prétraitement de données :

On supprime les lignes ayant des valeurs manquantes à l'exception des variables « Agent » et « Company » , car 'NULL' est présentée comme l'une de ses valeurs.

On Vérifie que l'horodatage de la variable reservation_status_date doit apparaître après ou à la même date que l'entrée variable arrival_date

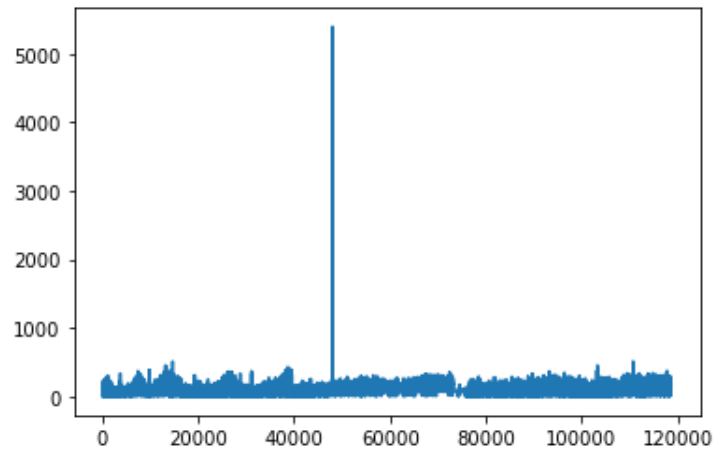
```
df1['arrival_date']= pd.to_datetime(df1['arrival_date'])
## If no id of agent or company is null, just replace it with 0
df1[['agent','company']] = df1[['agent','company']].fillna(0.0)
df1['country'].fillna(df.country.mode().to_string(), inplace=True)


df1['children'].fillna(round(df.children.mean()), inplace=True)
df1 = df1.drop(df1[(df1.adults+df1.babies+df1.children)==0].index)
df1[['children', 'company', 'agent']] = df1[['children', 'company', 'agent']].astype('int64')
data = df1.copy()
```

1.1 Prétraitement de données :

On supprime les lignes ayant des valeurs manquantes à l'exception des variables « Agent » et « Company » , car 'NULL' est présentée comme l'une de ses valeurs.

```
: plt.plot(data['adr'].values)
[<matplotlib.lines.Line2D at 0x24b805cc8b0>]
```



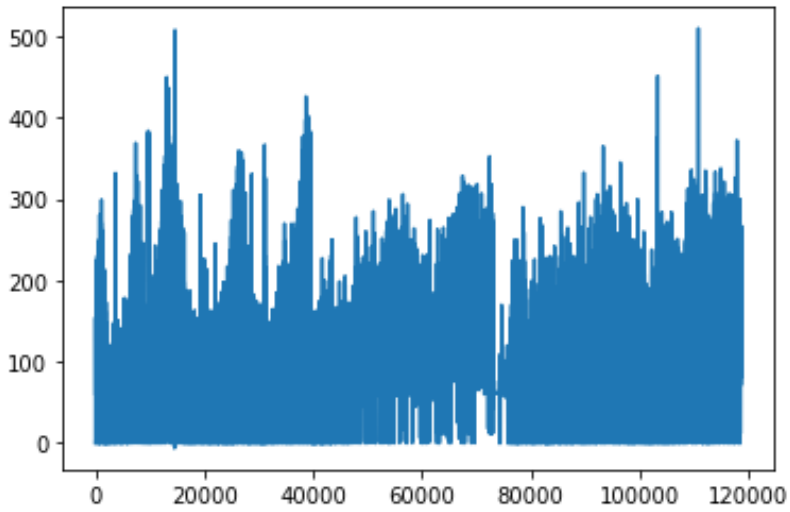
1.2 Exploration des données

On étudie le rapport de corrélation entre nos variables.

```
max_adr = (max(data['adr']))
data.drop(data.loc[data['adr'] == max_adr].index, inplace = True)
plt.plot(data['adr'].values)
```

```
: plt.plot(data['adr'].values)
```

```
[<matplotlib.lines.Line2D at 0x24b808443a0>]
```



1.2 Standardize the ADR feature

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample x is calculated as:

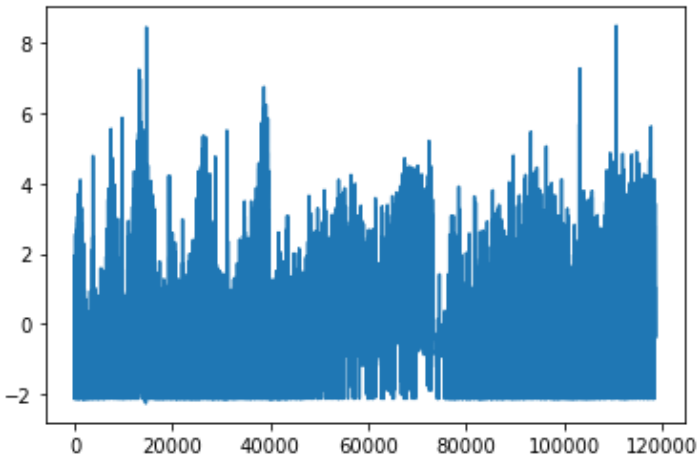
z = (x - u) / s

1.2 Exploration des données

On étudie le rapport de corrélation entre nos variables.

```
|: cols_to_norm = ['adr']
   data[cols_to_norm] = StandardScaler().fit_transform(data[cols_to_norm])
   plt.plot(data['adr'].values)
```

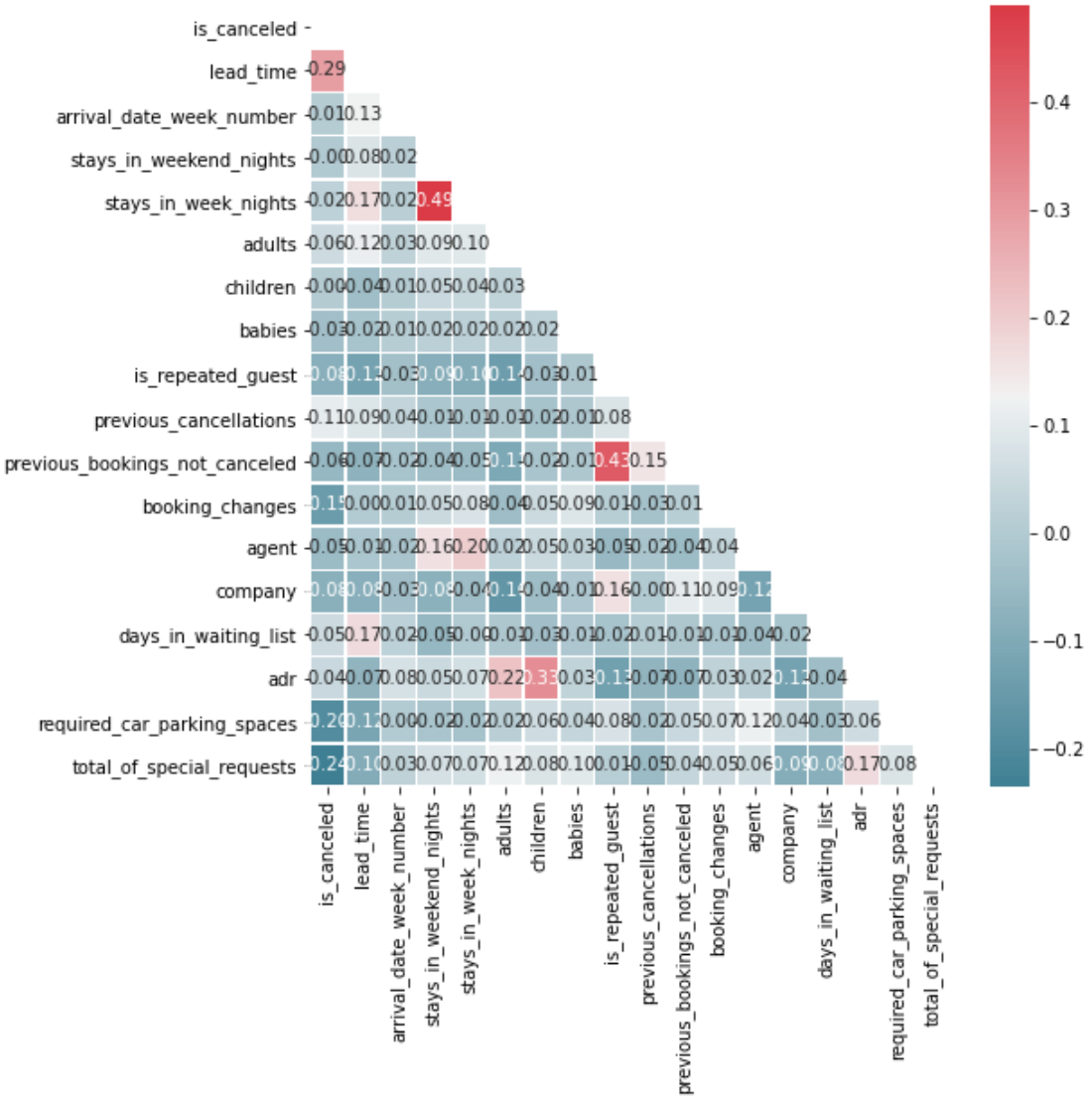
```
[<matplotlib.lines.Line2D at 0x24b80342520>]
```



By checking with just eye it looks that mean is near to zero and std is near to 1, we can verify it

1.2 Exploration des données

On étudie le rapport de corrélation entre nos variables.



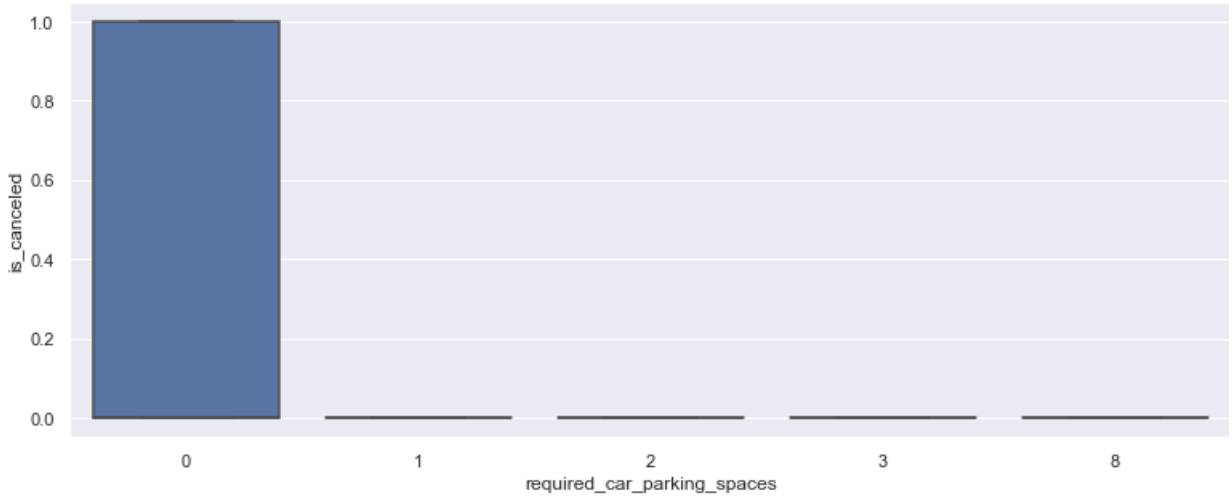
Sur la base des codes ci-dessus, nous pouvons conclure que les fonctionnalités qui ont la plus grande corrélation avec la cible ('is_cancelled') sont required_car_parking_spaces, lead_time, booking_changes, adr et adults.

FIGURE 1.6

1.2 Analyse de données

On étudie en bref la relation entre notre variable de réponse 'is_canceled' et les variables qui y sont fort correllées .

1.3.1 Is_canceled & required_car_parking_spaces




```
: data[data['required_car_parking_spaces'] == 0]['is_canceled'].value_counts()
```

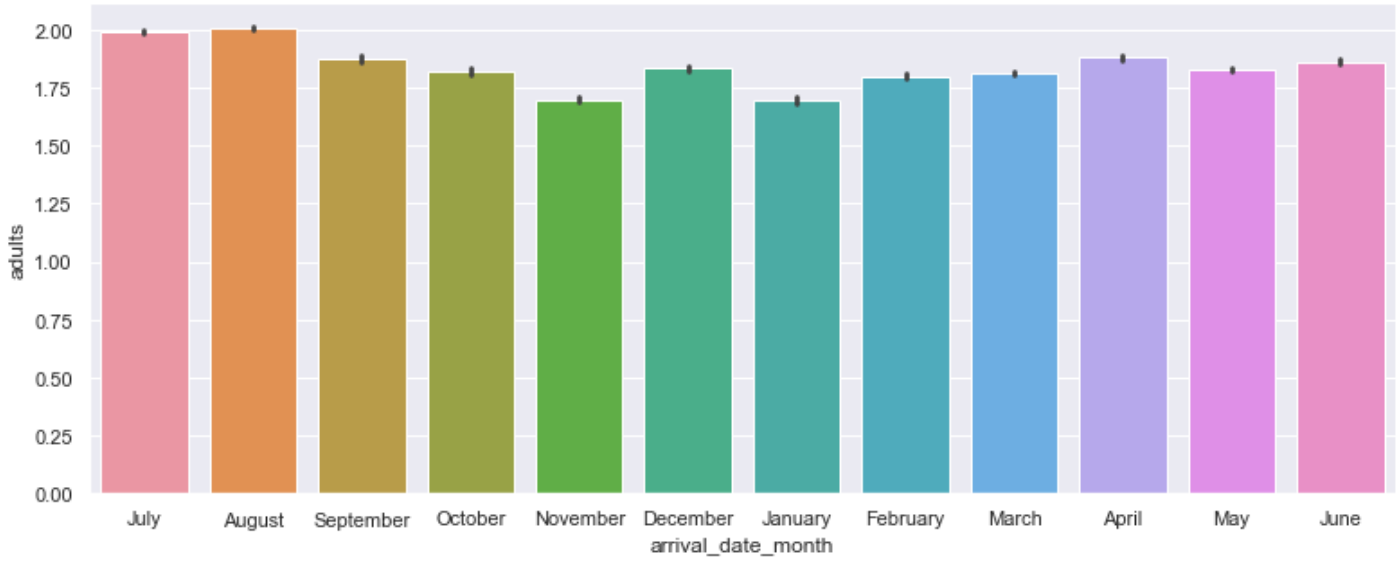
```
0    67287
1    44138
Name: is_canceled, dtype: int64
```

D'après les visualisations ci-dessus, la plupart des réservations annulées provenaient de ceux qui n'ont besoin que d'une place de parking.

FIGURE 1.6

1.3 Analyse de données

1.3.2 Is_canceled & required_car_parking_spaces




```
: data[data['arrival_date_month'] == "November"]['adults'].mean()
```

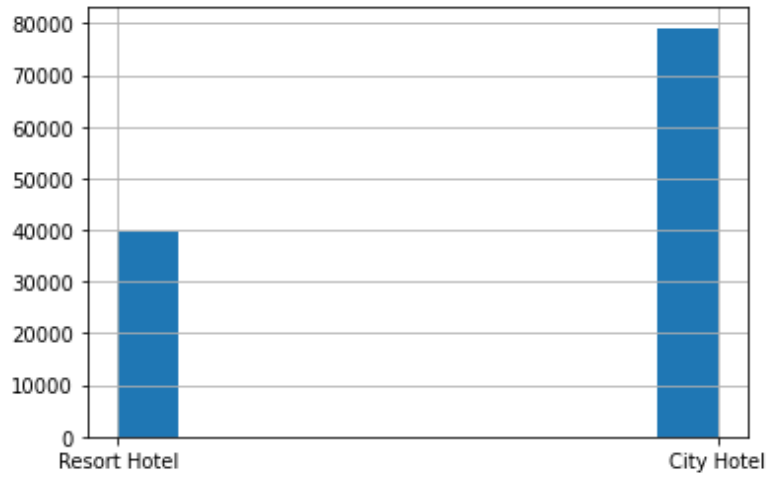
```
1.6994503045609866
```

Sur la base du graphique ci-dessus, la plupart des adultes arrivent dans leurs hôtels en juillet et août, avec plus de 2,0 adultes. Le barplot nous donne la moyenne des adultes et non leur somme.

FIGURE 1.6

1.3 Analyse de données

Combien de Resort Hotels and City Hotels nous disposons?



Nous pouvons voir que la majorité de nos données se trouvent dans les hôtels de la ville, nous allons diviser nos données entre les bases de données de la station et de la ville (resort et city_ht)

```
resort = data[data.hotel == "Resort Hotel"].drop("hotel",axis = 1)
city_ht = data[data.hotel == "City Hotel"].drop("hotel",axis = 1)
```

FIGURE 1.6

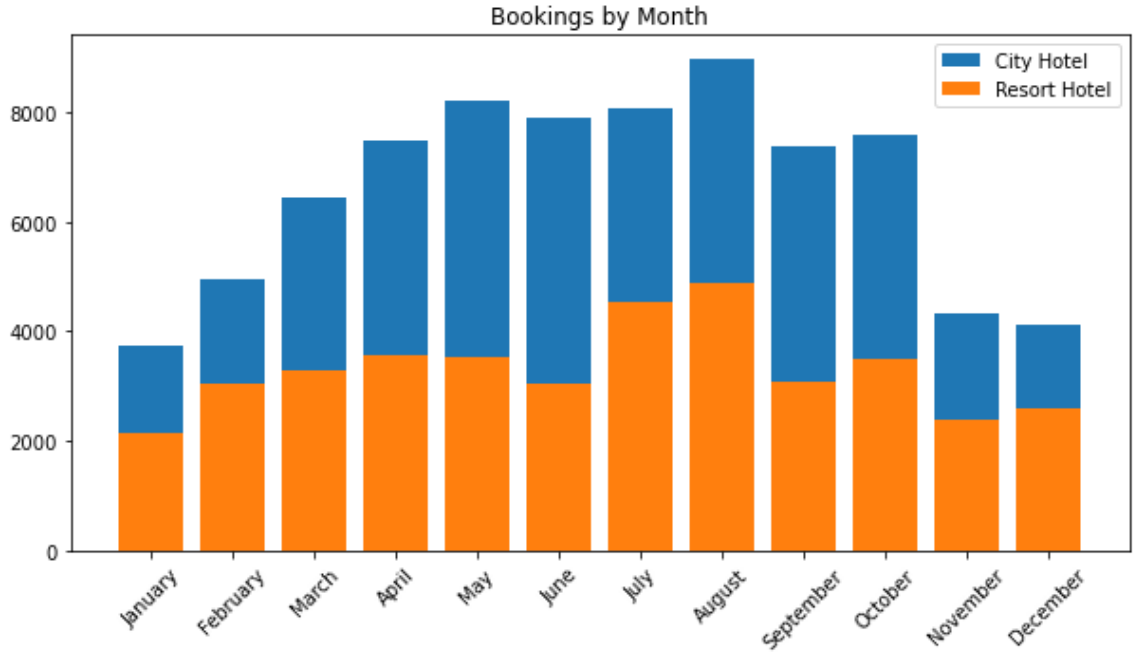
1.3 Analyse de données

Nous voyons que le mois d'août a la majorité des réservations, à la fois pour les hôtels de la ville et les hôtels de villégiature. Nous pourrions classer les saisons comme:

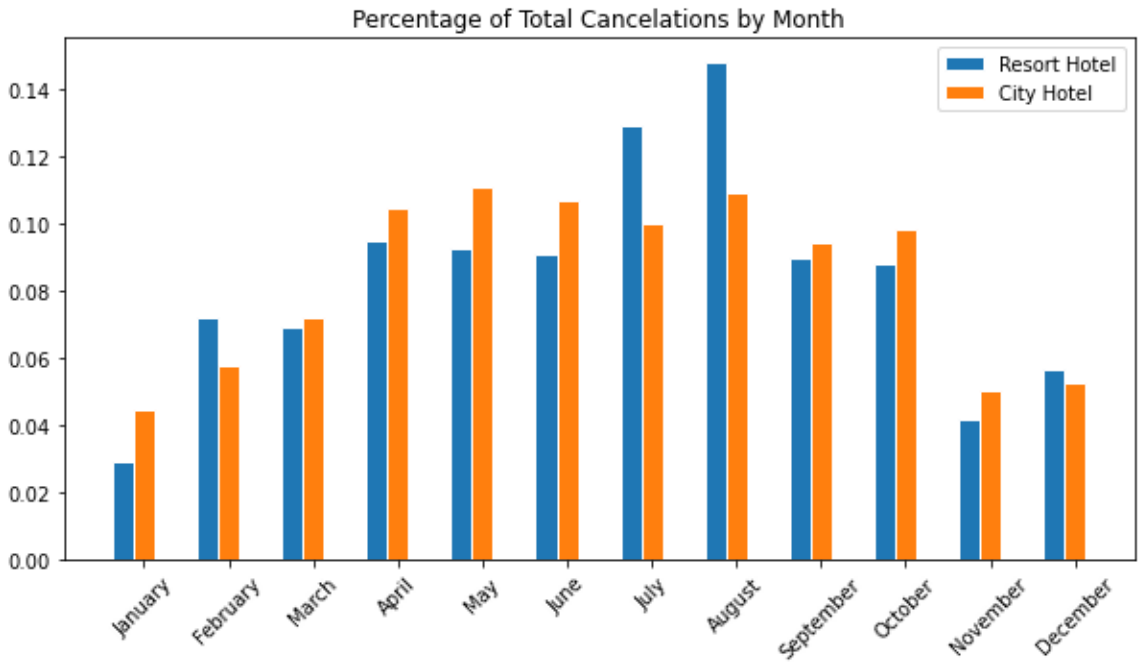
- basse saison: __ janvier, décembre, novembre, février__

- moyenne saison: __Mars, septembre, avril, octobre__

- haute saison: __ juin, juillet, mai, août__



1.3 Analyse de données



Comme on peut le voir, la plupart des annulations se produisent en haute saison. Ce qui est attendu, car la plupart des réservations sont en cours.

II) Machine Learning :

Après l'importation, le prétraitement et l'exploration de notre jeu de données, vient l'étape attendu ' Construction de modèles d'apprentissage automatique et évaluation de leurs résultats '

1) Is_canceled ciblé

1.1 CONSTRUCTION DE MODÈLES ET RÉGLAGE DES HYPERPARAMÈTRES :

Tout d'abord, on se contente des 5 variables les plus corrélés à notre variable cible ' is_canceled ', ensuite on divise notre données en 2 tranches : jeux de données d'entrainement et jeux de données de test, ensuite on applique la régression logistique sur la variable de réponse ' is_canceled ' .

```
df2 = data.copy()
data2 = df2[['is_canceled','required_car_parking_spaces','lead_time','booking_changes','adr','adults']]
data3 = data2.copy()
X = data3.drop(['is_canceled'], axis=1)
y = data3['is_canceled']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, shuffle=False)
log = LogisticRegression().fit(X_train, y_train)
print("log.coef_: {}".format(log.coef_))
print("log.intercept_: {}".format(log.intercept_))
```

```
log.coef_: [[-4.05779633  0.00655547 -0.77285235  0.0097256   0.08433395]]
log.intercept_: [-2.00015595]
```

```
: y_pred = log.predict(X_test)
```

II) Machine Learning :

1) Is_canceled ciblé

Model Performance

```
: print("Training set score: {:.2f}".format(log.score(X_train, y_train)))
  print("Test set score: {:.2f}".format(log.score(X_test, y_test)))
```

```
Training set score: 0.69
Test set score: 0.69
```

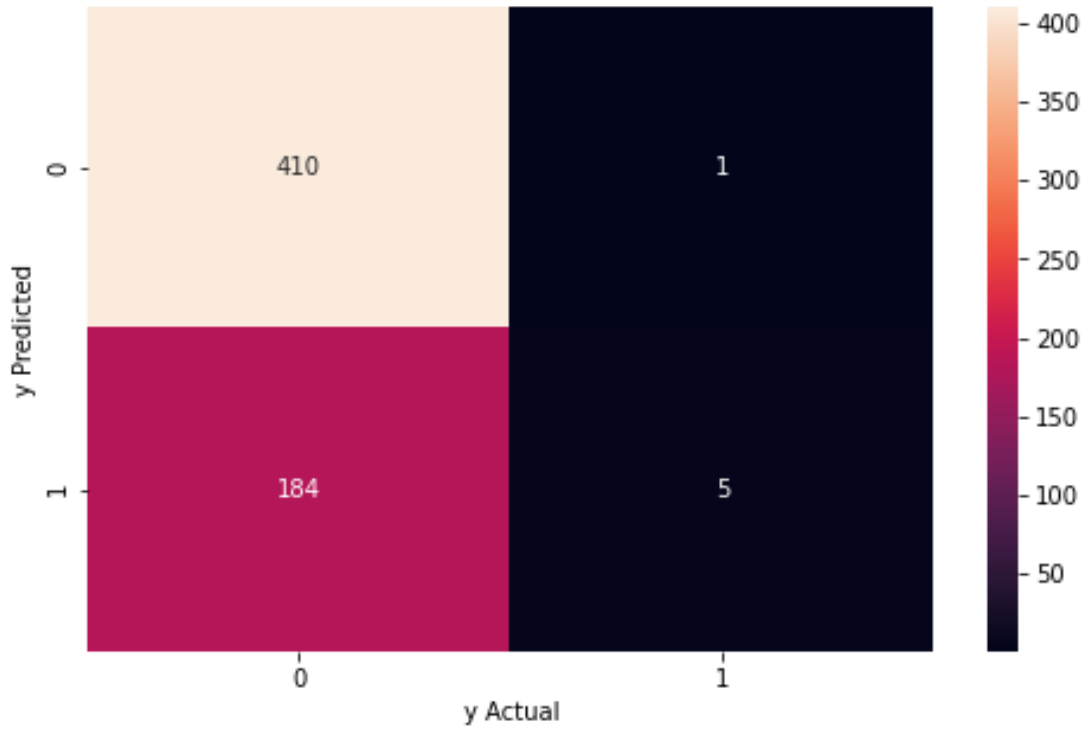
1.2 Evaluation Model :

On commence par la matrice de confusion

```
: confusion_matrix(y_test,y_pred)
```

```
array([[410,   1],
       [184,   5]], dtype=int64)
```

Dans cette partie on va se contenter de metre les résultas après l'execution du code, puisque c'est la chose la plus inmportante de cette etude, pour plus de détails veuillez consulter le code Python



Sur ce modèle, nous pouvons voir que False Negative (Predicted 0, Actual 1). C'est quelque chose dont nous devons nous préoccuper, car je pense que c'est quelque chose qui pourrait entraîner le plus de pertes pour l'entreprise. Si cela se produit, l'entreprise prédit à tort que quelqu'un a effectivement annulé le livre, mais a noté qu'elle n'annulait pas ses réservations. Pour cette raison, l'entreprise n'a pas gagné d'argent parce que ces personnes ne l'ont pas payée.

II) Machine Learning :

2) ADR ciblé

2.1 CONSTRUCTION DE MODÈLES ET RÉGLAGE DES HYPERPARAMÈTRES :

Déployez un modèle d'apprentissage automatique qui prédit l'ADR en fonction de l'ensemble de fonctionnalités que vous jugez importantes. Expliquez vos choix pour les fonctionnalités et le type de modèle ML. Si vous en utilisez, comment pouvez-vous gérer les fonctionnalités catégorielles?

Q: comment pouvez-vous gérer les fonctionnalités catégorielles?

Réponse: J'ai utilisé la méthode Label Encoder pour gérer les fonctionnalités catégorielles. Il convertira la valeur des étiquettes de type de chaîne en

entre 0 et n_classes-1.

```
from sklearn.preprocessing import LabelEncoder

ADR_Dateset = data['adr'].values

data.drop('adr', axis=1, inplace=True)

data_LabelEncoded=data.apply(LabelEncoder().fit_transform) # Applying label encoder

feature_ADR=data_LabelEncoded.values  # ADR feature is y and all other are X

X_train, X_test, y_train, y_test = train_test_split(feature_ADR, ADR_Dateset, random_state=42)
# we split the dataset between train and test
```

II) Machine Learning :

2) ADR ciblé

2.1 CONSTRUCTION DE MODÈLES ET RÉGLAGE DES HYPERPARAMÈTRES :

Linear Regression for ADR prediction

```
:  from sklearn.linear_model import LinearRegression

   regressor = LinearRegression()
   linear_reg = regressor.fit(X_train, y_train)
   score_training = linear_reg.score(X_train, y_train)
   score_test = linear_reg.score(X_test, y_test)

   print("Training set score: {:.2f}".format(score_training))
   print("Test set score: {:.2f}".format(score_test))
```

```
Training set score: 0.40
Test set score: 0.40
```

2-degree Poly Regreesion for ADR prediction

Comme le score du test est bien moindre, c'est pourquoi nous avons appliqué la régression polynomiale pour augmenter les chances d'un bon ajustement du modèle.

II) Machine Learning :

2) ADR ciblé

2.1 CONSTRUCTION DE MODÈLES ET RÉGLAGE DES HYPERPARAMÈTRES :

Applying Random forest regressor

```
: X_train, X_test, y_train, y_test = train_test_split(feature_ADR, ADR_Dateset, random_state=42)
```

```
: # Fitting the Random Forest Regression Model to the dataset

from sklearn.ensemble import RandomForestRegressor
regressor= RandomForestRegressor(n_estimators=10,random_state=0)
RF_regressor=regressor.fit(X_train,y_train)

score_training = RF_regressor.score(X_train, y_train)
score_test = RF_regressor.score(X_test, y_test)

print("Training set score: {:.2f}".format(score_training))
print("Test set score: {:.2f}".format(score_test))
```

```
Training set score: 0.98
Test set score: 0.89
```

Somme des erreurs au carré (prédite vs réelle) | Fonction de score de régression R ^ 2 (coefficient de détermination)

```
: predicted=RF_regressor.predict(X_test)
  squared_errors = (y_test - predicted) ** 2

  print ('squared_errors:',np.sum(squared_errors))



  from sklearn.metrics import r2_score
  print ('R2 score:',r2_score(y_test, predicted))
```

```
squared_errors: 3386.5469722643084
R2 score: 0.8854006704852415
```

II) Machine Learning :

```
: #polynomial regression on dataset

from sklearn.preprocessing import PolynomialFeatures

poly_regressor=PolynomialFeatures(degree=2) # initilization of 2d polynomail features
X_poly=poly_regressor.fit_transform(X_train) # Converting our features to 2 degree

linear_regressor=LinearRegression()
linear_reg_poly=linear_regressor.fit(X_poly,y_train)


score_training = linear_reg_poly.score(X_poly, y_train)

X_test_poly=poly_regressor.fit_transform(X_test)
score_test = linear_reg_poly.score(X_test_poly, y_test)

print("Training set score: {:.2f}".format(score_training))
print("Test set score: {:.2f}".format(score_test))
```

```
Training set score: 0.65
Test set score: 0.65
```

2.2 Model evaluation

Nous pouvons observer que le score a été augmenté à partir d'une simple régression linéaire multiple lorsque la régression polynomiale est appliquée. Mais toujours son plus bas. Une chose peut être remarquée que le modèle n'est toujours pas trop ajusté.