Data Mining

Khaoula Tlili

3 DNI 1

Table of Contents

01

Introduction

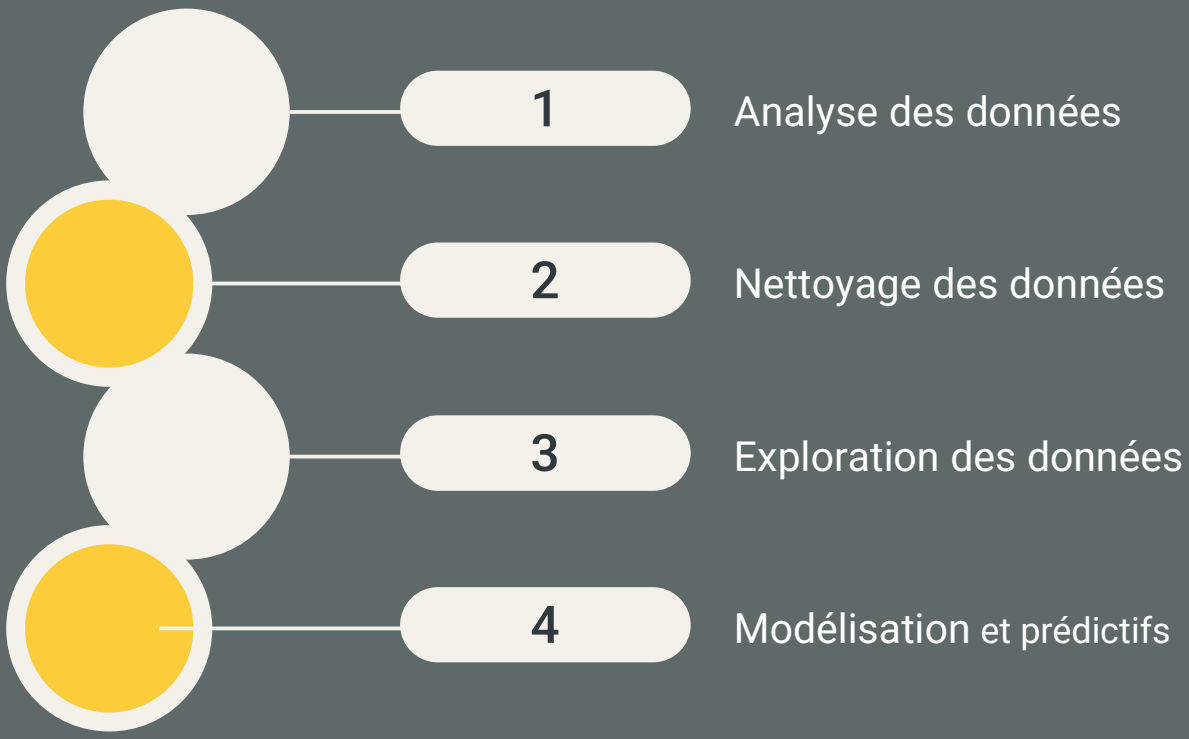
Introduction

Les données jouent un rôle essentiel dans la prise de décisions éclairées et l'optimisation des opérations dans divers secteurs. Deux ensembles de données, "Hotel Data" et "Sales Car," fournissent une opportunité précieuse pour l'analyse et la compréhension des tendances et des facteurs influençant les réservations d'hôtels et les prix de vente des voitures d'occasion. Ce rapport se penche sur ces deux jeux de données, les examinant en détail et offrant un aperçu des analyses, des méthodes de nettoyage des données et des modèles potentiels.

02

Objectif de l’analyse

Objectif de l’analyse...



03

Description des Datasets

Description des Datasets

01

Hotel Data

02

Sales Car

3.1. Hotel Data

Cet ensemble de données capture les détails des réservations d'hôtels, incluant des informations telles que les dates d'arrivée et de départ, le type de chambre, le nombre de personnes, le nombre d'enfants, des détails sur les clients, le statut de la réservation (annulée ou non), et d'autres paramètres pertinents.

Importation des bibliothèques

Importation des bibliothèques pandas et datetime

In [50]:
```
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
```

Chargement de fichier

Chargement des données depuis un fichier CSV

In [51]:
```
data = pd.read_csv("hotel.csv")
```

Informations sur les données

Affichage des noms des colonnes

```
In [52]: data.columns

Out[52]: 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', 'agent',
                'company', 'days_in_waiting_list', 'customer_type', 'adr',
                'required_car_parking_spaces', 'total_of_special_requests',
                'reservation_status', 'reservation_status_date'],
               dtype='object')
```

Affichage des 10 premières lignes

```
In [54]: data.head(10)
```

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_i
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	
5	Resort Hotel	0	14	2015	July	27	1	0	
6	Resort Hotel	0	0	2015	July	27	1	0	

3.2. Sales Car

Cet ensemble de données offre un aperçu détaillé des ventes de voitures d'occasion, présentant des informations sur la marque de la voiture, le modèle, l'année de fabrication, le kilométrage, le prix de vente et l'emplacement de la vente.

Chargement de fichier

Chargement des données depuis un fichier CSV

```
dataRg = pd.read_csv('Car_sales.csv')
```

Informations sur les données

Information sur data

```
In [135]: dataRg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Manufacturer         157 non-null    object
 1   Model                157 non-null    object
 2   Sales_in_thousands   157 non-null    float64
 3   __year_resale_value  121 non-null    float64
 4   Vehicle_type         157 non-null    object
 5   Price_in_thousands   155 non-null    float64
 6   Engine_size          156 non-null    float64
 7   Horsepower           156 non-null    float64
 8   Wheelbase            156 non-null    float64
 9   Width                156 non-null    float64
 10  Length               156 non-null    float64
 11  Curb_weight          155 non-null    float64
 12  Fuel_capacity        156 non-null    float64
 13  Fuel_efficiency      154 non-null    float64
 14  Latest_Launch        157 non-null    object
 15  Power_perf_factor    155 non-null    float64
dtypes: float64(12), object(4)
```

04

Nettoyage des données

Nettoyage des données

Le nettoyage des données est une étape cruciale pour garantir la fiabilité des analyses. Les données manquantes, les valeurs aberrantes et d'autres problèmes potentiels seront traités dans les sections de nettoyage des données spécifiques à chaque ensemble.

Nettoyage des données pour Hotel Data

Sélection des colonnes à supprimer

In [58]:
```
cols_to_drop = ['arrival_date_week_number', 'babies', 'distribution_channel', \
                'previous_cancellations', 'previous_bookings_not_canceled', 'booking_changes', \
                'deposit_type', 'agent', 'company', 'required_car_parking_spaces', \
                'total_of_special_requests', 'reservation_status', 'reservation_status_date']
data_clean = data.drop(cols_to_drop, axis=1)
# Suppression des colonnes inutiles
del cols_to_drop
```

Suppression des lignes contenant des valeurs manquantes (NaN) et affichage de s infos

71]:
```
data_clean.dropna(inplace=True)
data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 118898 entries, 0 to 119389
Data columns (total 19 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   hotel                       118898 non-null  object
 1   is_canceled                 118898 non-null  int64
 2   lead_time                   118898 non-null  int64
 3   arrival_date_year           118898 non-null  int64
 4   arrival_date_month          118898 non-null  object
 5   arrival_date_day_of_month   118898 non-null  int64
 6   stays_in_weekend_nights     118898 non-null  int64
 7   stays_in_week_nights        118898 non-null  int64
 8   adults                      118898 non-null  int64
 9   children                    118898 non-null  float64
 10  meal                        118898 non-null  object
 11  country                     118898 non-null  object
 12  market_segment              118898 non-null  object
 13  is_repeated_guest           118898 non-null  int64
 14  reserved_room_type          118898 non-null  object
```

Conversion de la colonne 'children' en type entier

```
72]: data_clean['children'].astype('int64')
```

```
72]: 0         0
     1         0
     2         0
     3         0
     4         0
              ..
     119385    0
```

Création d'une nouvelle colonne 'guest_number' en combinant 'adults' et 'children'

```
[73]: data_clean['guest_number'] = data_clean['adults'] + data_clean['children']
```

```
[74]: data_clean.head()
```

[74]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	chil
0	Resort Hotel	0	342	2015	July	1	0	0	2	
1	Resort Hotel	0	737	2015	July	1	0	0	2	
2	Resort Hotel	0	7	2015	July	1	0	1	1	
3	Resort Hotel	0	13	2015	July	1	0	1	1	
4	Resort Hotel	0	14	2015	July	1	0	2	2	

Conversion du mois de texte en format numérique

```
[84]: month_map = {'January': 1, 'February': 2, 'March': 3, 'April': 4, 'May': 5, 'June': 6, 'July': 7, 'August': 8, \
                   'September': 9, 'October': 10, 'November': 11, 'December': 12}
      data_clean['arrival_date_month'] = data_clean['arrival_date_month'].map(month_map)
      data_clean.head()
```

[84]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_day_of_month	meal	country	market_segment	is_repeated_guest	reserved
2	Resort Hotel	0	7	2015	7	1	BB	GBR	Direct	0	
3	Resort Hotel	0	13	2015	7	1	BB	GBR	Corporate	0	
4	Resort Hotel	0	14	2015	7	1	BB	GBR	Online TA	0	
5	Resort Hotel	0	14	2015	7	1	BB	GBR	Online TA	0	
6	Resort Hotel	0	0	2015	7	1	BB	PRT	Direct	0	

Conversion de la colonne 'date' en format de date datetime

[86]:
```
def convert_dt(cell):
    tmp = cell.split('/')
    tmp2 = [int(tt) for tt in tmp]
    return dt.datetime(tmp2[0], tmp2[1], tmp2[2])

data_clean['date'] = data_clean['date'].apply(convert_dt)
data_clean['date'].head()
```

t[86]:
```
2   2015-07-01
3   2015-07-01
4   2015-07-01
5   2015-07-01
6   2015-07-01
Name: date, dtype: datetime64[ns]
```

Filtrage des données

[90]:
```
# Limit the analysis to exactly two years: from 1 July 2015 to 1 July 2017
data_clean.drop(data_clean[data_clean['date']>dt.datetime(2017, 6, 30)].index, inplace=True)
```

Attribution des valeurs numériques '1' et '2' pour l'année d'arrivée (1 pour la première année, 2 pour la deuxième)

[97]:
```
data_clean['arrival_date_year'][(data_clean['date']>=dt.datetime(2015, 7, 1)) & (data_clean['date']<=dt.datetime(2016, 6, 30))] =
data_clean['arrival_date_year'][(data_clean['date']>=dt.datetime(2016, 7, 1)) & (data_clean['date']<=dt.datetime(2017, 6, 30))] =
```

Filtrage des données en fonction du type d'hôtel ¶

[162]:
```
# filtre les données pour ne conserver que celles où le type d'hôtel est "Resort Hotel"
resort_h = data_clean[data_clean['hotel']=='Resort Hotel']
#cette ligne filtre les données pour ne conserver que celles où le type d'hôtel est "City Hotel"
city_h = data_clean[data_clean['hotel']=='City Hotel']
```

suppression des colonnes 'index' et 'hotel'

[102]:
```
#supprimer la colonne "index" et la colonne "hotel" ... nhotha heading ...
data_clean = data_clean.reset_index().drop('index', axis=1)
resort_h = resort_h.reset_index().drop(['index', 'hotel'], axis=1)
city_h = city_h.reset_index().drop(['index', 'hotel'], axis=1)
```

Analyse de quelques données

Analyser le taux d'annulation de réservation ¶

In [200]:
```
with pd.ExcelWriter('all_bookings_clean.xlsx') as writer:
    data_clean.to_excel(writer, sheet_name='data')
    resort_h.to_excel(writer, sheet_name='rh')
resort_h = pd.read_excel('all_bookings_clean.xlsx', index_col=0, sheet_name='rh')
```

In [201]:
```
cancel_rh = resort_h['is_canceled'].value_counts().reset_index()
pie_values_rh = [cancel_rh.iloc[0, 1]/cancel_rh['is_canceled'].sum(), cancel_rh.iloc[1, 1]/cancel_rh['is_canceled'].sum()]

# visualisation
colors = ['#a1c181', '#619b8a']
labels = ['Not canceled', 'canceled']

fig = plt.figure(figsize=(12, 4))
ax = fig.add_subplot(121)
ax.pie(pie_values_rh, labels=labels, autopct='%1.1f%%', colors=colors)
ax.set_title('Resort Hotel')

del pie_values_rh, labels, fig, ax
```

Conclusion

Le jeu de données "Hotel Data" nécessiterait une analyse binaire pour prédire les annulations de réservations, c’est-à-dire nécessiterait l’utilisation des méthodes de classification.

05

Choix de modèle pour Sales Car

5.1. Choix de modèle pour Sales Car

La Régression Linéaire :

La Régression Linéaire est une technique d'analyse statistique qui vise à établir une relation linéaire entre une variable dépendante (dans ce cas, les prix de vente des voitures d'occasion) et un ensemble de variables indépendantes (telles que la marque, le modèle, l'année de fabrication, le kilométrage, etc.). Dans le contexte du jeu de données "Sales Car", l'objectif est de développer un modèle prédictif qui puisse estimer les prix de vente en fonction de ces caractéristiques.

Importation des bibliothèques

Importation des biblio nécessaires

In [150]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Prétraitement des données

In [152]:
```
# Prétraitement des données
# Supprimer les lignes avec des valeurs manquantes
dataRg.dropna(inplace=True)
```

Sélectionner les caractéristiques (variables indépendantes) et la cible (variable dépendante)

In [153]:
```
# Sélectionner les caractéristiques (variables indépendantes) et la cible (variable dépendante)
X = dataRg[['Price_in_thousands', 'Engine_size', 'Horsepower', 'Curb_weight']]
y = dataRg['Sales_in_thousands']
```

Diviser les données en ensembles d'entraînement et de test

In [154]:
```
# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Créer le modèle de régression linéaire

In [155]:
```
# Créer le modèle de régression linéaire
regressor = LinearRegression()
```

Entraîner le modèle sur les données d'entraînement

In [156]:
```
# Entraîner le modèle sur les données d'entraînement
regressor.fit(X_train, y_train)
```

Out[156]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

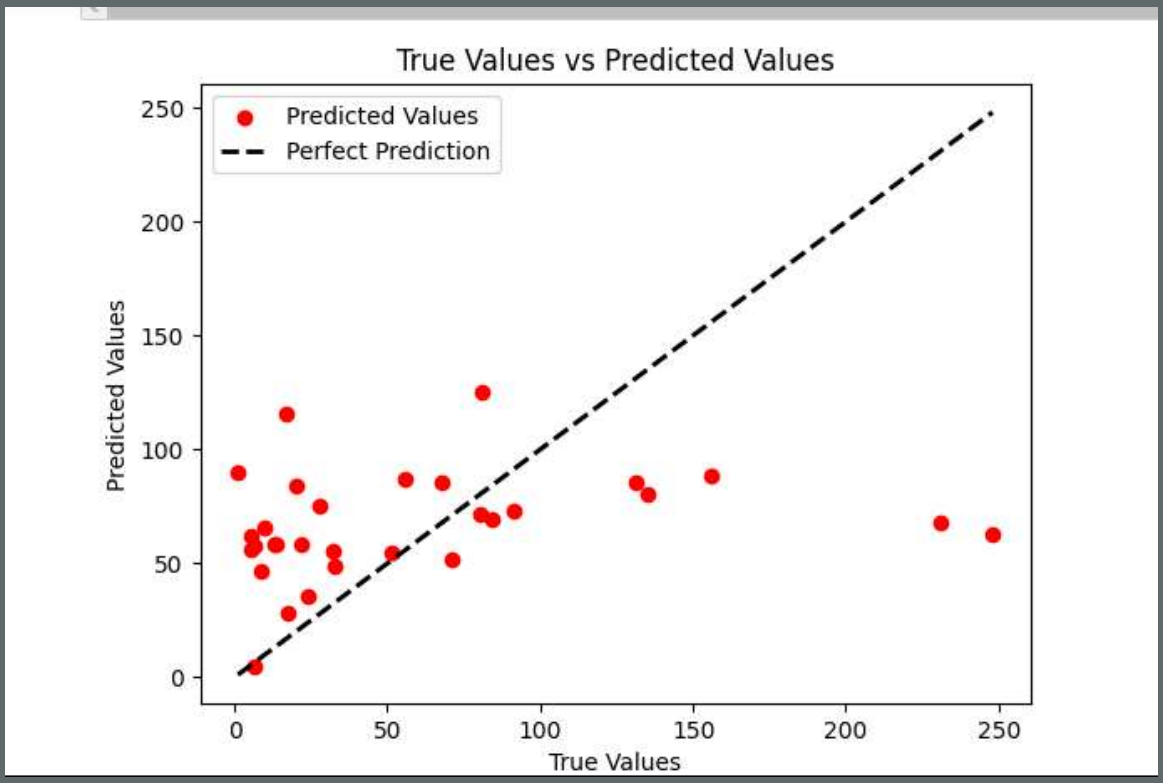
Faire des prédictions sur l'ensemble de test

In [157]:
```
# Faire des prédictions sur l'ensemble de test
y_pred = regressor.predict(X_test)
```

Visualisation

Visualisation des résultats

In [159]:
```
# Visualisation des résultats
plt.scatter(y_test, y_pred, color='red', label='Predicted Values')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='black', lw=2, linestyle='--', label='Perfect Prediction')
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('True Values vs Predicted Values')
plt.legend()
plt.show()
```



Conclusion

La régression linéaire a été choisie pour son adéquation à la nature continue de la variable cible dans le jeu de données "Sales Car", offrant ainsi une approche robuste pour modéliser et prédire les prix de vente des voitures d'occasion en fonction de diverses caractéristiques.

5.2. Choix de modèle pour Hotel Data

La Régression Logistique :

La régression logistique est une technique de modélisation statistique adaptée aux problèmes de classification, où la variable cible est binaire. L'objectif de la régression logistique est de modéliser la probabilité d'annulation d'une réservation en fonction de plusieurs caractéristiques, telles que le délai de réservation, le type de repas, le segment de marché, le type de chambre réservée, et le type de chambre assignée.

Importation des bibliothèques

Importation des biblio nécessaires

```
import pandas as pd
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, roc_auc_score
```

Partie 1 : Exploration des données

```
# Sélectionnez les variables pertinentes pour le Resort Hotel
features_resort = resort_h[['lead_time', 'meal', 'market_segment', 'reserved_room_type', 'assigned_room_type']]
```

```
# Gérez les données catégoriques avec one-hot encoding
features_resort = pd.get_dummies(features_resort, columns=['meal', 'market_segment', 'reserved_room_type', 'assigned_room_type'],
```

```
# La variable cible (y) est 'is_canceled', 1 si la réservation est annulée, 0 sinon
target_resort = resort_h['is_canceled']
```

```
# Créez un modèle de régression logistique pour le Resort Hotel
model_resort = LogisticRegression()
```

Partie 2 : Entrainement

```
# Ajustez le modèle aux données d'entraînement du Resort Hotel
model_resort.fit(X_train_resort, y_train_resort)
```

```
C:\Users\khaoula tlili\anaconda3\envs\myenv-gpu\lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarnin
fgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Partie 3 : Test

```
# Faites des prédictions pour le Resort Hotel
y_pred_resort = model_resort.predict(X_test_resort)
```

```
# Calculez les probabilités pour la classe positive (1) pour le Resort Hotel
y_pred_prob_resort = model_resort.predict_proba(X_test_resort)[:, 1]
```

```
# Évaluez les performances du modèle pour le Resort Hotel
accuracy_resort = accuracy_score(y_test_resort, y_pred_resort)
print("Précision du modèle pour le Resort Hotel : {:.2f}%".format(accuracy_resort * 100))
```

```
Précision du modèle pour le Resort Hotel : 74.52%
```

```
conf_matrix_resort = confusion_matrix(y_test_resort, y_pred_resort)
print("Matrice de confusion pour le Resort Hotel :")
print(conf_matrix_resort)
```

```
Matrice de confusion pour le Resort Hotel :
[[4825  343]
 [1458  441]]
```

```
class_report_resort = classification_report(y_test_resort, y_pred_resort)
print("Rapport de classification pour le Resort Hotel :")
print(class_report_resort)
```

```
Rapport de classification pour le Resort Hotel :
              precision    recall  f1-score   support

           0       0.77      0.93      0.84      5168
           1       0.56      0.23      0.33      1899

    accuracy                           0.75      7067
   macro avg       0.67      0.58      0.59      7067
weighted avg       0.71      0.75      0.70      7067
```

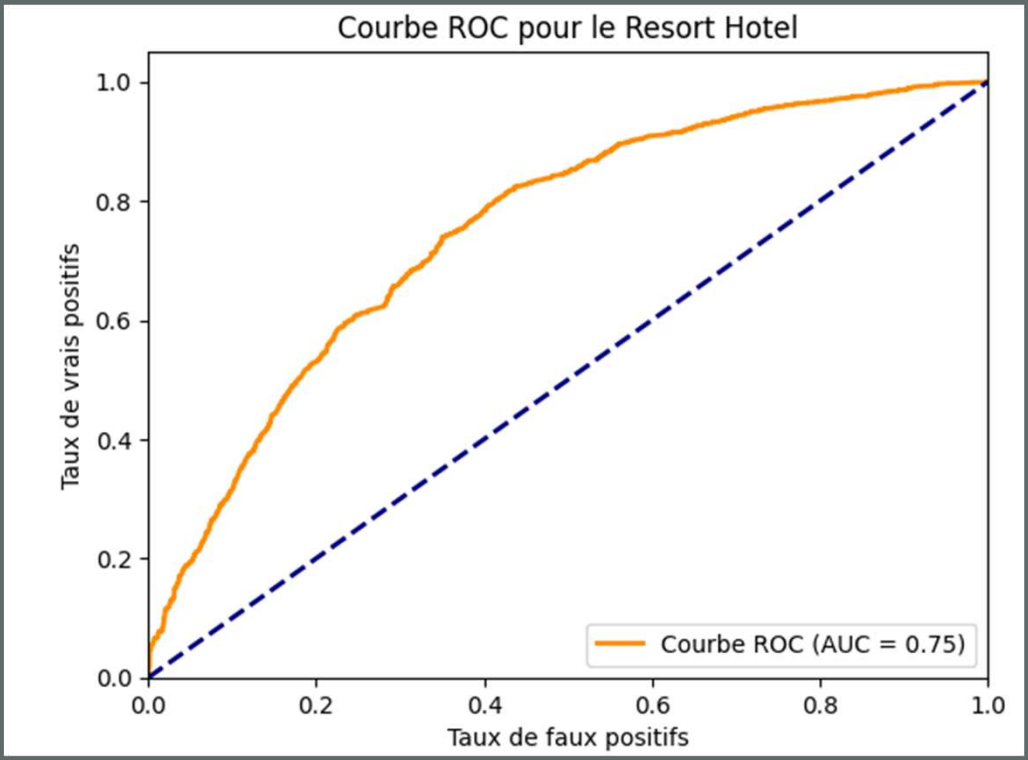
```
# Calcul de la courbe ROC pour le Resort Hotel
fpr_resort, tpr_resort, thresholds_resort = roc_curve(y_test_resort, y_pred_prob_resort)
```

```
# Calcul de l'aire sous la courbe ROC pour le Resort Hotel
roc_auc_resort = roc_auc_score(y_test_resort, y_pred_prob_resort)
```

Visualisation

Visualisation des résultats

```
# Tracage de la courbe ROC pour le Resort Hotel
plt.figure()
plt.plot(fpr_resort, tpr_resort, color='darkorange', lw=2, label='Courbe ROC (AUC = {:.2f})'.format(roc_auc_resort))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('Courbe ROC pour le Resort Hotel')
plt.legend(loc='lower right')
plt.show()
```



Conclusion

Le choix du jeu de données "Hotel Data" pour l'application de la régression logistique permet de construire un modèle prédictif visant à comprendre et prédire les annulations de réservations dans les hôtels.

Sources

https://github.com/aaqibqadeer/Hotel-booking-demand/blob/master/hotel_bookings.csv

https://github.com/mrdbourke/zero-to-mastery-ml/blob/master/data/car-sales.csv