Défi IA 2023 : *1001 Nights !*

BERNADA Ima - ROBERT Ambre

Université de Bordeaux

January 18, 2023

Sommaire

Introduction

Introduction

Contexte

Ce défi portait sur un problème de régression. La particularité étant que c'était à nous de collecter les données qui allaient constituer notre jeu de données d'entrainement. Ces dernières provenaient d'une agence de voyage fictive et étaient relatives à des réservations d'hôtels dans des villes Européennes.

Objectifs

Notre objectif était d'abord de collecter des données du site de l'agence via un scraper.
Ensuite nous devions prédire le prix d'une réservation d'hôtel selon le contexte de celle-ci.

Introduction

Contexte

Ce défi portait sur un problème de régression. La particularité étant que c'était à nous de collecter les données qui allaient constituer notre jeu de données d'entrainement. Ces dernières provenaient d'une agence de voyage fictive et étaient relatives à des réservations d'hôtels dans des villes Européennes.

Objectifs

Notre objectif était d'abord de collecter des données du site de l'agence via un scraper.
Ensuite nous devions prédire le prix d'une réservation d'hôtel selon le contexte de celle-ci.

Les données

Jeu de données test

Variables	Description
index	index de la ligne
order_request	identifiant de la recherche effectuée
city	nom de la ville
date	nombre de jours entre la date de la requête et le jour recherché
language	la langue de l'interface
mobile	support de la recherche (1 si téléphone, 0 si ordinateur)
avatar_id	identifiant de l'utilisateur fictif
hotel_id	identifiant de l'hôtel
stock	quantité de chambre encore à disposition dans l'hôtel
group	le groupe de l'hôtel
brand	la marque de l'hôtel
parking	la présence ou non de parking (payant ou non) dans l'hôtel (1 si oui, 0 sinon)
pool	la présence ou non d'une piscine (payante ou non) dans l'hôtel (1 si oui, 0 sinon)
children_policy	la présence ou non de restrictions concernant les enfants (2 si l'hôtel interdit les enfants de moins de 18 ou 21 ans, 1 si l'hôtel interdit les enfants de moins de 12 ans, 0 s'il autorise les enfants sans restrictions)

Table: Tableau des variables du jeu de données test.

Jeu de données test

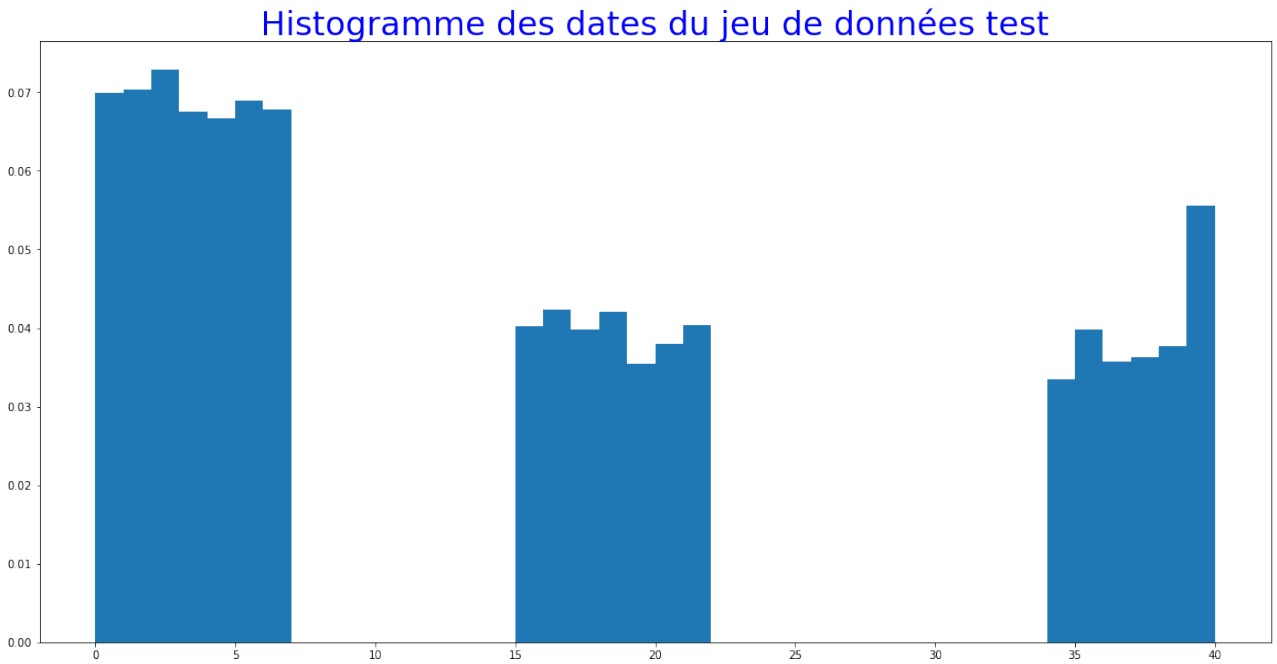


Figure: Histogramme des dates du jeu de données test.

Le jeu de données test ne contient pas toutes le dates possibles.

Jeu de données test

Common Values

Value	Count	Frequency (%)
amsterdam	1134	17.1%
paris	1125	16.9%
madrid	1094	16.5%
copenhagen	723	10.9%
rome	721	10.9%
vilnius	609	9.2%
vienna	568	8.5%
sofia	374	5.6%
valletta	296	4.5%

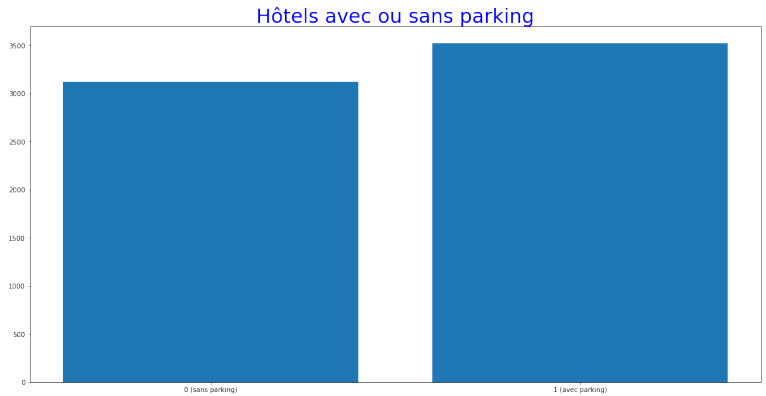
Figure: Répartition de la variable qualitative **city**.

Common Values

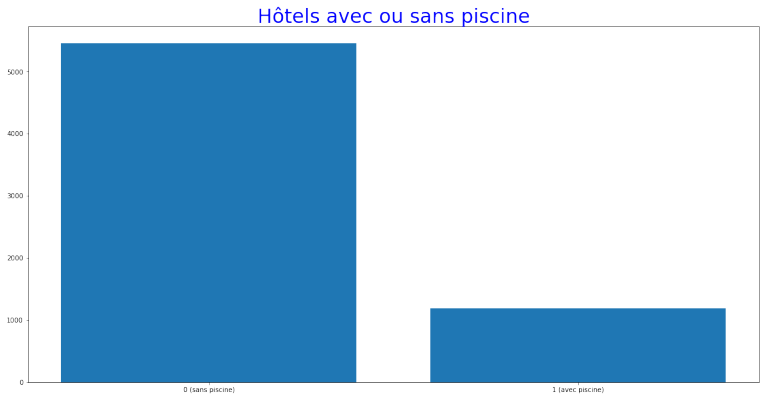
Value	Count	Frequency (%)
hungarian	967	14.6%
finnish	466	7.0%
austrian	461	6.9%
romanian	439	6.6%
slovakian	418	6.3%
swedish	404	6.1%
estonian	390	5.9%
bulgarian	356	5.4%
danish	346	5.2%
irish	209	3.1%
Other values (17)	2188	32.9%

Figure: Répartition de la variable qualitative **language**.

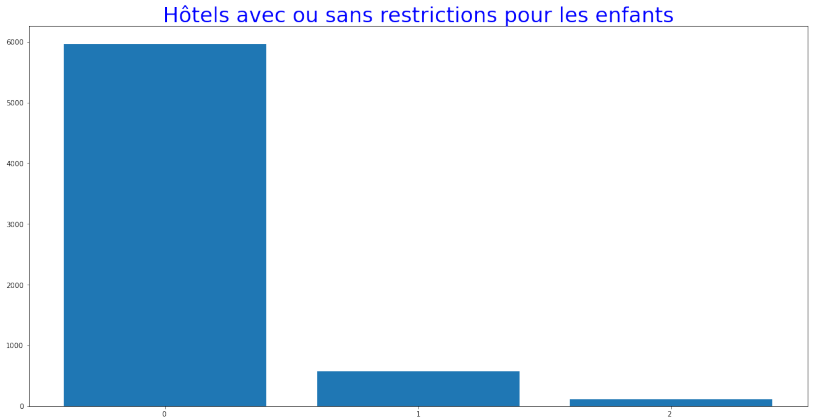
Jeu de données test



(a) Hôtels avec ou sans parking.



(b) Hôtels avec ou sans piscine.



(c) Hôtels avec ou sans restrictions pour les enfants.

Jeu de données test

Figure: Matrice des corrélations du jeu de données test.

Extraction des données

```python
requests1 = []
k = 0
#list1 = [i for i in range(44)]
list_date = [i for i in [1,2,3,4,5,6,15,16,17,18,19,20,21,34,35,36,37,38,39,40]]
city = ["amsterdam","paris","copenhagen","madrid","rome","sofia","valletta","vienna","vilnius"]
language = [ "austrian", "belgian", "bulgarian", "croatian", "cypriot", "czech", "danish", "dutch"
    , "estonian", "finnish", "french", "german", "greek", "hungarian", "irish", "italian", "
    latvian", "lithuanian", "luxembourgish", "maltese", "polish", "portuguese", "romanian", "
    slovakian", "slovene", "spanish", "swedish"]
mobile=[0,1]
for i in city :
    for j in language :
        for s in mobile :
            name = str(k+15000)
            r = requests.post(path(f"avatars/{user_id}/{name}"))
            params = {
            "avatar_name": name,
            "language": j,
            "city": i,
            "date": random.choice(list_date),
            "mobile": s,
            }
            k = k+1
            r = requests.get(path(f"pricing/{user_id}"), params=params)
            requests1.append(r)
```

Figure: Code pour l'extraction des données.

Jeu de données d'apprentissage

Variables	Description
index	index de la ligne
hotel_id	l'identifiant de l'hôtel
price	prix, en euros, d'une nuit dans un hôtel
stock	quantité de chambre encore à disposition dans l'hôtel
city	le nom de la ville où se trouve l'hôtel
date	nombre de jours entre la date de la requête et le jour recherché
language	la langue de l'interface
mobile	support de la recherche (1 si téléphone, 0 si ordinateur)
avatar_id	identifiant de l'utilisateur fictif
group	le groupe de l'hôtel
brand	la marque de l'hôtel
parking	la présence ou non de parking (payant ou non) dans l'hôtel (1 si oui, 0 sinon)
pool	la présence ou non d'une piscine (payante ou non) dans l'hôtel (1 si oui, 0 sinon)
children_policy	la présence ou non de restrictions concernant les enfants (2 si l'hôtel interdit les enfants de moins de 18 ou 21 ans, 1 si l'hôtel interdit les enfants de moins de 12 ans, 0 s'il autorise les enfants sans restrictions)

Table: Tableau des variables du jeu de données d'apprentissage.

Jeu de données d'apprentissage

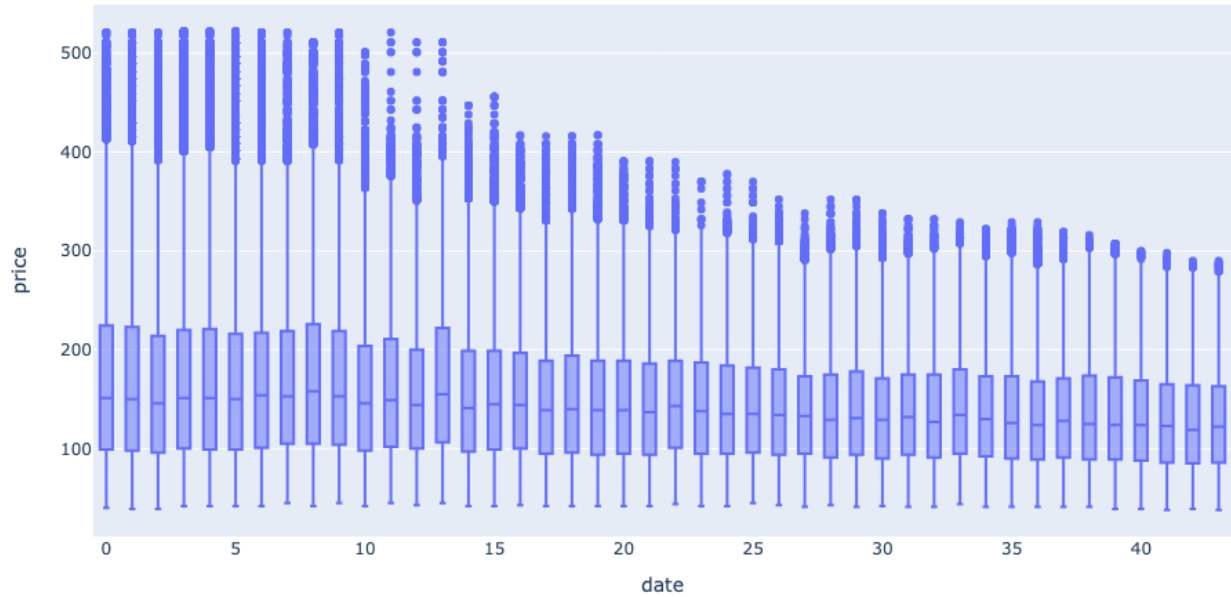


Figure: Boxplot des prix en fonction de la date.

Les prix sont plus faibles lorsque que la réservation est effectuée à l'avance.

Jeu de données d'apprentissage

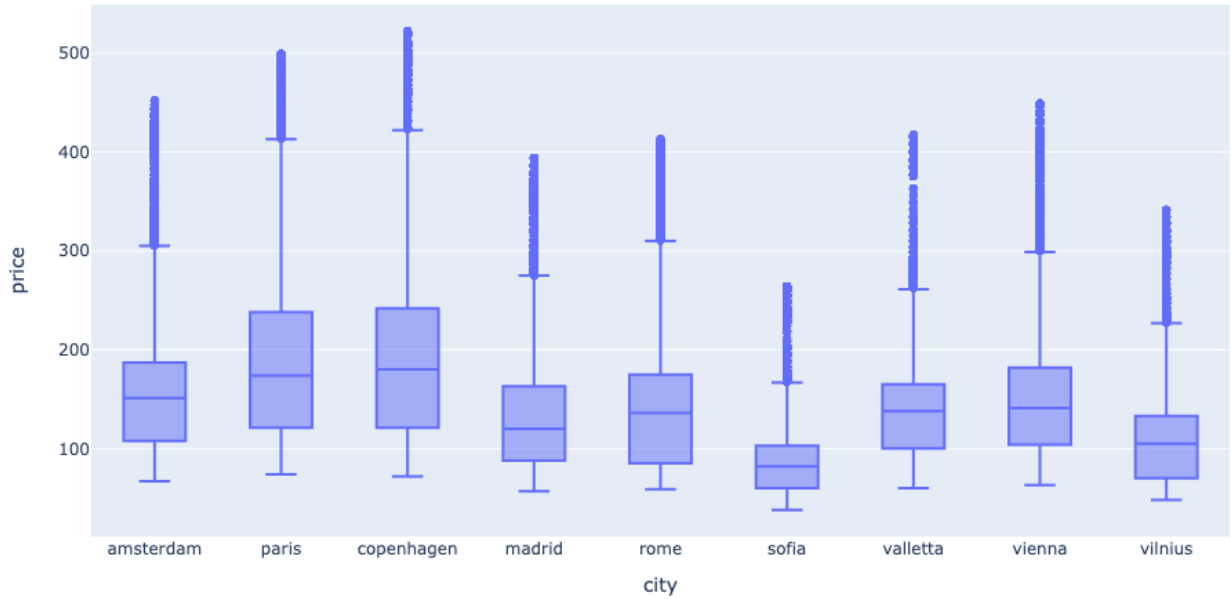


Figure: Boxplot des prix en fonction de la ville.

Les prix semblent varier en fonction de la ville de destination.

Jeu de données d'apprentissage

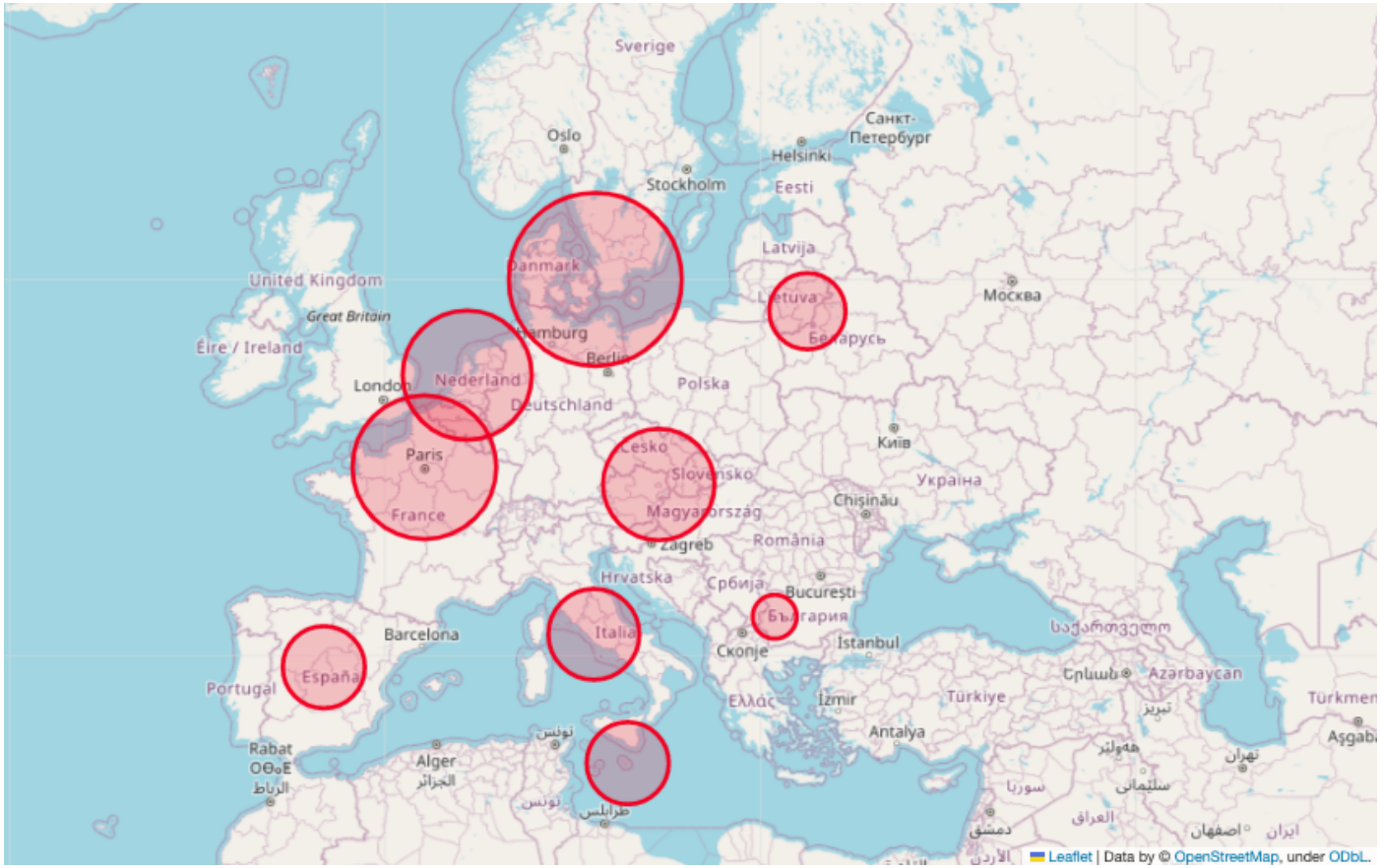
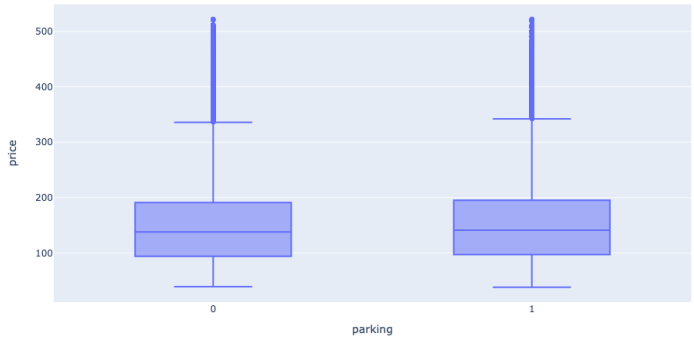
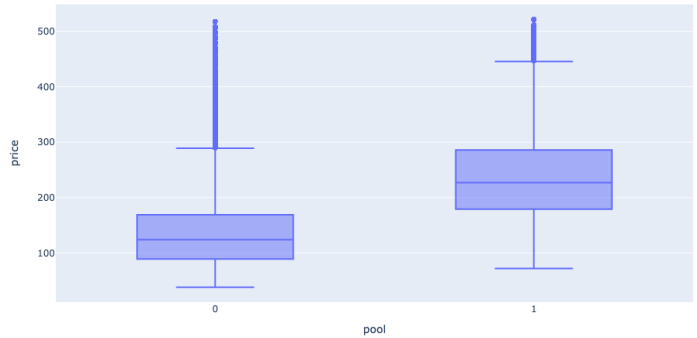


Figure: Carte de l'Europe illustrant la différence de prix en fonction de la ville.

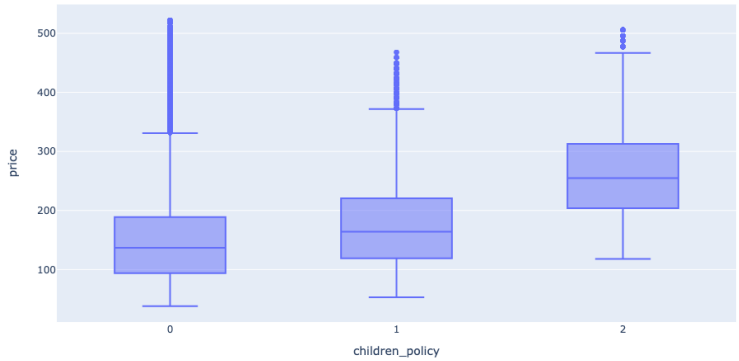
Boxplot du prix en fonction des caractéristiques de l'hôtel



(a) Parking.



(b) Piscine.



(c) Restrictions pour les enfants.

Jeu de données d'apprentissage

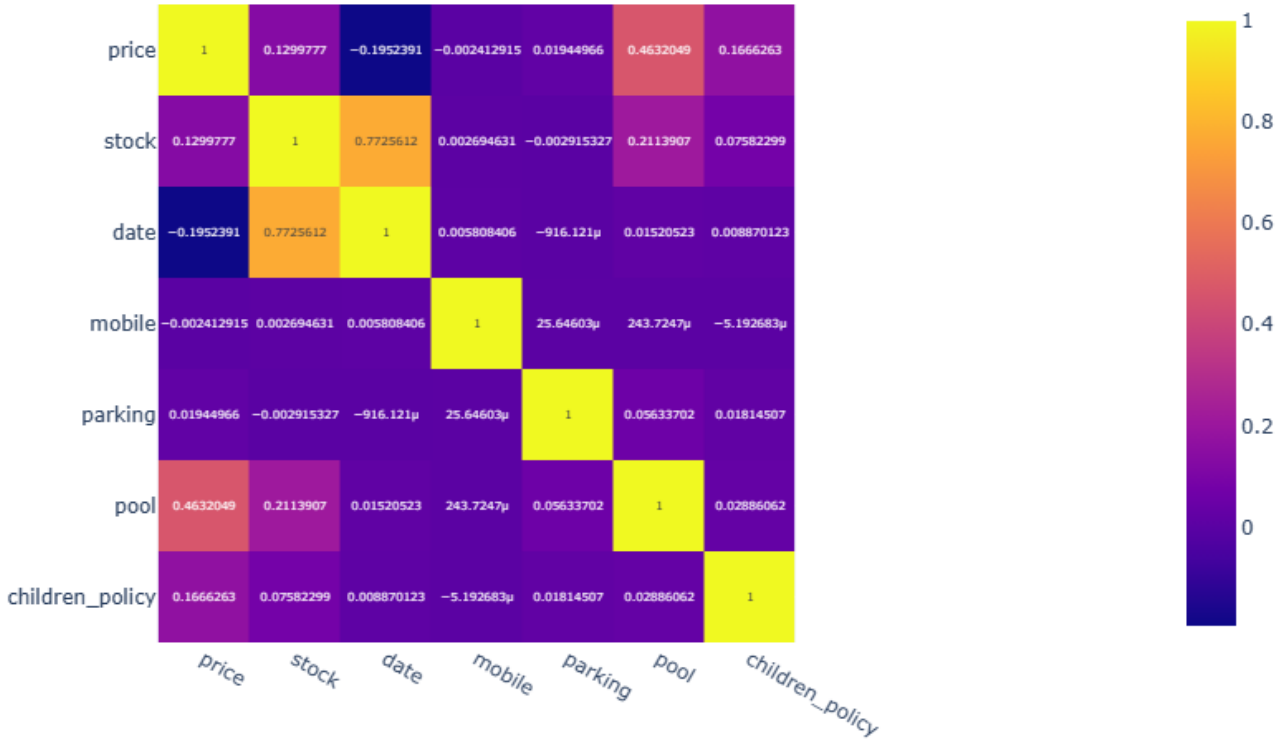


Figure: Matrice des corrélations du jeu de données d'entrainement.

Comparaison des deux jeux de données

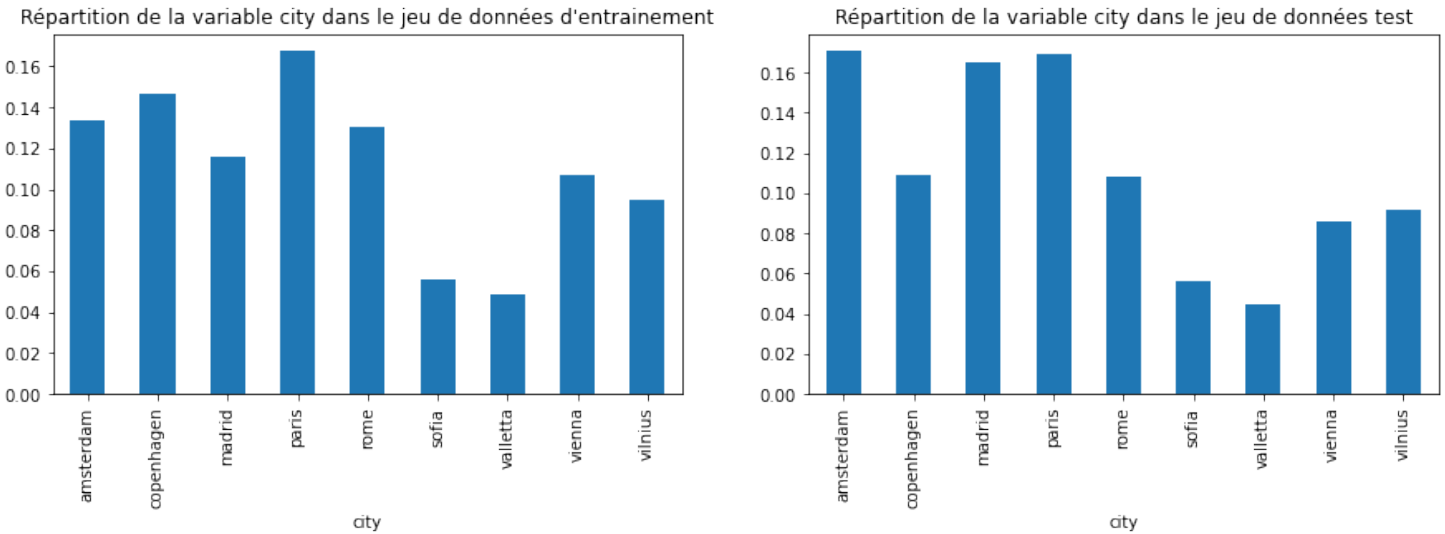


Figure: Diagrammes à barres des proportions de la variable city dans les ensemble d'entraînement et de test.

Comparaison des deux jeux de données

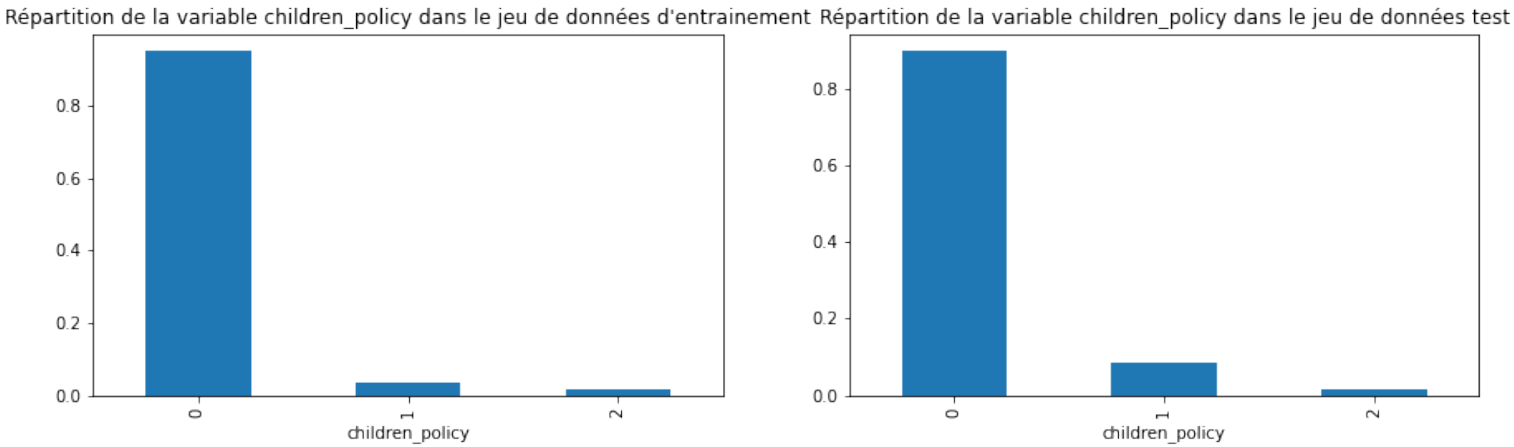


Figure: Diagrammes à barres des proportions de la variable children_policy dans les ensemble d'entraînement et de test.

Comparaison des deux jeux de données

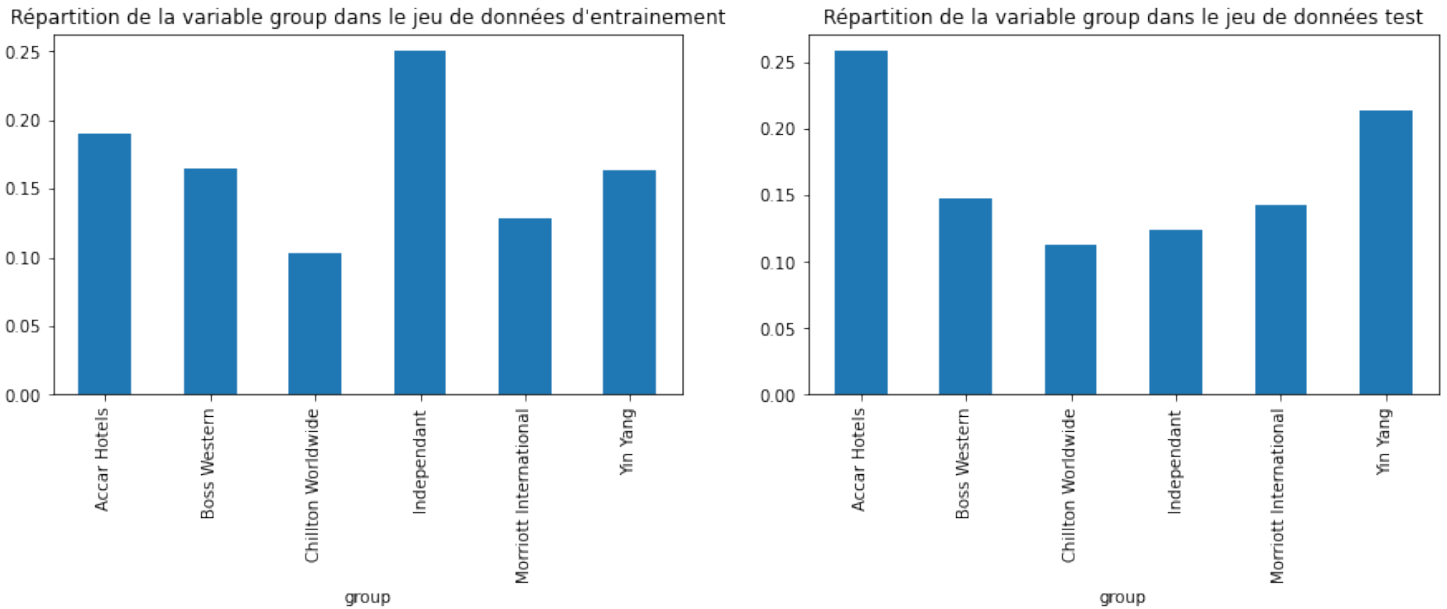


Figure: Diagrammes à barres des proportions de la variable groupe dans les ensemble d'entraînement et de test.

Préparation des données

```python
def prepare_data(dataset, testset, columnsTitles, train_size) :

    train = pd.get_dummies(dataset, drop_first = True)
    pred_df = pd.get_dummies(testset, drop_first = True)

    train_df, test_df = train_test_split(train, train_size = train_size)

    X_train = train_df.drop(['price', 'avatar_id','Unnamed: 0'], axis=1)
    X_test = test_df.drop(['price', 'avatar_id','Unnamed: 0'], axis=1)
    X_pred = pred_df.drop(['order_requests', 'avatar_id', 'index'], axis=1)
    columnsTitles = X_train.columns
    X_pred = X_pred.reindex(columns=columnsTitles)

    y_train = train_df['price']
    y_test = test_df['price']

    return(train_df, test_df, pred_df, X_train, y_train, X_test, y_test, X_pred)
```

Figure: Fonction permettant la préparation des données.

Méthodes

Méthodes utilisées

Liste des méthodes testées

- Prix moyen par modalité de variable catégorielle
- Régression linéaire
- Régression Lasso
- Régression Ridge
- Arbre de décision
- Forêt aléatoire
- Réseaux de neurones

Résultats

Évaluation de la performance

Root-mean-square error (RMSE)

$$\text{RMSE} = \sqrt{\mathbb{E}\left((\hat{y} - y)^2\right)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

où $\hat{y}_1, \hat{y}_2, ..., \hat{y}_n$ sont les valeurs prédites, $y_1, y_2, ..., y_n$ sont les valeurs réelles et n le nombre d'observations.

Résultats par méthodes

Méthodes	**Valeurs RMSE calculées**	**Scores obtenus sur Kaggle**
Prix moyen par paramètre de variable catégorielle	non calculée	93.68347
Régression Linéaire	29.22584	34.47581
Régression Lasso	29.36317	34.93777
Régression Ridge	29.16757	34.50374
Arbre de décision	2.88639	29.30552
Forêt aléatoire	2.67877	26.14481
Réseau de neurones	15.84681	19.82012

Table: Scores pour chaque méthode testée.

Premier réseau de neurones

```python
def model1():
    # Réseau feedforward
    model = tf.keras.Sequential() #lance un reseau ou on va pouvoir mettre des couches

    model.add(tf.keras.Input(shape=X_train.shape[1]))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(128*32, activation = 'relu'))
    model.add(tf.keras.layers.Dropout(0.2))

    model.add(tf.keras.layers.Dense(128*8, activation = 'relu'))
    model.add(tf.keras.layers.Dropout(0.2))

    model.add(tf.keras.layers.Dense(128*2, activation = 'relu'))
    model.add(tf.keras.layers.Dropout(0.2))

    model.add(tf.keras.layers.Dense(1))

    # Compilation du modèle
    model.compile(loss = tf.keras.losses.MeanSquaredError(), optimizer = 'adam', metrics = ['mae'])
    return model
```

Figure: Structure du premier réseau de neurones.

Second réseau de neurones

```python
def model2():
    # Réseau feedforward
    model = tf.keras.Sequential() #lance un reseau ou on va pouvoir mettre des couches

    model.add(tf.keras.Input(shape=X_train.shape[1]))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(128*8, activation = 'relu'))
    model.add(tf.keras.layers.Dropout(0.2))

    model.add(tf.keras.layers.Dense(128*4, activation = 'relu'))

    model.add(tf.keras.layers.Dense(128*2, activation = 'relu'))

    model.add(tf.keras.layers.Dense(128, activation = 'relu'))

    model.add(tf.keras.layers.Dense(64, activation = 'relu'))
    model.add(tf.keras.layers.Dropout(0.2))

    model.add(tf.keras.layers.Dense(1))

    # Compilation du modèle
    model.compile(loss = tf.keras.losses.MeanSquaredError(), optimizer = 'adam', metrics = ['mae'])
    return model
```

Figure: Structure du second réseau de neurones.

Performance du second réseau de neurones

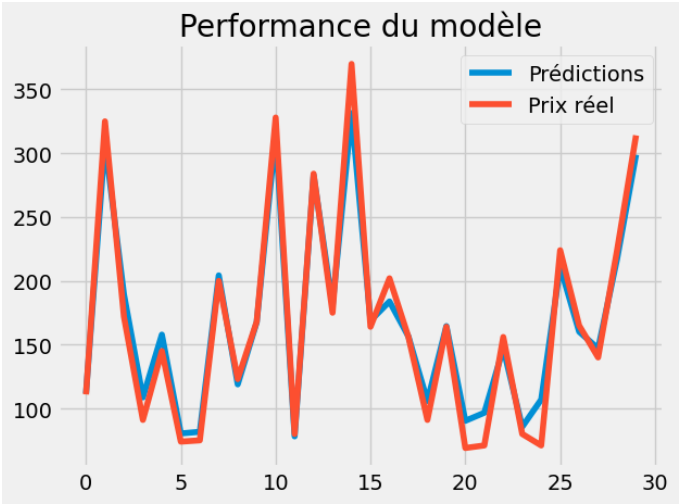


Figure: Comparaison des prix de réservation prédit et des prix effectifs.

Calcul de la RMSE

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \approx 15.8.$$

Performance du second réseau de neurones

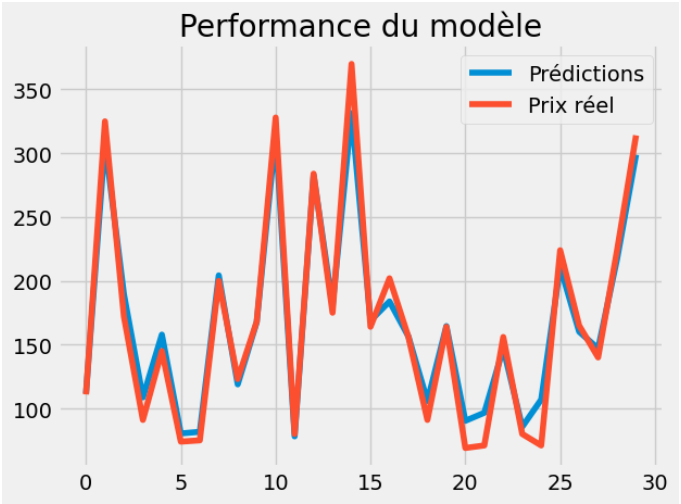


Figure: Comparaison des prix de réservation prédit et des prix effectifs.

Calcul de la RMSE

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \approx 15.8.$$

Conclusion

Conclusion

- Les réseaux de neurones nous ont permis d'obtenir les meilleurs résultats en termes de précision et de performance.
- Les méthodes de régression linéaires ne sont pas très efficaces car elles supposent une relation linéaire entre les variables réponse et explicatives.

Limites :

- Variable hotel_id,
- Pas d'historique pour les avatars dans l'ensemble d'entraînement.