جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

Département d'Informatique

**Master Sciences et Techniques en Systèmes d'Information décisionnels et Imagerie**

Module : Data Mining

Mini Projet

Intitulé

# Data Mining :
# Optimisation des prix

Préparé Par :
Ben Addi Mohamed

Encadré Par :
Pr. Sabiri Mohamed

Année Universitaire 2023 – 2024

# Table des matières

## 1 Définition des données :

Utilisation du data mining sur les données extraites du site web Airbnb pour développer une stratégie de prix optimale pour les hôtes Airbnb Dans cette démarche, nous chercherons à répondre à des questions clés telles que :

- **Comment le prix évolue-t-il par rapport au mois ?**
- **Existe-t-il une relation entre le quartier et le prix du logement ?**

En parallèle, nous développons un modèle prédictif visant à estimer les prix optimaux en considérant divers éléments tels que l'emplacement, le type de logement, et d'autres paramètres pertinents.

Enfin, le modèle sera intégré à une application web facile à utiliser, créée avec Streamlit. L'application permettra aux utilisateurs de rechercher et d'obtenir des prix recommandés pour leurs logements.

## 2 Collection des données :

Le site web de Kaggle propose un ensemble de données librement accessible pour des tâches de Data science et Machine learning.

Les données que j'ai utilisées se trouvent à cette adresse : https://www.kaggle.com/datasets/airbnb/seattle?select=listings.csv , Le jeu de données comprend trois fichiers : calendar.csv, listings.csv et reviews.csv :

- Calendrier, incluant l'identifiant de l'annonce ainsi que le prix et la disponibilité pour chaque jour.
- Listings, comprenant des descriptions complètes et la note moyenne des commentaires.
- Reviews, incluant un identifiant unique pour chaque évaluateur et des commentaires détaillés.

## 3 Préparation des données :

J'ai supprimé toutes **les colonnes inutiles** pour notre problématique :

```
listings.info()
```

```
87  cancellation_policy                3818 non-null   object
88  require_guest_profile_picture      3818 non-null   object
89  require_guest_phone_verification   3818 non-null   object
90  calculated_host_listings_count     3818 non-null   int64
91  reviews_per_month                  3191 non-null   float64
dtypes: float64(17), int64(13), object(62)
memory usage: 2.7+ MB
```

```
listings.info()
```

```
53  cancellation_policy                3818 non-null   object
54  require_guest_profile_picture      3818 non-null   object
55  require_guest_phone_verification   3818 non-null   object
56  calculated_host_listings_count     3818 non-null   int64
57  reviews_per_month                  3191 non-null   float64
dtypes: float64(12), int64(12), object(34)
memory usage: 1.7+ MB
```

**Conversion du prix** depuis une chaîne de caractères (String) au format 10$ en nombre décimal (Float) 10 :

```
calendar.head(2)
```

| | listing_id | date | available | price |
|---|---|---|---|---|
| 0 | 241032 | 2016-01-04 | t | $85.00 |
| 1 | 241032 | 2016-01-05 | t | $85.00 |

```
calendar['price']=calendar['price'].str[1:-3]
```

```
calendar['price']=calendar['price'].str.replace(',','').astype(float)
```

```
calendar.head(2)
```

| | listing_id | date | available | price |
|---|---|---|---|---|
| 0 | 241032 | 2016-01-04 | t | 85.0 |
| 1 | 241032 | 2016-01-05 | t | 85.0 |

**La Suppression des anomalies** :

```
listings['guests_included'].unique()
```

```
array([ 2,  1, 10,  6,  4,  8,  3,  7,  0,  5,  9, 11, 15, 13, 12],
      dtype=int64)
```

```
listings[listings['guests_included']==15]
```

| | id | name | host_id | host_name | host_since | host_location | host_about |
|---|---|---|---|---|---|---|---|
| 693 | 9282348 | A bedroom close to UW | 48203627 | Lihua | 2015-11-04 | Seattle, Washington, United States | NaN |

1 rows × 58 columns

Il n'y a qu'une seule ligne, je vais la supprimer.

```
listings=listings[listings['guests_included']!=15]
```

## 4 Visualisation et Analyse de Données :

Dans cette partie, je vais formuler des questions auxquelles je répondrai en utilisant des visualisations et des analyses de données.

### Comment le prix évolue par rapport au mois ?

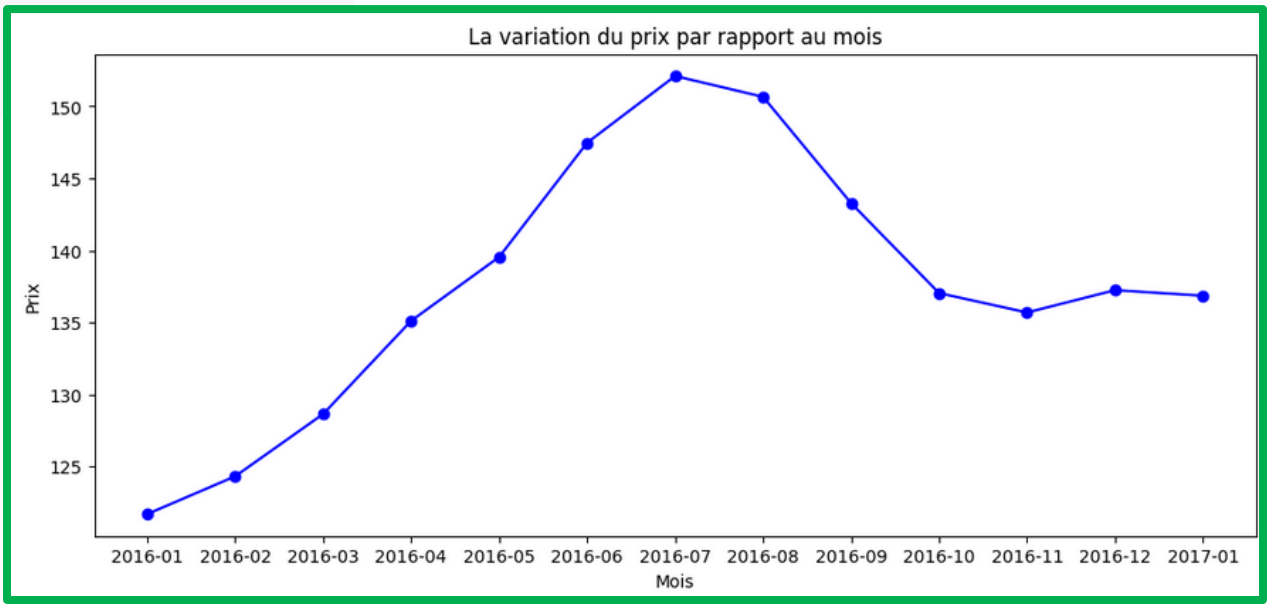
```
import matplotlib.pyplot as plt

x = grouping_month_price['month']
y = grouping_month_price['price']

# je vais convertir 'Period' en String
mois = [str(mois) for mois in x]

plt.figure(figsize=(12, 5))
plt.plot(mois, y, marker='o', linestyle='-', color='b')
plt.title('La variation du prix par rapport au mois')
plt.xlabel('Mois')
plt.ylabel('Prix')
#plt.legend()

plt.show()
```




✓ Comme nous observons ici, la plupart des annonces en juillet ont des prix plus élevés que ceux en janvier.

✓ Les prix passent du point le plus bas en janvier au point le plus élevé en juillet, puis redescendent jusqu'en décembre.

### Existe-t-il une relation entre le prix et le quartier ?

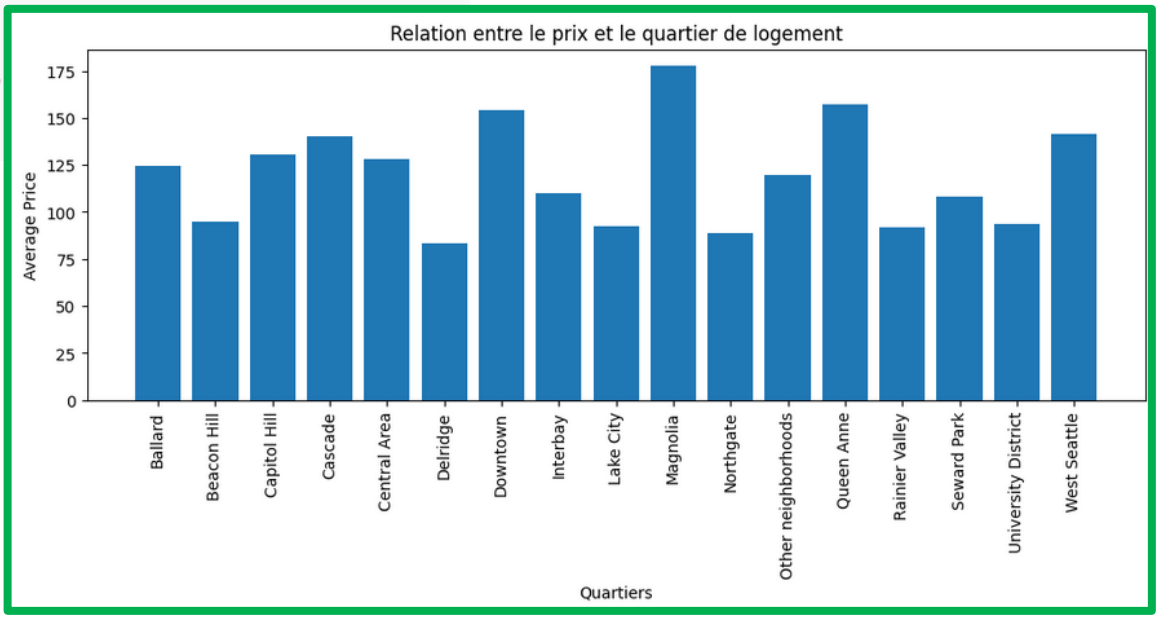
```
listings['neighbourhood_group_cleansed'].unique()
```

```
array(['Queen Anne', 'Ballard', 'Other neighborhoods', 'Cascade',
       'Central Area', 'University District', 'Downtown', 'Magnolia',
       'West Seattle', 'Interbay', 'Beacon Hill', 'Rainier Valley',
       'Delridge', 'Seward Park', 'Northgate', 'Capitol Hill',
       'Lake City'], dtype=object)
```

```
plt.figure(figsize=(12, 4))

plt.bar(neighbour_price['neighbourhood_group_cleansed'], neighbour_price['price'])

plt.xlabel('Quartiers')
plt.ylabel('Prix moyen')
plt.title('Relation entre le prix et le
plt.xticks(rotation='vertical')
plt.show()
```




✓ Comme nous pouvons le voir ici, il est préférable d'augmenter le prix si nous listons dans Magnolia, et de le baisser si nous listons à Delridge.

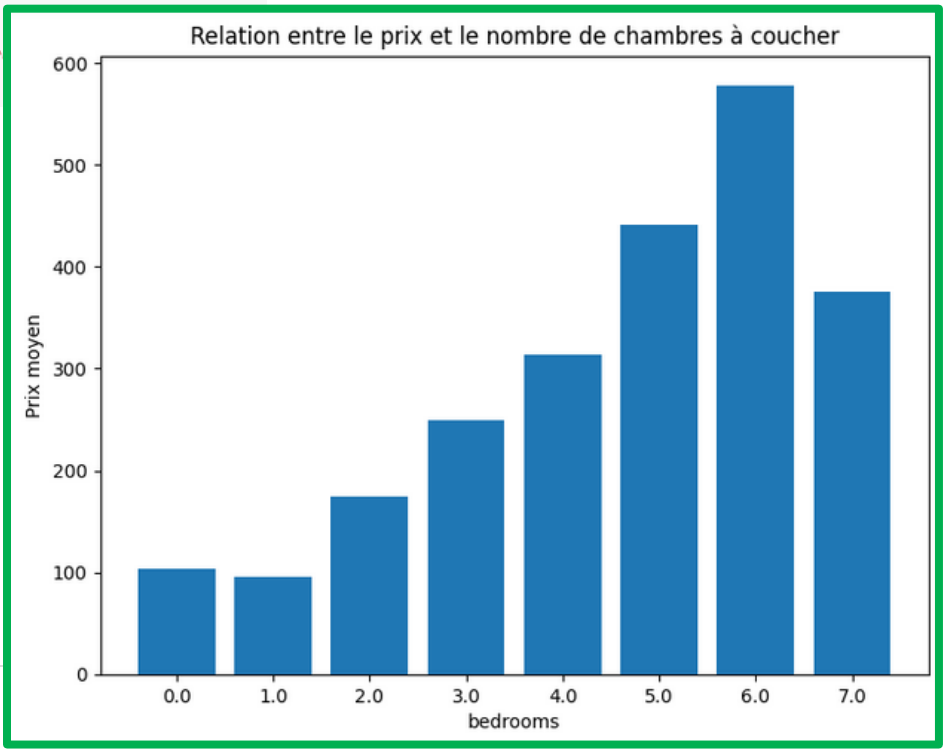
- **Existe-t-il une relation entre le prix et le nombre de chambres à coucher et de salles de bain ?**

```
plt.figure(figsize=(8, 6))

plt.bar([str(x) for x in bedrooms_price['bedrooms']], bedrooms_price['price'])

plt.xlabel('bedrooms')
plt.ylabel('Prix moyen')
plt.title('Relation entre le prix et le nombre de chambr
plt.show()
```

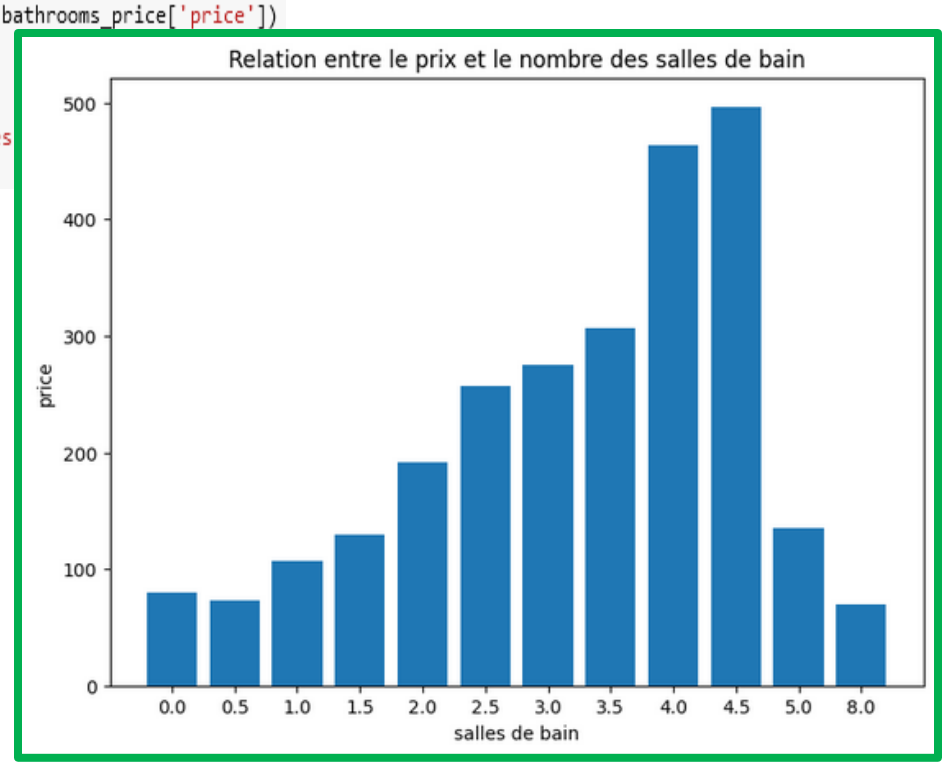
Comme nous pouvons le voir ici, Le prix augmente avec le nombre de chambres à coucher.



```
plt.figure(figsize=(8, 6))

plt.bar([str(x) for x in bathrooms_price['bathrooms']], bathrooms_price['price'])

plt.xlabel('salles de bain')
plt.ylabel('price')
plt.title('Relation entre le prix et le nombre des salles
plt.show()
```




La même remarque ici, Le prix augmente avec le nombre des salles de bain.

- **La communication de l'hôte, le temps de réponse et le taux de réponse influent-ils sur le prix ?**

```
communication_price.groupby(by=['host_response_time'],axis=0).mean()
```

| host_response_time | host_response_rate | review_scores_communication | price |
|---|---|---|---|
| a few days or more | 34.789474 | 9.206897 | 129.157895 |
| within a day | 86.247906 | 9.773305 | 132.819095 |
| within a few hours | 95.898760 | 9.760766 | 133.824380 |
| within an hour | 98.705674 | 9.850099 | 116.358747 |

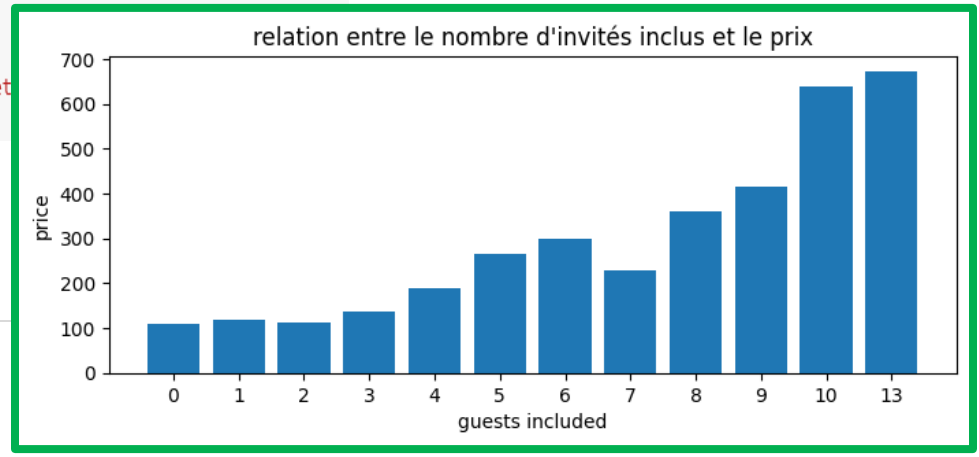
Il n'est pas évident qu'il existe une relation claire

- **Le nombre d'invités inclus influence-t-il le prix ?**

```
plt.figure(figsize=(8, 6))

plt.bar([str(x) for x in guests_included['guests_included']], guests_included['price'])

plt.xlabel('guests included')
plt.ylabel('price')
plt.title('relation entre le nombre d\'invités inclus et
plt.show()
```

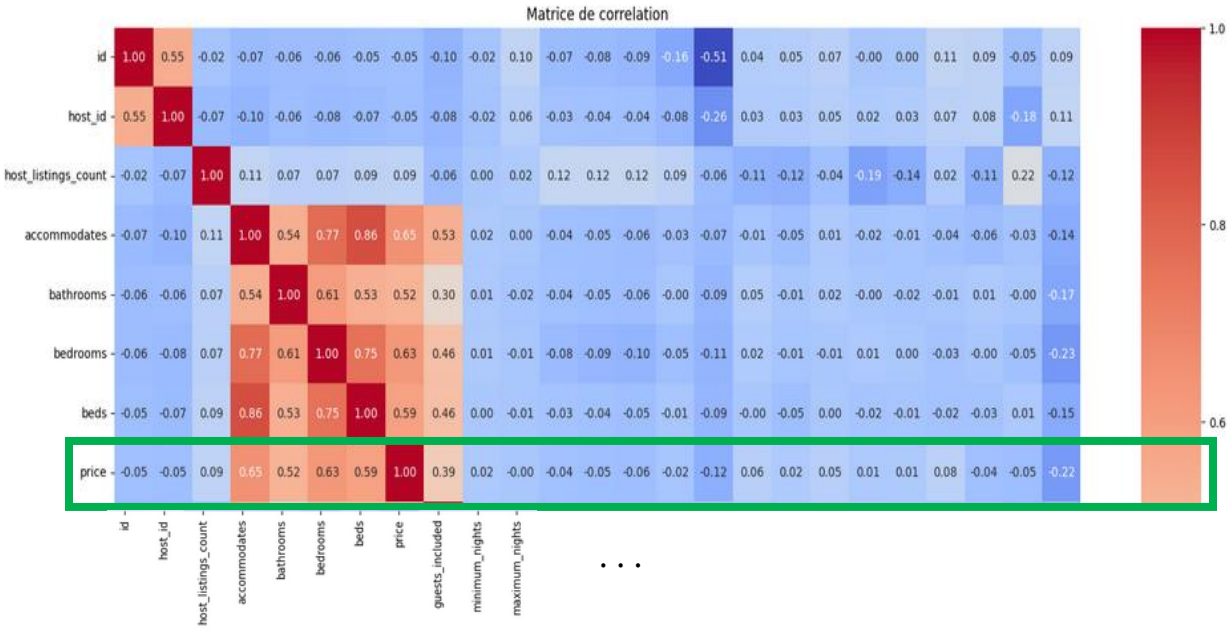



Comme nous pouvons le voir ici, si le nombre d'invités inclus est élevé, alors le prix augmente.

### Corrélation entre les variables :

Dans cette section, je vais calculer les corrélations entre les variables afin d'identifier les caractéristiques qui sont corrélées au prix.

```
# Matrice de correlation
correlation_matrix = listings.corr()
import seaborn as sns
```




✓ Comme nous pouvons le voir dans la carte thématique (Heatmap),le prix est fortement corrélé avec le nombre des invités (accom modates), le nombre de salles de bain, de chambres et de lits.

# 5 Machine Learning :

## 5.1 Préparation de données :

J'ai fusionné les attributs importants du fichier listings.csv avec la date et le prix du fichier calendar.csv en utilisant l'identifiant id_listings.

```
data = listings[['id','accommodates','bathrooms','beds','bedrooms','guests_included','neighbourhood_group_cleansed','price']
```

```
calendar['date']=pd.to_datetime(calendar['date'])
```

```
calendar2 = calendar[calendar['available']!='f']
```

```
merged_data = pd.merge(calendar2[['listing_id','date','price']], data, left_on='listing_id', right_on='id', how='inner',suff
```

J'ai séparé les données en caractéristiques et en la colonne cible.

```
X=merged_data[['accommodates','bathrooms','bedrooms','guests_included','neighbourhood_group_cleansed','month']]
```

```
y=merged_data['price']
```

Ensuite, j'ai converti les colonnes catégoriques (month, neighbourhoud_group_cleansed) en valeurs numériques à l'aide du codage onehotencoding.

```
from sklearn.preprocessing import OneHotEncoder
import numpy as np
ohe = OneHotEncoder()
transformed = ohe.fit_transform(X[['neighbourhood_group_cleansed']]).toarray()
column_labels = np.array(ohe.categories_).ravel()
ohe_df = pd.DataFrame(transformed, columns=column_labels)
X = pd.concat([X, ohe_df], axis=1)
X.drop(columns = 'neighbourhood_group_cleansed',inplace=True)
```

```
transformed = ohe.fit_transform(X[['month']]).toarray()
column_labels = np.array(ohe.categories_).ravel()
ohe_df = pd.DataFrame(transformed, columns=column_labels)
X = pd.concat([X, ohe_df], axis=1)
X.drop(columns = 'month',inplace=True)
```

## 5.2 Données d'entrainement et données de test :

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

## 5.3 Entrainement, test et évaluation des modèles :

### 5.3.3 Lasso (Least Absolute Shrinkage and Selection Operator) :

```
from sklearn.linear_model import Lasso
ls = Lasso(alpha = 0.1)
ls.fit(X_train,y_train)
```

```
Lasso(alpha=0.1)
```

```
y_pred_ls = ls.predict(X_test)
```

```
from sklearn.metrics import mean_absolute_percentage_error

mape = mean_absolute_percentage_error(y_test, y_pred_ls)
mape=mape * 100
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
```

```
Mean Absolute Percentage Error (MAPE): 35.15%
```

Les résultats du modèle montrent une erreur moyenne de prédiction de **31%**

Et une précision de prédiction de **58%.**

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred_ls)
print(f"r2 score: {r2:.2f}")
```

```
r2 score: 0.58
```

### 5.3.3 Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=40)
```

```
model.fit(X_train,y_train)
```

```
RandomForestRegressor(n_estimators=40)
```

```
y_pred_rf = model.predict(X_test)
```

```
mape_rf = mean_absolute_percentage_error(y_test, y_pred_rf)
mape_rf= mape_rf * 100
print(f"Mean Absolute Percentage Error (MAPE): {mape_rf:.2f}%")
```

```
Mean Absolute Percentage Error (MAPE): 21.15%
```

Les résultats du modèle montrent une erreur moyenne de prédiction de **21%**

Et une précision de prédiction de **84%.**

```
r2_rf = r2_score(y_test, y_pred_rf)
print(f"r2 score: {r2_rf:.2f}")
```

```
r2 score: 0.84
```

✓ Comme nous pouvons le voir ici, le modèle de forêt aléatoire donne de meilleurs résultats que le modèle Lasso.

## 5.4 Test du modèle avant le déploiement :

- **Si l'hôte propose 1 nombre d'invités, 1 salle de bains, 1 chambre, 1 invité inclus, dans le quartier de Delridge et au mois d'avril.**

```
model.predict(new_data1)
```

```
array([80.])
```

Le prix est bas !

D'après nos analyses, cette observation semble valide car la propriété a moins de lits, de salles de bains et se retrouve dans un quartier que nous avons identifié comme moins cher.

- **Si l'hôte propose 5 nombre d'invités, 5 salles de bains, 5 chambres, 5 invité inclus, dans le quartier de Central Area et au mois de juillet.**

```
model.predict(new_data2)
```

```
array([325.25])
```

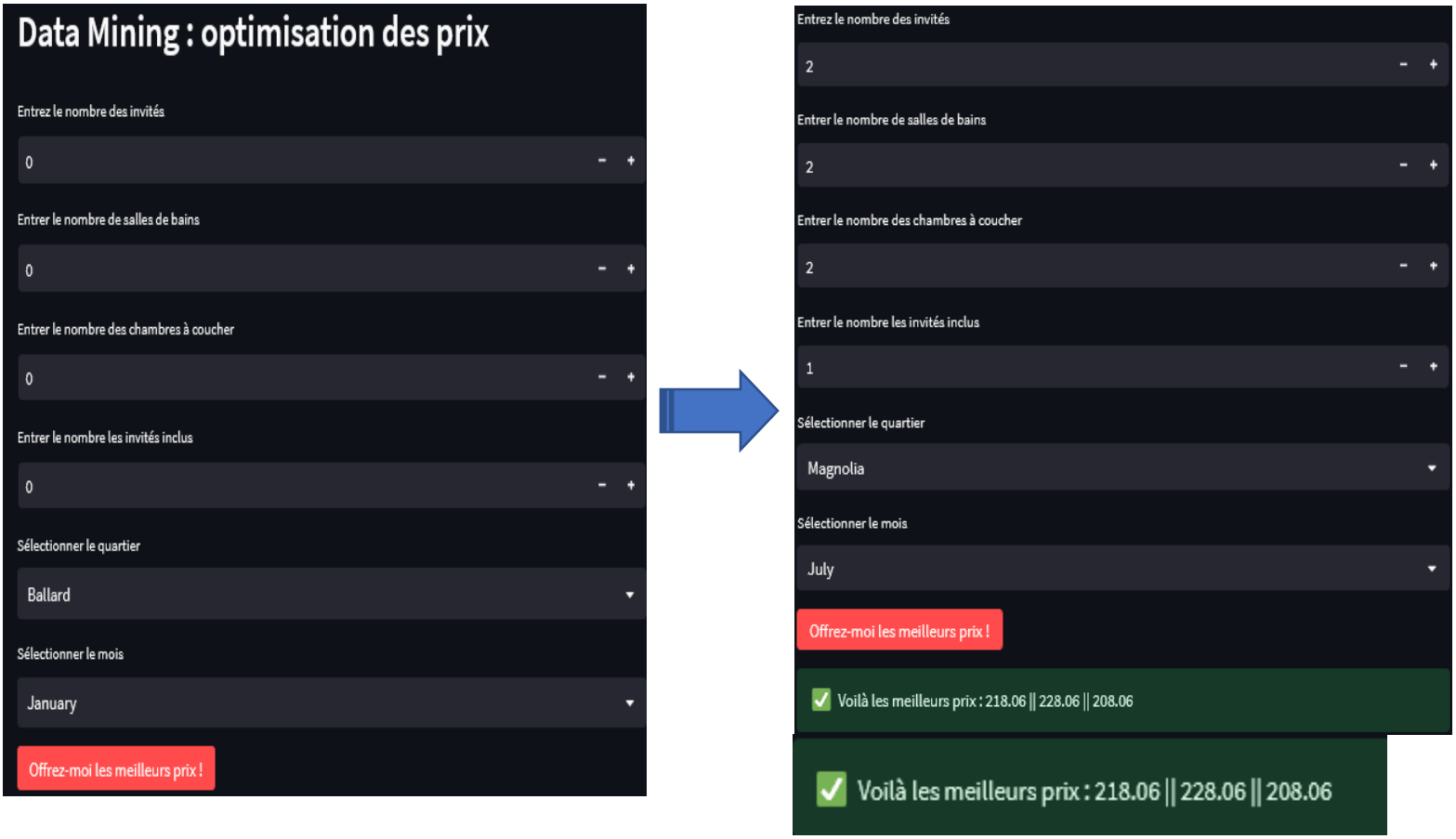
Le prix est élevé !

D'après nos analyses, cette observation semble valide car la propriété a plus de lits, de salles de bains et en juillet, mois que nous avons identifié comme étant plus chère.

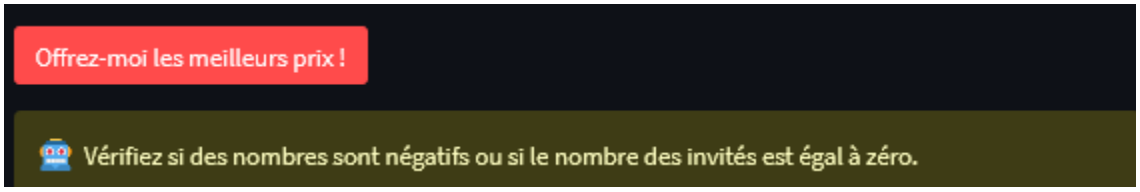
## 6 Le déploiement du modèle pour le monde réel :

```
import pickle
with open('optimisation_modele.pkl', 'wb') as file:
    pickle.dump(model, file)
```

J'ai utilisé la bibliothèque **Pickle** pour sauvegarder le modèle sous forme de fichier binaire, et en utilisant **Streamlit**, j'ai créé une interface web pour aider les hôtes à sélectionner le prix optimal.




- **En cas de valeur négative ou si le nombre d'invités (accommodates) est égal à 0 :**



## 7 Améliorations possibles :

Comment pouvons-nous améliorer ce projet d'optimisation des prix sur Airbnb grâce à l'apprentissage automatique ?

- Nous pourrions explorer des ensembles de données supplémentaires pour enrichir les prédictions et **Augmenter la précision du modèle.**

- Faire participer les utilisateurs pour fournir des commentaires sur le projet, ces commentaires pourraient être utilisés pour **Réduire le pourcentage d'erreurs.**

- **Transformer l'interface en une extension** pour l'utiliser directement sur le site web d'Airbnb.