IA - AirBnB

Objectif: mettre en forme exploitable pour une IA les data d'AirBnB

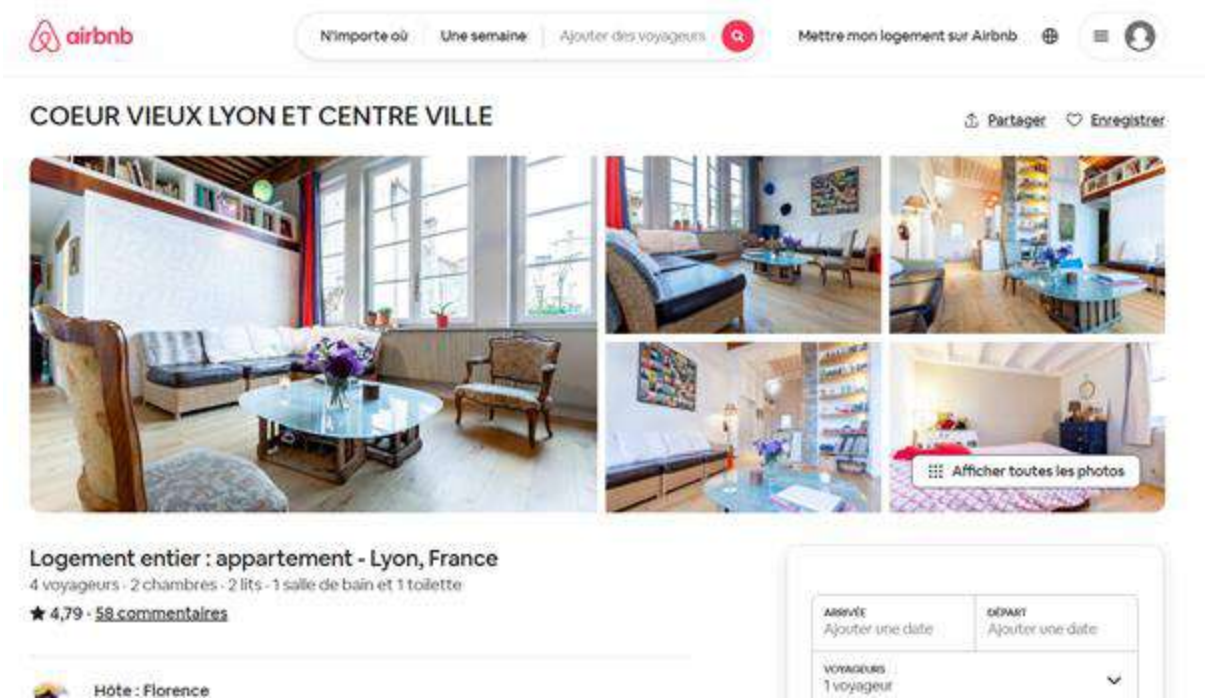
IA pour prédire les prix

Sommaire

Contexte du projet

On doit proposer le prix d’un nouveau logement sur airbnb.

La question c'est quoi le bon prix que l'on va proposer pour votre logement.



Collecte des données

http://insideairbnb.com/get-the-data/

Inside Airbnb
Adding data to the debate


```
lyon = pd.read_csv('/content/lyon.csv')
```

```
lyon.columns == paris.columns
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True])
```

```
print("le fichier Lyon contient", lyon.shape[0], "lignes et", lyon.shape[1], "colonnes.")
print("le fichier Paris contient", paris.shape[0], "lignes et", paris.shape[1], "colonnes.")
```

```
le fichier Lyon contient 9898 lignes et 75 colonnes.
le fichier Paris contient 67942 lignes et 75 colonnes.
```

Prétraitement des Données

sélection des colonnes

```
lyon1 = lyon.drop(columns=[ 'host_acceptance_rate', 'host_response_rate',
       'host_response_time', 'listing_url', 'scrape_id', 'last_scraped',
      'source', 'neighborhood_overview', 'picture_url', 'host_url', 'host_name',
      'host_since', 'host_location', 'host_about',
       'host_is_superhost', 'host_thumbnail_url', 'host_picture_url',
       'host_neighbourhood',
       'host_verifications',
         'neighbourhood',
       'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'latitude',
       'longitude',  'room_type',  'bathrooms',
     'bedrooms',  'amenities',
        'minimum_minimum_nights',
       'maximum_minimum_nights', 'minimum_maximum_nights',
       'maximum_maximum_nights', 'minimum_nights_avg_ntm',
       'maximum_nights_avg_ntm', 'calendar_updated',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped',
       'number_of_reviews_ltm', 'number_of_reviews_l30d', 'first_review',
       'last_review',
        'license', 'instant_bookable',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms'])
lyon1.head()
```

Nous avons sélectionné les colonnes les plus pertinentes pour évaluer les biens Airbnb, excluant les informations redondantes ou non essentielles afin de concentrer l'analyse sur les facteurs influant directement sur les décisions de location et de prix.

Prétraitement des Données

suppression des doublons

```
[ ]  print(lyon1.duplicated().sum())
     if lyon1.duplicated().sum() > 0:
       lyon1 = lyon1.drop_duplicates()
     lyon1.head()
```

```
[ ]  print(paris1.duplicated().sum())
     if paris1.duplicated().sum() > 0:
       paris1 = paris1.drop_duplicates()
     paris1.head()
```

Nous avons supprimer les lignes en double d'un DataFrame nommé `lyon1`, en affichant d'abord le nombre de doublons avant de les supprimer si nécessaire.

Prétraitement des Données

gestion des valeurs manquantes

```
cols= ['review_scores_value', 'review_scores_checkin',
       'review_scores_location', 'review_scores_accuracy',
       'review_scores_communication', 'review_scores_cleanliness',
       'reviews_per_month', 'review_scores_rating', 'beds',
       'host_total_listings_count', 'host_listings_count']
lyon1[cols] = lyon1[cols].apply(lambda x: x.fillna(x.mean()))
paris1[cols] = paris1[cols].apply(lambda x: x.fillna(x.mean()))
```

- Sélection des Caractéristiques Importantes
- Repérage et Remplissage des Cases Vides
- Application à Deux Ensembles de Données

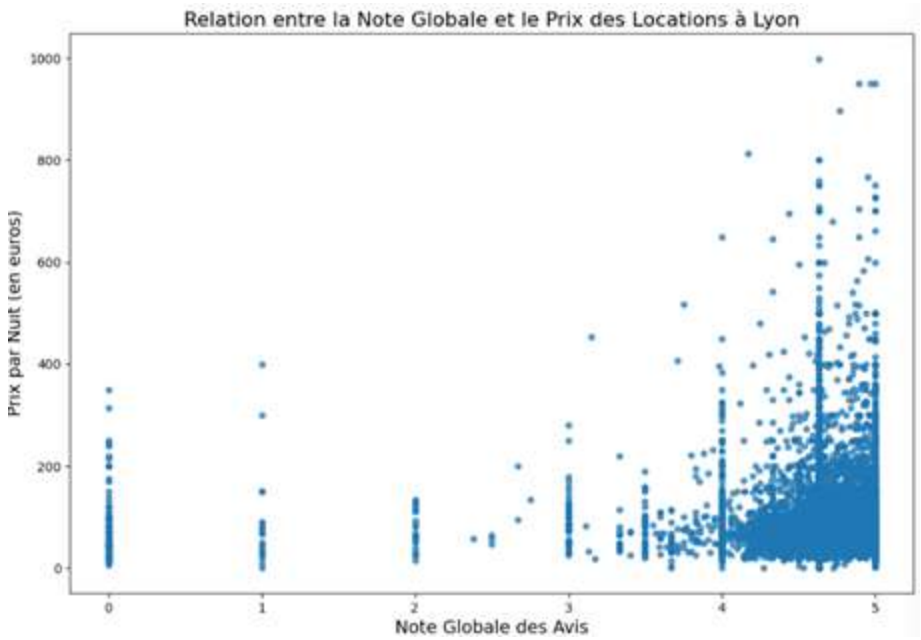
Prétraitement des Données

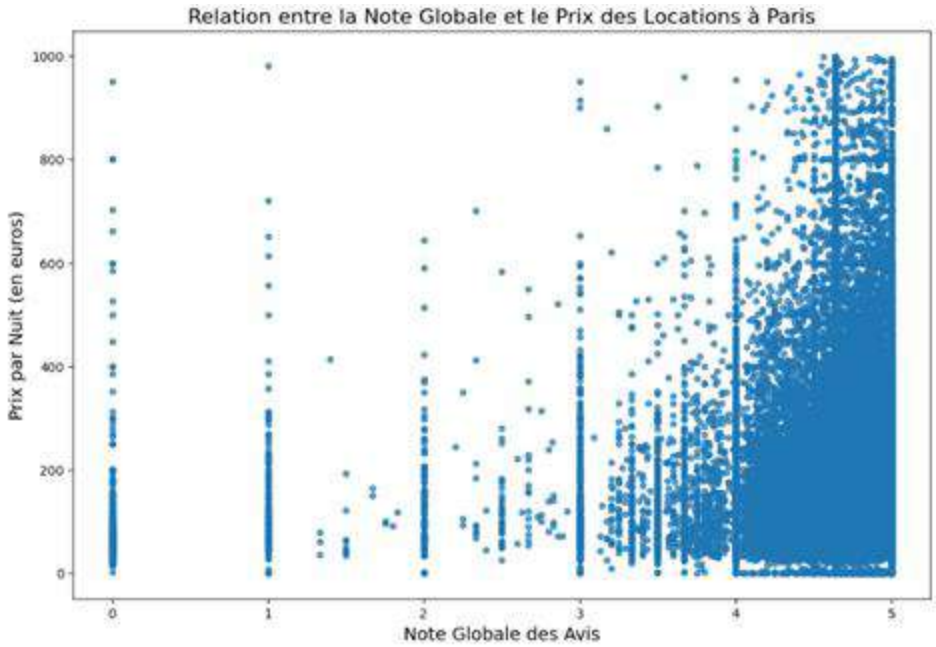
conversion des type de colonnes

```
lyon1['price'] = lyon1['price'].astype(str).str.extract('(\\d+)').fillna(0).astype(float)
lyon1['bathrooms_text'] = lyon1['bathrooms_text'].astype(str).str.extract('(\\d+)').fillna(0).astype(int)
paris1['price'] = paris1['price'].astype(str).str.extract('(\\d+)').fillna(0).astype(float)
paris1['bathrooms_text'] = paris1['bathrooms_text'].astype(str).str.extract('(\\d+)').fillna(0).astype(int)
```

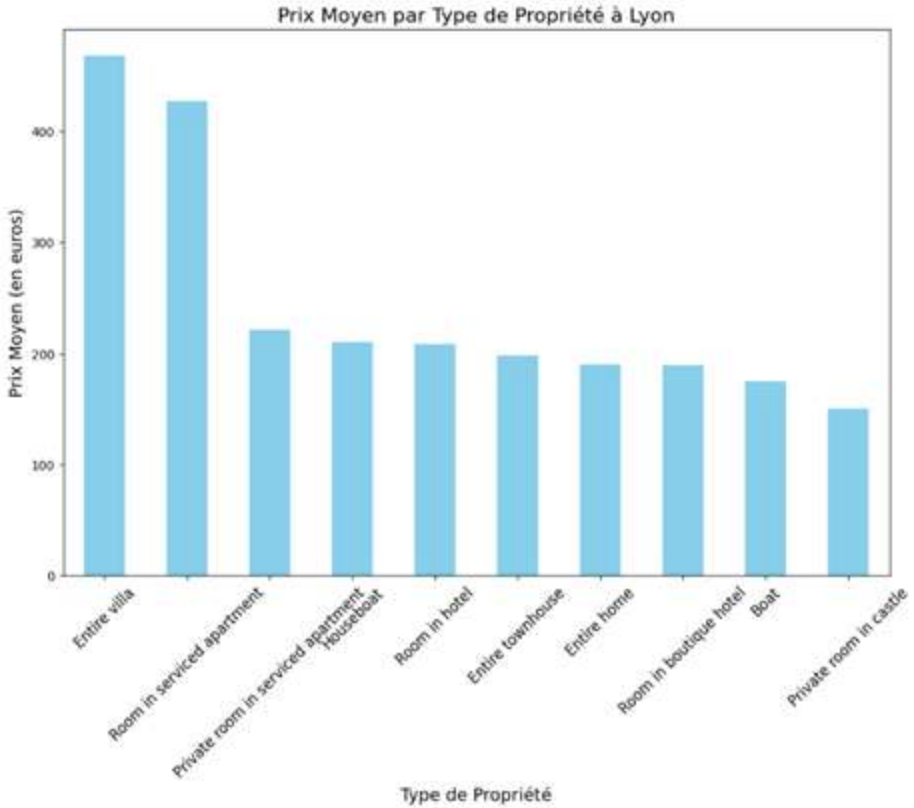
- Conversion de Texte en Chiffres
- Extraction de Nombres
- Uniformisation des Données

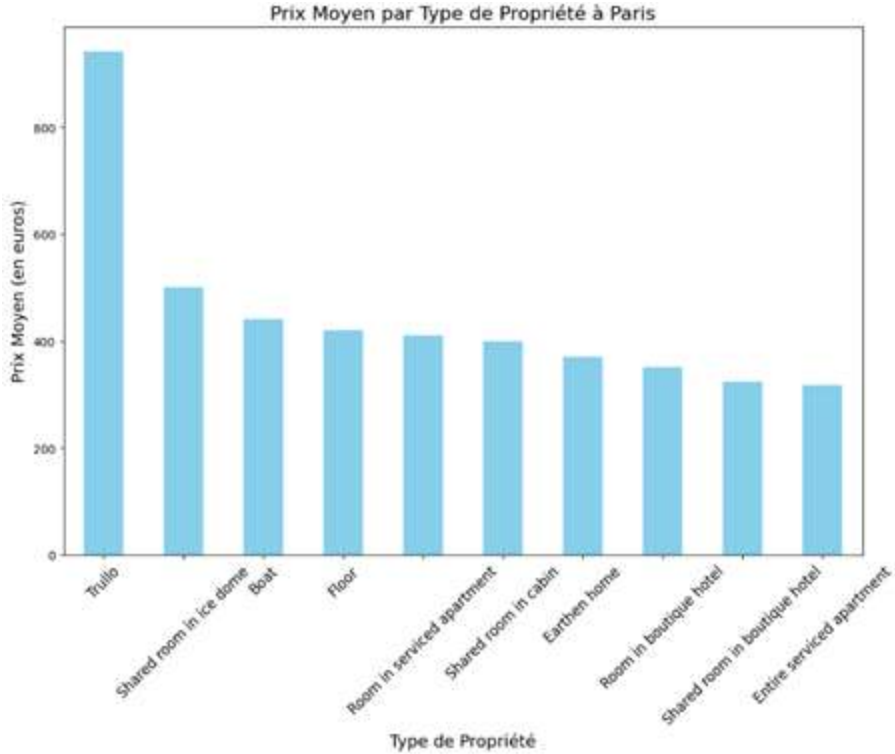
Analyse descriptive des données



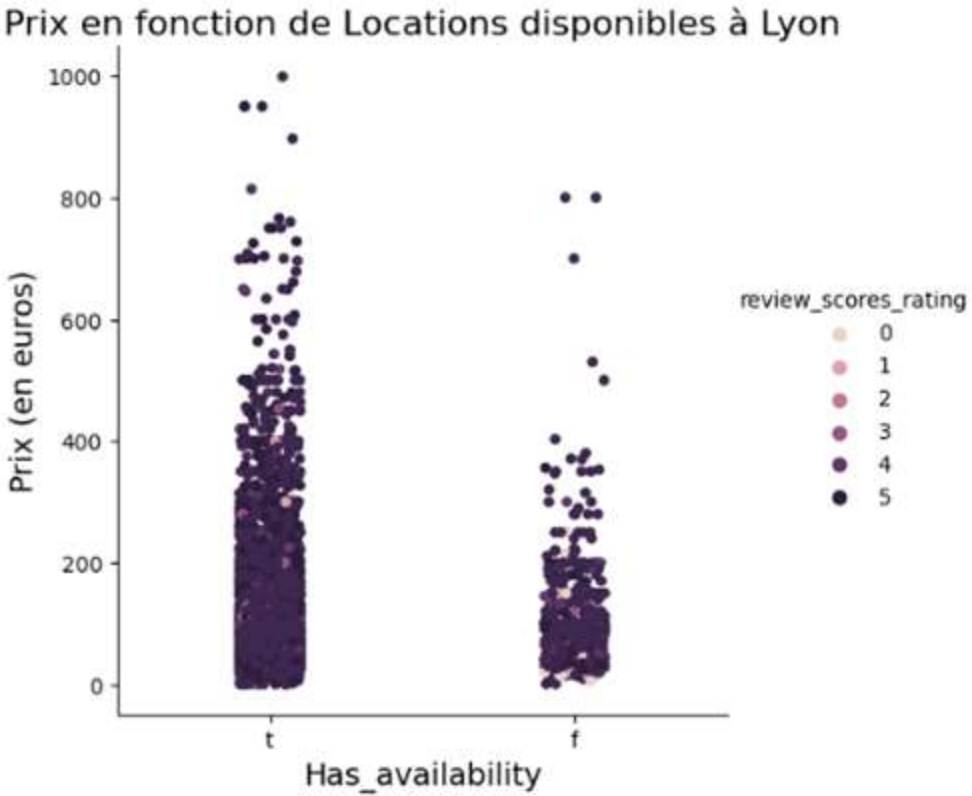


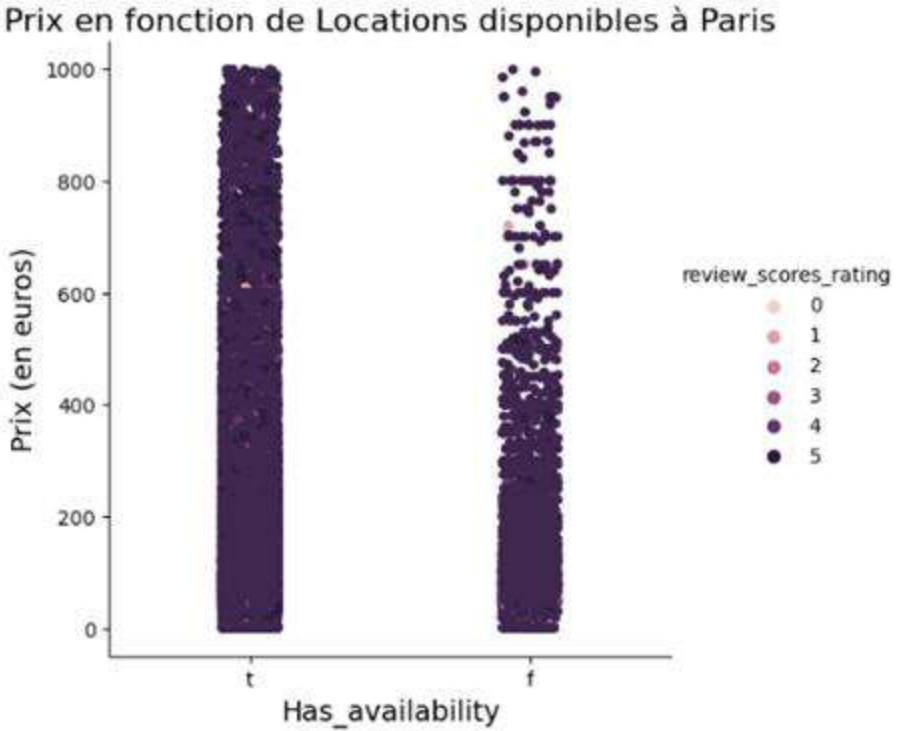
Analyse descriptive des données



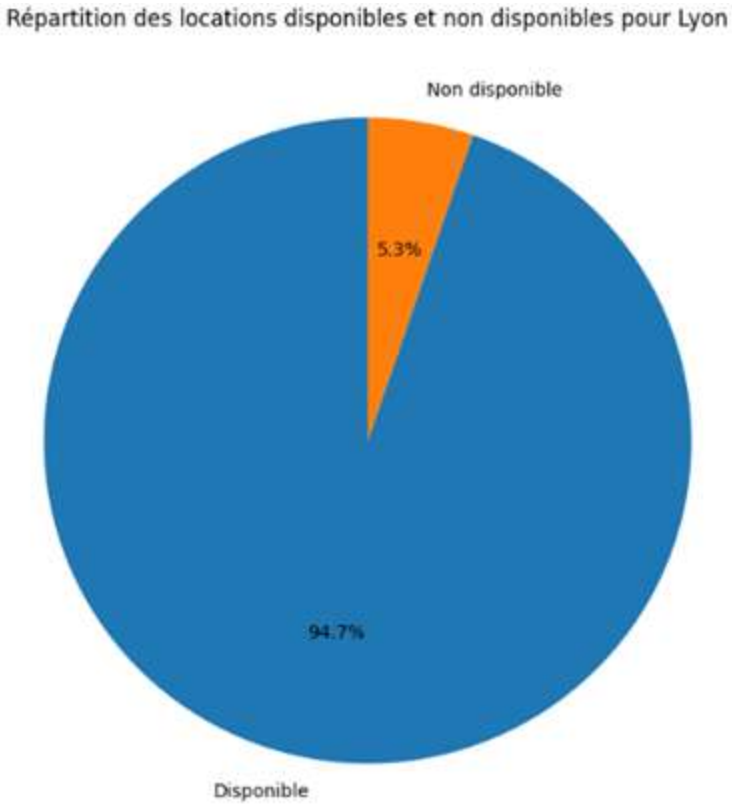


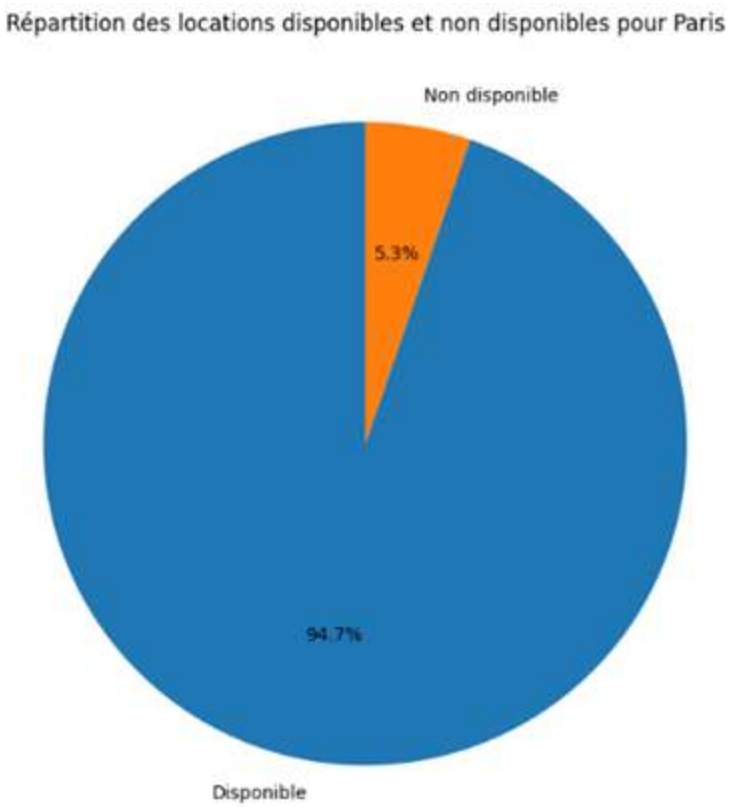
Analyse descriptive des données





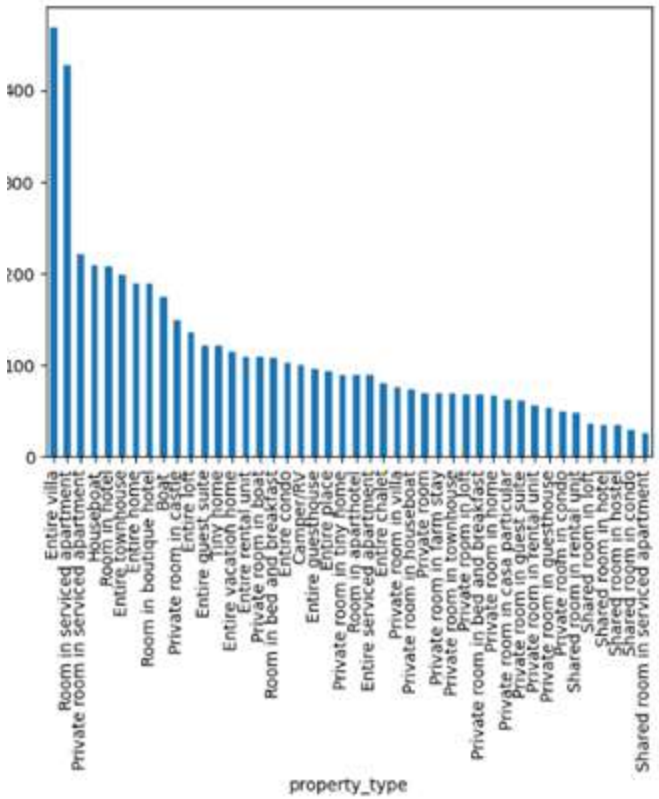
Analyse descriptive des données



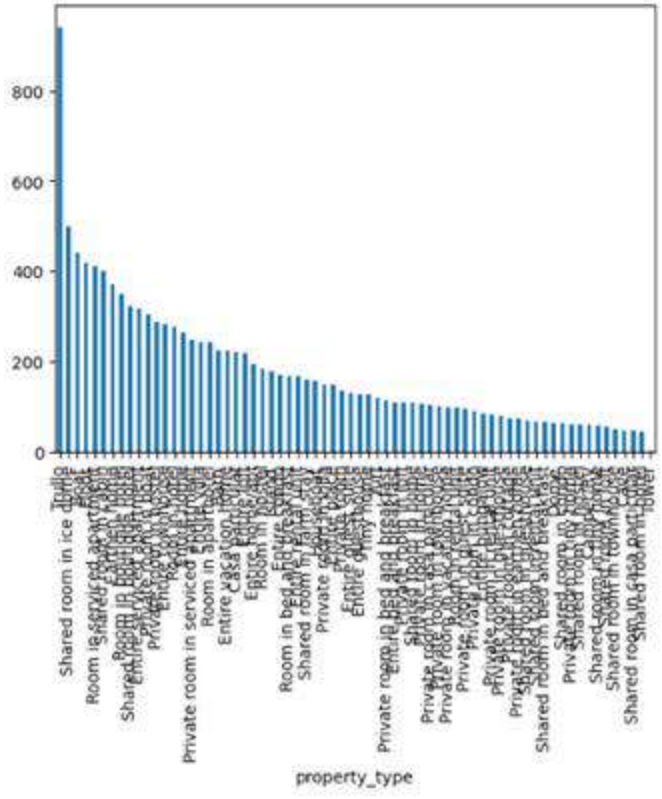


Analyse descriptive des données

Lyon

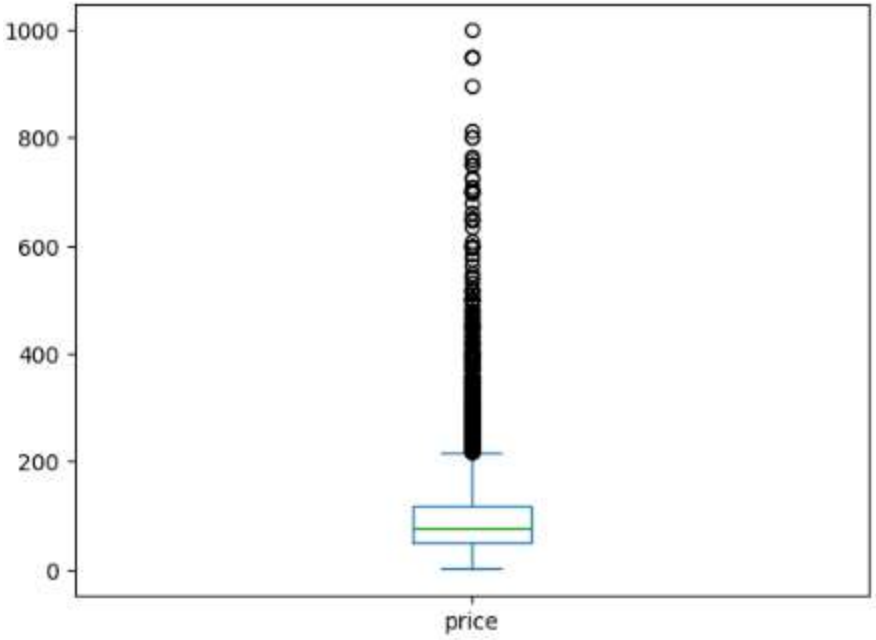


Paris

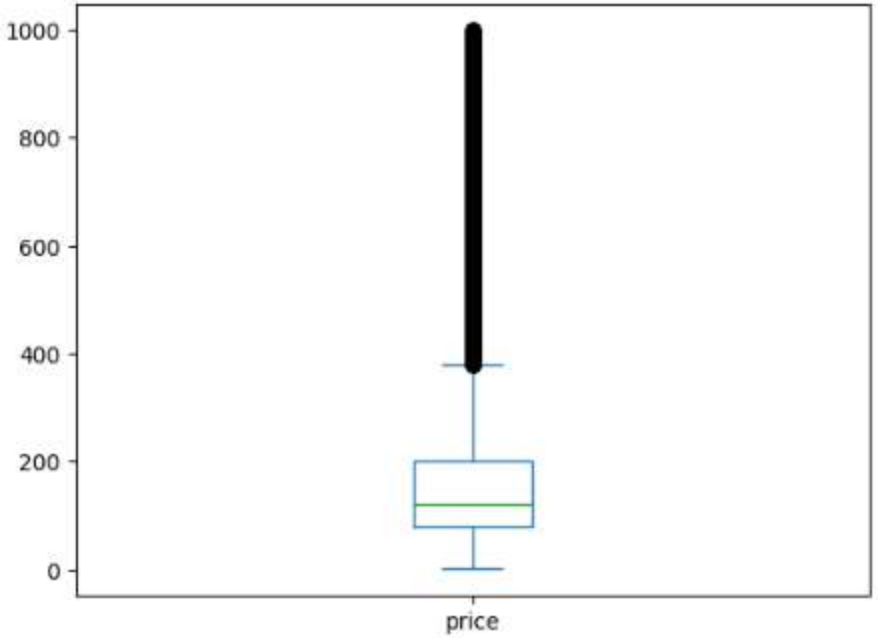


Gestion des valeurs aberrantes

Lyon



Paris



Gestion des valeurs aberrantes

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
print(scaler.fit(lyon2.price.values.reshape(-1, 1)))
print(scaler.data_max_)
print(scaler.transform(lyon2.price.values.reshape(-1, 1)))
```

```
MinMaxScaler()
[347.]
[[0.3583815 ]
 [0.34393064]
 [0.25722543]
 ...
 [0.26589595]
 [0.28612717]
 [0.23121387]]
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
print(scaler.fit(paris2.price.values.reshape(-1, 1)))
print(scaler.data_max_)
print(scaler.transform(paris2.price.values.reshape(-1, 1)))
```

```
MinMaxScaler()
[596.]
[[0.34957983]
 [0.49747899]
 [0.18319328]
 ...
 [0.46890756]
 [0.68067227]
 [0.54453782]]
```

Séparation des données en ensemble d'entraînement et de test

Lyon

```
from sklearn.model_selection import train_test_split
X_l = lyon1.drop('price', axis=1)
y_l = lyon1['price']
from sklearn.model_selection import train_test_split
X_train_l, X_test_l, y_train_l, y_test_l = train_test_split(X_l, y_l, test_size=0.4)
X_l.columns
```

Séparation des données en ensemble d'entraînement et de test

Lyon

Régression linéaire simple:

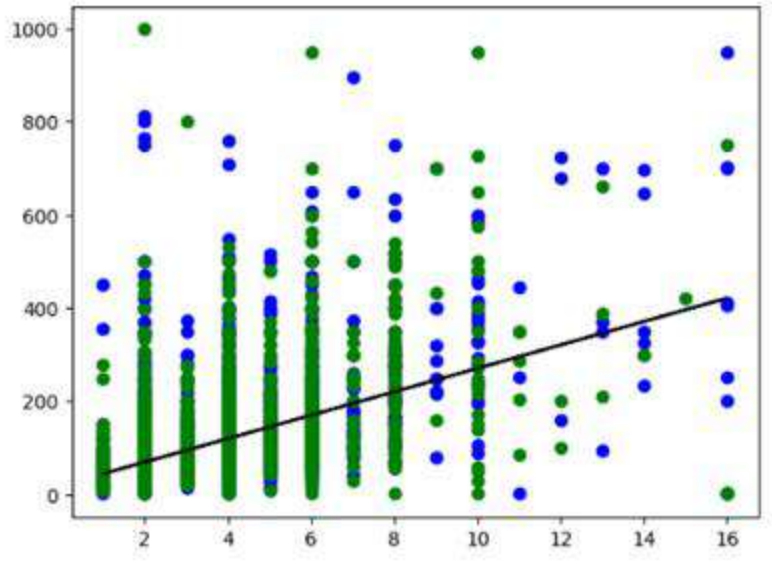
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
reg = LinearRegression()
reg.fit(X_train_l[['accommodates']], y_train_l)
print(reg.score(X_test_l[['accommodates']], y_test_l))
```

```
0.2648982485381898
```

```
[274] y_pred_l = reg.predict(X_test_l[['accommodates']])
      r2_l= r2_score(y_test_l, y_pred_l)
      print(f"Coefficient de détermination R² : {r2_l:.2f}")
```

```
Coefficient de détermination R² : 0.26
```

```
plt.scatter(X_train_l['accommodates'], y_train_l, color ='b')
plt.scatter(X_test_l['accommodates'], y_test_l, color ='g')
plt.plot(X_test_l['accommodates'], y_pred_l, color ='k')
plt.show()
```



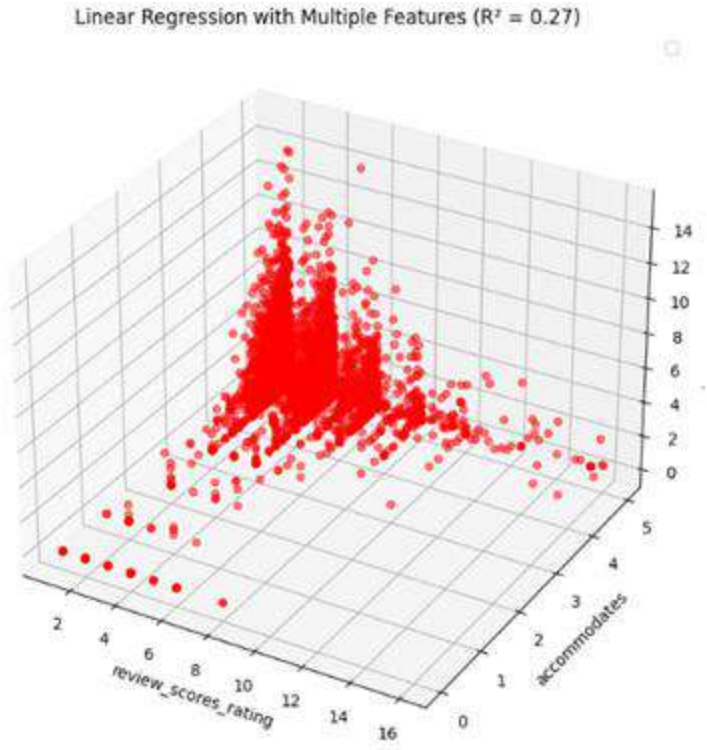
Séparation des données en ensemble d'entraînement et de test

Lyon

Régression linéaire multiple:

```
reg_multi_1 = LinearRegression()

# Entrainement du modèle sur l'ensemble d'entrainement avec la fonction fit()
X_train_multi_1 = X_train_1[['accommodates', 'review_scores_rating', 'reviews_per_month']]
X_test_multi_1 = X_test_1[['accommodates', 'review_scores_rating', 'reviews_per_month']]

reg_multi_1.fit(X_train_multi_1, y_train_1)
print(reg_multi_1.score(X_test_multi_1, y_test_1))

# Prédiction des prix sur l'ensemble de test
y_pred_multi_1 = reg_multi_1.predict(X_test_multi_1)


# Calcul du coefficient de détermination R²

r2_1= r2_score(y_test_1, y_pred_multi_1)
print(f"Coefficient de détermination R² : {r2_1:.2f}")
```

```
0.2666024112421024
Coefficient de détermination R² : 0.27
```



Séparation des données en ensemble d'entraînement et de test

Paris

```
from sklearn.model_selection import train_test_split
X_p = paris1.drop('price', axis=1)
y_p = paris1['price']
from sklearn.model_selection import train_test_split
X_train_p, X_test_p, y_train_p, y_test_p = train_test_split(X_p, y_p, test_size=0.4)
X_p.columns
```

Séparation des données en ensemble d'entraînement et de test

Paris

Régression linéaire simple:

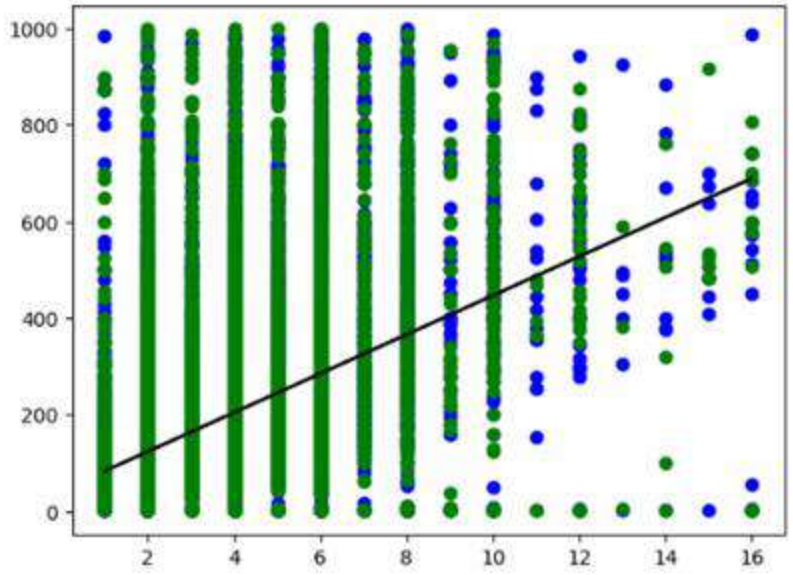
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
reg = LinearRegression()
reg.fit(X_train_p[['accommodates']], y_train_p)
print(reg.score(X_test_p[['accommodates']], y_test_p))
```

```
0.2017011057954663
```

```
[280] y_pred_p = reg.predict(X_test_p[['accommodates']])
      r2_p= r2_score(y_test_p, y_pred_p)
      print(f"Coefficient de détermination R² : {r2_p:.2f}")
```

```
Coefficient de détermination R² : 0.20
```

```
plt.scatter(X_train_p['accommodates'], y_train_p, color ='b')
plt.scatter(X_test_p['accommodates'], y_test_p, color ='g')
plt.plot(X_test_p['accommodates'], y_pred_p, color ='k')
plt.show()
```



Séparation des données en ensemble d'entraînement et de test

Paris

Régression linéaire multiple:

```
reg_multi_p = LinearRegression()

# Entrainement du modèle sur l'ensemble d'entrainement avec la fonction fit()
X_train_multi_p = X_train_p[['accommodates', 'review_scores_rating', 'reviews_per_month']]
X_test_multi_p = X_test_p[['accommodates', 'review_scores_rating', 'reviews_per_month']]

reg_multi_p.fit(X_train_multi_p, y_train_p)
print(reg_multi_p.score(X_test_multi_p, y_test_p))

# Prédiction des prix sur l'ensemble de test
y_pred_multi_p = reg_multi_p.predict(X_test_multi_p)


# Calcul du coefficient de détermination R²

r2_l= r2_score(y_test_p, y_pred_multi_p)
print(f"Coefficient de détermination R² : {r2_p:.2f}")
```

```
0.21123456539557628
Coefficient de détermination R² : 0.20
```