



## Les Plus Beaux Logis de Paris

*Analyser l'évolution des prix de l'immobilier avec Python*

*J. PAOLI*

*Juin 2025*

---

# Les Plus Beaux Logis de Paris

## Partie 1

# I. Analyse du marché de l'immobilier



***Nombre total de transactions analysées*** : 26 196

***Répartition*** : 24 353 appartements / 1 843 locaux commerciaux et assimilés

**QUEL EST LE MEILLEUR PORTEFEUILLE D'ACTIF ?**

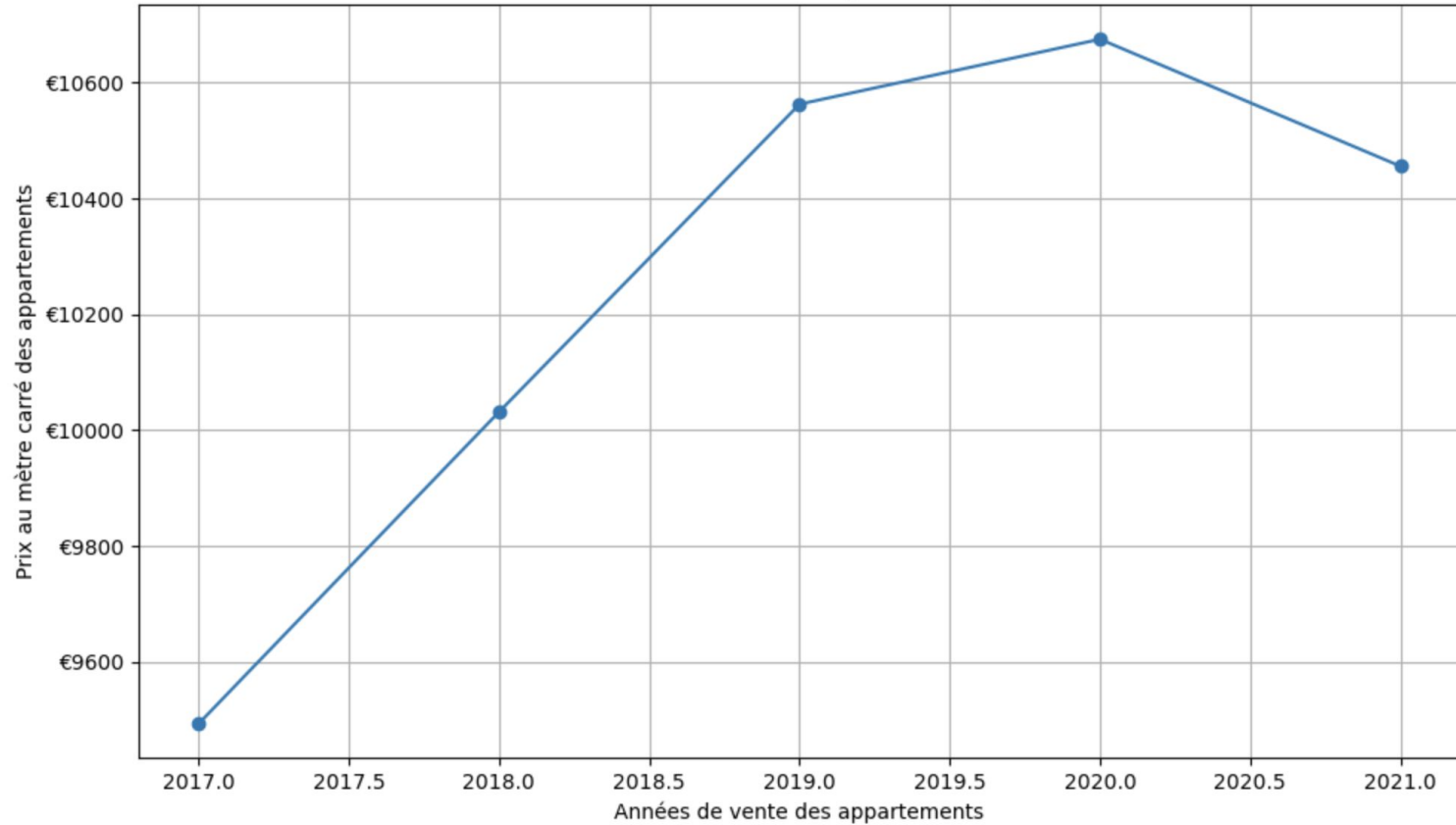
## II. Méthodologie suivie

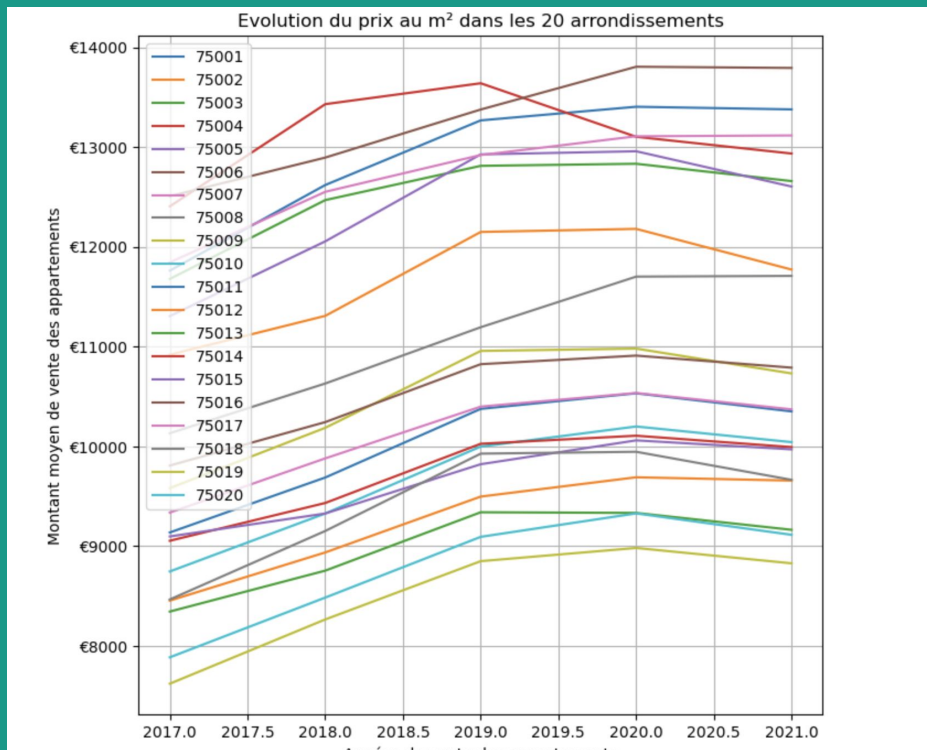
- ❖ Etablissement du prix au Mètre carré moyen ainsi que de la surface moyenne entre 2017 et 2021

Prix moyen au m<sup>2</sup> et surface moyenne des appartements par année :

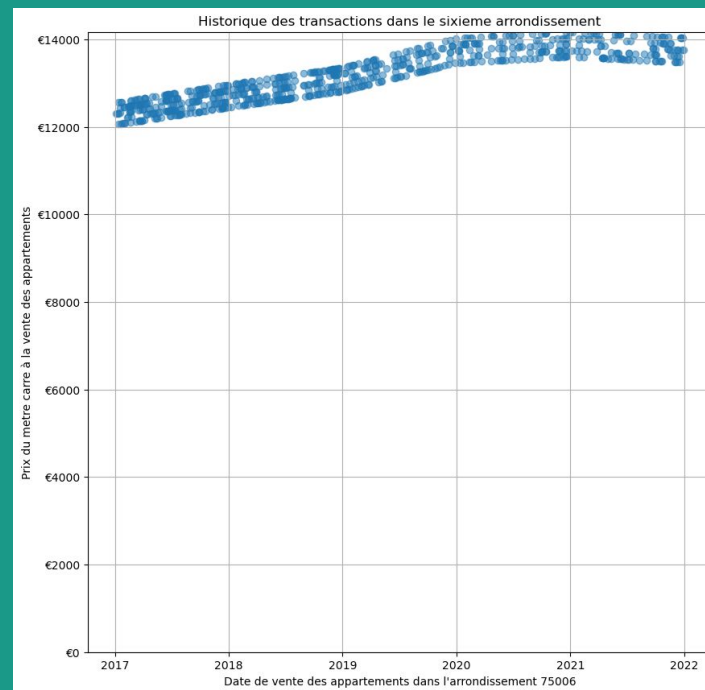
	<b>annee</b>	<b>prix_m2</b>	<b>surface_reelle</b>
<b>0</b>	2017	9492.859195	44.627737
<b>1</b>	2018	10031.403432	44.271671
<b>2</b>	2019	10562.712581	43.361645
<b>3</b>	2020	10674.872650	42.900654
<b>4</b>	2021	10455.600126	43.479864

Visualisation de la hausse moyenne des prix à Paris





- ❖ Suivi de l'évolution des prix du mètre carré dans tous les arrondissements
- ❖ Suivi des ventes avec le 6ème arrondissement en exemple



Le coefficient de corrélation de Spearman est : 0.9148  
P-value associée : 1.2000e-280

❖ Vérification de la véracité de notre hypothèse : est-ce que la surface d'un bien a un impact sur son prix ?

```
from scipy.stats import pearsonr

# Suppression des lignes avec valeurs manquantes
df_corr = df_appart.dropna(subset=['valeur_fonciere', 'surface_reelle'])

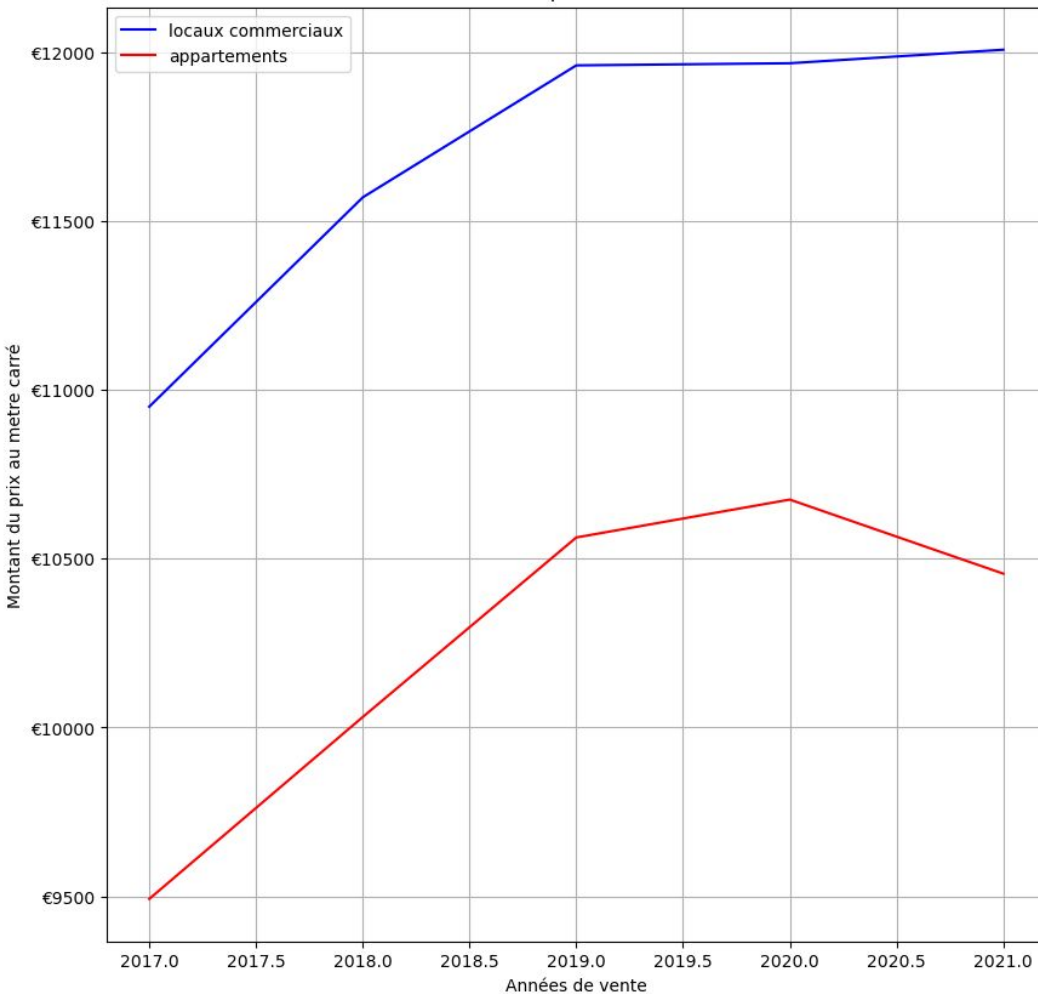
# Calcul de la corrélation de Pearson
coef, p = pearsonr(df_corr['surface_reelle'], df_corr['valeur_fonciere'])

print(f"Le coefficient de corrélation est de : {coef:.4f}")
print(f"P-value associée : {p:.4e}")
```

```
Le coefficient de corrélation est de : 0.9801
P-value associée : 0.0000e+00
```

```
# Le coefficient de corrélation est de 0.9801 avec une p-value de 0.0000e+00.
# La relation entre la valeur foncière et la surface est donc très forte, linéaire et statistiquement significative.
# Cela confirme que plus la surface est grande, plus le bien est cher.
```

Evolution des prix au metre carré



❖ Suivi de l'évolution des prix selon le type de local



# III. Résultat de l'Analyse



- Les prix ont **augmenté** partout
- La surface, elle, a « **diminué** »
- Cela a une **incidence** sur les ventes
- Les Locaux commerciaux restent ceux avec la plus **grosse valeur foncière**

# IV. L'algorithme de prédiction



## **A. Entraînement de l'algorithme :**

1. *Nettoyage des données : suppression des colonnes inutiles*
2. *Encodage one-hot : pour arrondissement et type de bien*
3. *Transformation : création de la variable prix\_m2*
4. *Séparation chronologique du dataset :*  
*Train = 2017-2020*  
*Test = 2021*
5. *Modèle : Régression Linéaire simple*
6. *Test de l'algorithme*

```
# On sépare le jeu de données entre échantillons d'apprentissage et de test

# 1. Vérifier que la date est bien en format datetime
df_encoded["date_mutation"] = pd.to_datetime(df_encoded["date_mutation"])

# 2. Split chronologique
train_df = df_encoded[df_encoded["date_mutation"].dt.year < 2021]
test_df = df_encoded[df_encoded["date_mutation"].dt.year == 2021]

# 3. Séparation entre variables explicatives et cible
X_train = train_df.drop(["valeur_fonciere", "date_mutation"], axis=1)
y_train = train_df["valeur_fonciere"]

X_test = test_df.drop(["valeur_fonciere", "date_mutation"], axis=1)
y_test = test_df["valeur_fonciere"]
```

```
from sklearn.linear_model import LinearRegression

# 1. Instanciation du modèle
model = LinearRegression()

# 2. Entraînement
model.fit(X_train, y_train)

# 3. Prédiction sur l'ensemble de test
y_pred = model.predict(X_test)

# 4. Affichage des prédictions comparées aux valeurs réelles
resultats = pd.DataFrame({
    'valeur_fonciere_reelle': y_test.values,
    'valeur_fonciere_predite': y_pred})

resultats.head(20)
```

	valeur_fonciere_reelle	valeur_fonciere_predite
0	3.009804e+05	3.232311e+05
1	1.056941e+06	9.866556e+05
2	2.308898e+05	2.772439e+05
3	3.865867e+05	3.747066e+05
4	3.460434e+05	3.733779e+05
5	3.833480e+06	2.918911e+06
6	1.280223e+06	1.163081e+06
7	9.264516e+05	8.468044e+05
8	2.483348e+05	2.875634e+05
9	6.394606e+05	6.029438e+05
10	2.763924e+05	3.177622e+05
11	6.767915e+05	6.236998e+05
12	5.038063e+05	4.895850e+05
13	3.527256e+05	3.696715e+05
14	5.099047e+05	4.934083e+05
15	3.197691e+05	4.228095e+05
16	5.830258e+05	5.540397e+05
17	3.019500e+05	3.029181e+05
18	6.336784e+05	5.931893e+05
19	8.400401e+05	7.631768e+05




## ❖ Calcul du % d'erreur absolu de l'algorithme

```
# Calcul de l'erreur moyenne absolue en pourcentage de la valeur réelle
erreurs_absolues = abs(y_test - y_pred)
erreur_moyenne_pourcent = (erreurs_absolues / y_test) * 100
erreur_moyenne_finale = erreur_moyenne_pourcent.mean()

# Affichage du résultat
print(f"Notre algorithme fait donc {erreur_moyenne_finale:.2f} % d'erreur en moyenne sur la prédiction de la valeur foncière.")
```

Notre algorithme fait donc 8.79% d'erreur en moyenne sur la prédiction de la valeur foncière.

Mes conclusions sur ce résultat et comment j'aurais pu aller plus loin :

-  Le score est inférieur au seuil de 10 %, donc l'objectif est atteint avec un modèle de régression linéaire simple.
-  Ce modèle suppose une relation linéaire entre les variables et la valeur foncière.
-  J'aurais pu :
  - tester d'autres modèles (régression ridge/lasso, arbres de décision, random forest), et surtout une méthode sklearn plus complexe mais plus fiable
  - analyser plus finement les erreurs (ex : erreurs très fortes sur les biens très chers ? sur certaines zones géographiques ?)



## **B. Test en situation réelle :**

1. *Nettoyage des données : suppression des colonnes inutiles (adresse, localisation, etc....)*
2. *Encodage one-hot : pour arrondissement et type de bien*
3. *Transformation : création d'une colonne surface\_relle*
4. *Lancement et étude de la valorisation*

Maintenant nous allons comparer la valorisation prédite pour les deux segments.

```
] : #Valorisation du portefeuille sur le segment des particuliers
# On ajoute les prédictions dans le DataFrame d'origine
df_actifs["valeur_fonciere_predite"] = y_pred_portefeuille

#Valorisation du portefeuille sur le segment des particuliers
val_particuliers = df_actifs[df_actifs["type_local"] == "Appartement"]["valeur_fonciere_predite"].sum() / 1_000_000
print('la valorisation du segment particulier est (en millions deuros):')
print(round(val_particuliers, 2))
```

```
la valorisation du segment particulier est (en millions deuros):
62.68
```

```
!] : #Valorisation du portefeuille sur le segment corporate
val_corporate = df_actifs[df_actifs["type_local"] != "Appartement"]["valeur_fonciere_predite"].sum() / 1_000_000
print('la valorisation du segment corporate est (en millions deuros):')
print(round(val_corporate, 2))
```

```
la valorisation du segment corporate est (en millions deuros):
91.28
```

---

# Les Plus Beaux Logis de Paris

## Partie 2

# I. Méthodologie suivie

## Classification des données issues du jeu de test :

### 1. Application d'un **K-Means** avec 2 clusters

*D'abord en créant une colonne prix au m2 ;*

*Puis en lançant le K-Means*



1

```
# On calcule le prix au mètre carré
df_clustering["prix_m2"] = df_clustering["valeur_fonciere"] / df_clustering["surface_reelle"]

# On supprime les colonnes dont l'information est désormais résumée dans prix_m2
df_clustering = df_clustering.drop(["valeur_fonciere", "surface_reelle"], axis=1)

df_clustering.head()
```

2

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# On sélectionne uniquement la colonne prix_m2 pour le clustering
X = df_clustering[["prix_m2"]]

# Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Application de KMeans avec 2 clusters
kmeans = KMeans(n_clusters=2, random_state=42)
df_clustering["cluster"] = kmeans.fit_predict(X_scaled)
```



	<b>code_postal</b>	<b>nom_commune</b>	<b>prix_m2</b>	<b>cluster</b>
<b>0</b>	75019	Paris 19e Arrondissement	9871.444128	0
<b>1</b>	75019	Paris 19e Arrondissement	10045.572493	0
<b>2</b>	75019	Paris 19e Arrondissement	9194.697790	0
<b>3</b>	75019	Paris 19e Arrondissement	9469.142168	0
<b>4</b>	75019	Paris 19e Arrondissement	7463.610005	1

## II. Résultat de la classification



En tout cas, le second algorithme classe automatiquement grâce à la méthode K-Means les Appartements et les Locaux commerciaux ;

Les Locaux commerciaux sont ceux avec un prix au mètre carré supérieur

Mais lui aussi a ses limites :

- Il n'y a que deux clusters ce qui limite l'analyse
- Encore une fois il ne prend en compte que des données simples

---

**MERCI POUR VOTRE ATTENTION !**