Les Plus Beaux Logis de Paris

Analyse de l’évolution des prix de l’immobilier via Python

Présenté par Gaspard-Fauvelle Angel

Analyse exploratoire des Données

Datasets :

- echantillon_a_classer.xlsx
- portefeuille_actifs.xlsx
- historique_immobilier_paris_2017_2021_vdef2.xlsx

Caractéristiques :

- Historique:
  - 9 colonnes, 26196 lignes
- Portefeuille:
  - 12 colonnes, 275 lignes
- Echantillon:
  - 4 colonnes, 40 lignes

Traitement réalisés :

- Nettoyages des données par la vérification, puis l'éventuelle suppression des :
  - lignes/valeurs dupliquées ou nulles ;
  - Conversions des types de données (réduire le poids du jeu de données)
- Analyse multivariée :
  - Création des sous-tableaux, contenant que les critères demandés ;
  - Préparation des graphiques se servant des sous-tableaux ;
  - Evaluation des corrélations ;

Processus et résultat de la conversion des jeux de données

Avant conversion (total de 1,9Mega octets) :

```
Voici les informations du tableau Historique:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26196 entries, 0 to 26195
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   date_mutation     26196 non-null  datetime64[ns]
 1   valeur_fonciere   26196 non-null  float64
 2   adresse_numero    26196 non-null  int64
 3   adresse_nom_voie  26196 non-null  object
 4   code_postal       26196 non-null  int64
 5   nom_commune       26196 non-null  object
 6   code_type_local   26196 non-null  int64
 7   type_local        26196 non-null  object
 8   surface_reelle    26196 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(4), object(3)
memory usage: 1.8+ MB
```

```
Voici les informations du tableau Portefeuille:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275 entries, 0 to 274
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   adresse_numero            275 non-null    int64
 1   adresse_nom_voie          275 non-null    object
 2   code_postal               275 non-null    int64
 3   code_commune              275 non-null    int64
 4   nom_commune               275 non-null    object
 5   surface_carrez            275 non-null    float64
 6   code_type_local           275 non-null    int64
 7   type_local                275 non-null    object
 8   surface_reelle_bati       275 non-null    int64
 9   nombre_pieces_principales 275 non-null    int64
 10  longitude                 275 non-null    float64
 11  latitude                  275 non-null    float64
dtypes: float64(3), int64(6), object(3)
memory usage: 25.9+ KB
```

```
Voici les informations du tableau Échantillon:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   valeur_fonciere  40 non-null     float64
 1   code_postal      40 non-null     int64
 2   nom_commune      40 non-null     object
 3   surface_reelle   40 non-null     int64
dtypes: float64(1), int64(2), object(1)
memory usage: 1.4+ KB
```

Après conversion (total de 0,776 Mega octets) :

```
Voici les informations du tableau Historique:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26196 entries, 0 to 26195
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   date_mutation     26196 non-null  datetime64[ns]
 1   valeur_fonciere   26196 non-null  float32
 2   adresse_numero    26196 non-null  int32
 3   adresse_nom_voie  26196 non-null  category
 4   code_postal       26196 non-null  category
 5   nom_commune       26196 non-null  category
 6   code_type_local   26196 non-null  category
 7   type_local        26196 non-null  category
 8   surface_reelle    26196 non-null  uint32
dtypes: category(5), datetime64[ns](1), float32(1), int32(1), uint32(1)
memory usage: 753.9 KB
```

```
Voici les informations du tableau Portefeuille type de données convertis:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275 entries, 0 to 274
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   adresse_numero            275 non-null    int32
 1   adresse_nom_voie          275 non-null    category
 2   code_postal               275 non-null    category
 3   code_commune              275 non-null    category
 4   nom_commune               275 non-null    category
 5   surface_carrez            275 non-null    float64
 6   code_type_local           275 non-null    category
 7   type_local                275 non-null    category
 8   surface_reelle            275 non-null    uint32
 9   nombre_pieces_principales 275 non-null    int32
 10  longitude                 275 non-null    float64
 11  latitude                  275 non-null    float64
 12  year_mutation             275 non-null    int64
dtypes: category(6), float64(3), int32(2), int64(1), uint32(1)
memory usage: 21.7 KB
```

```
Voici les informations du tableau Échantillon:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   valeur_fonciere  40 non-null     float32
 1   code_postal      40 non-null     category
 2   nom_commune      40 non-null     category
 3   surface_reelle   40 non-null     uint32
dtypes: category(2), float32(1), uint32(1)
memory usage: 764.0 bytes
```

Nettoyage et préparation des données

```
Dans la table Historique, il se trouve 16 lignes dupliquées, aucune cellule n'est vide.
```

```
Dans la table Portefeuille, il se trouve 2 lignes dupliquées, aucune cellule n'est vide.
```

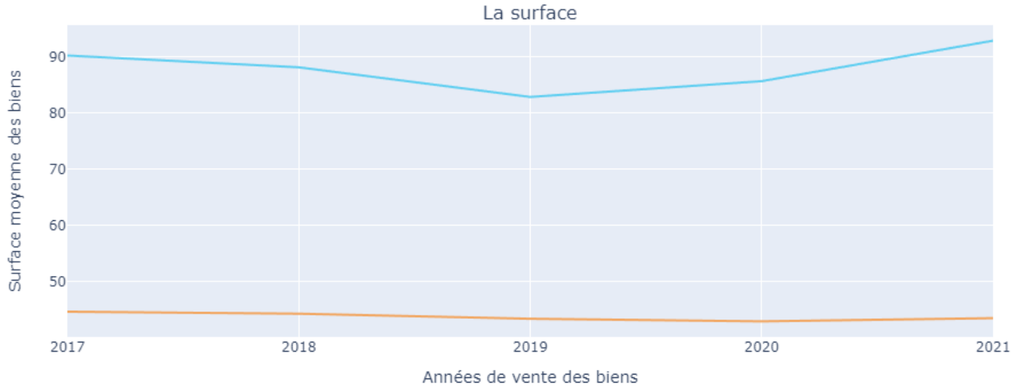
La liste des sous-tableaux créés :

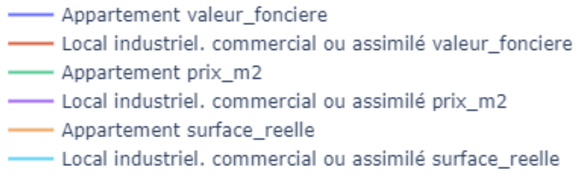
- filtrée ou non, des types de biens, du code postal ;
- Sélection des colonnes :
  - Valeur foncière, prix au m², code postal, surface réelle

```
Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique prix au m2

Les types de biens inclus dans ce jeu de données sont de types : Appartement et Local industriel. commercial ou assimilé.
Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique prix au m2
    filters: {'type_local': 'Appartement'}

Les types de biens inclus dans ce jeu de données sont de types : Appartement.
Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    group_by_date: True
    Les colonnes extraites sont : {'valeur_fonciere'}

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    filters: {'code_postal': 75001}
    group_by_date: True
    Les colonnes extraites sont : ['prix_m2']

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique prix au m2
    Les colonnes extraites sont : {'code_postal'}

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    filters: {'code_postal': 75001}
    Les colonnes extraites sont : ['valeur_fonciere', 'surface_reelle']
```

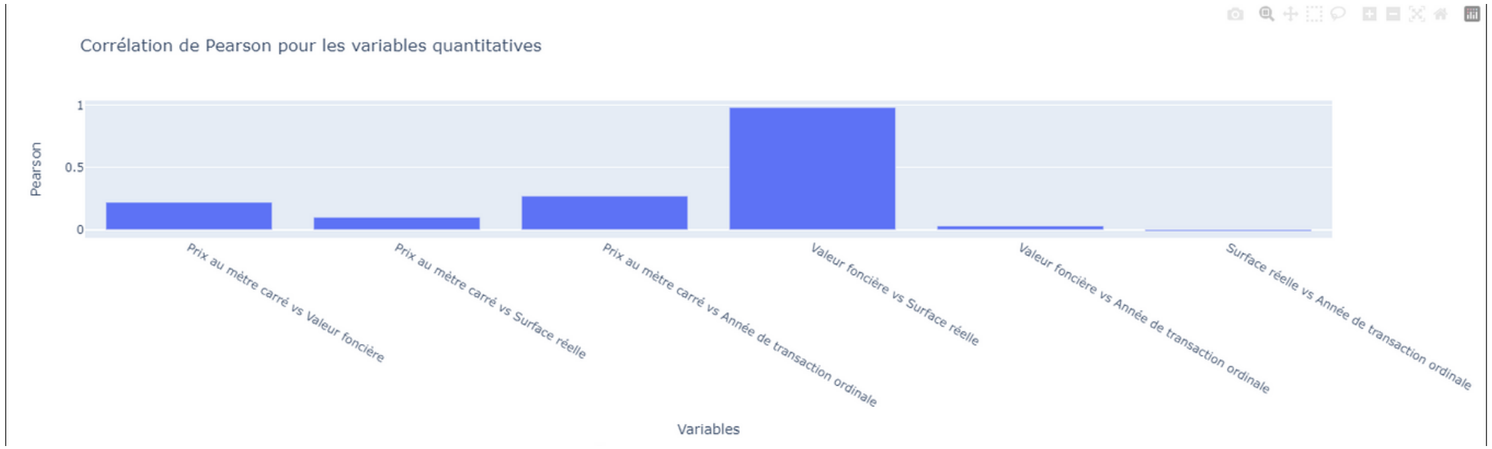
```
Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique prix au m2
    Les colonnes extraites sont : {'code_postal'}

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    filters: {'code_postal': 75001}
    Les colonnes extraites sont : ['valeur_fonciere', 'surface_reelle']

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    group_by_date: True
    Les colonnes extraites sont : ['prix_m2']

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    filters: {'code_postal': 75006}
    Les colonnes extraites sont : {'year_mutation'}

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique appartements
    filters: {'code_postal': 75006}
    Les colonnes extraites sont : ['year_mutation', 'prix_m2']

Appel de la fonction request_analyze_graph avec les arguments suivants :
    nom_csv: historique des biens
    filters: {'code_postal': 75006}
    Les colonnes extraites sont : ['type_local', 'prix_m2']
```

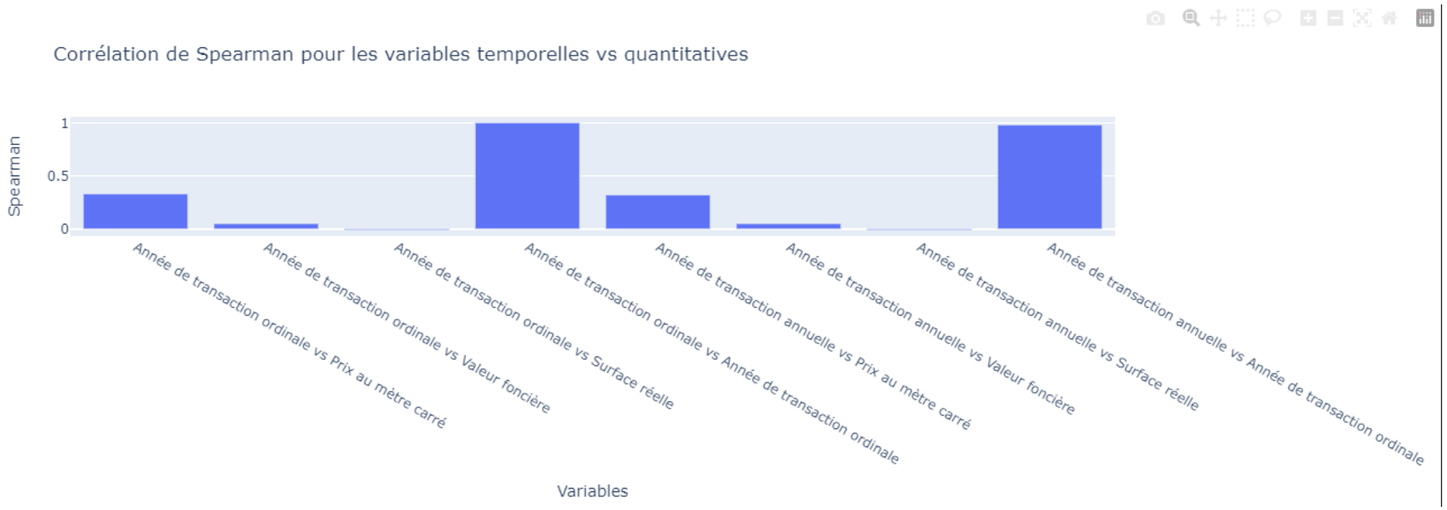
Compilation des graphiques retirés



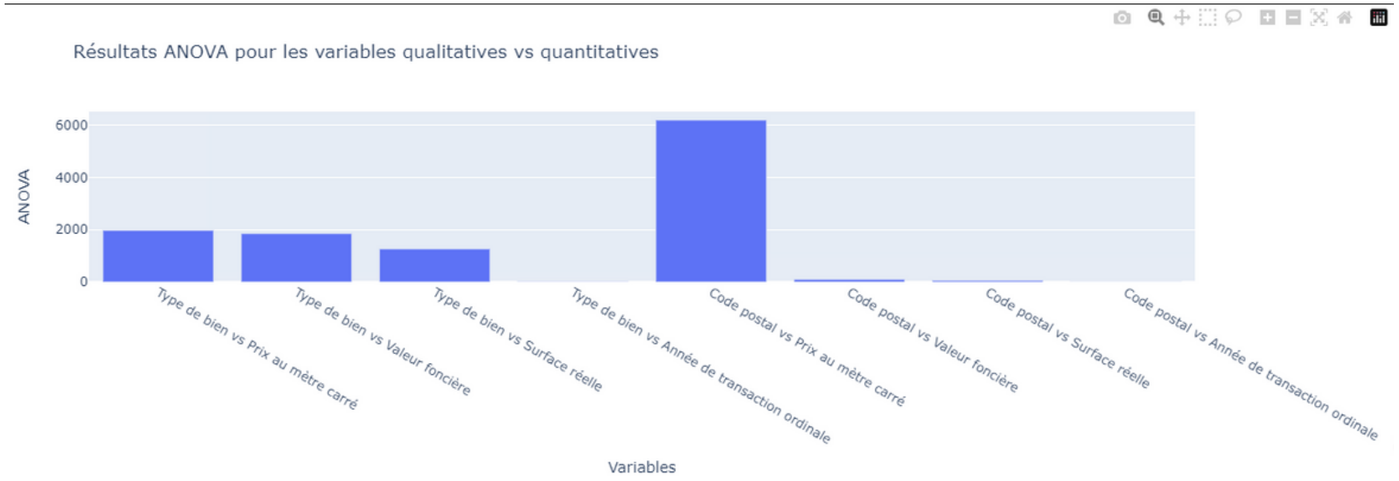


Vérification des corrélations





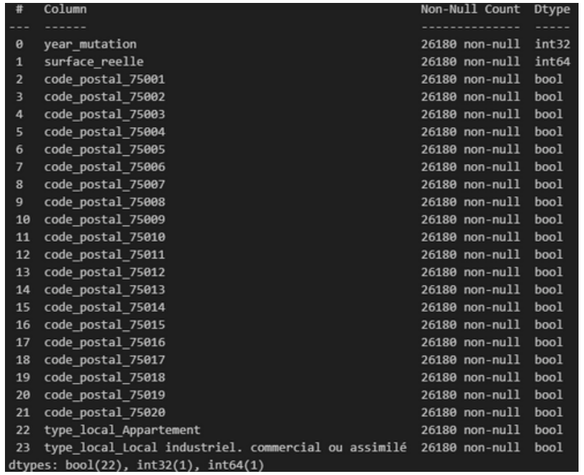
Vérification des corrélations



Pré processus des données

Méthode employés :

- dummies afin de créer les features code postal et type de bien ;
- Attribution des features sous x ;
- Atribution de la cible sous y (par conséquent retiré du jeu de données) ;



#	Column	Non-Null Count	Dtype
0	year_mutation	26180 non-null	int32
1	surface_reelle	26180 non-null	int64
2	code_postal_75001	26180 non-null	bool
3	code_postal_75002	26180 non-null	bool
4	code_postal_75003	26180 non-null	bool
5	code_postal_75004	26180 non-null	bool
6	code_postal_75005	26180 non-null	bool
7	code_postal_75006	26180 non-null	bool
8	code_postal_75007	26180 non-null	bool
9	code_postal_75008	26180 non-null	bool
10	code_postal_75009	26180 non-null	bool
11	code_postal_75010	26180 non-null	bool
12	code_postal_75011	26180 non-null	bool
13	code_postal_75012	26180 non-null	bool
14	code_postal_75013	26180 non-null	bool
15	code_postal_75014	26180 non-null	bool
16	code_postal_75015	26180 non-null	bool
17	code_postal_75016	26180 non-null	bool
18	code_postal_75017	26180 non-null	bool
19	code_postal_75018	26180 non-null	bool
20	code_postal_75019	26180 non-null	bool
21	code_postal_75020	26180 non-null	bool
22	type_local_Appartement	26180 non-null	bool
23	type_local_Local industriel. commercial ou assimilé	26180 non-null	bool

dtypes: bool(22), int32(1), int64(1)

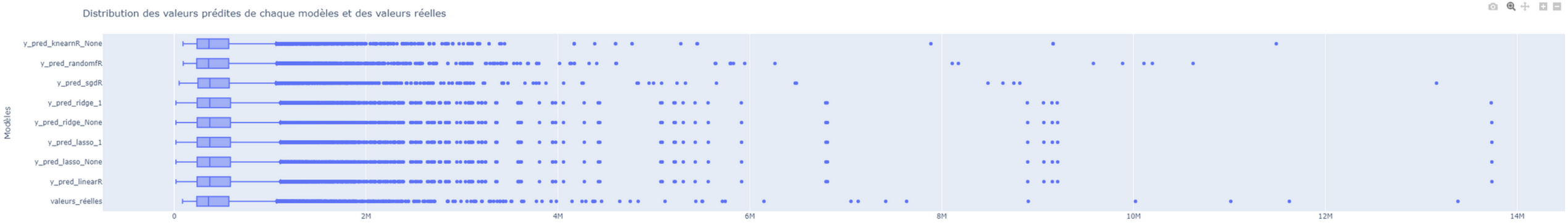
Mise à l’échelle des features et cibles ;

Séparation en sous-ensemble des datasets en données d’entraînements et de tests, de respectivement 33% et 67% :

```
X_train, X_test, y_train, y_test = tts(X_scaled, y, test_size=0.33, random_state=2)
```

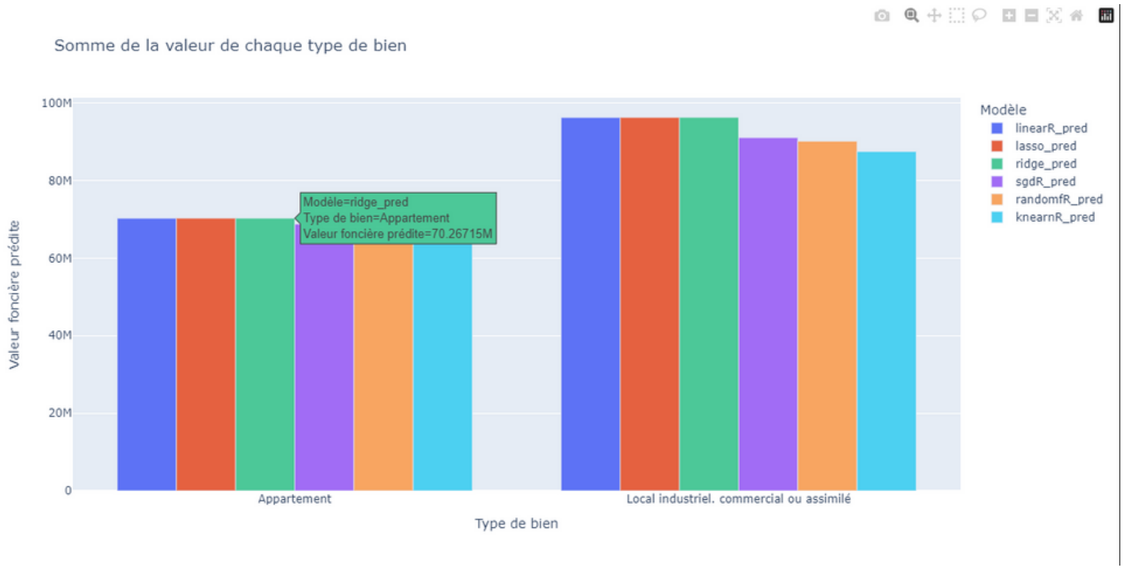
Résultats

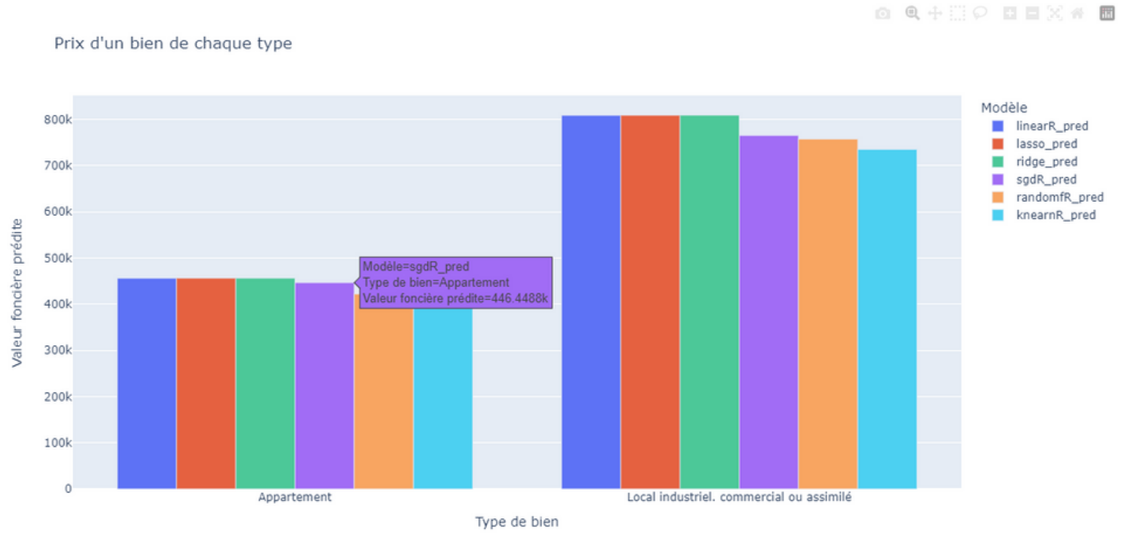
	Modèle	n_neighbors	alpha	Temps d'exécution (secondes)	MAPE
6	RandomForestRegressor	NaN	NaN	0.46	2.22
7	KNeighborsRegressor	1.00	NaN	0.61	3.31
8	KNeighborsRegressor	3.00	NaN	0.50	3.33
9	KNeighborsRegressor	5.00	NaN	0.52	3.63
5	SGDRegressor	NaN	NaN	0.02	7.01
0	LinearRegression	NaN	NaN	0.01	8.72
1	Lasso	NaN	1.00	2.84	8.72
2	Lasso	NaN	10.00	0.28	8.72
3	Ridge	NaN	1.00	0.00	8.72
4	Ridge	NaN	10.00	0.01	8.72

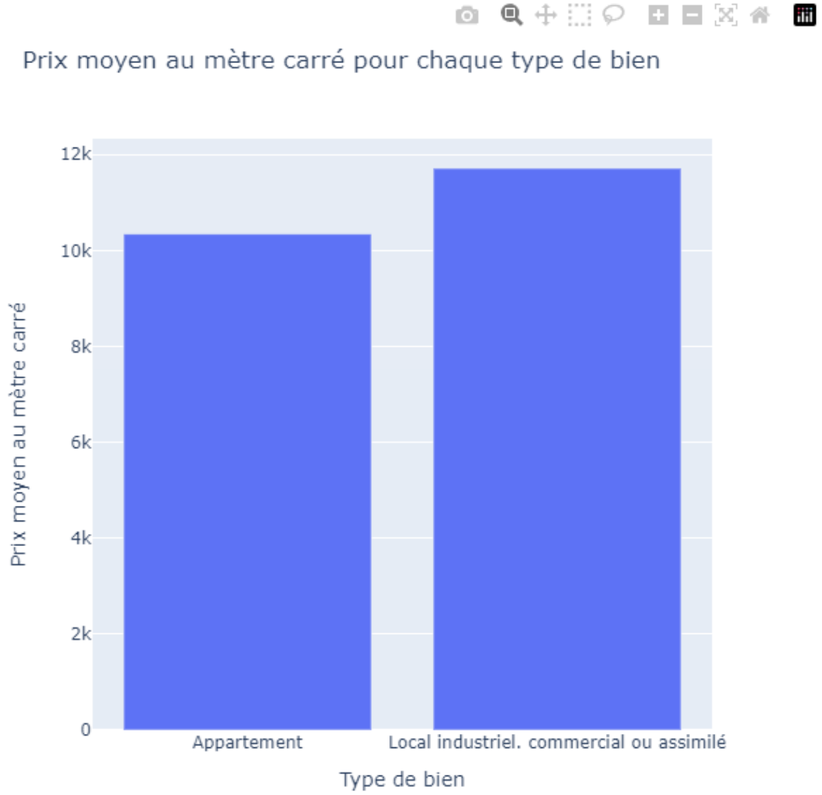


Graphique en boite de la valeur foncière pour chaque modèles (en dernier se trouve les valeurs issu du jeu de données)

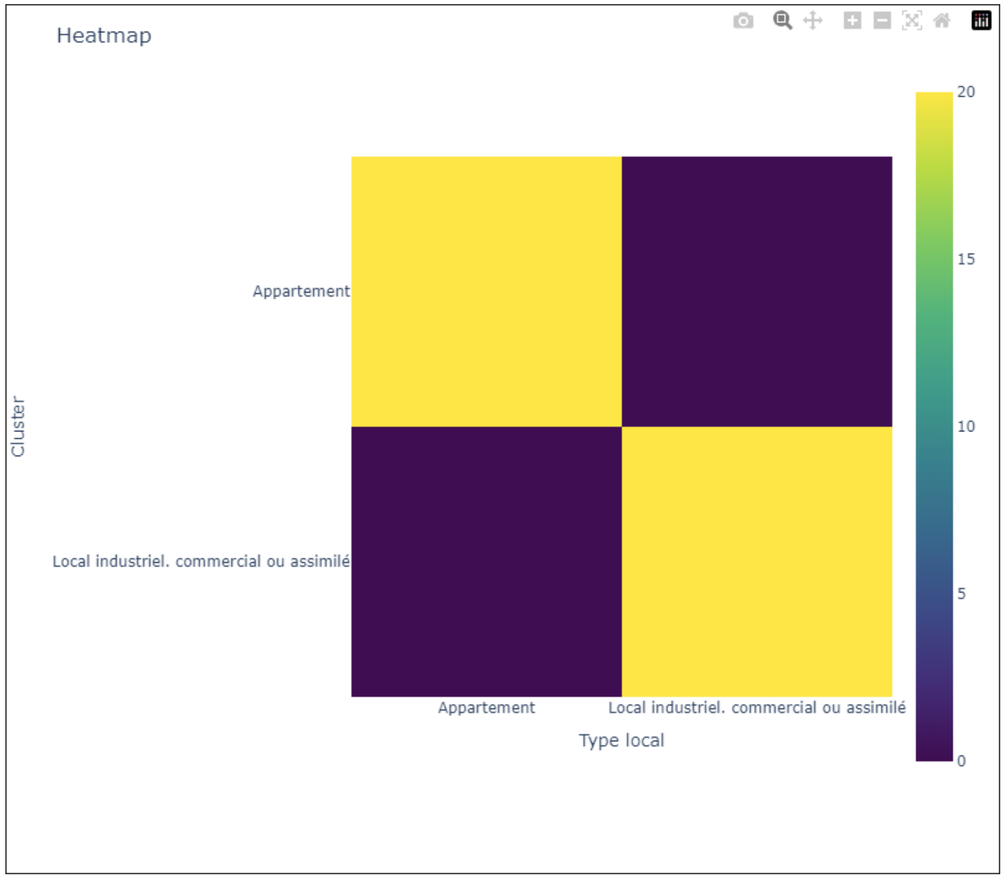
Applications des prédiction sur le portefeuille



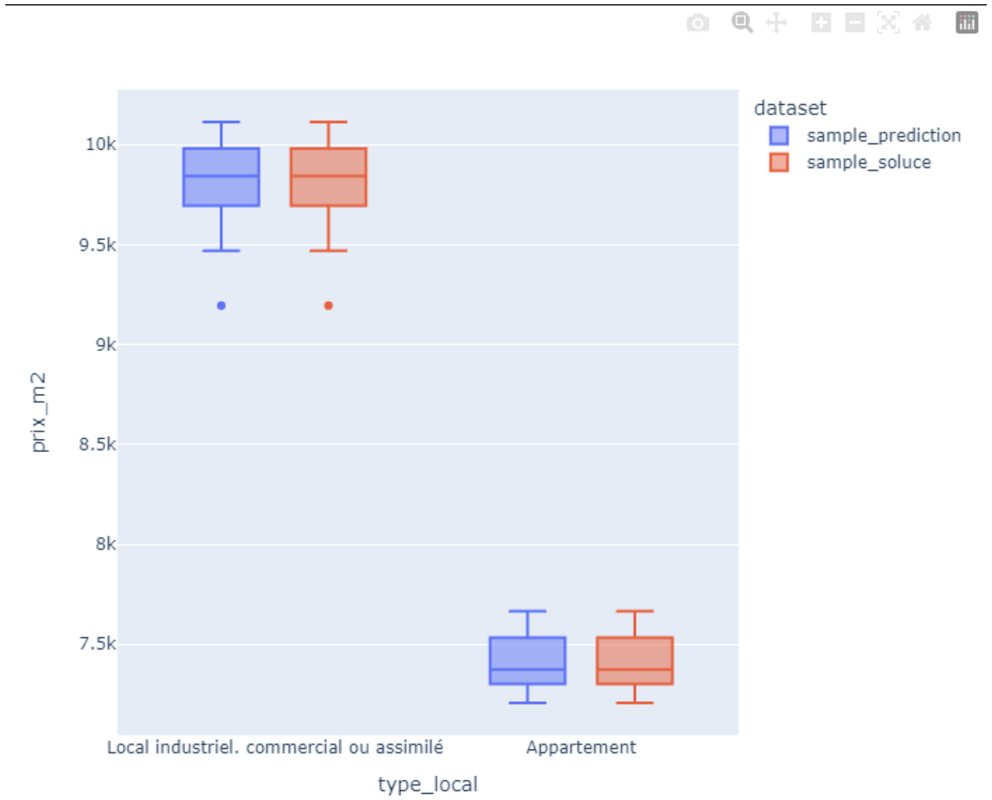




Résultat du clustering sur le jeu de données des échantillons



Distribution des catégories selon le type de bien



Graphique en boite du prix au m² des types de biens

Merci d'avoir suivi cette présentation