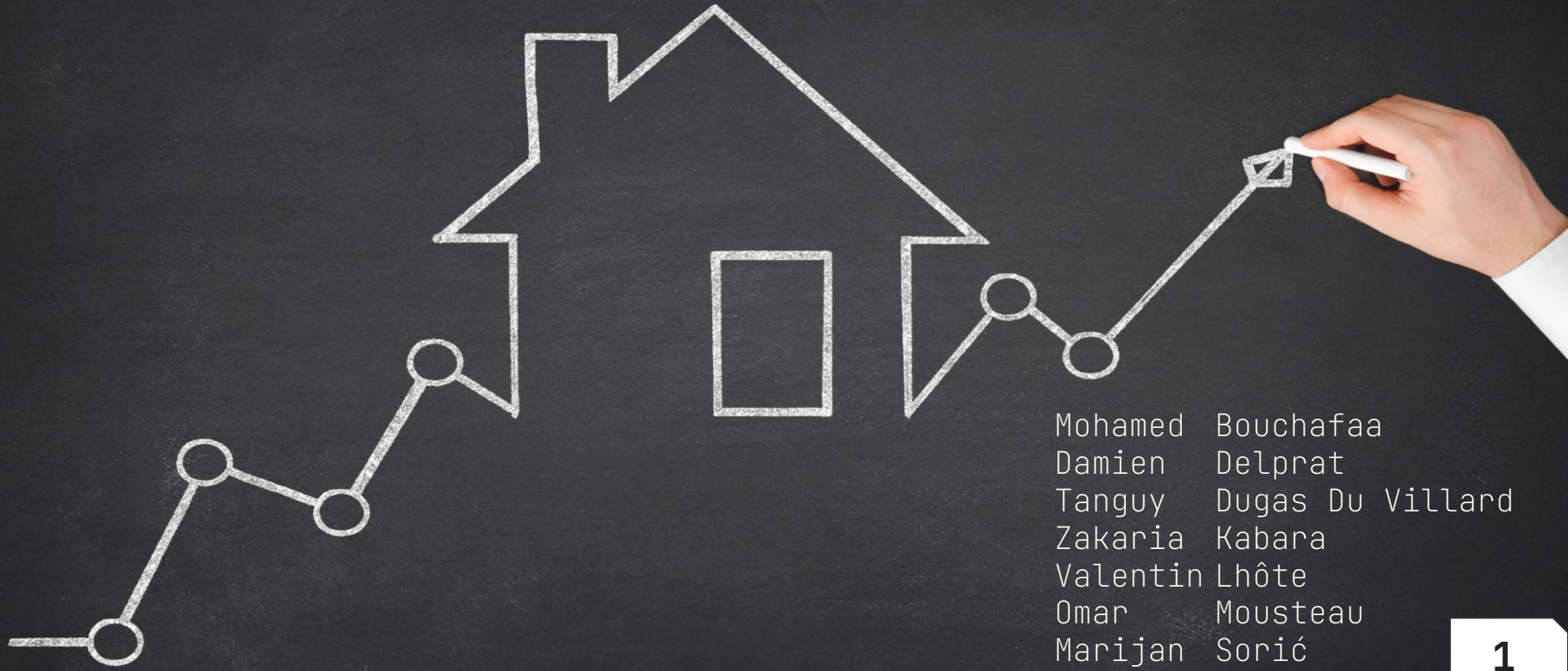
# Challenge Machine Learning




Mohamed Bouchafaa
Damien Delprat
Tanguy Dugas Du Villard
Zakaria Kabara
Valentin Lhôte
Omar Mousteau
Marijan Sorić

# Sommaire :

# Contexte du challenge :

REAL ESTATE


→ **Principe** : Réaliser un modèle pour prédire le prix d’un bien immobilier.

# Contexte du challenge :

Données à disposition :

x_train

*variables explicatives pour l'entraînement*

y_train

*variable(s) cible(s) pour l'entraînement*

x_test

*variables explicatives pour le test*

Structure des résultats:

Fichier csv :

→ Colonne id_annonce

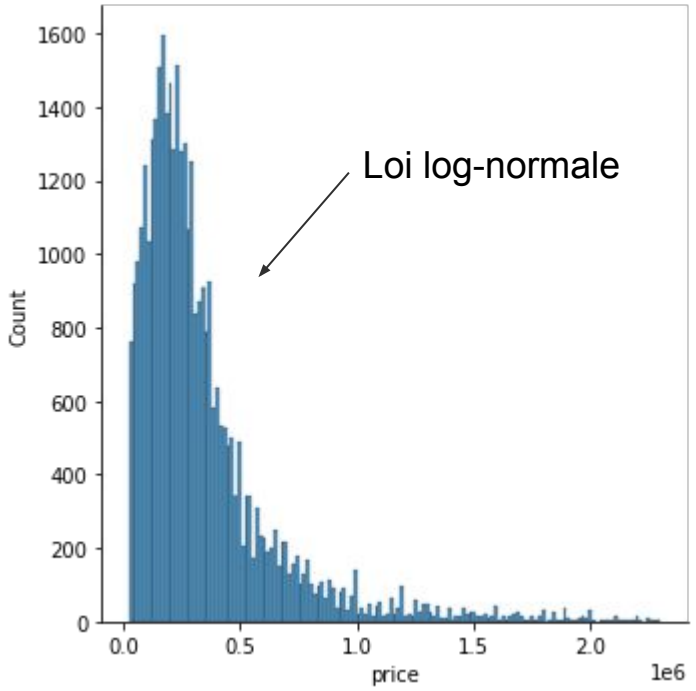
→ Colonne Price

# Analyse des données :

Features :

| | |
|---|---|
| id_annonce | int64 |
| property_type | object |
| approximate_latitude | float64 |
| approximate_longitude | float64 |
| city | object |
| postal_code | int64 |
| size | float64 |
| floor | float64 |
| land_size | float64 |
| energy_performance_value | float64 |
| energy_performance_category | object |
| ghg_value | float64 |
| ghg_category | object |
| exposition | object |
| nb_rooms | float64 |
| nb_bedrooms | float64 |
| nb_bathrooms | float64 |
| nb_parking_places | float64 |
| nb_boxes | float64 |
| nb_photos | float64 |
| has_a_balcony | float64 |
| nb_terraces | float64 |
| has_a_cellar | float64 |
| has_a_garage | float64 |
| has_air_conditioning | float64 |
| last_floor | float64 |
| upper_floors | float64 |

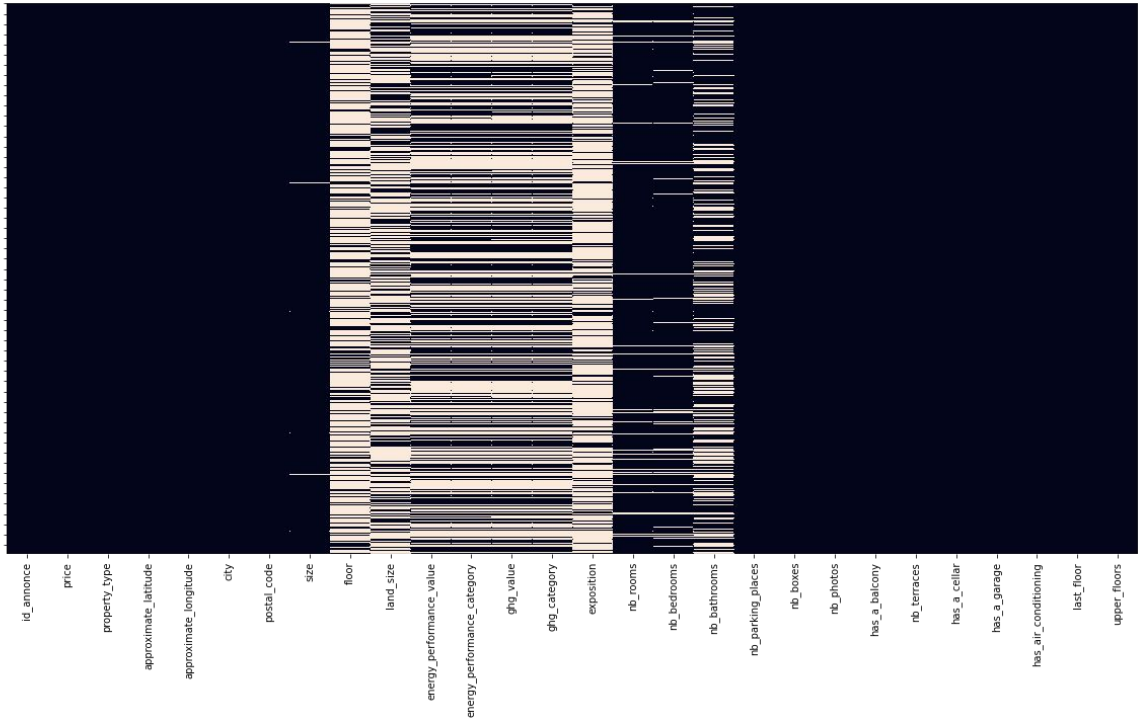
Répartition des prix :



# Valeurs manquantes :

Pourcentage de valeurs manquantes :

| | |
|---|---|
| id_annonce | 0.000000 |
| has_air_conditioning | 0.000000 |
| has_a_garage | 0.000000 |
| has_a_cellar | 0.000000 |
| nb_terraces | 0.000000 |
| has_a_balcony | 0.000000 |
| nb_photos | 0.000000 |
| nb_boxes | 0.000000 |
| nb_parking_places | 0.000000 |
| last_floor | 0.000000 |
| upper_floors | 0.000000 |
| price | 0.000000 |
| property_type | 0.000000 |
| approximate_latitude | 0.000000 |
| approximate_longitude | 0.000000 |
| postal_code | 0.000000 |
| city | 0.000000 |
| nb_rooms | 0.355940 |
| size | 0.648319 |
| nb_bedrooms | 7.131507 |
| nb_bathrooms | 37.310113 |
| floor | 38.994470 |
| energy_performance_category | 50.746838 |
| energy_performance_value | 50.746838 |
| ghg_value | 51.840081 |
| ghg_category | 51.840081 |
| exposition | 71.766351 |
| land_size | 95.893981 |



# Preprocessing :

1) *Quelles **colonnes** prendre en compte ?*
2) *Comment gérer les **valeurs manquantes** ?*
3) *Comment gérer les **valeurs catégorielles** ?*
4) *Quelles **modifications supplémentaires** apporte-t-on aux données?*

# Choix des colonnes :

Suppression de certains features :

- ***postal_code*** → On a déjà l'information de la ville
- ***energy_performance***
- ***ghg***
- ***exposition***

| | |
|---|---|
| energy_performance_category | 48.972383 |
| energy_performance_value | 48.972383 |
| ghg_value | 50.412117 |
| ghg_category | 50.412117 |
| land_size | 58.303896 |
| floor | 73.926889 |
| exposition | 75.663669 |

Pourcentages de valeurs manquantes les plus élevés

# Valeurs manquantes :

Pour chaque colonne avec des valeurs manquantes il a fallu trouver une solution

En effet, même le X_test.csv contient des valeurs manquantes.

→ On ne pouvait pas se permettre de supprimer les lignes avec des NaN.

On merge donc les 2 set X_train.csv et X_test.csv pour le preprocessing

| | |
|---|---|
| id_annonce | 0.000000 |
| has_air_conditioning | 0.000000 |
| has_a_garage | 0.000000 |
| has_a_cellar | 0.000000 |
| nb_terraces | 0.000000 |
| has_a_balcony | 0.000000 |
| nb_photos | 0.000000 |
| nb_boxes | 0.000000 |
| nb_parking_places | 0.000000 |
| last_floor | 0.000000 |
| upper_floors | 0.000000 |
| price | 0.000000 |
| property_type | 0.000000 |
| approximate_latitude | 0.000000 |
| approximate_longitude | 0.000000 |
| postal_code | 0.000000 |
| city | 0.000000 |
| nb_rooms | 0.355940 |
| size | 0.648319 |
| nb_bedrooms | 7.131507 |
| nb_bathrooms | 37.310113 |
| floor | 38.994470 |
| energy_performance_category | 50.746838 |
| energy_performance_value | 50.746838 |
| ghg_value | 51.840081 |
| ghg_category | 51.840081 |
| exposition | 71.766351 |
| land_size | 95.893981 |

# Valeurs manquantes :

- Colonne ***floor*** :

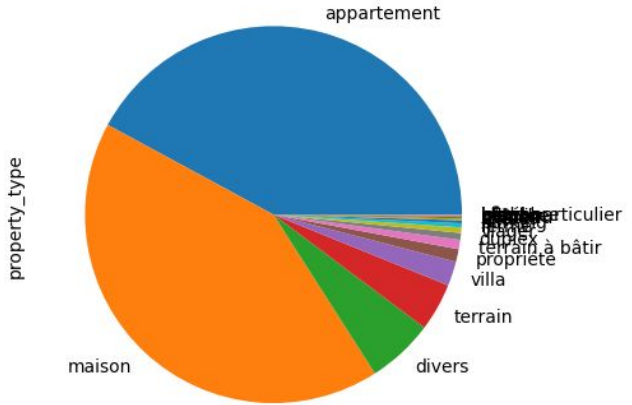
→ On fixe l'étage à 0 pour les biens autres que des appartements

- Colonne ***land_size*** :

→ On fixe le *land_size* de certains biens à la valeur 0.

- Colonne ***nb_bathrooms*** :

→ On choisit, le nombre de salles de bains à 1 ou 0. (et 5 pour les hôtels)



# Encodage des valeurs catégorielles :

→ On encode les valeurs de type de propriété via la méthode ***one-hot-encoding***.

| id | color |
|---|---|
| 1 | red |
| 2 | blue |
| 3 | green |
| 4 | blue |

**One Hot Encoding**

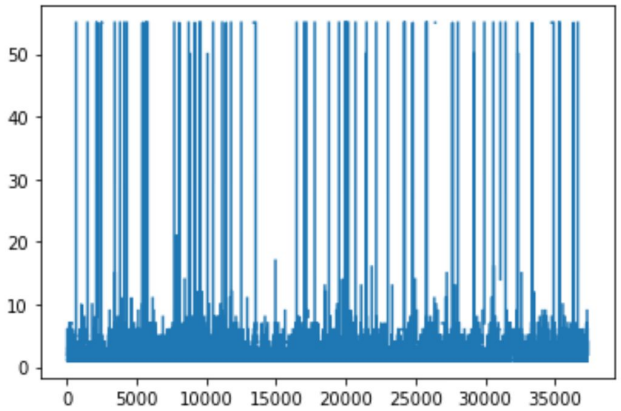
| id | color_red | color_blue | color_green |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 |


→ On encode les **villes par fréquence**

# Dernières valeurs manquantes :

L’idée est de remplir les données manquantes par des valeurs que l’on impose:
→ Utilisation de la **moyenne**, cependant pas adaptée à toutes les données, exemple :

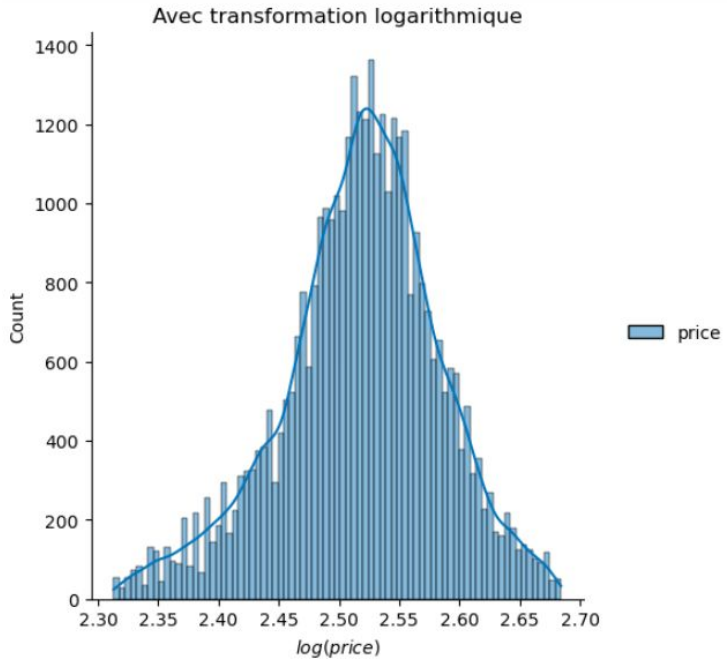
données pour floor :




→ Solution : Utilisation d’un **modèle de ML** pour prédire les dernières valeurs manquantes. Utilisation de **KNN-Imputer**

# Dernières modifications :

→ On applique le log à la colonne des prix :




→ On normalise les données, on prend la fonction *RobustScaler* :
Il s'agit d'une normalisation de type z-score :

$$z = \frac{x - \mu}{\sigma}$$

$\mu$ = Mean
$\sigma$ = Standard Deviation

# Choix du modèle :

- On procède d'abord à un **train_test_split**, avec 20 % des données dans le validation set
- Notre métrique est la *mean_absolute_percentage_error (x100)*

$$\frac{prix_{predit} - prix_{reel}}{prix_{reel}} * 100$$

On a un problème de régression :

→ On entraîne donc plusieurs modèles de régression et on les teste. (Sans hyperparamétrage)

# Choix du modèle :

Résultats (pourcentage d’erreur absolue moyen) :

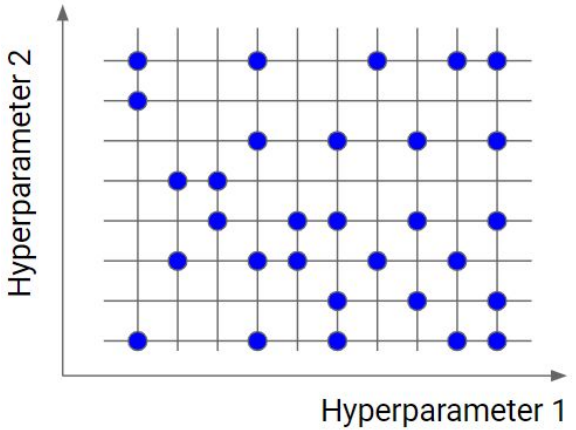
- Error for LR = 118.5 %
- Error for Ridge = 119.8 %
- Error for Lasso = 79.0 %
- Error for ElasticNet = 79.0 %
- Error for CART = 47.2 %
- **Error for RF = 31.6 %**
- Error for ADA = 51.2 %
- Error for GBM = 40.8 %
- **Error for XGBoost = 30.7 %**
- Error for Deep Learning > 60 %

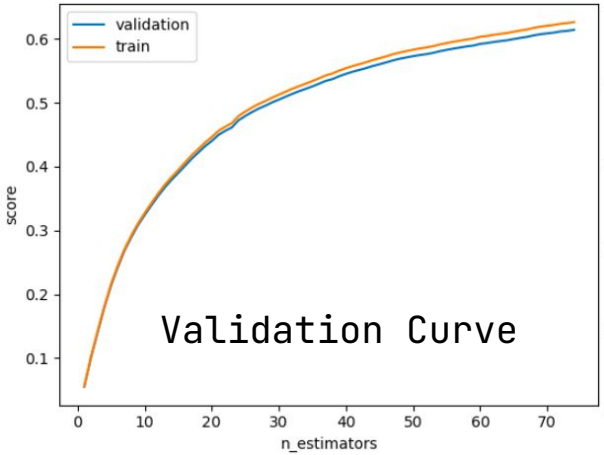
# Hyper-paramétrage :

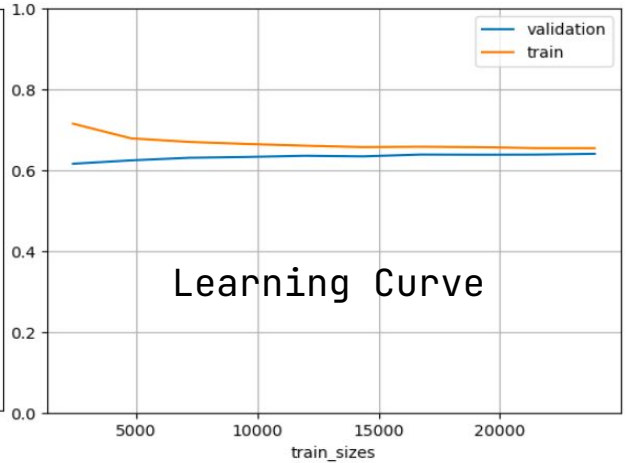
model = **XGBoost**(hyperparamètres)

***GridSearchCV***

- Overfitting : Cross-Validation Test
- Convergence : itératif.




Hyper-paramètres :
- learning_rate
- max_depth
- lambda
- …

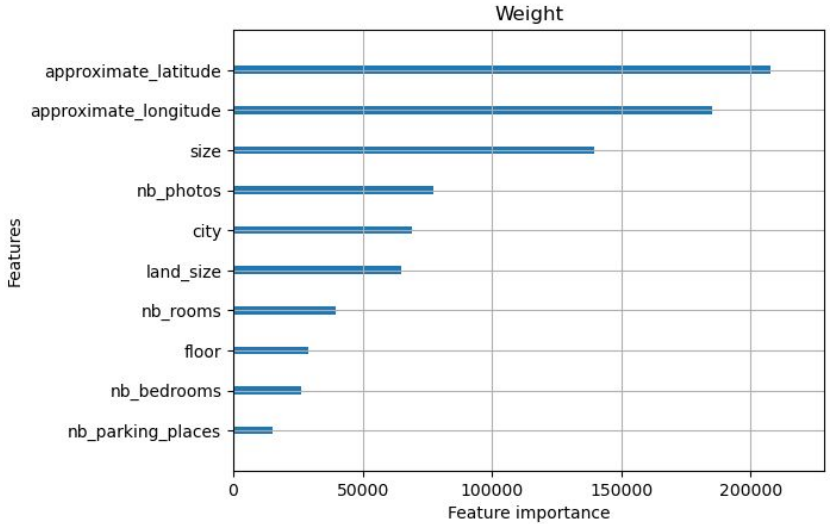
# Résultat au challenge

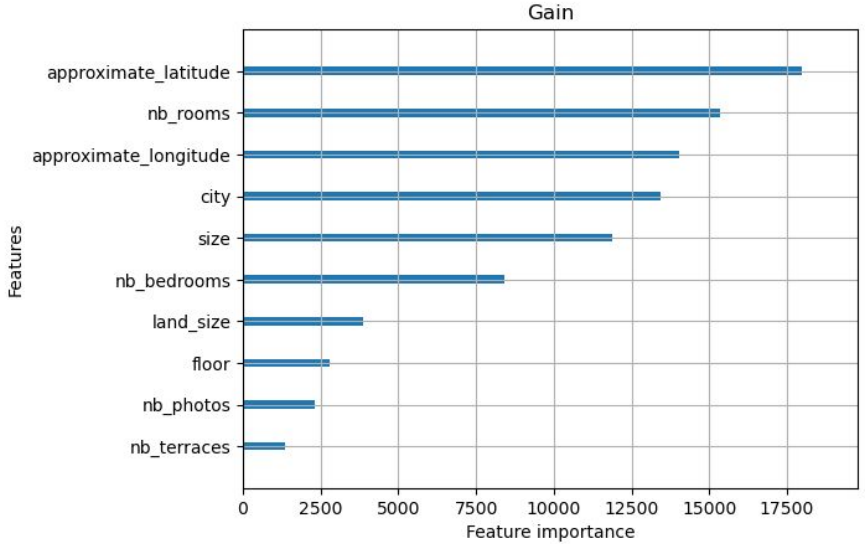
| Rang | Date | Participant(s) | Score public |
|---|---|---|---|
| 1 | 27 février 2022 12:26 | pednt | 21,0161 |
| 2 | 22 janvier 2023 13:34 | Clem1 & FélixD | 22,0982 |
| 3 | 15 décembre 2022 20:09 | ArnaudMARECHAL | 22,4500 |
| 4 | 6 décembre 2022 16:48 | ulrich777 | 23,0190 |
| 5 | 3 février 2023 18:51 | anasstheone123 & Abdellah.Laassairi | 23,1294 |
| 6 | 13 février 2022 20:25 | aho | 23,6855 |
| 7 | 5 février 2023 17:24 | zheng_zixuan & xlsf | 24,5563 |
| **8** | **28 mars 2023 12:32** | **valentinlhote** Groupe E | **24,8739** |
| 9 | 9 mars 2023 09:16 | VictorHoffmann | 25,0579 |
| 10 | 25 mars 2023 13:54 | msoric & OmarMousteau | 25,4371 |

8/170

R2= 0.803

# Résultat: Feature importance




⇒ Importance de la ville/zone géographique

# Conclusion

⇒ Résultats satisfaisants : jeu de donné initial assez complet + préprocessing

⇒ Score très sensible au préprocessing.

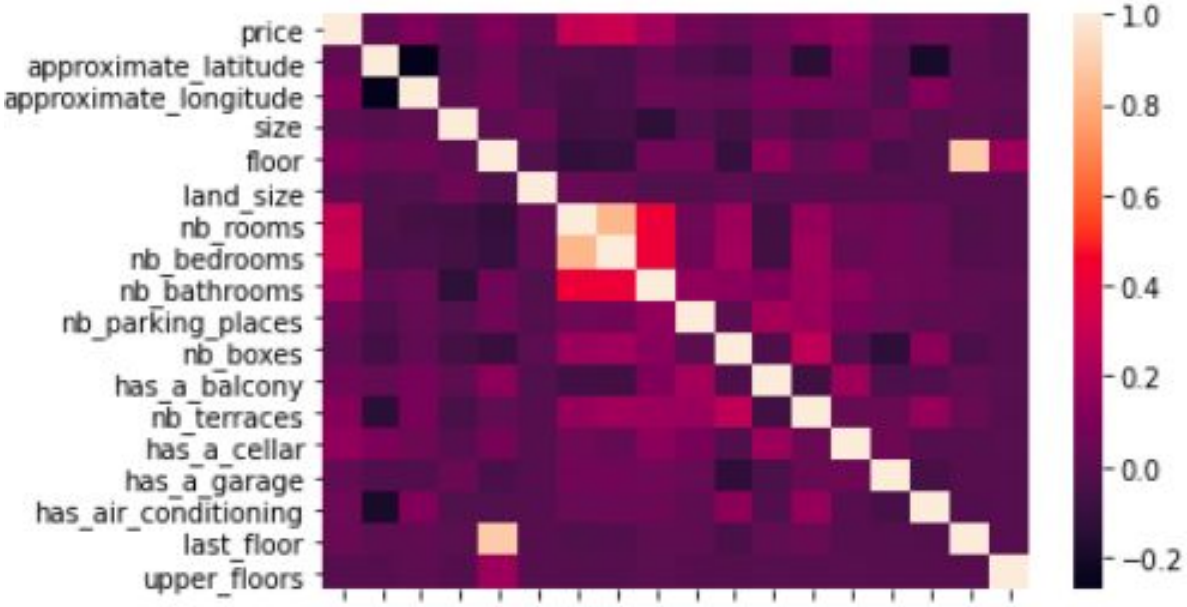
Axes d’amélioration :

⇒ Réduction de dimension en sélectionnant des variables pour exploiter plus de modèles

⇒ Prise en compte des photos (luminosité)

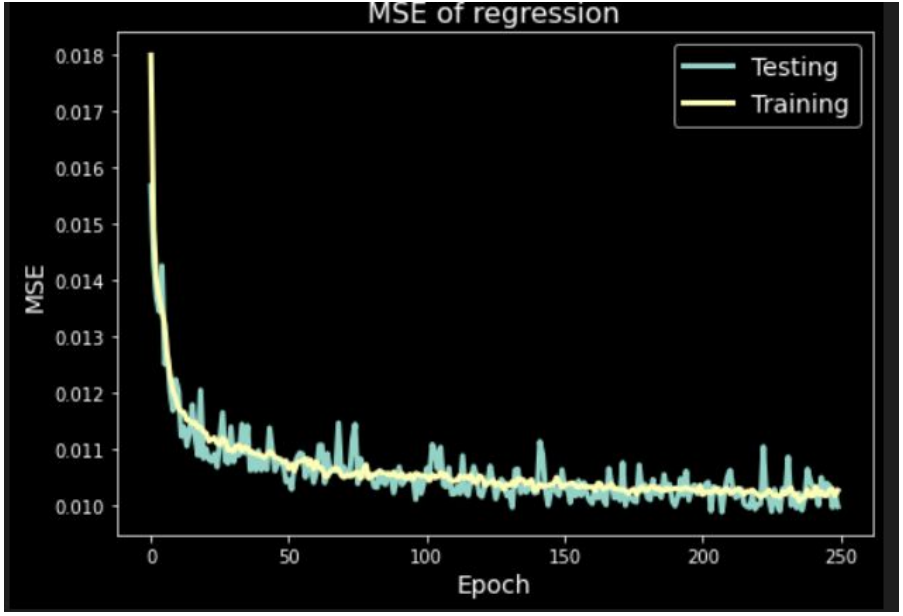
⇒Meilleur prise en compte de la géographie (API Google Maps)

# Annexes



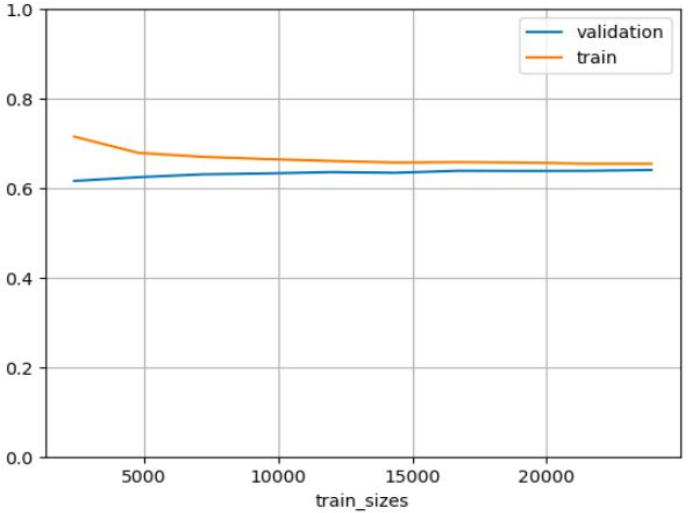

Corrélation entre les variables

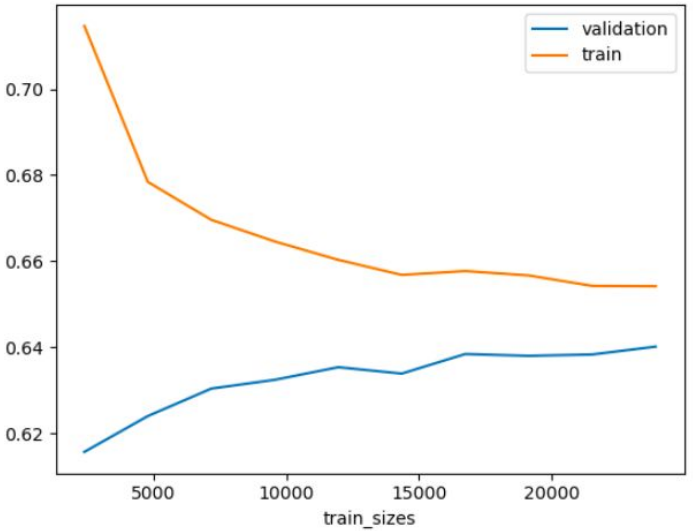
# Annexes




Exemple de convergence d’un système Itératif : Réseaux de Neurones (Deep Learning)

# Annexes




Learning Curve