# Projet 8 : Prédiction des prix des logements (Kaggle)

# Plan du projet :

[Projet 8 : Prédiction des prix des logements (Kaggle) 1](#_Toc72061069)

[Plan du projet : 2](#_Toc72061070)

[I. Le contexte du projet 3](#_Toc72061071)

[II. Les données du projet 3](#_Toc72061072)

[III. Exploration et nettoyage des données : 4](#_Toc72061073)

[1. Exploration : 4](#_Toc72061074)

[2. Nettoyage des données : 6](#_Toc72061075)

[3. Matrice de corrélation : 6](#_Toc72061076)

[4. Préparation des variables : 7](#_Toc72061077)

[IV. Modélisation 8](#_Toc72061078)

[1. Les différents modèles : 8](#_Toc72061079)

[2. Les résultats de la compétition Kaggle et approches de perfectionnement : 9](#_Toc72061080)

[V. Conclusion : 10](#_Toc72061081)

[VI. Références : 10](#_Toc72061082)

## Le contexte du projet

Dans le cadre de ma formation en « Ingénieur Machine Learning », on travaillera sur un projet de prédiction des prix des biens immobiliers à partir des différentes variables à notre disposition.

L’objectif est d’avoir le meilleur score de prédiction et de soumettre ce modèle sur la compétition Kaggle.

Kaggle est un site communautaire basé sur le *data science* Il est possible de créer des compétitions avec récompenses à la clé (de l’argent ou des contrats d’embauche principalement). Le commanditaire de la compétition (souvent une entreprise) soumet un problème, composé d’une description et d’un ou plusieurs jeux de données, sur Kaggle, et le site s’occupe d’organiser une compétition autour de ce problème. Les participants choisissent donc les challenges auxquels ils veulent participer et tentent de résoudre le problème soumis de la meilleure façon possible, tout cela en compétition avec les autres participants. Pour cela, on a la possibilité de soumettre un fichier afin d’obtenir un score qui nous positionne dans un classement publique (qui regroupe tous les participants). Il est donc possible, à tout moment, de savoir qui sont les premiers, et de comparer son propre score à celui des autres.

## Les données **du** projet

Les données à exploiter sont des données mises à disposition sur Kaggle (<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>) , elles sont composées de 3 fichiers : train.csv, test csv, submission.csv

* Train.csv, contient les données d’entrainement : 1460 lignes et 81 colonnes
* Test.csv, contient les données de test : 1459 lignes et 80 colonnes.
* Submission.csv, contient un modèle de fichier à soumettre, résultats de la prédiction.

La variable cible ou la variable à prédire est le « SalePrice ».

Nous avons 79 variables explicatives, composées de : 54 variables catégorielles et 25 variables numériques.

Une fois les données téléchargées et ses contenus bien identifiés, nous allons pouvoir les explorer, nettoyer et apporter des modifications pour qu’elles soient optimales pour nos modélisations.

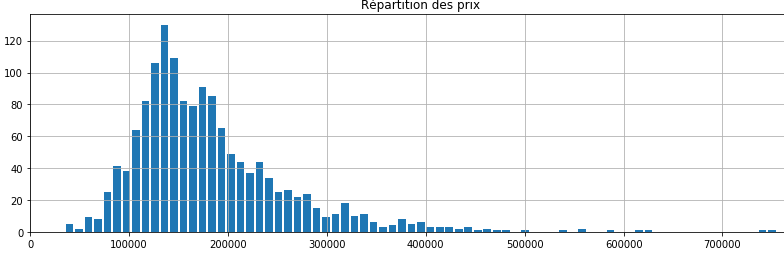
## **Exploration et nettoyage des données :**

Pour travailler sur ces données, il faut les connaître, d’où le travail d’exploration qui va suivre.

### Exploration :

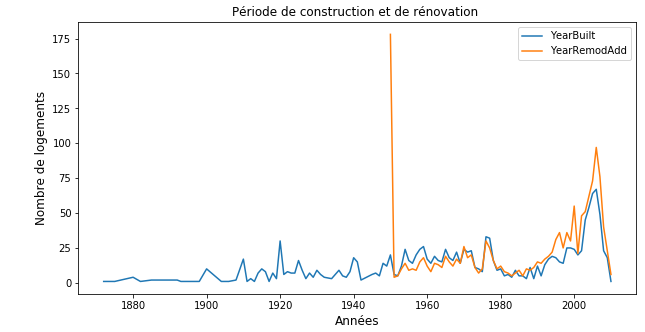
Pour avoir une première vision sur l’ensemble de nos données, nous allons étudier le prix en lui-même mais aussi des variables temporelles telles que les dates de construction, les dates de rénovation et la période de vente.

* La répartition des prix



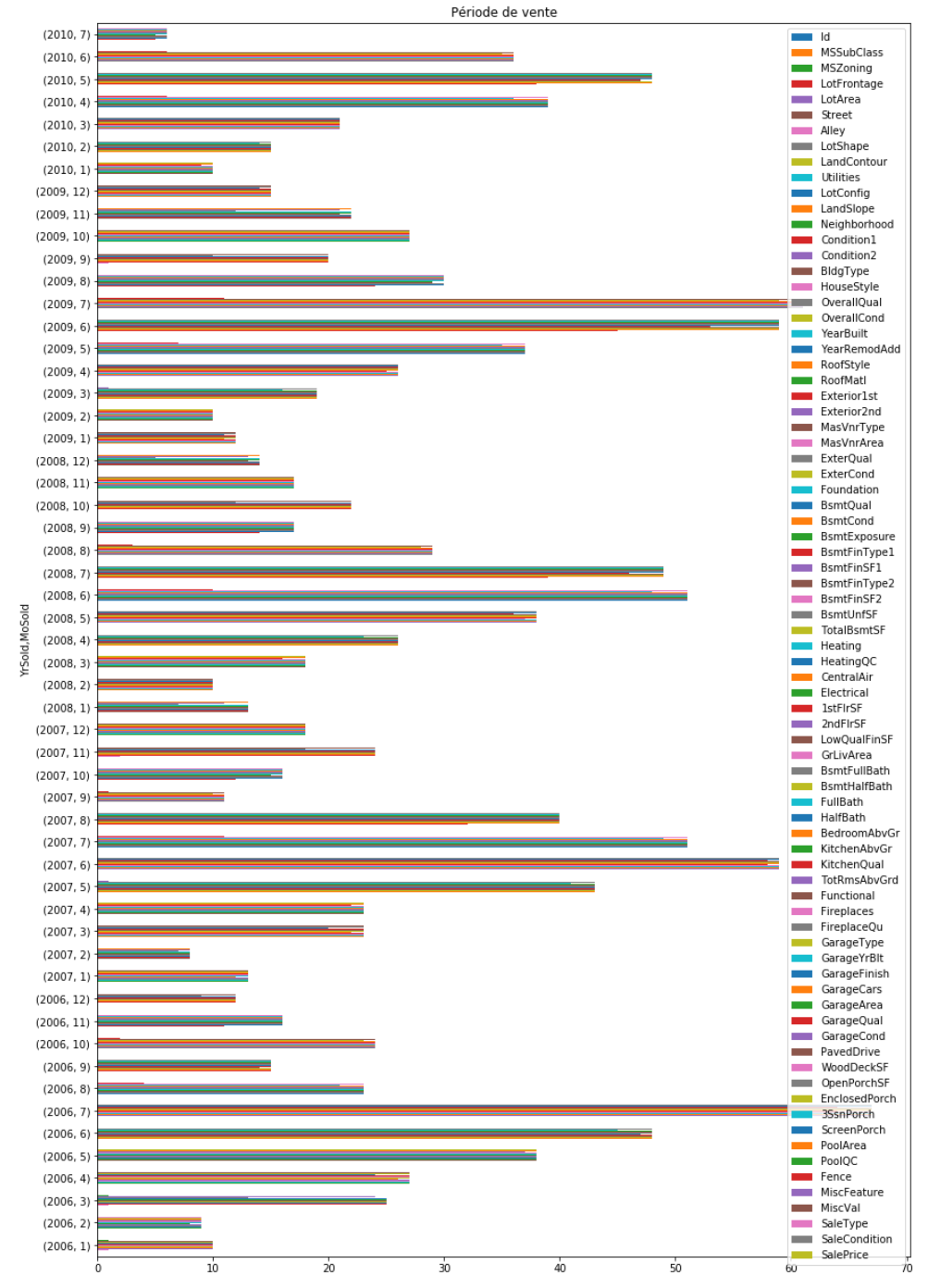
On peut dire que les prix moyens de logement se concentrent entre 100 000 et 250 000

* Les périodes importantes



On constate que la quantité de construction est assez stable dans le temps jusqu’au début des années 90 ou nous avons commencé à construire en quantité, c’est peut-être aussi lié au début des constructions des immeubles.

Les données de rénovation, commence en 1950 et la tendance de rénovation suit la tendance de construction donc on peut dire que quand il y une forte demande de construction, l’activité de rénovation aussi suit la tendance.



Les périodes de vente aussi sont très concentrées autour du mois de Mai, Juin, Juillet

### Nettoyage des données :

Le nettoyage des données est une partie importante du travail d’un *data scientist* avant les modélisations car les résultats en dépendront.

Voici quelques préparations effectuées sur les données :

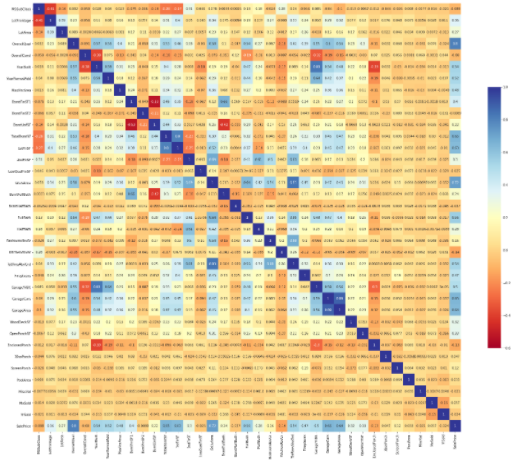
* Identification et suppression des valeurs aberrantes entre le prix et la surface habitable
* *Identification des variables manquantes*
* *Imputation des valeurs manquantes si possible,* à partir des données existantes et la connaissance métier
* *Suppression des variables à très faible pouvoir explicative,* à moins 3 cardinalités

Après nettoyage, nous sommes passés de :

2919 lignes et 80 colonnes à 2042 lignes pour 75 colonnes

### Matrice de corrélation :

Nous utilisons la matrice de corrélation pour identifier les corrélation des variables sur la variable cible :



On voit sur cette graphe que certaines variables ont besoin d’être mis en évidence car elles ont un impact significatif sur le prix comme :

* L’âge du logement
* Rénovation faite ou non sur un logement
* Surface totale habitable d’un logement
* Ratio entre la surface habitable et la surface de la propriété
* Ratio entre le nombre de Salles de bains et le nombre de chambres

Donc nous avons créé ces nouvelles variables pour optimiser l’exploitation de nos données.

### Préparation des variables :

Transformations des variables numériques :

Nous avons constaté que la distribution des prix est très asymétrique, donc nous utiliserons la valeur transformée en logarithme pour corriger ce biais

|  |  |
| --- | --- |
|  |  |

La méthode exponentielle nous permettra d’inverser cette valeur à la fin pour revenir aux valeurs réelles.

De même, les variables explicatives sont transformées si les biais sont supérieurs à 0.5, l’objectif est d’obtenir une distribution normale pour chaque variable après transformation et une variance constante

Traitement des variables catégorielles :

Les variables catégorielles ne peuvent pas être traitées directement par nos algorithmes, donc elles ont besoin d’être encodées.

Mais afin de limiter la création des nouvelles variables en utilisant «one-hot-encoder  » sur toutes les variables catégorielles, nous avons identifié celles qui ont des valeurs ordonnées et ont une échelle d’appréciation comme ‘la qualité de finition de cuisine’ par exemple (Ex,Gd,Ta,Fa,Po,Absent) et changer ces valeurs par (5,4,3,2,1,0). Sur les autres variables numériques nous avons appliqué le «one-hot-encoder  »

A la fin de cette opération, nous avons 254 features, qui nous paraissent trop important, donc nous allons identifier les variables les plus importantes pour réduire les dimensions de notre modèle.

Réduction dimensionnelle selon l’importance des variables :

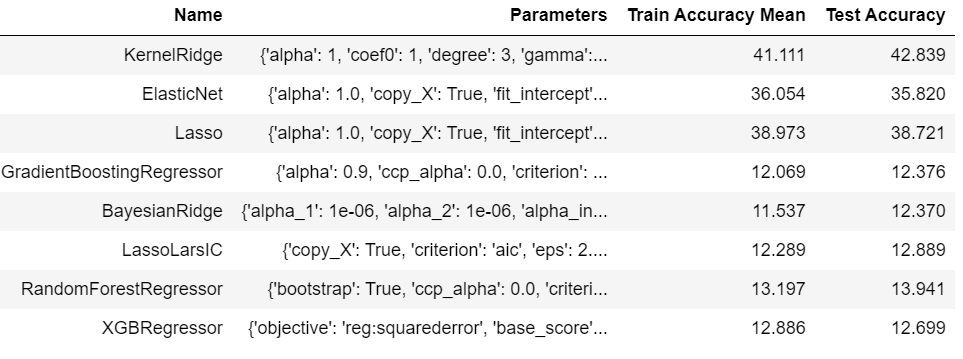
Nous identifions les importances des variables grâce à **sklearn.feature\_selection.**SelectFromModel et en utilisant modèle XGBoost pour limiter à 43 variables, cette réduction de dimension est utile pour réduire le cout des données mais aussi le temps d’exécution d’entrainement.

## Modélisation

### Les différents modèles :

* **Modèle de base sans optimisation :**

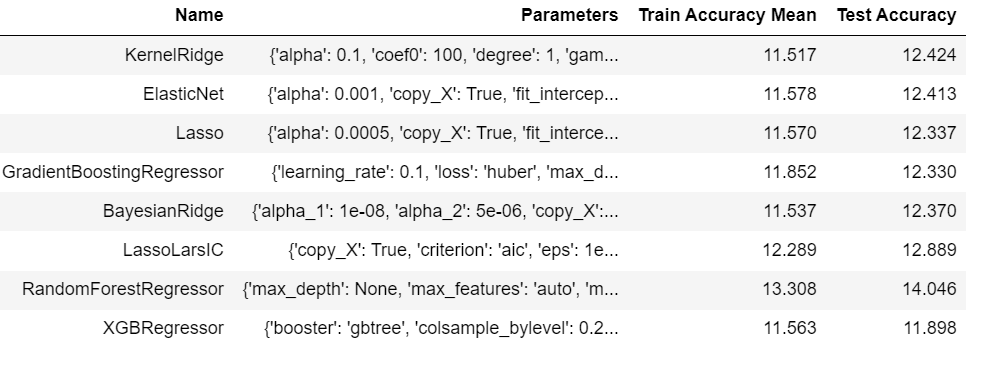
Nous avons testé plusieurs modèle pour savoir le quelle sera le plus adapté à notre problématique et voici les résultats :



On voit ici que selon le modèle choisi les résultats sont très variables, d’où la nécessité de tester sur plusieurs modèles. Mais on constate qu’avec les paramètres par défaut, le modèle *BaysienRidge* est le plus performant des 7 modèles avec le RMSE(test) à 12.370.

* **Optimisation avec GridSerachCV :**

Puis, nous avons utilisé la méthode GridSearchCV pour trouver les valeurs des paramètres adéquats et pour avoir le meilleur score possible pour chaque modèle :



Les scores de chaque modèle se sont améliorés pour se rapprocher du score de *BayesianRidge* et *XGBRegressor* est même devenu meilleur un RMSE (test) à 11.898

* **Combinaison des modèles :**

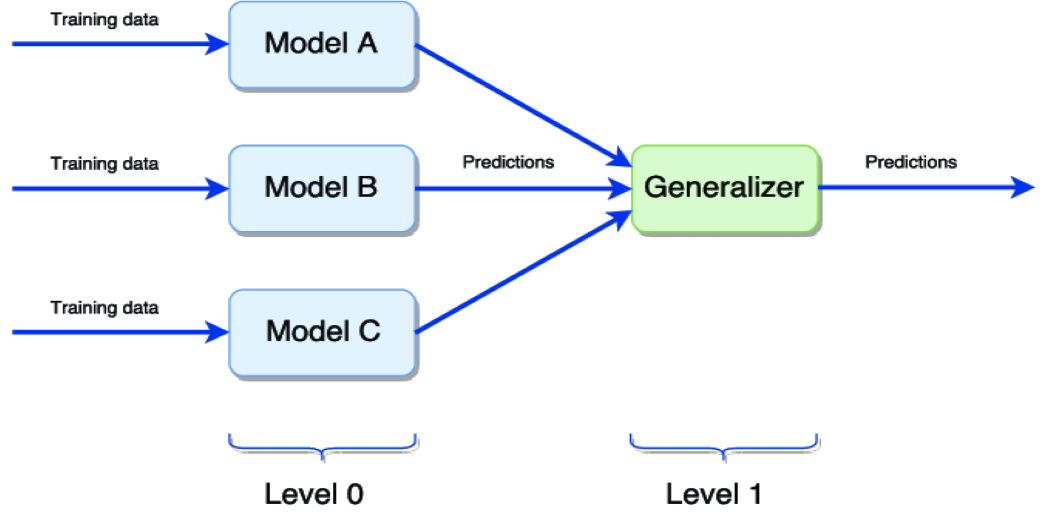
Pour avoir un modèle final plus stable avec très peu de variance, il existe plusieurs façons de combiner les modèles comme :

* La combinaison par vote simple : VotingRegressor()
* La combinaison par vote pondérée : VotingRegressor() en précisant les poids de chaque modèle
* Le Stacking : StackingRegressor(), traitement sur 2 niveaux de modélisation :

Mais nous allons nous concentrer sur ce dernier dans notre projet

**Fonctionnement de StackingRegressor :**

Ce modèle s’appelle aussi généralisation empilée, c’est une technique qui combine plusieurs modèles de régression via un méta-régresseur, le schéma ci-dessous représente son architecture :



Le schéma représente un modèle avec un empilement des 3 modèles de base.

Au **Niveau 0,** on trouvera les 3 estimateurs (ou modèles) de base qui feront des prédictions à partir des données d’entrainements en entrée. Il est important d’avoir un niveau 0 composé des différents types d’algorithme d’apprentissage (linéaire, arbre, …) pour assurer l’hétérogénéité du modèle final

Puis au **Niveau 1**, on trouvera un modèle final, appelé méta-modèle, qui généralise les prédictions faites au niveau 0 pour obtenir la sortie finale. Le méta-modèle utilise les valeurs de prédiction de chaque modèle de bases pour entrainer son modèle.

**Pourquoi la généralisation ?**

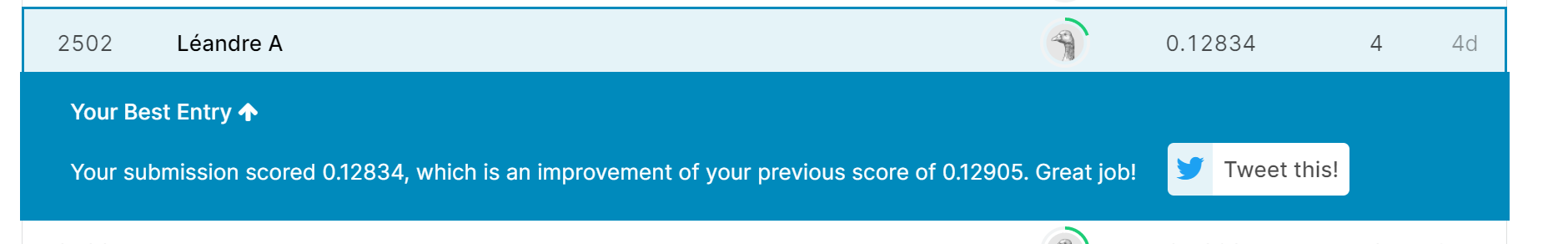
La généralisation nous assure une constance et un maximum de précision quel que soit les données et le problème à résoudre, tandis que la performance des algorithmes de base (niveau 0) fonctionne mieux dans un problème et moins bien dans un autre problème.

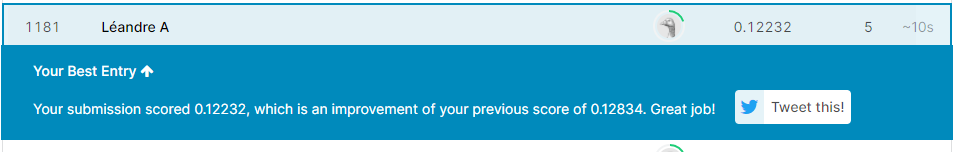
Et voici le résultat de StackingRegressor sur notre projet :

|  |  |
| --- | --- |
| Train | Test |
|  |  |

### Les résultats de la compétition Kaggle et approches de perfectionnement :

En m'inspirant des autres solutions sur Kaggle, nous avons pu successivement améliorer le score en passant de 2502e à 1181e au classement en appliquant les modifications suivantes :





Sur les variables catégorielles :

- Ne pas appliquer one-hot-encoding systématiquement sur toutes les variables catégorielle, mais traiter chaque variable au cas par cas, et se basant sur les explications des variables dans l’énoncé de Kaggle, exemple :

1) Les variables catégorielles ordonnées peuvent être remplacé :

"Fa":1, "TA":2, "Gd":3, "Ex":4 pour ne pas à générer trop des variables supplémentaires

Sur la modélisation,

* Utiliser RMSE pour la métrique sur les différents tests en local car c’est la métrique utilisée par Kaggle pour cette compétition.
* Ne pas garder toutes les variables en entrée du modèle mais  que top 50  des variables importantes pour la modélisation car un nombre réduit des variable sera toujours moins onéreux pour le modèle.
* Appliquer le stacking-modèle qui permet d’avoir un score plus stable malgré les différences des données en entrée du modèle

## Conclusion :

Pour conclure l’étude sur ce projet, nous avons vu

Nous avons pu voir dans ce notebook, les différentes étapes de préparation des données, en passant par le remplissage des valeurs manquantes, suppression des variables non renseignées et non-impactantes pour réduire les dimensions et création des variables qui nous semblent importantes. Une première modélisation, puis l'optimisation des valeurs des paramètres et à la fin la technique d'empilement des modèles.

Des améliorations peuvent-être apportées sur la partie "features ingeenering" avec une bonne connaissance de chaque variable et beaucoup de tests.

## Références :

**https://www.kaggle.com/chasseur/housing-price-by-chasseur/comments?select=test.csv**

[**https://www.kaggle.com/alaasedeeq/house-price-prediction-top-8**](https://www.kaggle.com/alaasedeeq/house-price-prediction-top-8)

https://datascientest.com/algorithmes-de-boosting-adaboost-gradient-boosting-xgboost