



UFR 6

Université Paul Valéry, Montpellier III

Projet Master 1 Mathématiques et Informatique Appliquées aux Sciences Humaines et Sociales

---

# Analyse du challenge Kaggle - Prédiction des prix immobiliers

Omia Laasili

Janvier 2025

---

Encadrant universitaire : Lèbre Sophie

Année universitaire 2025-2026

# Résumé

Ce travail utilise le challenge Kaggle House Prices sur des données immobilières d'Ames (Iowa). L'objectif est de prédire le prix de vente des logements à partir de leur caractéristiques : c'est un problème de régression. On dispose de train.csv (avec prix) et test.csv (sans prix) pour faire une soumission au format Kaggle. Le projet met en pratique des méthodes de data science vues en cours, dans un contexte proche du monde professionnel de l'immobilier.

# Table des matières

# Table des figures

# Introduction

Ce travail s'appuie sur un projet réalisé à partir d'un challenge proposé sur la plateforme Kaggle, intitulé **House Prices – Advanced Regression Techniques**. Le challenge repose sur des données immobilières provenant de la ville d'Ames, dans l'Iowa aux États-Unis. Le jeu de données regroupe de nombreuses informations décrivant des logements, comme leur surface, leur année de construction, le nombre de pièces, mais aussi des critères liés à la qualité des matériaux ou à l'environnement du bien.
L'objectif principal de ce travail est de prédire le prix de vente d'un logement, noté SalePrice, à partir des variables explicatives disponibles. Il s'agit d'un problème de régression, puisque la variable à expliquer est quantitative. Il y a trois fichiers principaux : train.csv, qui contient 1460 maisons avec leurs caractéristiques et leur prix de vente, tesċsv, qui contient 1459 maisons pour lesquelles le prix doit être prédit, et sample_submission.csv, qui donne un exemple du format attendu pour la soumission. Dans le cadre du master MIASHS, ce projet est en adéquation car il mobilise des compétences en statistiques, en modélisation et en analyse de données, tout en s'appuyant sur un cas concret. Il correspond également à ce qui pourrait être demandé dans une mission de data analyst en entreprise, où l'objectif est de construire un modèle prédictif et de comprendre ses résultats pour prendre de meilleures décisions.
En effet, aujourd'hui, la data science occupe une place importante dans le secteur de l'immobilier. De nombreuses agences immobilières, plateformes en ligne ou entreprises spécialisées s'appuient sur des data analysts ou des data scientists pour analyser les données du marché, mieux comprendre les tendances et améliorer leurs décisions. Il existe même des entreprises spécialisées dans l'exploitation des données immobilières, comme Casafari, qui propose une grande base de données immobilières. Ces solutions sont principalement destinées aux professionnels de l'immobilier, tels que les agences, afin de leur fournir des informations utiles pour l'analyse du marché, la prise de décision, ainsi que la réduction des erreurs et des risques. Cette entreprise basée au Portugal illustre l'importance de la data science dans le domaine immobilier et montre qu'il pourrait très bien correspondre à une mission confiée à un data scientist.

# Sommaire

## 0.1 Présentation des données

Les données utilisées dans ce projet sont fournies sous la forme de trois fichiers principaux.
Le jeu d'entraînement train.csv, contenant 1460 individus correspondant à des maisons, avec 81 colonnes au total : 79 variables explicatives, une variable identifiant (Id) et la variable cible, le prix de vente du bien (SalePrice).
Le fichier test.csv contient 1459 individus avec les mêmes variables mais sans le prix de vente, qui doit être prédit.
Le fichier sample_submission.csv fournit un exemple du format attendu pour la soumission des résultats sur la plateforme Kaggle.
Les variables explicatives décrivent différents aspects des logements. Les fichiers contiennent des informations sur le terrain (surface, type de rue, forme de la parcelle, etc.), le bien immobilier (style, superficie habitable, nombre de pièces, année de construction, etc.), les pièces non habitables (garage, piscine, clôture, etc.), le quartier (localisation dans la ville) ainsi que d'autres caractéristiques. Certaines variables sont quantitatives (continues ou discrètes) et d'autres sont qualitatives (catégorielles nominales ou ordinales). On note que de nombreuses variables comportent des valeurs manquantes (NA), souvent signifiant l'absence d'une caractéristique (par exemple, Alley = NA signifie "pas d'allée").
Le fichier de description des données fourni avec le challenge (data_description.txt) précise le sens de chaque variable et les modalités. Ce fichier a été utilisé comme référence tout au long du projet afin de mieux comprendre les données et traiter correctement les variables. .

## 0.2 Problématique

Le jeu de données utilisé dans ce projet contient beaucoup de variables décrivant les logements, alors que le nombre de maisons observées reste relativement limité. Si l'on utilise trop de variables dans un modèle de régression, celui-ci peut très bien expliquer les données d'entraînement, mais se révéler peu fiable lorsqu'on l'applique à de nouvelles données. Il est donc important de ne pas construire un modèle trop complexe.
L'objectif de ce travail est donc de trouver un modèle qui permette de prédire correctement le prix de vente des logements, tout en restant suffisamment simple et stable.

## 0.3 Nettoyage et préparation des données

### 0.3.1 Gestion des valeurs manquantes

Je me suis d'abord occupé des nombreuses valeurs manquantes dans les fichiers CSV, j'ai créé le notebook python "nettoyage_preparation.ipynb" pour cela. Dans un premier

temps, le nombre de valeurs manquantes par variable a été identifié afin de repérer les colonnes concernées.
Certaines valeurs manquantes correspondent à l'absence réelle d'une caractéristique du logement. Par exemple, la variable MasVnrType, qui donne le type de revêtement extérieur, prend la valeur manquante lorsque la maison ne possède aucun revêtement. Dans ce cas, j'ai remplacé les valeurs manquantes par "None".
De même, pour certaines variables numériques liées aux surfaces ou aux équipements, une valeur manquante signifie que l'élément n'existe pas dans le logement. Par exemple, lorsqu'une maison ne possède pas de garage ou de sous-sol aménagé, la surface associée est manquante. J'ai remplacé ces valeurs manquantes par 0.
Après ce premier traitement, seules 8 colonnes présentaient encore des valeurs manquantes ("Exterior1st", "Exterior2nd", "SaleType", "Functional", "Utilities", "MSZoning", etc.). Mais pour ces variables, il manquait seulement une ou deux valeurs maximum, on va les remplacer par la valeur observée la plus fréquente de la colonne (le mode). Par exemple, la variable "KitchenQual" donne la qualité de la cuisine (soit Ex, Gd, TA, Fa ou Po). C'est une variable qualitative ordinale qui possède une seule valeur manquante, pour éviter de supprimer un individu, on remplace.
Enfin, la variable "LotFrontage" (Linear feet of street connected to property), qui représente la longueur de façade donnant sur la rue, contenait 259 données manquantes pour le train et 227 pour le test. Il s'agit d'une variable quantitative continue. J'ai remplacé les valeurs manquantes par la médiane de la colonne.
Pour finir, je me suis intéressée à la variable SalePrice, qui correspond au prix de vente du logement. Dans le cadre du challenge Kaggle, les soumissions sont évaluées à l'aide de l'erreur quadratique moyenne (RMSE) calculée sur le logarithme du prix de vente, c'est-à-dire l'erreur entre le logarithme du prix prédit et celui du prix observé. J'ai ajouté la nouvelle colonne "SalePrice_log" sur le train avec la valeur du log du prix de vente. Notre nouvelle variable cible sera donc la variable "SalePrice_log".

### 0.3.2 Préparation des données pour la modélisation

Après le nettoyage des données, je me suis intéressée à la préparation des variables pour la modélisation. Pour cela, j'ai créé un nouveau notebook Python intitulé modelisation.ipynb.
Dans un premier temps, j'ai séparé les variables explicatives de la variable cible. Les variables explicatives correspondent à l'ensemble des caractéristiques des logements, tandis que la variable cible est le logarithme du prix de vente (SalePrice_log). J'ai supprimé la colonne identifiant (Id) car elle n'est pas utile ici.
Ensuite, j'ai séparé les variables quantitatives des variables qualitatives. Cette étape est nécessaire car ces deux types de variables ne sont pas traités de la même manière dans les modèles de régression.
J'ai encodé les variables qualitatives à l'aide du one-hot encoding, et standardisé les va-

riables quantitatives pour pouvoir les utiliser dans les modèles de régression. Enfin, j'ai d'abord construit ce prétraitement à partir des données d'entraînement, puis je l'ai appliqué aux données de test.

## 0.4 Modélisation

### 0.4.1 Exploration des corrélations et colinéarité

#### Heatmap des corrélations

Après la préparation des données, j'ai analysé les corrélations entre les variables quantitatives à partir du fichier d'entraînement nettoyé, dans le notebook modelisation.ipynb. Pour cela, j'ai calculé la matrice de corrélation et représenté les résultats sous forme de heatmap.

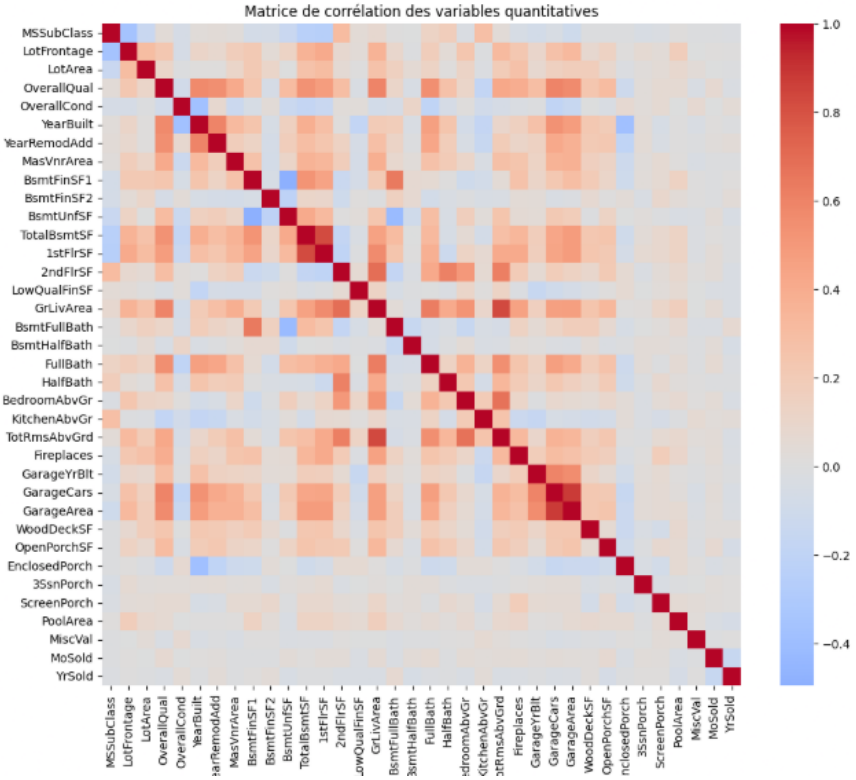


FIGURE 1 – Matrice de corrélation des variables quantitatives

Le graphique met en évidence plusieurs corrélations positives entre certaines variables, en particulier celles liées aux surfaces des logements. Par exemple, les variables GrLivArea (surface habitable au-dessus du sol) , 1stFlrSF (surface du rez-de-chaussée) et TotalBsmtSF (surface totale du sous-sol) possèdent des corrélations élevées entre elles, ce qui est cohérent puisqu'elles décrivent des caractéristiques proches du logement. De la même manière, les variables GarageArea (aire du garage ) et GarageCars (nombre de places de garage) , ou encore YearBuilt (année de construction) et OverallQual (qualité

globale du logement) sont très corrélées, ce qui indique peut-être que les logements plus récents sont de meilleure qualité.

#### Identification des corrélations fortes

Dans un second temps, j'ai extrait, à partir de cette matrice, un tableau récapitulant les paires de variables présentant des corrélations élevées, en choisissant un seuil de 0,7. Les résultats montrent notamment une forte corrélation entre GarageCars et GarageArea, ainsi qu'entre GrLivArea et TotRmsAbvGrd (nombre total de pièces avec le sous-sol exclus). On observe également une corrélation importante entre TotalBsmtSF et 1stFlrSF (surface du rez-de-chaussée). Ces relations concernent principalement les variables décrivant des surfaces ou des caractéristiques proches, ce qui confirme la présence de colinéarité entre certaines variables.

| | | 0 |
|---|---|---|
| **GarageCars** | **GarageArea** | 0.882475 |
| **GarageArea** | **GarageCars** | 0.882475 |
| **GrLivArea** | **TotRmsAbvGrd** | 0.825489 |
| **TotRmsAbvGrd** | **GrLivArea** | 0.825489 |
| **TotalBsmtSF** | **1stFlrSF** | 0.819530 |
| **1stFlrSF** | **TotalBsmtSF** | 0.819530 |

FIGURE 2 – Liste des variables avec corrélation supérieure a 0.80

#### Interprétation

Ces corrélations montrent la présence de colinéarité entre plusieurs des variables explicatives. En régression linéaire, cela peut rendre l'estimation des coefficients instable et fausser le modèle. Cette analyse nous indique qu'il faudra utiliser des méthodes de régression pénalisées pour la suite.

### 0.4.2 Modèles de régression

#### Régression linéaire classique

Pour commencer, j'ai estimé une régression linéaire classique à l'aide de la fonction **LinearRegression** de la bibliothèque scikit-learn, afin d'obtenir une base. La soumission associée à ce modèle donne un score de 0,15573 sur Kaggle. Ce résultat signifie que, en moyenne, les prédictions s'écartent du prix réel d'environ 15 % Mais on peut encore améliorer ce score en utilisant des modèles pénalisés.

**Régression linéaires avec pénalisation : Ridge, Lasso et Elastic Net**

Pour améliorer les performances de la régression linéaire classique, j'ai estimé des modèles de régression pénalisée. Les modèles Ridge, Lasso et Elastic Net ont été ajustés à l'aide de la bibliothèque scikit-learn, en utilisant les classes RidgeCV, LassoCV et ElasticNetCV. Les paramètres de régularisation ont été sélectionnés par validation croisée à 5 folds sur les données d'entraînement.
Lors de la soumission, la régression linéaire classique obtient un score de 0,15573. Les modèles pénalisés donnent de meilleurs résultats, avec un score de 0,13241 pour Ridge, 0,13294 pour Lasso et 0,13285 pour Elastic Net. Les différences entre les modèles pénalisés sont assez faibles, mais ils permettent tous d'améliorer le score de la régression linéaire classique.

**Modèle non linéaire : Random Forest**

J'ai aussi testé un modèle de Random Forest avec la bibliothèque scikit-learn, en utilisant la classe RandomForestRegressor. J'ai entraîné le modèle avec 300 arbres et en gardant les autres paramètres par défaut. J'ai aussi évalué les performances par validation croisée, qui donne un RMSE d'environ 0,1426. La soumission donne un score de 0,14301, contre environ 0,132 pour les modèles pénalisés.

**Méthodes de boosting : Gradient Boosting et XGBoost**

J'ai ensuite testé des méthodes de boosting afin d'évaluer leurs performances. Deux modèles ont été estimés : un modèle de Gradient Boosting et un modèle XGBoost. La soumission réalisée avec le modèle de Gradient Boosting obtient un score de 0,13235, ce qui est très proche des scores obtenus avec les modèles de régression pénalisé. Pour XGBoost, la soumission donne un score de 0,12917, ce qui correspond au meilleur score obtenu. Ce résultat montre que le modèle XGBoost est celui qui offre la meilleure prédiction parmi l'ensemble des modèles testés.

## 0.5 Régression linéaire sur R

Dans cette partie, je me concentre uniquement sur la régression linéaire. Pour cela, j'utilise le fichier train_nettoye.csv, sans encodage one-hot car R gère directement les variables qualitatives sous forme de facteurs.

### 0.5.1 Modèle complet

Après avoir gardé 20 % des données du train pour la validation, j'ai estimé une régression linéaire multiple sur log(SalePrice) avec toutes les variables explicatives du train. Le modèle obtient un $R^2$ ajusté très élevé (0,9281), ce qui montre qu'il explique très bien les

données. Mais de nombreux coefficients ont des p-values élevées, comme FireplaceQuFa qui a une p-value d'environ 0.7928, ce qui montre que le modèle est trop complexe (79 régresseurs).

### 0.5.2 Sélection de variables

#### AIC

J'ai ensuite appliqué une sélection backward avec commes critère l'AIC (par défaut) car on a beaucoup de variables. Le meilleur modèle correspondant contient 51 variables mais , conserve un $R^2$ ajusté légèrement plus élevé (0.9287) que le modèle complet, ce qui montre qu'il explique toujours très bien les données du train avec un modèle plus simple. Après la sélection du meilleur modèle par AIC, j'évalue ses performances sur l'échantillon de test en prédisant toujours la variable log(SalePrice). Mais certains individus du jeu de test me posaient problème car ils possèdent des modalités de variables qualitatives qui n'étaient pas présentes dans le jeu d'entraînement, alors que ces variables ont été gardées pour le modèle.
Donc pour pouvoir faire la prédiction, j'ai préféré supprimer ces individus, qui représente un faible partie. J'ai calculé les SSE et SST pour obtenir le $R^2$ sur le jeu de test. Pour vérifier que j'ai assez d'individus pour la prédiction, j'utilise les variables n_util_aic et n_total_test et on voit que j'ai respectivement 289 et 292 donc, je n'ai pas perdu trop d'individus et l'évaluation du modèle par AIC reste assez fiable.
Pour finir, le modèle s'ajuste très bien aux nouvelles données du test, en effet, le coefficient de détermination obtenu sur l'échantillon de test est de 0.9221974, ce qui signifie que le modèle arrive à capter plus de 92% de la variance.

#### BIC

J'ai aussi appliqué une sélection backward avec comme critère le BIC cette fois-ci, plus pénalisant que l'AIC. Le modèle retenu contient seulement 24 variables et obtient un $R^2$ ajusté sur le train de 0.9066, ce qui est très bien aussi. On remarque donc que la baisse du $R^2$ par rapport au modèle AIC est très faible malgré que le nombre de variables ait baissé de presque la moitié, ce qui montre que l'AIC supprime beaucoup de variables peu significatives mais conserve un modèle qui explique bien les données.
Je suis ensuite le même schéma pour le modèle retenu par le critère BIC. Là aussi, très peu d'observations sont exclues, avec 290 individus finalement utilisés pour la prédiction sur 292 dans le jeu de test. Le $R^2$ obtenu sur le jeu de test est de 0.9293, ce qui montre que le modèle BIC explique plus de 92 % de la variance des données aussi, mais en s'appuyant sur environ deux fois moins de variables que le modèle AIC (24 contre 51).

**Comparaison**

```
MODELE AIC
 - Nombre de variables : 51
 - R² ajusté (train) : 0.9286595
 - SSE (test) : 3.955222
 - SST (test) : 50.83664
 - R² (test) : 0.9221974
 - Observations utilisées (test) : 289 / 292

MODELE BIC
 - Nombre de variables : 24
 - R² ajusté (train) : 0.9066455
 - SSE (test) : 3.65401
 - SST (test) : 51.66069
 - R² (test) : 0.9292691
 - Observations utilisées (test) : 290 / 292
```

FIGURE 3 – Sortie R comparant les deux modèles

En comparant les deux modèles, on remarque que le modèle BIC obtient un $R^2$ ajusté très légérement plus élevé sur le test que le modèle AIC, alors que le modèle AIC avait un meilleur $R^2$ ajusté sur le train.
Cela montre que le fait d'ajouter beaucoup de variables dans le modèle AIC, non seulement n'améliore pas les performances mais peut même les dégrader (sur le jeu de test), ce qui laisse aussi penser que le modele AIC fais peut être un peu de surajustement en gardant autant de variables.

### 0.5.3 Vérification des hypotheses

**Linéarité**

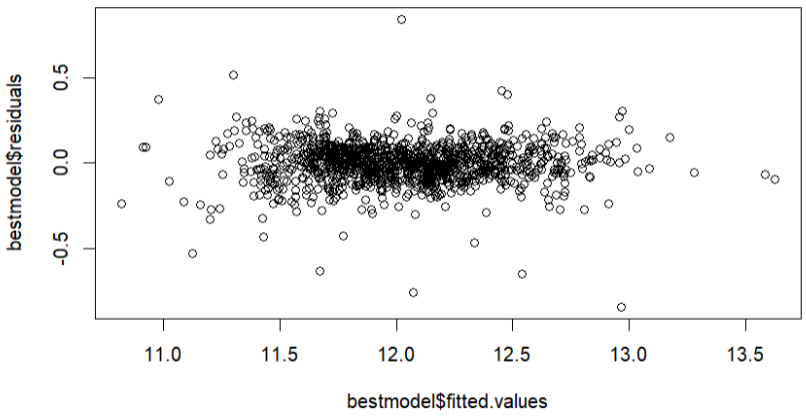


FIGURE 4 – Résidus en fonction des valeurs ajustées.

Le graphique des résidus en fonction des valeurs ajustées ne montre pas de structure particulière. Les résidus sont globalement centrés autour de zéro et n'ont pas une tendance

particulière, donc on peut dire que cette hypothèse est respectée.

### Homoscédasticité

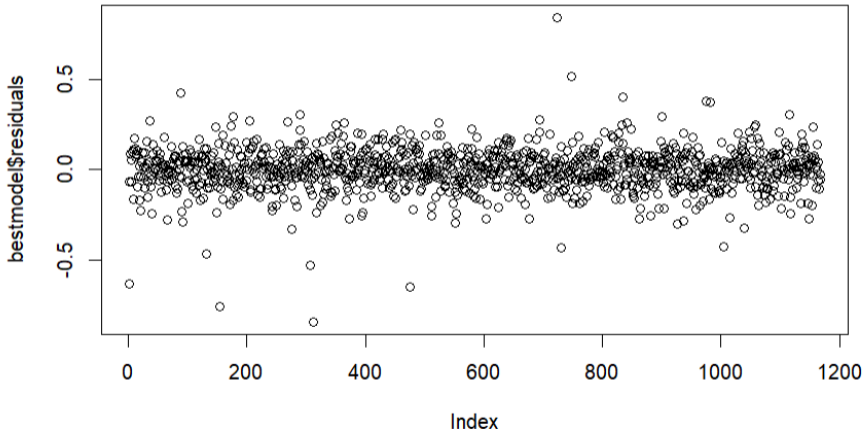


FIGURE 5 – Résidus en fonction des observations.

On voit que la dispersion des résidus est la même de gauche à droite sur toute la plage des valeurs prédites, sauf pour quelque points aberrants. La largeur du nuage reste constante entre -0.3 et +0.3 environ, cela confirme que la variance des erreurs est constante.

### Normalité

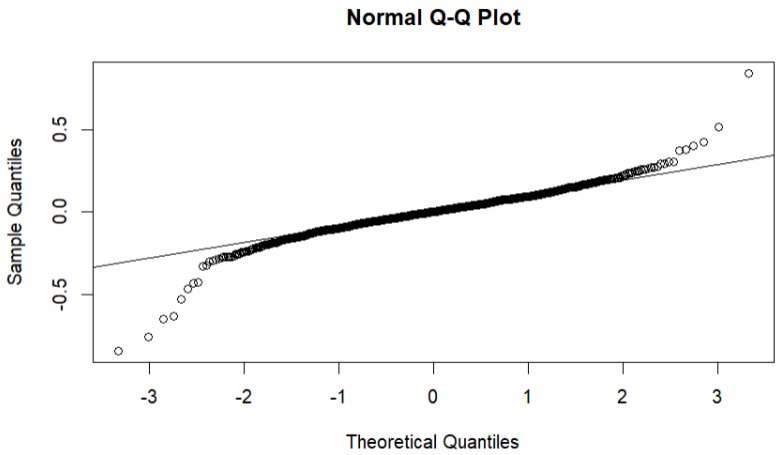


FIGURE 6 – Q-Q plot des résidus

Pour voir si les résidus de notre modèle sont bien normalement distribués, on utilise un QQ-plot. On voit que les résidus suivent sont aligné avec la droite au centre, mais ce n'est pas le cas aux extremités. En haut, à droite, les résidus observés sont plus grands

que les théoriques, cela siginife que le modèle sous-estime les prix de vente très élevées, et inversement, le modèle surestime les prix très bas ( les résidus observés sont plus petits que théorique), malgré qu'on ait utilisé une échelle logarithmique. Pour compter les outliers, j'ai utilisé la fonction outlierTest() de la librairie car, qui a identifié 8 outliers significatifs. Mais, globalement le modèle reste de bonne qualité, je pense qu'on peut aussi valider cette hypothèse. Les trois hypothèses sont validées donc on peut valider ce modèle à 24 variables.

### 0.5.4 Soumission Kaggle

Enfin j'ai voulu tester mon modèle BIC sur Kaggle. J'ai réalisé une première soumission mais j'avais oublié de compter les 20% de données que j'avais utilisées pour le "test", j'ai soumis avec 80% du train et j'ai obtenu 0,14951. Ensuite en ajoutant les anciens 20% de test au train, j'obtiens 0,14388.
Ces scores sont légèrement meilleurs que la régression linéaire classique (0,15573), ce qui montre que le modèle BIC à 24 variables améliore un peu la prédiction. On voit aussi que avec plusde données d'entraînement, le score s'améliore. Si on avait une plus grande base de données on pourrait encore améliorer les performances pour limiter le nombre d'outliers et mieux gérer les modalités qui ne sont pas fréquentes car on a vu que certaines modalités des variables qualitatives sont très rares dans les données. Le modèle a donc du mal à bien estimer leur effet sur le prix de vente, ce qui dégrade la qualité de la prediction. Mais, ce challenge est fait pour des méthodes de boosting, comme XGBoost, qu'on a testé et avec lequel on a otenu le meilleur score jusqu'ici.

## 0.6 Perspectives

Premièrement, on a vu que certaines variables qualitatives étaient déséquilibrées. Par exemple, la variable MiscFeature compte 1406 logements avec la modalité None, ce qui laisse très peu d'observations pour les autres modalités (Gar2 ( deuxième garage ), TenC ( terrain de tennis ), Elev ( ascenseur ) etc.). Dans ce cas, le modèle a peut-être du mal à bien apprendre l'effetsur le prix. Une amélioration possible aurait été de fixer un seuil minimal de fréquence ou de regrouper les modalités peu représentées ou bien, supprimer les colonnes concernées.
Aussi, on a vu que ce challenge était plus adapté à des méthodes de boosting. Je me renseignerais sur les méthodes de fine-tuning comme la recherche par grille (Grid Search) ou la recherche aléatoire (Random Search), pour mieux ajuster les paramètres des modèles et essayer d'obtenir un meilleur score.

# Conclusion

Pour conclure, ce projet montre bien que la data science peut vraiment être utile dans l'immobilier : avec des données sur les caractéristiques d'un logement, on arrive déjà à construire des modèles capables de prédire correctement les prix et d'aider à comprendre ce qui les influence. Même si certains points restent difficiles à gérer (modalités rares, outliers, etc.), cela illustre bien comment ces méthodes peuvent être utilisées par des agences pour analyser le marché, réduire les erreurs et les risques, ainsi qu' améliorer la prise de décision.

## 0.7 Bibliographie


— Yasmin Narcizo (2024), Comment l'utilisation de l'analyse des données dans l'immobilier renforce votre réputation en tant qu'agent
— Scikit-learn documentation – ColumnTransformer & preprocessing
— Scikit-learn documentation - 1. Supervised learning - 1.1. Linear Models
— Cours du semestre