Rapport de projet

Estimation de biens immobiliers

PAR : OULEYMATOU KONE KANTE, LÉA YANG ET YOUSSRA BELMEKKI

Introduction

Dans le cadre de notre projet de fin d'année en cours de Machine Learning, nous avons participé au **Real Estate Price Prediction Challenge**. Ce défi s'inscrit dans un contexte économique et technologique en constante évolution, où optimiser les prises de décision devient essentiel dans de nombreux secteurs.
En particulier, le marché immobilier français est en pleine mutation. L'évaluation précise des biens y est devenue un enjeu stratégique. L'objectif de ce challenge est donc de développer un modèle de machine learning capable de prédire avec fiabilité le prix des logements.

Présentation du Data Set

Le dataset utilisé est un extrait de données issues du marché immobilier français. Il comprend environ 50000 observations. Pour garantir un entraînement et une évaluation robustes, les données ont été divisées en deux sous-ensembles :

- Un jeu d'entraînement de 40000 observations (soit 80% du total), utilisé pour construire le modèle.
- Un jeu de test de 10000 observations (20%), utilisé pour évaluer les performances sur des données inédites.
- Il y a 28 colonnes dont 27 représentent les features.

A - Description détaillée des variables :

La variable cible est le prix, une variable numérique continue. Les autres variables se répartissent en deux types :

- Numériques, comme nb_rooms (nombre de pièces) ou approximate_longitude (coordonnées de localisation), qui sont pris en compte par nos modèles de machine learning
- Catégorielles, comme property_type (type de bien) ou has_a_cellar (présence d'une cave) qui ne peuvent pas servir à l'entrainement de nos modèles et qu'on doit par conséquent modifier ou supprimer.

B - Analyse de la qualité des données et premiers constats

L'analyse préliminaire a révélé la présence de valeurs manquantes dans plusieurs variables, ce qui représente un réel défi pour la modélisation. La Figure 1 met en évidence ce phénomène. Certaines variables, comme exposition et étage, présentent un taux de valeurs manquantes très élevé (respectivement 75,6 % et 73,9 %), tandis que d'autres, comme type de logement, sont complètes.
Cette visualisation a joué un rôle clé dans l'orientation de nos choix de prétraitement, car la qualité des données influe directement sur la fiabilité et la performance des prédictions réalisées par les modèles.

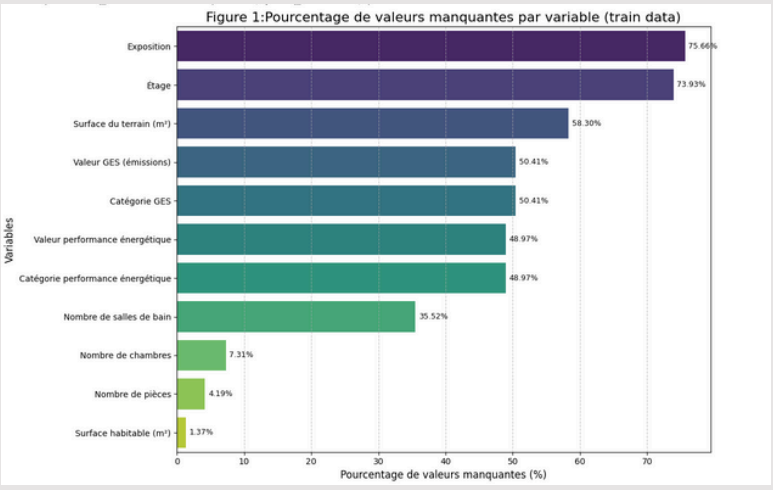


Figure 1

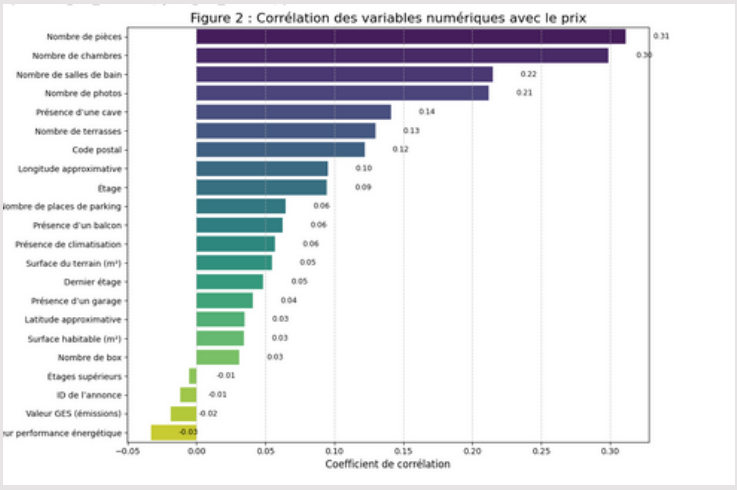
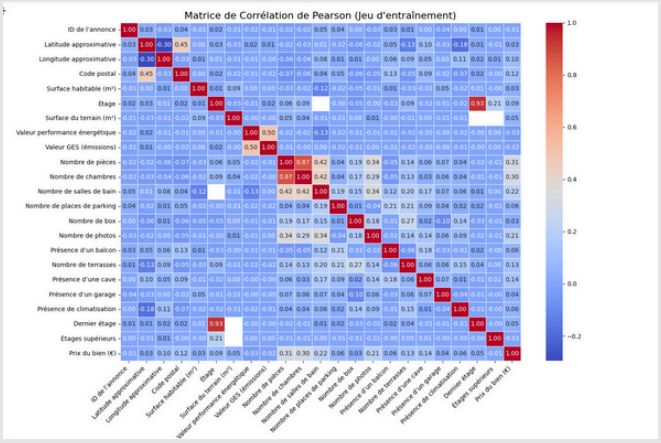


Figure 2

Figure 3



Afin de mieux comprendre l'influence des différentes caractéristiques des logements sur leur valeur, nous avons analysé l'existence d'une corrélation linéaire entre les variables numériques de notre jeu de données et le prix.
Les variables les plus corrélées au prix sont le nombre de pièces (0,31), le nombre de chambres (0,30) et le nombre de salles de bain (0,22). Toutefois, la majorité des coefficients de corrélation restent faibles, proches de zéro, ce qui suggère l'absence de relation linéaire directe et forte entre ces variables et le prix.
Cela laisse penser que le lien entre les caractéristiques des logements et leur valeur est plus complexe et probablement non linéaire, ce qui justifie l'utilisation de modèles capables de capturer ce type de relations.

I-Nettoyage des données

La qualité et la préparation des données sont des étapes indispensables dans un projet de machine learning, car elles conditionnent directement la performance des modèles.
Dans le cadre de ce challenge, nous avons rapidement été confrontés à deux difficultés majeures : la présence de nombreuses valeurs manquantes, et l'impossibilité pour les algorithmes de traiter directement les variables catégorielles sous forme textuelle. Ces deux problématiques sont étroitement liées à la structure même des données, et ont donc demandé une attention particulière dès les premières étapes du projet.
Pour avancer sereinement vers la modélisation, nous avons dû mettre en place des stratégies adaptées à ces enjeux. Cette section détaille les choix que nous avons faits pour nettoyer, transformer et enrichir notre jeu de données afin de le rendre pleinement exploitable par les modèles d'apprentissage.

A - Gestion des valeurs manquantes et vérifications logiques

AAvant d'appliquer des méthodes classiques d'imputation, nous avons d'abord procédé à une vérification logique et contextuelle des données, afin de garantir leur cohérence.
Nous nous sommes notamment concentrés sur les variables size (surface habitable) et land_size (surface du terrain). Leur interprétation dépend fortement du type de bien. Par exemple, pour des appartements, duplex ou chambres, il est normal que la variable land_size soit absente, puisqu'il n'y a pas de terrain. À l'inverse, pour des terrains ou terrains à bâtir, il est logique que la surface habitable (size) soit vide.
En suivant cette logique, nous avons remplacé par 0 les valeurs manquantes qui étaient structurellement absentes, plutôt que réellement perdues. Cette approche nous a permis de réduire significativement le taux de valeurs manquantes, notamment pour la variable size, qui est passé de 58 % à 17 %.
Concernant les autres variables numériques, nous avons ensuite appliqué une imputation, en privilégiant la médiane plutôt que la moyenne. La médiane étant moins sensible aux valeurs extrêmes, elle permet une meilleure représentation des données réelles, tout en limitant l'impact des outliers sur le modèle.

B- Encodage des Variables Catégorielles

Les algorithmes de machine learning ne peuvent traiter que des données numériques. Par conséquent, toutes les variables catégorielles, comme type de logement, ont dû être transformées. Pour cela, nous avons utilisé la méthode de One-Hot Encoding (OHE), qui consiste à créer une nouvelle colonne pour chaque modalité, avec une valeur binaire (1 ou 0) selon la présence ou non de cette modalité dans l'observation.
Avant d'appliquer cette transformation, nous avons d'abord remplacé les valeurs manquantes dans les variables catégorielles par la valeur 'Inconnu', afin de ne pas perdre d'information lors de l'encodage.
Nous avons ensuite porté une attention particulière aux variables binaires, comme has_cellar ou has_garage, qui indiquent la présence ou non de certains équipements. En analysant leur fréquence, nous avons constaté que la grande majorité des logements (environ 80 %) n'avaient ni garage, ni climatisation.
Ces variables présentaient donc une faible variabilité et se révélaient peu discriminantes pour les modèles. Elles n'apportaient que peu d'information utile à l'analyse, et risquaient même de générer du bruit. Nous avons donc pris la décision de les supprimer du jeu de données.

Nous avons ensuite identifié les features multi-catégorielles, comme type de logement ou ville, et examiné le nombre de classes qu'elles contiennent. Cette étape est essentielle pour choisir la méthode d'encodage la plus appropriée. Pour les variables comme energy_performance_category ou ghg_category, qui suivent un classement de type A à G, nous avons utilisé un encodage ordinal. Cette méthode permet de conserver la notion d'ordre entre les catégories, ce qui est pertinent pour ce type de données.
Pour les autres variables catégorielles, nous avons appliqué le One-Hot Encoding (OHE), car il permet de représenter les catégories sans introduire d'ordre arbitraire entre elles — contrairement à un encodage numérique simple qui pourrait induire une relation hiérarchique inexistante. L'OHE crée une nouvelle colonne pour chaque modalité unique d'une variable, où chaque ligne reçoit la valeur 1 si cette modalité est présente, et 0 sinon.
L'application de ces techniques, et en particulier du One-Hot Encoding, a naturellement entraîné une augmentation significative du nombre de features dans le dataset. En effet, pour chaque variable catégorielle transformée, le nombre de colonnes générées est égal au nombre de catégories uniques. Cette transformation peut rapidement faire passer le nombre total de variables de quelques dizaines à plusieurs centaines, voire des milliers, ce qui a un impact direct sur la taille du dataset, le temps de calcul des modèles, et parfois même sur la mémoire nécessaire à l'entraînement.

III-Modélisation

On voit avec une heatmap (Figure 3) qu'aucune caractéristique n'a de corrélation forte avec le prix mais que certaines variables suivent quand même légèrement le prix, c'est le cas du nombre de pièces et de chambres avec un taux de corrélation de 0,3, d'autres villes suivent légèrement cette corrélation également. Ainsi en observant cela, nous avons décidé dans un premier temps de réaliser une régression linéaire qui est le modèle le plus simple pour commencer la prédiction des données.

Notre première soumission avec la régression linéaire en utilisant toutes les villes du dataset de training nous avait permis d'avoir un score de 60,61. Cependant l'utilisation de toutes les villes du dataset avec le onehot encoding a fait exploser le nombre de colonnes du training data, rendant l'entraînement des modèles autres que régression linéaire impossible. Nous avons alors décidé de supprimer toutes les villes en nous disant que les latitudes et les longitudes donnaient déjà ces informations mais nous avons obtenu un score de 83,43 pour la régression linéaire. Finalement, nous avons dû trouver un équilibre entre performance et temps de calcul entre les deux approches en ne prenant que les 20 villes qui apparaissent le plus, ce qui nous a permis d'avoir un score de 75,84. (Ce chiffre 20 a été choisi de manière arbitraire et nous aurions pu utiliser des pipelines avec GridSearchCV ou RandomizedsearchCV pour optimiser ce paramètre mais la mise en place de ces calculs aurait été trop coûteux.)

En observant ces scores, on peut facilement en déduire qu'inclure toutes les villes permet d'obtenir une meilleure précision, mais au prix d'un modèle plus complexe. Cela peut être dû au fait que le modèle de régression linéaire ne prend pas les coordonnées spatiales en compte donc les latitudes et longitudes ne sont pas utiles pour prédire les prix dans ce modèle, d'où l'importance de spécifier clairement les villes. De plus, la performance des scores peut être expliquée par les faibles corrélations des variables avec le prix. Ainsi, nous avons testé des modèles plus complexes qui utilisent les coordonnées spatiales: la régression avec les k-plus proches voisins, RandomForest et Xgboost.

Ces modèles nécessitent des hyperparamètres tels que le nombre de voisins pour les plus proches voisins, ou la profondeur des arbres pour RandomForest. Ces hyperparamètres sont importants pour entraîner un bon modèle et leur choix est crucial pour éviter l'overfitting et l'underfitting. Ainsi, nous avons utilisé les fonctions GridSearchCV et RandomizedSearchCV de sklearn qui réalise k-fold cross validation automatiquement pour des données d'hyperparamètre données. L'utilisation de ces fonction a nécessité pas mal de temps mais nous a permis d'obtenir des résultats plutôt satisfaisants que nous avons décidé de comparer en utilisant comme mesure de performance l'estimateur des moindres carrés, la racine de l'estimateur des moindres carrés qui nous permet de voir l'erreur moyenne des prédiction en euros et le R-carré, le coefficient de détermination et mesure la proportion de la variance des valeurs d'une variable dépendante expliquée par le modèle de régression. Le coefficient de détermination R^2 est défini mathématiquement comme la somme des carrés des résidus expliqués par le modèle divisée par la somme totale des carrés. Un R^2 proche de 1 signifie que le modèle offre un ajustement pertinent aux données, expliquant une grande partie de la variance observée.

Nous avons également calculé le score de cross validation calculé avec R-carré. Ce score nous permet d'évaluer la performance réelle du modèle en réalisant des tests sur les données qu'il n'a pas vu pour observer sa capacité à généraliser le modèle. Nos données d'entraînement sont divisées en 5 folds, entrainé que 4 et testé sur 1, cette opération est répétée 5 fois. Ainsi nous obtenons 5 scores de cross validation. Nous utilisons la moyenne de ces scores pour la comparer au r-carré calculé précédemment sans cross-validation et nous regardons également l'écart-type de ces scores pour voir si le modèle est stable.

	Modèle	MSE (train)	RMSE (train)	R² (train)
0	Regression lineaire	6.716106e+10	259154.504448	0.296187
1	kNeighborsRegressor	4.700845e+10	216814.324850	0.507376
2	RandomForest+GridSearchCV	2.336226e+10	152847.188049	0.755176
3	RandomForest+RandomizedSearchCV	3.029185e+09	55038.032031	0.968256
4	xgboost+GridSearchCV	3.029185e+09	55038.032031	0.968256
5	xgboost+RandomizedSearchCV	2.346077e+09	48436.321764	0.975414

	Modèle	score de cross validation	moyenne du score)	écart-type
0	Regression linéaire	[0.28182217, 0.30285313, 0.25542904, 0.2964687...	0.283737	0.016341
1	knn	[0.3935685489036954, 0.40678901784264576, 0.38...	0.389718	0.009657
2	RandomForest+GridSearchCV	[0.6612307119824241, 0.6980879500778442, 0.676...	0.672925	0.014289
3	RandomForest+RandomizedSearchCV	[0.7499490543449587, 0.7735412414180676, 0.758...	0.755950	0.009961
4	Xgboost+GridSearchCV	[0.7878524206724886, 0.8124624707308541, 0.789...	0.789459	0.012418
5	Xgboost+RandomizedSearchCV	[0.7948668693325299, 0.8167260240268442, 0.802...	0.799760	0.009436

Aussi, dans le cadre de notre projet, nous avons essayé de comprendre le fonctionnement de XGBoost, un modèle puissant souvent utilisé en machine learning.
Random Forest construit plusieurs arbres de décision en parallèle, chacun sur un échantillon différent des données, puis combine leurs résultats pour obtenir une prédiction stable et réduire le risque d'erreur.
XGBoost, en revanche, construit les arbres de manière séquentielle : chaque nouvel arbre vise à corriger les erreurs faites par les arbres précédents. Cette méthode, appelée boosting, permet souvent d'améliorer la précision, mais nécessite un réglage plus précis des paramètres.

IV-ALLER PLUS LOIN : PERSPECTIVES D'AMÉLIORATION ET COMPROMIS

Dans le cadre de ce data challenge, nous disposions d'images pour chaque logement. L'exploitation de ces données visuelles aurait pu considérablement améliorer la qualité de nos prédictions. Cependant, faute de temps, nous n'avons pas pu les intégrer dans notre modèle. Nous avons néanmoins réfléchi à la manière dont cette analyse aurait pu être menée.

Pour traiter ces images, nous nous serions appuyés sur les Réseaux de Neurones Convolutifs (CNN), une technologie abordée dans notre cours de Deep Learning. Ces réseaux auraient permis d'extraire de nouvelles caractéristiques à partir des images, telles que la luminosité, le style de décoration, les couleurs ou encore la perception de l'espace. L'intégration de ces informations visuelles aurait sans doute affiné nos analyses et amélioré la qualité des prédictions.

Cela dit, l'ajout de ces nouvelles features soulève un défi important : il aurait considérablement alourdi notre dataset et ralenti le temps d'entraînement des modèles. Ce constat illustre un enjeu central en data science : le compromis entre précision du modèle et contraintes opérationnelles. En effet, nous avons observé que plus un modèle est performant, plus son temps d'entraînement a tendance à augmenter, ce qui peut aussi engendrer des coûts de stockage plus élevés.

Nous pensons que, dans un contexte professionnel, les data scientists sont régulièrement confrontés à ce dilemme et doivent trouver un équilibre entre performance optimale et ressources disponibles (temps, calcul, stockage).

Ce challenge a été une excellente opportunité d'approfondir notre compréhension des modèles de Machine Learning étudiés durant le semestre, en les appliquant à un cas concret. Au-delà de la technique, cette expérience nous a permis de sortir de notre zone de confort et d'appréhender de manière plus concrète la manière dont les professionnels de la data science travaillent et réfléchissent au quotidien.