Prévision des prix de l'immobilier

BELKHITER Yannis, FORAY Léo-Paul, MU Maxime, TEXIER Lucas

5 avril 2023

Résumé

Le marché de l'immobilier, que ce soit en Europe ou en Amérique du Nord, est complexe à étudier et à suivre. Les prix varient pour de multiples raisons, et il est difficile d'estimer la valeur d'un bien, même pour un professionnel. Notre projet vise à utiliser des algorithmes d'apprentissage automatique pour établir un modèle permettant d'évaluer le prix d'une maison en fonction de ses caractéristiques, et d'analyser l'influence des différentes caractéristiques sur le prix des maisons. Le jeu de données utilisé est celui de l'immobilier à Ames, Iowa, USA, et nous utiliserons différents modèles tels que la régression linéaire, l'arbre de décision, la forêt aléatoire et XGBoost. Notre travail est basé sur un jeu de données Kaggle composé de 81 colonnes et 1460 lignes, avec pour objectif de prédire la colonne "SalePrice" (le prix en dollars). La base de données Kaggle a été mise à disposition dans le cadre d'une compétition Kaggle. Nous avons extrait une base de données sans nous intéresser au réglement du concours. Dans l'ensemble, notre projet vise à fournir des informations sur le marché immobilier et à offrir un outil pratique pour estimer la valeur d'un bien immobilier en fonction de ses caractéristiques.

Table des matières

1 Introduction

1.1 Sujet et problématique

Le marché de l'immobilier, que ce soit en Europe ou en Amérique du Nord, est complexe à étudier et à suivre. Les prix varient pour de multiples raisons et il est complexe d'estimer la valeur d'un bien, même pour un professionnel. Vendre et acheter sont donc deux tâches complexes qui nécessitent une étude approfondie du marché, car les montants impliqués peuvent aller de quelques dizaines de milliers de dollars à plusieurs centaines pour certaines propriétés. Il est alors nécessaire de trouver des méthodes efficaces pour déterminer quels paramètres influencent le plus les ventes.

Nous nous proposons donc d'utiliser des algorithmes d'apprentissage automatique pour établir un modèle permettant d'évaluer le prix d'une maison à partir de ses caractéristiques, et d'analyser l'influence des différentes caractéristiques sur les prix des maisons. Le jeu de données porte sur l'immobilier à Ames, Iowa, aux Etats-Unis, sur lequel nous utiliserons différents modèles d'apprentissage supervisé.

1.2 État de l'art

Les problèmes semblables sont nombreux, il existe de nombreuses bases de données disponibles, qui portent sur des zones géographiques différentes, mais surtout s'intéressent à des caractéristiques différentes. Ainsi il existe d'importantes différences entre les jeux de données disponibles pour la Californie, ceux de Chicago ou encore ceux disponibles pour des villes européennes.

Par exemple, l'article "Housing Price Prediction Using Machine Learning Algorithms in COVID-19 Times"[1] présente différents algorithmes d'apprentissage automatique utilisés pour évaluer la valeur de propriétés provenant d'un dataset de propriétés de la ville d'Alicante, en Espagne. Cet article est très récent, publié en Novembre 2022, et propose des performances une approche complète du problème, comparant diverses méthodes de régression pour résoudre celui-ci. Il présente de plus un état de l'art complet et à jour mettant en valeur des travaux portant sur l'étude du marché immobilier à l'aide de l'apprentissage automatique, et ceux pour presque toutes les régions du monde, et ce particulièrement au alentours de la période Covid (2020 - 2022).

Notre jeu de données a été récupéré sur le site Kaggle. Il a été rendu disponible dans le cadre d'une compétition ouverte à tous il y a de cela 3 ans, ainsi de nombreux résultats sont déjà disponibles pour cet exercice. Il est alors intéressant de noter que certains compétiteurs annoncent des résultats de 100%, mais après vérification leur code utilise des données supplémentaires donnant le prix de chaque maison sur le jeu de test. Ainsi les meilleurs programmes obtiennent un score de 0.00044, cependant Kaggle ne précise pas le moyen de calcul des scores, on ne sait donc pas réellement ce que signifie cette performance.

Model	Name	CV-Validation in Training Set (*SD*)	R^2 Score		
			Training Set	Test Set	Overfitting (%)
Linear Regression	*lr*	0.8048 (0.0060)	0.8056	0.8052	-
Random Forest Regressor	*rf*	0.9036 (0.0049)	0.9970	0.9135	+9.1
Extra-Trees Regressor	*et*	0.9101 (0.0040)	0.9997	0.9178	+8.9
Gradient Boosting Regressor	*gbr*	**0.9125** (0.0034)	0.9952	**0.9192**	**+8.3**
Extreme Gradient Boosting	*xgbm*	0.9094 (0.0041)	0.9900	0.9178	+7.9
Light Gradient Boosting Machine	*lgbm*	0.9076 (0.0044)	0.9902	0.9140	+8.3

FIGURE 1 – Performance des algorithmes dans l'article [1]

1.3 Choix de l'apprentissage

Les modèles d'apprentissage automatique sont conçus pour apprendre à partir de données et à trouver des modèles qui peuvent aider à prédire les résultats futurs. Ces modèles sont également capables de gérer des données complexes et de grandes dimensions. Les résultats de ces modèles peuvent être quantifiés à l'aide de métriques, qui attestent de leur efficacité.

L'apprentissage automatique est très flexible, et permet aussi de s'adapter à chaque base de données. En effet, chaque modèle est entraîné sur une base de données d'entraînement et testé sur une base de données test. L'intérêt de l'apprentissage automatique réside aussi dans le fait de pouvoir appliquer notre modèle à une très grande quantité de données afin d'affiner au mieux les résultats obtenus.

Plus concrètement, notre problème invite à utiliser le machine learning, car il n'existe pas de modèles mathématiques permettant de calculer le prix d'une maison à partir de ses caractéristiques, qui sont elles très nombreuses, ce qui rend l'étude d'autant plus difficile. Il est donc pertinent dans ce cas, avec autant de données labellisées, de bonnes qualités et facilement accessibles, de se tourner vers l'apprentissage automatique.

2 La base de données

Notre travail s'appuie sur une base de données (BDD) fournie par Kaggle dans le cadre d'une compétition. Celle-ci recense des informations sur les propriétés de la ville d'Ames, Iowa, Etats-Unis.

Notre travail s'appuie sur une base de données (BDD) fournie par Kaggle dans le cadre d'une compétition. Celle-ci recense des informations sur les propriétés de la ville d'Ames, Iowa, Etats-Unis.

Au total, la BDD recense 81 colonnes et 1460 lignes. Le but de l'exercice est de prédire la colonne "SalePrice" c'est-à-dire les prix des maisons à prédire. Le prix des maisons varie de 30k à 755k dollars, cela justifie l'utilisation d'un tel modèle.

La base de données utilisée est un fichier csv nommé **Ames_houses.csv**

Pour commencer, nous avons extrait la colonne "SalePrice" du DataFrame pandas "df_global" issu de la base de données Ames_houses.csv, puis nous l'avons stockée à part. Le DataFrame "df_global" contient 79 colonnes au total.

2.1 Traitement des valeurs manquantes

Notre base de données peut contenir des valeurs qui ne sont pas renseignées par l'utilisateur. Pour éviter les erreurs de prédiction et les biais dans les données, il est important de veiller à ce que notre base de données ne contient pas de valeurs manquantes. En effet, la plupart des algorithmes d'apprentissage ne sont pas capables de traiter directement les données manquantes, ce qui peut affecter la qualité du modèle et entraîner des erreurs de prédiction.

Afin de résoudre ce problème, nous avons effectué un comptage du nombre total de données présentes dans la base de données. Nous avons ensuite tracé graphiquement la distribution des valeurs non renseignées par colonne :

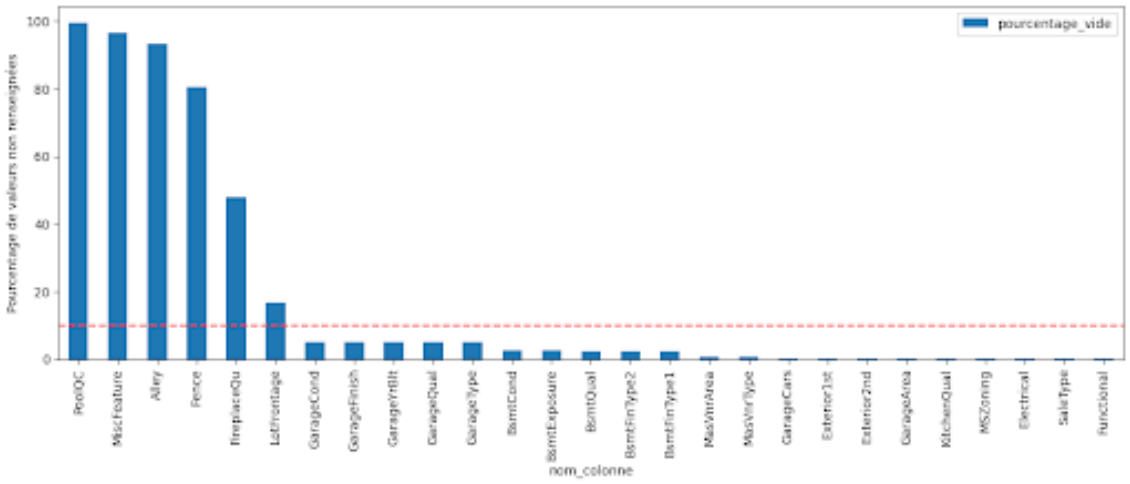

FIGURE 2 – Distribution des valeurs non renseignées

D'après ce graphique, nous avons constaté que de nombreuses colonnes contiennent peu de valeurs renseignées. Pour pallier ce problème, nous avons décidé d'établir un seuil de filtrage de 10%, ce qui signifie que seules les colonnes qui ont plus de 90% des données renseignées seront conservées. Le choix du seuil a été fait par tâtonnement en analysant les résultats.

On a ainsi filtré les six colonnes suivantes sur la base de données "df_global" : "PoolQC", "Misc-Feature", "Alley", "Fence", "FireplaceQu", "LotFrontage". Notre base de données possède maintenant 73 colonnes.

Après avoir réduit notre base de données conformément au seuil de filtrage fixé, nous devons désormais remplir les valeurs manquantes des colonnes qui ont plus de 90% des valeurs renseignées. Pour ce faire, nous avons traité deux cas différents :

— Pour les valeurs numériques, nous avons calculé la moyenne des valeurs de chaque colonne, puis remplacé les valeurs manquantes par cette moyenne. Là, nous avons choisi la moyenne mais nous aurions très bien pu prendre d'autre fonction comme la médiane mar exemple.
— Pour les valeurs catégorielles, nous avons compté le nombre d'occurrences de chaque valeur dans la colonne, puis remplacé les valeurs manquantes par celle la plus fréquente. Ici, une autre solution aurait pu être de mettre la valeur None pour certaines colonnes lorsqu'une valeur n'est pas renseignée en se diant que les vendeurs peuvent consciemment ne pas remplir un champs pour une caractéristiques que leur bien ne possède pas.

Enfin, nous avons vérifié que notre base de données ne contenait plus de valeurs manquantes.

2.2 Traitement des données catégorielles

Il est important de traiter les données catégorielles car de nombreux algorithmes d'apprentissage ne peuvent pas traiter directement les données catégorielles. Par conséquent, il est nécessaire de les transformer en une forme que les algorithmes peuvent comprendre. Cela peut être fait en utilisant des techniques telles que l'encodage one-hot, l'encodage binaire ou l'encodage de fréquence.

Parmi les techniques citées ci-dessus, nous avons choisi l'encodage one-hot qui transforme les variables catégorielles en un ensemble de variables binaires (0 ou 1) qui indiquent la présence ou l'absence d'une catégorie particulière. Chaque catégorie de la variable catégorielle est représentée par une colonne dans le nouvel ensemble de données, et chaque ligne contient un 1 pour la catégorie correspondante et un 0 pour toutes les autres catégories.

Nous avons débuté en utilisant la fonction OneHotEncoder de la bibliothèque Sk-learn pour encoder les colonnes qui contiennent des valeurs catégorielles. Cependant, nous avons finalement opté pour la fonction "get_dummies()" de Pandas, qui permet de transformer directement la base de données.

2.3 Séparation du jeu de données

Nous avons ensuite divisé l'ensemble de données en un ensemble d'entraînement et un ensemble de test à des fins d'évaluation des performances des modèles d'apprentissage automatique. Puisque le but de l'apprentissage automatique est de créer des modèles prédictifs qui peuvent généraliser à de nouveaux exemples de données qu'ils n'ont jamais vus auparavant.

Remarque : Nous avons encodé avant de diviser notre jeu de données, ainsi certaines colonnes peuvent correspondre à des valeurs uniquement comprises dans le jeu de test, cependant lors de l'encodage (cf 2.5) ces colonnes ne contiennent que des 0 pour le jeu d'entraînement : les colonnes seront ainsi supprimées.

Pour évaluer la capacité de généralisation d'un modèle, il est donc nécessaire de le tester sur un ensemble de données distinct de celui sur lequel il a été entraîné. En utilisant la méthode train_test_split() de Sk-learn, on peut diviser l'ensemble de données de manière aléatoire en un ensemble d'entraînement et un ensemble de test avec un pourcentage spécifié pour chacun d'entre eux. Cela permet de s'assurer que les données d'entraînement et de test sont représentatives de l'ensemble de données complet.

Nous avons donc séparé la base de données "df_global" en une base de données d'entraînement ("X_train", "y_train") et en une base de données test ("X_test", "y_test"). Pour tous nos modèles, nous avons pris 80% des données pour la base de données train et 20% pour la base de données test.

2.4 Normalisation des données

La normalisation des données est une étape qui permet de mettre à l'échelle toutes les valeurs de la base de données dans une plage standardisée entre 0 et 1. Si les données ne sont pas normalisées, les variables ayant des plages de valeurs plus grandes peuvent avoir un impact disproportionné sur le calcul de la distance ou de la similarité par rapport aux autres variables. Cela peut fausser les résultats et conduire à des prédictions moins précises.

Pour mettre en place la normalisation, nous avons utilisé la bibliothèque StandardScaler de Sk-learn. La méthode fit_transform() est utilisée sur la base de données d'entraînement "X_train" pour ajuster

les paramètres de normalisation à cette base de données, puis la méthode transform() est utilisée sur la base de données de test “X_test” pour normaliser cette dernière en utilisant les paramètres obtenus sur la base d’entraînement. On fait la même procédure sur la variable cible (SalePrice). Cependant, nous constatons que la normalisation ne conduit pas à de meilleurs résultats que lorsque nous n’avons pas normalisé les données. Pour diminuer la complexité de nos implémentations, nous avons décidé de ne pas utiliser la normalisation.

2.5 Application de la corrélation

On calcule ensuite la matrice de corrélation entre les différentes variables de la base de données d'entraînement, puis on filtre les colonnes ayant une corrélation supérieure à 0,5 (seuil trouvé par tâtonnement) avec la variable cible (SalePrice). Enfin, on supprime la variable cible de cette liste de colonnes sélectionnées.

Nous nous concentrons exclusivement sur la base de données d'entraînement car l'utilisation de la base de données de test pourrait biaiser notre modèle. Pendant l'entraînement, les sorties de la base de données de test ne doivent pas être connues afin de ne pas altérer la validité des résultats.

En sélectionnant les colonnes ayant une corrélation supérieure à 0,5 avec la variable cible, on cherche à identifier les variables qui ont une forte influence sur la variable cible (SalePrice) et qui peuvent donc être utiles pour prédire les valeurs de la variable cible. Cette méthode permet de réduire le nombre de variables à utiliser dans la modélisation et de se concentrer sur les variables les plus importantes. Pour vérifier nos résultats, nous avons construit une carte thermique de la base de données train à partir de la bibliothèque Seaborn.

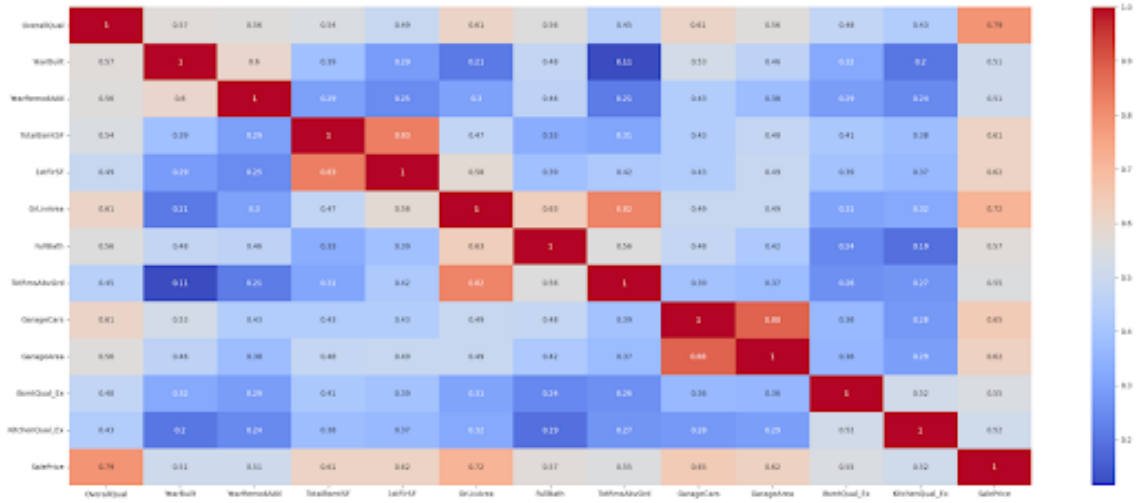

FIGURE 3 – Carte thermique de la base de données "X_train"

Nous vérifions que les valeurs de la colonne "SalePrice" sont supérieures au seuil de 0,5 que nous avions préalablement fixé.

Ainsi, notre base de données ne contient plus que les valeurs les plus corrélées par rapport à la caractéristique "SalePrice" et notre base de données est désormais prête à être utilisée par les modèles d'apprentissage automatique.

3 Formalisation du problème

3.1 Formulation du problème d'apprentissage automatique

Nous possédons un jeu de données de 1460 valeurs et 81 caractéristiques, incluant le prix, qui est la valeur à déterminer pour chacune de ces valeurs. Ainsi, pour chaque valeur $x_n = \{x_{i,n} \mid i \in [1, 80]\}$, $n \in [0, 1459]$ on cherche à prédire son prix associé y_n. Pour cela, nous allons utiliser un ensemble d'hypothèses H, dans l'optique de trouver une hypothèse $h \in H$, tel que pour tout $n \in [0, 1459]$, $h(x_n) = y_n$.

Il s'agit ici d'un problème de régression. On peut cependant le décliner en classification sous réserve de transformer le prix en classe de prix. Dans notre étude nous avons considérer pour la classification de 4 classes (0, 1, 2, 3) correspondant aux 4 quartiles, ainsi les maisons de classe 0 sont les plus modestes et inversement pour la classe 3.

3.2 Plan d'Expérience DOE

Pour formaliser cette étude, il est aussi important de réaliser un plan d'expérience, pour établir la population, l'échantillon étudié, ainsi que la problématique qui motive notre recherche.

De fait, les 1460 valeurs dont nous disposons pour réaliser notre étude forment notre population. Nous utiliserons un échantillon de 20% d'entre elles pour effectuer notre apprentissage, soit 292 valeurs. A l'aide de ces données, nous souhaitons comparer les performances des modèles existants sur le problème formulé dans le 3.1, ainsi que d'analyser l'influence des différentes caractéristiques sur le résultat.

3.3 Approches étudiées

Dans un premier temps, nous allons nous intéresser aux approches de régression à travers ces différents modèles :

1. **Régression linéaire :** ce modèle établit une ligne de tendance à partir des données d'entraînement, et se base sur son coefficient directeur pour effectuer des prédictions.
2. **Decision Tree regressor :** les modèles d'arbre de décision sont faciles à implémenter et à lire, et proposent des performances intéressantes dans diverses situations, mais ont tendance à être surentraînés.
3. **Random Forest regressor :** un modèle de Random Forest est un ensemble d'arbres de décision travaillant sur des échantillons aléatoires et distincts de caractéristiques et valeurs initiales. Leur fonctionnement est décrit plus précisément en annexe.
4. **Gradient Boosting :** ce modèle consiste à fusionner les prédictions de modèles simples mais mauvais pour combiner leurs forces afin d'obtenir un modèle meilleur. L'objectif est de diminuer au plus l'erreur du modèle principal, on s'intéresse donc au gradient de l'erreur, d'où le nom du modèle. Il crée donc différents arbres les uns après les autres dans l'optique d'améliorer progressivement ce modèle principal.
5. **Extreme Gradient Boosting :** ce modèle cherche lui aussi à établir un meilleur modèle à partir de modèles moins bons, comme pour le gradient boosting. Cependant celui-ci crée les arbres parallèlement, comme le ferait un modèle Random Forest.

Ensuite, nous nous sommes intéressés à des méthodes de classification :

1. **Régression Logistique :** la régression logistique est un modèle de classification binaire. Comme nous avons plusieurs classes (ici 4), nous utilisons une stratégie One vs Rest : nous étudions chaque classe avec les 3 autres.
2. **Decision Tree regressor :** on utilise le modèle de Decision Tree appliqué aux problèmes de classification.
3. **Random Forest classifier :** on utilise le modèle de Random Forest appliqué aux problèmes de classification. Leur fonctionnement est décrit plus précisément en annexe.

4 Résultats et solution finale

4.1 Les métriques utilisées

Pour comparer nos résultats, nous avons choisi d'utiliser quatre métriques pour la régression et 2 pour la classification :

1. **Root Mean Square Error (RMSE)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1}$$

2. **Mean Absolute Error (MAE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

3. **Mean Square Error (MSE)**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

4. **R^2 Score**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

5. **L'exactitude**
   Cette métrique n'apparaît que pour la classification. C'est le pourcentage de bonnes réponses.

6. **Matrice de confusion**
   La matrice de confusion permet d'observer la répartition des prédictions pour chaque classe. Cela permet de se rendre compte de la répartition des mauvaises prédictions.

4.2 Nos résultats pour la problématique de régression

4.2.1 Comparaison des différents modèles

Nos résultats sont les suivants :

Modèle	RMSE	MAE	MSE	R^2
Linear	38382,8	22846,2	1521618151	0,716
Decision tree	42178,2	27317,8	1835769740	0,712
Random Forest	33062,4	20272,2	1139606668	0,7843
Gradient Boosting	**30906,5**	**19469,2**	**1008633722**	**0,818**
Extreme Gradient Boosting	33290	21166,9	1157834287	0,7963

TABLE 1 – Comparaison des modèles de régression

Pour tous les résultats, la valeur de toutes les métriques résulte d'une moyenne de 20 itérations pour augmenter la fiabilité des mesures.

4.2.2 Influence de la taille du nombre de données d'entraînement

Une première analyse a été de faire varier la taille de notre base de données et donc la taille de notre jeu d'entraînement pour tenter de déterminer une taille optimale, et de fixer un seuil de sur-entraînement (overfitting). Pour chaque modèle et chaque valeur de n, nous avons réalisé 20 entrainements , sur base de donnée d'entrainement de taille n séléctionnées aléatoirement dans la base de données initiale. Nous avons ensuite fait varier la taille n de 200 à 1460 avec un pas de 50.

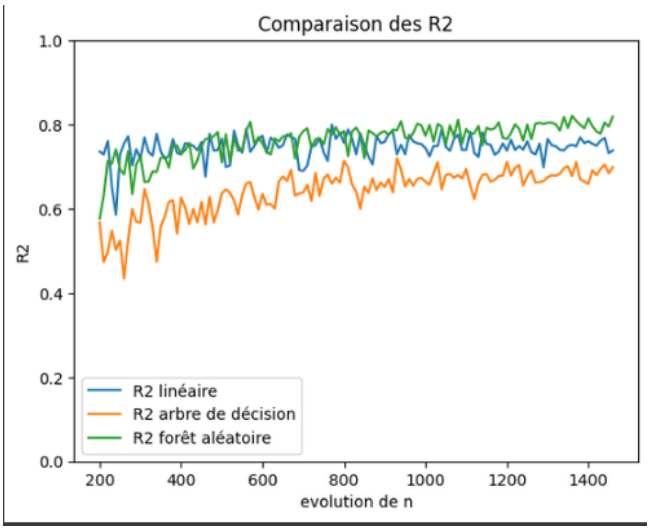

FIGURE 4 – Comparaison des valeurs de R^2 des différents modèles en fonction de la taille de la base de données d'entraînement

Nous pouvons voir sur cette figure que la valeur de R2 augmente avec la valeur de n, et stagne à partir d'un certain moment. Le modèle Random Forest semble se détacher des deux autres, mais nous ne parvenons pas à cerner une valeur n_optimale puisque les valeurs de R^2 des différents modèles ne chutent pas. Pour cela, il faudrait augmenter la taille de notre base de données initiale. Cependant, on peut observer que la valeur de R^2 ne chute pas brutalement : il n'y a pas de surentraînement.

4.2.3 Influence du nombre d'arbres de décisions pour les forêts aléatoires

Par la suite, nous nous sommes interessé au modèle de forêt aléatoire. Il est possible de faire varier le nombre a d'arbres de décision généré par le modèle lors de chaque entrainement. De fait, pour un dataset fixé, nous avons fait varier ce nombre a.

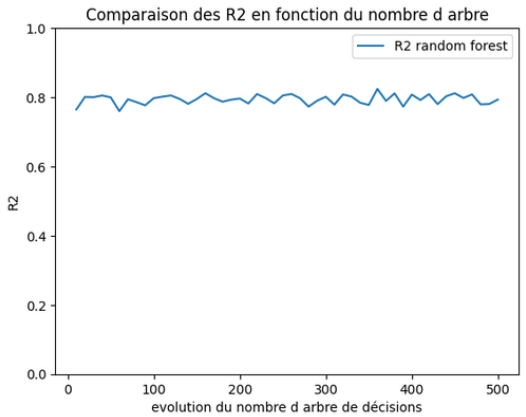

FIGURE 5 – Comparaison des valeurs de R2 des différents modèles en fonction de la taille de la base de données

Nous observons une augmentation en score R2 pour les premières valeurs de a, puis les performances du modèle stagnent, ce qui signifie qu'augmenter le nombre d'arbres de décision n'a plus d'impact sur le résultat.

4.3 Modèle retenu pour la régression

C'est le modèle Gradient Boosting qui obtient les meilleures performances pour toutes les métriques que nous avons choisies. C'est donc ce modèle que nous retenons concernant la régression.

4.4 Nos résultats pour la problématique de classification

4.4.1 Comparaison des différents modèles

Nos résultats sont les suivants :

Modèle	Accuracy
Random Forest	**0,718**
Régression Logistique	0,6751
Decision Tree	0,6513

Table 2 – Comparaison des modèles de classification

Aussi, nous avons utilisé comme autre métrique les matrices de confusions. La matrice a permis de se rendre contre de la répartition des prédictions. Ainsi, on a constaté que pour les modèles Random Forest et Decision Tree, les erreurs sont globalement réparties sur toutes les classes : c'est-à-dire que pour une maison modeste de classe 0, certaines prédictions donnent la classe 3 ce qui est assez embêtant au vu de l'objectif de l'étude. Au contraire, la régression logistique a des mauvaises prédictions avec les classes voisines à celle réelle : ceci est plus en accord avec notre démarche. Ici, on peut souligner un défaut de notre démarche : deux prix très proches en dollars peuvent appartenir à deux classes différentes du fait de la frontière entre classe.

4.5 Modèle retenu pour la classification

Au vu de la métrique Exactitude, il serait cohérent de sélectionner le modèle Random Forest cependant, dans l'esprit de notre démarche on retient le modèle de régression logistique car il ne commet pas d'erreur de prédictions aberrantes.

5 Conclusion

À travers notre étude, nous avons montré que ce problème pouvait et devait être traité avec de l'apprentissage automatique, et que différents modèles existent pour y répondre. Après avoir travaillé notre jeux de données pour ne garder que les données réellement pertinentes et utilisables par nos IA, nous avons testé et comparé ces différents modèles, pour finalement identifier le Gradient Boosting comme le plus performant sur notre jeux de données. Nous avons débuté par une phase de recherche sur tous ces différents modèles, pour savoir lesquels étaient les plus adaptés à notre cas, mais aussi sur nos données et les moyens disponibles pour les nettoyer. Cette phase de nettoyage des données a sans doute été la partie la plus complexe du travail, et nous avons alors compris à quel point la qualité des données influe sur les résultats. Que ce soit par la modification de l'ordre des opérations, l'ajout ou la suppression de colonnes, l'encodage des valeurs catégorielles, nous avons réalisé de nombreux tests avant d'obtenir ces résultats.

5.1 Pistes d'améliorations

Même si les résultats obtenus sont intéressants, il reste de la place pour l'amélioration. Certains modèles tels la forêt aléatoire sont plus performants sur de plus grandes bases de données, et l'on peut remarquer sur ce graphique que plus on utilise de données, plus le score de ces trois modèles augmente (cf. Figure 4).

Il aurait sans doute aussi été intéressant de plus manipuler les caractéristiques, en essayant éventuellement de supprimer certaines qui ont été gardées, pour voir si elles ne trompent pas nos modèles, et au contraire tenter de créer de nouvelles colonnes pour mettre en valeur de nouvelles informations à partir de celles dont on dispose.

Même si nous avons testé plusieurs modèles aux fonctionnements différents, il reste d'autres modèles que nous n'avons pas utilisés, et qui aurait pu l'être dans ce cas-là, notamment le classificateur Bayésien naÏf, le CNN ou encore le perceptron.

Lors de l'évaluation des données, nous avons opté pour certaines métriques, mais d'autres telles que le rappel et la précision existent et auraient elles aussi été intéressantes à calculer dans ce contexte. La division du jeu de données entre jeux d'entraînement et jeux de test (80-20) peut aussi être remise en question, et nous pourrions étudier l'impact d'une validation croisée k-Fold. La répartition entre les différentes classes n'étant pas homogène sur ce jeux de données, cette dernière méthode pourrait apporter une réelle valeur à un futur travail.

Table des figures

Liste des tableaux