Analyse exploratoire des données

Informations générales

Le jeu de données utilisé, provenant de la compétition Kaggle *House Prices – Ames, Iowa*, comporte **1 460 observations** et **81 variables** (dont la variable cible `SalePrice`). Cette dimension permet d'entraîner des modèles suffisamment complexes, tout en restant exploitable sans infrastructure de calcul particulière.

Les 80 variables explicatives sont réparties en trois grandes catégories, identifiées lors de l'exploration initiale. Les **variables numériques continues** sont au nombre de 34 ; elles décrivent principalement des surfaces (en pieds carrés), des dates, des comptages (nombre de salles de bain, cheminées, etc.) et des dimensions du terrain. Les **variables catégorielles ordinales** représentent 18 colonnes : elles encodent des niveaux de qualité ou de condition évalués sur une échelle (par exemple la qualité générale `OverallQual`, la qualité de la cuisine `KitchenQual`, ou la finition du garage `GarageFinish`). Les **variables catégorielles nominales** constituent les 27 colonnes restantes ; elles décrivent des caractéristiques sans ordre intrinsèque, comme le type de toiture `RoofStyle`, le quartier `Neighborhood` ou le type de fondation `Foundation`.

Une première observation majeure concerne les **valeurs manquantes** : plusieurs variables présentent un taux d'absence très élevé. `PoolQC` manque à 99,5%, `MiscFeature` à 96,3%, `Alley` à 93,8% et `Fence` à 80,8%. Dans le contexte immobilier d'Ames, Iowa, ces absences ne sont pas aléatoires : elles signifient souvent que le bien ne possède pas l'équipement considéré (pas de piscine, pas d'allée, etc.). Cette observation oriente dès la phase exploratoire la stratégie de traitement qui sera adoptée plus tard.

Compréhension de la variable cible

La variable cible `SalePrice` représente le prix de vente final de chaque bien. Ses statistiques descriptives révèlent un prix moyen de \$180 921 avec un écart-type de \$79 443, un minimum de \$34 900 et un maximum de \$755 000. La médiane, à \$163 000, est notablement inférieure à la moyenne, ce qui constitue un premier indice d'asymétrie.

La distribution de `SalePrice` présente en effet une **asymétrie positive et élevée**, confirmée par un coefficient d'asymétrie (skewness) de **1,88**. Cela signifie que la queue de la distribution s'étire vers les prix élevés : la majorité des maisons sont vendues dans une fourchette modérée, mais un nombre limité de biens à prix très élevés tire la moyenne vers le haut. Le coefficient d'applatissement (kurtosis) vaut **6,54**, ce qui indique une concentration forte des valeurs autour du centre avec des valeurs aberrantes en queue.

Cette distribution est cohérente avec la réalité : la plupart des gens ont des revenus moyens et achètent ainsi des maisons de gamme moyenne, tandis que les propriétés de prestige forment un segment minoritaire qui influence fortement les statistiques centrales.

Pour corriger cette asymétrie, une transformation logarithmique a été appliquée à la variable cible via `log(1 + SalePrice)`. Après cette transformation, le coefficient d'asymétrie chute à **0,12** et le kurtosis à **0,81**, rendant la distribution beaucoup plus proche d'une loi normale. Cette transformation sera utilisée systématiquement lors de l'entraînement des modèles, puis inversée pour retrouver les prix en dollars lors de l'évaluation.

Analyse des variables numériques

Statistiques simples

Une analyse descriptive détaillée a été conduite sur les 34 variables numériques, complétée par le calcul du coefficient de variation, des coefficients d'asymétrie et d'applatissement, ainsi que du pourcentage de valeurs manquantes pour chaque variable.

Interprétation des statistiques descriptives

Plusieurs variables numériques présentent des distributions très asymétriques. `LotArea` (surface du terrain) possède un skewness de 12,2, ce qui reflète l'existence de quelques terrains extrêmement grands par rapport à la grande majorité. `LotFrontage` (largeur du front de terrain) montre une asymétrie de 2,16 avec un coefficient de variation de 0,35, indiquant une dispersion modérée. Les variables temporelles comme `YearBuilt` sont en revanche très concentrées (skewness de $-0{,}61$, coefficient de variation de 0,015), ce qui signifie que la plupart des maisons du jeu de données ont été construites dans une période relativement resserrée autour de 1971.

Corrélation avec SalePrice

Le calcul de la corrélation de Pearson entre chaque variable numérique et `SalePrice` permet d'identifier les facteurs les plus déterminants du prix. Les résultats montrent une claire hiérarchie :

Variable	**Corrélation avec SalePrice**
GrLivArea (surface habitable)	0,709
GarageCars (capacité garage)	0,640
GarageArea (surface garage)	0,623
TotalBsmtSF (surface sous-sol)	0,614
1stFlrSF (surface 1er étage)	0,606
FullBath (salles de bain complètes)	0,561
TotRmsAbvGrd (nombre de pièces)	0,534
YearBuilt (année de construction)	0,523
YearRemodAdd (année de rénovation)	0,507
GarageYrBlt (année construction garage)	0,486

Analyse des variables les plus corrélées

Interprétation économique des corrélations observées

Les corrélations observées semblent cohérentes. La variable `GrLivArea`, qui mesure la surface habitable au-dessus du sol, est la plus corrélée avec le prix (0,709) : dans un marché résidentiel, la surface utilisable est un facteur prix unitaire très direct. Le garage, tant par sa capacité en véhicules que par sa surface, contribue aussi à la valeur du bien, ce qui suggère une importance de l'automobile dans la vie quotidienne à Ames. Les surfaces du sous-sol et du premier étage participent pour leur part à la surface totale du bien.

Les variables temporelles — année de construction et année de rénovation — montrent des corrélations modérées mais significatives (0,52 et 0,51 respectivement). Une maison récente ou récemment rénovée est bien perçue comme plus attractive par le marché. À

l'inverse, les variables comme `MoSold` (mois de vente, corr. 0,05) ou `YrSold` (année de vente, corr. −0,03) n'ont quasiment aucune influence linéaire sur le prix, ce qui suggère que le marché d'Ames a été relativement stable sur la période observée.

Analyse des variables catégorielles ordinales

Les 18 variables catégorielles ordinales encodent des évaluations qualitatives des maisons, organisées selon des niveaux explicitement ordonnés. Les variables de qualité et de condition (comme `ExterQual`, `BsmtQual`, `KitchenQual`, `GarageQual`) suivent le même schéma d'évaluation : *Po* (Poor), *Fa* (Fair), *TA* (Typical/Average), *Gd* (Good), *Ex* (Excellent). D'autres variables comme `BsmtExposure` ou `GarageFinish` possèdent leurs propres échelles.

L'analyse par boîtes (boxplots) de `SalePrice` en fonction de chacune de ces variables montre un motif clair : dans la grande majorité des cas, l'augmentation du niveau de qualité entraîne une hausse médiane du prix. Par exemple, les maisons évaluées *Ex* en qualité extérieure (`ExterQual`) ou en qualité de cuisine (`KitchenQual`) affichent des prix médians nettement supérieurs à celles notées *TA*. Cette relation s'interprète directement sur le marché : les acheteurs sont prêts à payer davantage pour des finitions, des équipements et des conditions de meilleure qualité.

La variable `OverallQual`, qui représente la qualité générale de la maison, exerce une influence particulièrement forte : les prix médians augmentent presque strictement avec chaque niveau supplémentaire de qualité, confirmant son rôle central dans la fixation du prix.

Analyse des variables catégorielles nominales

Les 27 variables catégorielles nominales décrivent des caractéristiques qualitatives sans ordre naturel. Pour chacune, une analyse détaillée a été conduite, portant sur la distribution des modalités et sur le prix moyen associé à chaque catégorie.

Plusieurs variables présentent une distribution très déséquilibrée. `CentralAir` (climatisation centrale) ne possède que deux modalités : 93,5% des maisons en disposent (modalité *Y*). Néanmoins, la différence de prix entre les deux catégories est considérable : les maisons sans climatisation centrale ont un prix moyen de \$105 264, contre \$186 187 pour celles qui en sont équipées. Ce décalage s'explique par l'importance du confort climatique dans une région aux hivers rigoureux.

La variable `Neighborhood` (quartier) est l'une des plus discriminantes en termes de prix. Elle possède 25 modalités avec des prix moyens très variés. Les quartiers comme *NoRidge* ou *NridgHt* sont associés à des prix nettement supérieurs à la moyenne, tandis que des quartiers comme *IDOTRR* ou *BrkSide* correspondent à des segments plus accessibles. Le quartier condensé à lui seul une grande partie de l'information géographique et sociale du bien.

`BldgType` (type de bâtiment) montre que les maisons individuelles (*1Fam*) constituent 83,6% du jeu de données avec un prix moyen de \$185 764, tandis que les copropriétés attenantes (*TwnhsE*) sont proches à \$181 959. Les duplex et maisons mitoyennes présentent des prix moyens sensiblement plus bas, autour de \$128 000 à \$136 000.

Analyse temporelle

L'analyse temporelle a été enrichie par la création de trois variables dérivées : `HouseAge` (âge de la maison au moment de la vente), `RemodAge` (années écoulées depuis la dernière rénovation), et `IsNew` (indicateur binaire si la maison a été vendue l'année de sa construction).

Les statistiques montrent que l'âge moyen des maisons est de **36,5 ans** avec une médiane de 35 ans, indiquant un stock de logements relativement homogène en termes d'ancienneté. Parmi les 1 460 biens, **64 maisons** (4,4%) ont été vendues l'année de leur construction, ce qui témoigne d'un marché de construction neuve actif.

L'évolution temporelle des prix entre 2006 et 2010 révèle une relative stabilité du marché. Les prix moyens oscillent entre \$177 000 et \$186 000 sans tendance ascendante ou descendante marquée sur cette période de cinq ans. Les prix médians sont quant à eux légèrement décroissants, passant de \$164 000 en 2006 à \$155 000 en 2010, un mouvement cohérent avec le ralentissement économique général observé à cette époque aux États-Unis.

L'analyse de la saisonnalité montre que les prix médians varient selon le mois de vente, avec un pic en été (juin–juillet) et des prix plus bas en hiver. Ce phénomène est classique dans l'immobilier : les ventes d'été bénéficient de conditions plus favorables à la visite et à la décision d'achat.

Prétraitement

Séparation des données

Le jeu de données a été divisé en un ensemble d'entraînement et un ensemble de test selon un rapport de **80/20**, soit respectivement **1 168 observations** pour l'entraînement et **292 observations** pour le test. Cette répartition est standard dans le domaine : elle garantit un volume d'entraînement suffisant pour que les modèles apprennent les patterns du marché immobilier, tout en réservant un échantillon de taille significative pour évaluer la capacité de généralisation des modèles sur des données inobservées lors de l'apprentissage.

Il est crucial que tous les calculs de statistiques (moyennes, modes, médians par groupe) nécessaires à l'imputation et à la transformation soient appris exclusivement sur l'ensemble d'entraînement, puis appliqués indépendamment sur l'ensemble de test. Cette discipline évite toute fuite d'information (*data leakage*) qui fausserait l'évaluation de la performance.

Traitement des valeurs manquantes

Le traitement des valeurs manquantes a été articulé autour d'une logique économique claire, fondée sur les observations de la phase exploratoire. Les variables manquantes ont été réparties en quatre groupes selon leur signification dans le contexte immobilier.

Les **15 variables catégorielles** présentant des absences d'équipement (comme `PoolQC`, `MiscFeature`, `Alley`, `Fence`, `FireplaceQu`, `GarageType`, `GarageFinish`, etc.) ont été imputées par la valeur *"None"*, qui signifie littéralement l'absence de l'équipement concerné. Les **10 variables numériques** correspondantes (comme `GarageYrBlt`, `GarageArea`, `GarageCars`, `MasVnrArea`, etc.) ont été imputées par la valeur **0**, car si un équipement n'existe pas, sa surface ou sa quantité est nulle.

La variable `LotFrontage` a fait l'objet d'un traitement particulier : elle a été imputée par

la **médiane calculée par quartier** (`Neighborhood`), car la largeur du front de terrain est fortement influencée par la géographie locale du quartier. Enfin, **8 variables catégorielles** résiduelle (comme `MSZoning`, `Electrical`, `SaleType`) dont le taux d'absence est très faible ont été imputées par leur **mode**.

Rôle de la classe dédiée au traitement

L'ensemble de cette logique a été encapsulée dans une classe Python héritant de `BaseEstimator` et `TransformerMixin` de scikit-learn, intitulée `MissingValuesHandler`. Cette architecture permet d'intégrer l'imputation dans un `Pipeline` scikit-learn, ce qui garantit que les paramètres d'imputation (médiane par quartier, modes) sont calculés uniquement sur les données d'entraînement et appliqués de manière cohérente sur les données de test, sans risque de fuite d'information.

Correction des anomalies et incohérences

Une vérification a été effectuée pour détecter les incohérences logiques dans les données. Le principal cas identifié concerne la variable `GarageYrBlt` : certaines valeurs indiquaient une année de construction du garage postérieure à celle de la maison elle-même, ce qui est physiquement impossible dans les cas concernés. Ces valeurs ont été corrigées en remplaçant `GarageYrBlt` par `YearBuilt` pour les observations concernées.

Cette correction est nécessaire car une telle incohérences, si elle n'est pas traitée, pourrait introduire du bruit dans les features temporelles dérivées (comme l'âge du garage) et affecter la qualité des prédictions.

Création de nouvelles features

La phase d'ingénierie des features a consisté à créer des variables plus informatives à partir des variables brutes, en mobilisant la connaissance du domaine immobilier.

Des **features de surface agrégées** ont été créées : `TotalSF` (surface totale du bien, incluant le sous-sol et les étages), `TotalSF_AboveGround` (surface hors-sol), `TotalPorchSF` (surface totale des porches et terrasses). Ces agrégations condensent l'information dispersée dans plusieurs colonnes en une seule variable plus directement lisible par les modèles.

Des **indicateurs binaires** ont été introduits pour signaler la présence ou l'absence d'équipements clés : `HasBasement`, `Has2ndFloor`, `HasPorch`, `HasDeck`, `HasPool`, `HasGarage`, `HasFireplace`. Ces features permettent aux modèles de capter l'effet discontinu que peut avoir la mere présence ou l'absence d'un équipement sur le prix, au-delà de sa surface.

Des **features temporelles** ont été dérivées : `HouseAge` (âge de la maison), `RemodAge` (ancienneté de la dernière rénovation), `IsNew` et `HasBeenRemod` (indicateurs binaires), ainsi que `HouseAgeBin` qui découpe l'âge en catégories (Nouveau, Récent, Modéré, Ancien, Très ancien). Une variable `FireplaceScore` combinant le nombre de cheminées et leur qualité a également été introduite pour capturer l'effet conjoint de ces deux dimensions.

Les variables brutes utilisées pour la construction de ces nouvelles features (comme `1stFlrSF`, `2ndFlrSF`, `YearBuilt`, etc.) ont été supprimées pour éviter la redondance.

Transformation des variables ordinales

Les variables ordinales ont été converties en valeurs numériques à l'aide de mappings manuels, respectant l'ordre inhérent des catégories. Les variables de qualité (`ExterQual`, `BsmtQual`, `KitchenQual`, etc.) ont été encodées sur une échelle de 0 à 5, correspondant respectivement à *None*, *Po*, *Fa*, *TA*, *Gd*, *Ex*. D'autres variables ont reçu des mappings spécifiques : `GarageFinish` sur une échelle de 0 à 3, `Functional` sur une échelle de 0 à 7 (du plus degradé *Sal* au plus fonctionnel *Typ*), et les variables géographiques comme `LotShape` et `LandContour` selon leur propre ordre logique.

Cette transformation est justifiée car les algorithmes numériques ne peuvent exploiter directement les labels textuels. Le mapping manuel, plutôt qu'un encodage automatique comme `LabelEncoder`, permet de garantir que l'ordre imposé est cohérent avec la réalité : une maison de qualité *Ex* reçoit bien un score supérieur à celle de qualité *Gd*.

Encodage des variables catégorielles nominales

Les variables catégorielles nominales restantes (24 colonnes après les transformations précédentes, incluant `Neighborhood`, `MSZoning`, `Exterior1st`, etc.) ont été encodées par **One-Hot Encoding** via la classe `OneHotEncoder` de scikit-learn, intégrée dans un `ColumnTransformer`.

Le One-Hot Encoding a été retenu car ces variables ne possèdent aucun ordre naturel : un encodage numérique simple (0, 1, 2...) introduirait une relation d'ordre artificielle entre les catégories, ce qui pourrait induire les modèles linéaires en erreur. Le paramètre `handle_unknown='ignore'` a été activé pour gérer éventuellement les catégories rares apparaissant uniquement dans le jeu de test.

Transformation des variables numériques (skewness)

Les variables numériques présentant une forte asymétrie (coefficient d'asymétrie supérieur à 0,75 en valeur absolue) ont été identifiées. Une transformation `log(1 + x)` a été planifiée pour ces variables, qui compresse les queues de distribution et rapproche les données d'une forme plus symétrique. Cette transformation est bénéfique pour les modèles linéaires dont les hypothèses reposent sur une certaine régularité de la distribution des données.

Standardisation

Une standardisation des variables numériques a été effectuée via un `StandardScaler`, intégré dans le Pipeline au niveau du `ColumnTransformer`. Cette transformation recentre chaque variable autour de zéro et lui donne une variance unitaire.

La standardisation est indispensable pour les modèles sensibles à l'échelle des variables, comme les modèles linéaires régularisés (Ridge, Lasso, ElasticNet) et les algorithmes basés sur des distances (KNN, SVR). Sans cette étape, une variable comme `LotArea` (exprimée en pieds carrés, donc en milliers) dominerait artificiellement les autres variables dans l'optimisation du modèle.

Sélection de features

Une analyse de l'information mutuelle entre chaque feature et la variable cible `SalePrice` a été conduite sur l'ensemble d'entraînement. L'information mutuelle mesure la dépendance

statistique entre deux variables de façon non-linéaire, ce qui la rend plus adaptée que la corrélation de Pearson pour des modèles complexes comme les arbres de décision.

Cette analyse permettait d'identifier les features apportant le plus d'information explicative et, le cas échéant, d'éliminer celles dont la contribution est négligeable. Elle contribue à la réduction de la dimensionnalité et à l'allégement du charge computationnelle lors de l'entraînement.

Débogage avant la fin du prétraitement

Une classe `DebugTransformer` a été insérée dans le Pipeline pour vérifier à chaque étape l'état des données : nombre de dimensions, présence de valeurs manquantes, types des colonnes. Cette approche permet de détecter rapidement les erreurs de transformation et de garantir la cohérence des données avant d'alimenter les modèles.

Vérifications finales avant modélisation

À la fin du Pipeline de prétraitement, les données ont été vérifiées : les dimensions finales s'atteignent **85 variables** après transformation (avant One-Hot Encoding), puis **216 colonnes** après encodage complet. L'absence de valeurs manquantes a été confirmée pour les deux ensembles, entraînement et test. Le jeu de données est à ce stade entièrement numérique et prêt à être ingurgité par les algorithmes de modélisation.

Modélisation

Douze modèles de régression ont été testés en phase baseline, sans optimisation préalable des hyperparamètres, sur les données transformées par le Pipeline. La variable cible a été préalablement transformée par `log(1 + SalePrice)` pour l'entraînement, puis inversée pour l'évaluation des métriques en dollars.

Les modèles testés se répartissent en trois familles. Les **modèles linéaires** – Ridge, Lasso, ElasticNet, BayesianRidge et HuberRegressor – apprendent une relation linéaire entre les features et le prix, avec différents mécanismes de régularisation pour éviter le surapprentissage. Les **modèles basés sur des arbres** – DecisionTree, RandomForest, ExtraTrees, GradientBoosting et AdaBoost – découvent des partitions successives de l'espace des features pour prédire le prix, sans hypothèse de linéarité. Les **autres modèles** – KNN et SVR – utilisent respectivement la moyenne des voisins les plus proches et une optimisation dans un espace transformé.

Les résultats de la phase baseline sont présentés ci-dessous, triés par RMSE sur l'ensemble de test :

Modèle	RMSE Test ($)	R^2 Test	Temps (s)
HuberRegressor	21 211	0,935	5,46
Ridge	21 555	0,933	0,10
BayesianRidge	23 052	0,924	0,06
ExtraTrees	28 956	0,879	2,67
GradientBoosting	29 667	0,873	1,12
RandomForest	31 349	0,859	2,90
AdaBoost	35 932	0,814	1,20
SVR	36 514	0,808	0,23
KNN	37 240	0,800	0,04
DecisionTree	43 241	0,731	0,06
Lasso	84 310	−0,023	0,03
ElasticNet	84 310	−0,023	0,01

Le modèle **HuberRegressor** s'est placé en tête avec un RMSE de $21 211 et un R^2 de 0,935, ce qui signifie qu'il explique environ 93,5% de la variance du prix. Ce résultat remarquable pour un modèle linéaire s'explique par la transformation logarithmique de la cible qui a rendu la relation plus linéaire, et par le Pipeline de prétraitement qui a produit des features très pertinentes. Le HuberRegressor se distingue particulièrement car il utilise une fonction de perte robuste aux valeurs aberrantes : il minimise l'erreur quadratique pour les résidus proches de zéro, puis passe à une fonction linéaire pour les résidus éloignés, ce qui lui permet de gérer sans difficulté les quelques biens atypiques du marché.

Le modèle **Ridge** se place juste derrière avec un RMSE de $21 555 et un R^2 de 0,933, suivi de **BayesianRidge** ($23 052, $R^2 = 0{,}924$). Les trois premiers modèles sont tous linéaires, ce qui confirme l'efficacité de la transformation log et de l'ingénierie de features. Les modèles basés sur des arbres (ExtraTrees, GradientBoosting, RandomForest) ont affiché des performances solides, avec des R^2 entre 0,86 et 0,88. En revanche, Lasso et ElasticNet ont échoué à produire une prédiction utile (R^2 négatif), ce qui indique que leur paramètre de régularisation par défaut (alpha = 1,0) était trop fort et avait éliminé trop de signal.

Optimisation des hyperparamètres

Intérêt de l'optimisation

Les résultats baseline sont obtenus avec les hyperparamètres par défaut de chaque modèle. Or ces valeurs par défaut ne sont pas nécessairement adaptées au jeu de données spécifique utilisé. L'optimisation des hyperparamètres vise à trouver la configuration de chaque modèle qui maximise ses performances sur les données disponibles, en utilisant une validation croisée sur l'ensemble d'entraînement pour éviter le surapprentissage.

Les quatre meilleurs modèles de la phase baseline – HuberRegressor, Ridge, BayesianRidge et ExtraTrees – ont été retenus pour cette phase d'optimisation, via une recherche aléatoire (`RandomizedSearchCV`) avec une validation croisée à 5 plis et 50 itérations.

Lecture des résultats obtenus

Le modèle **HuberRegressor** a été optimisé sur ses deux hyperparamètres principaux : `epsilon`, le seuil de transition entre la perte quadratique et la perte linéaire, et `alpha`, le paramètre de régularisation L2. Les meilleurs paramètres trouvés sont `epsilon = 1,35` et `alpha = 10,0`. Après optimisation, le RMSE descend à \$19 731 avec un R^2 de 0,944, représentant un gain de \$1 480 par rapport à la baseline. C'est le seul modèle parmi les quatre à montrer une amélioration significative après optimisation.

Le modèle **BayesianRidge** a été optimisé sur ses quatre hyperparamètres de régularisation (`alpha_1`, `alpha_2`, `lambda_1`, `lambda_2`). Le RMSE atteint \$23 053 avec un R^2 de 0,924, identique à la baseline : le modèle était déjà à son point de convergence optimal avec ses paramètres par défaut.

Le modèle **Ridge** a été optimisé sur son paramètre `alpha`, explorant une plage de valeurs sur une échelle logarithmique de 10^{-3} à 10^3. La valeur optimale retenue est `alpha = 138,9`. Le RMSE obtient \$24 716 avec un R^2 de 0,912, ce qui représente une légère déterioration par rapport à la baseline (\$21 555). Ce résultat s'explique par le fait que le meilleur alpha selon le score de validation croisée n'est pas nécessairement celui qui minimise l'erreur sur ce jeu de test particulier.

Le modèle **ExtraTrees** a été optimisé sur le nombre d'arbres et la profondeur maximale. La meilleure configuration utilise 600 arbres sans limitation de profondeur. Le RMSE obtient \$28 239 avec un R^2 de 0,885, un gain modeste de \$716 par rapport à la baseline.

Le modèle **HuberRegressor optimisé** ressort donc comme le meilleur modèle global, avec le RMSE le plus bas (\$19 731) et le R^2 le plus élevé (0,944) parmi tous les modèles testés.

Évaluation du modèle final

Interprétation des métriques utilisées

Trois métriques principales ont été utilisées pour évaluer les modèles. Le **RMSE** (Root Mean Squared Error) mesure l'écart quadratique moyen entre les prix prédit et les prix réels, exprimé en dollars. Une valeur faible indique une meilleure précision ; la racine carrée garantit que la métrique est exprimée dans la même unité que la variable cible. Le **MAE** (Mean Absolute Error) mesure quant à lui l'écart moyen en valeur absolue, moins sensible aux grandes erreurs que le RMSE. Le $\mathbf{R^2}$ (coefficient de détermination) exprime la proportion de la variance du prix qui est expliquée par le modèle : une valeur proche de 1 indique un bon ajustement, une valeur proche de 0 indique que le modèle ne fait pas mieux qu'estimer la moyenne.

Lecture des performances du modèle retenu

Le modèle retenu est le **HuberRegressor** après optimisation (`epsilon = 1,35`, `alpha = 10,0`). Sur l'ensemble de test composé de 292 observations non utilisées lors de l'entraînement, il atteint un RMSE de \$19 731 et un R^2 de 0,944. Ce résultat signifie que, en moyenne, les prédictions du modèle s'écartent du prix réel d'environ 19 700 dollars, et que le modèle explique plus de 94% de la variabilité des prix dans le marché d'Ames.

L'analyse visuelle des prédictions confirme ces résultats chiffrés. Le nuage de points des

prix réels par rapport aux prix prédits montre une relation très linéaire, avec la quasi-totalité des points concentrés autour de la droite de régression parfaite. Les résidus sont distribués de façon quasi-symétrique autour de zéro, sans signe de biais systématique, ce qui valide la qualité du modèle. Quelques outliers subsistent pour les biens de très haut prix, ce qui est inévitable vu le faible nombre d'observations dans ce segment.

Pour mettre cette performance en perspective, une erreur moyenne de \$19 731 sur un prix médian de \$163 000 représente un écart relatif d'environ 12%. Dans le domaine de l'estimation immobilière automatisée, un tel niveau de précision est très satisfaisant et fournit une estimation utile comme point de départ pour une évaluation professionnelle.

Sauvegarde du modèle et conclusion

Rôle de la sauvegarde du modèle

La sauvegarde du modèle entraîné a été effectuée via la bibliothèque `joblib`, qui a persisté l'ensemble des paramètres appris – les coefficients du modèle, les statistiques d'imputation, les mappings ordinaux, les paramètres du scaler – sous forme d'un fichier `best_model.pkl`. Cette sauvegarde permet que le Pipeline complet puisse être rechargé ultérieurement pour effectuer des prédictions sur de nouveaux biens, sans nécessité de ré-entraîner le modèle. En production, cette sauvegarde garantit la reproductibilité et la cohérence des prédictions.

Conclusion

Le projet a abouti à un modèle de prédiction du prix des maisons opérationnel sur le marché d'Ames, Iowa, capable d'expliquer plus de **94%** de la variabilité des prix avec une erreur moyenne d'environ **19 700 dollars**.

La chaîne de traitement repose sur un Pipeline scikit-learn intégré, composé de l'imputation contextuelle des valeurs manquantes, de la correction des anomalies, de l'ingénierie de features (surfaces agrégées, indicateurs binaires, variables temporelles), de l'encodage ordinal et nominal, puis de la standardisation. Le modèle final, un HuberRegressor optimisé sur la transformation logarithmique de la cible, a démontré une capacité de généralisation excellente sur les données de test. Sa robustesse aux valeurs aberrantes, inhérente à sa fonction de perte, l'a placé au premier rang devant les autres modèles linéaires et les ensembles d'arbres.

Pour Laplace Immo, ce modèle représente un outil d'estimation rapide et fiable : il permet de produire une valeur de référence pour chaque bien à partir de ses caractéristiques physiques et de localisation, en quelques millisecondes. Cette estimation peut ensuite servir de base pour les negotiations commerciales, l'orientation des clients vers les biens adaptés à leur budget, ou l'identification des biens sous-évalués sur le marché.