House Prices- Advanced Regression Techniques

Participation à une compétition Kaggle

Par Manon Giraud

Table des matières

Introduction

Contexte

Le site Kaggle propose une grande variété de compétitions pour des Data Scientists, sous la forme de projets similaires à ce que l'on peut trouver dans le monde du travail. Outre les possibles récompenses monétaires, ces compétitions permettent donc de tester ses capacités en matière de Machine Learning, sur des données de qualité. Il est également possible de partager son code avec d'autres utilisateurs, et d'en discuter.

Données

La compétition que j'ai choisie utilise des données concernant des habitations dans la ville de Ames, en Iowa, aux Etats-Unis. Toutes sortes d'informations sur ces maisons sont disponibles, du nombre d'étages à la qualité de la cuisine. Le but est de se servir de ces informations pour prédire le prix de la maison, en dollars. L'erreur utilisée est la racine de l'erreur quadratique moyenne logarithmique.

Les données sont réparties en deux tableaux : un contenant les données d'entraînement, qui inclut donc le prix des habitations, et un contenant les données test, qui ne les inclut bien sûr pas. Les deux tableaux contiennent environ 1450 individus, et 79 variables explicatives.

Ces données peuvent être considérées comme très fiables, et sont similaires à un autre jeu de données fréquemment utilisé : le Boston Housing dataset.

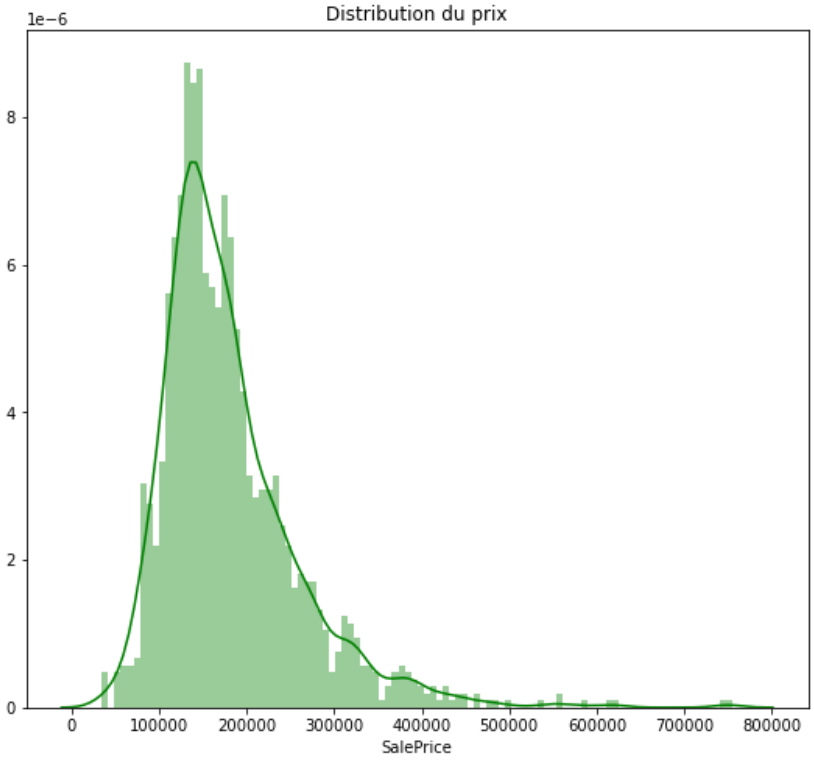
Analyse exploratoire

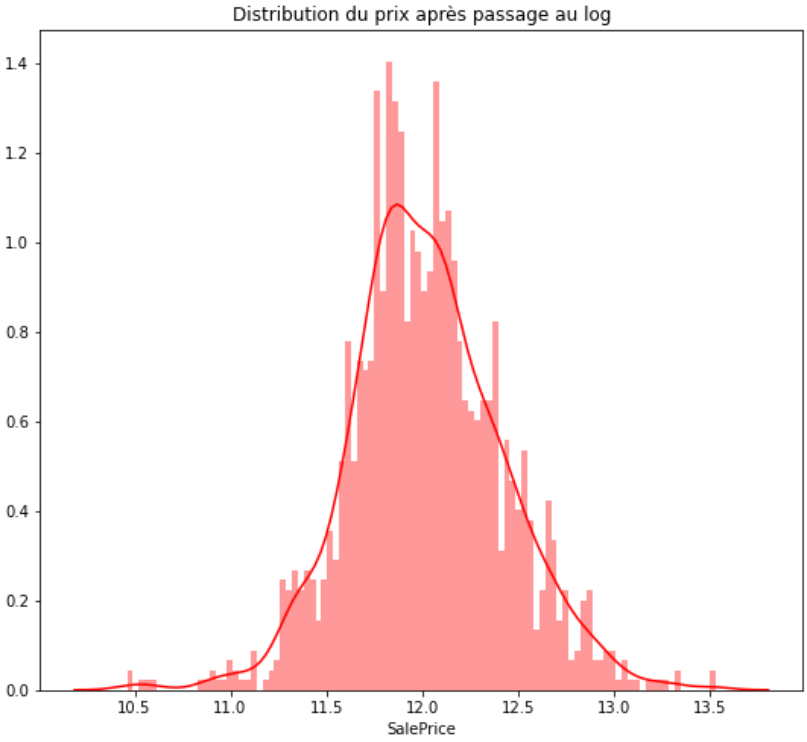
Analyse univariée

Avant toute chose, j'ai examiné la variable d'intérêt, SalePrice. Les statistiques et le graphique montrent que cette variable n'est pas distribuée selon la loi normale : il y a une longue traîne à droite. Cela peut fausser les analyses, étant donné que plusieurs modèles supposent une distribution normale. De plus, même pour les modèles pour lesquels ce n'est pas nécessaire, ces valeurs loin de la moyenne risquent de biaiser le modèle.

Ce problème peut potentiellement être résolu en utilisant le passage au log.

En examinant la distribution du prix après le passage au log, on voit une distribution bien plus proche de la loi normale. De plus, l'erreur sur les données test étant calculée sur le logarithme des prix, nous avons une raison supplémentaire d'utiliser des données transformées.

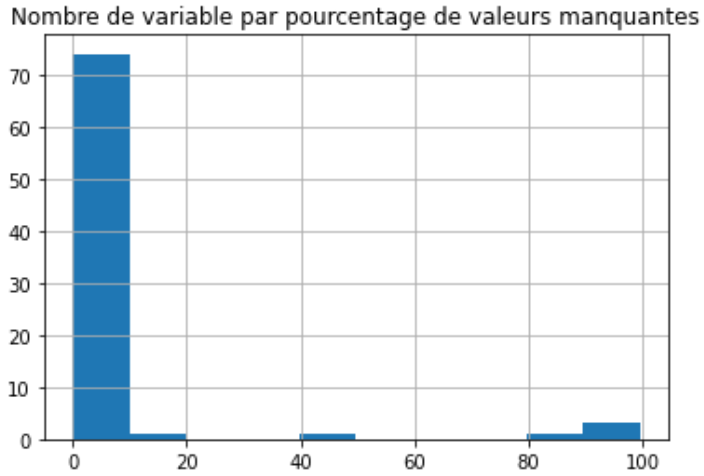




J'ai également examiné la distribution des autres variables quantitatives. Il est révélé que certaines variables, comme par exemple la surface murale vernie, ont en majorité la valeur 0. Il serait donc avisé de les remplacer par des variables booléennes, indiquant si la valeur de la variable d'origine est 0 ou non.

Pour cette analyse, comme pour le graphique de la distribution du prix, je me suis aidée du code provenant de « Detailed exploratory data analysis with python ».

Finalement, j'ai examiné le pourcentage de valeurs manquantes par variable :

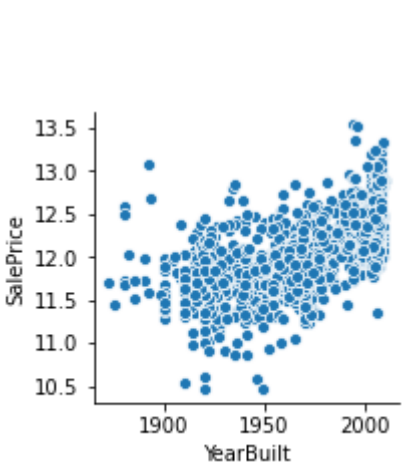


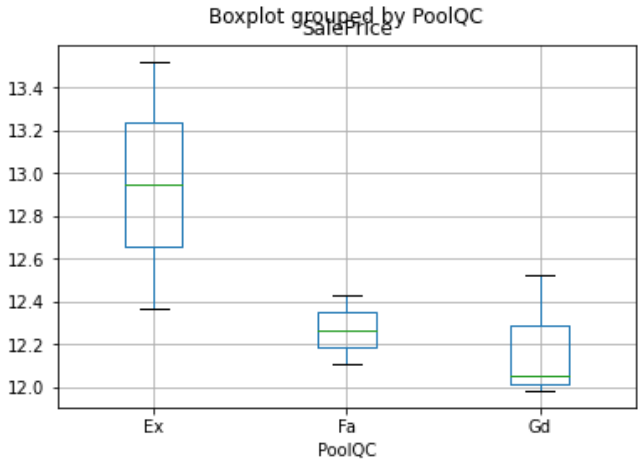
Très peu de variables ont une grande quantité de valeurs manquantes, et en examinant ces variables en question, on s'aperçoit que les valeurs manquantes ne représentent pas une absence d'information, mais indiquent plutôt que la variable n'est pas applicable à l'individu en question : par exemple, la variable indiquant la qualité de la cheminée sera une valeur manquante pour les habitations n'ayant pas de cheminée.

Il n'est donc pas nécessaire de supprimer des variables ayant trop de valeurs manquantes.

Analyse bivariée

Après avoir examiné la distribution des variables, j'ai cherché à regarder leur relation avec la variable d'intérêt, SalePrice. J'ai utilisé pour les variables quantitatives un nuage de point, et pour les variables qualitatives, des boxplot par catégorie :





Les informations révélées par ces graphiques pourront se révéler intéressantes lors de la comparaison avec les variables sélectionnées par le modèle.

Transformation des données

En plus du passage au log de la variable SalePrice et de la transformation en variable booléenne de certaines variables qualitatives, il faut effectuer plusieurs transformations sur les variables. Voyons lesquelles.

Valeurs extrêmes

Il est souvent nécessaire de retirer les valeurs extrêmes, qui peuvent biaiser les modèles. Cependant, je n'ai détecté aucune valeur trop extrême sur le jeu de données. J'ai donc décidé de ne pas supprimer d'individus pour cette raison.

Valeurs manquantes

Comme vu plus haut, certaines variables ont des valeurs manquantes, qui doivent être éliminées. En effet, très peu de modèles sont capables de les prendre en compte.

La méthode de gestion des valeurs manquantes dépend de la variable. Pour toutes les variables qualitatives, elles seront traitées comme une classe supplémentaire.

Dans le cas de LotFrontage, qui représente la longueur de rue reliée à l'habitation, les valeurs manquantes signifient simplement que l'habitation n'a pas un accès direct à la rue, et peuvent donc être substituées par 0.

Pour MasVnrArea, qui représente l'aire d'une surface, les valeurs manquantes ne signifient pas que la surface est absente, étant donné que la variable possède déjà des individus qui prennent la valeur 0. Cela signifie que ces valeurs sont réellement manquantes. Cela ne concerne que 3 individus ; nous pouvons donc simplement les supprimer.

Pour GarageYrBlt, une valeur manquante signifie qu'il n'y a pas de garage. Il est cependant impossible de remplacer ces valeurs par 0, pour des raisons évidentes. Supprimer ces individus n'est pas non plus envisageable : cela représenterait une perte d'information importante. J'ai donc décidé d'utiliser une imputation par la moyenne.

Variables qualitatives

Les modèles statistiques ne peuvent généralement pas prendre en compte les variables qualitatives sans transformation préalable. C'est ce que j'ai donc fait.

Tout d'abord, un examen des effectifs par classe nous montre que plusieurs variables ont des classes contenant trop peu d'individus. Utiliser ces classes dans les modèles n'est donc pas recommandé. J'ai donc regroupé en une classe Other tous les individus appartenant à une classe d'un effectif inférieur à 30, et ce, pour toutes les variables.

Enfin, il faut encoder les variables et classes restantes. Le choix évident est le OneHotEncoding, cependant, cet encodage n'est pas recommandé pour les variables ayant trop de classes. J'ai donc décidé d'utiliser le TargetEncoding, qui remplace une classe par la moyenne de la variable d'intérêt pour les individus appartenant à cette classe, pour ces variables en question. Il s'est avéré que toutes les variables qualitatives sont soit binaires, soit possédant trop de classes pour le OneHotEncoding. Je ne l'ai donc pas utilisé.

Choix du modèle

Conseils d'autres utilisateurs

Par souci d'économie de temps, j'ai décidé de ne pas tester différents modèles et de comparer leurs performances, mais plutôt de me fier aux travaux d'autres Data Scientists, publiés sur la plateforme Kaggle. Le notebook « Comparison Between 5 Regression Algorithm » fait exactement ce que le titre implique, et compare la régression linéaire, les SVR, les forêts aléatoires, la régression ridge et les réseaux de neurones, concluant que les forêts aléatoires sont les modèles les plus performants.

J'ai donc travaillé uniquement sur les forêts aléatoires, en cherchant les meilleurs hyperparamètres.

Optuna et la recherche d'hyperparamètres

Optuna est un outil très utile dans la recherche d'hyperparamètres, utilisé notamment dans « Feature Engineering Ideas and Optuna ». Le fonctionnement de l'algorithme est le suivant :

Les hyperparamètres à faire varier sont définis, et leur type ainsi que la méthode de variation sont choisis. Par exemple, un hyperparamètre peut être un entier, un nombre décimal, ou une option parmi plusieurs. Si le paramètre est numérique, Optuna peut choisir sa valeur complètement au hasard dans l'intervalle sélectionné, ou donner un plus gros poids aux grandes (ou petites) valeurs.

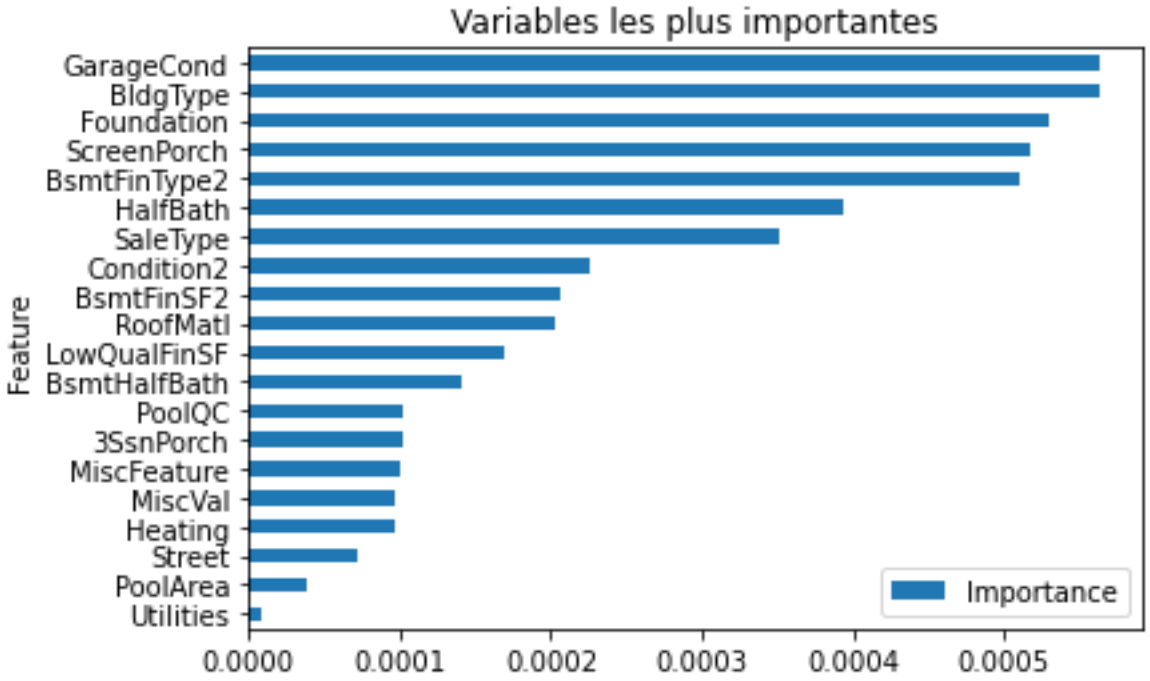
Le modèle, et les hyperparamètres qui ne varieront pas, sont bien sûr également définis.

La méthode de comparaison est ensuite définie. Cela regroupe la statistique utilisée pour la comparaison – dans mon cas la RMSE – et les données utilisées pour le test. Il est possible d'utiliser une validation croisée sur les données utilisées pour l'entraînement, mais pour éviter le surapprentissage, j'ai préféré utiliser des données test séparées.

Enfin, l'algorithme peut optimiser les paramètres. Ils sont choisis au hasard lors du premier passage, mais le choix est ensuite partiellement basé sur les résultats précédents. A la fin de l'exécution, l'algorithme fournit les hyperparamètres les plus performants.

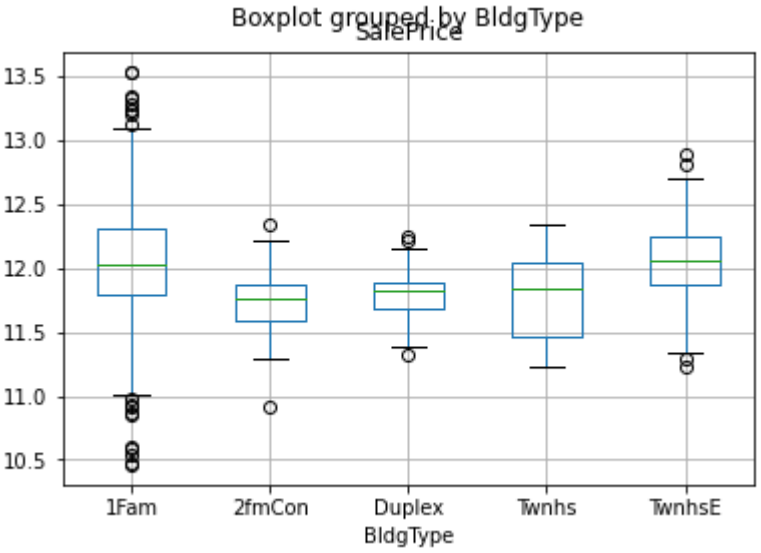
Importance des variables

En plus du score sur les données d'évaluation, que je discuterai plus bas, une méthode d'évaluation du modèle est l'importance des variables. Dans un bon modèle, quelques variables significatives devraient apparaitre, et être cohérentes avec les données.



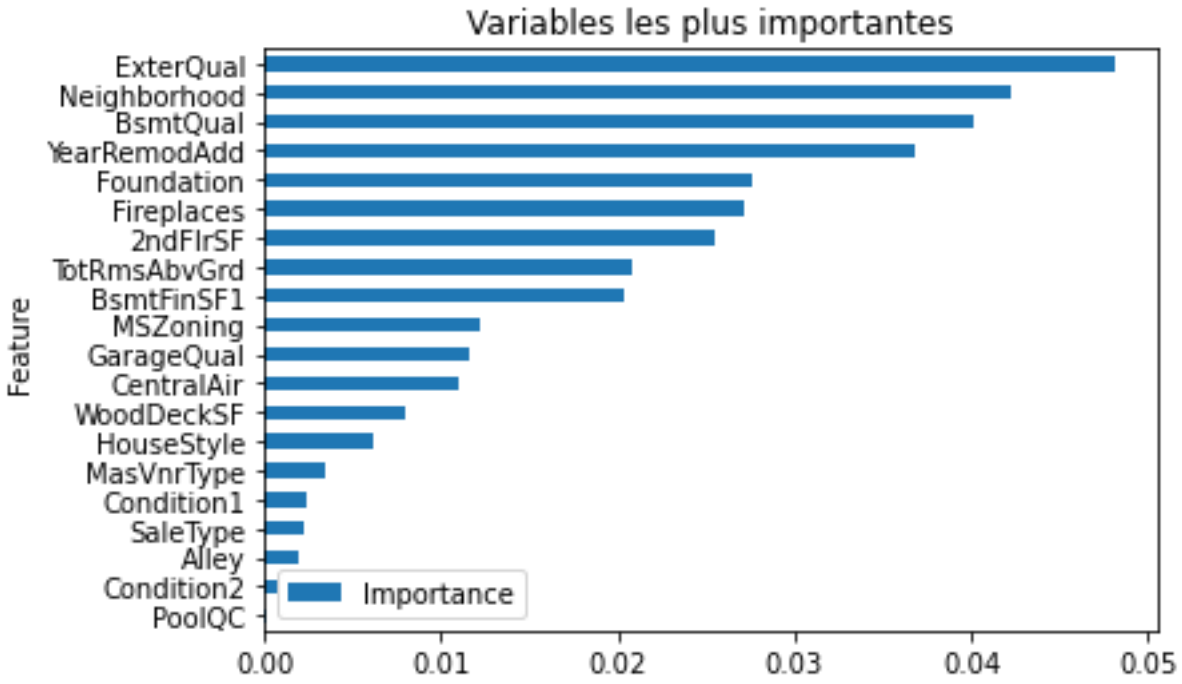
Les variables les plus importantes ne sont pas celles auxquelles on s'attendrait. Leur importance totale, même pour les variables les plus importantes, est de plus très faible, et ne correspond pas à ce à quoi on s'attendrait au vu de l'analyse exploratoire.

Par exemple, BldgType, la deuxième variable la plus importante, ne montre pas de différence significative sur les boxplots de SalePrice par classe de BldgType.



Cette méthode n'est donc pas très utile pour l'interprétation, quelles que soient ses performances.

Une solution se présente : présélectionner les variables en se servant de l'analyse exploratoire. J'ai donc sélectionné les 28 variables qui me semblaient les plus pertinentes, et de nouveau cherché les hyperparamètres optimaux. Ceux-ci se trouvent être très similaires aux hyperparamètres du premier modèle.



Les variables importantes sont plus pertinentes, et ont une importance 100 fois plus grande que pour l'autre modèle. De plus, le score est légèrement meilleur. Cette approche est donc la bonne.

Evaluation et partage

Ayant déjà utilisé les données test séparées au préalable des données d'entraînement pour sélectionner les hyperparamètres, il faut bien sûr utiliser des données complètement nouvelles pour évaluer mes deux modèles. Heureusement, ceci étant une compétition Kaggle, un outil d'évaluation est déjà disponible. Celui-ci calcule la RMSE sur le logarithme de la variable d'intérêt, c'est-à-dire la même statistique que j'ai déjà utilisé pour évaluer mes modèles.

Fait étonnant, le score sur les données d'évaluation est en fait meilleur que sur les données test (0.15163 et 0.14969 contre 0.14345 et 0.14324). C'est bien sûr dû partiellement au hasard, mais cela montre néanmoins qu'il n'y a certainement pas de surapprentissage.

La pré-sélection des variables avant l'utilisation des forêts aléatoires montre à mon sens des résultats très intéressants ; j'ai donc décidé de partager le code comparant ces approches avec la communauté Kaggle.

Conclusion

Les forêts aléatoires sont effectivement efficaces sur ce jeu de donné : le score ainsi obtenu me place au milieu du classement des compétiteurs, et ce, sans avoir fait des recherches longues et poussées pour affiner le modèle.

Mon approche de pré-sélection des variables a donné un résultat qui me satisfait, et qui compense l'un des défauts des forêts aléatoires, soit l'utilisation fréquente de variables peu utiles. C'est pourquoi j'ai décidé de partager ces conclusions potentiellement utiles avec la communauté Kaggle.

A l'avenir, cette approche pourrait peut-être être poussée encore plus loin, en testant différentes sélections de variables, et peut-être même en combinant des variables.

Références

Comparison Between 5 Regression Algorithm. (s.d.). Récupéré sur https://www.kaggle.com/l33tc0d3r/comparison-between-5-regression-algorithm

Detailed exploratory data analysis with python. (s.d.). Récupéré sur https://www.kaggle.com/ekami66/detailed-exploratory-data-analysis-with-python

Feature Engineering Ideas and Optuna. (s.d.). Récupéré sur https://www.kaggle.com/cathylren/feature-engineering-ideas-and-optuna