Université de Montréal

Rapport de projet

Réalisé par
Sandrine Bédard (20139790)
Simon Chasles (20123806)
Robin Legault (20129064)

Dans le cadre du cours
Fondements théoriques en science des données (STT 3795)

Travail présenté à
Guy Wolf

Département de mathématiques et statistique
Faculté des arts et des sciences
Université de Montréal

30 avril 2021

Table des matières

1 Introduction

Avec l'arrivée de la pandémie actuelle, nombreux sont les gens qui ont réalisé l'importance d'être bien chez soi, et qui ont décidé de déménager pour vivre dans une maison qui correspond davantage à leurs attentes. Lorsque ce processus est entamé, il est important de se bâtir un budget monétaire à respecter, lequel oriente et restreint la recherche pour la demeure idéale. On peut alors se pencher sur les critères qui déterminent le prix de vente des maisons. À prime abord, il semble naturel de considérer certaines variables importantes comme la superficie de la propriété ou l'année de construction. Toutefois, même des variables plus subtiles comme la proximité de la maison à une voie ferrée ainsi que la présence d'une légère inclinaison sur le terrain peuvent avoir une influence sur le prix de vente. En considérant de plus en plus de telles variables explicatives, le phénomène devient rapidement trop complexe pour être traité sans avoir recours à une modélisation formelle. C'est pourquoi nous faisons appel à la science des données, laquelle peut nous aider à saisir et comprendre les mécanismes qui sont à l'oeuvre dans le contexte de la vente de maisons.

Dans ce projet, nous tentons de construire un modèle capable de prédire le prix de vente de maisons à partir de données sur ces dernières. En particulier, nous utilisons des méthodes d'apprentissage automatique que nous appliquons au jeu de données disponible sur **ce lien**. Il s'agit d'une compétition *Kaggle* dans laquelle on retrouve au total 79 variables explicatives concernant des centaines de maisons situées à Ames en Iowa qui se sont vendues entre 2006 et 2010. L'idée principale de ce projet est d'utiliser une multitude de méthodes de régression afin de bâtir une liste de prédictions, laquelle pourra dans un second temps être traîtée comme un nouvel ensemble d'attributs utilisables pour cette même tâche de régression. Pour chercher à produire la meilleure prédiction possible, nous utiliserons ainsi le *stacking*, cette approche ensembliste consistant à empiler des modèles de régression en utilisant successivement les sorties d'une couche de modèles comme attributs de la couche subséquente. Dans ce rapport, nous présentons une description du jeu de données utilisé ainsi que le prétraitement que nous y avons appliqué. On énonce les différents algorithmes exploités, pour poursuivre avec une présentation des processus d'optimisation effectués. On termine avec les principaux résultats obtenus, suivis d'une discussion sur les conclusions qu'on peut établir.

2 Objectifs

Un des objectifs principaux de ce projet est de se familiariser avec une gamme de méthodes de prétraitement de données et d'algorithmes d'apprentissage automatique applicables au problème de régression. Nous voulons nous servir d'un contexte particulier afin d'appliquer ces méthodes sur un grand jeu de données. Nous désirons notamment évaluer la qualité des prédictions d'une multitude de modèles appliqués à ce jeu données (par exemple, la régression *Ridge*, la régression par machines à vecteurs de support ou la régression par arbres de décisions). Nous dénotons ces modèles par les modèles de **premier niveau**. Également, nous aimerions comprendre quels sont les ingrédients nécessaires au bon fonctionnement d'un modèle ensembliste qui bâtit de nouvelles prédictions à

partir des prédictions des modèles de premier niveau. Nous dénotons un tel modèle par un modèle de **deuxième niveau**. Finalement, nous voulons appliquer des méthodes de prétraitement à un jeu de données bien réel dont la complexité est représentative de ce que nous pouvons rencontrer dans des cas d'application concrets.

3 Description des données analysées

Les données proviennent du *Ames Housing Dataset* et ont été produites par Prof. Dean De Cock de la Truman State University. Il s'agit des ventes de propriétés résidentielles dans la ville de Ames, en Ioawa, entre 2006 et 2010. Les données sont contenues dans deux fichiers : *train.csv* et *test.csv*. Respectivement, ils contiennent 1460 et 1459 exemples, pour un total de 2919. Chaque exemple contient 79 attributs catégoriels, numériques et booléens. Plusieurs d'entre eux traitent d'un même aspect. Par exemple, 4 s'intéressent à la taille du bâtiment ou de ses pièces, 7 au garage et 9 du sous-sol. Voici un tableau décrivant les 15 attributs que l'on a identifié comme étant les plus influents pour la tâche en utilisant l'algorithme de *gradient boosting* sur arbres de décision *CatBoost*. Notez que toutes les tailles sont exprimées en pieds carrés.

Nom	Description	Type	Importance (%)
OverallQual	Qualité des matériaux et de la finition	Numérique	18.92
GrLivArea	Taille du logement	Numérique	14.65
TotalBsmtSF	Taille du sous-sol	Numérique	8.04
LotArea	Taille du terrain	Numérique	3.98
YearBuilt	Année de construction	Numérique	3.49
1stFlrSF	Taille du premier étage	Numérique	3.16
YearRemodAdd	Année des rénovations	Numérique	3.08
Fireplaces	Nombre de foyer(s)	Numérique	2.91
BsmtFinSF1	Taille de l'aire terminée dans le sous-sol - type 1	Numérique	2.69
GarageCars	Nombre de véhicules dans le garage	Numérique	2.42
GarageArea	Taille du garage	Numérique	2.39
OverallCond	Condition générale du logement	Numérique	1.96
GarageType	Emplacement du garage	Catégoriel	1.40
CentralAir	Air climatisé central	Booléen	1.31
ExterQual	Qualité des matériaux extérieurs	Catégoriel	1.20

Table 1: Description des attributs

Une description complète des données est disponible dans le fichier *data_ description.txt* . La variable à prédire est, sans surprise, SalePrice. Les prix de vente varient entre 34 900 $US et 755 000 $US, avec une valeur médiane et moyenne de 163 000 $US et 180 921 $US respectivement. La différence entre la moyenne et la médiane est notamment expliquée par la présence de deux valeurs extrêmes. En effet, deux logements ont été vendus à un prix supérieur ou égal à 745 000 $US, tandis que tous les autres l'ont été à un prix inférieur à 630 000 $US.

4 Méthodologie

4.1 Prétraitement des données

Après avoir chargé les fichiers *.txt* dans un *data frame* avec la librairie *Pandas*, nous avons retiré certaines données aberrantes. Par exemple, si on affiche SalePrice en fonction de GrLivArea, on peut identifier deux points aberrants (quand 4500 $<$ GrLivArea et SalePrice $<$ 200 000). En fait, on peut même dire que ces deux points aberrants sont des points **influents**, puisque leur valeurs en abscisse sur ce graphique sont éloignées du reste du nuage de points. Ceci signifie que si nous incluons ces deux points dans notre modèle, la droite d'une régression linéaire en sera remarquablement affectée (elle sera aplatie, pente plus petite) ce qui est peu représentatif de la tendance générale du reste du jeu de données. On choisit donc de les retirer de notre ensemble d'entraînement. Notons du même coup que les deux valeurs extrêmes ayant un prix de vente supérieur à 700 000 sont des points très bénéfiques, puisque leurs abscisses sont éloignées du centre de masse, mais que leur ordonnée est alignée avec la tendance du reste du nuage (points non aberrants). Ils ont comme effet de réduire la variance d'une régression linéaire et donc d'améliorer la confiance qu'on a en nos prédictions.

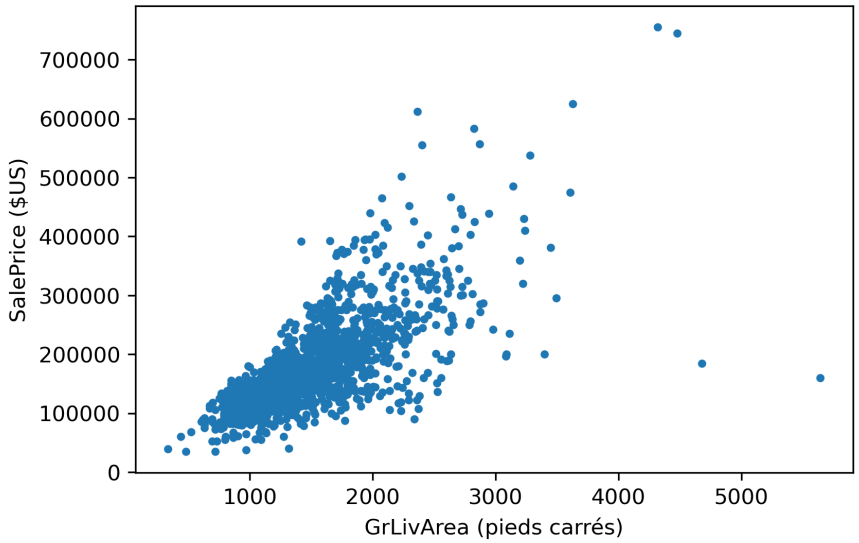


FIGURE 1: SalePrice en fonction de GrLivArea

Ensuite, nous avons remarqué que certains attributs contenaient un grand nombre de valeurs manquantes. Le tableau suivant répertorie les 5 attributs avec le plus de valeurs manquantes :

Nom	Proportion de valeurs manquantes
PoolQC	99.69%
MiscFeature	96.40%
Alley	93.21%
Fence	80.43%
FireplaceQu	48.68%

TABLE 2: Attributs avec valeurs manquantes

Nous avons décidé de supprimer les attributs avec 50% ou plus de valeurs manquantes, ce qui laisse seulement FireplaceQu du tableau ci-dessus. Aussi, nous avons transformé les variables catégorielles en variables booléennes. Prenons par exemple la variable Street, qui note le type de revêtement extérieur. Les deux valeurs possibles sont : Grvl (gravier) et Pave (pavé). La fonction `get_dummies` crée 2 attributs : Street_Grvl et Street_Pave, où les valeurs possibles sont 0 ou 1. En appliquant cette méthode sur les autres attributs, le nombre total d'attributs passe de 79 à 269.

Finalement, on remarque que la distribution de SalePrice est asymétrique à droite.

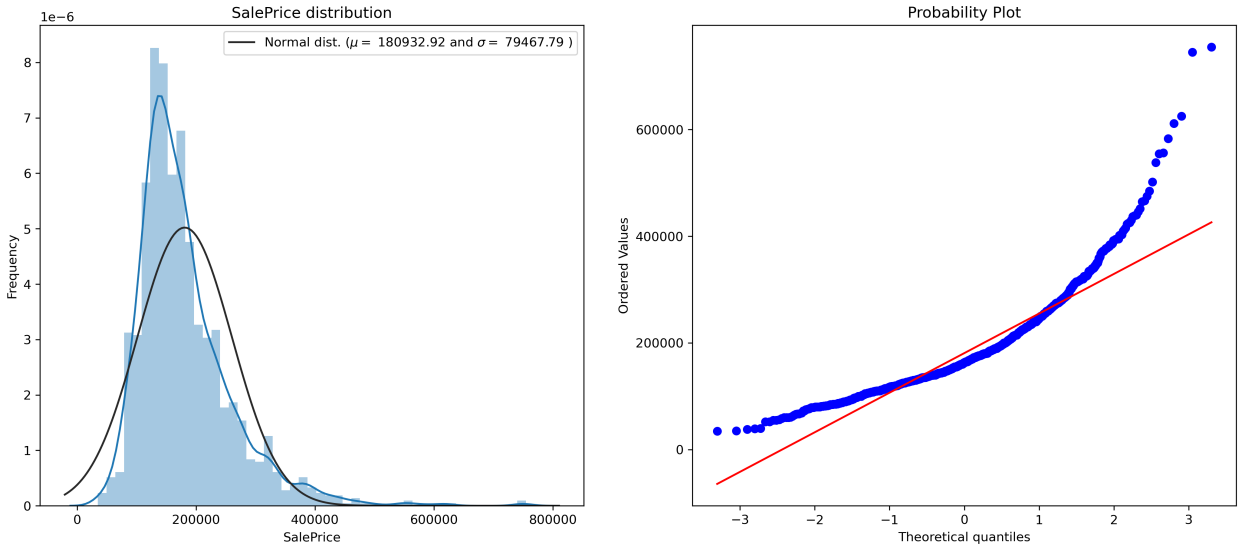


FIGURE 2: Analyse de SalePrice avec la librairie *Seaborn* (avant transformation)

Comme les modèles linéaires performent mieux lorsque la distribution des variables réponses est approximativement normale, nous devons identifier une transformation qui permettre de retrouver la normalité des données. Il se trouve que de prendre le log des prix de vente permet d'arriver à ce résultat. On utilisera donc cette transformation en tant que nouvelle variable réponse.

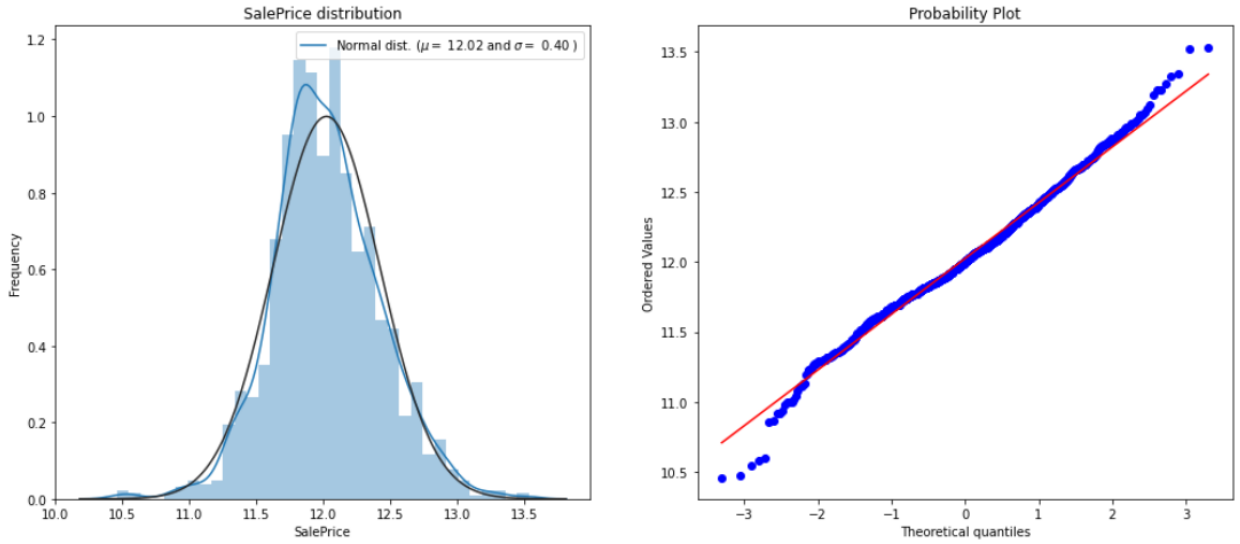


FIGURE 3: Analyse de SalePrice avec la librairie *Seaborn* (après transformation)

4.2 Assignation des attributs à chaque modèle

Afin qu'un modèle de deuxième niveau soit performant, il est important de retrouver une certaine indépendance au sein de ses variables explicatives (aussi nommées attributs). Rappelons que les attributs correspondent dans ce cas aux prédictions faites par les modèles de premier niveau. En effet, si les vecteurs de la matrice de design X sont très corrélés, on a un phénomène de redondance qui s'installe et les variables explicatives contiennent donc une quantité d'information assez pauvre comparativement à un ensemble de vecteurs de design peu corrélées. Additionnellement, dans les cas extrêmes, si deux vecteurs (colonnes) de la matrice de design X sont colinéaires (ou très près de l'être), alors la matrice $X^\top X$ n'est pas inversible (ou son inverse est alors très instable) et alors, la régression *Ridge*, si utilisée au deuxième niveau, ne possède pas de solution bien définie et unique (ou présente des prédictions à très grande variance). Nous voulons donc éviter que nos modèles de premier niveau produisent des prédictions trop similaires. D'une certaine manière, le but des modèles de deuxième niveau est d'être capable d'identifier les erreurs commises par les modèles de premier niveau, ce qui exige qu'ils ne commettent pas tous les mêmes. Ainsi, afin que le modèle de deuxième niveau fasse des prédictions riches, fiables et précises, nous devons avoir de faibles corrélations entre les prédictions des modèles de premier niveau, tout en conservant une certaine qualité de prédiction, bien sûr.

Afin d'induire de l'indépendance au sein des prédictions des modèles de premier niveau, nous entraînons nos modèles sur des ensembles d'attributs distincts. L'idée est donc similaire à celle utilisée

par les forêts aléatoires, où chaque arbre est entraîné à partir d'une poignée d'attributs sélectionnés aléatoirement pour produire des prédictions qui seront ensuite agrégées au moyen d'un vote de majorité. Également, nous voulons que nos prédictions au premier niveau soient assez performantes, sans quoi le modèle de deuxième niveau ne pourra jamais faire des prédictions justes. Pour ce faire, comme présenté à la section 3 sur la description des données analysées, nous utilisons *CatBoost* afin de déterminer quels sont les attributs les plus importants dans notre jeu de données. Ensuite, pour chaque modèle de premier niveau, la sélection de ses variables explicatives se fait en sélectionnant les j attributs les plus importants ainsi que k attributs aléatoires parmi les autres attributs disponibles. Ici, j et k jouent donc le rôle d'hyperparamètres pour le modèle de deuxième niveau.

Notons alors que nous n'avons pas à nous limiter à une seule instance de chaque modèle de premier niveau. En effet, deux instances de régresseurs *Lasso* vont par exemple produire des prédictions distinctes, puisqu'ils s'entraînent sur des attributs différents. Ceci motive l'introduction d'un nouvel hyperparamètre pour le modèle de deuxième niveau, soit M, le nombre de copies de chaque modèle construit. Finalement, pour que nos modèles de premier niveau produisent les meilleures prédictions possible, nous pouvons faire de l'analyse en composantes principales (une fois les $j + k$ attributs sélectionnés) et nous pouvons effectuer une optimisation de leurs hyperparamètres respectifs. Le travail d'optimisation qui en découle est présenté dans la section qui suit.

4.3 Optimisation des modèles de premier niveau

Premièrement, pour introduire de la variance dans les prédictions des modèles de premier niveau, il est d'abord avantageux d'utiliser une multitude de modèles distincts. Pour ce faire, nous utilisons 10 modèles disponibles dans la librairie *sklearn* de *Python*, soient un régresseur qui utilise les machines à vecteurs de support noté `SVR`, un régresseur probabiliste qui utilise l'inférence bayésienne noté `BayesianRidge`, un régresseur à base d'arbres de décision noté `DecisionTreeRegressor`, un régresseur à base de voisinage noté `KNeighboursRegressor`, trois modèles de régression linéaire régularisée notés `Ridge`, `Lasso` et `ElasticNet` et finalement, des versions robustes aux valeurs aberrantes pour ces trois derniers modèles.

Pour contrôler davantage l'exactitude et la variance des prédictions de ces modèles, nous utilisons les paramètres j, k et M décrits à la section précédente ainsi que PCA et $HYPER\_TUNING$. Ces deux derniers paramètres booléens permettent d'optimiser les résultats des modèles de premier niveau en indiquant si oui ou non nous faisons de l'analyse en composantes principales et si nous optimisons les modèles de premier niveaux avec leurs hyperparamètres respectifs. Nous avons donc un espace à 5 paramètres au sein duquel nous cherchons à trouver la configuration optimale. Pour évaluer la qualité d'une configuration, nous nous fions à la mesure de coût utilisée par la compétition *Kaggle* soit la *root mean squared logarithmic error (RMSLE)* définie comme suit :

$$RMSLE(\hat{y}, y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(\log(\hat{y}_i) - \log(\hat{y}_i)\Big)^2} \tag{1}$$

où y est le vecteur des réponses et $\hat{y}$ est le vecteur des prédictions, tous deux de taille n.

Pour réaliser cette recherche, nous devons effectuer plusieurs tests, ce qui nous force à entraîner un grand nombre de fois nos 10 algorithmes. À des fins d'efficacité en temps, comme l'activation des paramètres $HYPER\_TUNING$ et PCA augmente considérablement le temps de calcul, nous nous concentrerons sur les paramètres j, k et M pour notre optimisation. Cela est acceptable puisque ces deux paramètres peuvent être activés de manière indépendante aux trois derniers, et qu'il est assez certain qu'ils améliorent les résultats pour toutes les configurations possibles. Également, afin de structurer la recherche dans notre espace de paramètres, nous avons tenu bon de s'établir une configuration de référence pour les 5 paramètres à tester, puis de modifier un seul paramètre à la fois. La configuration de référence dite «baseline» est la suivante : $j = 5$, $k = 40$, $M = 3$, $PCA = 0$ et $HYPER\_TUNING = 0$.

Le premier paramètre à faire varier est j puisque c'est lui qui a le plus d'influence sur les autres. En effet, le paramètre j contrôle le nombre d'attributs importants que chaque modèle est forcé d'utiliser. Une petite valeur de j signifie une grande variance entre les modèles et donc une grande importance pour k et M, alors qu'une grande valeur de j signifie moins de variance entre les modèles et donc moins d'importance pour k et M. Pour chaque configuration testée, comme il y a de la variance dans les résultats qui est induite par la sélection aléatoire des k attributs, nous réalisons 5 exécutions de notre modèle global pour obtenir 5 scores *RMSLE*, et nous notons la valeur moyenne, la valeur maximale et la valeur minimale de ces mesures. Pour le modèle de deuxième niveau, nous utilisons un modèle simple et classique, soit la régression *Ridge*. Maintenant que tous ces détails sont énoncés, on peut présenter comment évolue la *RMSLE* lorsque le paramètre j varie et que les autres paramètres sont fixés à leur valeur *baseline* :

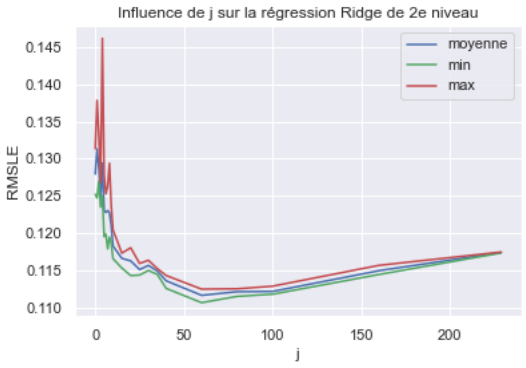


FIGURE 4: Influence du paramètre j sur la performance d'une régression *Ridge* au deuxième niveau

De ce graphique, nous pouvons observer deux choses. Premièrement, plus la valeur de j est grande, moins nous avons de variance dans les résultats, ce qui confirme l'explication énoncée au paragraphe précédent. Deuxièmement, le graphique semble avoir une allure convexe avec un minimum autour de $j = 60$. Ceci est une assez grande valeur de j, ce qui signifie que le modèle de deuxième niveau semble préférer l'exactitude de chaque prédicteur de premier niveau à leur indépendance. En d'autres mots,

il est préférable que nos modèles de premier niveau soient performants, même si ceci implique que leurs prédictions sont plus similaires. Il est à noter qu'on a également testé les configurations avec une petite valeur de j et de grandes valeurs de M, comme $j = 3$, $k = 40$ et $M = 36$, mais les résultats obtenus étaient nettement moins bons que ceux obtenus avec $j = 60$.

Une fois, j fixé à 60, on veut faire des tests pour déterminer k et M. On commence par M puisqu'on croit que son comportement devrait être le même peu importe la valeur de k. On obtient les meilleurs résultats lorsque $M = 3$, ce qui signifie qu'il est tout de même important d'avoir plusieurs copies distinctes d'un même modèle. Finalement, une fois qu'on a fixé $j = 60$ et $M = 3$, on peut faire une recherche pour trouver la meilleure valeur de k. On obtient $k = 70$.

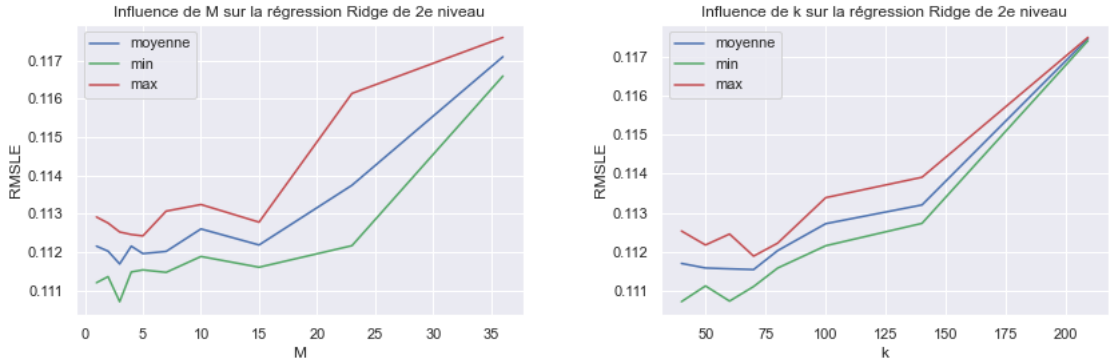


FIGURE 5: Influence des paramètres M et k sur la performance de *Ridge* au deuxième niveau

Maintenant qu'on a fixé $j = 60$, $k = 70$ et $M = 3$ comme meilleure configuration possible, nous pouvons faire de l'analyse en composantes principales. En effet, $j + k = 130$ attributs sont utilisés pour chaque modèle. Pour que chacun d'entre eux puisse sélectionner le nombre de composantes principales qui lui convient le mieux, nous définissons la liste suivante : $PCA = [5, 10, 15, 25, 50, 100, 130]$. L'idée est que chaque modèle construit va tester, par validation croisée, quel est le nombre de composantes principales parmi la liste précédente qui lui permet d'obtenir la meilleure performance. Pour donner quelques exemples, on peut noter que `SVR` sélectionne presque toujours ses 10 premières composantes principales, `BayesianRidge` sélectionne presque toujours ses 130 premières composantes principales (la totalité), et certains modèles comme `Lasso` sélectionnent des nombres de composantes principales variables parmi 15, 25 ou 50.

En résumé, nous construisons 3 instances de 10 modèles de premier niveau distincts (donc 30 modèles en tout) que nous entraînons à l'aide des 60 attributs les plus importants (à des fins de performance) ainsi que 70 attributs supplémentaires aléatoires (à des fins de variance). Chacun de ces 30 modèles fait de l'analyse en composantes principales pour optimiser sa capacité à généraliser sur de nouvelles observations, et les hyperparamètres propres à chacun des 30 modèles sont également optimisés lorsque $HYPER\_TUNING = 1$. Cela nous donne donc 30 vecteurs de prédiction qui consituent les attributs sur lesquels entraîner nos modèles de deuxième niveau. La construction des ensembles

d'entraînement et de validation pour les modèles de deuxième niveau est décrite davantage à la section suivante.

4.4 Construction des attributs pour le deuxième niveau

Pour chacun des 30 modèles construits, l'ensemble d'entraînement de 1460 exemples du *Ames Housing Dataset* est divisé en 20 blocs à l'aide de la fonction `KFold` de la librairie *Scikit-Learn* qui fait le partitionnement du jeu de données de manière aléatoire (et donc différente d'un modèle à l'autre). De manière itérative, chaque bloc est utilisé une fois comme ensemble de test tandis que les 19 autres servent d'ensemble d'entraînement. Pour chaque modèle, les prédictions faites sur les 20 ensembles de test ainsi construits constituent un attribut de l'ensemble d'entraînement qui sera fourni aux modèles de deuxième niveau. Cette procédure a comme avantage de préserver la taille originale de l'ensemble (1460 exemples) pour les méthodes de deuxième niveau tout en évitant de faire des prédictions sur des données déjà vues lors de l'entraînement.

L'ensemble de test des modèles de deuxième niveau a donc été construit à partir des prédictions faites sur l'ensemble de test au premier niveau. Les attributs du jeu de données du deuxième niveau sont les différents prédictions faites par les modèles du premier niveau.

4.5 Sélection de modèles

L'approche utilisée pour la sélection et l'optimisation des modèles de premier niveau nous a permis d'identifier une gamme de modèles qui permettent tous d'effectuer des prédictions de qualité satisfaisante pour notre tâche de régression.

Or, dans le contexte d'un modèle à deux niveaux, les prédictions de chacun de ces modèles pour les données d'entraînement constituent les attributs à utiliser pour l'entraînement du modèle de deuxième niveau. Même si nos tests précédents permettent de conclure que les prédictions effectuées par chacun des modèles de premier niveau contiennent des informations pertinentes pour la tâche de deuxième niveau, ceci n'implique pas pour autant qu'il soit pertinent de les inclure toutes dans notre ensemble d'attributs construits. Par analogie, en régression linéaire multiple, même si une variable explicative possède une corrélation significative avec la variable réponse que l'on souhaite prédire, il est possible qu'il soit préférable de la retirer du modèle si l'information qu'elle apporte est déjà contenue dans une ou plusieurs autres variables explicatives du modèle (ce qu'on appelle dans ce cas de la proche-multicolinéarité).

Cette comparaison nous motive à adapter au contexte des modèles à deux niveaux les techniques de sélection d'attributs qui sont utilisées en régression linéaire multiple et en régression logistique. En effet, pour trouver le sous-ensemble de p attributs menant à la meilleure performance de validation, une recherche exhaustive demanderait de considérer 2^p modèles distincts, ce qui devient rapidement impossible en pratique. C'est pourquoi des méthodes de sélection itératives des attributs sont fréquemment utilisées dans ce contexte.

En particulier, les approches de sélection de modèles dites « forward » et « backwards » sont parmi les plus utilisées. Dans le premier cas, la procédure débute par l'ajustement d'un modèle de régression trivial qui n'utilise qu'un terme constant comme attribut. On peut alors calculer une certaine fonction de coût sur l'ensemble de validation pour évaluer la qualité de ce premier modèle. Ensuite, on répète cette procédure pour les p modèles obtenus par l'ajout d'une des variables explicative à notre ensemble d'attributs utilisés, en conservant le modèle menant à la plus faible erreur de validation. À chaque étape, on ajoute ainsi un attribut à notre modèle, en arrêtant l'exécution lorsque l'ajout de chacun des attributs restants ne mène plus à une amélioration de la mesure de performance choisie.

Symétriquement, la procédure d'élimination *backwards* consiste à commencer avec le modèle complet, c'est-à-dire avec le modèle qui inclut toutes les variables explicatives candidates, puis à éliminer à chaque étape la variable qui mène à la plus grande amélioration de notre mesure de performance lorsqu'elle est retirée du modèle, en arrêtant la procédure lorsqu'aucun modèle réduit n'améliore la performance de validation.

Dans notre cas d'application, en considérant les 10 modèles de premier niveau décrits précédemment et pour une valeur donnée du paramètre M, qui détermine le nombre de modèles aléatoires de premier niveau entraînés pour chaque algorithme, l'ensemble des attributs contient $10 \cdot M$ vecteurs de prédictions. Pour simplifier la procédure de sélection, on ajoute cependant une contrainte qui stipule que les M versions d'un modèle doivent toutes être incluses dans l'ensemble final des attributs si au moins l'une d'entre elles l'est. On se ramène ainsi à un ensemble de 10 groupes de M attributs à traiter lors de la procédure de sélection.

Pour évaluer la qualité d'une sélection de modèles de premier niveau, on calcule la *RMSLE* des prédictions obtenues sur l'ensemble de validation croisée 20-*fold* par une régression Ridge entraînée à partir des prédictions de niveau de premier niveau sélectionnés.

Les tableaux suivants résument les résultats obtenus pour les procédures de sélection *forward* et *backwards* respectivement. Les hyper-paramètres de premier niveau utilisés sont $j = 60$, $k = 70$ et $M = 3$.

Modèle	Modèle de premier niveau ajouté	RMSLE
$E_{f,1}$	`Bayes`	0.11208
$E_{f,2}$	`LassoRobust`	0.11116
$E_{f,3}$	`KNeighbors`	0.11066
$E_{f,4}$	`Ridge`	0.11047
$\boldsymbol{E_{f,5}}$	`ElasticNetRobust`	**0.11046**
$E_{f,6}$	`ElasticNet`	0.11052

(a) Par procédure *forward*

Modèle	Modèle de premier niveau éliminé	RMSLE
$E_{b,10}$	-	0.11107
$E_{b,9}$	`RidgeRobust`	0.11094
$E_{b,8}$	`Tree`	0.11065
$E_{b,7}$	`Lasso`	0.11058
$E_{b,6}$	`SVR`	0.11052
$\boldsymbol{E_{b,5}}$	`ElasticNet`	**0.11046**
$E_{b,4}$	`ElasticNetRobust`	0.11047

(b) Par élimination *backwards*

FIGURE 6: Sélection de modèles par différentes procédures

Il est intéressant de constater que les deux procédures mènent exactement à la même solution, malgré le caractère heuristique de ces procédures de sélection. Le modèle sélectionné par les deux approches, qui utilise 5 des 10 modèles de premier niveau, sera conservé pour la suite.

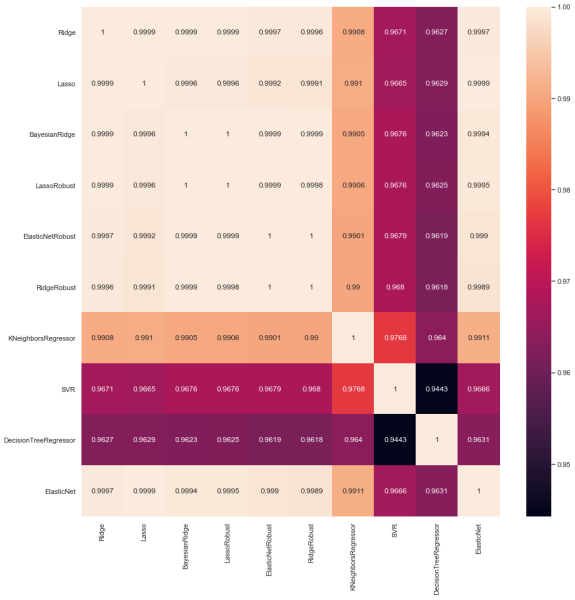
4.6 Modèles de 2e niveau

Les tests précédents ont été réalisés en utilisant uniquement le modèle `Ridge` au deuxième niveau. Pour la construction de notre modèle final, nous désirions toutefois comparer la performance de `Ridge` à d'autres alternatives.

Pour ce faire, nous avons réutilisé au deuxième niveau l'ensemble de 10 modèles qui ont été utilisés au premier niveau. Chacun d'entre eux a été utilisé en utilisant la même technique d'optimisation des hyperparamètres et de sélection du nombre de composantes principales qui a été décrite précédemment. Le tableau suivant présente, en ordre croissant, la *RMSLE* obtenue par validation croisée pour chacun des modèles de deuxième niveau entraîné sur l'ensemble d'attributs sélectionné dans la section précédente.

Rang	Algorithme	RMSLE
1	`Ridge`	0.11046
2	`Lasso`	0.11058
3	`BayesianRidge`	0.11059
4	`LassoRobust`	0.11076
5	`ElasticNetRobust`	0.11091
6	`RidgeRobust`	0.11092
7	`KNeighboursRegressor`	0.11833
8	`SVR`	0.12134
9	`DecisionTreeRegressor`	0.16176
10	`ElasticNet`	0.18020

(a) Performance sur l'ensemble d'attributs $E_{f,5}$



(b) Matrice de corrélation sur l'ensemble de test

FIGURE 7: Résultats pour les modèles de deuxième niveau

On pourrait simplement choisir comme modèle final l'algorithme de deuxième niveau étant associé à la plus faible perte sur l'ensemble de validation. Cependant, comme on utilise dans ce projet une approche ensembliste, on décide plutôt d'étendre au second niveau le raisonnement fondamental de ce type d'approche, soit que la combinaison de plusieurs modèles est souvent préférable au meilleur des modèles individuels. En fait, il serait même possible de répéter la procédure appliquée ici pour construire un modèle de régression ayant un nombre arbitrairement élevé de niveaux. La

corrélation entre les prédictions des meilleurs modèles de second niveau est cependant très élevée. Ainsi, la matrice des attributs que l'on obtiendrait au modèle de troisième niveau serait sujette à des problèmes de colinéarité importants, ce qui ne permettrait pas d'obtenir des résultats pertinents en réappliquant des modèles d'apprentissage. On se contentera donc de prendre la moyenne des prédictions de plusieurs modèles de deuxième niveau pour conclure.

4.7 Autres méthodes ensemblistes

Au cours de ce projet, nous avons construit plusieurs modèles ensemblistes en utilisant l'approche du *stacking* à partir de plusieurs modèles de régression relativement simples. Cependant, il existe une riche littérature autour des méthodes ensemblistes en apprentissage automatique et plusieurs méthodes que nous n'avons pas implantées ont le potentiel de se montrer efficaces pour notre tâche. Pour comparer l'efficacité de notre approche avec celle d'autres méthodes ensemblistes populaires, nous avons profité du fait que plusieurs d'entres-elles sont implantées dans diverses librairies *Python*. En particulier, nous avons testé plusieurs paramétrisations des modèles de *gradient boosting* `xgboost`, `lightgbm` et `catboost` ainsi que du modèle de forêt aléatoire `RandomForestRegressor` de *sklearn* proposées sur des forums et par la communauté Kaggle. Chacun de ces modèles a été entraîné sur la totalité des attributs de notre jeu de données. Nous avons identifié une seule configuration menant à des performances satisfaisantes pour ces modèles, soit la configuration du modèle `xgboost` proposée par un usager de *Stack Exchange*. Le *RMSLE* obtenu par validation croisée pour ce modèle était de 0.11468, ce qui le classerait au 7^e rang parmi nos modèles de deuxième niveau.

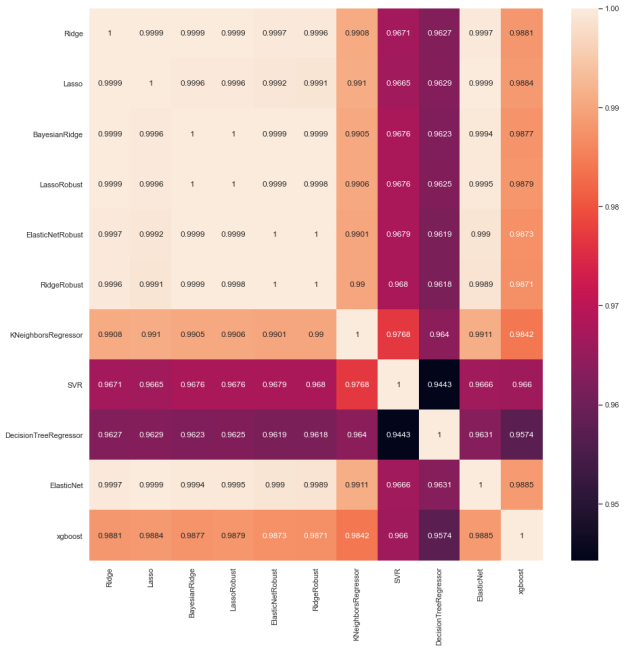


FIGURE 8: Matrice de corrélation sur l'ensemble de test : modèles de deuxième niveau et `xgboost`

Comme le modèle `xgboost` est entraîné sur l'ensemble des attributs initiaux et non sur les prédictions

des modèles de premier niveau, la corrélation entre ses prédictions et les prédictions des meilleurs modèles de deuxième niveau est considérablement plus faible que la corrélation entre les prédictions de ces derniers. Puisque l'erreur de validation de `xgboost` est tout de même assez faible, on s'attend à ce que la mise en commun par une simple moyenne pondérée de ses prédictions avec celles des modèles de deuxième niveau mène à des gains de performance appréciables.

5 Modèle final et résultats

Pour construire notre modèle ensembliste final, nous considérons la moyenne des p meilleurs modèles à deux niveaux identifiés précédemment et calculons le *RMSLE* pour chacun des ces modèles agrégés. Par exemple, le modèle $p = 3$ prend la moyenne des prédictions des modèles `Ridge`, `Lasso` et `BayesianRidge` de deuxième niveau. Pour chaque valeur de p, on considère également différentes pondérations α du modèle `xgboost` dans les prédictions. Le graphique suivant illustre l'erreur de validation obtenue en fonction de p et α. Puisqu'un creux notable est identifiable en $p = 8$, on fixe cette valeur, puis on effectue une recherche fine sur l'intervalle $[0, 0.5]$ pour trouver la meilleure valeur du paramètre α.

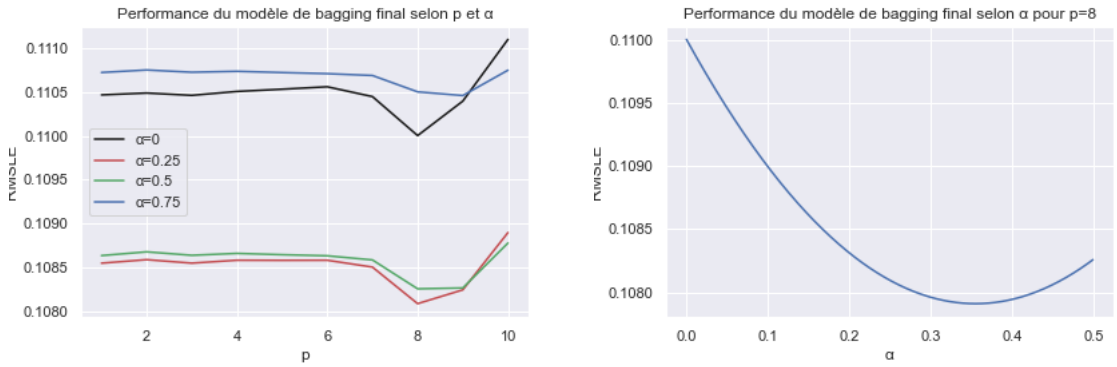


FIGURE 9: Influence des paramètre p et α sur la performance du modèle final

Le plateau apparent que l'on observe pour les valeurs de p entre 1 et 6 est explicable par la très forte corrélation qui lie les prédictions de nos 6 meilleurs modèles de deuxième niveau (voir Figure 7 (b)). Les vecteurs de prédiction de ces six modèles étant pour l'essentiel identiques, toutes les combinaisons possibles de ces prédictions le sont aussi.

Le modèle final contient donc les 8 meilleurs modèles à deux niveaux et accorde une pondération de $\alpha^* = 0.355$ au modèle `xgboost`. Il atteint une erreur de validation de 0.10790 selon le critère *RMSLE*.

Lorsqu'utilisé pour prédire les données de test, ce modèle nous permet d'obtenir un score de 0.1216 sur le classement de Kaggle, ce qui nous place dans le top 10% pour cette compétition publique.

6 Conclusion

Ce projet nous aura permis de tester une multitude de modèles de régression et de les comparer entre eux. On remarque que les modèles `Ridge` et `BayesianRidge` donnent de très bons résultats comparativement aux modèles `SVR`, `DecisionTreeRegressor` et `KNeighboursRegressor`. Toutefois, aucun de ces modèles ne produit de résultats aussi performants que les résultats obtenus par les méthodes ensemblistes basées sur le *stacking* et la mise en commun des modèles de deuxième niveau. En effet, Le meilleur modèle de premier niveau que nous avons pu identifier lorsqu'entraîné sur l'ensemble des attributs disponibles est une configuration de `BayesianRidge` qui a atteint un score de validation de 0.11282. Ceci est significativement supérieur au score de validation de 0.10790 de notre modèle final.

Néanmoins, bien que les modèles `SVR`, `DecisionTreeRegressor` et `KNeighboursRegressor` soient moins performants, on remarque que certains d'entre eux contribuent positivement aux performances des modèles de deuxième niveau. Il semble en être ainsi parce que ces modèles produisent des prédictions qui sont peu corrélés aux prédictions des autres modèles. Également, on peut noter que notre configuration optimale des paramètres j, k et M accorde une prévalence à la précision des prédictions des modèles de premier niveau au détriment de leur indépendance. Il semble que la variance induite par l'utilisation de 10 algorithmes différents est suffisante pour introduire assez de diversité au sein des prédictions. Justement, la sélection des 10 algorithmes de premier niveau que l'on a finalement utilisés a été motivée par nos expériences initiales, qui ont notamment mené au rejet des réseaux de neurones comme régresseurs de premier niveau. Cependant, nous sommes conscients que notre recherche de candidats au premier niveau est loin d'être exhaustive et que la construction d'une gamme plus diversifiée de modèles aurait le potentiel d'augmenter la diversité des attributs construits utilisés au deuxième niveau et, en définitive, d'améliorer la performance globale de notre approche.

Également, ce projet nous aura permis d'appliquer plusieurs méthodes d'analyse et de prétraitement de données. Notamment, l'utilisation de l'analyse en composantes principales améliore nettement la capacité que les modèles de premier niveau ont à généraliser sur de nouveaux exemples.

Une avenue future qui serait intéressante à explorer concerne l'utilisation conjointe de prédictions de modèles de premier niveau et de modèles ensemblistes dans la construction du modèle final. Effectivement, le modèle proposé n'exploite que des prédictions de deuxième niveau effectuées à partir de prédictions de modèles de premier niveau entraînés sur des sous-ensembles restreints d'attributs. La bonne qualité de modèles plus complets au premier niveau (comme `BayesianRidge`) pourrait probablement être mise à profit dans la phase finale de notre approche ensembliste. Pour les mêmes raisons qui font que les prédictions de `xgboost` ne sont pas trop fortement corrélées avec celles de nos modèles de deuxième niveau, une même conclusion est à anticiper pour les modèles de premier niveau, d'où le potentiel de leur utilisation à la dernière étape du *stacking*.

7 Contribution des membres de l'équipe

7.1 Contribution de Sandrine Bédard

Prétraitement complet des données. Implantation de la classe `Model` et de la structure pour tester les hyperparamètres de manière automatique et reproductible. Rédaction du modèle SVR et de ses paramètres. Tests d'optimisation pour les hyperparamètres j, k et M. Initialisation de la structure du rapport et rédaction des sections 3, 4.1 et 4.4.

7.2 Contribution de Simon Chasles

Décorticage initial du jeu de données afin d'en saisir le contenu de manière instinctive. Rédaction de parties du code responsable des modèles de premier niveau : (1) code structurant l'utilisation des hyperparamètres j, k, M, PCA et $HYPER\_TUNING$, (2) fonctions effectuant l'analyse en composantes principales, et (3) division du jeu de données permettant la construction des ensembles d'entraînement et de test pour les modèles de deuxième niveau. Optimisation des hyperparamètres j, k et M. Rédaction des sections 1, 2, 4.2 et 4.3 du rapport.

7.3 Contribution de Robin Legault

Écriture du code permettant de faire l'analyse d'importance et la sélection aléatoire des attributs. Sélection de l'ensemble des attributs à utiliser pour les modèles de deuxième niveau (sélection *forward* et *backwards* des prédicteurs de premier niveau). Développement, optimisation et analyse des modèles de deuxième niveau et de toutes les approches ensemblistes testées. Construction du modèle final par *stacking*. Rédaction des sections 4.5, 4.6, 4.7 et 5 du rapport.

Références

[1] Donghai Guan et al. "A Review of Ensemble Learning Based Feature Selection". In : *IETE Technical Review* 31.3 (2014), p. 190-198. DOI : 10.1080/02564602.2014.906859. eprint : https://doi.org/10.1080/02564602.2014.906859. URL : https://doi.org/10.1080/02564602.2014.906859.

[2] *House Prices - Advanced Regression Techniques*. URL : https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview/description.

[3] Omer Sagi et Lior Rokach. "Ensemble learning : A survey". In : *WIREs Data Mining and Knowledge Discovery* 8.4 (2018), e1249. DOI : https://doi.org/10.1002/widm.1249. eprint : https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1249. URL : https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249.