

Travail fait par

Matis Brassard-Verrier (111 182 740)

Alyson Marquis (111 183 605)

Alexis Picard (111 182 200)

Samuel Provencher (111 181 794)

Apprentissage statistique en actuariat

ACT-3114

Rapport 2

Présenté à

Marie-Pier Côté

École d'actuariat

Université Laval

22 avril 2020

Table des matières

Introduction	1
Modèle de base	1
Ajustement des modèles	2
Modèle linéaire généralisé avec une régularisation Lasso	2
Modèle des k plus proches voisins	2
Arbre de décision	2
Ensemble d'arbres de décisions agrégées par <i>bagging</i>	3
Forêt aléatoire	4
Modèle de <i>gradient boosting</i>	4
Comparaison des modèles	5
Interprétation des meilleurs modèles	6
Conclusion	7
Bibliographie	8

Introduction

Dans le cadre du travail, nous allons tenter de modéliser le prix de vente des maisons dans la région de Seattle (King County, USA) en utilisant de nombreuses caractéristiques ayant une incidence sur la valeur d'une maison. La variable réponse à prédire, soit le prix de vente d'une maison, est une valeur positive évaluée en dollars américains. La modélisation de cette variable pourrait être utile pour différentes raisons dans un contexte actuariel. Comme la somme assurée d'une maison a un lien très fortement proportionnel à son prix de vente, une compagnie d'assurance pourrait être intéressée de modéliser le prix de vente de maisons dans des nouveaux développements immobiliers afin de tenter de prédire les futures soumissions d'assurance habitation et d'offrir des offres personnalisées aux acheteurs de ces nouvelles maisons. Dans un autre contexte, au niveau de la gestion des risques, certains assureurs ont un portefeuille de prêts hypothécaires ou utilisent des produits dérivés sur prêts hypothécaires pour se couvrir du risque (*hedging*). Ainsi, il pourrait être intéressant d'avoir une estimation des montants de prêts hypothécaires dans une région donnée en se basant sur le prix de vente des maisons afin de mieux gérer le risque de la compagnie. La pertinence de trouver cette variable qu'est le prix de vente des maisons devient alors fort intéressante.

Le jeu de données utilisé sera le suivant : `kc_house_sales` (House sales in King County, USA). Dans les prochaines sections, sept modèles seront étudiés, dont deux qui le seront plus en profondeur. Pour ce faire, 80 % des données seront utilisées pour effectuer l'entraînement des modèles et 20 % seront réservées pour tester ainsi que comparer les modèles entre eux.

Modèle de base

Un bon modèle de base a été choisi en utilisant une technique étudiée dans le cours ACT-2003 Modèles linéaires en actuariat, soit la régression linéaire multiple. Ce type de modèle a été choisi en raison de sa simplicité et parce qu'il s'adapte bien au jeu de données. En effet, la variable réponse *price* est monétaire et possède une distribution asymétrique. Il a été vu, qu'en présence de ce type de variable réponse, une régression linéaire multiple en appliquant une transformation logarithmique sur la variable réponse était appropriée. Tel que mentionné dans la première partie de ce travail, la transformation logarithmique permet de s'approcher de la distribution d'une loi normale, ce qui rend la variable réponse plus facile à modéliser. Pour construire le modèle, seulement l'échantillon d'entraînement a été utilisé. De plus, le modèle utilise toutes les 17 variables explicatives. Cependant, aucune interaction entre les variables explicatives n'a été considérée afin de garder le modèle simple et facilement interprétable. Certaines variables catégorielles à plusieurs niveaux, dont l'importance des interactions étaient négligeables augmentaient le temps de calculs et rendaient le modèle plus difficilement interprétable. Ainsi, dans l'idée d'avoir un modèle de base simple, il a été décidé de ne pas considérer les interactions dans ce modèle. En outre, une sélection de variable formelle n'a pas été effectuée contrairement à ce qui est habituellement fait lorsqu'on veut raffiner un modèle linéaire multiple.

Ajustement des modèles

Modèle linéaire généralisé avec une régularisation Lasso

Dans le cadre du travail, il a été choisi d'effectuer un modèle linéaire généralisé avec une régularisation de type Lasso. Notre choix s'est arrêté sur ce type de régularisation, puisque la régularisation Lasso permet d'effectuer la sélection de variables. Pour se faire, il suffit de minimiser l'équation de score suivante :

$$S^{Lasso} = \sum_{i=1}^p (Y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p |\beta_j|,$$

où p est le nombre de paramètres du modèle et λ est le paramètre de régularisation. La minimisation de cette équation mènera à des coefficients β exactement égal à zéro, sélectionnant ainsi les variables du modèle. Nous avons utilisé la méthode implantée dans le paquetage **glmnet** pour choisir le paramètre λ ainsi que pour bâtir le modèle.

Afin de modéliser le prix de vente des maisons à King County, le modèle linéaire généralisé avec une régularisation Lasso a été construit à l'aide de l'échantillon d'entraînement. Le paramètre de régularisation a été choisi à l'aide d'une validation croisée à six plis. Cette validation croisée est intégrée dans la fonction `cv.glmnet`. Ainsi, la valeur optimale de ce paramètre est de $\lambda = 0.0000791264$. Le modèle retenu est composé de sept variables explicatives. Le modèle est aussi constitué de 22 termes d'interaction.

Modèle des k plus proches voisins

Un modèle qui a été testé est celui des k plus proches voisins. Étant donné que nous avons un problème de régression, la fonction `knn.reg` du paquetage **FNN** a été utilisée pour construire ce modèle.

Le modèle des k plus proches voisins est simpliste. Afin de prédire une observation dont les valeurs des variables explicatives sont comprises dans le vecteur \mathbf{x}_0 , il faut regarder l'ensemble des k plus proches voisins de \mathbf{x}_0 , c'est-à-dire les observations qui minimisent la distance Euclidienne. Puis, la prévision du point \mathbf{x}_0 est la moyenne des variables réponses des observations comprises dans l'ensemble des k plus proches voisins.

Il faut d'abord déterminer la valeur optimale de k , soit le nombre de voisins à considérer. Pour se faire, une validation croisée à 10 plis a été utilisée. La fonction `train` du paquetage **caret** a été utilisée pour faire cette validation croisée. Étant donnée que le modèle des k plus proches voisins est fondé sur la distance Euclidienne, les données ont été standardisées avant de procéder à la validation croisée. La métrique choisie pour sélectionner la valeur de k est l'erreur quadratique moyenne (`metric="RMSE"`). Ainsi, la valeur de k qui minimisait l'erreur quadratique moyenne est $k = 9$.

Arbre de décision

Un modèle qui a été décidé de tester est un arbre de décision, mais plus précisément dans le cas présent, des arbres de régression. Pour ce faire, l'algorithme *classification and regression tree* (CART) implanté dans le paquetage **rpart** a été utilisé.

Il a tout d'abord été décidé d'optimiser l'hyperparamètre `minbucket`, soit le nombre minimal d'observations dans une feuille de l'arbre. Une méthode manuelle a dû être utilisée parce qu'on ne peut pas optimiser cet hyperparamètre à l'aide des méthodes habituelles. Plusieurs valeurs ont été testées entre `minbucket = 1` et `minbucket = 200` afin de se donner une idée. Par la suite, la recherche a été raffinée et la valeur qui a ainsi été trouvée est de `minbucket = 9`.

Avec cet hyperparamètre défini, un arbre de régression a été construit en utilisant toutes les variables explicatives ainsi qu'un paramètre de complexité nul ($cp = 0$). L'arbre de régression est obtenu en spécifiant `method="anova"` et permet de trouver l'arbre minimisant l'erreur quadratique moyenne. Seulement

l'échantillon d'entraînement a été utilisé pour entraîner ce modèle, l'échantillon test étant réservé pour analyser les performances prédictives du modèle. Afin d'optimiser le paramètre de complexité, une validation croisée en 10 plis a été effectuée. Cette validation croisée est implantée de base dans la fonction `rpart`, donc aucune programmation supplémentaire n'a été nécessaire. Ainsi, le paramètre de complexité optimal est de 0.0000353939. Ce choix optimal a été utilisé pour élaguer l'arbre de régression et ainsi réduire la variance de la prédiction. L'élaguage représente un bon compromis entre le biais et la variance de la prédiction.

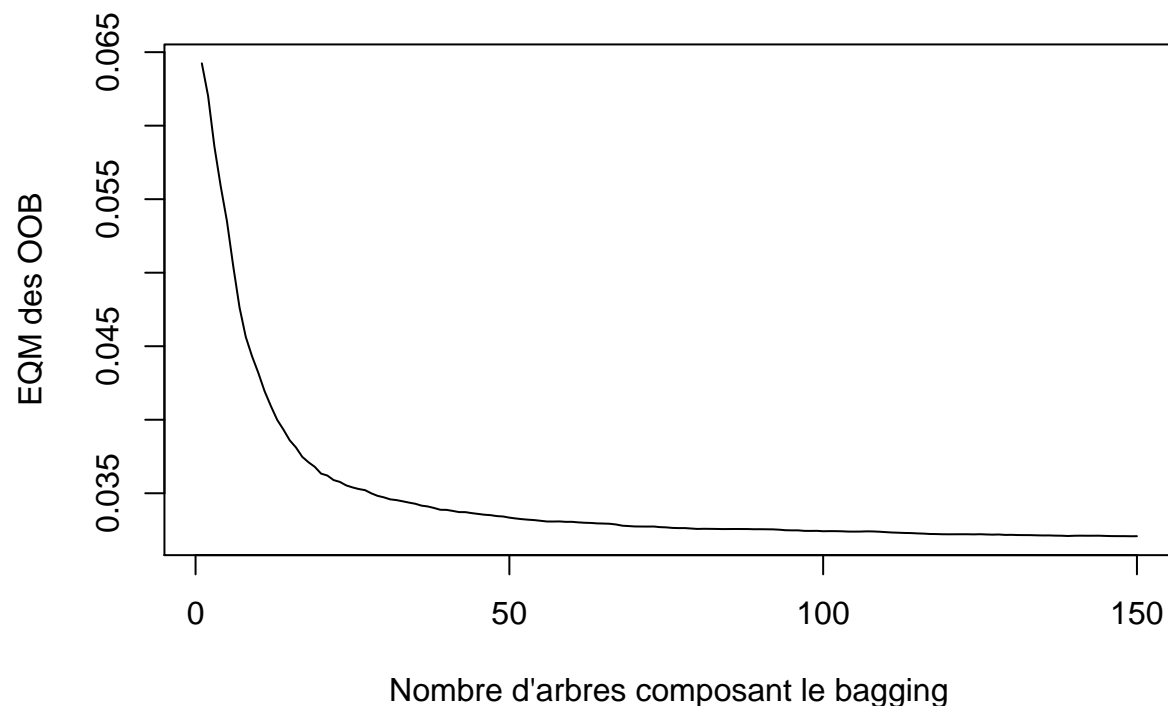
Il aurait été intéressant de représenter graphiquement ce modèle puisqu'il s'agit d'une façon de bien comprendre ce type de modèle. Cependant, il n'a pas été possible de présenter ce modèle sous forme de graphique. En effet, malgré le fait que l'arbre a été élagué, il était trop complexe pour être agréable à regarder pour l'oeil. Ceci est dû au fait que le modèle a été entraîné avec 17276 observations, ce qui est un nombre assez important.

Ensemble d'arbres de décisions agrégées par *bagging*

Un autre modèle qui a été testé est celui du bagging avec des arbres de régression. Étant donné le type de la valeur réponse recherchée soit le prix d'une maison, cette valeur est numérique continue et donc les arbres de régression sont ceux compatibles pour prédire cette dite valeur. Pour ce faire, l'algorithme *classification and regression with Random Forest* (randomForest) implanté dans le paquetage `randomForest` a été utilisé.

Un échantillon bootstrap avec remise de la même taille que l'échantillon d'entraînement a été sélectionné par l'algorithme `randomForest` (`sampszie= nrow(donnees.train)`). Le nombre de variables échantillonnées aléatoirement a donc été déterminé à 17 (`mtry=17`) étant donné que notre base de données d'entraînement comportait 18 variables avec la variable réponse incluse. La seule paramètre qui était à déterminer était celui du nombre d'arbres de régression créés avec le bagging. Pour ce faire un graphique de l'erreur quadratique moyenne (EQM) des observations OOB en fonction du nombre d'arbres composant le bagging a été tracé afin de déterminer quand l'EQM des OOB se stabilisait à son minimum. Sur le graphique suivant, on voit d'ailleurs que 150 arbres étaient suffisants et c'est donc le nombre qui a été retenu (`ntree=150`).

Graphique X: EQM des observations OOB en fonction du nombre d'arb



À noter qu'aucun élagage n'a été fait sur les arbres de régression créé par le bagging et donc ceux-ci avaient un paramètre `cp=0`.

Forêt aléatoire

Un autre modèle qui a été testé est celui de la forêt aléatoire. Pour ce faire, l'algorithme *classification and regression with Random Forest* (`randomForest`) implanté dans le paquetage `randomForest` a été utilisé.

Le bagging et la forêt aléatoire sont très similaires à la différence que la forêt aléatoire permet de décorréler les arbres à l'aide de deux manières.

De 1, on choisit un échantillon bootstrap plus petit que le bagging, c'est pourquoi dans notre cas un `sampsize= 0.6*nrow(donnees.train)` a été choisi ce qui équivaut à utiliser 60% des données de l'échantillon d'entraînement.

De 2, à chaque itération dans la construction de l'arbre, on choisit aléatoirement m prédicteurs qui seront les candidats pour la séparation (`mtry`). Ce choix de m optimal a été fait à l'aide d'une validation croisée à 5 plis. La fonction `train` du paquetage `caret` a été utilisée pour faire cette validation croisée. La métrique choisie pour sélectionner la valeur de m est l'erreur quadratique moyenne (`metric="RMSE"`). Ainsi, la valeur de m qui minisait l'erreur quadratique moyenne est $m = 8$.

Pour ce qui est des autres paramètres, les mêmes valeurs ont été gardé pour le bagging et la forêt aléatoire.

Modèle de *gradient boosting*

Comparaison des modèles

Interprétation des meilleurs modèles

Conclusion

Bibliographie

1. Kaggle, harlfoxen (2017). House sales in King County, USA. Récupéré le 27 février 2020 de <https://www.kaggle.com/harlfoxem/housesalesprediction>.
2. Max Kuhn (2020). caret : Classification and Regression Training. R package version 6.0-85. <https://CRAN.R-project.org/package=caret>
3. Terry Therneau and Beth Atkinson (2019). rpart : Recursive Partitioning and Regression Trees. R package version 4.1-15. <https://CRAN.R-project.org/package=rpart>
4. Stephen Milborrow (2019). rpart.plot : Plot ‘rpart’ Models : An Enhanced Version of ‘plot.rpart’. R package version 3.0.8. <https://CRAN.R-project.org/package=rpart.plot>
5. A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.
6. Brandon Greenwell, Bradley Boehmke, Jay Cunningham and GBM Developers (2019). gbm : Generalized Boosted Regression Models. R package version 2.1.5. <https://CRAN.R-project.org/package=gbm>
7. Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL <http://www.jstatsoft.org/v33/i01/>.
8. Alina Beygelzimer, Sham Kakadet, John Langford, Sunil Arya, David Mount and Shengqiao Li (2019). FNN : Fast Nearest Neighbor Search Algorithms and Applications. R package version 1.1.3. <https://CRAN.R-project.org/package=FNN>