

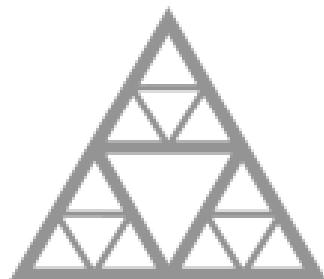
ÉCOLE DES PONTS PARISTECH

PRAMA Challenge Rendu

Groupe 3

Xiaohang HOU, Zhongchang LUO, Thi Phuong Lan VU
Ping'an YANG, Guoqing YIN

5 juin 2022



École des Ponts

ParisTech

Table des matières

1	Introduction	1
2	Exploration des données	1
3	Modélisation à l'aide des méthodes de régression	2
3.1	Validation croisée à k blocs	2
3.2	Régression linéaire	2
3.3	Régression linéaire multiple	3
3.4	Régression ridge	3
3.5	Régression LASSO-LARS	3
3.6	Méthode de Nadaraya-Watson	4
3.7	Generalized Additive Models (GAM)	4
3.8	Bagging	5
3.9	CART	5
3.10	Random Forest	6
3.11	L2boosting	6
3.12	SVM	7
3.13	BP Neural Network	7
4	Comparaison de différentes méthodes	8
5	Prédictions pour le jeu de test	11

1 Introduction

L'objectif de ce projet est d'appliquer les connaissances acquises dans le cours afin de développer un modèle approprié pour prédire les prix des maisons situées dans le comté de King dans l'état de Washington. Les bases de données initiales contenaient 20 variables qui sont : l'identifiant de commande, la date de la transaction, la taille et nombre de pièces, de salles de bains et de jardins, le nombre d'étages, la vue, la décoration, le code postal, la latitude et la longitude. Ainsi, nous avons considérés que notre base de donnée est assez riche et que les variables sont bien adéquates pour une analyse future.

Chemin d'approche

Tout d'abord, afin de s'impliquer la modélisation du modèle, nous avons traité les variables qualitatives en les donnant des valeurs numériques correspondantes. Nous allons ensuite élaborer l'exactitude des 12 modèles de régression (la régression linéaire, la régression multiple, la régression ridge, la régression LASSO-LARS, la méthode de Nadaraya-Watson et la régression de GAM kernel, Bagging, CART, Random Forest, L2boosting, SVM, BP Neural Network) en comparant leurs différents résultats par plusieurs méthodes comme la validation croisée et l'utilisation des valeurs RMSE et MAPE.

2 Exploration des données

En premier temps, nous importons le jeu d'apprentissage et de test *Traindata* et *Testdata* respectivement en csv. Nous commençons par tracer Nuages de points avec la fonction *pd.plotting.scattermatrix* pour observer les caractéristiques et les relations des données.

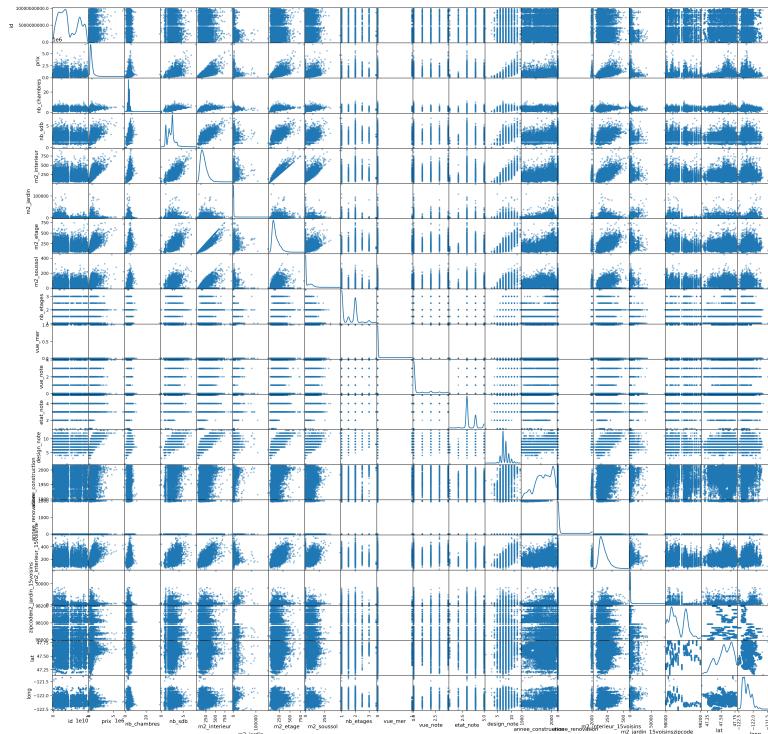


FIGURE 1 – Matrice du nuage de points

Ensuite, nous supprimons les points de données avec des prix anormaux en utilisant un tracé de ligne de boîte. Nous avons trouvé de nombreuses valeurs aberrantes pour la superficie du jardin et le nombre de pièces, par exemple un jardin de 150 000 pieds carrés et une maison de 33 pièces. Nous avons donc nettoyé les données et supprimé les aberrations.

Nous avons ensuite fusionné l'échantillon d'apprentissage et l'échantillon test pour faciliter le traitement ultérieur des données.

Ailleurs, on remarque la durée d'utilisation est un facteur assez important pour le prix de la maison. Ainsi, nous transformons les données de date dans l'ensemble de données en les divisant par des chaînes de caractères. Cette durée est considérée comme le temps entre la date de la vente et la date de la rénovation la plus récente. S'il n'a jamais été rénové depuis sa construction, la date limite d'utilisation est égale à la date de la vente moins la date de la construction. Nous les avons converti ensuite en des chiffres Float et supprimé les variables date de vente, date de construction, date de rénovation, etc.

Nous constatons que le prix d'une maison est toujours lié à son lieu d'implantation. Afin de traiter les codes postaux qui sont des variables qualitatives, nous utilisons la méthode de One-Hot Encoding pour les convertir en variables quantitatives. Nous divisons les prix moyens de chaque code postal en dix groupes. Les codes postaux ensuite sont remplacés par les noms de groupe et les dix noms de groupe sont ajoutés à la variable caractéristique. Si le code postal d'un point de données appartient à un groupe (à l'intérieur de ce code postal), la valeur de la variable pour ce groupe est 1, sinon elle est 0. De cette façon, nous avons possibilité de supprimer les variables de longitude, de dimension, de code postal spécifique, etc. et ajouté dix variables booléennes.

Nous avons alors séparé les variables de caractéristiques pour l'échantillon test et l'échantillon d'apprentissage. Nous avons normalisé les variables dans l'échantillon d'apprentissage en calculant la moyenne et la variance des variables et puis en utilisant la formule suivante : (variable - moyenne)/écart-type.

En outre, nous avons normalisé l'échantillon test avec la moyenne et la variance des variables de l'échantillon d'apprentissage. La formule est la suivante : (variable de l'échantillon test - moyenne de l'échantillon d'apprentissage)/écart-type de l'échantillon d'apprentissage.

3 Modélisation à l'aide des méthodes de régression

3.1 Validation croisée à k blocs

Nous avons défini la fonction de validation croisée à k blocs pour l'évaluation ultérieure de différents modèles de régression.

3.2 Régression linéaire

Nous prenons une validation croisée à k=5 blocs pour diviser les données en l'échantillon d'apprentissage et l'échantillon test. Nous avons ensuite appliqué la méthode de Régression linéaire à l'échantillon d'apprentissage. On a choisi la variable unique "m2_interieur", comme la r. et obtenu les paramètres du modèle en ajustant l'échantillon d'apprentissage. Ces paramètres ont ensuite été appliqués pour donner la prédiction.

En outre, pour chaque validation croisée, nous avons obtenu les valeurs RMSE et MAPE. Nous avons fait la moyenne des 5 valeurs RMSE des 5 validations croisées et nous avons obtenu : Moyenne de RMSE = 164255.92. Nous avons fait la moyenne des 5 valeurs MAPE des 5 validations croisées et le résultat est : Moyenne de MAPE = 32.91 .

3.3 Régression linéaire multiple

La régression linéaire multiple est une généralisation de la régression linéaire simple, dans le sens où cette approche permet d'évaluer les relations linéaires entre une variable réponse et plusieurs variables explicatives (de type numérique ou catégoriel).

Nous prenons une validation croisée à $k=5$ blocs pour diviser les données en l'échantillon d'apprentissage et l'échantillon test. Nous avons ensuite appliqué la méthode de Régression linéaire multiple à l'échantillon d'apprentissage et obtenu les paramètres du modèle en ajustant l'échantillon d'apprentissage. Ces paramètres ont ensuite été appliqués pour donner la prédiction.

En outre, pour chaque validation croisée, nous avons obtenu les valeurs RMSE et MAPE. Nous avons fait la moyenne des 5 valeurs RMSE des 5 validations croisées et nous avons obtenu : Moyenne de RMSE = 91973.34. Nous avons fait la moyenne des 5 valeurs MAPE des 5 validations croisées et le résultat est : Moyenne de MAPE = 15.70.

3.4 Régression ridge

La régression Ridge est une méthode d'estimation des coefficients des modèles de régression multiple dans des scénarios où des variables linéairement indépendantes sont fortement corrélées. Elle a été utilisée dans de nombreux domaines, notamment l'économétrie, la chimie et l'ingénierie.

Ici, en prenant 5 blocs ($k = 5$), pour diviser les données en l'échantillon d'apprentissage et l'échantillon test, nous avons utilisé "RidgeCV" et obtenu le paramètre de régularisation optimal α par validation croisée par pas de 1 de 10 à 4000. Nous avons ajusté l'échantillon d'apprentissage avec le paramètre optimal α obtenu, puis nous l'avons appliquée à l'échantillon test pour la prédiction, et nous avons obtenu les résultats de la prédiction.

En outre, pour chaque validation croisée, nous avons obtenu les valeurs RMSE et MAPE et leurs valeurs moyennes : Moyenne de RMSE = 91965.86 et Moyenne de MAPE = 15.69.

3.5 Régression LASSO-LARS

L'analyse de régression linéaire est la corrélation entre deux variables, à savoir la variable indépendante et la variable indépendante. La corrélation entre les variables ne consiste pas seulement en deux variables, mais il peut y avoir une corrélation entre trois variables ou plus appelée régression linéaire multiple. L'analyse de régression linéaire multiple comporte de nombreuses variables indépendantes, il existe donc une corrélation entre deux ou plusieurs variables indépendantes. Cette variable indépendante corrélée est appelée multicolinéarité. Réduire la multicolinéarité et augmenter la précision des modèles de régression linéaire peuvent utiliser la méthode LASSO (Least Absolute Shrinkage and Selection Operator). LASSO est une méthode d'analyse de régression qui effectue à la fois la sélection et la régularisation des variables afin d'améliorer la précision de la prédiction et l'interprétabilité du modèle statistique qu'elle produit.

Ici, nous gardons le même nombre de bloc $k=5$. Nous avons respectivement transformé X et Y de l'échantillon d'apprentissage dans le format de tableau requis par la fonction "lars_path" et obtenu les chemins de Lasso et leurs images selon les paramètres de régularisation calculés à l'aide de l'algorithme LARS en utilisant "lars_path" (cf. FIGURE 2).

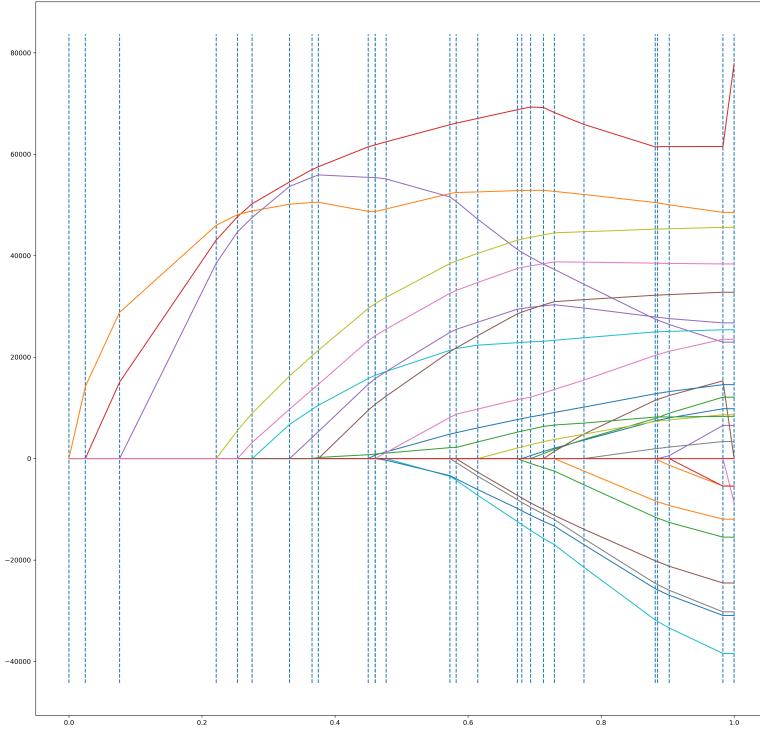


FIGURE 2 – les chemins de Lasso

Nous avons appliqué "LassoLarsCV" pour ajuster les paramètres et appliqué les paramètres du modèle obtenu à l'échantillon test pour la prédiction. En utilisant la même méthode des parties précédentes, on a obtenu : Moyenne de RMSE = 91965.33 et Moyenne de MAPE = 15.69.

3.6 Méthode de Nadaraya-Watson

La régression de Nadaraya-Watos (NW) apprend une fonction non linéaire en utilisant une moyenne pondérée par le noyau des données. Le montage NW peut être effectué sous forme fermée et est généralement très rapide. Cependant, le modèle appris n'est pas parcimonieux et souffre donc au moment de la prédiction.

Nous avons alors utilisé la régression à noyau avec le nombre de bloc =5. Comme chaque élément de la régression à noyau est une variable distincte, un trop grand nombre d'entre elles entraînerait des opérations lentes, nous avons donc choisi la variable unique "m2_interieur", comme la régression linéaire.

De même façon, nous avons : Moyenne de RMSE = 163370.96 et Moyenne de MAPE = 32.74. C'est proche de ce que nous avons obtenu en régression linéaire univariée.

3.7 Generalized Additive Models (GAM)

Un GAM est un modèle linéaire avec une différence clé par rapport aux modèles linéaires généralisés tels que la régression linéaire. Un GAM est autorisé à apprendre des caractéristiques non linéaires. Pour ce faire, nous remplaçons simplement les coefficients bêta de la régression linéaire par une fonction flexible qui permet des relations non linéaires (nous verrons les mathématiques plus tard). Cette fonction flexible est appelée spline. Les splines sont des fonctions complexes qui nous permettent de modéliser des relations non linéaires pour chaque entité. La

somme de nombreuses splines forme un GAM. Le résultat est un modèle très flexible qui a encore une partie de l'explicabilité d'une régression linéaire.

En prenant $k = 5$, nous avons ajusté une spline à chacune des variables caractéristiques et avons additionné les résultats obtenus. Les valeurs moyennes de RSME et MAPE sont : Moyenne de RMSE = 80341.23 et Moyenne de MAPE = 13.78.

3.8 Bagging

L'algorithme de bagging (en anglais : Bootstrap aggregating) est un algorithme d'apprentissage de groupe dans le domaine de l'apprentissage automatique, qui peut être combiné avec d'autres algorithmes de classification et de régression pour améliorer leur précision et leur stabilité, tout en évitant le sur-apprentissage en réduisant la variance des résultats.

L'aspect le plus important de l'algorithme de mise en sac est celui des estimateurs. $n_estimator$ est le nombre d'estimateurs de base, plus il y en a, mieux c'est, mais plus cela prend de la mémoire, plus l'algorithme est lent. À partir d'une certaine valeur, l'effet de régression cesse progressivement de s'améliorer de manière significative. Afin d'équilibrer le temps et la précision, 200 a été choisi. Les graphiques montrent les résultats de RMSE et MAPE pour différents nombres d'estimateurs.

Bagging		
n_estimator	RMSE	MAPE
50	76093.32563	12.11808874
100	75891.58553	12.07189808
150	75571.60681	11.9983804
200	75514.77974	12.00264312
250	75455.73393	11.99439212
300	75481.54335	12.00205274
350	75360.71327	11.97865979

FIGURE 3 – Comparaison des différents résultats selon le nombre des estimator

3.9 CART

L'algorithme du cart est une méthode d'apprentissage qui produit une distribution de probabilité conditionnelle d'une variable aléatoire Y étant donné une variable aléatoire d'entrée X. Il suppose que l'arbre de décision est un arbre binaire avec des nœuds internes dont les caractéristiques ne peuvent prendre que les valeurs oui et non, ce qui le divise en deux branches, gauche et droite. Pour ce faire, il divise l'espace des caractéristiques de la variable d'entrée en un nombre fini de pavés et détermine la distribution de probabilité prédictive sur ces pavés.

Tout d'abord, nous devons construire l'arbre maximal à l'aide d'une procédure forward. La profondeur maximale de l'arbre est de 10. Après, on doit créer une suite imbriquée d'arbres, de complexité décroissante. On choisit de la meilleure profondeur, qui minimise l'erreur MSE. Enfin, il faut élaguer les branches inutiles (déterminer l'arbre optimal).

En conséquence, pour l'algorithme CART, nous avons obtenu un score RMSE de 96007 et un score MAPE de 15,29.

3.10 Random Forest

Parmi les méthodes de Machine Learning, Les Forêts aléatoires ou Random Forest en anglais, sont une méthode la plus utilisée grâce à leur précision, leur facilité d'utilisation et leur robustesse. Elle fournissent deux méthodes pour la sélection de variables : la diminution moyenne de l'impureté et la diminution moyenne de précision. Random forest est un cas particulier de la méthode dite de Bagging.

Dans cette partie, afin de déterminer le nombre des arbres dans la forêt (n_estimators), on a supposé que ce nombre soit grand, qui vaut $50*i$ arbres. Après avoir défini une fonction de la ligne de régresseur, on compare les erreurs de extbf RSME et de extbf MAPE. Les résultats obtenus sont donnés dans la figure suivante :

Random Forest		
n_estimator	RMSE	MAPE
50	76308.76743	12.11106171
100	75848.14894	12.04459353
150	75607.47615	12.01344925
200	75527.1383	12.00052664
250	75518.60307	12.00288892
300	75496.36633	11.99913149
350	75472.28982	11.99173437

FIGURE 4 – Comparaison des différents résultats selon le nombre des arbres

On peut voir que quand les erreurs de MAPE sont quasiment les mêmes, celles de RSME varient beaucoup. Nous avons donc choisi extbf 200 arbres qui nous donnent des valeurs d'erreurs assez faibles.

3.11 L2boosting

Un algorithme L2 boosting pour l'estimation d'une fonction de régression à partir d'un plan aléatoire est présenté, qui consiste à ajuster de façon répétée une fonction d'un espace fonctionnel non linéaire fixe aux résidus des données par les moindres carrés et à définir l'estimation comme une combinaison linéaire des estimations des moindres carrés qui en résultent. Le fractionnement de l'échantillon est utilisé pour décider après combien d'itérations de lissage des résidus l'algorithme se termine. Le taux de convergence de l'algorithme est analysé dans le cas d'une variable réponse non bornée. La méthode est utilisée pour ajuster une somme de maxima ou de minima de fonctions linéaires à un ensemble de données donné et est comparée à d'autres estimations de régression non paramétrique à l'aide de données simulées.

Ici, on souhaite déterminer la valeur de n_estimators, la limite supérieure des estimateurs à laquelle le boosting est terminé (avec une valeur par défaut de 50). On procède d'abord en cherchant la profondeur optimale en utilisant la fonction *GradientBoostingRegressor(random_state = 0)*. Ensuite, on continue en calculant les erreurs pour différentes profondeurs de l'arbre. Le tableau suivant présente les résultats que nous avons obtenus.

L2Boosting							
n_estimators	n_iter_no_change	validation_fraction	learning_rate	max_depth	RMSE	MAPE	
50	50	1	1	8	108159.4!	17.45410417	
50	50	1	0.01	8	73125.96!	11.74539195	
50	50	1	0.05	8	145230.4!	29.82216141	
50	50	1	0.1	8	78274.40!	13.1151161	
50	50	0.1	0.1	8	73509.03!	11.7700737	
50	50	0.5	0.1	8	76563.35!	12.23469623	
50	50	0.3	0.1	8	74751.11!	11.96323954	
50	100	0.3	0.1	8	74751.11!	11.96323954	
50	10	0.3	0.1	8	74751.11!	11.96323954	
100	50	0.3	0.1	8	73682.91!	11.7424493	
200	50	0.3	0.1	8	73621.71!	11.73452008	
300	50	0.3	0.1	8	73621.71!	11.73452008	
200	50	0.1	0.1	8	72494.81!	11.53368203	

FIGURE 5 – Compaison des différents résultats obtenus selon le n_estimators

Le n_estimators choisi devrait celui qui donne les erreurs de RMSE et de MAPE les plus faibles. Ici, un n_estimators de l'ordre de extbf 200 est plus adaptable.

3.12 SVM

Nous prenons une validation croisée à k=5 blocs pour diviser les données en l'échantillon d'apprentissage et l'échantillon test. Nous avons appliqué SVR (Support Vector Regression). Pour la fonction noyau, nous avons choisi la fonction "rbf" et pour le gamma, nous avons choisi "auto". Pour la sélection des paramètres "C" et "epsilon", nous avons constaté que plus la valeur de C est grande et plus la valeur d'epsilon est petite, meilleur est le résultat de la régression. Après avoir sélectionné les valeurs actuelles (C=1000000, epsilon = 0.00000001), nous avons constaté que si nous continuons à augmenter C et à diminuer epsilon, la qualité de la régression ne s'améliore pas de manière significative et le temps de calcul devient nettement plus long.

Pour chaque validation croisée, nous avons obtenu les valeurs RMSE et MAPE. Nous avons fait la moyenne des 5 valeurs RMSE et MAPE des 5 validations croisées et nous avons obtenu : Moyenne de RMSE = 78714.99, Moyenne de MAPE = 12.43.

Le résultat final montre que le score Kaggle du SVM (222792.94) est mauvais, ce qui peut être dû à un sur-ajustement causé par une valeur C trop grande et une valeur epsilon trop petite.

3.13 BP Neural Network

D'abord, nous avons converti le format des données en format tenseur. Nous avons ensuite défini la fonction de perte et le réseau (un réseau linéaire avec quatre couches cachées). Nous avons appliqué l'optimiseur Adam, qui contient des informations sur le réseau, le taux d'apprentissage, la désintégration des poids. Nous avons essayé différentes combinaisons pour la sélection des paramètres du nombre d'apprentissage (epoch), du taux d'apprentissage, de la désintégration des poids et de la taille des lots. Nous avons constaté que trop de nombre d'apprentissage (epoch=200) conduisait à un sur-ajustement et à des erreurs plus importantes (cf. FIGURE 6). Le taux d'apprentissage est sensible, trop grand peut conduire à ne pas trouver la solution optimale, trop petit peut conduire à tomber dans un optimum local. Les effets de la taille du lot et de la désintégration du poids sont relativement faibles. L'augmentation de la taille du lot réduit les erreurs, et un réglage raisonnable de la désintégration du poids réduisant le sur-ajustement.

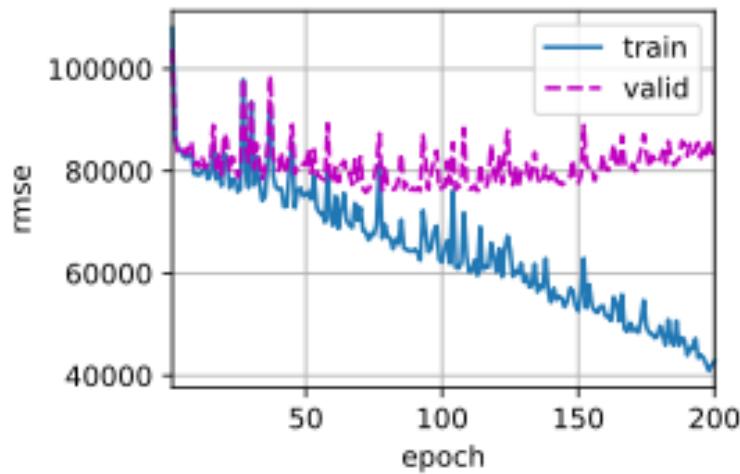


FIGURE 6 – l'effet du nombre d'apprentissage (epoch)

Finalement, nous avons choisi "nombre d'apprentissage" (epoch) = 100, "taux d'apprentissage" = 0.1, "désintégration des poids" = 0, "taille des lots" = 256. Nous avons fait la moyenne des 5 valeurs RMSE et MAPE des 5 validations croisées et nous avons obtenu : Moyenne de RMSE = 77891.23, Moyenne de MAPE = 12.27.

4 Comparaison de différentes méthodes

Nous allons suivre dans cette partie l'évaluation des 12 méthodes utilisées. Leur différence en fonction de la valeur moyenne de RMSE obtenues est présentée dans la FIGURE 7 et la FIGURE 8.

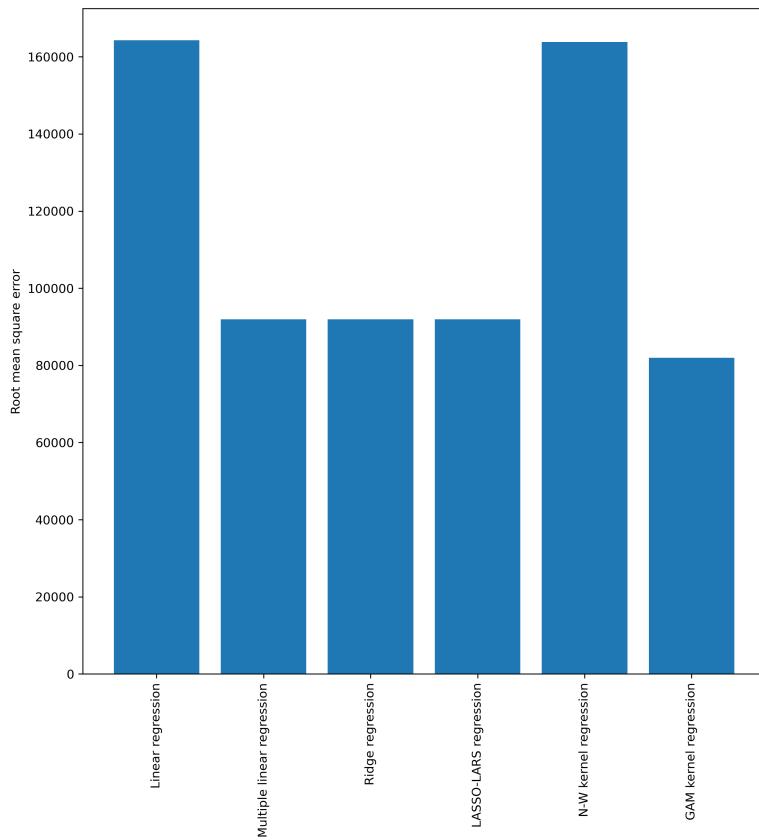


FIGURE 7 – Comparaison de Moyenne de RMSE - 1

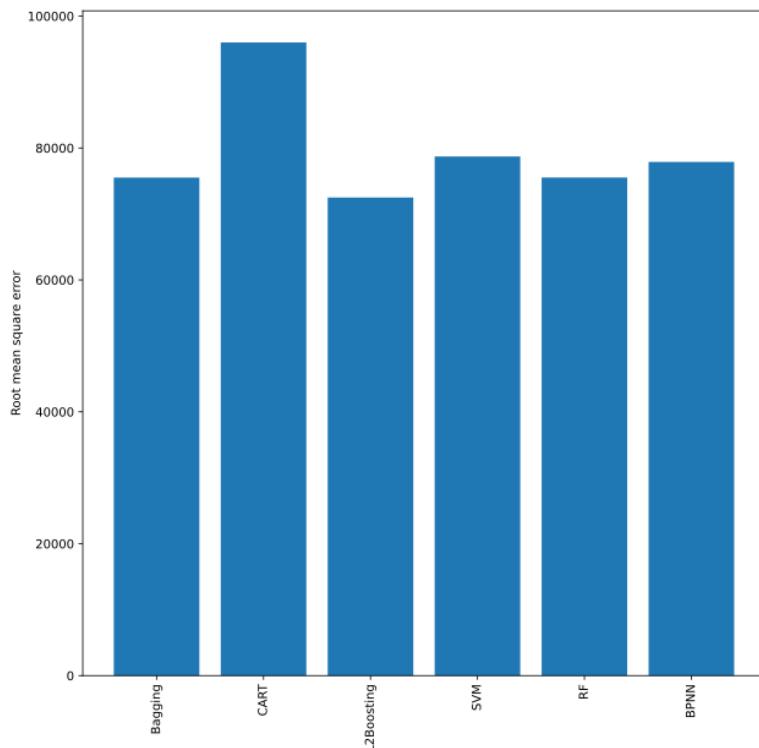


FIGURE 8 – Comparaison de Moyenne de RMSE - 2

Nous pouvons observer que la méthode "L2boosting" présente une valeur moyenne de

RMSE la plus faible. A contrario, cette valeur est plus importante dans la Méthode de Nadaraya-Watson et la "Régression linéaire". Les Moyennes de RMSE obtenues en appliquant respectivement la "Régression linéaire multiple", la "Régression ridge", la "Régression LASSO-LARS", "CART" sont similaires (environ de 92000 à 96000). Les Moyennes de RMSE de "GAM", "Bagging", "Random Forest", "SVM" et "BP Neural Network" sont similaires (environ de 76000 à 80000), et légèrement supérieures à la Moyenne de RMSE obtenue en appliquant la méthode "L2boosting".

La comparaison des valeurs moyennes de MAPE obtenues en appliquant les 12 modèles différents ci-dessus est présentée dans la FIGURE 9 et 10.

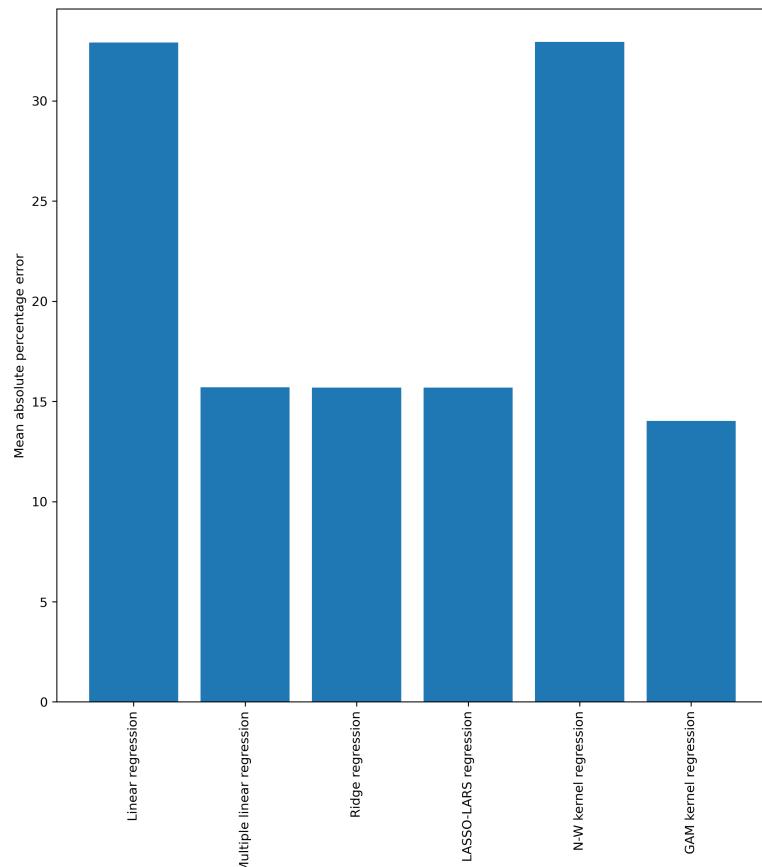


FIGURE 9 – Comparaison de Moyenne de MAPE - 1

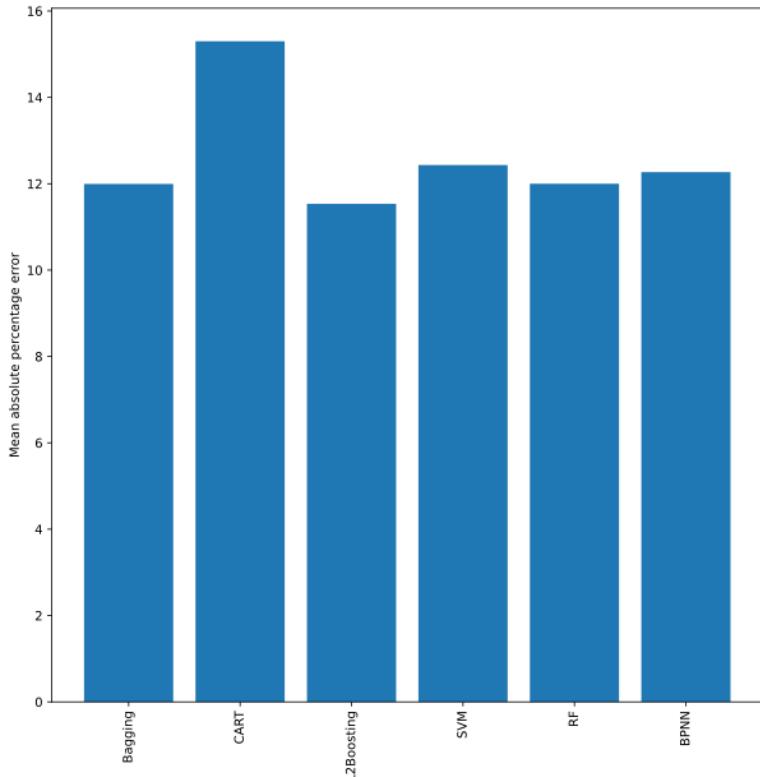


FIGURE 10 – Comparaison de Moyenne de MAPE - 2

Comme la figure nous montre, la valeur moyenne de MAPE est plus faible dans le modèle de "L2boosting" et plus élevée dans le modèle de Nadaraya-Watson et la "Régression linéaire". Les autres méthodes nous donnent les valeurs assez similaires.

En conclusion, parmi les 12 méthodes, "L2boosting" nous permet d'obtenir la Moyenne de RMSE et la Moyenne de MAPE les plus faibles.

5 Prédictions pour le jeu de test

Après l'optimisation des paramètres, nous avons appliqué chacun des douze modèles entraînés respectivement à la prédiction du jeu de test pour obtenir les résultats de prédiction finaux. Et les scores obtenus sur la plateforme Kaggle sont indiqués dans la TABLE 1.

Méthode	Score
Régression linéaire	261607
Régression linéaire multiple	195522
Régression ridge	195621
Régression LASSO-LARS	195561
Méthode de Nadaraya-Watson	272017
GAM	221269
Bagging	205376
CART	220383
Random Forest	204781
L2boosting	197481
SVM	222792
BP Neural Network	193849

TABLE 1 – Les scores sur Kaggle