Université Claude Bernard Lyon 1

Master 1

Mathématiques appliquées, statistique

Projet en mathématiques appliquées

Forêts Aléatoires

Étudiants

Baptiste BIAUSQUE

Martine VODOUNON

Enseignant encadrant

Anne-Laure FOUGERES

28 avril 2022

Table des matières

Table des figures

Liste des tableaux

1 Introduction

Les forêts aléatoires sont des méthodes d'ensembles non paramétriques pour l'estimation et la prévision, décrites pour la première fois dans leur forme générale en 1995 par Tin Kam Ho [1], alors informaticien chez IBM. Elles s'appuient sur des arbres de décision qui effectuent des séparations dans l'espace des données d'entrée, en vue de formuler des prévisions sur une variable cible.

Les arbres décisionnels sont utilisés dans de très nombreux domaines, scientifiques comme industriels, aussi bien pour prédire une quantité que pour classer des ensembles de données. Ils sont surtout connus sous leur forme *CART*[1], introduite dans Breiman *et al.* [2] en 1984. Il s'agit de la forme d'arbre à laquelle nous nous intéressons dans ce rapport, bien qu'il en existe d'autres comme *ID3* et son successeur *C4.5*, ou encore *MARS*.

Les arbres *CART* ont été développés en vue de répondre à des critiques formulées à l'encontre des techniques pré-existantes d'arbres décisionnels. Ils se distinguent par leur caractère binaire et leur méthode de séparation des nœuds que nous détaillerons. Étant donné leurs défauts, nous expliquerons pourquoi et comment les arbres sont agrégés en forêts de manière à fournir des résultats satisfaisants.

Enfin, sur la base de deux jeux de données, nous illustrerons des résultats de prévision obtenus sur `R` en régression et en classification. Puis nous comparerons les résultats à ceux obtenus par une technique de réseaux de neurones. À cette occasion, nous détaillerons des techniques d'optimisation de quelques paramètres utilisés par les bibliothèques de manière à affiner la qualité des prévisions.

L'essentiel des explications de la partie théorique de ce rapport s'appuie sur l'ouvrage de Poggi et Genuer (2019) [3], ainsi que sur l'exposé présenté par ces mêmes auteurs, à l'occasion des Journées d'Étude en Statistique 2016, de la Société Française de Statistique [2].

1. *Classification And Regression Trees* (littéralement « arbres de classification et de régression »)
2. Le support de la présentation peut être trouvé à l'adresse suivante :
`http://www.nathalievialaneix.eu/teaching/jes2016/docs/poggi_JES2016-cart_rf.pdf`

2 Arbres *CART*

Les arbres *CART*, comme toutes les autres méthodes d'arbres, visent à fournir des prédicteurs, c'est-à-dire des fonctions qui associent aux données une prédiction concernant les valeurs probables d'une variable cible. Précisons tout de suite le cadre de l'estimation ainsi que les notations utilisées.

2.1 Contexte et notations

Considérons un ensemble de données composé de $p+1$ variables avec $n+k$ observations [3], parmi lequel nous souhaitons prédire la valeur que prendra une variable en fonction des p autres. L'objectif de la démarche est de construire un modèle effectuant cette association.

2.1.1 Échantillons d'apprentissage et de validation

Une pratique courante consiste à séparer l'ensemble en deux échantillons : le premier servant à l'élaboration du modèle, et le second pour l'évaluer. Ainsi, l'on peut apprécier la capacité du modèle à appréhender des données qui lui sont *a priori* inconnues.

Notons $\mathcal{L}_{n+k}$ l'ensemble des données. Pour la suite, nous considérerons exclusivement l'échantillon d'apprentissage $\mathcal{L}_n$ constitué de n observations, puisque nous nous intéressons à la construction du modèle. Généralement, l'ensemble d'apprentissage contient de l'ordre de 80% des données : c'est la répartition que nous emploierons pour les applications.

2.1.2 Modèles, estimation et prédiction

Concernant les variables de l'ensemble de données, celles-ci se décomposent en deux catégories : les p variables dites « explicatives » sont les coordonnées du vecteur d'entrée $X \in \mathbb{R}^p$, et la variable cible à prédire – aussi appelée « sortie »–, que nous notons Y. Selon la nature qualitative ou quantitative (respectivement) de la variable cible, le problème relève soit de la classification soit de la régression (respectivement). La forme du modèle dépend donc de la nature du problème étudié.

Nous distinguons également le travail d'estimation de celui de la prédiction. L'estimation consiste à identifier et expliciter le lien fonctionnel entre les variables explicatives et la variable cible pour des observations connues. La prédiction, elle, est centrée sur la recherche d'une valeur probable, mais inconnue formellement, pour la variable cible à partir de valeurs connues des variables explicatives.

Modèle de classification

Soit $\{1, \dots, L\}$ l'ensemble des modalités de Y. La classification est effectuée par vote majoritaire sur les probabilités pour Y d'appartenir à chacune des classes, conditionnellement aux valeurs prises par X. On écrit le prédicteur $\hat{h}$ associé au modèle :

$$\begin{array}{rccc} \hat{h}: & \mathbb{R}^p & \to & \{1, \dots, L\} \\ & x & \mapsto & \underset{\ell \in \{1,\dots,L\}}{\arg\max}\, \mathbb{P}\left(Y = \ell | X = x\right). \end{array}$$

3. n observations pour l'apprentissage et k pour la validation du modèle, cf. section 2.1.1

Notons que les probabilités d'appartenance à chacune des classes ne sont pas à valeurs binaires, c'est-à-dire qu'elles ne sont pas égales à 1 sur une classe et 0 sur toutes les autres classes. Nous préciserons ce détail dans la partie 2.1.3.

Modèle de régression

Le modèle s'écrit sous la forme d'une fonction de régression inconnue η appliquée aux données d'entrée et d'un bruit ϵ :

$$Y = \eta(X) + \epsilon.$$

Contrairement à un modèle paramétrique comme la régression linéaire, η n'est pas contrainte à une forme pouvant se décrire via un nombre fini de paramètres.

2.1.3 Évaluation de l'erreur de prédiction

L'erreur de prédiction s'entend comme l'erreur commise lors de l'application du prédicteur à des données inconnues du modèle à sa construction. Il s'agit de l'erreur de généralisation du modèle. En effet, une problématique commune à toutes les procédures d'apprentissage automatique est le sur-apprentissage, c'est-à-dire la construction d'un modèle trop spécifique à l'échantillon d'apprentissage, créant une erreur systématique de généralisation.

Nous cherchons donc à minimiser cette erreur pour maximiser l'employabilité du prédicteur sur de nouvelles données. Son expression dépend de la nature du problème considéré.

Pour la classification

La métrique considérée pour l'erreur de généralisation en classification est la proportion de mauvais classements :

$$\mathbb{E}\left[\mathbb{1}_{\hat{h}(X)\neq Y}\right] = \mathbb{P}\left(\hat{h}(X) \neq Y\right).$$

Pour la régression

L'erreur de généralisation d'un modèle de régression est son erreur quadratique de prévision :

$$\mathbb{E}\left[\left(\hat{h}(X) - Y\right)^2\right]$$

Nous avons maintenant tous les éléments nécessaires pour construire les arbres *CART*.

2.2 Construction des arbres

Les arbres *CART* ont une spécifité dans leur physionomie. Comme la plupart des arbres de décision, ils sont « enracinés » : ils démarrent à partir d'un nœud unique considéré comme leur racine. Puis chaque nœud est séparé en plusieurs branches. La particularité de *CART* est que ses séparations sont binaires : il y a toujours exactement deux séparations à chaque nœud. Les derniers nœuds de l'arbre – ceux qui ne sont pas séparés en branches – sont appelés les « feuilles ». Elles indiquent les prédictions associées aux entrées.

Le processus de construction d'un arbre *CART* se fait en deux étapes. La première consiste à développer au maximum un arbre qui enchaîne les séparations des données d'entrée. La seconde est l'élagation des branches les moins utiles au regard d'un critère donné, de manière à simplifier l'arbre.

2.2.1 Développement de l'arbre maximal

La construction de l'arbre maximal est un processus itératif. À chaque étape, le sous-ensemble des entrées que constitue le nœud parent considéré est partitionné. Ces partitions sont appelées « découpes ». En notant X^j la $j^{ème}$ variable d'entrée, une découpe s'écrit sous les formes :

$\{X^j = \ell\} \cup \{X^j \neq \ell\}, j \in \{1, \dots, p\}, \ell \in \{1, \dots, L\}$ pour des variables qualitatives ; et
$\{X^j \leqslant d\} \cup \{X^j > d\}, j \in \{1, \dots, p\}, d \in \mathbb{R}$ pour des variables quantitatives.

L'enjeu de la procédure est la recherche d'une découpe optimale, c'est-à-dire d'une partition qui minimise une fonction de coût particulière. Une autre distinction des arbres *CART* concerne le choix de ces fonctions. Soient t le nœud parent, et respectivement t_L et t_R ses deux nœuds fils issus de la découpe :

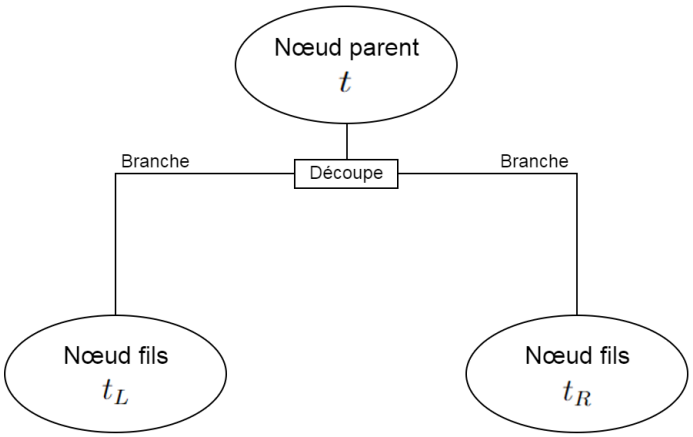

FIGURE 1 – Schéma de la découpe d'un nœud d'un arbre *CART*

Pour la classification

Soit $\hat{p}_t^\ell$ la proportion d'observations présentes dans le nœud t et appartenant à la classe ℓ. Alors la fonction de pureté de Gini est définie par $\Phi : t \mapsto \sum_{\ell=1}^{L} \hat{p}_t^\ell \left(1 - \hat{p}_t^\ell\right)$: cette fonction est proportionnelle à l'hétérogénéité des observations. La découpe optimale est celle qui maximise la quantité :

$$\Phi(t) - \left(\frac{\#t_L}{\#t} \Phi(t_L) + \frac{\#t_R}{\#t} \Phi(t_R) \right).$$

Ceci revient à chercher la découpe qui rend les plus homogènes possibles les sous-échantillons introduits dans chacun des deux nœuds fils, ce qui traduit l'idée recherchée de séparation binaire de l'espace des variables d'entrée.

Pour la régression

Dans les modèles de régression, les découpes optimales sont celles qui minimisent les variances des sous-échantillons dans chaque nœud fils, et plus précisément la quantité suivante :

$$\frac{\#t_L}{\#t}\mathrm{Var}\left(t_L\right) + \frac{\#t_R}{\#t}\mathrm{Var}\left(t_R\right).$$

Dans chaque cas, la stratégie de découpe ne fait pas intervenir la qualité de l'estimation, ceci afin d'éviter un sur-apprentissage évident du fait de la personnalisation de l'arbre à l'ensemble d'apprentissage $\mathcal{L}_n$.

L'arbre maximal, noté T_{max}, est obtenu à l'issue de ce processus, lorsqu'une condition d'arrêt préalablement définie – portant généralement sur le nombre d'observations restantes dans un nœud ou sur la taille de l'arbre – est atteinte. Chaque nœud t se voit assigner une valeur : la modalité de la classe majoritaire en classification, et $\overline{Y_t}$ en régression.

2.2.2 Élagage de l'arbre

L'arbre T_{max}, malgré les précautions prises dans les stratégies de découpe des nœuds, souffre souvent d'une très grande complexité et d'un sur-apprentissage chronique. Pour faire l'analogie avec l'estimation paramétrique, le biais de T_{max} est très faible mais la variance de ses prédictions est importance. L'élagage permet de résoudre en partie ces problèmes en simplifiant T_{max}, de manière à obtenir un arbre moins spécifique à $\mathcal{L}_n$, donc plus généralisable.

Principe

L'élagage consiste à construire, à partir de T_{max}, une suite de sous-arbres imbriqués $(T_i)_i$ en élaguant à chaque étape les branches associées à la découpe la moins utile au regard d'un critère de pénalisation à minimiser.

Le critère valorise l'ajustement du modèle aux données d'apprentissage et pénalise sa taille, qui symbolise sa complexité. En notant $|T|$ le nombre de feuilles d'un arbre T, il s'écrit sous la forme :

$$crit_\alpha\left(T\right) = \overline{err}\left(T\right) + \alpha|T|, \alpha \geqslant 0.$$

α est un coefficient de pénalisation choisi par l'utilisateur : il permet de faire la balance entre la fonction d'erreur et la complexité du modèle. Par le biais de α, l'utilisateur peut décider d'accorder à un critère plus de poids qu'à l'autre.

$\overline{err}\left(T\right)$ mesure l'erreur d'ajustement de l'arbre T aux données d'apprentissage. Sa forme varie selon le contexte du problème :

$$\overline{err}\left(T\right) = \frac{1}{n}\sum_{\{t \text{ feuille de } T\}}\sum_{(X_i,Y_i)} \mathbb{1}_{\{\hat{h}(x_i)\neq y_i\}} \text{ en classification ; et}$$

$$\overline{err}\left(T\right) = \frac{1}{n}\sum_{\{t \text{ feuille de } T\}}\sum_{(X_i,Y_i)} \left(Y_i - \overline{Y_t}\right)^2 \text{ en régression.}$$

où X_i et Y_i sont respectivement l'entrée et la sortie de la $i^{ème}$ observation.

Le dernier arbre obtenu est l'arbre minimal, avec une seule feuille : cet arbre est fortement biaisé, mais avec une faible variance de prédiction. Il est donc très peu spécifique, peu précis, mais stable dans ses prévisions.

Explication de l'algorithme

Soit T_t la portion de l'arbre issue d'un nœud t bien choisi. Afin de comprendre l'intérêt et le fonctionnement de l'algorithme d'élagage, nous remarquons que l'erreur d'ajustement obtenue en suivant T_t est meilleure que celle obtenue en élaguant l'arbre au nœud t. En effet, t est choisi de telle manière que l'égalité suivante soit vraie pour un $\alpha \geqslant 0$ donné :

$$\begin{aligned} crit_\alpha(t) &= crit_\alpha(T_t) \\ \overline{err}(t) + \alpha &= \overline{err}(T_t) + \alpha|T_t| \\ \text{d'où } \overline{err}(t) - \overline{err}(T_t) &= \alpha(|T_t| - 1) \geqslant 0. \end{aligned}$$

Cela signifie qu'en élaguant la portion T_t, l'arbre perd généralement en qualité d'ajustement. L'enjeu est donc de trouver le nœud t qui minimise α.

Pour ce faire, l'algorithme calcule, à chaque itération, le nœud pour lequel un élagage de la branche maximise la simplification de l'arbre issu de la précédente itération, tout en minimisant la perte d'ajustement subie. Cela revient à chercher, à la $k^{ème}$ itération, un α_k qui minimise la quantité suivante :

$$\min_{\{t \text{ nœud interne de } T\}} \frac{\overline{err}(t) - \overline{err}(T_t)}{|T_t| - 1}.$$

Nous obtenons, à l'issue de cette procédure, une suite de K arbres imbriqués $(T_k)_{1 \leqslant k \leqslant K}$ et une suite de poids $(\alpha_k)_{1 \leqslant k \leqslant K}$ strictement croissante [4]. En effet, au fur et à mesure des itérations, l'élagation de nouvelles branches se fait à un prix croissant : le gain de simplification est moins important au regard de la perte d'ajustement subie.

Ainsi, pour tout $1 \leqslant k \leqslant K-1$ et pour tout $\alpha \in [\alpha_k, \alpha_{k+1}[$, le sous-arbre qui minimise $crit_\alpha$ est un arbre issu de l'algorithme d'élagage [5]. Ce résultat est tout particulièrement intéressant, puisqu'il permet à l'utilisateur de réaliser un arbitrage sur la complexité limite souhaitée pour le modèle, et de trouver un arbre optimal qui maximise l'ajustement aux données compte tenu de cette contrainte. Dans ce cadre, il est courant sur logiciel de considérer les α_k comme les valeurs d'un paramètre de complexité noté `cp`.

2.3 Avantages et inconvénients

Les arbres *CART* présentent des avantages considérables par rapport à d'autres méthodes d'estimation et de prévision, avantages listés ci-dessous :

Polyvalence

D'abord, ils ne posent pas de contrainte sur la forme du prédicteur, ce qui leur permet de gérer des formes de dépendances que d'autres méthodes comme la régression linéaire ne permettent pas de modéliser.

4. Résultat théorique démontré par Breiman *et al.* [2]
5. Idem.

Gestion des valeurs manquantes

Ensuite, les arbres peuvent gérer des valeurs manquantes pour certaines variables explicatives. En effet, pour chaque nœud, l'ensemble des découpes possibles peut être ordonné selon leur capacité à réduire l'hétérogénéité de l'échantillon. Les découpes dites « concurrentes » à la découpe optimale sont appelées découpes de substitution. Ainsi, une observation à valeurs manquantes peut poursuivre le cheminement en utilisant une découpe de substitution et déduire une prédiction. Cette faculté des arbres de décision offre la possibilité d'éviter de recourir à des pratiques d'imputation de valeurs dont la validité théorique est parfois douteuse et qui peuvent polluer les données.

Faible sensibilité aux valeurs aberrantes

Une autre qualité concerne la gestion des valeurs aberrantes. La nature non paramétrique de l'estimation par arbres de décision rend la méthode moins sensible aux valeurs extrêmes que les méthodes d'estimation paramétrique.

Faible complexité

C'est également une méthode assez peu coûteuse à mettre en place, avec une faible complexité algorithmique allant de $\mathcal{O}(pn \ln n)$ pour un arbre équilibré, à $\mathcal{O}(pn^2)$ pour un arbre déséquilibré, en forme de « peigne ». Les premiers sont rapides à mettre en œuvre tandis que les seconds sont un peu plus coûteux en temps.

Facilité d'interprétation

Enfin, une grande qualité des arbres de décision est leur facilité d'interprétation : le cheminement de la racine à la feuille permet de lire très facilement tous les critères de séparation qui conduisent à la prédiction. Nous pouvons ainsi comprendre quels sont les éléments qui déterminent la prédiction, et même mesurer l'importance de chaque variable dans la prédiction. En effet, par construction, les premières découpes de l'arbre sont celles qui réduisent le plus l'hétérogénéité de l'échantillon, et les suivantes voient cette réduction décroître. Donc les variables des premières découpes peuvent être interprétées comme plus importantes que celles des dernières découpes.

Toutefois, il faut noter que cette mesure est biaisée, bien que nous pouvons utiliser les découpes concurrentes pour mieux apprécier l'importance des variables par cette méthode. En pratique, l'importance des variables est appréciée au travers de méthodes d'ensembles dans le cadre des forêts, plus que par la lecture d'un seul arbre.

Inconvénients

En contrepartie, les arbres de décision présentent un inconvénient de taille. En dépit des techniques d'élagage qui ne corrigent que partiellement ce problème, ils restent très spécifiques à l'ensemble des données d'apprentissage et instables dans leurs prédictions. Ils peuvent engendrer des erreurs de généralisation importantes, par un sur-apprentissage inhérent à leur nature. Cela limite sérieusement leur employabilité, en pratique, sur des données réelles.

Ce point noir majeur a toutefois pu être solutionné en appliquant des méthodes d'ensembles aux arbres de décision, dans le cadre des forêts aléatoires. Ce sont ces méthodes que nous allons maintenant détailler.

3 Forêts aléatoires

Les forêts sont une collection d'arbres de décision. Connaissant les points faibles des arbres *CART*, leur objectif est de fournir un ensemble de prédicteurs les plus variés possibles, afin de minimiser l'erreur de généralisation du modèle et compenser l'erreur des prédicteurs les moins pertinents. Le principe des forêts est le suivant : par des méthodes aléatoires, nous construisons un ensemble d'arbres ; ensuite, les prévisions de la forêt pour une observation donnée sont bâties comme l'agrégation des prédictions de chaque arbre pour cette entrée.

3.1 Construction des forêts

Pour pouvoir construire des arbres aux prévisions variées, nous avons besoin d'introduire des méthodes dites de « perturbation ». En effet, les arbres *CART* sont très sensibles aux fluctuations sur l'ensemble d'apprentissage $\mathcal{L}_n$, ce qui nous laisse plusieurs pistes à explorer. Pour la suite, nous appellerons « règle de base » la procédure de construction d'un arbre *CART* sur un échantillon de données d'apprentissage.

3.1.1 *Bagging*

La première méthode, et la plus connue, a été introduite par L. Breiman [4] : il s'agit de la ***b**ootstrap **agg**regat**ing*** – « agrégation *bootstrap* »–, généralement condensée dans l'expression *bagging* (que nous pourrions traduire par « ensachage »).

Cette méthode consiste à effectuer un tirage aléatoire, uniforme et avec remise, des observations dans $\mathcal{L}_n$, puis à appliquer la règle de base à l'échantillon *bootstrap* tiré. La procédure est répétée autant de fois que nous souhaitons construire d'arbres. Notons $\left(\mathcal{L}_n^{\Theta_j}\right)_j$ la suite des échantillons *bootstrap* tirés : ces échantillons sont appelés des « sacs » (*bags*). Les tirages sont indépendants et nous obtenons à la fin un prédicteur pour chaque échantillon $\mathcal{L}_n^{\Theta_j}$.

L'avantage principal de cette méthode est de construire facilement et rapidement une collection de prédicteurs variés sur tout $\mathcal{L}_n$. Le *bagging* a également un autre intérêt que nous développerons ultérieurement.

3.1.2 Entrées aléatoires

L'autre méthode la plus connue, et en pratique la plus utilisée, est celle des entrées aléatoires – « *random inputs* ». Elle s'appuie sur l'introduction par l'utilisateur d'un paramètre $m \leqslant p$.

Le procédé reprend le principe du *bagging* ; mais au lieu d'appliquer la règle de base sur toutes les variables de $\mathcal{L}_n^{\Theta_j}$, les découpes des nœuds sont effectuées sur un nombre restreint de variables d'entrée. Cela signifie que parmi les p variables explicatives, un tirage uniforme avec remise de m d'entre-elles est opéré, puis la découpe optimale est

choisie. Ainsi, nous obtenons des prédicteurs plus variés encore que ceux obtenus par le *bagging*.

Une étude de Genuer *et al.* [6] suggère des valeurs à préférer pour le paramètre m : $\sqrt{p}$ pour des problèmes de classification, et $p/3$ pour des problèmes de régression. Toutefois, pour obtenir un meilleur résultat en classification avec un grand nombre de variables, il est suggéré de sélectionner un m plus grand.

Les forêts aléatoires à entrées aléatoires sont couramment appelées « *random forests with random inputs* », abrégé en *RF-RI*. Les implémentations de forêts aléatoires sur des logiciels sont très souvent des *RF-RI*, étant donné la simplicité et la popularité de la méthode.

3.1.3 Autres méthodes de perturbation

Beaucoup d'auteurs ont développé des méthodes de construction de forêts aléatoires différentes de celles évoquées à l'instant, avec des stratégies de perturbation des arbres assez variées mais qui reprennent souvent en partie des éléments déjà vus.

Nous pouvons citer la méthode des sous-espaces aléatoires – « *random subspaces* »– développée par T.K Ho [7], qui ressemble à celle des entrées aléatoires, à deux exceptions. D'abord, cette méthode travaille sur l'ensemble d'apprentissage complet plutôt que sur des échantillons *bootstrap*. Ensuite, le tirage aléatoire des variables explicatives est réalisé avant la construction des arbres, et il reste fixe pour l'ensemble de l'arbre.

Une autre stratégie, proposée par Freund et Schapire [8], consiste à appliquer la règle de base sur un échantillon *bootstrap* puis, récursivement, appliquer à chaque étape cette même règle sur l'échantillon pour lequel l'erreur du prédicteur obtenu précédemment est la plus importante. Les tirages effectués ne sont donc plus indépendants, contrairement au cas du *bagging*.

Une variante des forêts à entrées aléatoires est celle des arbres dits « extrêmement aléatoires » (*ERT* pour « *Extremely Randomized Trees* ») [9]. Leur principe est de construire des arbres pour lesquels les découpes des nœuds sont choisies parmi un ensemble de découpes entièrement aléatoire. C'est-à-dire que non seulement les variables d'entrée sont tirées aléatoirement, mais les points de découpe le sont également. La découpe choisie est celle qui minimise l'hétérogénéité de l'ensemble des données d'entrée, au regard d'une fonction de coût désignée.

Une dernière méthode, beaucoup plus simple, consiste à bruiter les sorties de l'ensemble d'apprentissage : une variable de bruit aléatoire est ajoutée à chaque observation de la variable cible [10].

Les points communs de toutes ces procédures sont l'exploitation de l'instabilité des arbres générés par la règle de base et l'obtention d'une collection de prédicteurs aux sorties les plus variées possibles. **C'est pourquoi les forêts utilisent très souvent des arbres non élagués**. Dans nos applications, nous utiliserons exclusivement les *RF-RI* pour une raison de disponibilité des bibliothèques sur `R`. Toutefois, les autres méthodes sont intéressantes du fait de l'innovation qu'elles apportent sur la construction des arbres comme des forêts. Certaines comme les *ERT* peuvent même donner des résultats meilleurs

que les *RF-RI* [9] : elles fournissent en ce sens des pistes d'amélioration des algorithmes connus.

3.2 Agrégation des sorties

À cette étape de la construction des forêts aléatoires, nous disposons, quelle que soit la méthode employée, d'une collection de prédicteurs $\left(\hat{h}_j\right)_{1\leqslant j\leqslant q}$, c'est-à-dire d'un ensemble de q arbres fournissant des prédictions variées. Le reste du travail consiste maintenant à unifier ces prévisions : c'est l'agrégation des sorties. La méthode varie selon le contexte du problème.

En classification

L'agrégation est effectuée par vote majoritaire des prédictions des arbres entre les différentes classes possibles. Le prédicteur de la forêt aléatoire pour une entrée X_i s'écrit alors :

$$\hat{h}_{RF}\left(X_i\right) = \underset{\ell\in\{1,\dots,L\}}{\arg\max} \sum_{j=1}^{q} \mathbb{1}_{\hat{h}_j(X_i)=\ell}.$$

En régression

Les prédictions des arbres sont moyennées pour former la sortie finale. Le prédicteur de la forêt s'écrit :

$$\hat{h}_{RF}\left(X_i\right) = \frac{1}{q}\sum_{j=1}^{q} \hat{h}_j\left(X_i\right).$$

3.3 Erreur *OOB*

Nous pouvons maintenant évoquer un nouvel avantage majeur du *bagging*, que nous avions précédemment mis de côté. La méthode nous fournit un certain nombre d'échantillons de $\mathcal{L}_n$, que nous avions noté $\left(\mathcal{L}_n^{\Theta_j}\right)_{1\leqslant j\leqslant q}$.

Ceci nous permet d'évaluer, pour chaque observation X_i, l'erreur de prédiction commise par l'ensemble des arbres pour lesquels X_i n'a pas été retenu à leur construction. Nous disons alors que X_i est « hors du sac » pour ces arbres (« *Out Of Bag* » en anglais).

L'erreur *OOB* désigne l'erreur moyenne commise sur toutes les prédictions $\hat{Y}_i$ des Y_i par les arbres pour lesquels X_i est hors du sac. Cette erreur s'écrit :

$$err_{OOB} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\hat{Y}_i\neq Y_i} & \text{en classification,} \\ \frac{1}{n}\sum_{i=1}^{n} \left(\hat{Y}_i - Y_i\right)^2 & \text{en régression.} \end{cases}$$

C'est une estimation partielle de l'erreur de généralisation de la forêt : en effet, elle n'est évaluée que sur des parties de celle-ci, sur un nombre d'arbres fluctuant selon les observations. L'erreur *OOB* ne représente donc pas exactement l'erreur de généralisation de toute la forêt aléatoire. Néanmoins, elle fournit une excellente mesure particulièrement utile pour définir les paramètres q et m de la forêt, respectivement le nombre d'arbres et le nombre de variables explicatives retenues pour la méthode des entrées aléatoires.

4 Mise en œuvre

Nous allons mettre en application les techniques d'arbres de décision et de forêts aléatoires que nous venons de présenter. Nous nous intéresserons à deux problèmes. Le premier est un problème de classification bien connu dans le domaine de l'apprentissage automatique : il s'agit de prédire le sort d'un certain nombre de passagers du Titanic, sur la base d'informations issues des registres de la compagnie ayant affrété le paquebot. Le second relève de l'estimation de la valeur marchande de voitures d'occasion à partir de données de transactions recueillies sur le marché automobile britannique. C'est une occasion de mettre à l'épreuve les forêts aléatoires sur un problème de régression.

Pour chacun des problèmes, nous chercherons à comparer l'efficacité des méthodes d'arbres et de forêts à celle des réseaux de neurones. Nous procéderons comme suit : nous illustrerons et évaluerons la performance de plusieurs arbres de décision ; ensuite nous construirons des forêts dont nous tenterons d'optimiser les paramètres suivant un protocole ; enfin nous comparerons la qualité des prévisions avec celles fournies par les réseaux de neurones.

Les programmes servant à la présente application sont entièrement développés sur `R`. Pour la construction des arbres de décision et des forêts aléatoires, nous utilisons respectivement les bibliothèques `rpart` et `randomForest`. Cette dernière est une implémentation sur `R` de l'algorithme originel de Breiman et Cutler [11]. Les réseaux de neurones sont développés à l'aide de la bibliothèque `Tensorflow for R` et en particulier de son module `Keras`[6]. L'évaluation des prévisions est réalisée en utilisant des fonctions de la bibliothèque `caret`.

Tous les programmes et jeux de données utilisés dans ce rapport sont fournis en annexe. Ils sont proposés en téléchargement libre sur un dépôt GitHub.

4.1 Application à un problème de classification : le sort des passagers du Titanic

Le jeu de données présente, après nettoyage, 891 observations pour onze variables explicatives. Nous cherchons à prédire, pour chaque passager, s'ils ont survécu au drame ou non : la variable cible `Survived` est donc en deux classes.

Nous disposons des informations suivantes :

Nom	Description de la variable
`Pclass`	La classe du billet avec lequel le passager a voyagé
`Sex`	Le sexe du passager (masculin ou féminin)
`Age`	L'âge du passager
`SibSp`	Le nombre de frères et sœurs, et d'époux à bord
`Parch`	Le nombre de parents et d'enfants du passager à bord
`Fare`	Le prix total du billet
`Embarked`	Le port d'embarquement
`Deck`	Le pont où se trouvait la cabine du passager

6. Les programmes ont été exécutés avec un GPU NVIDIA GeForce GTX 1660 Ti.

`SizeGp`	Le taille du groupe qui a voyagé sur le même billet
`FarePerPers`	Le tarif du billet par personne
`Notable`	Vrai si le passager est une personne avec un statut particulier

TABLE 1 – Liste des variables explicatives du jeu de données Titanic

La variable `Embarked` a trois modalités possibles : Cherbourg, Southampton et Queenstown. Les ponts vont de A à G, avec une exception pour une cabine sur un pont T.

Les données sont séparées en deux échantillons : celui destiné à l'apprentissage du modèle contient 712 observations, l'autre contient 173 observations et sera destiné à l'évaluation des résultats de prédiction.

4.1.1 Prévisions par arbres de décision

Commençons par modéliser nos données à l'aide d'arbres de décision. Dans toute cette partie, nous utilisons des arbres *CART*. Dans les sections 2.2.1 et 2.2.2, nous avons expliqué la procédure de construction des formes maximale et élaguées de ces arbres.

Arbre maximal

Voici l'arbre maximal obtenu pour nos données : pour le construire, nous avons fixé comme unique condition qu'un nœud ne soit découpé que s'il contient au moins deux observations.

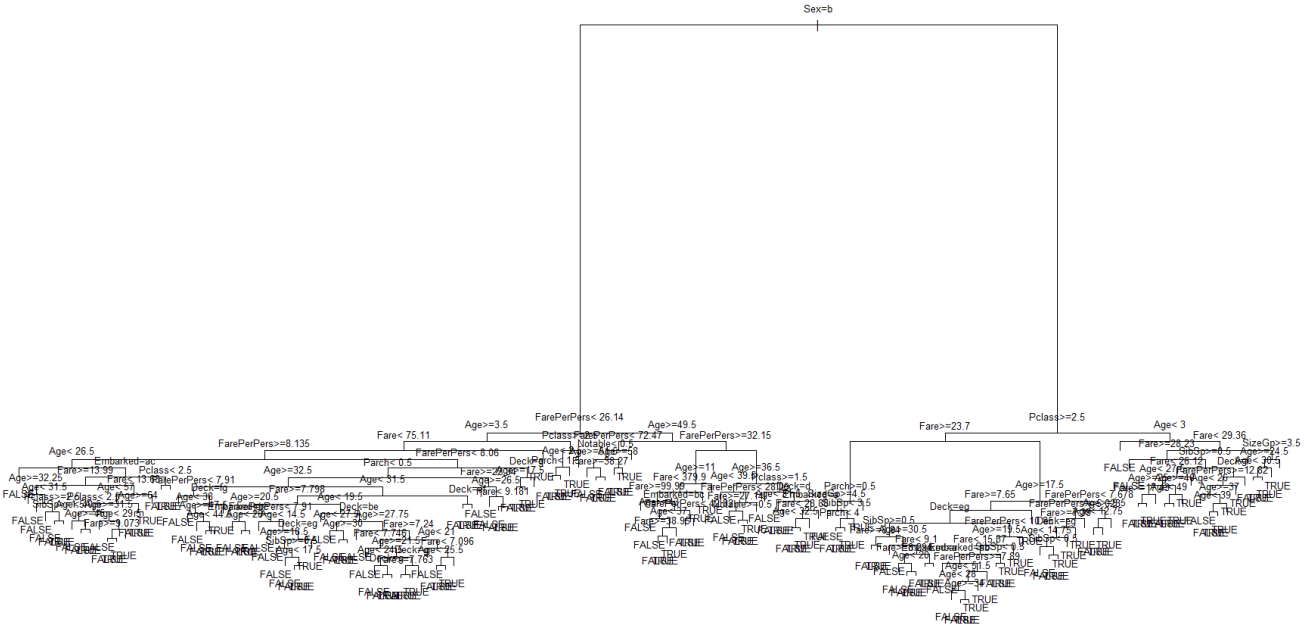

FIGURE 2 – Arbre maximal (Titanic)

Nous pouvons constater que cet arbre est très complexe comme prévu, et surtout peu lisible. En voici une version sans texte, pour pouvoir observer la forme de l'arbre :

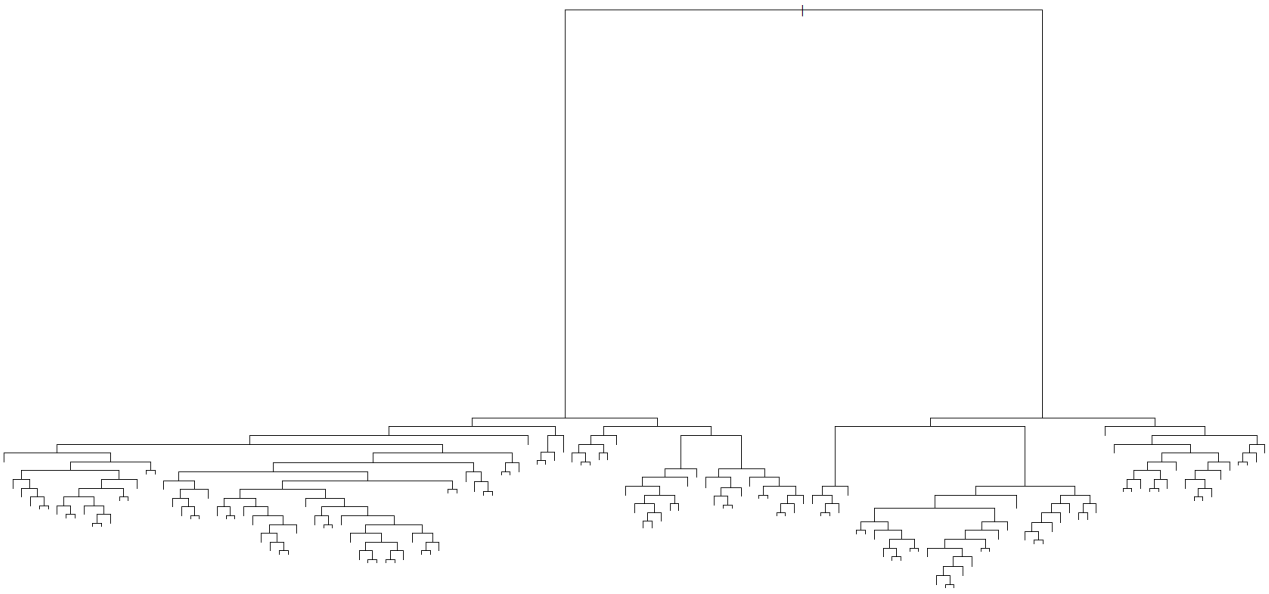

FIGURE 3 – Arbre maximal sans légende (Titanic)

Cet arbre ne compte pas moins de 143 feuilles. Et sa très grande spécificité pose effectivement un problème quant à la qualité des prédictions. Voici les résultats obtenus sur l'échantillon de validation :

Prédiction \ Réalité	Survécu	Décédé
Survécu	46	25
Décédé	22	80

TABLE 2 – Matrice de confusion de l'arbre maximal (Titanic)

La précision (*accuracy*) du modèle, qui est calculée comme la proportion de bons classements, est de 72,83% (IC 95% : $[65{,}56; 79{,}31]$ %). C'est un résultat assez médiocre, qui témoigne des défaillances de l'arbre maximal. Il y a cependant un moyen d'améliorer cela, grâce à l'arbre optimal.

Arbre optimal

Nous pouvons, à l'aide d'un graphe généré par la commande `plotcp`, rechercher l'étape de la construction de l'arbre maximal pour laquelle l'erreur de généralisation était minimale :

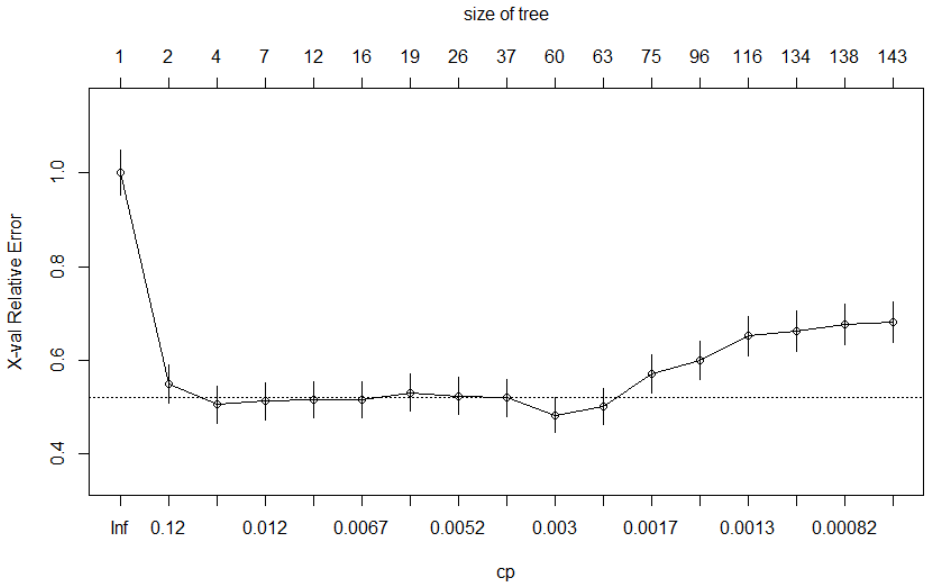

FIGURE 4 – Évolution de l'erreur de généralisation en fonction de la taille de l'arbre (Titanic)

Sur ce graphe, chaque point représente un arbre élagué. Nous retrouvons le paramètre de complexité `cp` que nous avions évoqué à la fin de la section 2.2.2. Ce paramètre nous intéresse tout particulièrement car c'est celui qui nous permet de définir une condition d'arrêt à l'algorithme d'élagage. Nous fixons donc `cp` à la valeur qui minimise l'erreur de validation croisée [7] : ici `cp` $\approx 0{,}0024$.

Nous pouvons maintenant élaguer l'arbre maximal pour obtenir l'arbre optimal :

7. `rpart` intègre une méthode de validation croisée : c'est une estimation de l'erreur de généralisation de l'arbre sur des échantillons d'observations uniformément tirés pour l'élagage.

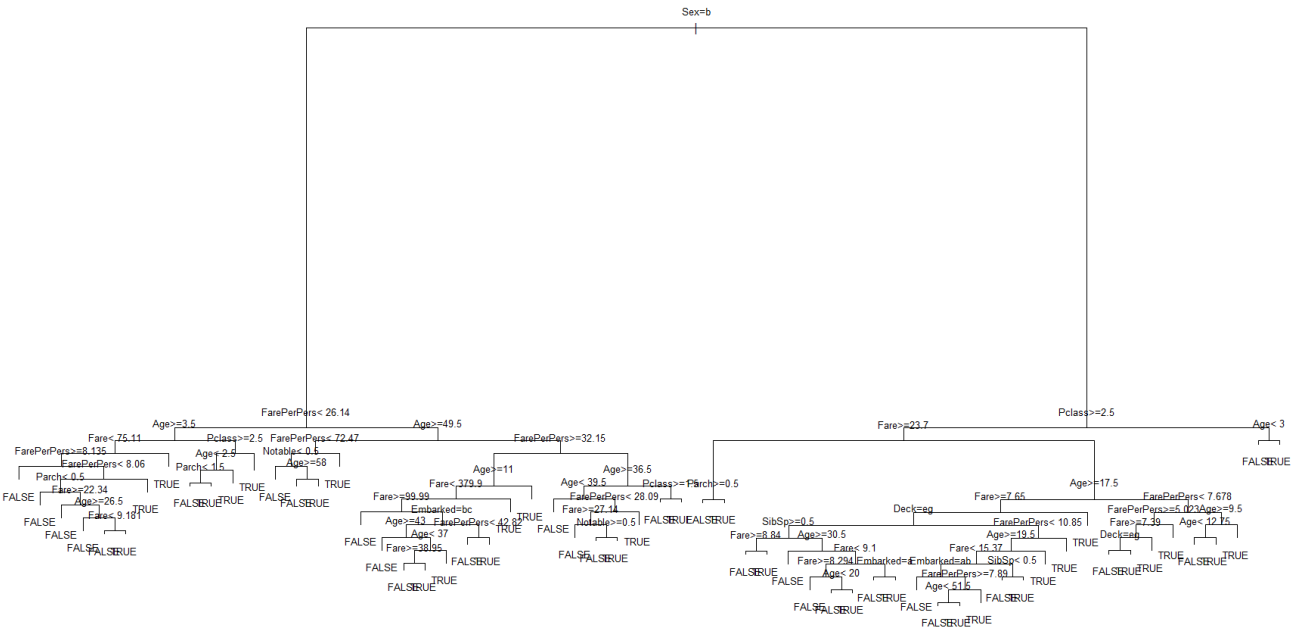

FIGURE 5 – Arbre optimal (Titanic)

L'arbre optimal est moins complexe que son prédécesseur maximal (60 feuilles) ; il reste cependant encore trop complexe pour permettre une interprétation simple de la classification opérée. Examinons néanmoins la qualité des prédictions fournies :

Prédiction \ Réalité	Survécu	Décédé
Survécu	49	13
Décédé	19	92

TABLE 3 – Matrice de confusion de l'arbre optimal (Titanic)

Nous obtenons un score de précision de 81,5% (IC 95% : [74,9; 86,99] %), ce qui représente une amélioration substantielle par rapport à l'arbre maximal. Il s'agit en principe du meilleur résultat d'estimation que nous pouvons obtenir sur un seul arbre. Toutefois, il est possible d'obtenir une qualité d'ajustement proche de celle-ci avec un arbre beaucoup plus simple à l'aide de la règle dite du « 1 écart-type », énoncée dans Breiman *et al.*[2].

Arbre simplifié à 1 écart-type

Revenons à la figure 4 et notons que la ligne pointillée est située à un écart-type au-dessus du point qui représente l'arbre optimal. Cette ligne nous permet de sélectionner l'arbre le plus simple (avec le moins de feuilles) situé à moins d'un écart-type de l'erreur de validation croisée de l'arbre optimal. Voici l'arbre simplifié obtenu :

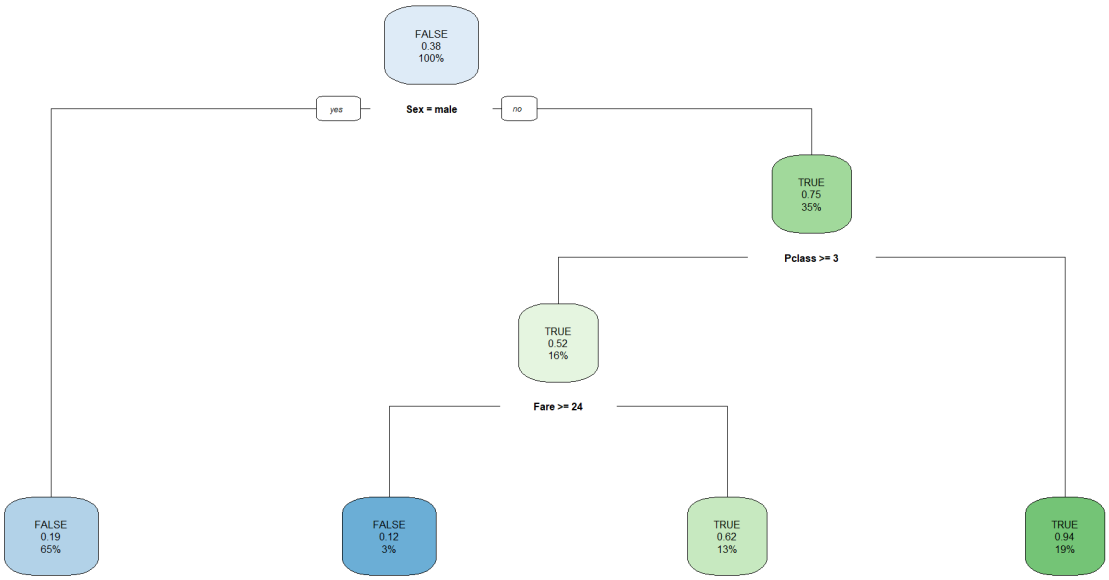

FIGURE 6 – Arbre simplifié à 1 écart-type (Titanic)

Nous l'avons représenté à l'aide de l'extension `rpart.plot` qui permet d'afficher des arbres de décision de manière claire et stylisée. L'interprétation de la classification produite est immédiate : les hommes, et les femmes en $3^{ème}$ classe dont le billet vaut plus de 24 £ sont classés comme décédés ; les autres sont annoncés comme survivants. Examinons l'erreur de prédiction sur l'échantillon de validation :

Prédiction \ Réalité	Survécu	Décédé
Survécu	45	15
Décédé	23	90

TABLE 4 – Matrice de confusion de l'arbre à 1 écart-type (Titanic)

Le score de précision est de 78,03% (IC 95% : $[71{,}12; 83{,}93]$ %), ce qui est un résultat surprenant pour un modèle aussi simple voire simpliste que celui-ci. Ce constat est toutefois à nuancer : il est rare en pratique de pouvoir obtenir une telle simplification. Généralement, les arbres à 1 écart-type sont moins simplifiés que cela, par rapport à l'arbre optimal.

4.1.2 Prévisions par forêts aléatoires

De manière générale, les arbres individuels, même sous leur forme optimale, sont instables et sensibles à l'ensemble d'apprentissage. Nous allons donc exploiter les propriétés des forêts aléatoires afin d'obtenir un modèle plus robuste.

L'enjeu principal, lorsque nous générons des forêts aléatoires, est de trouver la bonne combinaison de paramètres qui minimise l'erreur de validation. Nous rappelons ces paramètres : le nombre d'arbres générés et le nombre de variables explicatives retenues pour la méthode des entrées aléatoires, appelés `ntree` et `mtry` dans la bibliothèque.

Première forêt

Pour commencer, nous générons une forêt de 5000 arbres et conservons la valeur par défaut de `mtry`. Voici les résultats de prédiction obtenus par cette première forêt :

Prédiction \ Réalité	Survécu	Décédé
Survécu	47	12
Décédé	21	93

TABLE 5 – Matrice de confusion d'une forêt aléatoire par défaut (Titanic)

Nous obtenons un score de précision de 80,92% (IC 95% : $[74{,}27; 86{,}49]$ %). Voyons si nous pouvons améliorer cela.

Recherche des paramètres optimaux

Pour cette partie, nous évaluons les forêts au travers de l'erreur *OOB* comme estimateur de l'erreur de généralisation. La bibliothèque `randomForest` nous permet d'observer la distribution des fréquences de mise « hors du sac » des observations :

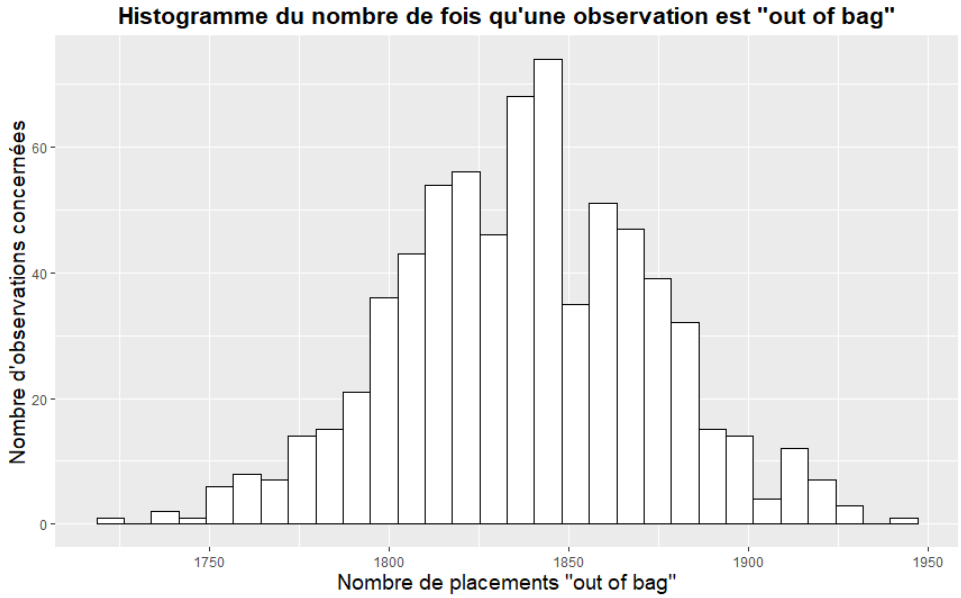

FIGURE 7 – Histogramme de la mise « hors du sac » des observations (Titanic)

Ceci nous permet d'apercevoir la taille des échantillons sur la base desquels l'erreur *OOB* est évaluée : ici, chaque observation se retrouve placée hors du sac environ 1800 à 1900 fois. Nous pouvons également représenter l'erreur *OOB* en fonction du nombre d'arbres dans la forêt :

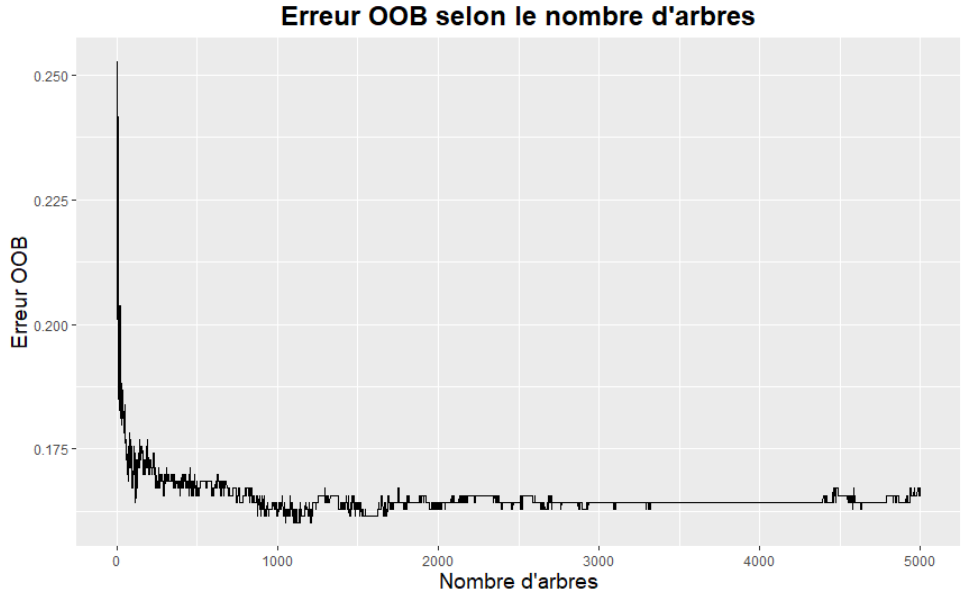

FIGURE 8 – Évolution de l'erreur *OOB* en fonction du nombre d'arbres (Titanic)

Nous constatons que l'erreur *OOB* décroît fortement sur les premiers arbres avant de se stabiliser à un certain niveau. Retenons le nombre d'arbre à partir duquel cette stabilisation a lieu et fixons `ntree` = 2000.

Concernant le paramètre `mtry`, étant donné que nous disposons de 11 variables explicatives, il nous est possible d'évaluer l'erreur *OOB* moyenne sur toutes les valeurs possibles. Nous obtenons les résultats suivants pour dix répétitions par valeur de `mtry` :

`mtry`	1	2	3	4	5	6	7	8	9	10	11
Erreur *OOB*	0,206	0,182	0,166	0,173	0,175	0,178	0,181	0,185	0,183	0,186	0,19

TABLE 6 – Erreurs *OOB* moyennes pour chaque valeur de `mtry`, 10 répétitions

Nous fixons donc `mtry` = 3, en remarquant la correspondance avec la valeur théorique conseillée $\sqrt{p}$, après arrondi. Examinons les résultats de la forêt nouvellement construite :

Prédiction \ Réalité	Survécu	Décédé
Survécu	47	12
Décédé	21	93

TABLE 7 – Matrice de confusion d'une forêt aléatoire optimisée (Titanic)

Nous obtenons les mêmes résultats que la première forêt, avec le même score de précision et le même intervalle de confiance. Nous en déduisons que la forêt optimisée produit la même qualité de prédiction avec 60% d'arbres en moins, ce qui la rend plus rapide à générer.

Importance des variables

Exploitons maintenant un avantage majeur des arbres et des forêts aléatoires : leur capacité à quantifier l'importance des variables pour le classement des observations. Dans

la section 2.3, nous avons défini une première mesure basée sur la contribution de chaque variable à la réduction de l'hétérogénéité des données classées. Observons ce que cette mesure suggère :

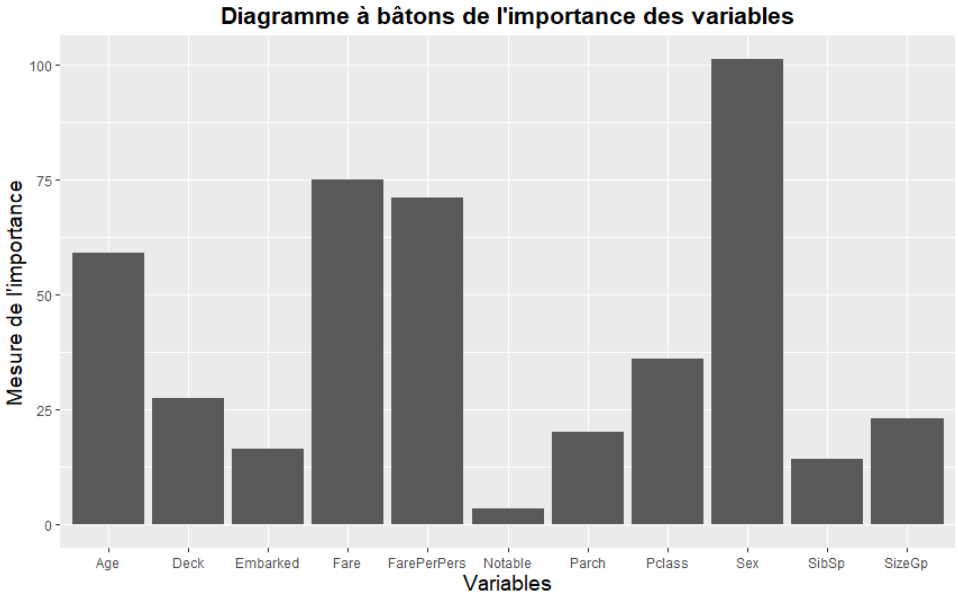

FIGURE 9 – Importance des variables selon l'arbre *CART* optimal (Titanic)

L'arbre optimal suggère que la variable la plus importante est de loin le sexe des passagers. Viennent ensuite le tarif du billet et l'âge, puis la classe de voyage. Les autres variables contribuent dans une moindre mesure à la classification. Ceci est cohérent avec les interprétations que nous avons formulées.

La forêt aléatoire *RF-RI* définit l'importance des variables d'une manière différente : elle est appréciée en effectuant des permutations sur les valeurs des variables et en examinant l'erreur *OOB* obtenue, de manière à mesurer la sensibilité de l'erreur à chaque variable. Examinons l'importance des variables au sens de la forêt :

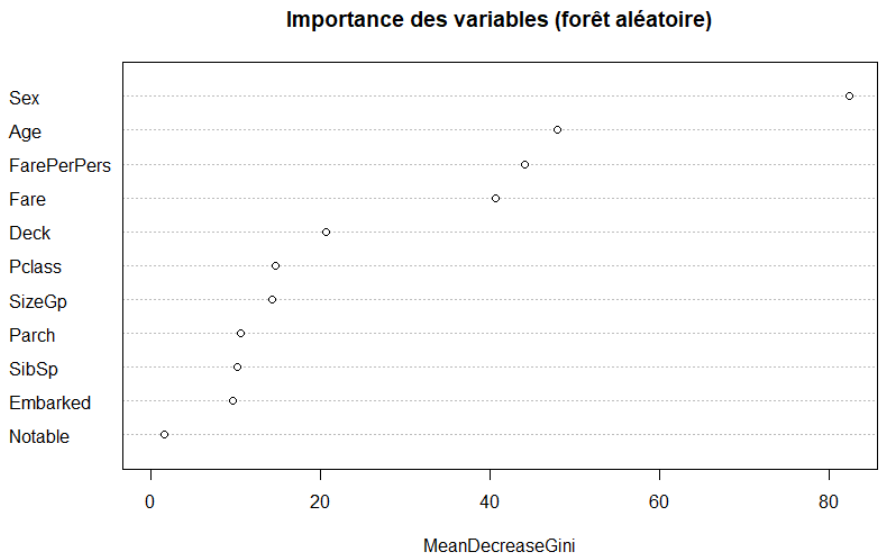

FIGURE 10 – Importance des variables selon la forêt aléatoire optimale (Titanic)

La forêt aléatoire fait à peu près la même description, à l'exception qu'elle accorde plus d'importance au pont de la cabine dans laquelle le passager a voyagé.

4.1.3 Comparaison avec les résultats obtenus par réseau de neurones (classifieur *MLP*)

Nous proposons de comparer les résultats de prédiction que nous avons collectés par les arbres *CART* et les forêts aléatoires avec ceux que peuvent produire des réseaux de neurones.

Nous faisons le choix d'un modèle de classification à perceptrons multi-couches (*MLP Classifier*) constitué comme suit : une couche cachée de 15 neurones ainsi qu'une couche de sortie de 2 neurones. Le modèle est entraîné sur l'ensemble d'apprentissage entier sur dix périodes. `Keras` sur `R` nous permet d'effectuer le suivi en temps réel de la précision obtenue (en ordonnée) sur les deux échantillons – apprentissage et validation – à chaque itération (en abscisse) :

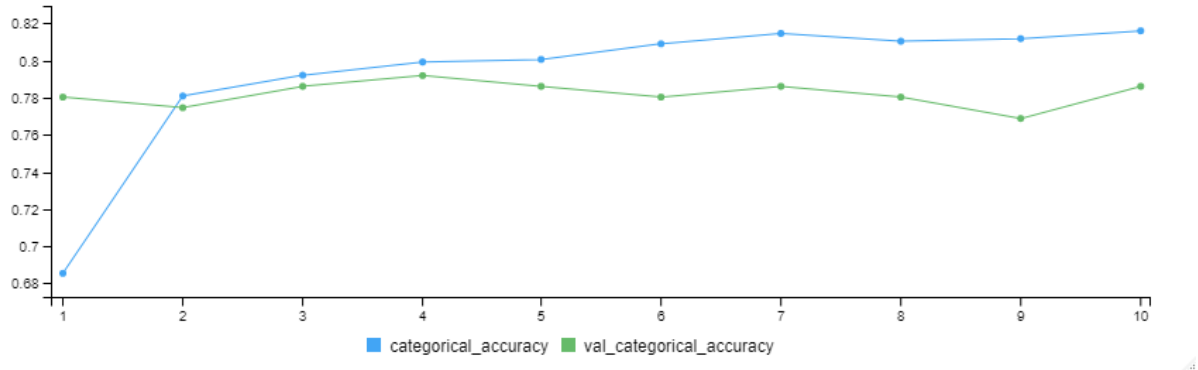

FIGURE 11 – Évaluation du modèle par réseau de neurones en temps réel (Titanic)

Voici les prédictions du modèle :

Prédiction \ Réalité	Survécu	Décédé
Survécu	45	14
Décédé	23	91

TABLE 8 – Matrice de confusion d'un réseau de neurones (Titanic)

Le réseau de neurones obtient un score de précision de 78,61% (IC 95% : [71,75; 84,47] %). Ce résultat est légèrement moins bon que celui obtenu avec nos méthodes d'arbres et de forêts aléatoires, pour une complexité et un coût en temps de calcul très significativement supérieurs (de l'ordre de 2 secondes par période).

4.1.4 Bilan de l'expérience

Sur ce jeu de données, nous avons pu examiner et comparer la qualité de prédiction fournie par plusieurs formes d'arbres *CART*, des forêts aléatoires et un réseau de neurones. Nous pouvons ainsi trancher parmi ces méthodes pour le problème de classification que nous venons de traiter.

L'arbre *CART* optimal est celui qui nous a fourni les meilleures prédictions. Toutefois, il convient de garder à l'esprit qu'il ne s'agit que d'un seul arbre, et qu'il n'a été évalué que sur un seul échantillon qui n'illustre pas forcément ses faiblesses. L'arbre simplifié à 1 écart-type est également intéressant à considérer, surtout pour des problèmes complexes. La forêt optimale, elle, a proposé des prédictions d'une qualité très proche, avec des garanties plus solides sur la généralisation à d'autres échantillons de données. Le réseau de neurones s'est rapproché de ces performances, mais il est fortement handicapé par la lourdeur de la méthode. Ainsi, pour des problèmes de classification analogues à celui-ci, nous concluons en faveur des forêts et des arbres de décision.

4.2 Application à un problème de régression : estimation de la valeur d'une voiture d'occasion

Traitons maintenant un problème d'une autre nature. Nous disposons d'un ensemble de données bien plus conséquent que le précédent : 117 893 observations des caractéristiques de voitures d'occasion vendues sur le marché britannique. Voici les informations que nous possédons :

Nom	Description de la variable
`brand`	La marque du véhicule (9 modalités)
`model`	Le modèle du véhicule (195 modalités)
`year`	L'année de première mise en circulation
`transmission`	Le mode de transmission de la boîte de vitesses (4 modalités)
`mileage`	Le kilométrage du véhicule lors de la vente
`fuelType`	Le type de carburant consommé (5 modalités)
`tax`	Le montant de la taxe annuelle appliquée sur le véhicule
`mpg`	La consommation du véhicule (en milles par gallon)
`engineSize`	La cylindrée du véhicule

TABLE 9 – Liste des variables explicatives du jeu de données Cars

Notre objectif est de prédire la valeur marchande (variable `price`) d'un véhicule à partir de ces informations. Comme pour le problème précédent, nous séparons notre jeu de données en deux échantillons : l'ensemble d'apprentissage contient 94 314 observations et l'ensemble de validation 20 427 observations.

4.2.1 Prévisions par arbres de décision

Nous reprenons la même procédure que pour le problème de classification.

Arbre maximal

Si, pour le problème précédent, l'arbre maximal était déjà complexe, celui-ci l'est encore beaucoup plus. Voici une représentation simplifiée de l'arbre maximal obtenu :

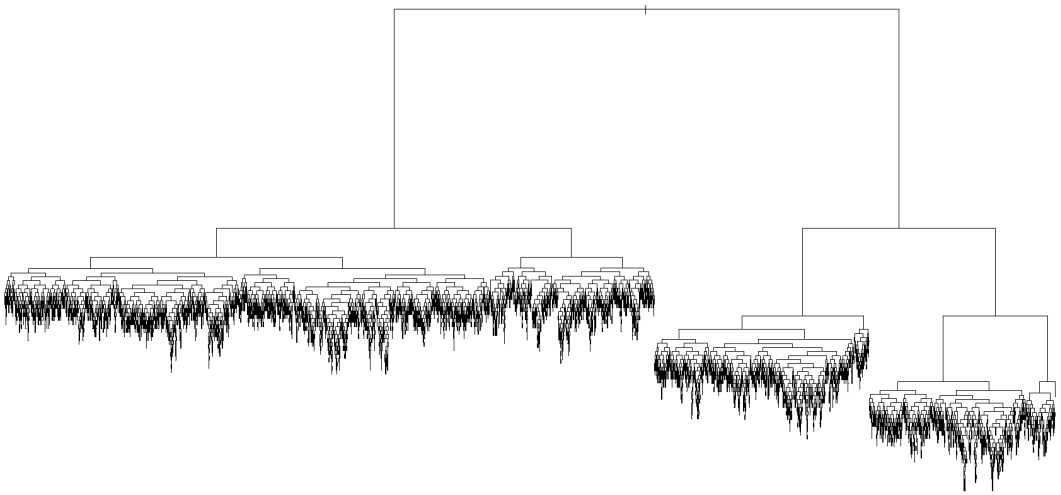

FIGURE 12 – Arbre maximal sans légende (Cars)

Les feuilles sont indiscernables à l'œil : en effet, elles sont au nombre de 7 552 dans cet exemple. Concernant les prédictions, comme nous traitons d'un cas de régression, nous apprécions leur qualité au regard de la racine de leur erreur quadratique moyenne

(*Root Mean Squared Error*, *RMSE*) et du score R^2. Ce dernier est une mesure de l'erreur d'ajustement, comprise entre 0 et 1 et qui s'exprime comme :

$$R^2 = 1 - \frac{\text{Somme des carrés des résidus}}{\text{Somme des carrés totaux}} = 1 - \frac{\sum_{i=1}^{n} \left(\hat{Y}_i - Y_i\right)^2}{\sum_{i=1}^{n} \left(Y_i - \overline{Y}\right)^2}.$$

Le score R^2 peut être interprété comme la part de la variance des données expliquée par le modèle. Examinons l'erreur de prédiction commise par l'arbre maximal :

RMSE	**Score R^2**
2202,63	0,9518

TABLE 10 – Scores obtenus par l'arbre maximal (Cars)

Malgré sa complexité, l'arbre maximal fournit ici une qualité de prédiction très satisfaisante. Voyons si ce résultat peut être amélioré.

Arbre optimal

Nous recherchons l'arbre optimal pour nos données : pour cela, nous commençons par visualiser le graphe de la commande `plotcp`, afin d'examiner l'évolution de l'erreur de validation croisée :

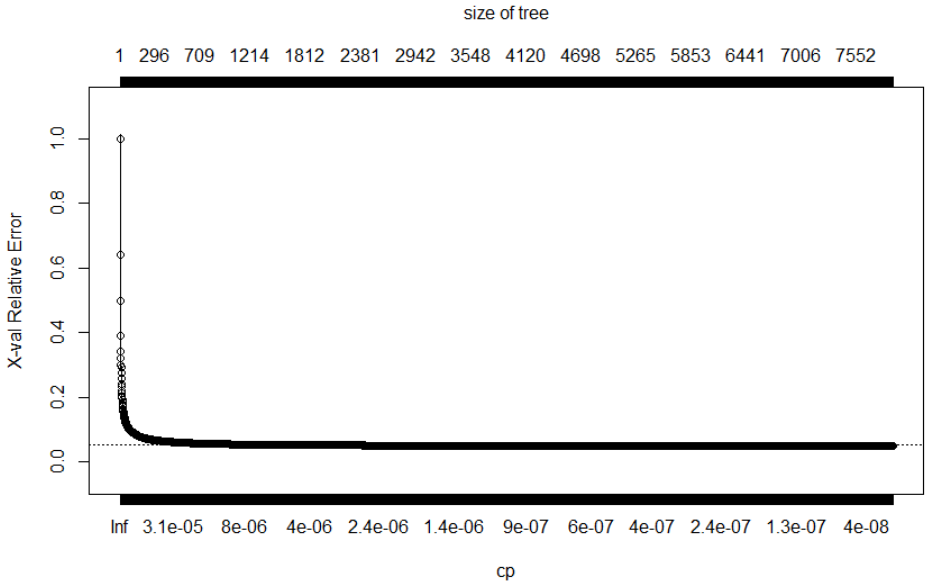

FIGURE 13 – Évolution de l'erreur de généralisation en fonction de la taille de l'arbre (Cars)

Nous avons un grand nombre d'arbres candidats. Ici, l'arbre optimal est obtenu pour `cp` $\approx 3.59 \times 10^{-7}$, ce qui correspond à un arbre à 5 606 feuilles. [8] En voici la représentation :

8. Il n'y a pas de relation générale qui lie `cp` au nombre de feuilles, sans faire appel à l'expression de la perte de qualité d'ajustement décrite dans l'algorithme d'élagage (section 2.2.2). Le nombre de feuilles indiqué ici est spécifique à notre exemple.

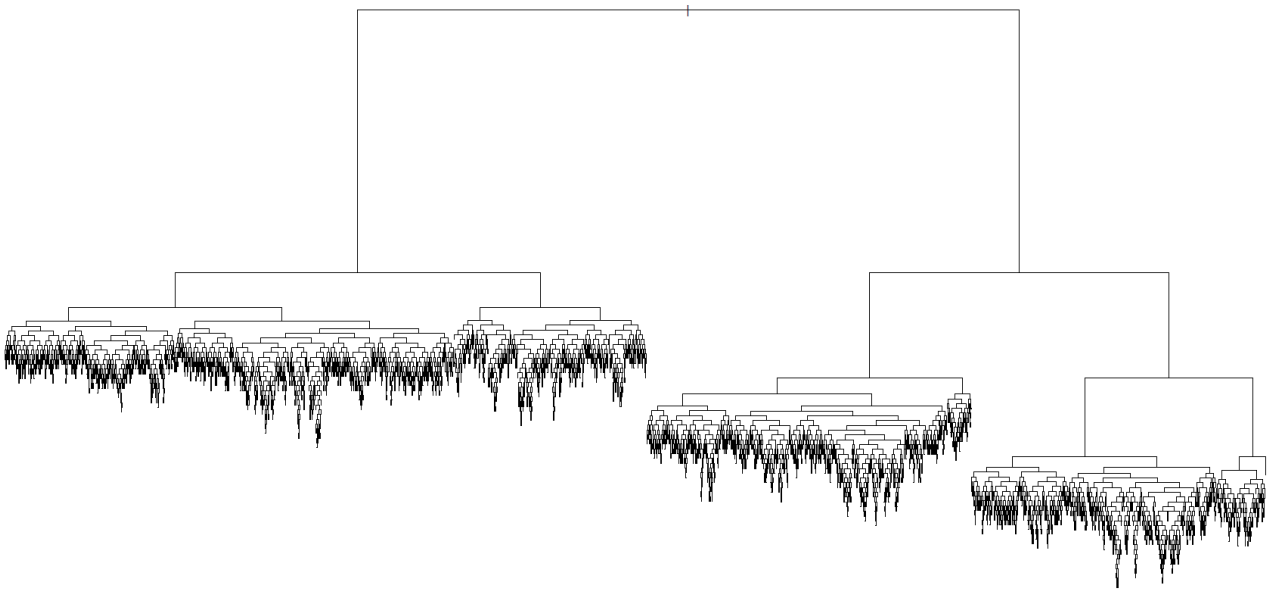

FIGURE 14 – Arbre optimal (Cars)

Nous pouvons maintenant générer des prédictions à l'aide de cet arbre et comparer leurs erreurs à celles de l'arbre maximal :

RMSE	**Score R^2**
2200,38	0,9519

TABLE 11 – Scores obtenus par l'arbre optimal (Cars)

Le score R^2 obtenu est sensiblement le même que celui de l'arbre maximal. Nous pouvons relever une très légère amélioration du *RMSE*. Ainsi, l'arbre optimal a généré des prévisions du même ordre de qualité avec près d'un quart de feuilles en moins, ce qui se ressent sur la durée d'exécution de la commande de prédiction. Et il est possible d'accélérer encore le programme en effectuant une nouvelle simplification.

Arbre simplifié à 1 écart-type

En reprenant le même procédé que celui mis en œuvre pour les données du Titanic, nous construisons l'arbre simplifié à 1 écart-type de l'erreur de validation croisée minimale :

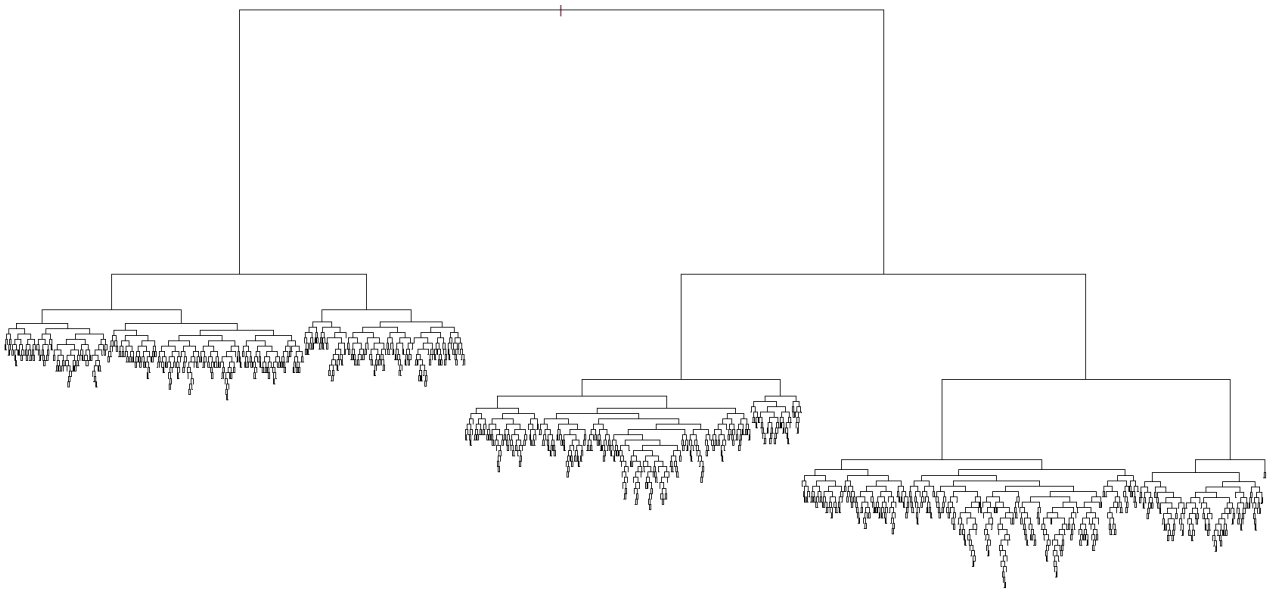

FIGURE 15 – Arbre simplifié à 1 écart-type (Cars)

Cet arbre compte 1 589 feuilles : environ 80% des feuilles ont donc été retirées après élagage. Cela représente une simplification drastique, comparé à l'arbre optimal, bien que moins drastique que la simplification obtenue pour le problème du Titanic (97% des feuilles retirées). Voici les résultats de prédiction pour cet arbre :

RMSE	**Score R^2**
2244,17	0,9499

TABLE 12 – Scores obtenus par l'arbre à 1 écart-type (Cars)

La détérioration du score R^2 est très faible et le *RMSE* n'a que légèrement augmenté. Donc l'arbre à 1 écart-type, malgré sa réduction significative, fournit toujours une qualité de prédiction très satisfaisante, et ce avec un temps d'exécution minimal. Il semble donc être un excellent compromis parmi les trois arbres que nous venons de détailler.

4.2.2 Prévisions par forêts aléatoires

À présent, intéressons-nous aux forêts aléatoires pour comparer les résultats que nous pouvons obtenir avec ceux donnés par les arbres *CART*.

Première forêt

Étant donné le grand nombre d'observations dans l'échantillon d'apprentissage, nous restreignons le nombre d'arbres dans la forêt à `ntree` = 50, ce afin de limiter le temps de calcul. Également, une limitation technique de la bibliothèque nous impose d'abandonner la variable `model` : en effet, celle-ci est déclinée en 195 modalités alors que l'algorithme n'en supporte que 57 au maximum. Nous devons donc nous reposer sur la variable `brand` pour regrouper l'information contenue dans les modèles des véhicules.

Cette première forêt nous donne les résultats de prédiction suivants :

RMSE	Score R^2
2499,01	0,9391

TABLE 13 – Scores obtenus par une forêt aléatoire par défaut (Cars)

La qualité d'ajustement est satisfaisante pour une première, mais la construction de la forêt dure près de cinq minutes. Nous allons tenter d'optimiser les paramètres de la forêt pour à la fois simplifier celle-ci tout en améliorant si possible son erreur de prédiction.

Recherche des paramètres optimaux

Comme dans le cas précédent, nous nous appuyons sur l'erreur *OOB* pour rechercher la forêt optimale. Le nombre conséquent d'observations dans l'ensemble d'apprentissage n'est pas de nature à inquiéter vis-à-vis de la taille des échantillons « hors du sac ».

En contexte de régression, l'erreur *OOB* est mesurée par l'erreur quadratique moyenne de prédiction (*Mean Squared Error*, *MSE*). Visualisons l'évolution de celle-ci :

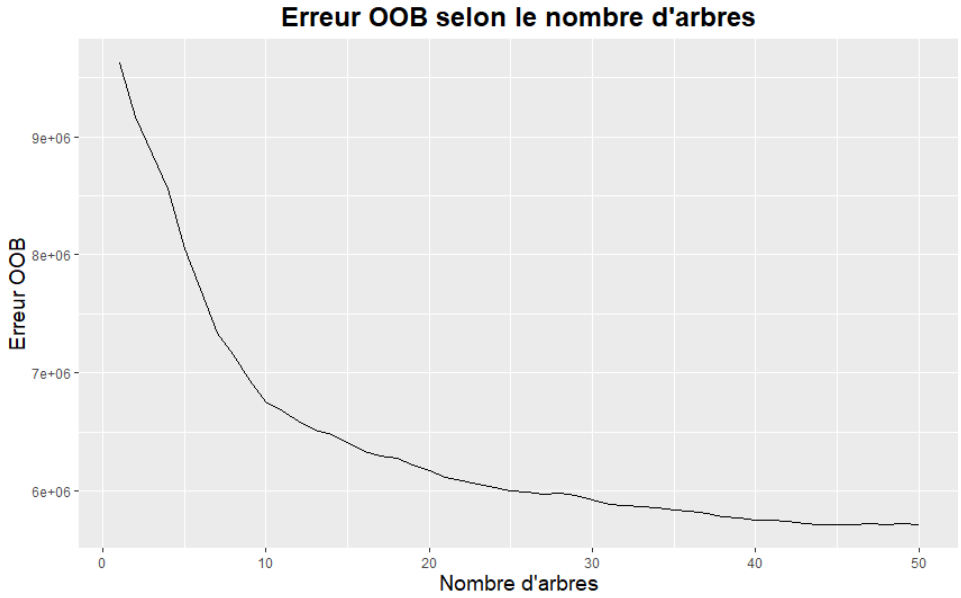

FIGURE 16 – Évolution de l'erreur *OOB* en fonction du nombre d'arbres (Cars)

Nous remarquons que l'erreur ne décroît plus beaucoup passé 30 arbres. Nous pouvons confirmer cela en examinant l'évolution du score R^2 sur les échantillons *OOB* :

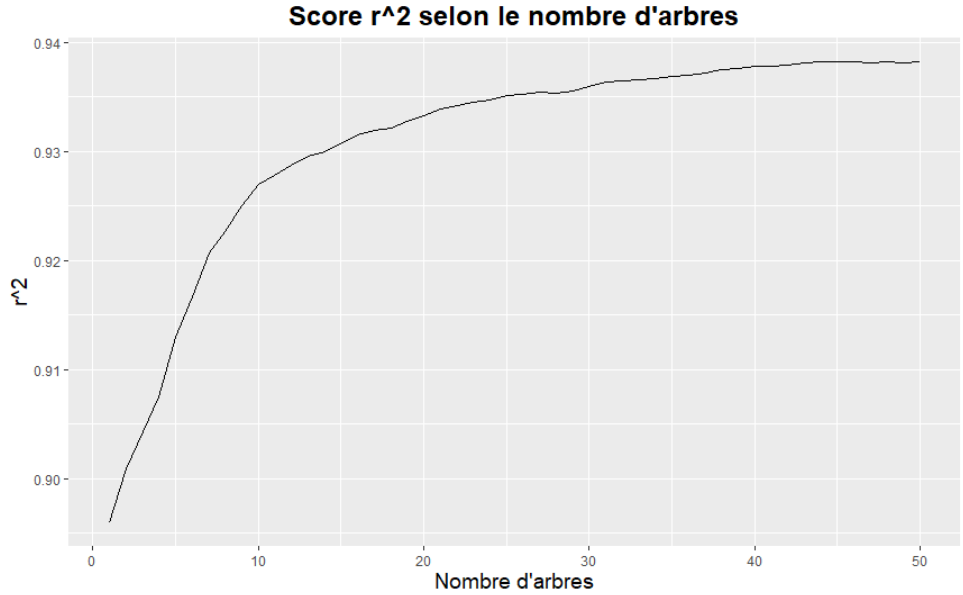

FIGURE 17 – Évolution du score R^2 en fonction du nombre d'arbres (Cars)

En effet, passé 30 arbres, le score R^2 se stabilise entre 0,93 et 0,94 et ne varie que très peu. Nous fixons donc cette valeur pour `ntree`.

Concernant la valeur optimale de `mtry`, nous décidons de conserver la valeur par défaut. En effet, celle-ci est égale à $p/3$, p étant le nombre de variables explicatives, ce qui nous donne `mtry` $= 3$. Malheureusement, changer cette valeur rallonge beaucoup trop la durée d'exécution (de l'ordre d'une demi-heure), ce que nous considérons comme trop pénalisant comparé aux autres méthodes. Étant donné que nous n'avons retenu que 9 variables explicatives dans notre jeu de données, nous acceptons le paramètre par défaut. Notre forêt « optimale » nous donne donc les résultats suivants :

RMSE	**Score R^2**
2502,79	0,9389

TABLE 14 – Scores obtenus par une forêt aléatoire optimisée (Cars)

Les scores obtenus sont très proches de ceux de la forêt par défaut : ils sont légèrement meilleurs. Cela signifie que nous pouvons retrouver un bon modèle en retirant 40% des arbres qui constituaient la première forêt. Cette simplification est salutaire pour les durées de construction des forêts : en effet, nous descendons à environ deux minutes de temps de calcul pour une forêt.

Importance des variables

Les arbres et les forêts que nous avons générés sont basés sur des échantillons différents, en ce sens qu'une variable a été écartée pour la construction des forêts. C'est peut-être cela qui a une incidence sur la qualité des prévisions. Afin de vérifier cette hypothèse, nous pouvons faire appel à la mesure de l'importance des variables.

Au sens de l'arbre *CART* optimal, l'importance des variables est ordonnée comme suit :

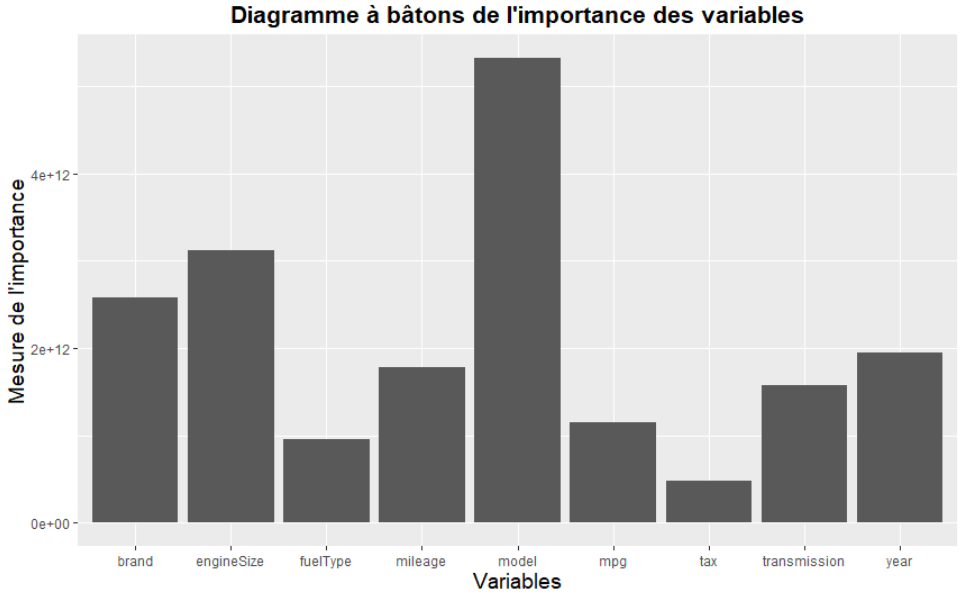

FIGURE 18 – Importance des variables selon l'arbre *CART* optimal (Cars)

Nous pouvons constater que le modèle du véhicule a une place prépondérante dans l'estimation de sa valeur, ce qui confirme une intuition commune en la matière. Viennent ensuite la cylindrée du véhicule puis sa marque. Voyons maintenant comment les forêts estiment l'importance des variables :

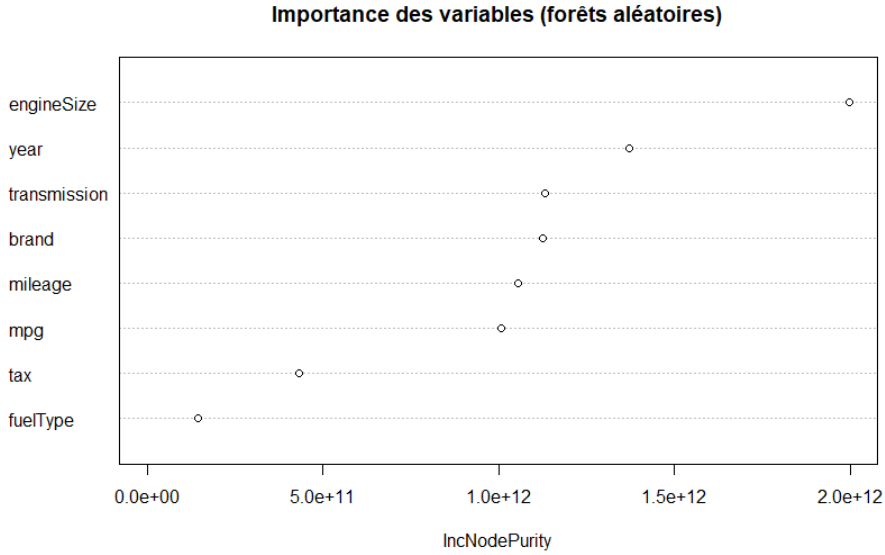

FIGURE 19 – Importance des variables selon la forêt aléatoire optimale (Cars)

Privés de l'information sur les modèles des véhicules, les forêts accordent une grande importance à leur cylindrée, puis à leur ancienneté. La marque du véhicule est reléguée à la quatrième place, au même niveau que le type de transmission. Nous en déduisons que la forêt optimale, qui fournit des résultats très satisfaisants malgré l'absence d'une variable clé, sait récupérer l'information manquante dans les autres variables. Ceci est une bonne illustration de la robustesse de la méthode.

Nous avons ainsi une idée de la manière dont les arbres *CART* et les forêts procèdent pour déduire des prévisions sur le prix des véhicules qui leur sont soumis. Nous voudrions maintenant savoir si les réseaux de neurones sont capables de fournir de tels résultats.

4.2.3 Comparaison avec les résultats obtenus par réseau de neurones (régresseur *MLP*)

La nature des données impose quelques traitements préalables pour la mise en œuvre d'un réseau de neurones. D'abord, les variables qualitatives sont encodées dans des indicatrices : elles valent 1 si l'observation appartient à la catégorie visée et 0 sinon. Nous obtenons donc une indicatrice par modalité des variables qualitatives, ce qui nous donne un total de 218 variables explicatives. Ensuite, les observations sont « mises à l'échelle » : cela signifie qu'elles sont multipliées, pour chaque variable, par un facteur d'échelle qui les replace dans l'intervalle $[0, 1]$.

Pour notre problème, nous bâtissons un modèle de régression à perceptrons multicouches (*MLP Regressor*) constitué comme suit : une première couche cachée de 25 neurones, une seconde couche cachée de 10 neurones, et enfin une couche de sortie avec un unique neurone. Le modèle est entraîné sur 30 périodes avec des portions de l'ensemble de l'apprentissage, appelées « lots », qui contiennent chacune 737 observations. Nous effectuons le suivi en temps réel de l'entraînement du réseau, au travers de l'évaluation par l'erreur quadratique moyenne (*MSE*) d'estimation et de prédiction :

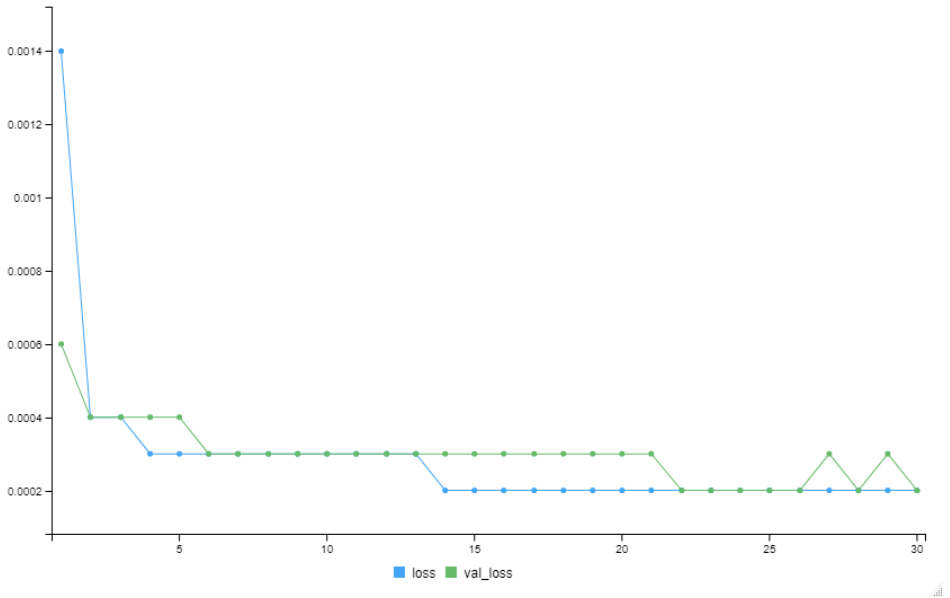

FIGURE 20 – Évaluation du modèle par réseau de neurones en temps réel (Cars)

L'entraînement dure environ une minute (2 secondes par période) et le réseau de neurones nous donne, après remise à l'échelle de départ des prédictions, les résultats suivants :

RMSE	**Score R^2**
2449,00	0,9406

TABLE 15 – Scores obtenus par un réseau de neurones (Cars)

Les prévisions sont d'une précision similaire à celles des arbres *CART* et des forêts.

4.2.4 Bilan de l'expérience

Sur ce jeu de données, les conclusions que nous formulons sont plus nuancées que pour le problème de classification précédent. En effet, nous devons intégrer des difficultés liées aux durées d'exécution de chaque programme comme au travail supplémentaire que peuvent nécessiter certaines méthodes.

Ici aussi, les meilleures prédictions ont été fournies par l'arbre optimal. L'arbre simplifié à 1 écart-type nous a permis de générer beaucoup plus rapidement des prédictions de toute aussi bonne qualité, ce qui peut s'avérer très utile en production, pour des applications destinées au grand public par exemple.

Si nous souhaitons pouvoir obtenir des analyses plus fines et éventuellement des prédictions plus robustes, alors la forêt aléatoire optimale constitue une bonne solution. En effet, les forêts aléatoires proposent une définition plus pertinente de l'importance des variables, comme nous l'avons évoqué dans la section 2.3. Toutefois, la génération des forêts aléatoires peut coûter très cher en temps de calcul, ce qui peut limiter l'exploitation de leur plein potentiel.

Enfin, les réseaux de neurones ont fourni une bonne qualité de prédiction pour une durée d'exécution raisonnable. Néanmoins, ils sont beaucoup plus difficiles à implémenter sur logiciel que les arbres et les forêts, et ils nécessitent un travail supplémentaire sur les données pour déduire de bonnes prédictions.

5 Conclusion

Nous avons mis à l'épreuve, tout au long de ce projet, un ensemble de méthodes de prévision, que nous avons comparées sur deux problèmes très divers de classification et de régression. Nous pouvons à présent faire un bilan général de ces expériences et conclure sur l'efficacité comparée des arbres de décision et des forêts aléatoires.

Nous avons d'abord constaté leur grande facilité d'utilisation. Les bibliothèques `rpart` et `randomForest` fournissent d'excellents outils prêts-à-l'emploi pour réaliser des prévisions de bonne qualité. Les arbres de décision et les forêts possèdent en outre de nombreux avantages des réseaux de neurones, tout en contournant certains de leurs défauts, notamment la durée d'entraînement des modèles et parfois un travail de traitement supplémentaire sur les jeux de données. Ils sont également réglables, et ce facilement, avec des paramètres accessibles et compréhensibles. Certaines bibliothèques `R` comme `caret` offrent même la possibilité à l'utilisateur de rechercher automatiquement les paramètres optimaux des forêts.

De plus, les arbres de décision exploitent tout leur potentiel sur des problèmes de classification : ils sont très faciles à interpréter, surtout dans leurs formes simplifiées. Complétés des forêts et de la capacité de ces dernières à mesurer l'importance des variables, ils permettent à l'utilisateur de comprendre véritablement ses données, plus que de simplement les prédire. Les arbres et les forêts aléatoires fonctionnent également tout aussi bien en régression, bien qu'ils perdent certaines de leurs qualités dans ce contexte.

Ainsi, quel que soit le problème à traiter, les méthodes d'arbres et de forêts sont des alliés au service du scientifique des données, des techniques qui pourront lui permettre de s'approprier les données et parfois de gagner du temps. Elles représentent en ce sens des outils indispensables.

Références

[1] Tin Kam Ho : Random decision forests. *In Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[2] Leo Breiman, Jerome H Friedman, Richard A Olshen et Charles J Stone : *Classification and Regression Trees*. Wadsworth and Brooks/Cole Monterey, CA, USA, 1984.

[3] Robin Genuer et Jean-Michel Poggi : *Les forêts aléatoires avec R*. Presses universitaires de Rennes, 2019.

[4] Leo Breiman : Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[5] Leo Breiman : Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Robin Genuer, Jean-Michel Poggi et Christine Tuleau : Random forests : some methodological insights. *arXiv preprint arXiv :0811.3619*, 2008.

[7] Tin Kam Ho : The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[8] Yoav Freund et Robert E Schapire : Experiments with a new boosting algorithm. *In icml*, volume 96, pages 148–156. Citeseer, 1996.

[9] Pierre Geurts, Damien Ernst et Louis Wehenkel : Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[10] Leo Breiman : Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[11] Andy Liaw et Matthew Wiener : Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

Annexe

Les jeux de données et les programmes utilisés dans ce rapport sont disponibles en téléchargement via le lien suivant :

`https://github.com/Baptiste-Biausque/projet-m1-forets-aleatoires`

La liste des bibliothèques `R` utilisées pour ce projet :

Nom	Utilisation
`tidyverse`	Un ensemble de bibliothèques utiles pour la gestion de données
`DMwR`	Mise à l'échelle des données (réseaux de neurones)
`rpart`	Construction des arbres *CART*
`rpart.plot`	Extension pour l'affichage des arbres
`randomForest`	Construction des forêts aléatoires
`caret`	Évaluation des prévisions et optimisation des forêts
`Tensorflow for R`	Un ensemble pour l'apprentissage automatique
`Keras`	Construction des réseaux de neurones