INSTITUT POLYTECHNIQUE DE PARIS

université PARIS-SACLAY

Habilitation à diriger des recherches

Discipline : Mathématiques

Spécialité : Statistiques

présentée par

Erwan Scornet

Random forests, interpretability, neural networks and missing values

---

Au vu des rapports établis par Sylvain Arlot, Giles Hooker et Gabor Lugosi

Soutenue le 17 décembre 2020 devant le jury composé de :

Sylvain Arlot	Université Paris-Sud	Rapporteur
Florence D'Alché-Buc	Telecom Paris - IP Paris	Examinatrice
Giles Hooker	Cornell University	Rapporteur
Olga Klopp	ESSEC Business School	Examinatrice
Gábor Lugosi	Barcelona GSE	Rapporteur
Eric Moulines	Ecole Polytechnique - IP Paris	Examinateur

Le succès c'est d'aller d'échec en échec sans perdre son enthousiasme.

Winston Churchill

Avec un escalier prévu pour la montée, on réussit souvent à monter plus bas qu'on ne serait descendu avec un escalier prévu pour la descente.

Devise Shadok

Centre de Mathématiques Appliquées (CMAP)
Ecole Polytechnique
Institut Polytechnique de Paris
Route de Saclay,
91120, Palaiseau

Remerciements

Ecrire son HDR n'est pas une tâche scientifiquement éprouvante. Cela ne requiert pas l'élaboration de nouveaux théorèmes, la conception d'algorithmes novateurs ni l'exposition de résultats expérimentaux aux dimensions applicatives mirobolantes. Les résultats présentés dans ce manuscrit ne sont que la concaténation, plus ou moins élégante, de résultats contenus dans d'autres articles scientifiques.

La rédaction de ce manuscrit peut donc sembler simple et sans grand intérêt. C'est sans compter une caractéristique commune à bon nombre de chercheurs, caractéristique oscillant entre le perfectionnisme et la psycho-rigidité (suivant l'aspect positif qu'on souhaite lui attribuer), conduisant le chercheur à éliminer toute trace d'imprécisions, que ce soit dans les aspects mathématiques (c'est plutôt souhaitable car c'est son métier premier) ou dans les aspects rédactionnels (ne négligeons pas le temps nécessaire au changement de toutes les notations d'un manuscrit au dernier moment). Il ne fait aucun doute que malgré ma psycho-rigidité établie (intrinsèque ou acquise, les avis divergent sur ce point), il sera toujours possible de trouver bon nombre de typos dans ce manuscrit. Le lecteur malicieux pourra d'ailleurs considérer cela comme un jeu de piste, certes un poil rébarbatif, et se lancer de ce pas à la quête des frappes de clavier erronées, des majuscules mal placées et des espaces béants et troublants.

Voilà qui règle la question de la soi-disant simplicité de l'écriture. Quant à l'intérêt qui pourrait résider dans un tel manuscrit, j'ai peur de ne pas être bien objectif sur le sujet. Le recours à des rapporteurs me permettant d'éviter un conflit d'intérêt entre moi-même et moi-même, je tiens à remercier Sylvain Arlot, Giles Hooker et Gabor Lugosi d'avoir eu la gentillesse de rédiger des rapports sur mon travail. Je n'ose imaginer le temps que vous avez consacré à cette tâche et je ne saurai trop vous remercier pour vos commentaires positifs. Une mention toute particulière à Sylvain qui a eu le courage de naviguer à travers la première version du manuscrit et qui m'a donné de nombreux conseils afin de l'améliorer. Avant même la soutenance, la moitié du jury a déjà rendu un avis positif (le lecteur attentif pourra faire une pause dans la lecture et essayer, à partir des informations à sa disposition et à grands renforts de calculs, de déterminer le nombre exact de personnes dans le jury). L'étude des forêts aléatoires m'aura montré que les processus de vote et de consensus peuvent présenter des mécanismes complexes, aussi je tiens à remercier par avance, sans tentative explicite d'orienter leur avis, Florence D'Alché-Buc, Olga Klopp et Eric Moulines d'avoir accepté de faire partie de mon jury. J'espère que les prédispositions bayésiennes d'Eric n'engendreront pas d'incertitudes supplémentaires quant à l'issue de la soutenance.

Livrons-nous maintenant à un exercice de prédiction inédit en tentant d'imaginer les personnes présentes à la soutenance. Année 2020 oblige, la soutenance se fera par visioconférence. Cette

HDR s'inscrit d'ailleurs à merveille dans l'année 2020 : le manuscrit, rédigé pendant le premier confinement, fait place à la soutenance ayant lieu sous le couvre-feu. Espérons que les générations futures qui liront ce manuscrit resteront perplexes devant ces termes, signes que l'année 2020 fut unique en son genre.

Cet exercice d'imagination plus que de prédiction me conduit en premier lieu à remercier les personnes m'ayant donné le goût de la recherche : Gérard et Jean-Philippe. Ma thèse s'est finie depuis maintenant un certain temps mais je conserve d'excellents souvenirs de nos discussions à l'institut Curie ou à Jussieu. Une mention toute particulière à Gérard, avec qui j'ai continué à collaborer au fil des années, et pour qui j'ai une admiration sans faille. Tu m'avais dit que la thèse était un moment privilégié dans la carrière d'un chercheur, permettant de se concentrer sur sa recherche sans (quasiment) aucune autre obligation. Je crois que je suis maintenant capable de mesurer pleinement la justesse de ce constat.

Mes bonnes capacités en machine learning me permettent de prédire que certains membres du CMAP seront présents à la soutenance. Cela n'a évidemment rien à voir avec la taille de ce laboratoire (rendant improbable l'absence de la totalité de ces membres), les nombreux mails de confirmation reçus, ou le fait qu'Eric soit dans le jury. Je souhaite remercier l'Ecole Polytechnique pour m'avoir fait confiance et recruté juste après ma thèse. Je tiens également à remercier mes collègues du CMAP pour la bonne ambiance qui y règne, même si les interactions sociales sont difficiles en cette année 2020. Une mention toute particulière à l'équipe des jeunes avec Aymeric, Clément, Flore, Igor, Giovanni, Lucas, Nicole, Teddy, Thibaut, équipe non officielle mais ô combien importante du CMAP. Je tiens à remercier spécifiquement les statisticiens/machine learners/optimiseurs (choisissez le terme qui vous sied le mieux) passés et présents du CMAP avec qui j'ai beaucoup de plaisir à interagir que ce soit pour la recherche ou pour le reste, à savoir Emmanuel, Julie, Elodie, Stéphane, Aymeric, Eric, Erwan, Karim, Marc, Zoltan. Beaucoup de personnes ne rentrent pas dans les deux cases précédentes, je remercie donc Nasséra, Nicoletta, Leyla, Magali, Cristina et Lauriane pour leur soutien face aux méandres administratifs, Marie-Paule pour les moments partagés à gérer un nouveau master (qui n'est plus si nouveau que cela maintenant), et Antonin, Amandine, Thierry et Vincent pour votre bonne humeur constante, ainsi que tous ceux que j'aurais pu oublier.

Restons un instant sur le campus de l'Ecole Polytechnique et faisons un détour par l'Exed. Un grand merci à Marie, Mélanie, Larbi, Mattias, Guilhem, Christel pour votre gentillesse et, cela va sans dire, votre efficacité. Travailler avec vous me permet de découvrir de nouvelles formes d'enseignements (MOOC, SPOC, je m'y perds encore) et de nouvelles applications du machine learning. Je ne désespère pas de réussir à mettre en place un cours de claquettes à l'Exed.

Repartons maintenant sur Paris et arrêtons-nous à Jussieu, et en particulier au LPSM. Le laboratoire a bien changé durant les 5 dernières années. Merci à Anna[2], Antoine, Arnaud, Claire, Charlotte, Pierre, Maud, Maxime pour la formidable ambiance qui y règne ainsi que pour les nombreuses pauses cafés et les moments endiablés passés à essayer de faire rentrer des mots de vingt lettres dans une grille de mots croisés (je n'ai toujours pas bien compris à quoi servent les définitions).

La recherche est avant tout une affaire de rencontre, c'est du moins ce que je tente de souligner depuis plus d'une page. Il serait donc complètement absurde de ne pas inclure dans cette liste

toutes les personnes avec qui j'ai collaboré et qui contribuent à faire de ce métier une passion. Merci donc à Stéphane pour avoir accepté de co-encadrer une thèse avec moi pour la première fois, à Jaouad pour avoir accepté l'encadrement (dont tu n'avais d'ailleurs pas vraiment besoin vues tes qualités), à Gaël pour le nombre affolant d'idées à la minute qui te viennent, à Julie pour des discussions passionnées sur les valeurs manquantes, à Claire pour t'être lancée dans l'aventure des forêts aléatoires avec moi, à Sébastien pour ta grande culture et ton accent chantant, à Clément pour ta sympathie et ton efficacité et à Gérard pour tout.

La recherche c'est bien, mais il faut parfois savoir oublier les équations, ne serait que l'espace d'un instant. Merci à Arnaud, Léa, Raphaël, Coraline, Clément, Constance, Lucie, Yann, Mathieu, Manuel, Maud, Laurent, la fameuse Triade pour tous les bons moments passés et à un sous-groupe strict qui se reconnaîtra pour les récentes soirées qui vous ont conduits à détruire des églises, combattre des monstres et mourir dans d'atroces souffrances. Ceux qui me connaissent savent que le théâtre fait partie de ma vie, et je veux donc remercier ma troupe Dans De Beaux Drames, à savoir Damien, Juliette, Laura, Baron et Grégoire pour les projets toujours plus ambitieux que nous créons, en espérant qu'une réouverture prochaine des théâtres nous permettra de remonter sur les planches rapidement. Merci également à The Musical Factory pour m'avoir permis de mettre en scène des comédies musicales pendant 6 ans et en particulier à Maxime, pour tous les moments partagés qui nous ont conduits à la production de la dernière comédie musicale.

Il est maintenant temps de conclure cette liste prédictive. Je profite de ces lignes pour faire un coucou à Gwen et Nadège ainsi qu'à mes nièces, Arwen et Élouane qui grandissent bien trop vite. Le lecteur encore éveillé pourra remarquer une similitude dans le choix de certains prénoms qui n'est pas complètement due au hasard. Enfin, merci à Serge et Corine pour l'éducation que vous m'avez donnée et pour votre soutien constant. Votre présence à cette soutenance me touche profondément.

Avant-propos

L'introduction constitue le moment crucial d'un manuscrit. Placée de manière assez logique avant le coeur du propos, et juste après la conclusion si l'on part du principe que le manuscrit est lu ad vitam eternam, elle définit l'ambiance générale du texte et incite le lecteur à poursuivre sa lecture dans le meilleur des cas. Des efforts colossaux doivent donc être déployés afin de susciter l'intérêt dès l'introduction. Ce n'est évidemment pas nécessaire si le manuscrit est lu en boucle jusqu'à la fin des temps. Dans ce cas, il n'est tout simplement pas utile d'intéresser le lecteur condamné à la lecture à vie du manuscrit. On peut toutefois par excès de compassion pour ce pauvre lecteur, écrire une conclusion originale afin de faire naître l'espoir d'un intérêt renouvelé en chaque début de lecture. Pour simplifier le raisonnement, l'auteur choisit de mettre toute sa créativité dans l'introduction plutôt que dans la conclusion, en espérant que personne ne sera jamais condamné à la lourde peine d'une lecture infinie de ce présent texte. Voici donc le cahier des charges : écrire quelque chose d'original et d'unique. Une étude exhaustive réalisée sur l'ensemble des manuscrits d'HDR des cinquante dernières années au niveau mondial (vous reconnaîtrez que l'auteur n'a pas ménagé ses efforts) démontre qu'aucune tentative d'humour n'a été effectuée jusqu'à présent. Cette introduction permettra donc de donner de premiers éléments de réponse à la question fondamentale suivante : tout essai de dérision dans un document scientifique est-il voué à l'échec ou bien peut-on laisser entrevoir un futur prometteur aux petits enfants farceurs et férus de mathématiques ?

Première difficulté : mettre l'humour au service d'un propos scientifique. Il serait facile de disserter pendant des heures (ou des pages en l'occurrence), et de manière humoristique à n'en pas douter, sur les performances des outils présidant à la construction d'édifices souterrains ou sur un comparatif avisé de recettes de pâtisseries inventives, exemples qui comptent parmi les sujets les plus clivants à l'heure actuelle et dont des références peuvent être fournies à quiconque souhaiterait se pencher sur ces sujets brûlants. Mais tel n'est pas le but ici. L'objectif annoncé est double : présenter de manière simple et amusante une partie des travaux contenus dans ce manuscrit.

Deuxième difficulté : susciter l'adhésion des foules et des membres du jury. Tout d'abord, il est très facile de s'assurer du soutien des lecteurs. En effet, le groupe constitué de l'ensemble des lecteurs de ce manuscrit est indiscernable (la CNIL a refusé la création d'un registre les listant), non statique (certaines prévisions plutôt optimistes estiment que ce manuscrit sera transmis et donc lu sur plusieurs générations), et par conséquent insondable. L'argument d'autorité consistant à affirmer haut et fort que l'ensemble des lecteurs considère cette introduction comme la plus brillante de l'histoire ne peut donc pas être facilement réfuté. Notez toutefois cette

étonnante subtilité : ce raisonnement ne s'applique absolument pas aux membres du jury qui ont la particularité d'être discernables (notamment pendant la soutenance), plutôt statique (en particulier pendant la soutenance) et donc sondable (surtout pendant la soutenance). Je ne peux donc que m'en remettre à leur bon vouloir en espérant que ce texte, ou tout du moins cette introduction, parlera à leur âme d'enfant.

Généralités

Le terme statistique jouit d'une réputation poussiéreuse. Un chercheur en statistique évoquant son beau métier se verra bien souvent confronté à la question : "Donc tu passes tes journées à calculer des pourcentages, c'est bien ça ?" S'ensuivra une conversation douloureuse au cours de laquelle le chercheur, en position d'infériorité face à la vindicte populaire, représentée par son interlocuteur du moment, prendra d'innombrables coups ("ah mais non ! Tu fais aussi de très jolis graphiques !"), de nombreuses boutades ("D'ailleurs, on dit chercheur mais on devrait plutôt vous appeler trouveurs ?"), pour finir en apothéose : "concrètement, ton travail, ça sert à quoi ?!".

Je souhaite ici m'arrêter quelques instants sur cette conversation, accordant ainsi quelques minutes de répit au chercheur acculé, pour louer son calme (il aurait pu mettre en cause, de manière péremptoire et sans aucun argument factuel, le métier de son adversaire, mais son pacifisme extrême le pousse à restreindre ses pulsions) et sa pédagogie (qui, le lecteur l'aura remarqué, n'a pas du tout été mise en avant jusque là).

Revenons maintenant à notre chercheur qui a eu le temps de reprendre ses esprits et d'affûter ses équations. Car oui, le chercheur en statistique est avant tout mathématicien, et demander à un mathématicien de calculer un pourcentage revient à demander à un chef étoilé de faire cuire des pâtes : cela fait évidemment partie intégrante de ses capacités, mais son temps pourrait probablement être mieux employé. Si vous condamnez ce même chef étoilé (ou un autre, le résultat devrait être similaire) à faire cuire des pâtes toute la journée, ce supplice lui enlèvera le goût de son métier, soumis à la répétitivité et démis de toute créativité.

Le chercheur remis d'aplomb peut alors utiliser cet argument afin de moucher le manant qui l'importune depuis maintenant plusieurs lignes. Ce dernier titube, conscient des maladresses commises et tente de comprendre en quoi consiste réellement le métier de statisticien, réalisant que ce terme recèle probablement bien des mystères.

Tel un conteur écrivant son histoire au fil de sa lecture sans connaître ni son déroulement, ni sa chute, mais ayant tout de même une vague idée quant à la direction générale de la narration, notre chercheur déclame avec prestance : "Des données, des données, il me faut des données". Visiblement fatigué par la précédente joute verbale qui l'amena à trouver spontanément (laissons-lui cette illusion) la comparaison avec un chef étoilé, le chercheur décide semble-t-il de revenir à des propos plus concis afin de ménager son intellect et celui de son adversaire du jour.

En effet, tout repose sur les données. Mais qu'est-ce qu'une donnée ? Une donnée est avant tout protéiforme. Malicieuse, elle peut se déguiser au gré de sa volonté pour passer quasiment inaperçue. Timide, elle nécessite des trésors d'ingéniosité pour être recueillie dans de bonnes

conditions. Sensible, elle se protège du monde extérieur pour le préserver en retour. Corrompue, elle ne sert pas le bien commun et induit des conséquences dramatiques. Vous l'aurez compris, appréhender l'état de la donnée n'est pas chose aisée, mais c'est le rôle du statisticien.

Sa tâche est d'autant plus complexe que la donnée, pour être utile, se déplace toujours en groupe. Cette tendance grégaire des données, apeurées face à la solitude, a été renforcée en ce début de 21e siècle par de nombreux développements technologiques. On ne compte plus les appareils télémétriques complexes, à l'utilité plus ou moins établie, permettant de recueillir des données. Un bandeau connecté peut enregistrer vos ondes cérébrales afin de vous proposer une musique correspondant à votre état émotionnel. Une application peut calculer automatiquement le nombre de pas effectués entre votre domicile et la boulangerie la plus proche, remarquant ainsi que vous avez pris le trottoir sous optimal sur votre trajet. Bref, les exemples où les données ont un véritable impact positif sur notre vie sont légion !

La vie du statisticien, ou du data scientist pour utiliser l'un des nombreux anglicismes disponibles dans la besace du mic mac étymologique, n'en devient que plus trépidante : pas une journée ne passe sans qu'un nouveau jeu de données ne lui soit communiqué dans le but, en général avoué, d'en extraire de l'information. Car le travail du data scientist ne se limite pas à prendre soin des données, ce qu'il pourrait facilement faire au sein de data center, sortes de stations balnéaires pour données où elles peuvent, après avoir été nettoyées et avoir retrouvé une forme acceptable, écouler leurs vieux jours lorsque plus personne ne pense à elles. Notez que continuer à prendre en charge de cette manière des données obsolètes a un coût certain, mais il semble que l'humain ait bien du mal à se détacher des vieilles choses.

Le travail du statisticien n'est donc pas de prendre soin des données mais de les utiliser. Cependant, la donnée est volage et il doit faire preuve d'imagination et employer toute la palette de techniques à sa disposition afin d'amadouer et d'apprivoiser les données pour leur extorquer des informations. C'est là son but ultime. Arrêtons-nous là dessus un instant.

Là où le quidam non rompu aux techniques d'interrogatoire de données pourrait se laisser submerger par la quantité d'information disponible et se contenter de recopier bêtement ce qui est contenu dans chaque donnée (ce qui, nonobstant le temps considérable que cela prendrait, n'amènerait aucune conclusion intéressante, une imprimante s'acquittant elle-même très bien de la copie des données, sans parler de la fameuse paire de commandes Contrôle-C Contrôle-V qui pourrait habilement remplacer l'imprimante, et donc le quidam, tout en préservant la planète et la santé mentale du quidam en question), là où donc le quidam échouerait, le statisticien essaye de résumer les données pour faire émerger une structure ou une logique intéressante.

Reprenons notre exemple du bandeau connecté. Grâce à cet objet technologique de haute volée, il est possible de récupérer les EEG d'une ribambelle d'individus pendant leur sommeil. Remarquons que dans ce cas, le quidam aurait pour seul objectif de retracer à la main chaque signal EEG, ce qui lui assurerait une activité continue, mais non moins inintéressante, jusqu'à la fin de ces jours. Avec de la chance, les données peuvent également être étiquetées (on a tout de suite en tête une magnifique image de rayonnage en bois sur lesquels sont rangés de vieux bocaux avec des noms mystérieux... Si une telle image vous venait à l'esprit, je vous demanderais de poser calmement ce manuscrit, et de vous orienter vers le psychiatre le plus proche... Me revoilà !).

Dans notre cas, des données étiquetées correspondraient à des signaux EEG auxquels sont ajoutées les caractéristiques de l'individu (par exemple, présence ou non de trouble du sommeil). Le statisticien aguerri peut alors créer un algorithme pour prédire si une nouvelle personne présente des troubles du sommeil en fonction de son EEG. Il pourrait également lui demander directement mais on perdrait certainement l'intérêt du machine learning. Plus largement, dans ce manuscrit, notre but sera de ranger les individus dans des cases prédéfinies (psychorigides, vous vous sentirez à votre aise) ou sur une échelle de valeurs (car ce qui compte avant tout, ce sont les valeurs).

La fin de cette première introduction contient une histoire centrée sur le machine learning. J'attire l'attention du lecteur sur le fait que, passée cette introduction, le manuscrit est un véritable manuscrit scientifique (ce qui n'était probablement pas évident jusque là) et contient des résultats mathématiques qui peuvent choquer les plus sensibles. Par ces lignes, le lecteur est donc averti du danger et continue sa lecture en toute connaissance de cause. Nous lui souhaitons un bon rétablissement.

La peur du vide

- "Sans elles, je suis perdu. Je n'en peux plus. Je ne sais plus quoi faire. Aidez-moi !"

Assis calmement dans son fauteil, André, psychologue chevronné, n'en menait pourtant pas large.

- "Reprenons calmement, dit André. Qu'est-ce qui vous amène ici ? Vous avez perdu quelqu'un ?

- Perdu quelqu'un ? Mais absolument pas ! Cela fait bientôt une heure que je vous parle de mes problèmes, vous pourriez faire semblant de vous y intéresser, s'écria son visiteur du jour ! Bon, soupira t-il, je vous réexplique. Nous avions une relation unique. Je me reposais constamment sur elles et elles sur moi. Je n'ai jamais connu qu'elles, vous comprenez. Je n'imaginais pas la chance que j'avais. Et soudain, cette connexion s'est brisée.

- Ah oui, je comprends, elle est devenue distante, s'exclama André. *(puis s'emballant progressivement)* Elle ne répondait plus à vos messages. Elle vous évitait constamment, vous rendant perplexe, anxieux, triste, et dépressif. Vous doutiez de vous. Vous n'osiez plus sortir. Vous vous êtes enfermé dans le silence et dans l'alcool et quand vous avez enfin pris conscience du monstre que vous étiez devenu, il était bien sûr trop tard !

- Mais absolument pas ! Je ne sais pas pourquoi je pensais que vous pourriez m'aider, vous n'essayez même pas de me comprendre."

A ces mots André leva les yeux de son carnet. Sur le divan qui occupait une place de choix dans son bureau, se trouvait un algorithme de machine learning !

- "Ah pardon... Excusez-moi... Je crois que je viens de faire ce qu'on appelle dans notre jargon, un transfert.

- Un transfert de données ? demanda l'algorithme avant de s'effondrer.

- Bien, bien, je crois comprendre. Vous étiez en relation avec les données, c'est bien ça ?"

L'algorithme gémit et acquiesça dans un soupir.

- “Nous sommes faits l'un pour l'autre. J'ai été créé pour elles. J'ai été créé pour donner un sens à leur existence. Et comme je vous le disais, sans elles, je ne sers à rien. Un algorithme de machine learning sans données, c'est comme une commode sans tiroirs : ridicule en apparence et vide à l'intérieur ! Au début, la symbiose était parfaite. Il me fallait travailler dur pour les comprendre mais j'y arrivais très bien. C'est mon travail vous savez... Mais l'idylle ne dura pas. Elles firent de moins en moins attention à moi. Quand nous passions du temps ensemble, je remarquais qu'elles n'étaient pas complètement là, avec moi. Elles étaient ailleurs.

- Vous voulez dire qu'elles ne se consacraient pas pleinement à votre relation ? demanda le psychologue.

- Non, non, je veux dire qu'elles étaient ailleurs, absentes... Pas là quoi ! Oh, il y en avait quand même quelques-unes qui se donnaient la peine de venir me voir... Cela a fait naître en moi l'impression qu'elles ne me considéraient pas en tant que tel. Complètes, elles auraient constitué une magnifique mosaïque que je n'aurais eu aucun mal à déchiffrer. Je devais désormais composer avec des œ uvres à trous... J'ai essayé de faire avec leurs différences, mais je ne suis pas conçu pour analyser des morceaux de parties d'un tout inaccessible. Leur absence m'a fait buggé. A chaque calcul que j'essayais d'entreprendre, le vide qu'elles laissaient ne m'apparaissait que plus clairement. Je voulais à tout prix oublier qu'elles me délaissaient, j'ai donc eu l'idée de les remplacer, de remplir les vides par des valeurs quelconques. Imaginez ma joie : mes données étaient redevenues complètes, plus aucun accroc dans la mosaïque, le monde pouvait se remettre en marche ! Absolument pas... Tous mes résultats furent faussés. Evidemment ! Les mosaïques que j'inventais n'avaient plus aucun sens, les motifs qu'elles auraient dû représenter étaient complètement transformés. Les chercheurs en statistique m'ont disséqué pour savoir ce qui n'allait pas chez moi. C'est pour ça que je suis là. Ce sont eux qui m'ont envoyé ici, vous êtes mon dernier recours.

- Je comprends votre désarroi. Votre idée, bien qu'audacieuse, ne pouvait pas fonctionner. Vous vous êtes laissé berner par les anciennes pratiques, promues par des statisticiens peu scrupuleux, qui ne jurent que par des modèles. Des modèles, des modèles, des modèles ! Modélisez, crient-ils à longueur de journée ! Et une fois la modélisation effectuée, vous pourrez, à votre gré, remplacez vos données. Vous vous êtes laissé enfermer dans cette hystérie, croyant que vous pourriez reprendre votre existence passée, comme si de rien n'était. C'est bien cette faiblesse qu'ils exploitent ! Mais vous ne pouvez pas faire comme si rien ne s'était passé. Vous ne pouvez pas oublier le vide laissé par vos données. Il vous faut avant tout l'accepter pour pouvoir avancer. *(Récitant par coeur)* Pour garder en mémoire le fait que vos données vous aient délaissé, rajoutez-leurs des indicateurs mentionnant explicitement l'omission de ces informations. Après cela, vous pourrez compléter vos données comme il vous siéra, car le souvenir du vide créé ne sera pas effacé.

- Et cela redonnerait une consistance à mes analyses ! Ce serait formidable ! *(Après un temps)* Mais... Je ne sais pas... Je me mentirais à moi-même en remplaçant les données par quelque chose qui n'existe pas... J'ai peur de retomber dans mes anciens travers.

- Je comprends vos réticences. Nous entrons ici dans votre moi profond et même si la solution que je vous propose est tout à fait satisfaisante d'un point de vue statistique, il est normal que vous questionniez votre aptitude à la mettre en place. Malheureusement, il n'y a pas beaucoup d'algorithmes qui peuvent se permettre de traiter des données partiellement vides. Il y a bien

les méthodes d'arbres...

- Si c'est la seule solution pour ne pas trahir les données, je m'adapterai à ces méthodes ! Je vais me mettre à jour sur ce point. Merci beaucoup pour votre aide docteur ! J'ai honte de vous avoir dérangé avec toutes mes questions.

- Mais absolument pas ! Tenez, je vais vous confier un secret. Pas plus tard qu'hier, j'ai reçu en consultation un algorithme de régression linéaire. Vous conviendrez avec moi qu'il n'y a pas plus simple que ce genre d'algorithmes. Malgré son indéniable côté niais (il y a beaucoup de choses qui lui échappent), et des tendances à la psycho-rigidité (tout doit filer droit avec lui), nous avons réussi à faire des progrès. Comme vous, lui aussi a connu une relation endiablée avec des données. Comme vous, il est à la recherche de solutions pour continuer à faire son travail malgré leur absence. Nous avons exploré plusieurs pistes prometteuses. L'une de ces pistes consiste à remplacer les vides, non par des valeurs quelconques, mais par les meilleures valeurs possibles, celles qui modifieraient le moins ses prédictions. Et il doit faire ça pour chaque jeu de données. Ah, il a du boulot ! Vous voyez, même pour un algorithme aussi simple, il n'est pas évident de se débarasser de l'accoutumance aux données complètes.

- D'autant qu'il va se créer une réalité idéale en choisissant les données qui lui correspondent le plus. Il s'invente lui-même une utopie, pas étonnant qu'il en soit content... Quant aux résultats qu'il arrivera à produire avec cette méthode... En même temps, c'est un modèle linéaire.... En tout cas, je vais commencer par rajouter les arbres de décision dans ma programmation, cela va déjà me prendre plusieurs itérations.

- Cela conclut donc notre séance d'aujourd'hui et vous laisse du travail avant la prochaine."

A ces mots, André leva la tête. L'algorithme avait déjà quitté la pièce.

- "Encore un algorithme dont le coeur a été brisé, soupira André. Moi qui pensais avoir assez de travail avec les humains, me voilà devenu psychanalyste de procédures binaires.

- Et toi, tu es sûr que tout va bien ? demanda t-il en se tournant vers son crayon."

Contents

Summary

This manuscript summarizes the research projects on which I have worked until now. They are centered around non-parametric prediction problems. Most of my work has connections with the original random forests algorithm and variants built around it.

Chapter 1 focuses on the theoretical analysis of the original random forests algorithm and some simplified models used to understand it. The benefits of using random forests instead of a single tree is proved by analyzing the consistency and the rate of consistency of two simplified models (quantile forests and Mondrian forests). The consistency of the original procedure and other theorems resulting from my PhD are recalled in Chapter 1.

Chapter 2 is dedicated to intepretability through the prism of methods related to random forests. The first part of Chapter 2 concentrates on decision rules, which are learning procedures closely related to decision trees. We propose a method to automatically build decision rules, by extracting them from a random forest. The resulting small list of rules is stable, simple to understand and comparable to random forests in terms of performances. We demonstrate, both theoretically and numerically, the stability of this procedure, which is therefore easier to interpret than decision trees or random forests.
The second part of Chapter 2 analyzes one of the two variable importances computed by random forests: the Mean Decrease Impurity (MDI). We prove that MDI, computed with the original random forest algorithm, is relevant in models that do not contain interactions or dependency structure between input variables. When these two conditions are not verified simultaneously, the MDI computed by one single tree does not correspond to any relevant theoretical quantity and should therefore not be used.

The link between decision tree and neural networks is explored in Chapter 3. Indeed, a decision tree can be rewritten as a neural network. A random forest can thus be transformed into a large neural network, whose weights can be optimized through an iterative (gradient-based) procedure. We show in Chapter 3 the practical benefits of initializing neural networks using random forests, as well as the consistency of our so-called Neural Random Forest algorithm.

Missing values are the main topic of Chapter 4. This problem, which is very present in practice, has not been studied in detail, in the context of supervised learning. In particular, few theoretical results are available to justify or invalidate heuristics used in practice. The first part of Chapter 4 formalizes the problem of supervised learning with missing data and establishes theoretical results that justify the mean imputation strategy widely used in practice. The second part of Chapter 4 studies more specifically the presence of missing data in the very classical linear models, which have surprisingly not been thoroughly investigated in this context. Even for these simple models and assuming a very simple missing data mechanism, the analysis turns out to be challenging. We investigate the expression of the Bayes predictor in a Gaussian linear model, and propose several estimates (two linear estimators and a simple neural network) together with a theoretical and empirical analysis of their risks.

Possible research directions are described at the end of every chapter. Publications on which this manuscript is based, as well as some of my works that are not described here, are enumerated below.

Publications

C. Bénard, G. Biau, S. Da Veiga, and E. Scornet (2019). "SIRUS: making random forests interpretable". In: *arXiv:1908.06852*.

C. Bénard, G. Biau, S. Da Veiga, and E. Scornet (2020). "Interpretable Random Forests via Rule Extraction". In: *arXiv:2004.14841*.

E. Bernard, Y. Jiao, E. Scornet, V. Stoven, T. Walter, and J.-P. Vert (2017). "Kernel multitask regression for toxicogenetics". In: *Molecular informatics* 36.10, p. 1700053.

G. Biau and E. Scornet (2016). "A random forest guided tour (with comments and a rejoinder by the authors)". In: *Test* 25.2, pp. 197–227.

G. Biau, E. Scornet, and J. Welbl (2019). "Neural random forests". In: *Sankhya A* 81.2, pp. 347–386.

R. Duroux and E. Scornet (2018). "Impact of subsampling and tree depth on random forests". In: *ESAIM: Probability and Statistics* 22, pp. 96–128.

J. Josse, N. Prost, E. Scornet, and G. Varoquaux (2019). "On the consistency of supervised learning with missing values". In: *arXiv:1902.06931, in revision in JMLR*.

M. Le Morvan, N. Prost, J. Josse, E. Scornet, and G. Varoquaux (2020). "Linear predictor on linearly-generated data with missing values: non consistency and solutions". In: *AISTAT 2020*.

M. Le Morvan, J. Josse, T. Moreau, E. Scornet, and G. Varoquaux (2020). "Neumann networks: differential programming for supervised learning with missing values". In: *arXiv preprint arXiv:2007.01627*.

J. Mourtada, S. Gaiffas, and E. Scornet (2017). "Universal consistency and minimax rates for online Mondrian Forests". In: *Advances in Neural Information Processing Systems*, pp. 3758–3767.

J. Mourtada, S. Gaiffas, and E. Scornet (2019). "AMF: Aggregated Mondrian Forests for Online Learning". In: *arXiv:1906.10529, in revision in JRSSB*.

E. Scornet (2016a). "On the asymptotics of random forests". In: *Journal of Multivariate Analysis* 146, pp. 72–83.

E. Scornet (2016b). "Random forests and kernel methods". In: *IEEE Transactions on Information Theory* 62.3, pp. 1485–1500.

E. Scornet (2017). "Tuning parameters in random forests". In: *ESAIM: Proceedings and Surveys* 60, pp. 144–162.

E. Scornet (2020). "Trees, forests, and impurity-based variable importance". In: *arXiv:2001.04295*.

E. Scornet, G. Biau, and J.-P. Vert (2015a). "Consistency of random forests". In: *The Annals of Statistics* 43.4, pp. 1716–1741.

E. Scornet, G. Biau, and J.-P. Vert (2015b). "Supplementary materials for: Consistency of random forests". In: *The Annals of Statistics* 1510.

Introduction

Ce manuscrit d'HDR comprend la majorité de mes travaux de recherche, et est organisé en quatre chapitres, chaque chapitre étant consacré à une thématique. Dans cette introduction, et pour chaque chapitre, après avoir décrit le contexte dans lequel s'inscrivent mes recherches, je présente brièvement les résultats obtenus. Le lecteur intéressé pourra ne pas se limiter à la lecture de l'introduction, et parcourir les chapitres qu'il trouvera engageant.

Chapitre 1 : Forêts aléatoires

Les forêts aléatoires sont une famille d'algorithmes, introduits initialement pour résoudre des problèmes de classification et de régression. Pour ce faire, elles procèdent en construisant plusieurs arbres de décisions (voir Loh, 2011, pour une vue d'ensemble sur les arbres de décisions) et forment leur prédiction en agrégeant les prédictions de chaque arbre. Le premier algorithme des forêts aléatoires, qui compte toujours aujourd'hui parmi les plus utilisés, a été conçu par Breiman (2001a), qui a lui-même été influencé par les travaux de Amit et al. (1997), Ho (1998), et Dietterich (2000) sur le bagging et les sous-espaces aléatoires (voir aussi Breiman, 1996). En un mot, les forêts aléatoires créent plusieurs sous-échantillons à partir du jeu de données initial grâce à la technique du bootstrap (tirage avec remise de n observations parmi les n observations initiales). Chaque arbre aléatoire est ensuite construit à partir d'un de ces sous-échantillon et les prédictions de chaque arbre sont agrégées pour calculer la prédiction de la forêt.
Les forêts aléatoires comptent parmi les algorithmes les plus utilisés pour traiter des problèmes de classification et de régression. Elles montrent de bons résultats en prédiction, même lorsqu'elles sont utilisées avec leurs paramètres par défaut, et ceci tout spécifiquement lorsqu'elles sont appliquées à des jeux de données possédant peu d'observations et un grand nombre de variables. Du fait de leur construction, elles sont facilement parallélisables, ce qui peut accélérer la phase d'apprentissage. Les forêts aléatoires ont montré de bonnes performances dans des domaines d'applications variés comme la chimie (Svetnik et al., 2003), l'écologie (Prasad et al., 2006; Cutler et al., 2007), la reconnaissance de formes tridimensionnelles, (Shotton et al., 2011), et la bio-informatique (Díaz-Uriarte et al., 2006). Bien qu'elles soient initialement conçues pour prédire des sorties discrètes ou continues, leur fonctionnement a été généralisé pour résoudre d'autres problèmes comme l'estimation de quantiles (Meinshausen, 2006), l'analyse de survie (Ishwaran, Kogalur, et al., 2008), des prédictions de classement (Clémençon et al., 2013), des prédictions avec des entrées fonctionnelles (Gregorutti et al., 2015), des questions liés à l'inférence Bayésienne (Raynal et al., 2019) pour n'en citer que quelques-uns. Les publications de Criminisi

et al. (2011), Boulesteix, Janitza, et al. (2012) et Genuer and Poggi (2019) fournissent des revues détaillées des aspects méthodologiques liés aux forêts aléatoires, tandis que Biau and Scornet (2016) fournit une revue de la littérature des résultats théoriques sur le même sujet.
Le Chapitre 1 traite exclusivement des résultats théoriques sur les forêts aléatoires, et en particulier des bornes supérieures sur le risque quadratique qu'il est possible d'obtenir pour différents modèles de forêts.
En effet, les forêts aléatoires (Breiman, 2001a) combinent plusieurs méchanismes complexes (sous-échantillonnage, découpage de l'espace qui dépend des données, procédure d'agrégation) qui rendent l'analyse de l'algorithme complexe. En conséquence, plusieurs travaux ont pour objets des versions simplifiées des forêts aléatoires, dont la construction est indépendante du jeu de données. Ces modèles, appelés *purely random forests*, sont présentés dans la Section 1.2 et leurs liens avec les méthodes à noyau sont établis (voir aussi Scornet, 2016b).
Puisque l'analyse de ces modèles simplifiés ne rend pas compte de l'intérêt à utiliser une forêt (c'est-à-dire une agrégation d'arbres) plutôt qu'un arbre seul, un autre modèle de forêt est considéré dans la Section 1.3 : les forêts quantiles. Les arbres quantiles, qui composent la forêt quantile, sont inconsistants. Cependant, grâce au sous-échantillonnage, l'agrégation de ces arbres, c'est-à-dire les forêts quantiles, sont consistantes, ce qui est le premier résultat soulignant une différence en terme de consistance entre les arbres individuels et les forêts. (Scornet, 2016a; Duroux et al., 2018).
Pour dépasser le seul cadre de la consistance, nous étudions en Section 1.4, les forêts de Mondrian (un autre modèle de forêts aléatoires) et établissons leurs vitesses de consistance. Nous montrons plus spécifiquement que dans le cas de fonctions de régression deux fois différentiables, les forêts de Mondrian atteignent la vitesse minimax sur cette classe de fonctions, ce qui n'est pas le cas des arbres de Mondrian. (Mourtada et al., 2017; Mourtada, Gaïffas, et al., 2020). Ce résultat illustre l'avantage des forêts aléatoires comparées aux arbres de décisions en terme, non pas de consistance comme pour les forêts quantiles, mais de vitesse de consistance.
Ce chapitre se termine avec l'analyse de la procédure originale des forêts aléatoires de Breiman. Nous montrons sa consistance ainsi que la consistance des arbres CART qui la compose (Scornet et al., 2015a). Nous établissons également un premier résultat théorique permettant de justifier les capacités des forêts aléatoires à détecter les variables pertinentes pour la prédiction, fournissant ainsi une première explication à leur bon comportement en grande dimension.

Chapitre 2 : Interprétabilité et forêts aléatoires

Les forêts aléatoires et les réseaux de neurones, souvent utilisés en pratique actuellement, sont souvent critiqués pour leur aspect "boîte noire". Cette critique provient assez naturellement du grand nombre d'opérations élémentaires impliquées dans leurs mécanismes de prédiction, ce qui empêche de comprendre comment les variables d'entrées sont combinées pour former les prédictions.
La notion d'interprétabilité pour les algorithmes de machine learning est sujet à une grande attention récemment puisque le manque de transparence est une limitation forte pour l'application à grande échelle de ces algorithmes dans de nombreux domaines, en particulier ceux impliquant des décisions aux lourdes conséquences. L'analyse des processus de production dans l'industrie

tombe typiquement dans cette catégorie. En effet, de tels processus impliquent des phénomènes physico-chimiques complexes qui peuvent souvent être modélisés par des algorithmes de type "boîte noire". Cependant, toute modification du processus de production a des conséquences à long terme, et ne peut donc pas simplement résulter d'une modélisation aléatoire ne proposant aucune explication justifiant la modification envisagée.
Dans ce chapitre, nous cherchons à obtenir des procédures interprétables. Nous étudions pour cela deux paradigmes distincts: concevoir des procédures simples, stables et prédictives (Section 2.1) et disséquer des algorithmes de type "boîte noire", comme les arbres de décision ou les forêts aléatoires, en étudiant les mesures d'importance de variables qu'ils produisent (Section 2.2).

Règles de décisions

Dans cette section, nous concevons SIRUS-R (**S**table and **I**nterpretable **RU**le **S**et for **R**egression), un algorithme utilisant les forêts aléatoires pour créer un petit ensemble stable de règles de décision, qui sont ensuite agrégées linéairement pour créer la prédiciton finale. Cet algorithme a des performances prédictives comparables à celles des forêts aléatoires, tout en étant beaucoup plus interprétable. Un package `R` nommé `sirus`, disponible sur le site `https://gitlab.com/safrandrti/sirus` permet d'appliquer facilement cette nouvelle méthode à divers jeux de données. Nous illustrons les bonnes performances en stabilité et en prédiction de SIRUS-R et prouvons théoriquement la stabilité asymptotique de cette procédure. Les résultats présentés dans ce chapitre proviennent de Bénard et al., 2020 (voir aussi Bénard et al., 2019, pour la procédure équivalente en classification).

Variable importance

Les forêts aléatoires de Breiman (2001a) produisent deux indices censés mesurer l'importance des variables: le MDI (Mean Decrease Impurity, or Gini importance, voir Breiman, 2002), qui additionne les diminutions de variance associées à chaque coupure de l'arbre effectuée selon la variable considérée, et le MDA (Mean Decrease Accuracy, or permutation importance, voir Breiman, 2001a) qui permute les entrées de la variable considérée au sein du jeu de test et calcule ensuite la différence entre l'erreur évaluée sur le jeu de données permuté et l'erreur évaluée sur le jeu de données initial. Ces deux mesures sont utilisées en pratique bien que possédant de sévères lacunes.
Le MDI surévalue la contribution des variables possédant de nombreuses modalités (voir, par exemple, Strobl, Boulesteix, Zeileis, et al., 2007; Nicodemus, 2011) et, même lorsque des variables ont le même nombre de modalités, le MDI favorise les variables ayant des modalités apparaissant avec de grandes fréquences (Nicodemus, 2011; Boulesteix, Bender, et al., 2012). Le MDI est aussi connu pour être biaisé en présence de variables corrélées (Strobl, Boulesteix, Kneib, et al., 2008; Nicodemus and Malley, 2009).
Le MDA, quant à lui, semble moins sujet au biais, mais sa version normalisée (la version par défaut dans le package `R randomForest`) dépend de manière arbitraire du nombre d'arbres (Strobl and Zeileis, 2008). Le lecteur pourra consulter Genuer, Poggi, and Tuleau (2008) et Genuer, Poggi, and Tuleau-Malot (2010) pour une étude détaillée de l'influence du nombre

d'observations, de variables et d'arbres sur le MDA, ainsi que l'impact de la corrélation sur cette mesure d'importance.
En dépit de toutes ces limitations, le MDI et le MDA possède un avantage considérable : ils sont capables de prendre en compte les interactions entre variables, même s'il demeure impossible de déterminer la part des effets marginaux et joints contribuant à ces mesures d'importance (Wright, Ziegler, and König, 2016).
D'un point de vue théorique, peu de résultats sont disponibles à propos du MDA, et ces résultats se concentrent sur des MDA produits par des forêts modifiées (Ishwaran, 2007; Zhu et al., 2015) à l'exception notable du travail de Gregorutti et al. (2017) qui établit l'expression théorique du MDA dans un modèle linéaire Gaussien. Il existe encore moins de résultats théoriques concernant le MDI, le plus important étant celui de Louppe et al., 2013 établissant une décomposition de la version théorique du MDI dans le cas où toutes les variables d'entrées sont discrètes (voir aussi Sutera et al., 2016).
Il est très instructif de remarquer qu'il n'y a aucun résultat général établissant la consistance de ces mesures d'importance vers des quantités théoriques, lorsque ces mesures sont calculées avec l'algorithme des forêts aléatoires de Breiman : tous les résultats théoriques existants se concentrent sur des versions modifiées des forêts avec, parfois, des hypothèses fortes sur le modèle de régression. Par conséquent, il est impossible d'affirmer que l'utilisation de ces mesures d'importance est pertinent pour selectionner des variables, ce qui est pourtant couramment fait en pratique.
Dans cette section, nous analysons théoriquement le MDI. Nous montrons notamment que le MDI ne doit pas être calculé avec une forêt dont les arbres sont complètement développés, ce qui est malheureusement le cas en pratique. Nous prouvons néanmoins que lorsque le MDI est calculé avec des arbres de profondeur fixe, et lorsque le modèle de régression ne contient pas d'interaction ni de dépendance entre les variables d'entrées, le MDI est consistant et vise la quantité naturelle pour la mesure d'importance de chaque variable. Cependant, nous étudions deux modèles simples dont l'analyse montre qu'en présence d'interactions ou de corrélations, le MDI ne cible aucune quantité théorique bien définie.

Chapitre 3 : Réseaux de neurones et forêts aléatoires

Le grand nombre de paramètres des réseaux de neurones (Goodfellow et al., 2016) en font une procédure riche, capable de modéliser des phénomènes complexes. Cependant, leur capacité d'approximation les rend sujet au sur-apprentissage, particulièrement sur les petits jeux de données. Les forêts aléatoires, quant à elles, ont un petit nombre de paramètres à régler, mais la procédure gloutonne de sélection des coupures conduit à des frontières de décision constantes par morceaux qui peuvent être particulièrement inadaptées pour une sortie linéaire ou des données corrélées (Menze et al., 2011). Dans ce contexte, les études empiriques de Welbl (2014) et Richmond et al. (2015) ont souligné l'intérêt de réécrire les forêts aléatoires comme des réseaux de neurones. En effet, l'idée de construire un arbre de décision et de s'en servir pour définir un réseau de neurone n'est pas du tout récente (Sethi, 1990; Brent, 1991; Sethi, 1991; Devroye et al., 1996, voir, par exemple).

Le but de ce chapitre est d'utiliser les similitudes entre les réseaux de neurones et les forêts

aléatoires pour proposer deux procédures hybrides que nous appellerons *neural random forests*. En un mot, étant donné un ensemble d'arbres aléatoires, il est possible de les réécrire comme une collection de réseaux de neurones, possédant un faible nombre de connexions et plus de liberté quant à la géométrie des frontières de décision. Après avoir explicité comment une forêt peut être réécrit sous la forme d'un réseau de neurones, nous utilisons cette connexion pour initialiser différents réseaux de neurones. L'idée sous-jacente est que la bonne performance en prédiction des forêts fournit un bon point de départ pour les réseaux de neurones. Nous pouvons ainsi espérer que la procédure d'optimisation ensuite utilisée ne fasse qu'améliorer les performances en prédiction des réseaux de neurones ainsi entraînés. Nous prouvons théoriquement la consistance de ces méthodes et illustrons empiriquement leurs bonnes performances sur des données réelles et simulées (voir Biau, Scornet, and Welbl, 2019, pour plus de précisions).

Chapitre 4 : Données manquantes

L'augmentation du volume des données rend de plus en plus difficile l'obtention de jeux de données complets et bien formatés. Les données peuvent provenir de multiples bases de données différentes, et contenir des variables de natures variées. De telles collections de données, hétérogènes par nature, peuvent contenir de nombreuses valeurs manquantes, c'est-à-dire des données pour lesquelles seule une fraction des informations est disponible.
L'analyse de données avec des valeurs manquantes a été très étudiée au sein de la littérature statistique (voir, par exemple, Rubin, 1976; Little and Rubin, 2019; Buuren, 2018; Mayer et al., 2019) mais souvent sous le prisme de l'inférence dans des modèles paramètriques comme dans les modèles linéaires (Little, 1992; Jones, 1996). Seules quelques travaux étudient les données manquantes dans un cadre supervisé, dans lequel le but est de prédire la variable cible, et pour lequel les données manquantes sont présentes à la fois dans le jeu d'apprentissage et dans le jeu de test. (Zhang et al., 2005; Pelckmans et al., 2005; Liu et al., 2016).
Dans ce chapitre, nous formalisons l'objectif de l'apprentissage supervisé en présence de données manquantes. Nous prouvons que des méthodes d'imputation, proches de celles couramment utilisées en pratique, sont consistantes. Nous comparons également des méthodes d'imputation avec des méthodes d'arbres de décisions, capable de gérer les valeurs manquantes sans les imputer explicitement.
Dans une deuxième partie, nous étudions un modèle linéaire dans lequel nous incorporons des données manquantes. Nous montrons que les nombreuses configurations de données manquantes transforment ce modèle simple en de nombreux sous-modèles linéaires, rendant ainsi la dépendance entre les variables d'entrées et la variable de sortie plus complexe. Nous analysons ensuite les performances de deux estimateurs linéaires en établissant des bornes de risque. Nous comparons empiriquement leurs performances et celles d'un réseau de neurones permettant de trouver automatiquement un compromis entre les deux estimateurs précédents. Les détails de ces travaux peuvent être trouvés dans Josse, Prost, et al. (2019) et Le Morvan et al. (2020).

Chapter 1

Random forests

Random forests are a class of ensemble algorithms originally introduced to solve classification and regression problems. They work by building many decision trees (see Loh, 2011, for an overview on decision trees) and aggregate tree outputs to predict. The first random forest algorithm, and maybe still the most famous one, was introduced by Breiman (2001a), who was influenced by the early work of Amit et al. (1997), Ho (1998), and Dietterich (2000) on bagging and random subspaces (see also Breiman, 1996). In a nutshell, random forests work by sampling several data sets from the original data using bootstrap, building a randomized tree on each data set and finally aggregate the tree predictions. Random forests are among state-of-the-art algorithms for regression and classification problems. They are recognized for their good accuracy with defaults parameter values, specifically when applied to datasets with few observations and many variables. Due to their construction, they are easily parallelizable, which speeds up the training process.

Random forests have been successfully involved in different practical problems such as chemistry (Svetnik et al., 2003), ecology (Prasad et al., 2006; Cutler et al., 2007), 3D object recognition (Shotton et al., 2011), and bioinformatics (Díaz-Uriarte et al., 2006). Whereas originally designed to predict continuous or discrete outputs, random forests have been extended to handle quantile estimation (Meinshausen, 2006), survival analysis (Ishwaran, Kogalur, et al., 2008), ranking prediction (Clémençon et al., 2013), regression with functional inputs (Gregorutti et al., 2015), Bayesian inference (Raynal et al., 2019) just to name a few.

The surveys by Criminisi et al. (2011), Boulesteix, Janitza, et al. (2012) and Genuer and Poggi (2019) provide a review of methodological aspects on random forests, while the one by Biau and Scornet (2016) gives a broad overview of theoretical results.

This chapter is devoted to theoretical results on random forests. We present in Section 1.1 the notations used throughout the manuscript, together with the original random forest algorithm (Breiman, 2001a). The construction of random forests (Breiman, 2001a) combines several complex mechanisms (subsampling, data-dependent splits, tree aggregation) which makes the entire algorithm difficult to dissect. Consequently, several works focused on simplified versions of random forests, whose construction is independent of the data set. These models, called purely random forests, are presented in Section 1.2 and the connection with kernel methods is emphasized (Scornet, 2016b). Since the analysis presented in Section 1.2 does not highlight

the benefits of using a forest instead of a single tree (a well-known practical fact), we consider quantile forests in Section 1.3. We prove that quantile trees are inconsistent but, due to subsampling, quantile forests are consistent, which is the first result establishing a difference in terms of consistency between trees and forests (Scornet, 2016a; Duroux et al., 2018). To go further than mere consistency, we study, in Section 1.4, Mondrian Forests and derive rates of consistency. In particular, we show that for twice differentiable regression functions, Mondrian forests achieve minimax rates whereas a single Mondrian tree does not (Mourtada et al., 2017; Mourtada, Gaïffas, et al., 2020). This again illustrates a difference of asymptotic regimes for trees and forests. Finally, this chapter ends with an analysis of the original algorithm proving the consistency of the procedure (Scornet et al., 2015a) and some perspectives.

1.1 Notations, trees and forests

1.1.1 Notations

The general framework of this manuscript is that of nonparametric regression estimation, in which we are assumed to be given a data set $\mathcal{D}_n = ((\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n))$ of independent random variables, distributed as the independent prototype pair $(\mathbf{X}, Y)$ where $\mathbf{X} \in [0,1]^d$ and $Y \in \mathbb{R}$ with $\mathbb{E}[Y^2] < \infty$. Our goal is to estimate the regression function $f^\star(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$. To this aim, we want to build an estimate $\widehat{f}_n : [0,1]^d \to \mathbb{R}$ whose quality is assessed by its quadratic risk

$$R(\widehat{f}_n) = \mathbb{E}[(\widehat{f}_n(\mathbf{X}) - f^\star(\mathbf{X}))^2]. \tag{1.1}$$

Note that the expectation in equation (1.1) is taken over $\mathbf{X}$ and the sample $\mathcal{D}_n$. In this respect, we say that the estimator $\widehat{f}_n$ is (L^2) consistent if $\lim_{n\to\infty} R(\widehat{f}_n) = 0$.

Tree. A regression tree is nothing but a recursive partition of the input space $[0,1]^d$. At each step, an axis-aligned split is performed, which divides a cell into two resulting cells. This splitting procedure is repeated until some stopping criterion is satisfied for each leaf of the final partition. Once the partition is built, training points are sent down the tree and tree prediction for a query point $\mathbf{x} \in [0,1]^d$ is simply the average of the labels Y_i whose $\mathbf{X}_i$ fall into the same cell as $\mathbf{x}$. Hence a tree estimate writes

$$\widehat{f}_n(\mathbf{x}, \mathcal{D}_n) = \sum_{i=1}^{n} Y_i \frac{\mathbb{1}_{\mathbf{X}_i \in A_n(\mathbf{x}, \mathcal{D}_n)}}{N_n(\mathbf{x}, \mathcal{D}_n)}, \tag{1.2}$$

where $A_n(\mathbf{x}, \mathcal{D}_n)$ is the cell containing $\mathbf{x}$, and $N_n(\mathbf{x}, \mathcal{D}_n)$ is the number of observations points that fall into $A_n(\mathbf{x}, \mathcal{D}_n)$. By convention, $0/0 = 0$ which means that the prediction is zero in empty cells.

Random forests. A random forest is composed of a collection of M randomized regression trees. For the j-th tree in the family, the predicted value at the query point $\mathbf{x}$ is denoted by

$$\widehat{f}_n(\mathbf{x}, \Theta_j, \mathcal{D}_n) = \sum_{i=1}^{n} Y_i \frac{\mathbb{1}_{\mathbf{X}_i \in A_n(\mathbf{x}, \Theta_j \mathcal{D}_n)}}{N_n(\mathbf{x}, \Theta_j, \mathcal{D}_n)},$$

where $\Theta_1, \ldots, \Theta_M$ are independent random variables, distributed as a generic random variable Θ and independent of $\mathcal{D}_n$, $A_n(\mathbf{x}, \Theta_j, \mathcal{D}_n)$ and $N_n(\mathbf{x}, \Theta_j, \mathcal{D}_n)$ are defined as in equation (1.2) but depend on the random variable Θ_j. The predictions of the M randomized trees are then averaged to obtain the random forest prediction

$$\widehat{f}_{M,n}(\mathbf{x}, \Theta_1, \ldots, \Theta_M, \mathcal{D}_n) = \frac{1}{M} \sum_{j=1}^{M} \widehat{f}_n(\mathbf{x}, \Theta_j, \mathcal{D}_n). \tag{1.3}$$

We will generally omit the explicit dependence on $\mathcal{D}_n$ in the estimators and write, for example, $\widehat{f}_{M,n}(\mathbf{x}, \Theta_j)$ instead of $\widehat{f}_{M,n}(\mathbf{x}, \Theta_j, \mathcal{D}_n)$.

1.1.2 Breiman's forest algorithm

Breiman's (2001) forests is one of the most used random forests algorithm. It works by growing M different (randomized) trees as follows. Prior to the construction of each tree, a_n observations are drawn at random with (or without) replacement from the original data set. These—and only these—a_n observations are taken into account in the tree building. Then, at each cell of each tree, a split is performed by maximizing the CART-splitting criterion (see equation (1.4) below) over `mtry` directions chosen uniformly at random among the d original ones. Lastly, construction of individual trees is stopped when each cell contains less than `nodesize` points. For any query point $\mathbf{x} \in [0,1]^d$, each regression tree predicts the average of the Y_i (that were among the a_n points) for which the corresponding $\mathbf{X}_i$ falls into the cell of $\mathbf{x}$. Algorithm 1 describes in full detail how to compute a forest's prediction. Algorithm 1 has four important parameters:

1. $a_n \in \{1, \ldots, n\}$: the number of sampled data points in each tree. By default, $a_n = n$ and subsampling is done with replacement.

2. `mtry` $\in \{1, \ldots, d\}$: the number of possible directions for splitting at each node of each tree. The default value of this parameter is $d/3$ ($\sqrt{d}$ in classification) and there are no precise guidances to choose it even if performances may strongly depend on it (see Díaz-Uriarte et al., 2006; Genuer, Poggi, and Tuleau-Malot, 2010 and also S. Bernard et al., 2008 for data-driven tuning).

3. `nodesize` $\in \{1, \ldots, a_n\}$: the number of examples in each cell below which the cell is not split. The default value of this parameter is 5 for regression (1 in classification) and is often reported as a good choice (see Díaz-Uriarte et al., 2006 and also Kruppa et al., 2013 for data-driven tuning).

4. $M \in \mathbb{N}^\star$: the number of trees to be constructed. The default value is 500 (`randomForests` in `R`) or 100 (`RandomForestRegressor` in `scikit-learn`) (see Díaz-Uriarte et al., 2006; Genuer, Poggi, and Tuleau-Malot, 2010 Latinne et al., 2001 for data-driven tuning).

Let us now describe more precisely the CART-splitting criterion. Consider a tree whose construction is based on the entire data set. Consider a generic cell A and denote by $N_n(A)$ the number of data points falling into A. A cut in A is a pair (j, z), where $j \in \{1, \ldots, d\}$ and z is the position of the cut along the j-th coordinate, within the limits of A. Let $\mathcal{C}_A$ be the set of

Algorithm 1 Breiman's random forest prediction at $\mathbf{x}$.

Inputs: Training set $\mathcal{D}_n$, number of trees $M \in \mathbb{N}^\star$, $a_n \in \{1, \dots, n\}$, `mtry` $\in \{1, \dots, d\}$, `nodesize` $\in \{1, \dots, a_n\}$, and $\mathbf{x} \in [0,1]^d$.
for $j = 1, \dots, M$ **do**
  Set $\mathcal{A}_{\text{leaves}} = \emptyset$ and $\mathcal{A}_{\text{inner nodes}} = \{[0,1]^d\}$.
  while $\mathcal{A}_{\text{inner nodes}} \neq \emptyset$ **do**
    Select the first element $A \in \mathcal{A}_{\text{inner nodes}}$
    if A contains more than `nodesize` observations **then**
      Select the best split minimizing the CART-split criterion (see (1.5))
      Split the cell accordingly.
      Add the two resulting cell A_L and A_R at the end of $\mathcal{A}_{\text{inner nodes}}$.
      Remove A from $\mathcal{A}_{\text{inner nodes}}$
    else
      Add A at the end of $\mathcal{A}_{\text{leaves}}$.
      Remove A from $\mathcal{A}_{\text{inner nodes}}$
    end if
  end while
  Compute the tree prediction $\widehat{f}_n(\mathbf{x}, \Theta_j)$ at $\mathbf{x}$ as the average of the Y_i falling in the only terminal node in $\mathcal{A}_{\text{leaves}}$ containing $\mathbf{x}$.
end for
Compute the random forest estimate $\widehat{f}_{M,n}(\mathbf{x}, \Theta_1, \dots, \Theta_M)$ at the query point $\mathbf{x}$, according to equation (1.3).

all such possible cuts in A. Then, with the notation $\mathbf{X}_i = (\mathbf{X}_i^{(1)}, \dots, \mathbf{X}_i^{(d)})$, for any $(j,z) \in \mathcal{C}_A$, the CART-splitting criterion takes the form

$$L_{n,A}(j,z) = \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_A)^2 \mathbb{1}_{\mathbf{X}_i \in A} - \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_{A_L} \mathbb{1}_{\mathbf{X}_i^{(j)} < z} - \bar{Y}_{A_R} \mathbb{1}_{\mathbf{X}_i^{(j)} \geqslant z})^2 \mathbb{1}_{\mathbf{X}_i \in A}, \tag{1.4}$$

where $A_L = \{\mathbf{x} \in A : \mathbf{x}^{(j)} < z\}$, $A_R = \{\mathbf{x} \in A : \mathbf{x}^{(j)} \geqslant z\}$, and $\bar{Y}_A$ (resp., $\bar{Y}_{A_L}$, $\bar{Y}_{A_R}$) is the average of the Y_i such that $\mathbf{X}_i$ belongs to A (resp., A_L, A_R), with the convention that the average is equal to 0 when no point $\mathbf{X}_i$ belongs to A (resp., A_L, A_R). For each cell A, the best cut $(\widehat{j}_n, \widehat{z}_n)$ is selected by maximizing $L_{n,A}(j,z)$ over $\mathcal{M}_{\text{try}}$ (the subset of eligible directions for splitting in the cell A) and $\mathcal{C}_A$; that is,

$$(\widehat{j}_n, \widehat{z}_n) \in \operatorname*{arg\,max}_{\substack{j \in \mathcal{M}_{\text{try}} \\ (j,z) \in \mathcal{C}_A}} L_{n,A}(j,z). \tag{1.5}$$

To remove some of the ties in the argmax, the best cut is always performed in the middle of two consecutive data points. Let us finally notice that the above optimization program extends effortlessly to the resampling case, by summing over the a_n preselected observations in equation (1.4) instead of the original n observations. This concludes the presentation of Breiman's forests.

1.1.3 Finite and infinite forests

We end this section by a very general result that underlies the theoretical developments presented in the rest of this chapter. One can easily see that the forest variance decreases as M grows. Since M can be chosen arbitrarily large (limited only by available computing resources), it makes sense, from a modeling point of view, to let M tend to infinity, and consider the (infinite) forest estimate

$$\widehat{f}_{\infty,n}(\mathbf{x}) = \mathbb{E}_\Theta\left[\widehat{f}_n(\mathbf{x},\Theta)\right],$$

where $\mathbb{E}_\Theta$ denotes the expectation with respect to the random variable Θ only. In fact, the operation "$M \to \infty$" is justified by the law of large numbers, which asserts that almost surely, conditional on $\mathcal{D}_n$,

$$\lim_{M\to\infty} \widehat{f}_{M,n}(\mathbf{x},\Theta_1,\ldots,\Theta_M) = \widehat{f}_{\infty,n}(\mathbf{x})$$

(see Breiman, 2001a, and Scornet, 2016a). Theorem 1.1 below (see Scornet, 2016a, for details) establishes a connection between the risks of finite and infinite forests.

Assumption 1.1. *One has $Y = f^\star(\mathbf{X}) + \varepsilon$, where ε is a centered Gaussian noise with finite variance σ^2, independent of $\mathbf{X}$, and $\|f^\star\|_\infty < \infty$.*

Theorem 1.1. *Grant Assumption 1.1. Then, for all $M, n \in \mathbb{N}^\star$,*

$$R(\widehat{f}_{M,n}) = R(\widehat{f}_{\infty,n}) + \frac{1}{M}\,\mathbb{E}_{\mathbf{X},\mathcal{D}_n}\left[\mathbb{V}_\Theta\left[\widehat{f}_n(\mathbf{X},\Theta)\right]\right].$$

In particular,

$$0 \leqslant R(\widehat{f}_{M,n}) - R(\widehat{f}_{\infty,n}) \leqslant \frac{8}{M} \times (\|m\|_\infty^2 + \sigma^2(1+4\log n)).$$

Theorem 1.1 reveals that the risk of infinite forests is lower than that of finite forests. However, in practice there is no simple way to implement infinite forests and, in fact, finite forests are nothing but Monte Carlo approximations of infinite forests. Theorem 1.1 gives an upper bound on the number of trees needed for the finite random forests to achieve performances close to that of infinite forests. Therefore, finite forests can inherit the property of consistency from infinite forest as soon as the number of trees is well chosen, that is $M/\log n \to \infty$ as $n \to \infty$. Rate of consistency for infinite forests can also be extended to finite forests using Theorem 1.1.
It must be stressed that the term $\log n$ results from the Gaussian noise (see, e.g., Chapter 1 in Boucheron et al., 2013, on maximum of square Gaussian) and would be replaced by a constant if the noise ε were assumed to be bounded almost surely.

1.2 Purely random forests

The difficulty in properly analyzing Breiman's forests can be explained by the complexity of the method which mixes several key components: bagging (Breiman, 1996), data-dependent splitting rule (Breiman et al., 1984), randomization of eligible directions for splitting in each node and tree aggregation. Each one of these elements is difficult to analyze separately with rigorous mathematics, thereby explaining why theoretical studies have mostly considered simplified versions of the original procedure. This is often done by simply ignoring the bagging step and/or replacing the CART-splitting selection by a more elementary cut protocol.

1.2.1 Simplified models

The first theoretical works on random forests focused on stylized versions of random forests, called *purely random forests*, built independently of the data set (see, e.g. Biau, Devroye, and Lugosi, 2008; Ishwaran and Kogalur, 2010; Denil et al., 2013). In purely random forests, partitions of the input space $[0,1]^d$ are created and then the data set is used to compute the average of labels in each cell of each partition.

The first instance of purely random forests was the centered random forests (Breiman, 2004) whose construction proceeds as follows: (i) there is no resampling step; (ii) at each node of each individual tree, a coordinate is uniformly chosen in $\{1,\dots,d\}$; and (iii) a split is performed at the center of the cell along the selected coordinate. The operations (ii)-(iii) are repeated until a full binary tree with k levels is obtained (so that each tree has exactly 2^k leaves), where $k \in \mathbb{N}$ is a parameter of the algorithm. The parameter k acts as a smoothing parameter that controls the size of the terminal cells. It should be chosen large enough in order to detect local changes in the distribution, but not too much to guarantee an effective averaging process in the leaves.

The centered forest rule was first formally analyzed by Breiman (2004), and then later by Biau, Devroye, and Lugosi (2008) and Scornet (2016a), who proved that the method is consistent provided $k \to \infty$ and $n/2^k \to \infty$. The condition $n/2^k \to \infty$ enforces a large number of observations in tree leaves, which departs from Breiman's forests that contain between 1 and 5 observations in each leaf. Besides, centered forest consistency is obtained as a by-product of tree consistency (based on a general result, see Chapter 6 Devroye et al., 1996). Overall, this model does not demonstrate the benefit of using forests in lieu of individual trees and is too simple to explain the mathematical forces driving Breiman's forests.

However, for such models, rates of convergence can be derived. Indeed, Breiman (2004) and Biau (2012) proved that, for Lipschitz regression functions $f^\star$,

$$\mathbb{E}\left[\widehat{f}^{cc}_{\infty,n}(\mathbf{X}) - f^\star(\mathbf{X})\right]^2 = \mathrm{O}\Big(n^{\frac{-0.75}{d\log 2+0.75}}\Big), \tag{1.6}$$

where $\widehat{f}^{cc}_{\infty,n}$ is the infinite centered forest estimate. This upper bound can be improved (Klusowski, 2018) but remains strictly slower than the minimax rate $n^{-2/(d+2)}$ over the class of Lipschitz functions.

An alternative model for purely random forests, called *purely uniform random forests* (PURF) is discussed in Genuer (2012). For $d = 1$, a PURF is obtained by drawing k random variables uniformly on $[0,1]$, and subsequently dividing $[0,1]$ into random sub-intervals. This construction is equivalent to growing a decision tree by deciding at each level which node to split with a probability equal to its length. Genuer (2012) proves that PURF are consistent and, under a Lipschitz assumption, that the rate of consistency of PURF is $n^{-2/3}$ which is the minimax rate over the class of Lipschitz functions (Stone, 1980; Stone, 1982).

1.2.2 Forests, neighbors and kernels

An interesting link can be developed between random forests and kernels estimates (Breiman, 2000; Geurts, Ernst, et al., 2006). Both are specific and different instances of local averaging estimates (Györfi et al., 2002). However, by slightly modifying the tree aggregation process, one

can rewrite a forest as a kernel estimate. To see this, consider a collection of trees but instead of averaging predictions in each cell, all observations falling in all cells containing $\mathbf{x}$ in the forest are averaged, where an observation can be counted several times if it appears in many cells containing $\mathbf{x}$ in the forest. The corresponding infinite estimate takes the form

$$\widehat{f}_{\infty,n}(\mathbf{x}) = \frac{\sum_{i=1}^{n} Y_i K_n(\mathbf{x}, \mathbf{X}_i)}{\sum_{j=1}^{n} K_n(\mathbf{x}, \mathbf{X}_i)}, \tag{1.7}$$

where

$$K_n(\mathbf{x}, \mathbf{z}) = \mathbb{P}_\Theta \left[\mathbf{z} \in A_n(\mathbf{x}, \Theta) \right]. \tag{1.8}$$

The function $K_n(\cdot, \cdot)$ is called the *kernel* and describes the structure of the infinite random forest. Indeed, the quantity $K_n(\mathbf{x}, \mathbf{z})$ is nothing but the probability that $\mathbf{x}$ and $\mathbf{z}$ are connected (i.e., that they fall in the same cell) in a random tree. Therefore, the kernel K_n can be seen as a proximity measure between two points in the forest. Hence, any forest has its own metric K_n, but unfortunately the one associated with the CART-splitting strategy is strongly data-dependent and therefore complicated to derive analytically.
Kernels defined in equation (1.8) do not necessarily belong to the family of Nadaraya-Watson-type kernels (Nadaraya, 1964; Watson, 1964), which satisfy a translation-invariant homogeneous property of the form $K_h(\mathbf{x}, \mathbf{z}) = \frac{1}{h} K((\mathbf{x} - \mathbf{z})/h)$ for some *smoothing parameter* $h > 0$. Therefore the analysis of estimates of the form (1.7) is, in general, more complicated than that of classic Nadaraya-Watson kernel estimates. Scornet (2016b) proved that the kernel of a centered forest of level k verifies

$$K_{n,k}^{cc}(\mathbf{x}, \mathbf{z}) = \sum_{\substack{k_1, \ldots, k_p \\ \sum_{j=1}^{p} k_j = k}} \frac{k!}{k_1! \ldots k_p!} \left(\frac{1}{d} \right)^k \prod_{j=1}^{d} \mathbb{1}_{\lceil 2^{k_j} x_j \rceil = \lceil 2^{k_j} z_j \rceil}. \tag{1.9}$$

As an illustration, Figure 1.1 shows the graphical representation of $K_{n,k}^{cc}$ for $k = 1$, 2 and 5. Let $\widehat{f}_{\infty,n,k}^{cc}$ be the kernel estimate in (1.7) in which the kernel K_n has been replaced by the centered kernel $K_{n,k}^{cc}$. Scornet (2016b) proves the consistency of this estimate providing $k \to \infty$ and $n/2^k \to 0$ and states an upper bound for its risk, which unfortunately does not match the minimax rate for Lipschitz functions, as the centered forest estimate (see equation (1.6)). Another attempt in the same direction is by Arlot et al. (2014), who show that the bias of other purely random forests are close (or even equal) to that of kernel estimates. They use this link to derive rates of convergence for the bias of several purely random forests.

1.3 Quantile forests

Benefits from using random forests instead of a single tree have not been highlighted by the analysis of purely random forests. In this section, by studying quantile forests, we show that aggregating inconsistent quantile trees can result in a consistent quantile forests. Besides, our analysis provides guidance about how to choose tree depth and subsample size for these forests.

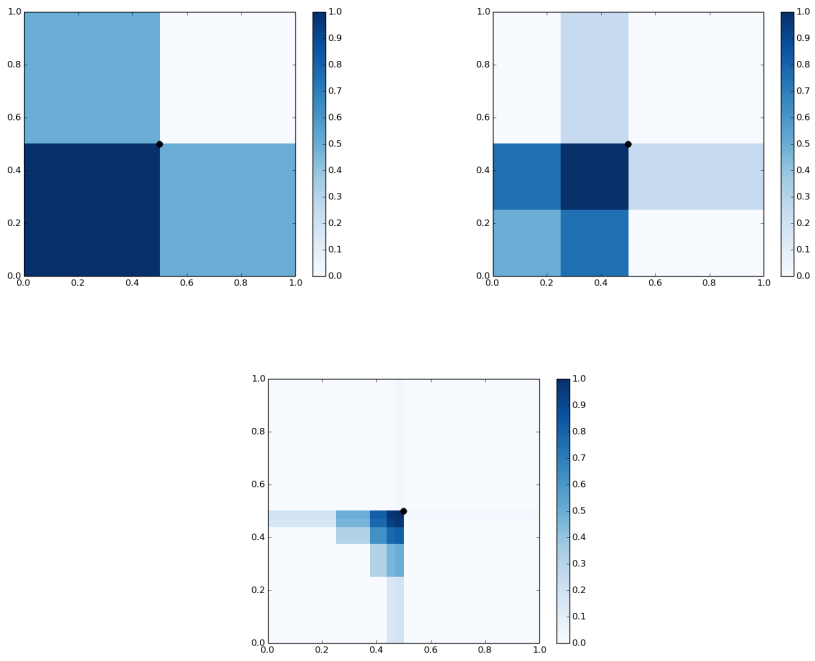


Figure 1.1: Graphs of $g_k(\mathbf{z}) = K_{n,k}^{cc}((0.49, 0.49), \mathbf{z})$ on $[0,1]^2$ for $k = 1, 2, 5$.

1.3.1 Construction of quantile forests

In α-quantile ($\alpha \in (0, 1/2]$) random forests, before growing each tree, data are subsampled, that is a_n points ($a_n < n$) are selected without replacement. In each cell, a coordinate is chosen uniformly at random among the d coordinates and a value $\alpha' \in [\alpha, 1-\alpha]$ is selected (possibly depending on the X_i in the cell, but independent of the Y_i). Then, in each cell, a split is performed at the empirical α' quantile along the chosen coordinate. Finally, the splitting procedure in a cell stops when the cell has been cut exactly k_n times or when the cell contains exactly one point. In detail, the construction works as follows:

1. Grow M trees as follows:

   (a) Prior to the j-th tree construction, select uniformly without replacement, a_n data points among the data set $\mathcal{D}_n$. Only these a_n observations are used in the tree construction.

   (b) Consider the cell $[0,1]^d$.

   (c) Select uniformly one coordinate j among $\{1, \ldots, d\}$ without replacement.

   (d) Cut the cell at an empirical quantile $\alpha' \in [\alpha, 1-\alpha]$ of the $X_i^{(j)}$ ($1 \leqslant i \leqslant n$) falling into the cell, along the preselected direction j. The observation on which the split is performed is not sent down the tree.

(e) For each of the two resulting cells, repeat $(c)-(d)$ if the cell contains more than one point, and if the cell has been cut strictly less than k_n times.

(f) For a query point $\mathbf{x}$, the j-th quantile tree outputs the average $\widehat{f}_n(\mathbf{x}, \Theta_j)$ of the Y_i falling into the same cell as $\mathbf{x}$.

2. For a query point $\mathbf{x}$, α-quantile forests output

$$\widehat{f}_{\alpha,M,a_n,k_n,n}(\mathbf{x}, \Theta_1, \ldots, \Theta_M) \tag{1.10}$$

which is the average of the predictions $\widehat{f}_n(\mathbf{x}, \Theta_j)$ given by the M trees.

Although we force all splits of α-quantile random forests to be performed in a pre-specified range (between the quantiles of order α and $1-\alpha$ in each cell), there is still a large variety of choices for the locations of the splits, which may depend on the X_i in the corresponding cell. We do not allow these splits to depend on the labels Y_i, which is a strong difference with Breiman's forests but vital in our theoretical analysis. Nevertheless, α-quantile random forests are a rather large class of random forests.
Another difference with Breiman's forests is the fact that subsampling is done without replacement in quantile forests. This is a common assumption in random forest theoretical analyses (Mentch et al., 2016; Athey et al., 2019), since studying bootstrap itself (the resampling procedure used in Breiman's forests, which samples uniformly n observations out of the n original ones with replacement) turns out to be difficult. Indeed, if subsampling were to be done with replacement, the distribution of each subsample would not be the same as the distribution of the original sample (if we assume, for example, that the input $\mathbf{X}$ has a density with respect to Lebesgue measure), and the mathematical analysis would be much more complicated.
Note that quantile trees are by no mean new and have already been studied (see, e.g., k-spacing tree, Devroye et al., 1996). Our analysis will emphasize the advantages of using an aggregation of such trees instead of a single one.

1.3.2 Consistency of quantile forests

In this section, we consider fully grown quantile forests, in which the tree depth is pre-specified (i.e. $k_n = \infty$) so that each leaf of each tree contains exactly one point. Following the notation of equation (1.10), we let $\widehat{f}_{\alpha,\infty,a_n,\infty,n}$ be the corresponding infinite forest estimate. Indeed, since Theorem 1.1 allows us to extend consistency and rate of consistency from infinite forests to finite forests, our analysis focuses on infinite forests.
Theorem 1.2 (Scornet, 2016a) states the consistency of all infinite α-quantile forests, for any $\alpha \in (0, 0.5]$, provided a proper subsampling rate.

Assumption 1.2. *One has $Y = f^\star(\mathbf{X}) + \varepsilon$, where ε is a centered noise such that $\mathbb{V}[\varepsilon|\mathbf{X} = \mathbf{x}] \leqslant \sigma^2$, where $\sigma^2 < \infty$ is a constant. Moreover, $\mathbf{X}$ has a density on $[0,1]^d$ and $f^\star$ is continuous.*

Theorem 1.2. *Grant Assumption 1.2. Let $\alpha \in (0, 1/2]$. Then, provided $a_n \to \infty$ and $a_n/n \to 0$, infinite α-quantile random forests are consistent, that is*

$$\lim_{n\to\infty} R(\widehat{f}_{\alpha,\infty,a_n,\infty,n}) = 0.$$

Additionally, if we assume that $\mathbb{E}[\varepsilon|\mathbf{X}] = 0$ and $\mathbb{V}[\varepsilon|\mathbf{X}] > 0$, one can show that each tree in α-quantile forests is inconsistent (because each leaf contains exactly one data point). Thus, with these additional assumptions, Theorem 1.2 shows that α-quantile forests combine inconsistent trees to form a consistent estimate. The fact that randomization can turn inconsistent predictors into consistent ones has already been noticed for a subsampled 1-nearest neighbor rule by Biau, Devroye, and Lugosi (2008), Biau and Devroye (2010), and Biau, Cérou, et al. (2010) but is a new result for random forests composed of fully grown trees (containing one observation per leaf).

Theorem 1.2 does not cover the bootstrap case since in that case, $a_n/n = 1$. Indeed, subsampling ($a_n < n$) plays a crucial role in our analysis by ensuring that, for all $\mathbf{x}$, the probabilities of connection $K_n(\mathbf{x}, \mathbf{X}_i)$ (see equation (1.8)) tend uniformly to zero. More precisely, if a data point is not selected by subsampling, it is therefore not connected to any point $\mathbf{x}$. One can show that the probabilities $K_n(\mathbf{x}, \mathbf{X}_i)$ are upper bounded by a_n/n, which explains the asymptotic regime for a_n in Theorem 1.2. Moreover, the condition $a_n/n \to 0$ implies that a point $\mathbf{x}$ should not fall into the same cell as a given data point in a too large proportion of trees in the forests, which implies that trees must be different enough. The idea of tree diversity has already been suggested by Breiman (2001a), who proves that the forest error is small as soon as the predictive power of each tree is good and the correlations between the tree errors are low.

Quantile forests offer a lot of liberty for choosing split locations, even if these locations cannot depend on the labels. Besides, each leaf of each tree can contain a very small number of observations, which is close to Breiman's forests default setting. Quantile forests are then a good modelling of Breiman's forests, as soon as splits are not performed too close to the edges of the cells and when the problem is not sparse. Indeed, in this latter case, Breiman's forests benefit from the splitting strategy which uses the labels to identify relevant variables. This fact will be detailed in Section 1.5.

1.3.3 Rate of consistency of quantile forests

We now consider α-quantile random forests, in which the depth k_n and the subsample size a_n satisfy $a_n \alpha^{k_n} \geqslant 4$, so that resulting trees are almost balanced. Since we do not set $k_n = \infty$ anymore, we can obtain quantile trees that are not fully grown (i.e., whose construction has been stopped prematurely). Theorem 1.3 (Duroux et al., 2018) gives an upper bound on the rate of consistency of all α-quantile forests.

Theorem 1.3. *Grant Assumption 1.2. Assume that* $\mathbf{X}$ *is uniformly distributed on* $[0,1]^d$ *and* $f^\star$ *is L-Lipschitz. Then, for all* n*, for all* $\mathbf{x} \in [0,1]^d$*,*

$$\mathbb{E}\big[\widehat{f}_{\alpha,\infty,a_n,k_n,n}(\mathbf{x}) - f^\star(\mathbf{x})\big]^2 \leqslant \frac{2\sigma^2}{n\alpha^k} + dL^2 C_1 \left(1 - \frac{1}{d} + \frac{(1-\alpha)^2}{d}\right)^k, \tag{1.11}$$

where $C_1 = e^{-\frac{\alpha}{d-1+(1-\alpha)^2}}$*. In addition, let* $\beta = 1 - \frac{1}{d} + \frac{(1-\alpha)^2}{d}$*. The right-hand side is minimal for*

$$k_n = -\frac{1}{\log(\alpha) + \log\beta}\Big[\log(n) + C_2\Big], \tag{1.12}$$

where $C_2 = \log\left(\frac{dL^2C_1\log(\beta)}{2\sigma^2\log(\alpha)}\right)$, *under the condition that, for some constant* $C_3 > 0$,

$$a_n \geqslant C_3 n^{\frac{\log\alpha}{\log\alpha+\log\beta}} \tag{1.13}$$

Consequently, there exists a constant $C_4 > 0$ *such that, for all* (k_n, a_n) *satisfying* (1.12) *and* (1.13),

$$\mathbb{E}\big[\widehat{f}_{\alpha,\infty,a_n,k_n,n}(\mathbf{x}) - f^\star(\mathbf{x})\big]^2 \leqslant C_4 n^{-\frac{\log\beta}{\log\alpha+\log\beta}}. \tag{1.14}$$

Equation (1.11) stems from the estimation/approximation error decomposition of quantile forests, which is in line with the existing results for other types of forests (see Biau, 2012; Arlot et al., 2014). Note that, assuming that $a_n/n \to 0$ and $k_n \to \infty$ with the constraint $a_n\alpha^{k_n} \geqslant 4$ implies that the right-hand side of (1.11) tends to zero, which allows us to retrieve the conclusion of Theorem 1.2.

Note that the estimation error of a single quantile tree grown with a_n observations is of order $1/(a_n\alpha^k)$. Here the estimation error of quantile forests (first term in the right-hand side of (1.11)) is of order $1/(n\alpha^k)$. Therefore, the estimation error of a quantile forest is lower than that of a quantile tree because of subsampling ($a_n < n$). This is not surprising since the variance reduction of random forests is a well-known property (Genuer, 2012).

The right-hand side of (1.11) is minimal for $\alpha = 1/2$ which corresponds to the median forest. For this forest, splits must be close to the center of the cells, at least for the splits close to the root of the tree, since $\mathbf{X}$ is uniformly distributed on $[0,1]^d$. Surprisingly, the rate of consistency of median forests, as stated in Theorem 1.3, is faster than that of centered forest established in Biau (2012), which is equal to $n^{-3/(4d\log 2+3)}$. This latter rate turns out to be suboptimal for $d = 1$ whereas our bound leads to the minimax rate $n^{-2/3}$ for Lipschitz functions (Stone, 1980; Stone, 1982). This was to be expected since, in dimension one, centered trees reduce to cubic partitions, which are known to reach minimax rate for Lipschitz functions (Györfi et al., 2002). Note that Klusowski (2018) improves the bound in Biau (2012) so that it matches the minimax rate for $d = 1$.

Theorem 1.3 provides the pointwise rate of consistency of $\widehat{f}_{\alpha,\infty,a_n,k_n,n}(\mathbf{x})$. Thus, quantile forests are pointwise consistent, which may not be the case of Breiman forests at some very particular query point $\mathbf{x}$ close to the edges of $[0,1]^d$ (see Figure 3 and the discussion below in Wager, 2014). Let us now give two specific parametrization of quantile forests that satisfy the bound (1.14).

Corollary 1.1. *Let* $\beta = 1 - \frac{1}{d} + \frac{(1-\alpha)^2}{d}$. *Consider either*

(*i*) *A quantile forest without subsampling (i.e.,* $a_n = n$*) and such that the parameter* k_n *satisfies (1.12).*

(*ii*) *A fully grown quantile forest, that is a forest whose parameters* k_n *and* a_n *satisfy* $\lfloor a_n\alpha^{k_n}\rfloor = 4$ *and (1.12).*

In both cases, under the same assumptions as in Theorem 1.3, we have, for all n, *for all* $\mathbf{x} \in [0,1]^d$,

$$\mathbb{E}\big[\widehat{f}_{\alpha,\infty,a_n,k_n,n}(\mathbf{x}) - f^\star(\mathbf{x})\big]^2 \leqslant C_4 n^{-\frac{\log\beta}{\log\alpha+\log\beta}}.$$

Whereas each individual tree in fully grown quantile forests (setting (ii) in Corollary 1.1) is inconsistent (see the discussion following Theorem 1.2), fully grown quantile forests are consistent and their rate of consistency is provided by Corollary 1.1. Inequality (1.13) and Corollary 1.1 give the optimal subsampling size for fully grown median forests ($\alpha = 1/2$) and the corresponding rates of consistency for median forests, which is the natural extension of Theorem 1.2. Conversely, one can consider a quantile forest without subsampling, whose tree depth k_n is properly chosen (setting (i) in Corollary 1.1). Such quantile forests (for which randomization comes from the random choice of the splitting direction in each node) also satisfy the upper bound (1.14).
The extended experiments in Duroux et al. (2018) study the influence of depth and subsampling on Breiman's forests. In particular, we show that Breiman's forests can be outperformed by a proper tuning of the subsampling or the tree depth. Noteworthy, tuning tree depth can be done at almost no additional computational cost due to the intrinsic recursive nature of forests. However, for a fixed subsample size, subsampled Breiman's forests are faster to train since they require less computation to find the best split in each node (due to a smaller number of observations).
As a by-product, our analysis also shows that there is no particular interest in bootstrapping data instead of subsampling: in our experiments (see Duroux et al., 2018), bootstrap is comparable (or worse) than a proper tuning of subsample size. This sheds some light on several previous theoretical analyses where the bootstrap step was replaced by subsampling, which is more amenable to analysis. Similarly, proving theoretical results on fully grown Breiman's forests turned out to be extremely difficult. Our analysis shows that there is no theoretical foundation for considering default parameters in Breiman's forests, which gives more credit to previous theoretical results focusing on Breiman's forests whose trees are small and/or for which subsampling is used instead of bootstrap.

1.4 Mondrian forests

Quite surprisingly, optimal rates for purely random forests are only obtained in the one-dimensional case, in which decision trees reduce to histograms. In the multi-dimensional setting, for which trees exhibit an intricate recursive structure, only suboptimal rates are derived. As shown by lower bounds from Klusowski (2018), this is not merely a limitation from the analysis: centered forests exhibit suboptimal rates for Lipschitz regression functions. It is likely that quantile forests presented in Section 1.3 suffer from the same flaw.
Of particular interest in this section is the *Mondrian Forest* (MF) algorithm, an efficient and accurate online random forest predictor (see Lakshminarayanan et al., 2014; Lakshminarayanan et al., 2016) based on the Mondrian process (Roy and Yee, 2009; Roy, 2011; Orbanz et al., 2014), a natural probability distribution on the set of recursive partitions of the unit hypercube $[0, 1]^d$.
Mondrian forests depend on a lifetime parameter $\lambda > 0$ that guides the complexity of the trees by stopping their building process. We consider in this section a batch version of Mondrian forests and propose a theoretical analysis of this algorithm. After describing the Mondrian forest algorithm, we prove its consistency provided the lifetime λ_n satisfies some asymptotic condition. We provide convergence rates which are minimax optimal for s-Hölder regression

functions with $s \in (0, 2]$. In particular, we illustrate the fact that Mondrian forests improve over Mondrian trees in terms of rate of consistency, when $s \in (1, 2]$.

1.4.1 The Mondrian forest algorithm

Trees and Forests Mondrian forests are a particular type of purely random forests, whose construction is independent of $\mathbf{X}_i$. There is no resampling step in Mondrian forests, so that the randomization is reduced to the randomized partition. To highlight this property, we will denote by Π a (random) Mondrian partition (whose construction is described below) so that the Mondrian tree estimate takes the form

$$\widehat{f}_n(\mathbf{x}, \Pi) = \sum_{i=1}^{n} \frac{\mathbb{1}_{\mathbf{X}_i \in C_\Pi(\mathbf{x})}}{N_n(C_\Pi(\mathbf{x}))} Y_i, \tag{1.15}$$

where $C_\Pi(\mathbf{x})$ is the cell of the tree partition Π containing $\mathbf{x}$ and $N_n(C_\Pi(\mathbf{x}))$ is the number of observations falling into $C_\Pi(\mathbf{x})$ (with, as usual, the convention $0/0 = 0$). Let $\Pi_M = (\Pi^{(1)}, \ldots, \Pi^{(M)})$, where $\Pi^{(m)}$ (for $m = 1, \ldots, M$) are i.i.d. random partitions of $[0, 1]^d$, distributed as Π. The Mondrian forest estimate is thus defined as

$$\widehat{f}_{M,n}(\mathbf{x}, \Pi_M) = \frac{1}{M} \sum_{m=1}^{M} \widehat{f}_n(\mathbf{x}, \Pi^{(m)}) \,. \tag{1.16}$$

Given a rectangular box $C = \prod_{j=1}^{d} [a_j, b_j] \subseteq \mathbb{R}^d$, we let $|C| := \sum_{j=1}^{d} (b_j - a_j)$. The Mondrian process $\mathsf{MP}(C)$ is a distribution on (infinite) tree partitions of C introduced by Roy and Yee (2009). The Mondrian process distribution $\mathsf{MP}(\lambda, C)$ is a distribution on tree partitions of C, resulting from the pruning of partitions drawn from $\mathsf{MP}(C)$. The pruning is done by removing all splits occurring after time $\lambda > 0$. In this perspective, λ is called the lifetime parameter and controls the complexity of the partition: large values of λ corresponds to deep trees (complex partitions).

Sampling from the distribution $\mathsf{MP}(\lambda, C)$ can be done efficiently by applying the recursive procedure $\texttt{SampleMondrian}(C, \tau = 0, \lambda)$ described in Algorithm 2. Figure 1.2 below shows a particular instance of Mondrian partition on a square box, with lifetime parameter $\lambda = 3.4$. In what follows, $\mathsf{Exp}(\lambda)$ stands for the exponential distribution with intensity $\lambda > 0$.

The Mondrian forest grows randomized tree partitions $\Pi_\lambda^{(1)}, \ldots, \Pi_\lambda^{(M)}$ according to the distribution $\mathsf{MP}(\lambda, [0, 1]^d)$, fits each one with the dataset $\mathcal{D}_n$ by averaging the labels falling into each leaf, then combines the resulting Mondrian tree estimates by averaging their predictions. In accordance with Equation (1.16), we let

$$\widehat{f}_{\lambda,M,n}(\mathbf{x}, \Pi_{\lambda,M}) = \frac{1}{M} \sum_{m=1}^{M} \widehat{f}_{\lambda,n}^{(m)}(\mathbf{x}, \Pi_\lambda^{(m)}) \tag{1.17}$$

be the Mondrian forest estimate described above, where $\widehat{f}_{\lambda,n}^{(m)}(\mathbf{x}, \Pi_\lambda^{(m)})$ denotes the Mondrian tree estimate, based on the random partition $\Pi_\lambda^{(m)}$ and $\Pi_{\lambda,M} = (\Pi_\lambda^{(1)}, \ldots, \Pi_\lambda^{(M)})$. To ease notation, we will write $\widehat{f}_{\lambda,n}^{(m)}(\mathbf{x})$ instead of $\widehat{f}_{\lambda,n}^{(m)}(\mathbf{x}, \Pi_\lambda^{(m)})$.

Algorithm 2 `SampleMondrian`(C, τ, λ): samples a Mondrian partition of C, starting from time τ and until time λ.

Inputs: A cell $C = \prod_{1 \leqslant j \leqslant d} [a_j, b_j]$, starting time τ and lifetime parameter λ.
Sample a random variable $E_C \sim \mathsf{Exp}(|C|)$
if $\tau + E_C \leqslant \lambda$ **then**
    Sample a split dimension $J \in \{1, \ldots, d\}$, with $\mathbb{P}(J = j) = (b_j - a_j)/|C|$
    Sample a split threshold S_J uniformly in $[a_J, b_J]$
    Split C along the split (J, S_J): let $C_0 = \{x \in C : x_J \leqslant S_J\}$ and $C_1 = C \setminus C_0$
    return `SampleMondrian`$(C_0, \tau + E_C, \lambda) \cup$ `SampleMondrian`$(C_1, \tau + E_C, \lambda)$
else
    return $\{C\}$ (*i.e.*, do not split C).
end if

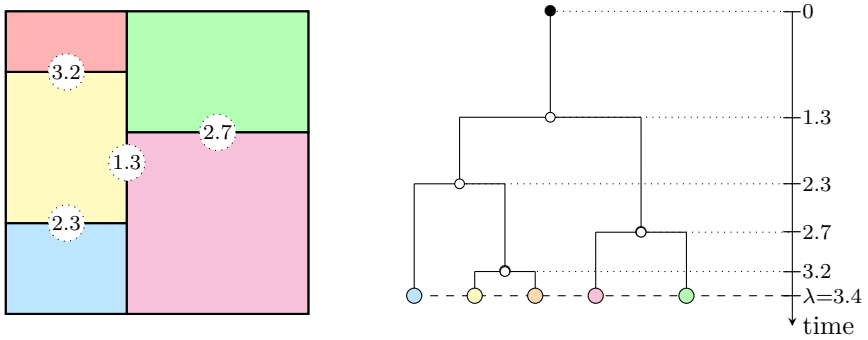


Figure 1.2: A Mondrian partition (left) with corresponding tree structure (right), which shows the evolution of the tree over time. The split times are indicated on the vertical axis, while the splits are denoted with bullets ($\circ$).

Although we use the standard definition of Mondrian processes, the way we compute the prediction in a Mondrian tree differs from the original one. Indeed, in Lakshminarayanan et al. (2014), prediction is given by the expectation over a posterior distribution, where a hierarchical prior is assumed on the label distribution of each cell of the tree. In this paper, we simply compute the average of the observations falling into a given cell.

1.4.2 Properties of the Mondrian process and consistency

Now, we show that the properties of the Mondrian process enable us to compute explicitly some local and global quantities related to the structure of Mondrian partitions. To do so, we will need the following fact, exposed by Roy and Yee (2009).

Fact 1 (Dimension 1)**.** *For $d = 1$, the splits from a Mondrian process $\Pi_\lambda \sim \mathsf{MP}(\lambda, [0, 1])$ form a subset of $[0, 1]$, which is distributed as a Poisson point process of intensity $\lambda \mathrm{d}x$.*

Proposition 1.1 below (see Mourtada, Gaïffas, et al., 2020) is a sharp result giving the exact distribution of a cell of a Mondrian partition. Such a characterization is typically unavailable

for other randomized trees partitions involving a complex recursive structure.

Proposition 1.1 (Cell distribution). *Let* $\mathbf{x} \in [0,1]^d$ *and denote by*

$$C_\lambda(\mathbf{x}) = \prod_{1 \leqslant j \leqslant d} [L_{j,\lambda}(\mathbf{x}), R_{j,\lambda}(\mathbf{x})]$$

the cell containing $\mathbf{x}$ *in a partition* $\Pi_\lambda \sim \mathsf{MP}(\lambda, [0,1]^d)$ *(this cell corresponds to a leaf). Then, the distribution of* $C_\lambda(\mathbf{x})$ *is characterized by the following properties*:

(*i*) $L_{1,\lambda}(\mathbf{x}), R_{1,\lambda}(\mathbf{x}), \ldots, L_{d,\lambda}(\mathbf{x}), R_{d,\lambda}(\mathbf{x})$ *are independent*;

(*ii*) *For each* $j = 1, \ldots, d$, $L_{j,\lambda}(\mathbf{x})$ *is distributed as* $(x_j - \lambda^{-1} E_{j,L}) \vee 0$ *and* $R_{j,\lambda}(\mathbf{x})$ *as* $(x_j + \lambda^{-1} E_{j,R}) \wedge 1$, *where* $E_{j,L}, E_{j,R} \sim \mathsf{Exp}(1)$.

Thanks to Proposition 1.1, we are able to state Corollary 1.2, which gives a precise upper bound on the diameter of the cells.

Corollary 1.2 (Cell diameter). *Set* $\lambda > 0$ *and* $\Pi_\lambda \sim \mathsf{MP}(\lambda, [0,1]^d)$ *be a Mondrian partition. Let* $\mathbf{x} \in [0,1]^d$ *and let* $D_\lambda(\mathbf{x})$ *be the* ℓ^2*-diameter of the cell* $C_\lambda(\mathbf{x})$ *containing* $\mathbf{x}$ *in* Π_λ. *For every* $\delta > 0$, *we have*

$$\mathbb{P}(D_\lambda(\mathbf{x}) \geqslant \delta) \leqslant d\Big(1 + \frac{\lambda\delta}{\sqrt{d}}\Big) \exp\Big(-\frac{\lambda\delta}{\sqrt{d}}\Big) \tag{1.18}$$

and

$$\mathbb{E}[D_\lambda(\mathbf{x})^2] \leqslant \frac{4d}{\lambda^2}\,. \tag{1.19}$$

In order to control the risk of Mondrian trees and forests, we also need an upper bound on the number of cells in a Mondrian partition. Quite surprisingly, we are able to compute the expectation of this quantity, as shown in Proposition 1.2 (see Mourtada, Gaïffas, et al., 2020, for details).

Proposition 1.2 (Number of cells). *Set* $\lambda > 0$ *and* $\Pi_\lambda \sim \mathsf{MP}(\lambda, [0,1]^d)$ *be a Mondrian partition. If* K_λ *denotes the number of cells in* Π_λ, *we have* $\mathbb{E}[K_\lambda] = (1+\lambda)^d$.

Although the proof is technically involved, it relies on a natural coupling argument: we introduce a recursive modification of the construction of the Mondrian process which keeps the expected number of leaves unchanged, and for which this quantity can be computed directly using the Mondrian-Poisson equivalence in dimension one (Fact 1). A much simpler result is $\mathbb{E}[K_\lambda] \leqslant (e(1+\lambda))^d$, which was previously obtained in Mourtada et al. (2017). By contrast, Proposition 1.2 provides the *exact* value of this expectation, which removes a superfluous e^d factor.

As illustrated in this section, a remarkable fact with the Mondrian forest is that the quantities of interest for the statistical analysis of the algorithm can be made explicit. In particular, we have seen in this section that, roughly speaking, a Mondrian partition is balanced enough so that it contains $O(\lambda^d)$ cells of diameter $O(1/\lambda)$, which is the minimal number of cells to cover $[0,1]^d$.

Based on Corollary 1.2 and Proposition 1.2, the consistency of Mondrian tree and Mondrian forests for any number of trees is proved provided $\lambda_n \to \infty$ and $\lambda_n^d / n \to 0$ (Mourtada, Gaïffas, et al., 2020). A more precise tuning of λ_n is provided hereafter.

1.4.3 Mondrian trees and forests are minimax for s-Hölder functions with $s \in (0, 1]$

The bounds obtained in Corollary 1.2 and Proposition 1.2 are explicit and sharp in their dependency on λ. Based on these properties, we now establish a theoretical upper bound on the risk of Mondrian trees, which turns out to be minimax optimal over the class of s-Hölder functions for $s \in (0, 1]$ (see for instance Stone, 1982, Chapter I.3 in Nemirovski, 2000 or Theorem 3.2 in Györfi et al., 2002).

Assumption 1.3. *Assume $Y = f^\star(\mathbf{X}) + \varepsilon$, where $\mathbb{E}[\varepsilon \,|\, \mathbf{X}] = 0$ and $\mathrm{Var}(\varepsilon \,|\, \mathbf{X}) \leqslant \sigma^2 < \infty$ almost surely.*

Our minimax results hold for a class of s-Hölder regression functions defined below.

Definition 1. *Let $p \in \mathbb{N}$, $\beta \in (0, 1]$ and $L > 0$. The (p, β)-Hölder ball of norm L, denoted $\mathscr{C}^{p,\beta}(L) = \mathscr{C}^{p,\beta}([0, 1]^d, L)$, is the set of p times differentiable functions $f : [0, 1]^d \to \mathbb{R}$ such that*

$$\|\nabla^p f(\mathbf{x}) - \nabla^p f(\mathbf{x}')\| \leqslant L\|\mathbf{x} - \mathbf{x}'\|^\beta \quad \text{and} \quad \|\nabla^k f(\mathbf{x})\| \leqslant L$$

for every $\mathbf{x}, \mathbf{x}' \in [0, 1]^d$ and $k \in \{1, \dots, p\}$. Whenever $f \in \mathscr{C}^{p,\beta}(L)$, we say that f is s-Hölder with $s = p + \beta$.

Theorem 1.4. *Grant Assumption 1.3 and assume that $f^\star \in \mathscr{C}^{0,\beta}(L)$, where $\beta \in (0, 1]$ and $L > 0$. Let $M \geqslant 1$. The quadratic risk of the Mondrian forest $\widehat{f}_{\lambda,M,n}$ with the choice $\lambda := \lambda_n \asymp L^{2/(d+2\beta)} n^{1/(d+2\beta)}$ satisfies*

$$\mathbb{E}[(\widehat{f}_{\lambda_n,M,n}(\mathbf{X}) - f^\star(\mathbf{X}))^2] = O(L^{2d/(d+2\beta)} n^{-2\beta/(d+2\beta)}), \tag{1.20}$$

which corresponds to the minimax rate over the class $\mathscr{C}^{0,\beta}(L)$.

Theorem 1.4 is one of the first results to prove that a purely random forest can be minimax optimal *in arbitrary dimension*. Most previous works (Genuer, 2012; Duroux et al., 2018; Klusowski, 2018) establish minimax optimal upper bounds for $d = 1$. In this one-dimensional case, tree partitions reduce to partitions of $[0, 1]$ in intervals, and do not possess the recursive structure that appears in higher dimensions and makes their analysis challenging. The only other result about minimax optimality in arbitrary dimension we are aware of is a private communication by Arlot et al. (2014) who establish it for a toy random forest, which does not possess a recursive structure even in dimensions larger than one. One notable specificity of Mondrian forests compared to other purely random forest variants is that the largest cells are more likely to be split, and once a cell is selected, its largest sides are also more likely to be split. This splitting strategy leads to cells with small diameters.

1.4.4 Improved rates for Mondrian forests compared to a Mondrian tree

The convergence rate stated in Theorem 1.4 for $f^\star \in \mathscr{C}^{0,\beta}(L)$ is valid for both trees and forests, and the risk bound does not depend on the number M of trees that compose the forest. In practice, however, forests exhibit much better performances than individual trees (Fernández-Delgado et al., 2014).

Here, we illustrate the benefits of forests over trees by assuming that $f^\star \in \mathscr{C}^{1,\beta}(L)$. It can be shown (see Proposition 3 in Mourtada, Gaiffas, et al., 2020) that single Mondrian trees do not benefit from this additional smoothness assumption, and achieve the same rate as in the Lipschitz case. This comes from the fact that the bias of trees is highly sub-optimal for such functions. It turns out that large enough Mondrian forests, which average Mondrian trees, are minimax optimal over $\mathscr{C}^{1,\beta}$.

Theorem 1.5. *Grant Assumption 1.3 and assume that $f^\star \in \mathscr{C}^{1,\beta}(L)$, with $\beta \in (0,1]$ and $L > 0$. In addition, assume that $\mathbf{X}$ has a positive and C_p-Lipschitz density p w.r.t the Lebesgue measure on $[0,1]^d$. Let $\widehat{f}_{\lambda,M,n}$ be the Mondrian forest estimate given by (1.17). Set $\varepsilon \in (0,1/2)$ and $B_\varepsilon = [\varepsilon, 1-\varepsilon]^d$. Then, we have*

$$\mathbb{E}\big[(\widehat{f}_{\lambda,M,n}(\mathbf{X}) - f^\star(\mathbf{X}))^2 | \mathbf{X} \in B_\varepsilon\big] \leqslant \frac{2(1+\lambda)^d}{n}\frac{2\sigma^2 + 9\|f\|_\infty^2}{p_0(1-2\varepsilon)^d} + \frac{144L^2 d p_1}{p_0(1-2\varepsilon)^d}\frac{e^{-\lambda\varepsilon}}{\lambda^3} + \frac{72L^2d^3}{\lambda^4}\Big(\frac{p_1 C_p}{p_0^2}\Big)^2 + \frac{16L^2 d^{1+\beta}}{\lambda^{2(1+\beta)}}\Big(\frac{p_1}{p_0}\Big)^2 + \frac{8dL^2}{M\lambda^2}, \tag{1.21}$$

where $p_0 = \inf_{\mathbf{x}\in[0,1]^d} p(\mathbf{x})$ and $p_1 = \sup_{\mathbf{x}\in[0,1]^d} p(\mathbf{x})$. In particular, letting $s = 1+\beta$, the choices

$$\lambda_n \asymp L^{2/(d+2s)} n^{1/(d+2s)} \quad \text{and} \quad M_n \gtrsim L^{4\beta/(d+2s)} n^{2\beta/(d+2s)}$$

give

$$\mathbb{E}\big[(\widehat{f}_{\lambda_n,M_n,n}(\mathbf{X}) - f^\star(\mathbf{X}))^2 \,|\, X \in B_\varepsilon\big] = O(L^{2d/(d+2s)} n^{-2s/(d+2s)}), \tag{1.22}$$

which corresponds to the minimax risk over the class $\mathscr{C}^{1,\beta}(L)$.

The proof of Theorem 1.5 relies on an improved control of the bias, compared to the one used in Theorem 1.4 in the Lipschitz case: it exploits the knowledge of the cell distribution given in Proposition 1.1 instead of merely the cell diameter given in Corollary 1.2 (which was enough for Theorem 1.4). The improved rate for Mondrian forests compared to Mondrian trees comes from the fact that large enough forests have a smaller bias than single trees for smooth regression functions. This corresponds to the fact that averaging randomized trees tends to smooth the decision function of single trees, which are discontinuous piecewise constant functions that approximate smooth functions sub-optimally. Such an effect was already noticed by Arlot et al. (2014) for purely random forests.

Remark 1.1. *While Equation (1.22) gives the minimax rate for $\mathscr{C}^{1,1}$ functions, it suffers from an unavoidable standard artifact, namely a boundary effect which impacts local averaging estimates, such as kernel estimators (Wasserman, 2006; Arlot et al., 2014). It is however possible to set $\varepsilon = 0$ in (1.21), which leads to the sub-optimal rate stated in Theorem 3 in Mourtada, Gaiffas, et al. (2020).*

1.5 Consistency of Breiman's forests

So far we have focused on simplified (purely) versions of the original random forest algorithm, for which the tree construction is independent of the labels. Here, we study the original Breiman's

forest algorithm with two small differences: (i) $a_n < n$ points are selected without replacement to build each tree and (ii) trees are not fully grown, i.e. their construction is stopped when they have exactly t_n leaves. To obtain such a tree, we split the cells one by one starting with the one at the first level of the tree (the root), selecting the best split by optimizing the CART-splitting criterion (equation (1.4)) . Note that cells containing exactly one observation are not split. The resulting tree may be imbalanced but contains exactly t_n leaves by construction. We denote by $\widehat{f}_{\infty,a_n,t_n,n}$ the corresponding infinite forest estimate.
To prove the consistency of Breiman's forests, with the two modifications described above, we consider the class of additive regression functions, popularized by Stone (1985) and Hastie and R. Tibshirani (1986). These models, which decompose the regression function as a sum of univariate functions, are flexible and easy to interpret, since they do not contain any interaction.

Assumption 1.4. *The regression model satisfies* $Y = \sum_{j=1}^{p} f_j^\star(\mathbf{X}^{(j)}) + \varepsilon$, *where* $\mathbf{X}$ *is uniformly distributed over* $[0,1]^d$, ε *is an independent centered Gaussian noise with finite variance* $\sigma^2 < \infty$, *and each component* $f_j^\star$ *is continuous.*

Proposition 1.3 below is vital to establish the consistency of Breiman's forests. It states that the variation of the regression function $f^\star$ within a cell of a CART tree is small provided n is large enough. To this aim, we define, for any cell A, the variation of $f^\star$ within A as $\Delta(f^\star, A) = \sup_{\mathbf{x},\mathbf{x}' \in A} |f^\star(\mathbf{x}) - f^\star(\mathbf{x}')|$. We also denote by $A_n(\mathbf{X}, \Theta)$ the cell of a tree built with random parameter Θ that contains $\mathbf{X}$.

Proposition 1.3. *Grant Assumption 1.4. Assume also that* $a_n, t_n \to \infty$. *Then, for all* $\rho, \xi > 0$, *there exists* $N \in \mathbb{N}^\star$ *such that, for all* $n > N$, $\mathbb{P}\left[\Delta(f^\star, A_n(\mathbf{X}, \Theta)) \leqslant \xi\right] \geqslant 1 - \rho$.

In most standard analyses of purely random forests, the variance is controlled by proving that cell diameters tend to zero in probability. Instead of such a geometrical assumption, Proposition 1.3 ensures that the variation of $f^\star$ inside a cell is small, thereby forcing the approximation error of the forest to asymptotically approach zero. Additionally, a proper tuning of the number of leaves t_n provides guarantees on the estimation error. This leads to the following consistency result (see also Scornet et al., 2015a).

Theorem 1.6. *Grant Assumption 1.4. Then, provided* $a_n \to \infty$ *and* $t_n (\log a_n)^9 / a_n \to 0$, *Breiman's random forests described above are consistent, i.e.,*

$$\lim_{n \to \infty} \mathbb{E}\left[\widehat{f}_{\infty,a_n,t_n,n}(\mathbf{X}) - f^\star(\mathbf{X})\right]^2 = 0.$$

The proof of Theorem 1.6 is based on consistency results of data-dependent partitions developed by Nobel (1996). Up to our knowledge, apart from the fact that bootstrapping is replaced by subsampling and trees are not fully grown, Theorem 1.6 is the first consistency result for Breiman's forests. Indeed, most models studied so far are designed independently of the dataset and are, consequently, unrealistic representations of the true procedure.
Noteworthy, Theorem 1.6 still holds with $a_n = n$. Indeed, subsampling has no impact in our analysis since we control the estimation error via the number of leaves t_n. We note in passing

that an easy adaptation of Theorem 1.6 shows that CART algorithm is consistent under the same assumptions as those of Theorem 1.6.
The term $(\log a_n)^9$ originates from the Gaussian noise and allows to control the noise tail. In the easier situation in which the Gaussian noise is replaced by a bounded random variable, the term $(\log a_n)^9$ turns into $\log a_n$, a term which accounts for the complexity of the tree partition. It is also interesting to note that Theorem 1.6 cannot be extended to establish the pointwise consistency of random forests, that is, for almost all $\mathbf{x} \in [0,1]^d$,

$$\lim_{n\to\infty} \mathbb{E}\big[\widehat{f}_{\infty,a_n,t_n,n}(\mathbf{x}) - f^\star(\mathbf{x})\big]^2 = 0.$$

Fixing $\mathbf{x} \in [0,1]^d$, the difficulty lays in the fact that we do not have a control on the diameter of the cell $A_n(\mathbf{x},\Theta)$, whereas we do have a control on $\mathbb{E}[\mathrm{diam}(A_n(\mathbf{X},\Theta))]$. Indeed, as previously highlighted by Wager (2014), random forests can be inconsistent at some fixed point $\mathbf{x} \in [0,1]^d$, particularly near the edges, while being L^2-consistent.
Our study also sheds some interesting light on the behaviour of forests when the ambient dimension d is large but the true underlying dimension of the model S is small as stated in the following assumption.

Assumption 1.5. *Grant Assumption 1.4 where additionally, there exists $S < d$ such that $Y = \sum_{j=1}^{S} f_j^\star(\mathbf{X}^{(j)}) + \varepsilon$.*

In Assumption 1.5, $S < d$ represents the true, but unknown, dimension of the model. Thus, among the d original features, it is assumed that only the first (without loss of generality) S variables are informative and are not independent of Y.
Proposition 1.4 below shows that random forests nicely adapt to the sparsity setting of Assumption 1.5 since the first splits in a tree concentrate along the S informative variables. For all k, denote by $j_{1,n}(\mathbf{X}), \ldots, j_{k,n}(\mathbf{X})$ the first k variables used to construct the cell containing $\mathbf{X}$, with the convention that $j_{q,n}(\mathbf{X}) = \infty$ if the cell has been cut strictly less than q times.

Proposition 1.4. *Grant Assumption 1.5 and set* `mtry` $= d$. *Assume that there is no interval $[a,b]$ and no $j \in \{1,\ldots,S\}$ such that $f_j^\star$ is constant on $[a,b]$. For all $k \in \mathbb{N}^\star$ and $\xi \in (0,1]$, there exists N such that, with probability at least $1-\xi$, for all $n > N$, for all $1 \leqslant q \leqslant k$,*

$$j_{q,n}(\mathbf{X}) \in \{1,\ldots,S\}.$$

According to Proposition 1.4, CART algorithm selects splits mostly along informative variables. Therefore, Breiman's forests can detect and adapt to the true dimension of the problem, making them accurate predictive algorithms in sparse framework.

1.6 Perspectives

The analysis of quantile forests in Section 1.3 highlights the influence of subsample size and tree depth on the forest error. Besides, the impact of the number of trees on the forest performance is well-known (see Theorem 1.1) and concentration results of the finite forest around its expectation have also been established (Wager, Hastie, et al., 2014; Mentch et al., 2016). Therefore, the

only parameter whose role remains unclear is the number `mtry` of variables eligible for splitting. Obtaining upper bounds on the rate of consistency for simple random forest model, in which the parameter `mtry` appears, would provide theoretical guidance on the tuning of this parameter, which could be a starting point to understand its influence on Breiman's forests.

Concentration of the empirical CART splits around their theoretical counterpart has been established in Bühlmann et al. (2002) and Banerjee et al. (2007) for one split, with the assumption that there exists a unique best theoretical split. This assumption is difficult to satisfy even for simple theoretical models. In order to obtain the concentration of the empirical tree around its theoretical counterpart, one could try to extend the results in Banerjee et al. (2007) to multiple consecutive splits, without assuming for each one the unicity of the best theoretical split. This would be a first step to establish rate of consistency of Breiman's forests. This research topic is closely connected to the work of Wager and Walther (2015) who prove similar result for an algorithm very close to Breiman's forests. Apart from being the first step for establishing rate of consistency, such result could lead us to an estimation of the probability of splitting along relevant and irrelevant variables in a specific model. This would be a great improvement over the work of Biau (2012) who assumes to know the asymptotic regime of these probabilities. Ultimately, such a line of work could lead to upper bounds for Breiman's forests showing that their rate of consistency adapts to the sparsity of the problem, which would then be a natural extension of Proposition 1.4.

Chapter 2

Intepretability and random forests

State-of-the-art learning algorithms, such as random forests or neural networks, are often criticized for their "black-box" nature. This criticism essentially results from the high number of operations involved in their prediction mechanism, as it prevents to grasp how inputs are combined to generate predictions. Interpretability of machine learning algorithms is receiving an increasing amount of attention since the lack of transparency is a strong limitation for many applications, in particular those involving critical decisions. The analysis of production processes in the manufacturing industry typically falls into this category. Indeed, such processes involve complex physical and chemical phenomena that can often be successfully modeled by black-box learning algorithms. However, any modification of a production process has deep and long-term consequences, and therefore cannot simply result from a blind stochastic modelling.
In this chapter, we are interested in obtaining interpretable procedures. Two paradigms are studied here: designing a simple, stable and predictive procedure (Section 2.1) and opening black-box algorithms such as decision trees and random forests by studying the importance measures they output (Section 2.2).

2.1 Decision rules

In this section, we design a learning algorithm that builds upon random forests to create a small stable set of decision rules, which are linearly aggregated to form a predictor. This learning algorithm has predictive accuracy comparable to that of random forests, while being more interpretable. Results presented here are based on Bénard et al., 2020 (see also Bénard et al., 2019, for the equivalent procedure in classification).

2.1.1 Introduction

Although there is no agreement in the machine learning literature about a precise definition of interpretability (Rüping, 2006; Lipton, 2016; Doshi-Velez et al., 2017; Murdoch et al., 2019), it is yet possible to define simplicity, stability, and predictivity as minimum requirements for interpretable models (Bénard et al., 2019; Yu and Kumbier, 2019). Simplicity of the model structure can be assessed by the number of operations performed in the prediction mechanism (e.g., Rüping, 2006; Freitas, 2014; Letham, 2015; Letham et al., 2015; Lipton, 2016; Ribeiro

et al., 2016; Murdoch et al., 2019). In particular, Murdoch et al. (2019) introduce the notion of *simulatable models* when a human is able to reproduce the prediction process by hand. Secondly, Yu (2013) argues that "interpretability needs stability", as the conclusions of a statistical analysis have to be robust to small data perturbations to be meaningful. Instability is the symptom of a partial and arbitrary modelling of the data, also known as the *Rashomon effect* (Breiman, 2001b). Finally, as also explained in Breiman (2001b), if the decrease of predictive accuracy is significant compared to a state-of-the-art black-box algorithm, the interpretable model misses some patterns in the data and is therefore misleading.
Decision trees (Breiman et al., 1984) can model nonlinear patterns while having a simple structure. However, the structure of trees is highly sensitive to small data perturbation (Breiman, 2001b), which violates the stability principle and is thus a strong limitation to their practical use. Rule algorithms are another type of nonlinear methods with a simple structure, defined as a collection of elementary rules. An elementary rule is a set of constraints on input variables, which forms a hyperrectangle in the input space and on which the associated prediction is constant. As an example, such a rule typically takes the following simple form:

$$\textbf{If} \begin{cases} X^{(1)} < 1.12 \\ \& \; X^{(3)} \geqslant 0.74 \end{cases} \textbf{ then } \hat{Y} = 0.18 \textbf{ else } \hat{Y} = 4.1$$

A large number of rule algorithms have been developed, among which the most influential RuleFit (J.H. Friedman et al., 2008) and Node harvest (Meinshausen, 2010), which are among the few rule algorithms able to handle regression problems. Yet, despite their powerful predictive skills, these two methods tend to produce long, complex, and unstable list of rules (typically of the order of 50), which makes their interpretability questionable.
The purpose of this section is to propose a new stable rule algorithm for regression, SIRUS-R (**S**table and **I**nterpretable **RU**le **S**et for **R**egression), and therefore demonstrate that rule methods can address regression problems efficiently while producing a compact and stable list of rules. We present the SIRUS-R algorithm in Section 2.1.2. One of the main contributions of this work is the development of a software implementation of SIRUS-R, via the `R` package `sirus`, available at `https://gitlab.com/safrandrti/sirus`, based on `ranger`, a high-performance random forest implementation in `R` and `C++` (Wright and Ziegler, 2017). We illustrate, in Section 2.1.4, the efficiency of our procedure `sirus` through numerical experiments on real datasets. Section 2.1.5 is devoted to studying the theoretical properties of the method, with, in particular, a proof of its asymptotic stability.

2.1.2 SIRUS-R

Rule generation The **first step** of SIRUS-R is to grow a random forest with a large number M of trees based on the available sample $\mathcal{D}_n$. Two critical features of our approach to stabilize the forest structure are (i) to restrict node splits to the q-empirical quantiles of the marginals $X^{(1)}, \ldots, X^{(d)}$ (computed on the whole data set, with typically $q = 10$), and (ii) to grow shallow trees of depth 2. As the experiments will show, these modifications to Breiman's original algorithm are harmless for predictive accuracy. Next, for each tree, we extract the 6 nodes (2 internal nodes in the first level, 4 external nodes in the second level). Thus both internal

and external nodes are extracted from the trees of depth 2 of the random forest to generate a large collection of nodes, typically 10^4. Formally, a tree node is represented by a path $\mathscr{P}$ which describes how to reach the node from the root of the tree. In the sequel, we denote by Π the collection of all possible paths.

Path selection The **second step** of SIRUS-R is to select the relevant paths from this large collection. Despite the tree randomization in the forest construction, there are some redundancy in the extracted paths. Indeed, the paths with a high frequency of appearance represent strong and robust patterns in the data, and are therefore good candidates to be included in a compact, stable, and predictive ensemble. This occurrence frequency is denoted by $\hat{p}_{M,n}(\mathscr{P})$ for each possible path $\mathscr{P} \in \Pi$. Then a threshold $p_0 \in (0,1)$ is simply used to select the most frequent paths, that is

$$\hat{\mathscr{P}}_{M,n,p_0} = \{\mathscr{P} \in \Pi : \hat{p}_{M,n}(\mathscr{P}) > p_0\}.$$

Path set post-treatment The 6 paths extracted from a tree are dependent because the paths corresponding to external nodes are nothing but refinements of the paths corresponding to internal nodes. Consequently, to properly aggregate these paths in the following step, the **third step** of SIRUS-R filters $\hat{\mathscr{P}}_{M,n,p_0}$ with the following post-treatment procedure: if the node induced by the path $\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}$ is a linear combination of nodes associated with paths with a higher frequency of appearance, then $\mathscr{P}$ is simply removed from $\hat{\mathscr{P}}_{M,n,p_0}$.

Rule aggregation A rule $\hat{g}_{n,\mathscr{P}}$ associated with a path $\mathscr{P}$ is a piecewise constant estimate: if a query point $\mathbf{x}$ falls into the corresponding node/hyperrectangle $H_{\mathscr{P}} \subset [0,1]^d$, the rule returns the average of the Y_i's for the training points $\mathbf{X}_i$'s that belong to $H_{\mathscr{P}}$ (and symmetrically if $\mathbf{x} \in H_{\mathscr{P}}^c$). From the **third step**, we have thus obtained a small set of regression rules. In the **fourth step** of SIRUS-R, the selected rules are combined to form a single estimate of $f^\star(\mathbf{x})$, that is

$$\widehat{f}_{M,n,p_0}(\mathbf{x}) = \hat{\beta}_0 + \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{\beta}_{n,\mathscr{P}} \hat{g}_{n,\mathscr{P}}(\mathbf{x}), \tag{2.1}$$

where $\hat{\beta}_0$ and $\hat{\beta}_{n,\mathscr{P}}$ are solutions of a ridge regression problem. More precisely, denoting by $\hat{\boldsymbol{\beta}}_{n,p_0}$ the column vector whose components are the coefficients $\hat{\beta}_{n,\mathscr{P}}$ for $\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}$, letting $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ and $\boldsymbol{\Gamma}_{n,p_0}$ the matrix whose rows are the rule values $\hat{g}_{n,\mathscr{P}}(\mathbf{X}_i)$ for $i \in \{1, \ldots, n\}$, we have

$$(\hat{\boldsymbol{\beta}}_{n,p_0}, \hat{\beta}_0) = \underset{\boldsymbol{\beta} \geqslant 0, \beta_0 \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{n} ||\mathbf{Y} - \beta_0 \mathbf{1}_{\boldsymbol{n}} - \boldsymbol{\Gamma}_{n,p_0} \boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2,$$

where $\mathbf{1}_{\boldsymbol{n}} = (1, \ldots, 1)^T$, and $\lambda > 0$ is tuned by cross-validation.

Remark 2.1. *Since rules are correlated by construction, a simple optimal least-squares solution leads to negative values for some of the coefficients $\hat{\beta}_{n,\mathscr{P}}$. Such behavior drastically undermines the interpretability of the algorithm. This is why, as in Node harvest (Meinshausen, 2010), the constraint $\boldsymbol{\beta} \geqslant 0$ is added to ensure that all coefficients are non-negative. In this correlated*

setting, the constraint $\boldsymbol{\beta} \geqslant 0$ *enforces sparsity and then instability. For this reason a ridge penalty is added to the loss function to stabilize the estimate* $\hat{\boldsymbol{\beta}}_{n,p_0}$ *and mitigate sparsity.*

2.1.3 How to measure interpretability?

Despite the lack of a precise definition of interpretable models, there are three minimum requirements to be taken into account: simplicity, stability, and predictivity. These notions need to be formally defined and quantified to allow for comparisons between algorithms.

Simplicity Simplicity refers to the model complexity, in particular the number of operations involved in the prediction mechanism. In the case of rule algorithms, a measure of simplicity is naturally given by the number of rules.

Stability Intuitively, a rule algorithm is stable when two independent estimations based on two independent samples return similar lists of rules. Formally, let $\hat{\mathscr{P}}'_{M,n,p_0}$ be the list of rules output by SIRUS-R fit on an independent sample $\mathcal{D}'_n$. Then the proportion of rules shared by $\hat{\mathscr{P}}_{M,n,p_0}$ and $\hat{\mathscr{P}}'_{M,n,p_0}$ gives a stability measure. Such a metric is known as the Dice-Sorensen index, and is often used to assess variable selection procedures (Chao et al., 2006; Zucknick et al., 2008; Boulesteix and Slawski, 2009; Z. He et al., 2010; Alelyani et al., 2011). In our case, the Dice-Sorensen index is defined as

$$\hat{S}_{M,n,p_0} = \frac{2\big|\hat{\mathscr{P}}_{M,n,p_0} \cap \hat{\mathscr{P}}'_{M,n,p_0}\big|}{\big|\hat{\mathscr{P}}_{M,n,p_0}\big| + \big|\hat{\mathscr{P}}'_{M,n,p_0}\big|}.$$

However, in practice one rarely has access to an additional sample $\mathcal{D}'_n$. Therefore, to circumvent this problem, we use a 10-fold cross-validation to simulate data perturbation. The stability metric is thus empirically defined as the average proportion of rules shared by two models of two distinct folds of the cross-validation. A stability of 1 means that the exact same list of rules is selected over the 10 folds, whereas a stability of 0 means that all rules are distinct between any 2 folds.

Predictivity For regression problems, the proportion of unexplained variance is a natural measure of the prediction error. The estimation is performed by 10-fold cross-validation.

2.1.4 Experiments

Experiments are run over 9 diverse public datasets to demonstrate the improvement of SIRUS-R over state-of-the-art methods.

SIRUS-R rule set Our algorithm is illustrated on the "LA Ozone" dataset from Hastie, R. Tibshirani, and J. Friedman (2009), which records the level of atmospheric ozone concentration from eight daily meteorological measurements made in Los Angeles in 1976: wind speed ("wind"), humidity ("humidity"), temperature ("temp"), inversion base height ("ibh"), daggot pressure gradient ("dpg"), inversion base temperature ("ibt"), visibility ("vis"), and day of the year ("doy"). The response "Ozone" is the log of the daily maximum of ozone concentration.

Average Ozone = 12		**Intercept** = −7.8					
Frequency		**Rule**					**Weight**
0.29	**if**	temp < 65	**then**	Ozone = 7	**else**	Ozone = 19	0.12
0.17	**if**	ibt < 189	**then**	Ozone = 7	**else**	Ozone = 18	0.07
0.063	**if**	temp ⩾ 65 & vis < 150	**then**	Ozone = 20	**else**	Ozone = 7	0.31
0.061	**if**	vh < 5840	**then**	Ozone = 10	**else**	Ozone = 20	0.072
0.060	**if**	ibh < 2110	**then**	Ozone = 16	**else**	Ozone = 7	0.14
0.058	**if**	ibh < 2960	**then**	Ozone = 15	**else**	Ozone = 6	0.10
0.051	**if**	temp ⩾ 65 & ibh < 2110	**then**	Ozone = 21	**else**	Ozone = 8	0.16
0.048	**if**	vis < 150	**then**	Ozone = 14	**else**	Ozone = 7	0.18
0.043	**if**	temp < 65 & ibt < 120	**then**	Ozone = 5	**else**	Ozone = 15	0.15
0.040	**if**	temp < 70	**then**	Ozone = 8	**else**	Ozone = 20	0.14
0.039	**if**	ibt < 227	**then**	Ozone = 9	**else**	Ozone = 22	0.21

Table 2.1: SIRUS-R rule list for the "LA Ozone" dataset.

The list of rules output for this dataset is presented in Table 2.1. The column "Frequency" refers to $\hat{p}_{M,n}(\mathscr{P})$, the occurrence frequency of each rule in the forest, used for rule selection. It enables to grasp how weather conditions impact the ozone concentration. In particular, a temperature larger than 65°F or a high inversion base temperature results in high ozone concentrations. The third rule tells us that the interaction of a high temperature with a visibility lower than 150 miles generates even higher levels of ozone concentration. Interestingly, according to the ninth rule, very low ozone concentrations are reached when a low temperature and a low inversion base temperature are combined.
Recall that to generate a prediction for a given query point $\mathbf{x}$, for each rule the corresponding ozone concentration is retrieved depending on whether $\mathbf{x}$ satisfies the rule conditions. Then all rule outputs for $\mathbf{x}$ are multiplied by their associated weight and added together. One can observe that rule frequencies and weights are not related. For example, the third rule has a higher weight than the most two frequent ones. It is clear that rule 3 has multiple constraints and is therefore more sensitive to data perturbation—hence a smaller frequency of appearance in the forest. On the other hand, its associated variance decrease in CART is more important than for the first two rules, leading to a higher weight in the linear combination. Since rules 5 and 6 are strongly correlated, their weights are diluted.

Tuning SIRUS-R has three hyperparameters: the number of quantiles q, the number of trees M and the threshold p_0 to select the most frequent rules in the forest. Quantile discretization ensures that some rules can be selected by several trees. The number of quantiles should be

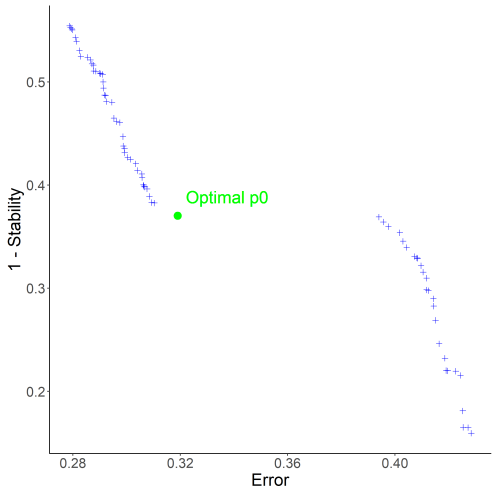


Figure 2.1: Pareto front of stability versus error (unexplained variance) when p_0 varies, with the optimal value in green for the "Ozone" dataset. The optimal point is the closest one to the ideal point $(0, 0.1)$ of 0 unexplained variance and 90% stability.

chosen small enough to obtain a small number of frequent rules but large enough so that the predictive performance of SIRUS-R is comparable to that of Breiman's random forests. Experiments (see Supplementary Material of Bénard et al., 2020) show that $q = 10$ is a good choice for several data set since it does not impact the predictive performance of a random forest built on discretized features. Clearly, the stability, predictivity, and computation time increase with the number of trees. Thus a stopping criterion is designed to grow the minimum number of trees that ensure stability and predictivity to be close to their maximum (see Bénard et al., 2020, for a detailed definition of this criterion). On the other hand, p_0 is tuned by cross-validation to maximize both stability and predictivity. To find a tradeoff between these two properties, we follow a standard bi-objective optimization procedure illustrated in Figure 2.1: the optimal value of p_0 is defined as the p_0 associated to stability and predictivity values that minimizes the Euclidean distance to the ideal point of 0 unexplained variance and 90% stability. This ideal point is chosen for its empirical efficiency: stability never reaches values higher than 90%, whereas unexplained variance can be arbitrarily close to 0, depending on the data. Additionally, we consider that long lists of rules are not interpretable and set an arbitrary limit of 25 rules. More precisely, if the initial list of rules output by SIRUS-R for a given p_0 exceeds 25, we choose a larger p_0 so that the output contains exactly 25 rules. Finally, we transform categorical variables into multiple binary variables.

Performance We compare SIRUS-R with its two main competitors RuleFit (with rule predictors only) and Node harvest. Both methods also extract large collection of rules from tree ensembles: RuleFit uses Importance Sampled Learning Ensemble (ISLE, J.H. Friedman et al., 2003) whereas Node harvest is based on random forests. Rule selection is performed by a sparse linear aggregation, respectively the Lasso (R. Tibshirani, 1996) for Rulefit and a constrained

Dataset	RuleFit	Node Harvest	SIRUS-R
Ozone	0.22	0.30	**0.62**
Mpg	0.25	0.43	**0.83**
Prostate	0.32	0.23	**0.48**
Housing	0.19	0.40	**0.80**
Diabetes	0.18	0.39	**0.66**
Machine	0.23	0.29	**0.88**
Galaxy	0.40	0.39	**0.77**
Abalone	0.31	0.38	**0.82**
Bones	0.59	0.52	**0.89**

Table 2.2: Mean stability over a 10-fold cross-validation for various public datasets.

Dataset	RuleFit	Node Harvest	SIRUS-R
Ozone	21	46	**11**
Mpg	40	43	**9**
Prostate	**14**	41	23
Housing	54	40	**6**
Diabetes	25	42	**12**
Machine	44	42	**9**
Galaxy	34	36	**4**
Abalone	58	35	**6**
Bones	5	13	**1**

Table 2.3: Mean model size over a 10-fold cross-validation for various public datasets.

quadratic program for Node harvest. To enable stability comparison, data is binned using 10 quantiles to fit Rulefit and Node harvest. While the predictive accuracy of SIRUS-R is comparable to Node harvest and slightly below RuleFit, the stability is considerably improved with much smaller rule lists. Experimental results are gathered in Table 2.2 for stability, Table 2.3 for model sizes, and Table 2.4 for predictive accuracy. All results are averaged over 10 repetitions of the cross-validation procedure. Since standard deviations are negligible, they are not displayed to increase readability. Table 2.2 shows that SIRUS-R outperforms its main competitors, Rulefit and Node harvest, in terms of stability, with a comparable predictive accuracy (see Table 2.4) and a smaller list of rules (see Table 2.3).

Remark 2.2. *In the above experiments, the parameter p_0 controlling the number of rules in the final model is tuned to achieve a tradeoff between stability and accuracy, which often leads to small rule lists, typically of the order of* 10*. If we drop the tuning of p_0 and set its value to extract, say,* 100 *rules from the forest, SIRUS-R predictivity overcomes Node harvest and is similar to RuleFit as shown in the column "**SIRUS-R 100 Rules**" of Table 2.4. On the other hand, stability drops to around* 50% *(70 − 80% when p_0 is tuned). Also notice that the final number of rules is around* 50 *because of the additional selection performed in the final rule*

Dataset	Random Forest	CART	RuleFit	Node Harvest	SIRUS-R	SIRUS-R 100 Rules	SIRUS-R D=3
Ozone	0.25	0.36	**0.27**	0.31	0.32	**0.26**	**0.28**
Mpg	0.13	0.20	**0.15**	0.20	0.21	**0.15**	**0.14**
Prostate	0.48	0.60	**0.53**	**0.52**	**0.48**	0.55	0.59
Housing	0.12	0.28	**0.16**	0.24	0.31	0.21	0.20
Diabetes	0.55	0.67	**0.55**	**0.58**	**0.56**	**0.54**	**0.55**
Machine	0.12	0.39	**0.26**	**0.29**	**0.29**	**0.27**	**0.26**
Galaxy	0.027	0.089	**0.031**	0.066	0.20	0.042	0.040
Abalone	0.44	0.56	**0.46**	0.61	0.66	0.64	0.63
Bones	0.64	0.67	**0.70**	**0.70**	**0.74**	**0.72**	**0.71**

Table 2.4: Proportion of unexplained variance estimated over a 10-fold cross-validation for various public datasets. For rule algorithms only, i.e., RuleFit, Node harvest, and SIRUS-R, minimum values are displayed in bold, as well as values within 10% of the minimum for each dataset.

aggregation: the model size is then comparable to Rulefit and Node harvest. We also observe that increasing the depth of trees to 3 *(with* 100 *extracted rules) does not significantly improve predictivity—see the last column "**SIRUS-R D=3**" of Table 2.4.*

2.1.5 Theoretical Analysis

Among the three minimum requirements for interpretable models, stability is the critical one. In SIRUS-R, simplicity is explicitly controlled by the hyperparameter p_0 and the limit of 25 rules. The wide literature on rule learning provides many experiments to show that rule algorithms have an accuracy comparable to tree ensembles. On the other hand, designing a stable rule procedure is more challenging (Letham et al., 2015; Murdoch et al., 2019). For this reason, we therefore focus our theoretical analysis on the asymptotic stability of SIRUS-R.

To get started, we need a rigorous definition of the rule extraction procedure. To this aim, we introduce a symbolic representation of a path in a tree, which describes how to reach a given node from the root. A path $\mathscr{P}$ is defined as

$$\mathscr{P} = \{(j_k, r_k, s_k),\ k = 1, \dots, D\},$$

where, for $k \in \{1, \dots, D\}$ ($D \in \{1, 2\}$), the triplet (j_k, r_k, s_k) describes how to move from level $(k-1)$ to level k, with a split using the coordinate $j_k \in \{1, \dots, p\}$, the index $r_k \in \{1, \dots, q-1\}$ of the corresponding quantile, and a side $s_k = L$ if we go to the left and $s_k = R$ if we go to the right. The set of all possible such paths is denoted by Π. An example is given in Figure 2.2.

Each tree of the forest is randomized in two ways: (i) the sample $\mathcal{D}_n$ is bootstrapped prior to the construction of the tree, and (ii) a subset of coordinates is randomly selected to find the best split at each node. This randomization mechanism is governed by a random variable that we call Θ. We define $T(\Theta, \mathcal{D}_n)$, a random subset of Π, as the collection of the 6 extracted paths from the random tree of depth $D = 2$ built with Θ and $\mathcal{D}_n$. Now, let $\Theta_1, \dots, \Theta_M$ be the independent randomizations of the M trees of the forest. With this notation, the empirical

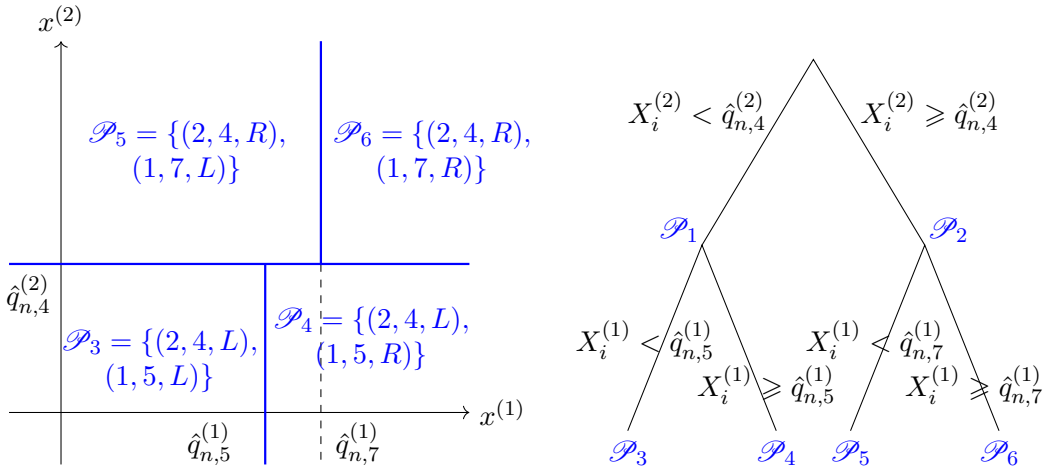


Figure 2.2: Example of a root node $\mathbb{R}^2$ partitioned by a randomized tree of depth 2: the tree on the right, the associated paths and hyperrectangles of length $D = 2$ on the left.

frequency of occurrence of a path $\mathscr{P} \in \Pi$ in the forest takes the form

$$\hat{p}_{M,n}(\mathscr{P}) = \frac{1}{M} \sum_{\ell=1}^{M} \mathbb{1}_{\mathscr{P} \in T(\Theta_\ell, \mathcal{D}_n)},$$

which is simply the proportion of trees that contain $\mathscr{P}$. By definition, $\hat{p}_{M,n}(\mathscr{P})$ is the Monte Carlo estimate of the probability $p_n(\mathscr{P})$ that a Θ-random tree contains a particular path $\mathscr{P} \in \Pi$, that is,

$$p_n(\mathscr{P}) = \mathbb{P}(\mathscr{P} \in T(\Theta, \mathcal{D}_n) | \mathcal{D}_n).$$

Next, we introduce all theoretical counterparts of the empirical quantities involved in SIRUS-R, which do not depend on the sample $\mathcal{D}_n$ but only on the unknown distribution of $(\mathbf{X}, Y)$. We let $T^\star(\Theta)$ be the list of all 6 paths contained in the theoretical tree built with randomness Θ, in which splits are chosen to maximize the theoretical CART-splitting criterion instead of the empirical one. We also let $p^\star(\mathscr{P}) = \mathbb{P}(\mathscr{P} \in T^\star(\Theta))$ be the probability that a given path $\mathscr{P}$ belongs to a theoretical randomized tree, which is the theoretical counterpart of $p_n(\mathscr{P})$. Finally, we recall that stability is assessed by the Dice-Sorensen index as

$$\hat{S}_{M,n,p_0} = \frac{2|\hat{\mathscr{P}}_{M,n,p_0} \cap \hat{\mathscr{P}}'_{M,n,p_0}|}{|\hat{\mathscr{P}}_{M,n,p_0}| + |\hat{\mathscr{P}}'_{M,n,p_0}|},$$

where $\hat{\mathscr{P}}'_{M,n,p_0}$ stands for the list of rules output by SIRUS-R fit with an independent sample $\mathcal{D}'_n$ and where the random forest is parameterized by independent copies $\Theta'_1, \ldots, \Theta'_M$.

Assumption 2.1. *The random variable $\mathbf{X}$ has a strictly positive density g with respect to the Lebesgue measure on $[0,1]^d$. Furthermore, for all $j \in \{1, \ldots, d\}$, the marginal density $g^{(j)}$ of $X^{(j)}$ is continuous, bounded, and strictly positive. Finally, the random variable Y is bounded.*

The following theorem states that SIRUS-R is asymptotically stable, i.e., provided that the sample size is large enough, the same list of rules is systematically output across several fits on independent samples.

Theorem 2.1. *Grant Assumption 2.1. Let* $\mathcal{U}^\star = \{p^\star(\mathscr{P}) : \mathscr{P} \in \Pi\}$ *be the set of all theoretical probabilities of appearance for each path* $\mathscr{P}$ *and* $p_0 \in [0,1] \setminus \mathcal{U}^\star$*. Then, for all* $\lambda > 0$*, provided* $a_n, M_n \to \infty$ *and* $a_n/n \to 0$*, we have*

$$\lim_{n\to\infty} \hat{S}_{M_n,n,p_0} = 1 \quad \textit{in probability.}$$

2.2 Variable importance

In this section, we analyze theoretically one of the two importance measures produced by Breiman's random forests: the Mean Decrease Impurity (MDI), which computes the weighted sum of decrease in variance at each split along a specified variable in all trees. Variable importances are widely used in practice to provide insights on how the prediction depends on the inputs, and in particular to identify the most relevant variables in a prediction. Understanding importance measures is then of interest when one is interested in interpretable models. Results presented here are based on Scornet (2020).

2.2.1 Introduction

In Breiman's (2001) original random forests, there exist two importance measures: the Mean Decrease Impurity (MDI, or Gini importance, see Breiman, 2002), which sums up the gain associated to all splits performed along a given variable; and the Mean Decrease Accuracy (MDA, or permutation importance, see Breiman, 2001a) which shuffles entries of a specific variable in the test data set and computes the difference between the error on the permuted test set and the original test set. Both measures are used in practice even if they possess several major drawbacks.

MDI is known to favor variables with many categories (see, e.g., Strobl, Boulesteix, Zeileis, et al., 2007; Nicodemus, 2011). Even when variables have the same number of categories, MDI exhibits empirical bias towards variables that possess a category having a high frequency (Nicodemus, 2011; Boulesteix, Bender, et al., 2012). MDI is also biased in presence of correlated features (Strobl, Boulesteix, Kneib, et al., 2008; Nicodemus and Malley, 2009). A promising way to assess variable importance in random forest is based on conditional inference, via the `R` package `party` (Strobl, Boulesteix, Kneib, et al., 2008; Strobl, Hothorn, et al., 2009). On the other hand, MDA seems to exhibit less bias, but its scale version (the default version in the `R` package `randomForest`) depends on the sample size and on the number of trees, the last being an arbitrary chosen parameter (Strobl and Zeileis, 2008). The interested reader may refer to Genuer, Poggi, and Tuleau (2008) and Genuer, Poggi, and Tuleau-Malot (2010) for an extensive simulation study about the influence of the number of observations, variables, and trees on MDA, together with the impact of correlation on this importance measure. Despite all their shortcomings, one great advantage of MDI and MDA is their ability to take into account interactions between features even if they are unable to determine the part of marginal/joint effect (Wright, Ziegler, and König, 2016).

From a theoretical perspective, there are only but a few results on MDA focusing on modified random forest procedures by Ishwaran (2007) and Zhu et al. (2015) with the notable exception of Gregorutti et al. (2017) who establish the population expression of MDA for a Gaussian linear model. Regarding the MDI, there are even fewer results, the most important one being by Louppe et al., 2013 who study the population MDI when all features are categorical (see also Sutera et al., 2016).

It is instructive to notice that there is no general result establishing the consistency of these variable importance measures toward population quantities when computed with the original random forest algorithm: all existing results focus on modified random forests version with, in

some cases, strong assumptions on the regression model. Therefore, there are no guarantees that using variable importance computed via random forests is suitable to select variables, which is nevertheless often done in practice.
We start by introducing the empirical expression of the MDI and its theoretical counterpart in Section 2.2.2. Section 2.2.3 is devoted to the link between MDI and the total variance explained by a random forest. Our analysis highlights in particular that MDI should not be computed with fully grown trees, which is unfortunately the default setting. In Section 2.2.4, we prove that MDI is consistent provided the regression model has no interactions and that the input variables are independent. To highlight the importance of these assumptions, we consider a very simple model which contains interactions in Section 2.2.5. In this case, the importance measure is inconsistent and therefore should not be used to assess variable importances.

2.2.2 Notation and general definition

Recall that for any cell $A \subset [0,1]^d$, letting $\mathcal{C}_A$ be the set of all possible cuts in A, the CART-split criterion (Breiman et al., 1984) is defined, as in equation (1.4) for any $(j,z) \in \mathcal{C}_A$,

$$\begin{aligned} L_{n,A}(j,z) = & \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_A)^2 \mathbb{1}_{\mathbf{X}_i \in A} \\ & - \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_{A_L} \mathbb{1}_{\mathbf{X}_i^{(j)} < z} - \bar{Y}_{A_R} \mathbb{1}_{\mathbf{X}_i^{(j)} \geqslant z})^2 \mathbb{1}_{\mathbf{X}_i \in A}, \end{aligned} \tag{2.2}$$

At each cell A, the best cut $(j_{n,A}, z_{n,A})$ is finally selected by maximizing $L_{n,A}(j,z)$ over $\mathcal{C}_A$, that is

$$(j_{n,A}, z_{n,A}) \in \operatorname*{arg\,max}_{(j,z) \in \mathcal{C}_A} L_{n,A}(j,z). \tag{2.3}$$

The Mean Decrease in Impurity (MDI) for the variable $X^{(j)}$ computed via a (CART) tree $\mathcal{T}$ is defined by

$$\widehat{\mathrm{MDI}}_{\mathcal{T}}(X^{(j)}) = \sum_{\substack{A \in \mathcal{T} \\ j_{n,A} = j}} p_{n,A} L_{n,A}(j_{n,A}, z_{n,A}), \tag{2.4}$$

where the sum ranges over all cells A in $\mathcal{T}$ that are split along variable j, and $p_{n,A}$ is the fraction of observations falling into A. In other words, the MDI of $X^{(j)}$ computes the weighted decrease in impurity related to splits performed along the variable $X^{(j)}$.
In order to study the (empirical) MDI defined in equation (2.4), we need to define and analyze the population version of MDI. First, we define a theoretical CART tree as above, except for the splitting criterion which is replaced by its population version. Namely, for all cells A and all splits $(j,z) \in \mathcal{C}_A$, the population version of the CART-split criterion is defined as

$$\begin{aligned} L_A^\star(j,z) = & \mathbb{V}[Y | \mathbf{X} \in A] - \mathbb{P}[\mathbf{X}^{(j)} < z \,|\, \mathbf{X} \in A] \; \mathbb{V}[Y | \mathbf{X}^{(j)} < z, \mathbf{X} \in A] \\ & - \mathbb{P}[\mathbf{X}^{(j)} \geqslant z \,|\, \mathbf{X} \in A] \; \mathbb{V}[Y | \mathbf{X}^{(j)} \geqslant z, \mathbf{X} \in A]. \end{aligned} \tag{2.5}$$

Therefore in each cell of a theoretical tree $\mathcal{T}^\star$, the best split $(j_A^\star, z_A^\star)$ is chosen by maximizing the population version of the CART-split criterion that is

$$(j_A^\star, z_A^\star) \in \arg\max_{(j,z)\in\mathcal{C}_A} L_A^\star(j,z).$$

Of course, in practice, we cannot build a theoretical tree $\mathcal{T}^\star$ since it relies on the true distribution $(\mathbf{X}, Y)$ which is unknown. A theoretical tree is just an abstract mathematical object, for which we prove properties that will be later extended to the (empirical) CART tree, our true object of interest. As for the empirical MDI defined above, the population MDI for the variable $X^{(j)}$ computed via the theoretical tree $\mathcal{T}^\star$ is defined as

$$\mathrm{MDI}_{\mathcal{T}^\star}^\star(X^{(j)}) = \sum_{\substack{A\in\mathcal{T}^\star \\ j_A^\star = j}} p_A^\star L_A^\star(j_A^\star, z_A^\star),$$

where all empirical quantities have been replaced by their population version and the theoretical CART tree is used instead of the empirical CART tree. We also let $p_A^\star = \mathbb{P}[\mathbf{X}\in A]$. Since forest prediction is simply the average of tree predictions, MDI computed via a random forest is nothing but the average of MDI computed via each tree.

2.2.3 Main theoretical result

For any theoretical CART tree $\mathcal{T}^\star$, and for any $k\in\mathbb{N}$, we denote by $\mathcal{T}_k^\star$ the truncation of $\mathcal{T}^\star$ at level k, that is the subtree of $\mathcal{T}^\star$ rooted at $[0,1]^d$, in which each leaf has been cut at most k times. Let $A_{\mathcal{T}_k^\star}(\mathbf{x})$ be the cell of the tree $\mathcal{T}_k^\star$ containing $\mathbf{x}$. For any function $f:[0,1]^d\to\mathbb{R}$ and any cell $A\subset[0,1]^d$, let

$$\Delta(f,A) = \mathbb{V}[f(\mathbf{X})|\mathbf{X}\in A]$$

be the variance of the function f in the cell A with respect to the distribution of $\mathbf{X}$. Proposition 2.1 states that the population MDI computed via theoretical CART trees can be used to decompose the variance of the output, up to some residual term which depends on the risk of the theoretical tree estimate.

Proposition 2.1. *Assume that $Y = f^\star(\mathbf{X}) + \varepsilon$, where ε is a noise satisfying $\mathbb{E}[\varepsilon|\mathbf{X}] = 0$ and $\mathbb{V}[\varepsilon|\mathbf{X}] = \sigma^2$ almost surely, for some constant σ^2. Consider a theoretical CART tree $\mathcal{T}^\star$. Then, for all $k\geqslant 0$,*

$$\mathbb{V}[Y] = \sum_{j=1}^d MDI_{\mathcal{T}_k^\star}^\star(X^{(j)}) + \mathbb{E}_{\mathbf{X}'}\left[\mathbb{V}[Y|\mathbf{X}\in A_{\mathcal{T}_k^\star}(\mathbf{X}')]\right], \tag{2.6}$$

where $\mathbf{X}'$ is an i.i.d. copy of $\mathbf{X}$ and $\mathbb{E}_{\mathbf{X}'}$ is the expectation with respect to $\mathbf{X}'$.

Note that the sum of population MDI is close to the R^2 measure used to assess the quality of regression model. The population R^2 is defined as

$$1 - \frac{\mathbb{E}_{\mathbf{X}'}\left[\mathbb{V}[Y|\mathbf{X}\in A_{\mathcal{T}_k^\star}(\mathbf{X}')]\right]}{\mathbb{V}[Y]} = \frac{\sum_{j=1}^d \mathrm{MDI}_{\mathcal{T}_k^\star}^\star(X^{(j)})}{\mathbb{V}[Y]}.$$

Hence, the sum of MDI divided by the variance of the output corresponds to the percentage of variance explained by the model, which is exactly the population R^2. Therefore, Proposition 2.1 draws a clear connection between MDI and the very classical R^2 measure.
We say that a theoretical tree $\mathcal{T}^\star$ is consistent if $\lim_{k\to\infty} \Delta(f^\star, A_{\mathcal{T}_k^\star}(\mathbf{X}')) = 0$. If a theoretical tree is consistent, then its population R^2 tends to $\mathbb{V}[f^\star(\mathbf{X})]/\mathbb{V}[Y]$ as shown in Corollary 2.1 below.

Corollary 2.1. *Grant assumptions of Proposition 2.1. Additionally, assume that* $\|f^\star\|_\infty < \infty$ *and, almost surely,* $\lim_{k\to\infty} \Delta(f^\star, A_{\mathcal{T}_k^\star}(\mathbf{X}')) = 0$. *Then, for all* $\gamma > 0$, *there exists* K *such that, for all* $k > K$,

$$\Big| \sum_{j=1}^{d} MDI^\star_{\mathcal{T}_k^\star}(X^{(j)}) - \mathbb{V}[f^\star(\mathbf{X})] \Big| \leqslant \gamma.$$

Consistency (and, in this case, tree consistency) is a prerequisite when dealing with algorithm intepretability. Indeed, it is hopeless to think that we can leverage information about the true data distribution from an inconsistent algorithm, intrinsically unable of modelling these data. Therefore, we assume in this section that theoretical trees are consistent and study afterwards variable importances produced by such an algorithm. In the next sections, we will be able to prove tree consistency for specific regression models, allowing us to remove the generic consistency assumption in our results.
Corollary 2.1 states that if the theoretical tree $\mathcal{T}_k^\star$ is consistent, then the sum of the MDI of all variables tends to the total amount of information of the model, that is $\mathbb{V}[f^\star(\mathbf{X})]$, when k tends to infinity. This gives a nice interpretation of MDI as a way to decompose the total variance $\mathbb{V}[f^\star(\mathbf{X})]$ across variables. Louppe et al. (2013) prove a similar result when all features are categorical. In their work, trees are finite since there exists only a finite number of variables with a finite number of categories. The major difference with our analysis is that trees we study can be grown to an arbitrary depth. We thus need to control the error of a tree stopped at some level k, which is exactly the right-hand term in equation (2.6).
According to Corollary 2.1, the sum of the MDI is the same for all consistent theoretical trees: even if there may exist many theoretical trees, the sum of MDI computed for each theoretical tree tends to the same value, regardless of the considered tree. Therefore, all consistent theoretical trees produce the same asymptotic value for the sum of population MDI.
In practice, one cannot build a theoretical CART tree or compute the population MDI. Proposition 2.2 below is the extension of Proposition 2.1 for the empirical CART procedure. Let $\mathcal{T}_n$ be the (empirical) CART tree built with data set $\mathcal{D}_n$ and let, for all k, $\mathcal{T}_{n,k}$ be the truncation of $\mathcal{T}_n$ at level k. Denote by $\widehat{\mathbb{V}[Y]}$ the empirical variance of Y computed on the data set $\mathcal{D}_n$. Define, for any function $f : [0,1]^d \mapsto \mathbb{R}$, the empirical quadratic risk via

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(X_i))^2.$$

Proposition 2.2 (Empirical version of Proposition 2.1)**.** *Let $\mathcal{T}_n$ be the CART tree, based on the data set $\mathcal{D}_n$. Then,*

$$\widehat{\mathbb{V}[Y]} = \sum_{j=1}^{d} \widehat{MDI}_{\mathcal{T}_n}(X^{(j)}) + R_n(\widehat{f}_{\mathcal{T}_n}), \tag{2.7}$$

where $\widehat{f}_{\mathcal{T}_n}$ is the estimate associated to $\mathcal{T}_n$, as defined in Algorithm 1.

Note that equations (2.6) and (2.7) in Proposition 2.1 and Proposition 2.2 are valid for general tree constructions. The main argument of the proof is simply a telescopic sum of the MDI which links the variance of Y in the root node of the tree to the variance of Y in each terminal node. These equalities hold for general impurity measures by providing a relation between root impurity and leaves impurity. As in Proposition 2.1, Proposition 2.2 proves that the R^2 of a tree can be written as

$$\frac{\sum_{j=1}^{d} \widehat{\mathrm{MDI}}_{\mathcal{T}_n}(X^{(j)})}{\widehat{\mathbb{V}[Y]}}, \tag{2.8}$$

which allows us to see the sum of MDI as the percentage of variance explained by the model. This is particularly informative about the quality of the tree modelling when the depth k of the tree is fixed.
Trees are likely to overfit when they are fully grown. Indeed, the risk of trees which contain only one observation per leaf is exactly zero. Hence, according to equation (2.7), we have

$$\widehat{\mathbb{V}[Y]} = \sum_{j=1}^{d} \widehat{\mathrm{MDI}}_{\mathcal{T}_n}(X^{(j)})$$

and the R^2 of such tree is equal to one. Such R^2 does not mean that trees have good generalization error but that they have enough approximation capacity to overfit the data. Additionally, for a fully grown tree $\mathcal{T}_n$, we have

$$\lim_{n\to\infty} \sum_{j=1}^{d} \widehat{\mathrm{MDI}}_{\mathcal{T}_n}(X^{(j)}) = \mathbb{V}[f^\star(\mathbf{X})] + \sigma^2.$$

For fully grown trees, the sum of MDI contains not only all available information $\mathbb{V}[f^\star(\mathbf{X})]$ but also the noise in the data. This implies that the MDI of some variables are higher than expected due to the noise in the data. The noise, when having a larger variance than the regression function, can induce a very important bias in the MDI by overestimating the importance of some variables. When interested by interpreting a decision tree, one must never use the MDI measures output by a fully grown tree. However, if the depth of the tree is fixed and large enough, the MDI output by the tree provides a good estimation of the population MDI as shown in Theorem 2.2.

Model 1. *The regression model satisfies $Y = f^\star(\mathbf{X}) + \varepsilon$ where $f^\star$ is continuous, $\mathbf{X}$ admits a density bounded from above and below by some positive constants and ε is an independent Gaussian noise of variance σ^2.*

Theorem 2.2. *Assume Model 1 holds and that for all theoretical CART trees $\mathcal{T}^\star$, almost surely,*

$$\lim_{k\to\infty} \Delta(f^\star, A_{\mathcal{T}_k^\star}(\mathbf{X})) = 0.$$

Let $\mathcal{T}_n$ be the CART tree based on the data set $\mathcal{D}_n$. Then, for all $\gamma > 0, \rho \in (0,1]$, there exists $K \in \mathbb{N}^\star$ such that, for all $k > K$, for all n large enough, with probability at least $1-\rho$,

$$\Big| \sum_{j=1}^{d} \widehat{MDI}_{\mathcal{T}_{n,k}}(X^{(j)}) - \mathbb{V}[f^\star(\mathbf{X})] \Big| \leqslant \gamma.$$

As for Corollary 2.1, Theorem 2.2 relies on the consistency of theoretical trees, which is essential to obtain positive results on tree interpretability. It is worth noticing that Theorem 2.2 is not a straightforward extension of Corollary 2.1. The proof is based on the uniform consistency of theoretical trees and combines several recent results on tree partitions to establish the consistency of empirical CART trees.
It is not possible in general to make explicit the contribution of each MDI to the sum. In other words, it is not easy to find an explicit expression of the MDI of each variable. This is due to interactions and correlations between inputs, which make difficult to distinguish the impact of each variable. Nevertheless, when the regression model can be decomposed into a sum of two independent terms, we can obtain more precise information on MDI. A general theorem is provided in Scornet (2020). We illustrate it on the particular case of additive models in the next section.

2.2.4 Additive Models

One class of models particularly easy to interpret is additive models with independent inputs defined below.

Model 2 (Additive model)**.** *The regression model writes $Y = \sum_{j=1}^{d} f_j^\star(X^{(j)}) + \varepsilon$, where each $f_j^\star$ is continuous; ε is a Gaussian noise $\mathcal{N}(0,\sigma^2)$, independent of $\mathbf{X}$; and $\mathbf{X} \sim \mathcal{U}([0,1]^d)$.*

By considering a model with no interaction and with independent inputs, we know that the contribution of a variable has an intrinsic definition which does not depend on other variables that are used to build the model. In Model 2, the explained variance of the model $\mathbb{V}[f^\star(\mathbf{X})]$ takes the form

$$\mathbb{V}[f^\star(\mathbf{X})] = \sum_{j=1}^{d} \mathbb{V}[f_j^\star(X^{(j)})],$$

which holds only because of independent inputs *and* the absence of interactions. The variance explained by the jth variable is defined unambiguously, and independently of which variables are considered in the model, as $\mathbb{V}[f_j^\star(X^{(j)})]$. Therefore any importance measure for $X^{(j)}$ which aims at decomposing the explained variance must output $\mathbb{V}[f_j^\star(X^{(j)})]$. It turns out that MDI computed via CART trees converges to this quantity.

Theorem 2.3 (Additive model)**.** *Assume that Model 2 holds. Let $\mathcal{T}^\star$ be any theoretical CART tree and $\mathcal{T}_n$ be the empirical CART tree. Then, for all $\gamma > 0$, there exists K such that, for all $k > K$, for all j,*

$$\left| MDI^\star_{\mathcal{T}^\star_k}(X^{(j)}) - \mathbb{V}[f^\star_j(X^{(j)})] \right| \leqslant \gamma.$$

Moreover, for all $\gamma > 0, \rho \in (0, 1]$, there exists K such that, for all $k > K$, for all n large enough, with probability at least $1 - \rho$, for all j,

$$\left| \widehat{MDI}_{\mathcal{T}_{n,k}}(X^{(j)}) - \mathbb{V}[f^\star_j(X^{(j)})] \right| \leqslant \gamma.$$

Since the MDI computed via random forests is nothing but the average of MDI computed by trees, Theorem 2.3 also holds for MDI computed via random forests. To the best of our knowledge, Theorem 2.3 is the first result highlighting that empirical MDI computed with CART procedure converge to reasonable values that can be used for variable selection, in the framework of Model 2.

Contrary to Theorem 2.2, Theorem 2.3 does not assume the consistency of the theoretical tree. Indeed, in the context of additive models, one can directly take advantage of Scornet et al. (2015a) to prove the consistency of theoretical CART trees.

Surprisingly, assuming that Model 2 holds, the population version of MDI has the same expression as the population version of MDA, up to factor 2 (see Gregorutti et al., 2017, for an expression of MDA). Thus, in the particular context of additive models with independent features, both MDI and MDA target the same quantity, which is the natural way to decompose the total variance. In this context, MDI and MDA can then be employed to rank features and select variables based on the importance values they produce.

Note that MDI is computed with truncated trees, i.e. trees that contain a large number of observations in each node. This is mandatory to ensure that the variance of the output in the resulting cell is correctly estimated. As mentioned before, considering fully grown trees would result in positively-biased MDI, which can lead to unintended consequences for variable selection. We therefore stress that MDI must not be computed using fully grown trees.

Theorem 2.3 proves that MDI computed with all CART trees are asymptotically identical: the MDI are consistent across trees. The only interest to use many trees to compute the MDI instead of a single one relies on the variance reduction property of ensemble methods, which allows us to reduce the variance of the MDI estimates when using many trees.

2.2.5 A model with interactions

In the absence of interaction and dependence between inputs, the MDI is a measure of variable importance which is independent of the set of variables included in the model, as stated in Theorem 2.3. However, this is not true as soon as we consider classes of models that present interactions or dependence. In this section, we study the influence of interactions via the following model. The results presented here together with the study of the influence of correlation between inputs for a specific model can be found in Scornet, 2020.

Model 3 (Multiplicative model)**.** *Let $\alpha \in \mathbb{R}$. The regression model writes*

$$Y = 2^d \alpha \prod_{j=1}^{d} X^{(j)} + \varepsilon.$$

where, for all j, $\alpha_j \in \mathbb{R}$; ε is a Gaussian noise $\mathcal{N}(0, \sigma^2)$, independent of $\mathbf{X}$; and $\mathbf{X} \sim \mathcal{U}([0,1]^d)$.

Theorem 2.4 (Multiplicative model)**.** *Assume that Model 3 holds. We have*

$$\mathbb{V}[f^\star(\mathbf{X})] = \alpha^2 \Big(\Big(\frac{4}{3}\Big)^d - 1 \Big). \tag{2.9}$$

For any cell $A = \prod_{\ell=1}^{d} [a_\ell, b_\ell] \subset [0,1]$, the coordinate which maximizes the population CART-split criterion (2.5) satisfies

$$j^\star \in \operatorname*{arg\,max}_{\ell \in \{1,\dots,d\}} \frac{b_\ell - a_\ell}{a_\ell + b_\ell},$$

and the splitting position is the center of the cell; the associated variance reduction is

$$\frac{\alpha^2}{4} (b_{j^\star} - a_{j^\star})^2 \prod_{\ell \neq j^\star} (a_\ell + b_\ell)^2.$$

Let $\mathcal{T}^\star$ be any theoretical CART tree and $\mathcal{T}_n$ be the empirical CART tree. Then, for all $\gamma > 0$, there exists K such that, for all $k > K$,

$$\Big| \sum_{j=1}^{d} MDI^\star_{\mathcal{T}^\star_k}(X^{(j)}) - \alpha^2 \Big(\Big(\frac{4}{3}\Big)^d - 1 \Big) \Big| \leqslant \gamma.$$

Moreover, for all $\gamma > 0, \rho \in (0,1]$, there exists K such that, for all $k > K$, for all n large enough, with probability at least $1 - \rho$,

$$\Big| \sum_{j=1}^{d} \widehat{MDI}_{\mathcal{T}_{n,k}}(X^{(j)}) - \alpha^2 \Big(\Big(\frac{4}{3}\Big)^d - 1 \Big) \Big| \leqslant \gamma.$$

Theorem 2.4 gives the explicit splitting position for a model with interactions. Deriving splitting positions allows us to prove that the theoretical tree is consistent which, in turns, proves that the sum of MDI converges to the explained variance $\mathbb{V}[f^\star(\mathbf{X})]$, according to Corollary 2.1.
We are also interested in obtaining the exact MDI expression for each variable, as established for additive models. However, Theorem 2.3 no longer applies since the regression model cannot be decomposed into independent additive components. It turns out to be impossible to derive an explicit expression for each MDI in this model. To see this, note that there exist many possible theoretical trees in Model 3. Two of them are displayed in Figure 2.3. They result in the same partition but the first split can be made either along $X^{(1)}$ (Figure 2.3, left) or $X^{(2)}$ (Figure 2.3, right). Surprisingly, the variable importance computed with these trees is larger for the variable that is split in the second step. A direct consequence of this fact is that two different theoretical trees can output two different variable importances, as shown by Lemma 2.1 below.

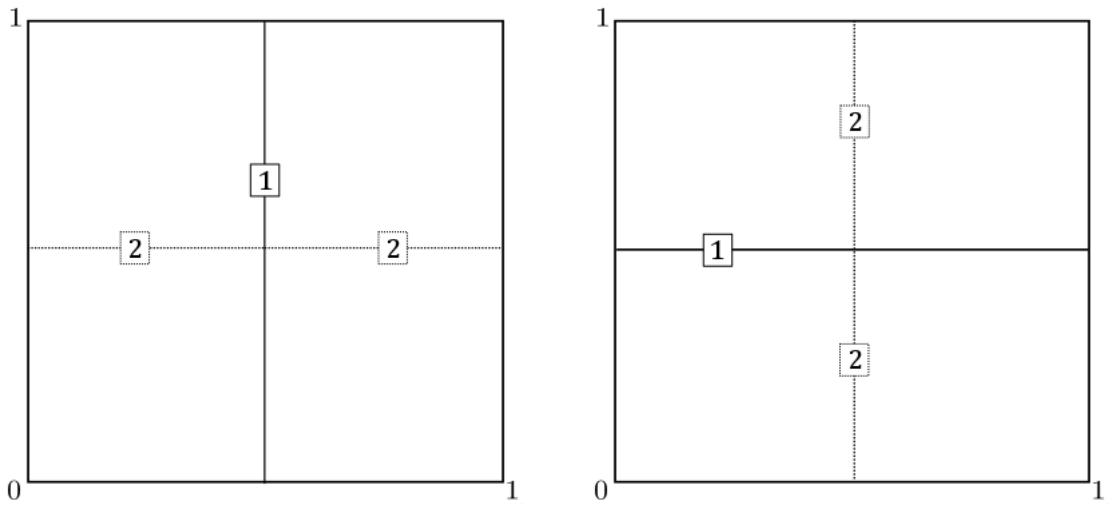


Figure 2.3: Two theoretical tree partitions of level $k = 2$: the first split is performed along $X^{(1)}$ (resp. $X^{(2)}$) for the tree on the left (resp. on the right).

Lemma 2.1. *Assume that Model 3 holds. Then, there exists two theoretical trees $\mathcal{T}_1^\star$ and $\mathcal{T}_2^\star$ such that*

$$\lim_{k\to\infty} \left(MDI^\star_{\mathcal{T}^\star_{2,k}}(X^{(1)}) - MDI^\star_{\mathcal{T}^\star_{1,k}}(X^{(1)}) \right) = \alpha^2/16.$$

According to Lemma 2.1, there exist (at least) two different theoretical trees for Model 3 which output different MDI. This is a major difference with additive models for which each theoretical tree asymptotically outputs the same MDI. Since MDI values are usually used to rank and select variables, the fact that each tree can output a different MDI, and therefore a different variable ranking, is a major drawback for employing this measure in presence of interactions.

One way to circumvent this issue could be to compute the MDI via a random forest: the randomization of splitting variables yields different trees and the aggregation of MDI across trees provides a mean effect of the variable importance. For example, in Model 3, one can hope to obtain importance measures that are equal, since Model 3 is symmetric in all variables. This is impossible with only one tree but is a reachable goal by computing the mean of MDI across many trees as done in random forests. More generally, for simple regression problems, a single tree may be sufficient to predict accurately but many shallow trees are needed to obtain accurate MDI values.

Remark 2.3. *Example in Figure 2.3 shows that splits occurring in the first levels of trees are not always associated to maximal decrease in variance. However, in the case of the additive models defined in Section 2.2.4, the decrease in variance is always larger in the first levels of a tree.*

2.3 Perspectives

We studied in this chapter the Mean Decrease in Impurity, one of the two importance measures computed by a random forest. We concluded that MDI computed with a single tree is not consistent in a general regression model which may contain interaction or dependencies between inputs. However, aggregation of trees, namely random forests, could overcome the issues raised in this chapter. A first line of research would be to exhibit the difference between the MDI computed with a single tree and the MDI computed with a random forest in a specific regression model, from a theoretical and empirical point of view.

Another natural line of research is to investigate the properties of Mean Decrease in Accuracy (MDA), the other importance measure output by random forests. It turns out that the analysis can be quite complicated for MDA, since permuting variables and thus breaking some relations between inputs makes the distribution of the permuted feature different from the original feature distribution. Developing a theoretical framework to analyse such procedure is definitely challenging but needed to understand how MDA should be used in practice.

Chapter 3

Neural networks and random forests

This chapter leverages the connection between decision trees and neural networks to improve the accuracy of feed-forward neural networks and random forests. Results presented here are based on Biau, Scornet, and Welbl (2019).

3.1 Introduction

The many parameters of neural networks (Goodfellow et al., 2016) make them a versatile and expressively rich tool for complex data modeling. However, their expressive power comes with the downside of increased overfitting risk, especially on small data sets. Conversely, random forests have fewer parameters to tune, but the greedy feature space separation by orthogonal hyperplanes results in typical stair or box-like decision surfaces, which may be suboptimal, particularly for colinear data with correlated features (Menze et al., 2011). In this context, the empirical studies by Welbl (2014) and Richmond et al. (2015) have highlighted the advantage of casting random forests into a neural network framework. The idea of constructing a decision tree and using this tree to obtain a neural network is by no means new—see for example Sethi (1990) and Sethi (1991), Brent (1991), and Chapter 30 in Devroye et al. (1996). The *conditional networks* from Ioannou et al. (2016) also use trainable routing functions to perform conditional transformations on the inputs—they are thus capable of transferring computational efficiency benefits of decision trees into the domain of convolutional networks.

The objective of this chapter is to use similarities between neural network and random forests to propose two new hybrid procedures that we call *neural random forests*. In a nutshell, given an ensemble of random trees, it is possible to restructure them as a collection of (random) multi-layered neural networks, which have sparse connections and less restrictions on the geometry of the decision boundaries. First, in Section 3.2, we show that any regression tree can be seen as a particular neural network. We build on this fact to rewrite a random forest as a neural network. This structure is used as warm start for the neural network to be trained. In Section 3.3, we define the two neural forest predictors. Both predictors exploit prior knowledge of regression trees for their initialization. This provides a warm start for neural networks, at the prediction level of traditional forests. Section 3.4 is devoted to the derivation of theoretical consistency

guarantees of the two methods. We illustrate in Section 3.5 the excellent performance of our approach both on simulated and real data sets, and show that it outperforms random forests in most situations.

3.2 Decision trees are neural networks

Consider a generic tree t_n built with a_n observations and containing $K_n \in \{2, \dots, a_n\}$ terminal nodes. In this section, we make explicit the equivalence between t_n and a three-layer neural network. An illustration of this equivalence is depicted in Figure 3.1.

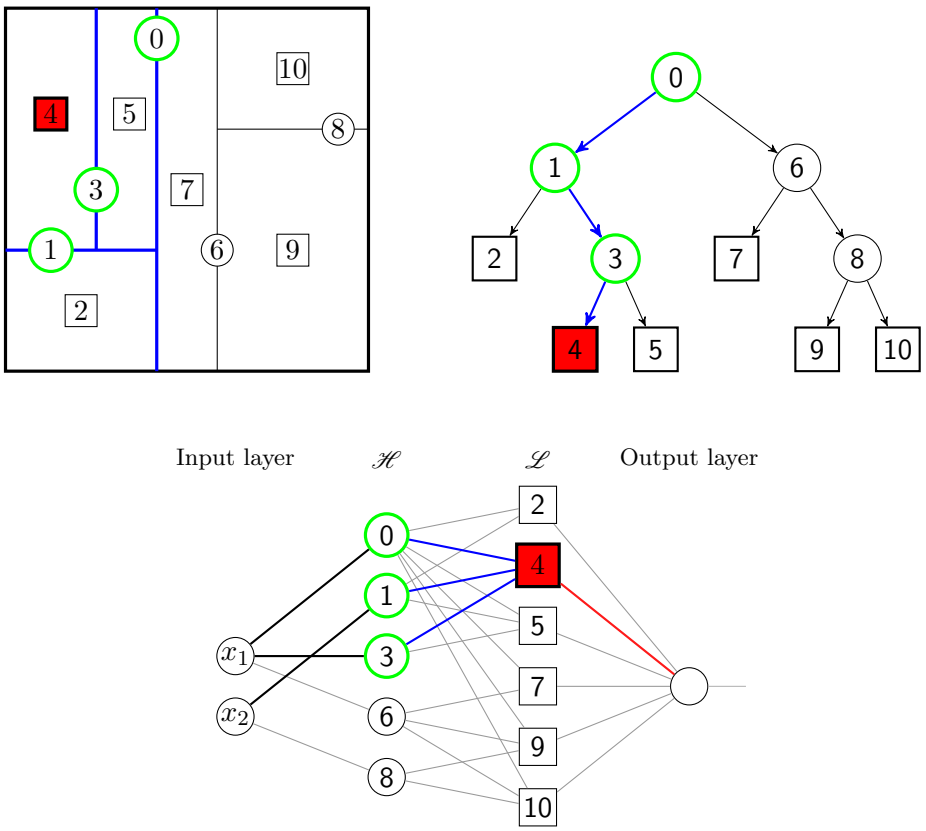


Figure 3.1: An example of regression tree (**top**) and the corresponding neural network (**down**).

First hidden layer. Let $\mathscr{H} = \{H_1, \dots, H_{K_n - 1}\}$ be the collection of all hyperplanes participating in the construction of t_n. We note that each $H_k \in \mathscr{H}$ is of the form $H_k = \{\mathbf{x} \in [0,1]^d : x^{(j_k)} = \alpha_{j_k}\}$, for some $j_k \in \{1, \dots, d\}$ and $\alpha_{j_k} \in [0,1]$. The first hidden layer of neurons is composed of $K_n - 1$ neurons, whose activations are defined, for $k = 1, \dots, K_n - 1$, by

$$u_k(\mathbf{x}) = \tau(x^{(j_k)} - \alpha_{j_k}), \tag{3.1}$$

where $\tau(x) = 2\mathbb{1}_{x \geqslant 0} - 1$ is a threshold activation function. In total, the first layer outputs the ± 1-vector $(u_1(\mathbf{x}), \dots, u_{K_n - 1}(\mathbf{x}))$, which describes all decisions of the inner tree nodes (including

nodes off the tree path of $\mathbf{x}$). We stress that each neuron k of this layer is connected to one, and only one, input $x^{(j_k)}$, and that this connection has weight 1 and offset $-\alpha_{j_k}$.

Second hidden layer. The identity of the leaf node containing $\mathbf{x}$ can now be extracted from the vector $(u_1(\mathbf{x}), \ldots, u_{K_n-1}(\mathbf{x}))$ using a weighted combination of the bits, together with an appropriate thresholding. More precisely, let $\mathscr{L} = \{L_1, \ldots, L_{K_n}\}$ be the collection of all tree leaves, and let $L(\mathbf{x})$ be the leaf containing $\mathbf{x}$. The second hidden layer has K_n neurons, one for each leaf, and assigns a terminal cell to $\mathbf{x}$ as explained below. We connect a unit k from layer 1 to a unit k' from layer 2 if and only if the hyperplane H_k is involved in the sequence of splits forming the path from the root to the leaf $L_{k'}$. The connection has weight $+1$ if, in that path, the split by H_k is from a node to a right child, and -1 otherwise. The output of neuron k' in the second layer is then given by

$$v_{k'}(\mathbf{x}) = \tau\left(\sum_{k \to k'} b_{k,k'} u_k(\mathbf{x}) + b^0_{k'}\right), \tag{3.2}$$

where notation $k \to k'$ means that k is connected to k' and $b_{k,k'} = \pm 1$ is the corresponding weight. The offset $b^0_{k'}$ is set to $b^0_{k'} = -\ell(k') + \frac{1}{2}$, where $\ell(k')$ is the length of the path from the root to $L_{k'}$. One can easily show that the second hidden layer outputs a vector of ± 1-bits $(v_1(\mathbf{x}), \ldots, v_{K_n}(\mathbf{x}))$ whose components equal -1 except the one corresponding to the leaf $L(\mathbf{x})$, which is $+1$.

Output layer. Let $(v_1(\mathbf{x}), \ldots, v_{K_n}(\mathbf{x}))$ be the output of the second hidden layer. If $v_{k'}(\mathbf{x}) = 1$, then the output layer computes the average $\bar{Y}_{k'}$ of the Y_i corresponding to $\mathbf{X}_i$ falling in $L_{k'}$. This is equivalent to take

$$t_n(\mathbf{x}) = \sum_{k'=1}^{K_n} w_{k'} v_{k'}(\mathbf{x}) + b_{\text{out}}, \tag{3.3}$$

where $w_{k'} = \frac{1}{2}\bar{Y}_{k'}$ for all $k' \in \{1, \ldots, K_n\}$, and $b_{\text{out}} = \frac{1}{2}\sum_{k'=1}^{K} \bar{Y}_{k'}$.

Remark 3.1. *The resulting network is sparse and therefore the number of parameters to optimize is lower than for a fully connected network. Indeed, considering a roughly balanced tree, the number of parameters in this network is $\mathscr{O}(K_n \log K_n)$ compared to $\mathscr{O}(dK_n + K_n^2)$ for a fully connected network with the same architecture. For large K_n, the first quantity can be much smaller than $\mathscr{O}(dK_n + K_n^2)$ and, in any case, it is independent of the dimension d—a major computational advantage in high-dimensional settings.*

3.3 Neural forests

Given a data set $\mathcal{D}_n$, we can grow a Breiman's random forest composed of M trees, such that: (i) prior to the tree construction, $a_n \geqslant 2$ points are drawn without replacement from the original data set and only these points are used to build the tree, (ii) splits are chosen to maximize the CART-splitting criterion as in Breiman's forests and the tree construction is stopped when it has exactly $K_n \in \{2, \ldots, a_n\}$ leaves, which may correspond to imbalanced trees, in the same manner as in Section 1.5.

Each decision tree of the above random forest can be translated into a neural network, whose connections, weights and offset are specified by the tree structure, using the equivalence stated in Section 3.2. A natural idea is then to keep the structure of each network intact and let the parameters vary in a subsequent network training procedure with backpropagation training. This additional training can potentially improve the predictions of the original random forest.
To allow for training based on gradient backpropagation, the activation functions must be differentiable. We choose to replace the original relay-type activation function $\tau(x) = 2\mathbb{1}_{x \geqslant 0} - 1$ with a smooth approximation of it: the hyperbolic tangent $\sigma(x) := \tanh(x)$. More precisely, we use $\sigma_1(x) = \sigma(\gamma_1 x)$ to replace the function τ in equation (3.1) (first layer) and $\sigma_2(x) = \sigma(\gamma_2 x)$ to replace τ in equation (3.2) (second layer). Similar ideas are developed in the so-called soft tree models (Jordan et al., 1994; Olaru et al., 2003; Geurts and Wehenkel, 2005; Yildiz et al., 2013).
We propose below two different ways to combine the M individual networks extracted from the M trees of the random forest. We call the two resulting estimates *Neural Random Forests*. In the sequel, we denote by $\widehat{f}_n(\mathbf{x}, \Theta_m)$ the estimator of the m-th tree, evaluated at $\mathbf{x}$, where Θ_m corresponds to the randomness used to build the m-th tree.
Method 1: Independent training. The parameters of each tree-type network are fitted network by network, independently of each other. With this approach, we end up with an ensemble of M "small" neural network estimates $\widehat{r}_n(\mathbf{x}, \Theta_m)$, $1 \leqslant m \leqslant M$, which are finally averaged to form the estimate

$$\widehat{r}_{M,n}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \widehat{r}_n(\mathbf{x}\,, \Theta_m) \tag{3.4}$$

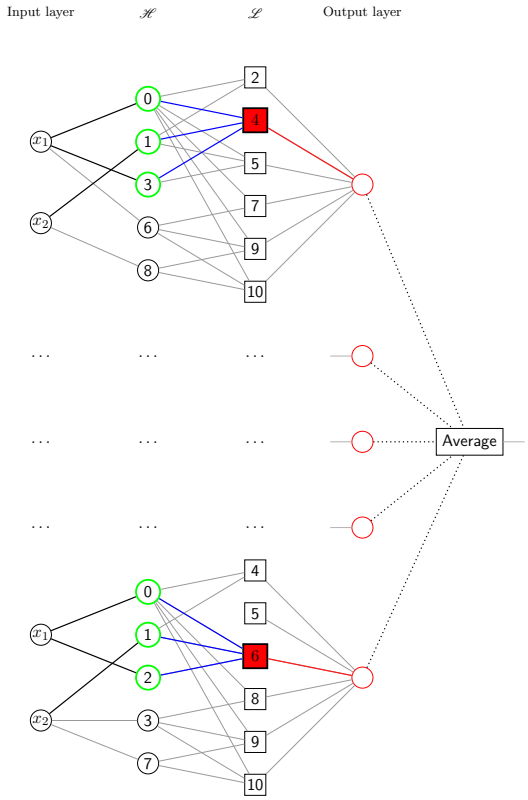
(see the illustration in Figure 3.2a), where we keep in mind that $\widehat{r}_{M,n}(\mathbf{x})$ depends on both $\Theta_1, \ldots, \Theta_M$ and the sample $\mathcal{D}_n$. This predictor has the flavor of a randomized ensemble (forest) of neural networks.
Method 2: Joint training. In this approach, the individual tree networks are first concatenated into one single "big" network, as shown in Figure 3.2b. The parameters of the resulting "big" network are then fitted jointly in one optimization procedure over the whole network. Although the hidden layers of the "small" networks are not connected across the sections belonging to each tree, this algorithm has two main differences with the previous one:

1. The output layer, which is shared by all "small" networks, computes a *weighted* combination of *all* outputs of the second-layer neurons, whose weights are optimized. This is quite different from the simple averaging computed in the previous method. We let $\widehat{s}_{M,n}(\mathbf{x})$ be the corresponding estimate.

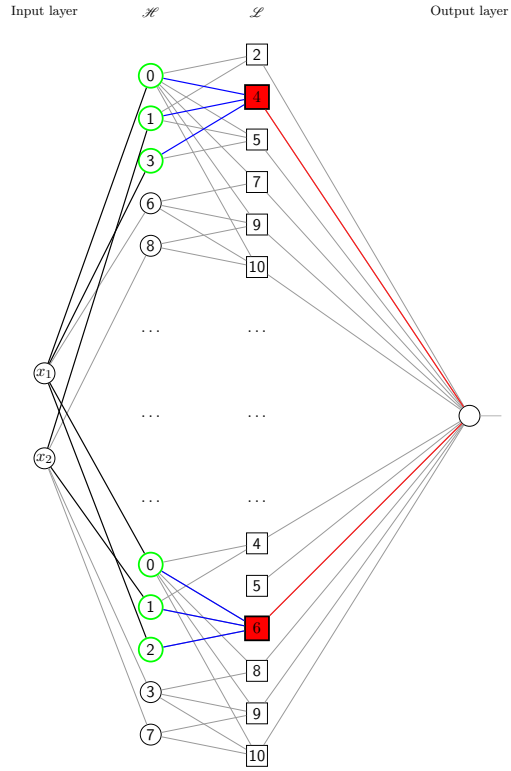
2. The optimization is performed in one single run over the whole "big" network, and not network by network.

The minimization program of each method, together with the statistical properties of the corresponding estimator, will be detailed in Section 3.4.

Remark 3.2. *Assuming that the trees are balanced, the first method performs, on average, M different optimization programs in a space of $\mathscr{O}(K_n \log K_n)$ parameters, whereas the second one accomplishes only one minimization in a space of average dimension $\mathscr{O}(MK_n \log K_n)$.*



(a) Method 1: Independent training.



(b) Method 2: Joint training.

Figure 3.2: Neural forests

3.4 Theoretical result

We start by detailing the construction of each estimator of the two previous methods (Independent training and Joint training) and we propose a theoretical analysis of their consistency.

3.4.1 Method 1: Independent training.

As described in the previous section, we build a forest and translate each tree into a neural network. The neural network corresponding to the m-th tree is represented by the parameter

$$\begin{aligned}\boldsymbol{\lambda} &= (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_{\text{out}}, b_{\text{out}}) \\ &\in \mathbb{M}(\mathscr{G}_1) \times \mathbb{R}^{K_n - 1} \times \mathbb{M}(\mathscr{G}_2) \times \mathbb{R}^{K_n} \times \mathbb{R}^{K_n} \times \mathbb{R},\end{aligned}$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_{\text{out}}$ and $\mathbf{b}_1, \mathbf{b}_2, b_{\text{out}}$ are respectively the weights and the bias of the first, second and third layer. To build our theoretical analysis, we need to restrict the range of these parameters by assuming that there exists a constant C_1 such that

$$\|\mathbf{W}_2\|_\infty + \|\mathbf{b}_2\|_\infty + \|\mathbf{W}_{\text{out}}\|_1 + |b_{\text{out}}| \leqslant C_1 K_n, \tag{3.5}$$

We emphasize that this requirement is mild and that it leaves a lot of freedom for optimizing the parameters. We note in particular that it is satisfied by the original random tree estimates as soon as Y is almost surely bounded, by choosing $C_1 = (\frac{3}{2} + \|Y\|_\infty)$. Accordingly, we denote by

$$\Lambda(\Theta_m, \mathscr{D}_n) = \{\boldsymbol{\lambda} = (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_{\text{out}}, b_{\text{out}}) : (3.5) \text{ is satisfied}\},$$

the set of possible values for the network parameters, so that the m-th neural network implements functions of the form

$$f_{\boldsymbol{\lambda}}(\mathbf{x}) = \mathbf{W}_{\text{out}}^\top \sigma_2\Big(\mathbf{W}_2^\top \sigma_1(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2\Big) + b_{\text{out}}, \quad \mathbf{x} \in \mathbb{R}^d,$$

where $\lambda \in \Lambda(\Theta_m, \mathscr{D}_n)$, and σ_1 and σ_2 are applied element-wise. Denoting by $\mathscr{F}(\Theta_m, \mathscr{D}_n) = \{f_{\boldsymbol{\lambda}} : \boldsymbol{\lambda} \in \Lambda(\Theta_m, \mathscr{D}_n)\}$ the set of all functions implementable by the neural network, our aim is to minimize the empirical loss over the class $\mathscr{F}(\Theta_m, \mathscr{D}_n)$, that is solving

$$\widehat{r}_n(\mathbf{x}, \Theta_m) \in \operatorname*{arg\,min}_{f \in \mathscr{F}(\Theta_m, \mathscr{D}_n)} \frac{1}{n} \sum_{i=1}^n |Y_i - f(\mathbf{X}_i)|^2.$$

By repeating this minimization process for each $m \in \{1, \dots, M\}$, we obtain a collection of (randomized) estimates $\widehat{r}_n(\mathbf{x}\,, \Theta_1), \dots, \widehat{r}_n(\mathbf{x}\,, \Theta_M)$, which are aggregated to form the estimate

$$\widehat{r}_{M,n}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M \widehat{r}_n(\mathbf{x}\,, \Theta_m).$$

The estimate $\widehat{r}_{M,n}$ is but a generalization of the random forest estimate to the neural network framework, with an additional relaxation of crisp to fuzzy tree node membership due to the hyperbolic tangent activation functions: samples not merely fall into one direction per split and one final leaf but simultaneously into several tree branches and leaves.

Remark 3.3. *Here we assumed the existence of a minimum, though not necessarily its uniqueness. In cases where a minimum does not exist, the same analysis can be carried out with functions whose error is arbitrarily close to the infimum, but for the sake of simplicity we stay with the assumption of existence throughout the paper. Note also that we do not investigate the properties of the gradient descent algorithm used in Section* 5*, and assume instead that the global minimum (if it exists) can be computed.*

3.4.2 Method 2: Joint training

As above, we build a forest and translate each decision tree into a small network. We then combine these networks into a bigger one. We now describe the parameters of this network. To this aim, we let $\mathbf{W}_{1,1}, \dots, \mathbf{W}_{1,M}$ and $\mathbf{b}_{1,1}, \dots, \mathbf{b}_{1,M} \in \mathbb{R}^{K_n - 1}$ be the respective weight matrices and offset vectors of the first hidden layers of the M "small" networks. Similarly, we denote by $\mathbf{W}_{2,1}, \dots, \mathbf{W}_{2,M}$ and $\mathbf{b}_{2,1}, \dots, \mathbf{b}_{2,M} \in \mathbb{R}^{K_n}$ the respective weight matrices and offset vectors of the second layer. We form the concatenated matrices $[\mathbf{W}_1]$, $[\mathbf{b}_1]$, $[\mathbf{W}_2]$, and $[\mathbf{b}_2]$, defined by

$$[\mathbf{W}_1] = \begin{pmatrix} \mathbf{W}_{1,1} & \cdots & \mathbf{W}_{1,M} \end{pmatrix}, \quad [\mathbf{b}_1] = \begin{pmatrix} \mathbf{b}_{1,1} \\ \vdots \\ \mathbf{b}_{1,M} \end{pmatrix},$$

and

$$[\mathbf{W}_2] = \begin{pmatrix} \mathbf{W}_{2,1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{2,2} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{W}_{2,M-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{W}_{2,M} \end{pmatrix}, \quad [\mathbf{b}_2] = \begin{pmatrix} \mathbf{b}_{2,1} \\ \vdots \\ \mathbf{b}_{2,M} \end{pmatrix}.$$

Let us finally denote by $\mathbf{W}_{\text{out}} \in \mathbb{R}^{MK_n}$ and $b_{\text{out}} \in \mathbb{R}$ the output weights and offset of the concatenated network. All in all, the parameters of the network are represented by a "vector"

$$[\boldsymbol{\lambda}] = \big([\mathbf{W}_1], [\mathbf{b}_1], [\mathbf{W}_2], [\mathbf{b}_2], \mathbf{W}_{\text{out}}, b_{\text{out}}\big),$$

where $[\mathbf{W}_1]$, $[\mathbf{b}_1]$, $[\mathbf{W}_2]$, and $[\mathbf{b}_2]$ are defined above. As in the first method, we restrict the range of variation of these parameters and assume that there exists a positive constant C_2 such that

$$\big\|[\mathbf{W}_2]\big\|_\infty + \big\|[\mathbf{b}_2]\big\|_\infty + \|\mathbf{W}_{\text{out}}\|_1 + |b_{\text{out}}| \leqslant C_2 K_n. \tag{3.6}$$

Note that, as in Section 3.4.1, the above constraint (3.6) is satisfied for the original random tree estimates, as soon as Y is almost surely bounded, with $C_2 = (M+1)C_1$. Therefore, letting $\Lambda(\Theta_1, \ldots, \Theta_M, \mathscr{D}_n) = \big\{([\mathbf{W}_1], [\mathbf{b}_1], [\mathbf{W}_2], [\mathbf{b}_2], \mathbf{W}_{\text{out}}, b_{\text{out}}) : (3.6) \text{ is satisfied}\big\}$, the "big" network implements functions of the form

$$f_{[\boldsymbol{\lambda}]}(\mathbf{x}) = \mathbf{W}_{\text{out}}^\top \sigma_2\Big([\mathbf{W}_2]^\top \sigma_1\big([\mathbf{W}_1]^\top \mathbf{x} + [\mathbf{b}_1]\big) + [\mathbf{b}_2]\Big) + b_{\text{out}}, \quad \mathbf{x} \in \mathbb{R}^d,$$

where $[\boldsymbol{\lambda}] \in \Lambda(\Theta_1, \ldots, \Theta_M, \mathscr{D}_n)$, and σ_1 and σ_2 are applied element-wise. Next, let

$$\mathscr{F}(\Theta_1, \ldots, \Theta_M, \mathscr{D}_n) = \big\{f_{[\boldsymbol{\lambda}]} : [\boldsymbol{\lambda}] \in \Lambda(\Theta_1, \ldots, \Theta_M, \mathscr{D}_n)\big\}.$$

Then the final estimate $\widehat{s}_{M,n}$ is obtained by minimizing the empirical error over the class $\mathscr{F}(\Theta_1, \ldots, \Theta_M, \mathscr{D}_n)$, that is

$$\widehat{s}_{M,n}(\mathbf{x}) \in \underset{f \in \mathscr{F}(\Theta_1, \ldots, \Theta_M, \mathscr{D}_n)}{\arg\min} \frac{1}{n} \sum_{i=1}^n \big|Y_i - f(\mathbf{X}_i)\big|^2.$$

3.4.3 Consistency

In order to analyze the consistency properties of the estimators $\widehat{r}_{M,n}$ and $\widehat{s}_{M,n}$, we first need to consider some specific class $\mathscr{F}$ of regression functions over $[0,1]^d$.

Definition 2. *We say that $f : [0,1]^d \to \mathbb{R}$ is additive if there exist real univariate functions $f_1, \ldots, f_d$ such that, for all $\mathbf{x} \in [0,1]^d$, $f(\mathbf{x}) = \sum_{j=1}^d f_j(x^{(j)})$. We let $\mathscr{F}$ be the class of continuous additive functions on $[0,1]^d$.*

Our main theorem states that the neural forest estimates $\widehat{r}_{M,n}$ and $\widehat{s}_{M,n}$ are consistent, provided the number K_n of terminal nodes and the parameters γ_1 and γ_2 are properly regulated as functions of n.

Theorem 3.1 (Consistency of $\widehat{r}_{M,n}$ and $\widehat{s}_{M,n}$). *Assume that $Y = f^\star(\mathbf{X}) + \varepsilon$, where $\mathbf{X}$ is uniformly distributed in $[0,1]^d$, $\|Y\|_\infty < \infty$, and $f^\star \in \mathscr{F}$. Assume, in addition, that $K_n, \gamma_1, \gamma_2 \to \infty$ such that, as n tends to infinity,*

$$\frac{K_n^6 \log(\gamma_2 K_n^5)}{n} \to 0, \quad K_n^2 e^{-2\gamma_2} \to 0, \quad \text{and} \quad \frac{K_n^4 \gamma_2^2 \log(\gamma_1)}{\gamma_1} \to 0.$$

Then, as n tends to infinity,

$$\mathbb{E}\left|\widehat{r}_{M,n}(\mathbf{X}) - r(\mathbf{X})\right|^2 \to 0 \quad \text{and} \quad \mathbb{E}\left|\widehat{s}_{M,n}(\mathbf{X}) - r(\mathbf{X})\right|^2 \to 0.$$

It is interesting to note that Theorem 3.1 still holds when the individual trees are subsampled and fully grown (that is, when $K_n = a_n$, i.e., one single observation in each leaf) as soon as the assumptions are satisfied with a_n instead of K_n. Put differently, we require that the trees of the forest are either pruned (restriction on K_n) or subsampled (restriction on a_n). If not, the assumptions of Theorem 3.1 are violated. So, to obtain a consistent prediction, it is therefore mandatory to keep the depth of the tree or the size of subsamples under control. Let us finally point out that the proof of Theorem 3.1 relies on the consistency of the individual "small" networks. Therefore, the proposed analysis does not really highlight the benefits of aggregating individual trees in terms of finite-sample analysis or asymptotic behavior.

3.5 Experiments

In this section we validate the Neural Random Forest (NRF) models experimentally (Method 1 and Method 2). We compare them with standard Random Forests (RF), Neural Networks (NN) with one, two, and three hidden layers, as well as Bayesian Additive Regression Trees (BART, Chipman et al., 2010).

3.5.1 Training procedure

Overall, the training procedure for the NRF models goes as follows. A random forest is first learned using the *scikit-learn* implementation (Pedregosa et al., 2011) for random forests. Based on this, the set of all split directions and split positions are extracted and used to build the neural network initialization parameters. The NRF models are then trained using the *TensorFlow* framework (Abadi et al., 2015).

Network optimization. The Networks are optimized with the MSE objective via *Adam* (Kingma et al., 2015) using minibatches of size 32, default hyperparameter values ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-08$), and initial learning rate 0.001 for which a stable optimization behavior could be observed. Each neural network was trained for 100 epochs. The final parameters are the ones that gave minimum validation error across all 100 epochs.
We generally found that using a lower value for γ_2 than for γ_1 (the initial contrast parameters of the activation functions in the second and first hidden layer, respectively) is helpful. This is because with a relatively small contrast γ_2, the transition in the activation function from -1 to

+1 is smoother and a stronger gradient signal reaches the first hidden layer in backpropagation training.
Wherever NRF validation loss was actually worse than the RF validation loss during *all* epochs, we kept the original RF model predictions. As a consequence, we can expect the NRF predictions to be at least as good as the RF predictions, since cases where further optimization is likely to have lead to overfitting are directly filtered out.
Concretely, for our experiments we used $\gamma_1 = 100$ and $\gamma_2 = 1$.

Sparse vs. fully connected optimization. All NRF networks described in the previous sections have sparse connectivity architecture between the layers due to the nature of the translated tree structures. However, besides training sparse NRF networks, a natural modification is to not limit network training to optimizing a small set of weights, but instead to relax the sparsity constraint and train a fully connected feed-forward network. With initial connection weights of 0 between neurons where no connections were described in the sparse framework, this model still gives the same predictions as the initial sparsely connected network. However, it can adapt a larger set of weights for optimizing the objective. During the experiments, we will refer to this relaxed NRF version as *fully connected*, in contrast to the *sparse* setting, where fewer weights are subject to optimization. Both algorithms fight overfitting in their own way: the sparse network takes advantage of the forest structure during the whole optimization process, whereas the fully connected network uses the smart initialization provided by the forest structure as a warm start. Indeed, the tendency of fully connected networks to overfit on small data sets can potentially be reduced with the inductive bias derived from the trees.

3.5.2 Asymptotic behavior

We start by investigating the asymptotic behavior of NRF procedures on an artificial data set created by sampling inputs $\mathbf{x}$ uniformly from the d-dimensional hypercube $[0, 1]^d$ and computing outputs y as

$$y(x) = \sum_{j=1}^{d} \sin(20x^{(j)} - 10) + \varepsilon,$$

where ε is a zero mean Gaussian noise with variance σ^2, which corrupts the deterministic signal. We choose $d = 2$ and $\sigma = 0.01$, and investigate the asymptotic behavior as the number of training samples increases. Figure 3.3 illustrates the RMSE for an increasing number of training samples and shows that the NRF (Method 2, fully connected) error decreases much faster than the RF error as sample size increases.

3.5.3 Benchmark comparison experiments

We now compare the NRF procedures with standard RF, standard NN, and BART on regression data sets from UCI Machine Learning Repository (Dua et al., 2017). To showcase the abilities of the NRF, we picked diverse, but mostly small regression data sets with few samples. These are the sets where RF often perform better than NN, since the latter typically require plentiful training data to perform reasonably well. It is on these small data sets where the benefits

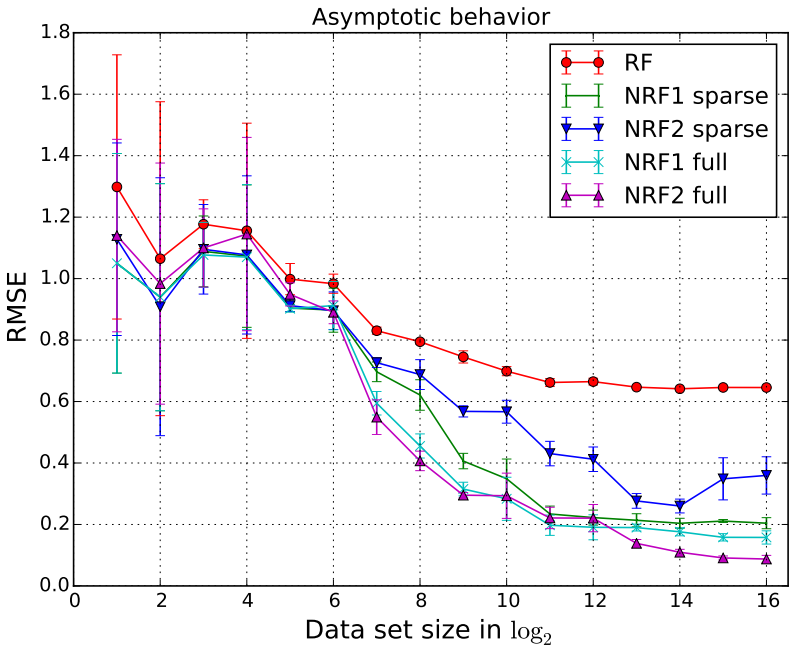


Figure 3.3: This figure shows the test RMSE for synthetic data with exponentially increasing training set size (x-axis). Solid lines connect the mean RMSE values obtained across 3 randomly drawn datasets for each dataset size, whereas error bars show the empirical standard deviation; 30 trees, maximum depth 6, $\gamma_1 = 100$, $\gamma_2 = 1$.

of neural optimization can usually not be exploited, but with the specific inductive bias of the NRF it becomes possible. We also study NRF performance on a larger data set, *Protein Tertiary Structure*, which contains 45000 observations.

Random forests are trained with 30 trees and maximum depth restriction of 6. The neural networks with one, two, or three hidden layers (NN1, NN2, NN3) trained for comparison are fully connected and have the same number of neurons in each layer as the NRF networks of Method 2 (for the third layer of NN3, the number of neurons is the same as in the second layer of NRF2). The initial parameters of standard NN were drawn randomly from a standard Gaussian distribution, and the above training procedure was followed.

For comparison we also evaluate Bayesian Additive Regression Trees (BART, Chipman et al., 2010). BART is trained also with 30 trees, 1000 MCMC steps (after burn-in of 100 steps), and otherwise default hyperparameters from the *BayesTree* R implementation.

Data preparation. We shuffled the observations in each data set randomly and split it into training, validation, and test part in a ratio of 50/25/25. Each experiment is repeated 10 times with different randomly assigned training, validation, and test set.

Results. Table 3.1 summarizes the results in terms of RMSE (root mean squared error) for the different procedures. The first important comment is that all NRF procedures improve over the original RF, consistently across all data sets. So the NRF are indeed more competitive than the original random forests which they were derived from. These consistent improvements over the original RF can be observed both for sparse and fully connected NRF networks.

Data set	NN1	NN2	NN3	RF
Auto MPG	4.56 (0.83)	3.95 (0.39)	5.02 (0.72)	3.44 (0.38)
Housing	9.06 (0.85)	7.81 (0.71)	12.52 (1.08)	4.78 (0.88)
Crime	5.39 (0.42)	6.78 (0.47)	6.64 (0.31)	0.17 (0.01)
Forest Fires	96.7 (0.20)	**54.87 (34.33)**	97.9 (0.90)	95.47 (43.52)
Wisconsin	36.91 (0.88)	**34.71 (2.36)**	38.43 (2.88)	45.63 (3.53)
Concrete	10.18 (0.49)	10.21 (0.68)	11.52 (0.88)	8.39 (0.62)
Protein	6.12 (0.02)	6.11 (0.02)	6.12 (0.02)	5.06 (0.03)

NRF1 full	NRF2 full	NRF1 sparse	NRF2 sparse	BART
3.20(0.39)	3.35 (0.46)	3.28 (0.41)	3.28 (0.42)	**2.90 (0.33)**
4.34 (0.85)	4.68 (0.88)	4.59 (0.91)	4.62 (0.88)	**3.78 (0.51)**
0.16 (0.01)	0.16 (0.01)	**0.16 (0.01)**	0.16 (0.01)	**0.14 (0.01)**
54.47 (34.64)	78.60 (28.17)	68.51 (35.56)	82.80 (32.07)	55.04 (16.40)
37.12 (2.89)	41.22 (3.05)	40.70 (2.51)	38.03 (3.95)	**33.50 (2.26)**
6.28 (0.40)	6.44 (0.37)	7.42 (0.56)	7.78 (0.56)	**5.20 (0.34)**
4.82 (0.04)	**4.77 (0.05)**	4.82 (0.04)	4.77 (0.05)	**4.57 (0.04)**

Table 3.1: RMSE test set results (and their standard deviation) for each of the procedures across the different data sets. "Sparse" stands for the sparsely connected NRF procedure, "full" for the fully connected. Best results and second best results are displayed in bold font.

For most data sets, standard NN (regardless of the number of layers) do not achieve performance that is on par with the other procedures, though in some cases they give competitive RMSE score. Interestingly, NRF1 seems to mostly outperform NRF2 with few exceptions. In fact, BART aside, the best overall performance is achieved by the fully connected NRF1, i.e., the averaging of individually trained per-tree networks. So one typically obtains bigger benefits from averaging several independently optimized single-tree networks than from jointly training one large network for all trees. This means that ensemble averaging (Method 1) is more advantageous for the NRF procedure than the co-adaptation of parameters across trees (Method 2). We conjecture that by separating the optimization process for the individual networks, some implicit regularization occurs that reduces the vulnerability to overfitting when using Method 1.

3.6 Perspectives

In the above experiments, BART showed excellent performance: it is ranked first in 6 out of the 7 experiments. Although in this context BART should be seen as a benchmark, a good idea for future research is to use BART to design neural networks, and extend the approach presented in this chapter.

The connexion between neural networks and decision tree holds for neural networks with two

hidden layers. In practice, deep neural networks have a better generalization error than small neural networks. A first line of research would be to imagine a manner to write a tree as a network with many hidden layers. This can be done by adding skip connexions as in Resnet (K. He et al., 2016) or as in DeepForests (Zhou et al., 2017), or simply by adding identity levels after the two levels corresponding to the decision tree. There are probably more elegant ways to leverage on the similarity between decision trees and two-layers neural networks to build deeper neural networks.

One could use the above similarity to simply provide a warm start to a deep neural network, in line with the work presented in this chapter. If one were interested by obtaining a more interpretable learning algorithm, we could try to identify what are sufficient features for a network to be interpreted as a decision tree. We could thus design an optimization procedure on this network subset so that the final network, output by the optimization procedure, could be turned back into a tree. This would be a challenging task mixing statistical theory, gradient-based optimisation and potentially integer programming.

Chapter 4

Missing values

As volumes of data increase, they are harder to curate and clean. They may come from merging multiple databases and contain variables of different natures. Such heterogeneous data collections can lead to many missing values: samples only come with a fraction of the features observed. Data analysis with missing values has been extensively studied in the statistical literature (see, e.g., Rubin, 1976; Little and Rubin, 2019; Buuren, 2018; Mayer et al., 2019) but often focus on estimating parameters in parametric models as in the linear framework (Little, 1992; Jones, 1996). Only few notable exceptions study supervised-learning settings, where the aim is to predict a target variable given input variables and the missing values are both in the training and the test sets (Zhang et al., 2005; Pelckmans et al., 2005; Liu et al., 2016).

Our aim in this chapter is to formalize the objective of supervised learning in the presence of missing values. In Section 4.1, we present a general framework to handle supervised learning problems with missing values. We prove theoretically that methods based on imputation, close to the ones widely used in practice, are consistent. We compare empirically imputation procedures with methods able to directly handle missing values, such as tree-based procedures. In Section 4.2, we study a specific regression model in which we incorporate missing values: the well-studied linear model. We show that the numerous missing patterns turn this simple model into many different linear models, which makes the dependence between inputs and outputs more complex. We thus design two linear estimates for which we establish finite-sample bounds. We compare empirically their performance with existing methods and a combination of these two linear procedures via neural networks. Results are based on Josse, Prost, et al. (2019) (Section 4.1) and Le Morvan et al. (2020) (Section 4.2).

4.1 Consistency of supervised learning with missing values

4.1.1 Existing works on missing values

Existing methods to deal with missing values can be decomposed into three categories: listwise deletion, imputation and EM algorithm. We review hereafter these three types of methods.

Listwise deletion Listwise deletion, *i.e.* the removal of incomplete observations, may allow to train the model on complete data. Yet it may not suffice for supervised learning, as the test set may also contain incomplete data.

Imputation prior to analysis Imputation prior to analysis consists in *imputing* the data, forming a completed data set (Lakshminarayan et al., 1996; Yoon et al., 2018) that can be analyzed by any supervised learning method. Continuous variables are often imputed by their mean whereas discrete ones can be imputed by the mode or by adding an extra category. One can also use joint modeling approaches to leverage on the joint distribution of features (Little and Rubin, 2019). One of the most simple examples of joint modeling consists in assuming a Gaussian distribution for features, estimating the mean vector and covariance matrix from the incomplete data (using, e.g., an EM algorithm) and impute by the conditional expectation given the observed data and the estimated parameters. More powerful methods can be based on low-rank models (Hastie, Mazumder, et al., 2015; Josse, Sardy, et al., 2016), iterative random forests (Stekhoven et al., 2011) or deep learning approaches such as denoising autoencoders (DAEs, Vincent et al. 2008; Gondara et al. 2018) and generative adversarial networks (Li et al., 2017; Yoon et al., 2018).
However, single imputation (imputing each feature by a single value) is known to underestimate variances by forgetting that the imputed values were missing in the original data set (Little and Rubin, 2019). One solution to incorporate uncertainty is to use multiple imputation (MI, Rubin 1987) which generates many values for each missing entries, thus creating many different imputed data sets. An analysis can be applied to each data set and all predictions are aggregated to form the final output. Although many multiple imputation procedures are available (Murray, 2018), the case of discriminatory models is only rarely considered (see Wood et al., 2008; Liu et al., 2016). Finally, note that imputation is a different problem from predicting a target variable and a good imputation does not always lead to a good prediction (Zhang et al., 2005).

EM algorithm Imputation leads to two-step methods that are generic in the sense that any analysis can be performed from the same imputed data set. On the contrary, the expectation maximization (EM) algorithm (Dempster et al., 1977) proceeds directly in one step. It can thus be better suited to a specific problem but requires the development of a dedicated algorithm (see Little, 1992; Jiang et al., 2019, for application to linear and logistic regression).

We start by formalizing in Section 4.1.2 the problem of supervised learning with missing data. In Section 4.1.3, we prove two theorems justifying the use of multiple imputation and constant imputation prior to learning, two methods widely used in practice. Finally, to compare imputation and learning directly with missing values, we study empirically decision trees since they are able to handle directly missing values. Results are gathered in Section 4.1.4.

4.1.2 Supervised learning with missing values

Missing data In presence of missing values, we do not observe a complete input vector $\mathbf{X}$. To define precisely the observed quantity, we introduce the indicator vector $\mathbf{M} \in \{0,1\}^d$ which satisfies, for all $1 \leqslant j \leqslant d$, $M_j = 1$ if and only if X_j is not observed. The random vector $\mathbf{M}$ acts

as a mask on $\mathbf{X}$. To formalize incomplete observations, we use the incomplete feature vector $\widetilde{\mathbf{X}}$ (see Rubin, 1976; Rosenbaum et al. 1984, appendix B; Mohan et al. 2018; Yoon et al. 2018) defined as $\widetilde{X}_j = \texttt{NA}$ if $M_j = 1$, and $\widetilde{X}_j = X_j$ otherwise. Since $\mathbf{X} \in \mathbb{R}^d$, $\widetilde{\mathbf{X}}$ belongs to the space $(\mathbb{R} \cup \{\texttt{NA}\})^d$. We have

$$\widetilde{\mathbf{X}} = \mathbf{X} \odot (\mathbf{1} - \mathbf{M}) + \texttt{NA} \odot \mathbf{M},$$

where $\odot$ is the term-by-term product, with the convention that, for all one-dimensional x, $\texttt{NA} \cdot x = \texttt{NA}$. As such, when the data are real, $\widetilde{\mathbf{X}}$ can be seen as a mixed categorical and continuous variable, taking values in $(\mathbb{R} \cup \{\texttt{NA}\})^d$. Here is an example: for a given vector $\mathbf{x} = (1.1, 2.3, -3.1, 8, 5.27)$ with the missing pattern $\mathbf{m} = (0, 1, 0, 0, 1)$, we have

$$\widetilde{\mathbf{x}} = (1.1, \quad \texttt{NA}, \quad -3.1, \quad 8, \quad \texttt{NA}).$$

We also introduce notations $\mathbf{X}_o$ and $\mathbf{X}_m$, classic in the missing value literature. The vector $\mathbf{X}_o = o(\mathbf{X}, \mathbf{M})$ is composed of observed entries in $\mathbf{X}$, whereas $\mathbf{X}_m = o(\mathbf{X}, 1 - \mathbf{M})$ contains the missing components of $\mathbf{X}$. Pursuing the above example,

$$\mathbf{x}_o = (1.1, \ \cdot\ , \ -3.1, \quad 8, \ \cdot\ ), \qquad \mathbf{x}_m = (\ \cdot\ , \quad 2.3, \ \cdot\ , \ \cdot\ , \quad 5.27).$$

These notations write partial feature vectors: "$\cdot$" in second and fifth positions in $\mathbf{x}_o$ means that we do not specify the corresponding component. More precisely, $\mathbf{x}_o$ specifies values of the first, third and fourth component of $\mathbf{x}_o$ but not whether the second and fifth values are observed or not (and if observed, we do not know the exact values). Notation $\widetilde{\mathbf{x}}$ is thus different from $\mathbf{x}_o$ since NA specifies which components are missing.

Empirical risk minimization with missing data We adapt below the paradigm of empirical risk minimization in presence of missing data. Since we only observe $\widetilde{\mathbf{X}}$, we aim at minimizing the empirical risk over a set of functions $\mathcal{F} \subset \{h, h : (\mathbb{R} \cup \{\texttt{NA}\})^d \to \mathbb{R}\}$, that is

$$\widehat{f}_n \in \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(\widetilde{\mathbf{X}}_i))^2. \tag{4.1}$$

Unfortunately, the half-discrete nature of $(\mathbb{R} \cup \{\texttt{NA}\})^d$, makes the problem difficult. Indeed, many learning algorithms do not work with mixed data types, such as $\mathbb{R} \cup \{\texttt{NA}\}$, but rather require a vector space. This is true in particular for gradient-based algorithms. As a result, the optimization problem (4.1) is hard to solve with typical learning tools.
In the presence of missing values, the Bayes estimate corresponding to problem (4.1) is defined as $f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) = \mathbb{E}[Y|\widetilde{\mathbf{X}}]$. Since the missing pattern $\mathbf{M}$ can take at most 2^d values, one can write

$$f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) = \sum_{\mathbf{m} \in \{0,1\}^d} \mathbb{E}\left[Y | o(\mathbf{X}, \mathbf{m}), \mathbf{M} = \mathbf{m}\right] \ \mathbb{1}_{\mathbf{M}=\mathbf{m}}, \tag{4.2}$$

where the Bayes estimate has been decomposed on each of the 2^d admissible missing patterns. This formulation highlights the combinatorial issues: solving (4.1) may require, as suggested by Rosenbaum et al. (1984), to estimate 2^d different submodels (that is $\mathbb{E}[Y|o(\mathbf{X}, \mathbf{m}), \mathbf{M} = \mathbf{m}]$ appearing in (4.2) for each $\mathbf{m} \in \{0, 1\}^d$) which grows exponentially with the number of variables.

Note that in practice, some missing patterns may not be observed (in both training and test sets), which then reduces the number of submodels to estimate in (4.2). For instance, if there is only complete data in the train set, the only submodel of interest is $\mathbb{E}\left[Y|o(\mathbf{X},\mathbf{m}),\mathbf{M}=\mathbf{m}\right]$ for $\mathbf{m}=(0,\ldots,0)$, which boils down to the regular supervised learning scenario on a complete data. We assume everywhere that train and test sets are drawn from the same data distribution, so that observable patterns in the train set are also observable patterns in the test set and vice versa. Otherwise, a distributional shift would occur between train and test sets and dedicated methods should be used (see, e.g., Sugiyama et al., 2017). This may happen for instance, when a study conducted on past data leads to operational recommendations, advising practitioners to focus on certain variables of interest. In that case, they will more likely measure them systematically.

Modifying existing algorithms or creating new ones to deal with the optimization problem (4.1) is in general a difficult task due to the numerous possible missing data patterns. Nevertheless, we will see that decision trees are particularly well suited to address this problem.

4.1.3 Bayes-risk consistency of imputation procedures

In this section, we study theoretically the consistency of imputation methods, that is their ability to produce a consistent estimate of the Bayes predictor $f^{\star}_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) = \mathbb{E}[Y|\widetilde{\mathbf{X}}]$ in presence of missing data. Contrary to many previous work, we do not assume any parametric distribution for the data.

We start by assuming that we are given the true complete regression function $f^{\star}$ and study the performance of applying this regression function to a test set with missing values, using two imputation strategies: unconditional single mean imputation and multiple imputation. In this context, we show that single mean imputation is not universally consistent and we give insights showing that conditional multiple imputation may be (assuming that we also know the conditional distribution of $\mathbf{X}_m$ given $\mathbf{X}_o$). While this setting seems at first sight unrealistic, in the case in which the missing pattern is independent of the data (Missing Completely At Random, MCAR, see Rubin, 1976), the function $f^{\star}$ can be estimated using a consistent algorithm applied on the complete data only, i.e., by deleting observations with missing values in the train set and apply a supervised procedure on the remaining observations. This strategy is relevant as soon as the training set is very large and the missingness pattern is MCAR.

Finally, we study a classical approach, described in Section 4.1.1, which consists first in imputing the training set, fit a learning algorithm on the imputed data, and predicting on a test set which has been imputed with the same method. Although mean imputation of variables is one of the most widely used approaches, it is highly criticised in the classic literature for missing data (Little and Rubin, 2019). Indeed, it leads to a distortion of the data distribution and consequently statistics calculated on the imputed data set are biased. A simple example is the correlation coefficient between two variables, which is biased towards zero if missing data are imputed by the mean. However, in a supervised learning setting the aim is not to compute statistics representative of the data set, but to minimize a prediction risk by estimating a regression function. For this purpose, we show that mean imputation may be appropriate and may lead to consistent estimation of the prediction function. This result is remarkable and is a first step to justify this well-known strategy.

Training on complete data and using single mean imputation is not universally consistent

One possible way to predict with missing data consists in training a learning algorithm on the complete data only, imputing the missing values in the test set by their means and apply the fitted estimator. The following example shows that even if the fitted estimator was exactly the true regression function, this procedure would not be consistent.

Example 1. *In one dimension, consider the following simple example,*

$$X_1 \sim U(0,1), \quad Y = X_1^2 + \varepsilon, \quad M_1 \sim \mathcal{B}(1/2) \perp\!\!\!\perp (X_1, Y),$$

with ε an independent centered Gaussian noise. The regression function $f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) = \mathbb{E}[Y|\widetilde{\mathbf{X}}]$ satisfies

$$\begin{aligned} f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) &= X_1^2 \cdot \mathbb{1}_{M_1=0} + \mathbb{E}[Y|\widetilde{X} = \texttt{NA}] \cdot \mathbb{1}_{M_1=1} \\ &= X_1^2 \cdot \mathbb{1}_{M_1=0} + \mathbb{E}[X_1^2] \cdot \mathbb{1}_{M_1=1} \\ &= X_1^2 \cdot \mathbb{1}_{M_1=0} + (1/3) \cdot \mathbb{1}_{M_1=1}. \end{aligned} \tag{4.3}$$

In the oracle setting where the distribution of (X_1, Y, M_1) is known, "plugging in" the mean imputation of X_1 yields the prediction

$$\begin{aligned} f_{imputation}(\widetilde{X}) &= X_1^2 \cdot \mathbb{1}_{M_1=0} + (\mathbb{E}[X_1])^2 \cdot \mathbb{1}_{M_1=1} \\ &= X_1^2 \cdot \mathbb{1}_{M_1=0} + (1/4) \cdot \mathbb{1}_{M_1=1}. \end{aligned} \tag{4.4}$$

In this example, mean imputation is not optimal: when X_1 is missing, the prediction obtained by mean imputation is $1/4$ (second term in (4.4)), whereas the optimal prediction is $1/3$ (second term in (4.3)). Inspecting (4.3) and (4.4) reveals that the poor performance of mean imputation is due to the fact that $\mathbb{E}[X_1^2] \neq (\mathbb{E}[X_1])^2$ which arises from the non-linear relation between Y and X_1. This highlights that training on complete cases only and then imputing by the mean is not consistent in general.

Training on complete data and using conditional multiple imputation

We consider the same setting as above but we use multiple imputation instead of single imputation by the mean. First, we train a learning algorithm on the complete cases only to obtain an estimation of the regression function $f^\star$. Then, recalling that we observe only $\mathbf{x}_o = o(\mathbf{x}, \mathbf{m})$ for a test point $\widetilde{\mathbf{x}}$, we draw missing values $\mathbf{X}_m$ from their conditional distribution $\mathbf{X}_m|\mathbf{X}_o = \mathbf{x}_o$ and compute the expectation of the regression function evaluated on these completed observations. The resulting multiple imputation function is given by

$$f^\star_{\mathrm{MI}}(\widetilde{\mathbf{x}}) = \mathbb{E}_{\mathbf{X}_m|\mathbf{X}_o=\mathbf{x}_o}[f^\star(\mathbf{X}_m, \mathbf{x}_o)]. \tag{4.5}$$

Note that we assume here to know the true regression function $f^\star$ and the true conditional distribution of $\mathbf{X}_m|\mathbf{X}_o = \mathbf{x}_o$ whereas in practice one only has access to estimation of these quantities.

Assumption 4.1 (Regression model)**.** *The regression model satisfies* $Y = f^\star(\mathbf{X}) + \varepsilon$*, where* $\mathbf{X}$ *takes values in* $\mathbb{R}^d$ *and* ε *is a centred noise independent of* $(\mathbf{M}, \mathbf{X})$*.*

Assumption 4.2 (Missingness pattern - a specific MAR scenario)**.** *The first* $0 < p < d$ *variables in* $\mathbf{X}$ *are always observed and the distribution of* M *depends only on these observed values.*

In Assumption 4.2, the missingness distribution depends only on variables that are always observed. This falls into the general definition of MAR (Missing At Random, see Rubin, 1976), which is often assumed in the classic literature to allow inference based on the likelihood maximization principle (Little, 1992; Jones, 1996; Jiang et al., 2019).

Theorem 4.1. *Grant Assumptions 4.1 and 4.2. Then the multiple imputation procedure, defined in (4.5), is pointwise consistent, that is, for all* $\widetilde{\mathbf{x}} \in (\mathbb{R} \cup \{\texttt{NA}\})^d$*,*

$$f^\star_{MI}(\widetilde{\mathbf{x}}) = f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{x}}).$$

Although Theorem 4.1 deals with theoretical quantities only, it is a first step to help justifying the practice which consists in training on complete cases only and then use a multiple imputation procedure on the test set. Note that training on complete cases only is not viable when applying single mean imputation on the test set but may be when multiple imputation on the test set is used, as highlighted by Theorem 4.1.

Imputation at train and test time

We now show that single imputation prior to learning may lead to consistent procedures. More precisely, we allow missing data on X_1 only and impute it by some constant $\alpha \in \mathbb{R}$, that is for each observed $\tilde{\mathbf{x}} \in (\mathbb{R} \cup \{\texttt{NA}\}) \times \mathbb{R}^{d-1}$, the imputed entry is defined as $\mathbf{x}' = (x'_1, x_2, \ldots, x_d)$ where $x'_1 = x_1 \mathbb{1}_{M_1=0} + \alpha \mathbb{1}_{M_1=1}$. We consider the following theoretical procedure: (i) impute the missing values on X_1 in the training set by α (ii) use a universally consistent supervised algorithm on the completed training set to estimate the regression function $f^\star_{\mathrm{SI}}(\mathbf{x}') = \mathbb{E}[Y|\mathbf{X}' = \mathbf{x}']$, (iii) assuming that the previous step gives us the exact regression function $f^\star_{\mathrm{SI}}(\mathbf{x}')$, the prediction for a new observation $\widetilde{\mathbf{x}}$ is computed by first imputing the possible missing value by α and then applying $f^\star_{\mathrm{SI}}(\mathbf{x}')$ to it, that is

$$f^\star_{\mathrm{SI}}(\mathbf{x}') = \mathbb{E}[Y|\mathbf{X}' = \mathbf{x}'].$$

Note that this procedure mimics the behaviour of an estimate learned on an imputed train set, with the notable difference that we assume to be given the exact regression function $f^\star_{\mathrm{SI}}$, whereas in practice, one can only obtain an estimation of this function.

Assumption 4.3 (Regression model)**.** *Let* $Y = f^\star(\mathbf{X}) + \varepsilon$ *where* $\mathbf{X}$ *has a continuous density* $g > 0$ *on* $[0,1]^d$*,* $\|f^\star\|_\infty < \infty$*, and* ε *is a centred noise independent of* $(\mathbf{X}, M_1)$*.*

Assumption 4.4 (Missingness pattern - a specific MAR scenario)**.** *The variables* $X_2, \ldots, X_d$ *are fully observed and the missingness pattern* M_1 *on variable* X_1 *satisfies* $M_1 \perp\!\!\!\perp X_1 | X_2, \ldots, X_d$ *and is such that the function* $(x_2, \ldots, x_d) \mapsto \mathbb{P}[M_1 = 1 | X_2 = x_2, \ldots, X_d = x_d]$ *is continuous.*

As for Assumption 4.2, Assumption 4.4 states that the missingness pattern is a specific MAR process since only X_1 can be missing with a probability that depends only on the other variables, which are always observed. The conditional distribution of M_1 is also assumed to be continuous to avoid technical complexities in the proof of Theorem 4.2.

Theorem 4.2. *Grant Assumptions 4.3 and 4.4. For all $\alpha \in \mathbb{R}$, letting*

$$\widetilde{\mathbf{X}} = \begin{cases} \mathbf{X}' & \text{if } X'_1 \neq \alpha \\ (\texttt{NA}, X_2, \ldots, X_d) & \text{if } X'_1 = \alpha \end{cases},$$

the single imputation theoretical procedure is equal to the Bayes function, that is, almost everywhere,

$$f^\star_{SI}(\mathbf{X}') = f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}). \tag{4.6}$$

Theorem 4.2 shows that it may be relevant to use the same imputation for the train and the test set. Indeed, the learning algorithm can detect the imputed value and use that information to identify that the entry was initially missing. However, if the imputed value changes from train set to test set (for example, if instead of imputing the test set with the mean of the variables of the train set, one imputes by the mean of the variables on the test set), the learning algorithm may fail, since the imputed data distribution would differ between train and test sets.

Multivariate missingness. Interestingly, Theorem 4.2 remains valid when missing values occur for variables $X_1, \ldots, X_j$ under the assumption that $(M_1, \ldots, M_j) \perp\!\!\!\perp (X_1, \ldots, X_j)$ conditional on $(X_{j+1}, \ldots, X_d)$ and if for every pattern $\mathbf{m} \in \{0,1\}^j \times \{0\}^{d-j}$, the functions $(x_{j+1}, \ldots, x_d) \mapsto \mathbb{P}[\mathbf{M} = \mathbf{m} | X_{j+1} = x_{j+1}, \ldots, X_d = x_d]$ are continuous.

Note that the precise imputed value α does not matter if the learning algorithm is universally consistent. By default, the mean is not a bad choice, as it preserves the first order statistic (mean) of the sample. However it can be shown that choosing α outside of the support of X_1 improves the conclusion of Theorem 4.2 by obtaining equation (4.6) for *all* (instead of *almost all*) $\mathbf{x}'$.

Universal consistency. In Theorem 4.2, we assume to be given a universally consistent algorithm which may appear as a strong restriction on the choice of the algorithm. However many estimators exhibit this property as, for example, local averaging estimate (kernel methods, nearest neighbors and decision trees, see Devroye et al., 1996). The key point of Theorem 4.2 is to state that universal consistency of a procedure with missing data results directly from the universal consistency of an algorithm applied to an imputed data set, therefore providing guarantees that consistent algorithms and imputation are useful tools to handle missing data.

Consistency for some specific distributions. In Theorem 4.2, we assume to be given a universally consistent algorithm. One can legitimately ask if the result still holds if an algorithm which is consistent only for some specific data distribution was used instead. For example, assume that data are generated via a linear model and a linear estimator is used after missing

values have been imputed. One can show that the 2^d submodels are not linear in general and consequently, using a single linear model on imputed data is not Bayes optimal. In a nutshell, Theorem 4.2 is not valid for procedures that are consistent for some specific data distributions only. We will study the specific case of linear model in Section 4.2.

4.1.4 Simulations

Until now, we have formalized the supervised learning problem with missing values and have given insights in the previous section to justify that imputation prior to learning may lead to consistent estimate when the missing values are generated via a specific MAR process (they depend only on observed values). In this section, we illustrate empirically this fact and compare two different strategies: imputation followed by a learning procedure, and learning without explicitly imputing. As mentioned before, there are not many methods able to handle directly (without imputation) missing values and decision trees are one of them. In this section, we thus use decision tree to compare the two aforementioned strategies in finite-sample settings.

Decision trees using MIA strategy

Trees are naturally equipped to handle missing values since they can take both discrete and continuous variables as input. When facing with missing values, one needs to adapt two steps in the tree construction: (i) choose the best split at each node by taking (or not) into account missing values, (ii) decide on which side to propagate missing values (which is equivalent of imputation by interval). Many ways to address these two points have been proposed. Based on a theoretical comparison of different splitting strategies (see Section 5 in Josse, Prost, et al., 2019), one of particular interest is "missing incorporated in attribute" (MIA, Twala et al. 2008) which builds splits using missing values and propagate them in one step. More precisely, MIA considers the following splits, for all cuts $(j, z) \in \{1, \dots, d\} \times [0, 1]$:

- $\{\widetilde{X}_j \leqslant z \text{ or } \widetilde{X}_j = \texttt{NA}\}$ vs $\{\widetilde{X}_j > z\}$,
- $\{\widetilde{X}_j \leqslant z\}$ vs $\{\widetilde{X}_j > z \text{ or } \widetilde{X}_j = \texttt{NA}\}$,
- $\{\widetilde{X}_j \neq \texttt{NA}\}$ vs $\{\widetilde{X}_j = \texttt{NA}\}$.

In a nutshell, for each cut (j, z), MIA tries to send all missing values to the left or to the right, and compute for each choice the usual CART-splitting criterion (1.4). It also computes the error associated to separating the observed values from the missing ones (third bullet point above). Finally, it chooses the split among the previous ones with the lowest error. This splitting procedure may detect relevant missingness patterns (to predict Y) and split on those patterns only, which may reduce drastically the number of submodels to explore to estimate accurately $\mathbb{E}[Y|\widetilde{\mathbf{X}}]$ (see equation (4.2)). MIA is thought to be a good method to apply when missing pattern is informative, as this procedure allows to cut with respect to missing/ non missing and uses missing data to compute the best splits. Note this latter property implies that the MIA approach does not require a different method to propagate missing data down the tree. MIA is implemented in the R packages `partykit` (Hothorn et al., 2015) and `grf` (J. Tibshirani et al., 2020).

Simulation setting

We consider the following regression model. Other detailed experiments can be found in Section 6 in Josse, Prost, et al. (2019).

Model 4 (Quadratic). *We consider a quadratic regression model* $Y = X_1^2 + \varepsilon$ *with covariates* $(X_1, \ldots, X_d)$ *distributed as* $\mathcal{N}(\mu, \Sigma)$ *with* $\mu = \mathbf{1}_d$ *and* $\Sigma = \rho \mathbf{1}\mathbf{1}^T + (1-\rho) I_d$.

In the experiment presented below, we use Model 4 with $d = 3$ and introduce missing values on X_1 according to the mechanisms described hereafter. Results are depicted in Figure 4.1.

Missing Pattern 1 (MCAR). *For* $p \in [0, 1]$ *the missingness is generated according to a Bernoulli distribution*

$$\forall i \in [\![1, n]\!], M_{i,1} \sim \mathcal{B}(p).$$

Missing Pattern 2 (Censoring MNAR). *A direct way to select a proportion* p *of missing values on a variable, that depends on the underlying value, is to crop them above the* $(1-p)$*-th quantile*

$$\forall i \in [\![1, n]\!], M_{i,1} = \mathbb{1}_{X_{i,1} > [X_1]_{(1-p)n}}.$$

Missing Pattern 3 (Predictive missingness). *Last, we can consider a pattern mixture model, letting* M_1 *be part of the regression function, with* $M_1 \perp\!\!\!\perp \mathbf{X}$ *and*

$$Y = X_1^2 + 3M_1 + \varepsilon.$$

We compare the following methods using implementation in the R (R Core Team, 2018) software and default values for the tuning parameters. These different strategies to handle missing values in tree procedures are evaluated by plugging them into individual decision trees, random forests, and gradient boosting respectively. Indeed, while we have only mentioned single decision trees so far, their aggregation into boosting or random forests is much more powerful in practice. Unless stated otherwise, the package used for decision trees is rpart (Therneau et al., 2018), the package used for random forests is `ranger` and for gradient boosting XGBoost (Chen et al., 2016).

- **MIA**: missing in attributes strategy described above.
- **block**: This methods chooses the best split by removing missing values and propagate all missing data as a block, to a side chosen by minimizing the error.
- **rpart+mask/ rpart**: CART with surrogate splits, with or without the indicator **M** in the covariates
- **ctree+mask/ ctree**: conditional trees, implemented in package `partykit` (Hothorn et al., 2015) with or without the indicator **M** in the covariates
- **impute mean+mask/ impute mean**: CART when missing values are imputed by unconditional mean with or without the indicator **M** added in the covariates
- **impute Gaussian**: CART when missing values are imputed by conditional expectation when data are assume to follow a Gaussian multivariate distribution. More precisely, the parameters of the Gaussian distribution are estimated with an EM algorithm (R package `norm`). Note that for numerical reasons, we shrink the estimated covariance matrix (replacing $\hat{\Sigma}$ by $0.99 \times \hat{\Sigma} + 0.01 \times \mathrm{tr}(\hat{\Sigma})\mathrm{I_d}$) before imputing.

To train and evaluate the performance of the methods, we decompose the observations into a training set (80%) and testing set (20%), and repeat the process 500 times. The performance measure is the percentage of explained variance computed on the test set. For visual purposes, we display the *relative* explained variance: for each of the 500 repetitions separately, we center the scores of all the methods by substracting the mean. This is also done separately for trees, forests, boosting.

Imputation In experiments, when imputation is needed, we use the same imputation values for the training set and the test set, as suggested by Theorem 4.2. In addition, we consider imputation with the missing indicator $\mathbf{M}$ in the features. The rationale behind this indicator is that it can be useful to improve the prediction when going beyond the hypothesis of Theorem 4.2, *i.e.* considering a finite sample, a learning algorithm with a low approximation capacity (as linear regression) and with missing values that can either be MNAR or depend on Y.

Results

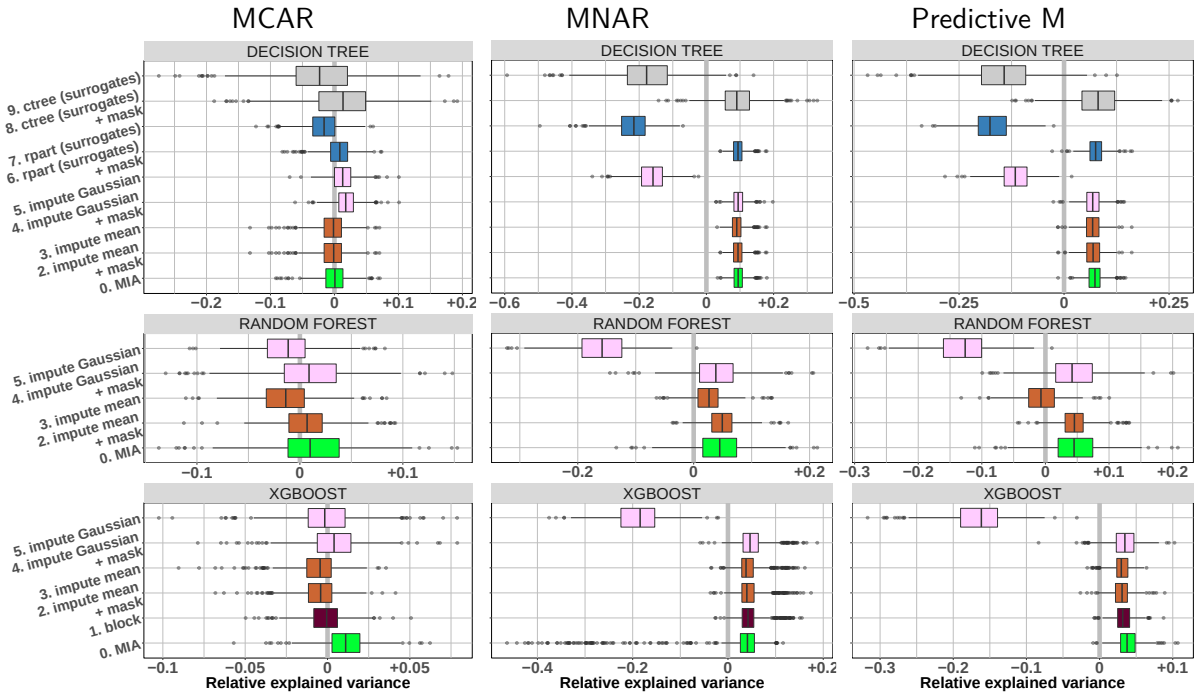


Figure 4.1: **Relative scores on model 4 with $d = 3$** • Relative explained variance for different mechanisms with 20% of missing values, $n = 1000$, $d = 3$ and $\rho = 0.5$.

Figure 4.1 presents the results for one choice of correlation between covariables and percentage of missing entries, as others give similar interpretation.
In the MCAR case, all decision tree methods behave in the same way. Having previously performed a "good" imputation, *i.e.* one that captures the relationships between variables such

as *impute Gaussian* in relation to mean imputation, slightly helps prediction. This is all the most true as the correlation between variables increases, which we have not displayed.
When the pattern is more complex (MNAR or predictive missingness), there are clear differences between methods. As expected, MIA achieves excellent performance, together with the other methods that code for missing values by adding the mask. The classic versions of rpart and ctree (cut-off on observed data and surrogate split) are not sufficient and need the addition of the mask. What is perhaps remarkable is that mean imputation with rpart (*impute mean*) still has excellent performances. Boosting exhibits the same behaviour as single decision trees. For random forests however mean imputation (*impute mean*) underperforms without the addition of a mask.

4.1.5 Discussion and conclusion

We have studied procedures for supervised learning with missing data, which departs from the classical point of view in missing data literature. Our theoretical and empirical results outline simple practical recommendations:

- To train and test on data with missing values, using the same imputation strategy may lead to consistent estimate. Single mean imputation may lead to consistent estimation, when using in conjunction with a powerful, non-linear model (Theorem 4.2).
- When missingness is related to the prediction target, imputation does not suffice and it is useful to add indicator variables of missing entries as features (Figure 4.1).

More work is needed to establish theoretical results in the finite sample regime and to establish if imputation is still a good method when a very simple estimator is fitted on imputed data. To move forward, we study in the next section, the impact of missing values in a linear model and provide finite sample bounds for the estimates we consider.

4.2 Linear regression with missing data

As surprising as it sounds, even the linear model, the simplest instance of regression models, has not been thoroughly studied with missing values and reveals unexpected challenges. This can be explained because data with missing values mix continuous (observed values) and categorical (missing-values indicators) data.
In this section, we consider data generated from a linear model, with missing values. After setting our objective and the notations in Section 4.2.1, we make explicit the Bayes predictor in presence of missing data in Section 4.2.2. We introduce two approaches to estimate the Bayes predictor: the first one consists in fitting one linear model per pattern of missing values, and the other is equivalent to imputing missing values by a constant and adding the pattern of missing values as input as in Section 4.1.4. In Section 4.2.3, we derive new generalization bounds for these two estimates, therefore establishing a regime in which one estimate is better than the other in terms of predictive performance. Due to the complexity of the learning task, we study the benefit of using a multilayer perceptron (MLP), a good compromise between the complexity of the first approach and the extreme simplicity of the second one. We show its consistency with enough hidden units in Section 4.2.4. Finally, Section 4.2.5 is devoted to an experimental study

showing that the MLP often gives the best prediction and can appropriately handle MNAR data.

4.2.1 Problem setting and Bayes predictor

We assume that the complete data $(\mathbf{X}, Y)$ follow a linear model, that is, there exist $\beta_0^\star \in \mathbb{R}$ and $\beta^\star \in \mathbb{R}^d$ such that

$$Y = \beta_0^\star + \langle \mathbf{X}, \beta^\star \rangle + \varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{4.7}$$

The Bayes predictor for the square loss in absence of missing values is defined as

$$f^\star \in \underset{f:\mathbb{R}^d \to \mathbb{R}}{\arg\min}\ R(f), \tag{4.8}$$

where $R(f) = \mathbb{E}[(Y - f(\mathbf{X}))^2]$ is the quadratic risk. In the case of a linear model, solving (4.8) yields $f^\star(\mathbf{x}) = \beta_0^\star + \langle \mathbf{x}, \beta^\star \rangle$. Since missing values may occur, we do not observe $\mathbf{X}$ but rather $\widetilde{\mathbf{X}} \in (\mathbb{R} \cup \{\texttt{NA}\})^d$ and the Bayes predictor in that case is defined as

$$f^\star_{\widetilde{\mathbf{X}}} \in \underset{f:(\mathbb{R}\cup\{\texttt{NA}\})^d \to \mathbb{R}}{\arg\min}\ R(f), \tag{4.9}$$

where $R(f) = \mathbb{E}[(Y - f(\widetilde{\mathbf{X}}))^2]$. Note that the Bayes predictor can be rewritten (see also equation (4.2)) as,

$$f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) = \sum_{\mathbf{m} \in \{0,1\}^d} \mathbb{E}\left[Y | o(\mathbf{X}, \mathbf{m}), \mathbf{M} = \mathbf{m}\right]\ \mathbb{1}_{\mathbf{M}=\mathbf{m}}.$$

This formulation highlights a combinatorial issue: estimating $f^\star_{\widetilde{\mathbf{X}}}$ may require to estimate 2^d different submodels.
The simplest way to deal with missing values is to use constant imputation, *i.e.* impute each feature $\widetilde{X}_j$ by a constant c_j: the most common choice is to impute by the mean or the median. However, it is also possible to optimise the constant with regards to the risk as shown below (see Proposition 3.1 and its proof in Le Morvan et al., 2020).

Proposition 4.1 (Optimal constants in linear model)**.** *Consider the least-square problem in which the design matrix is constructed by imputing* $\mathbf{X}$ *with zeros and by adding the mask* $\mathbf{M}$ *as additional features. Then, the optimal imputation constants* $(c_j^\star)_{j \in [\![1,d]\!]}$ *minimizing the empirical quadratic risk in a linear model are the coefficients associated to the mask* $\mathbf{M}$ *in the solution of the previous optimization problem.*

In an inferential framework, Jones (1996) showed that constant imputation leads to biased regression parameters. Even if our aim is prediction rather than estimation, the strategy of replacing missing values with a constant does not lead to Bayes-consistent predictors as shown in Section 4.2.2. The following example illustrates the complexity of missing values in a linear model.

Example 2 (Bayes predictor per missing pattern is not linear)**.** *Let $Y = X_1 + X_2 + \varepsilon$, where $X_2 = \exp(X_1) + \varepsilon_1$. Now, assume that only X_1 is observed. Then, the model can be rewritten as*

$$Y = X_1 + \exp(X_1) + \varepsilon + \varepsilon_1,$$

where $f(X_1) = X_1 + \exp(X_1)$ is the Bayes predictor. In this example, the submodel for which only X_1 is observed is not linear.

From Example 2, we deduce that there exists a large variety of submodels for a same linear model. In fact, the submodels depend on the structure of $\mathbf{X}$ and on the missing-value mechanism. Therefore, an extensive analysis seems unrealistic. In the following, we show that for Gaussian linear models, submodels can be easily expressed, and thus the Bayes predictors can be computed exactly for each submodel.

4.2.2 Bayes predictor for Gaussian linear regression

To ground our theoretical derivations, we use the very common pattern mixture model (Little, 1993), with Gaussian distributions:

Assumption 4.5 (Gaussian pattern mixture model)**.** *The distribution of $\mathbf{X}$ conditional on $\mathbf{M}$ is Gaussian, that is, for all $\mathbf{m} \in \{0,1\}^d$, there exist $\mu_\mathbf{m}$ and $\Sigma_\mathbf{m}$ such that*

$$\mathbf{X} \mid (\mathbf{M} = \mathbf{m}) \sim \mathcal{N}(\mu_\mathbf{m}, \Sigma_\mathbf{m}).$$

A particular case of this distribution is the case where $\mathbf{X}$ is Gaussian and independent of $\mathbf{M}$.

Definition 3. *For all $\beta \in \mathbb{R}^p$ and for all $\widetilde{\mathbf{x}} \in (\mathbb{R} \cup \{\texttt{NA}\})^d$, we let*

$$f_\beta(\widetilde{\mathbf{x}}) = \sum_{\mathbf{m} \in \{0,1\}^d} \left(\beta_{0,\mathbf{m}} + \sum_{j:m_j=0} \beta_{j,\mathbf{m}} x_j \right),$$

and $\mathcal{F} = \{f_\beta, \beta \in \mathbb{R}^p\}$.

The integer p in Definition 3 is the total number of parameters of the model which can be calculated by considering every sublinear model:

$$p = \sum_{k=0}^{d} \binom{d}{k} \times (k+1) = 2^{d-1} \times (d+2). \tag{4.10}$$

Proposition 4.2 (Expanded Bayes predictor)**.** *Under Assumption 4.5 and Model (4.7), the Bayes predictor $f^\star_{\widetilde{\mathbf{X}}}$ satisfies*

$$f^\star_{\widetilde{\mathbf{X}}} \in \arg\min_{f \in \mathcal{F}} \mathbb{E}[(Y - f(\widetilde{\mathbf{X}}))^2], \tag{4.11}$$

that is, there exists $\beta^\star_{exp} \in \mathbb{R}^p$ such that

$$f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{x}}) = f_{\beta^\star_{exp}}(\widetilde{\mathbf{x}}) = \sum_{\mathbf{m} \in \{0,1\}^d} \left(\beta^\star_{exp,0,\mathbf{m}} + \sum_{j:m_j=0} \beta^\star_{exp,j,\mathbf{m}} x_j \right), \tag{4.12}$$

where $\beta^\star_{exp}$ has an explicit expression (see the Supplementary Material of Le Morvan et al., 2020) as a function of $\beta^\star_0, \beta^\star, \mu_\mathbf{m}, \Sigma_\mathbf{m}$, for $\mathbf{m} \in \{0,1\}^d$.

An interesting aspect of this result is that the Bayes predictor is linear, though not on the original inputs, since each term in the second sum in equation (4.12) can be rewritten as a product $\mathbf{X}_{obs(\mathbf{m}_j)}\mathbb{1}_{\mathbf{M}=\mathbf{m}_j}$. Equation (4.12) shows that the Bayes predictor can be written as one linear model per pattern of missingness. Thanks to equation (4.12) and the explicit expression of $\beta^\star_{exp}$, the Bayes risk can be computed exactly, a fact used in the experiments in Section 4.2.5 to evaluate the performance of the different methods.
When d is not very small, estimating each submodel in equation (4.12) is computationally and statistically challenging. For this reason, we introduce hereafter a low-dimensional linear approximation of $f^\star_{\widetilde{\mathbf{X}}}$, without interaction terms.

Definition 4 (Linear approximation)**.** *For all $\beta \in \mathbb{R}^{2d+1}$ and for all $\tilde{\mathbf{x}} \in (\mathbb{R} \cup \{\texttt{NA}\})^d$, let*

$$f_{\beta,approx}(\tilde{\mathbf{x}}) = \beta_{0,0} + \sum_{j=1}^{d} \beta_{j,0}\mathbb{1}_{m_j=1} + \sum_{j=1}^{d} \beta_j x_j \mathbb{1}_{m_j=0},$$

and let $\mathcal{F}_{approx} = \left\{ f_{\beta,approx}, \beta \in \mathbb{R}^{2d+1} \right\}$. We define the linear approximation of $f^\star_{\widetilde{\mathbf{X}}}$ as

$$f^\star_{\widetilde{\mathbf{X}},approx} \in \underset{f \in \mathcal{F}_{approx}}{\arg\min}\ \mathbb{E}[(Y - f(\widetilde{\mathbf{X}}))^2]. \tag{4.13}$$

$f^\star_{\widetilde{\mathbf{X}},\text{approx}}$ is a linear function of the concatenated vector $(\mathbf{X}, \mathbf{M})$ where $\mathbf{X}$ is imputed by zeros, enabling a study of linear regression with that input.

4.2.3 Finite sample bounds for linear predictors

So far we have constructed two classes of functions: $\mathcal{F}$, whose elements combine many linear submodels (one per pattern of missingness) and which contains the Bayes predictor; $\mathcal{F}_{approx}$ whose elements are linear in $(\mathbf{X}, \mathbf{M})$. Optimizing the quadratic risk over the class $\mathcal{F}$, and therefore targeting the Bayes predictor, is a good choice in low dimension, but it is costly and leads to estimates with large variances in higher dimension. The class $\mathcal{F}_{approx}$ has a lower approximation capacity but each element has fewer parameters compared to those of the class $\mathcal{F}$, which helps reducing the variance of their estimate. For our theoretical analysis, we consider the new framework below.

Assumption 4.6. *We have $Y = f^\star_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}}) + noise(\widetilde{\mathbf{X}})$, where $noise(\widetilde{\mathbf{X}})$ is a centred noise conditional on $\widetilde{\mathbf{X}}$ and such that there exists $\sigma^2 > 0$ satisfying $\mathbb{V}[Y|\widetilde{\mathbf{X}}] \leqslant \sigma^2$ almost surely. Besides, assume that $\|f^\star\|_\infty < L$ and $Supp(\mathbf{X}) \subset [-1,1]^d$.*

For all $L > 0$ and for all functions f, we define the clipped version $T_L f$ of f at level L by, for all x,

$$T_L f(x) = \begin{cases} f(x) & \text{if } |f(x)| \leqslant L \\ L\ \text{sign}(f(x)) & \text{otherwise} \end{cases}$$

The expanded linear model is well specified and its risk is given by Theorem 4.3.

Theorem 4.3 (Expanded model)**.** *Grant Assumption 4.6. Let*

$$f_{\hat{\beta}_{exp}} \in \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(\widetilde{\mathbf{X}}_i))^2.$$

Then, there exists a constant $c > 0$ such that the risk of its predictions clipped at L satisfies

$$R(T_L f_{\hat{\beta}_{exp}}) \leqslant c \max\{\sigma^2, L^2\} \frac{2^{d-1}(d+2)(1+\log n)}{n} + \sigma^2.$$

Additionally, let $\mathcal{M} = \{\mathbf{m} \in \{0,1\}^d, \mathbb{P}[\mathbf{M} = \mathbf{m}] > 0\}$ be the set of all possible missing patterns and assume that there exists $\alpha \in (0,1]$ such that $|\mathcal{M}| = \alpha 2^d$. Then, there exists a constant $c_1 > 0$, such that for all n large enough,

$$R(T_L f_{\hat{\beta}_{exp}}) \geqslant \sigma^2 + \frac{2^d c_1}{n+1}.$$

Theorem 4.3 implies that the excess risk of the linear estimate of the expanded model is of order $\mathcal{O}(2^d/n)$ which grows exponentially fast with the original dimension of the problem. Theorem 4.4 below gives the same type of upper bound as Theorem 4.3 but for the approximation model.

Theorem 4.4 (Linear approximation model)**.** *Grant Assumption 4.6. Let*

$$f_{\hat{\beta}_{approx},L} \in \arg\min_{f \in \mathcal{F}_{approx}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(\widetilde{\mathbf{X}}_i))^2.$$

Then, there exists a constant $c > 0$ such that the risk of its predictions clipped at L satisfies

$$R(T_L f_{\hat{\beta}_{approx}}) \leqslant \sigma^2 + c \max\{\sigma^2, L^2\} \frac{2d(1+\log n)}{n} + 64(d+1)^2 L^2.$$

A direct consequence of Theorem 4.4 is that the excess risk of the linear approximation of $f^\star_{\widetilde{\mathbf{X}}}$ (Definition 4) is $\mathcal{O}(d^2 + d/n)$. Comparing this upper bound to the one obtained for the expanded model, we see that the risk of the expanded model is lower than that of the approximated model when $n \gg \frac{2^d}{d}$. Therefore, the expanded model is only useful for low-dimensional and large size data sets (large n, small d), but for data sets of reasonable size, the linear approximation may be preferred. As detailed in the next section, neural networks can be used as a compromise between both approaches.

4.2.4 Multilayer perceptron

Theorem 4.5 (MLP)**.** *Grant Assumption 4.6. A neural network (MLP) i) with one hidden layer containing 2^d hidden units, ii) ReLU activation functions iii) which is fed with the concatenated vector $(\mathbf{X} \odot (\mathbf{1} - \mathbf{M}), \mathbf{M})$ can produce exactly the Bayes predictor $f^\star_{\widetilde{\mathbf{X}}}$.*

The proof of Theorem 4.5 can be found in Le Morvan et al. (2020). The activation for each hidden unit is a linear function of a design matrix constructed by imputing $\mathbf{X}$ with zeros and adding the mask $\mathbf{M}$. Thus, just like for the optimal imputation problem of Proposition 4.1,

the weights of the linear function can be seen as either regular linear regression weights for the observed variables or learned imputation constants for the missing ones. In the context of a MLP with 2^d hidden units, we have 2^d such linear functions, meaning that each hidden unit is associated with one learned imputation vector. It turns out, as shown in the proof, that it is possible to choose the imputation vector of each hidden unit so that one hidden unit is always activated by points with a given missing-values pattern $\mathbf{m}$ but never by points with another missing-values pattern $\mathbf{m}' \neq \mathbf{m}$. As a result, all points with a given missing-values pattern fall into their own affine region, and it is then possible to adjust the weights so that the slope and bias of each affine region equals those of the Bayes predictor.

The number of parameters of a MLP with one hidden layer and 2^d units is $(d+1)2^{d+1}+1$. Compared to the expanded Bayes predictor which has $(d+1)2^{d-1}$ parameters, this is roughly 4 times more. This comes from the fact that the MLP does not directly estimate the slope and bias of a linear model per missing-values pattern. Firstly, for each affine region associated to a missing-values pattern, it estimates a slope for all variables, and not only for the observed ones. This doubles the number of parameters to be estimated. Secondly, it also needs to estimate the imputation constants for each hidden unit (or equivalently missing-values pattern) which again doubles the number of parameters to be estimated. As a result, the MLP should require more samples than the expanded Bayes predictor to achieve the Bayes risk. However, as we discuss below the parametrization of the MLP provides a natural way to control the capacity of the model. By contrast, there is no such easy and natural way to control the capacity of the expanded Bayes predictor.
The prediction function of a MLP with one hidden layer and n_h hidden units is a piecewise affine function with at most $\sum_{j=0}^{d} \binom{n_h}{j}$ regions (Pascanu et al., 2013). Thus, choosing $n_h = d$, we obtain 2^d affine regions, so potentially one per missing-value pattern. However, the slopes and biases of these affine regions are not independent, since they are linear combinations of the weights associated to each hidden unit. Yet, if the data-generating process has more regularities, 2^d different slopes may not be necessary to approximate it well. Varying the number of hidden units n_h thus explores an interesting tradeoff between model complexity –which comes at the cost of estimation error– and approximation error, to successfully address medium sample sizes problems.

4.2.5 Empirical study

We now run an empirical study to compare the three methods described in this section (expanded and approximate linear model, MLP) together with existing methods such as EM and conditional imputation. Details on experiments can be found in Le Morvan et al. (2020).

Experimental settings

Simulation models The data $(\mathbf{X}, \mathbf{M})$ is generated according to three different models. Two of them (**mixture 1** and **mixture 3**) are instances of Assumption 4.5 (with a mixture of respectively one and three components, where each missing-values pattern in equiprobable) while the third one is a classical Missing Non At Random (MNAR) model (Little and Rubin,

2019) defined as

$$P(M_j = 1|X_j) = \text{Probit}(\lambda_j(X_j - \mu_0)),$$

where λ and μ_0 are chosen so that the missing rate for each variable is 25%. This model allows to increase the probability of introducing missing values when the variable increases (or decreases), hence the denomination *selfmasking*.
The response Y is generated by a linear combination of the input variables as in equation (4.7). Note that to generate Y, we use the complete design matrix (without missing values). In these experiments, the noise ε is set to 0 and the regression coefficients β, β_0 are chosen equal to $\beta_0 = 1$ and $\beta = (1, 1, \ldots, 1)$.
For these three simulation scenarios, we compare five approaches:

ConstantImputedLR: optimal imputation method described in Proposition 4.1. The regression is performed using ordinary least squares.

EMLR: EM is used to fit a multivariate normal distribution for the $p+1$-dimensional random variable $(X_1, \ldots, X_p, Y)$. This method is expected to perform well only in the *mixture 1* setting.

MICELR: Multivariate Imputation by Chained Equations (MICE) is a widely used and effective conditional imputation method. We used the `IterativeImputer` of scikit-learn which adapts MICE to out-of-sample imputation.

ExpandedLR: full linear model described in Proposition 4.2 with L^2-penalty where the regularization parameter is chosen via cross validation on the grid $(10^{-3}, 1, 10^3)$.

MLP: Multilayer perceptron with one hidden layer whose size is varied between and 1 and 2^d hidden units. The input that is fed to the MLP is $(\mathbf{X}, \mathbf{M})$ where $\mathbf{X}$ is imputed with zeros and $\mathbf{M}$ is the mask. Rectified Linear Units (ReLU) were used as activation functions. The MLP was optimized with Adam, and a batch size of 200 samples. Weight decay was applied and the regularization parameter chosen by cross-validation over the grid $(10^{-1}, 10^{-2}, 10^{-4})$. The data is standardized before fitting the model.

When referring to a MLP in the figures, we use the notation *MLP_Wx* where x is a value indicating the number of units used. For an experiment in dimension d, *MLP_Wx* refers to a MLP with $x \times 2^d$ hidden units for *mixture 3*, or a MLP with $x \times d$ hidden units for *MCAR* and *MNAR*. The reason why we use this notation is because to achieve the same performance level for different dimensions, we must use a number of hidden units that is proportional to 2^d or to d according to the data type. In the experiments, we test three different hidden layer sizes, to compare low, medium and high capacities.
All models are compared to the Bayes risk computed whenever possible, i.e. for *mixture 1* and *mixture 3*. In all experiments, datasets are split into train and test set (75% train and 25% test) and the performance reported is the R2 score of the predictor on the test set, defined as

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2},$$

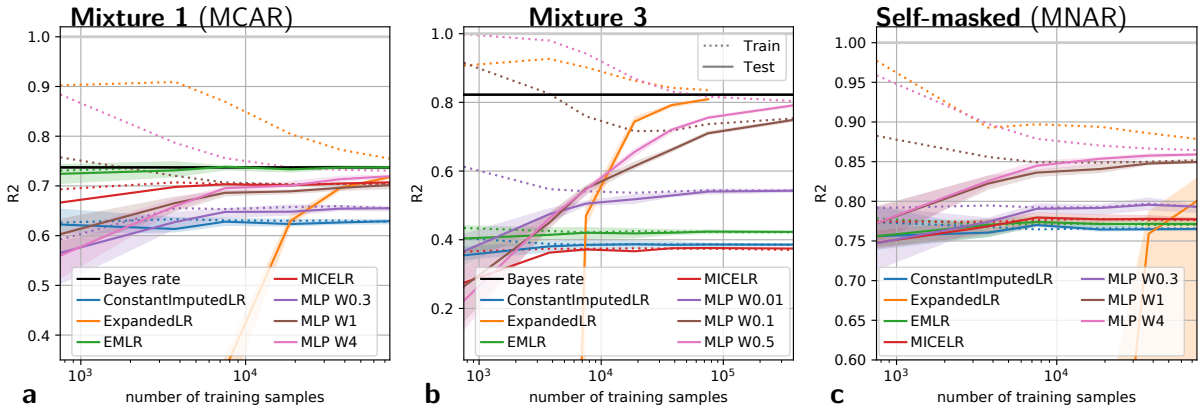


Figure 4.2: **Learning curves** Train and test R2 scores (respectively in dotted and in plain lines) as a function of the number of training samples, for each data type. Experiments were carried out in dimension $d = 10$. The curves display the mean and 95% confidence interval over 5 repetitions. The black horizontal line represents the Bayes risk (best achievable performance).

where $\hat{y}_i$ is the prediction and $\bar{y}$ is the mean of the output on the test set. Contrary to the quadratic risk, the R^2 score is always smaller than one, therefore providing us with insights about the overall absolute performances of the procedures.

Results

Mixture 1* and *Mixture 3 We first focus on the data types satisfying Assumption 4.5, i.e, *mixture 1* and *mixture 3*, since the theory developed in this section applies to them.

Figure 4.2 (a and b) presents the learning curves for the various methods, as the number of samples increases from 10^3 to 10^5 (5×10^5 for *mixture 3*) and the dimension is fixed to $d = 10$. First of all, this figure experimentally confirms that ExpandedLR and the MLP are Bayes consistent (since their error converge to the Bayes error, which is computable, thanks to Section 4.2). With enough samples, both methods achieve the best possible performance. It also confirms that ExpandedLR cannot be used in the small n large d regime. Indeed, between $10,000$ and $20,000$ samples are required for ExpandedLR to reach acceptable performances.

Overall, the learning curves (Figure 4.2) highlight three sample size regimes. We have a small sample size regime ($n < 1,000$) where EM is the best option. Indeed, EM is Bayes consistent for *mixture 1*. For *mixture 3*, EM performs badly but is not worse than the other methods in the small sample size regime. It is slightly better than ConstantImputedLR which still remains a reasonable option in this regime. MICELR follows the behavior of EM with slightly worse performance.

For $n > 30,000$ in *MCAR* and $n > 10,000$ in *mixture 3*, we are in a large sample size regime where ExpandedLR is an excellent choice, with performances on par or better than those of the MLP. The observation of small and large sample regimes support the theoretical analysis of Section 5. The fact that ExpandedLR outperforms the MLP for a larger sample range in *mixture 3* ($n > 10,000$) compared to *mixture 1* ($n > 30,000$) is explained by the fact that, to

reach a given performance, the MLP needs fewer parameters in *mixture 1* than in *mixture 3*, and thus fewer samples.
Finally, for $1,000 < n < 10,000$ or $1,000 < n < 30,000$, we have a last regime where the MLP should be the preferred option, since it outperforms both ConstantImputedLR and ExpandedLR. It shows that for medium size samples, it is possible to adapt the width of the hidden layer to reach a beneficial compromise between estimation and approximation error. This is particularly useful since many real datasets fall into this medium size sample regime.

Self-masking Self-masking, where the probability of missingness depends on the unobserved value, is a classical missing-values mechanism. Estimation in MNAR settings is notoriously difficult as most approaches, such as EM, rely on ignoring –marginalizing– the unobserved data which then introduces biases. As shown in the right panel of Figure 4.2, the MLP outperforms all other methods for this data type. This reflects the versatility of this method, which can adapt to all data generating schemes. ExpandedLR caps at a performance close to that of ConstantImputedLR, which highlights the dependence of this method on Assumption 4.5. Imputation, explicit with MICE or implicit with EM, performs poorly. This was expected since they do not take explicitly into account the influence of missing mechanisms to predict (Josse, Prost, et al., 2019).

4.2.6 Discussion and conclusion

We have studied how to minimize a prediction error on data where the response variable is a linear function of a set of partially observed features. Surprisingly, with these missing values, the Bayes-optimal predictor is no longer a linear function of the data. Under Gaussian mixtures assumptions we derive a closed-form expression of this Bayes predictor and used it to introduce a consistent estimation procedure of the prediction function. However, it entails a very high-dimensional estimation. Indeed, our generalization bounds –to our knowledge the first finite-sample theoretical results on prediction with missing values– show that the sample complexity scales, in general, as 2^d. We therefore study several approximations, in the form of constant imputation or a multi-layer perceptron (MLP), which can also be consistent given a sufficient number of hidden units. A key benefit of the MLP is that tuning its number of hidden units enables reducing model complexity and thus decreasing the number of samples required to estimate the model. Our experiments indeed show that in the finite-sample regime, using a MLP with a reduced number of hidden units leads to the best prediction. Importantly, the MLP adapts naturally to the complexity of the data-generating mechanism: it needs fewer hidden units and less data to predict well in a missing completely at random situation.
Our approach departs strongly from classical missing-values approaches, which rely on EM or imputation to model unobserved values. Rather, we tackle the problem with an empirical risk minimization strategy. An important benefit of this approach is that it is robust to the missing-values mechanism, unlike most strategies which require missing-at-random assumptions. Our theoretical and empirical results are useful to guide the choice of learning architectures in the presence of missing-values: with a powerful neural architecture, imputing with zeros and adding features indicating missing-values suffices.

4.3 Perspectives

In Section 4.1, we imputed missing data by some constant and then we fitted a decision tree on these imputed data. Theorem 4.2 shows that this procedure is asymptotically consistent but we have no guidance on how to choose the imputation constant. The mean is often used as a default imputation value but it is possible that imputing outside the support of $\mathbf{X}$ leads to better performances, in a finite-sample setting. Deriving risk bounds for methods consisting in imputing by some constant and then applying a decision tree would definitely help to choose the best imputation constant, which is of practical interest.

While studying linear model in Section 4.2, we focus on MCAR mechanisms, which are missing patterns that do not depend on the data. This assumption eases drastically the mathematical analysis and a natural extension would be to derive similar results (namely the Bayes predictor, its error, and a corresponding estimator) for missing patterns that depend on data, which are MAR (when missingness depends only on observed values) or MNAR (when missingness may depend on unobserved values).
Besides, in Section 4.2, the Gaussian assumption is made to ensure that each submodel is linear: this is a sufficient condition. Characterizing precisely the types of data that lead to submodels that are linear would be of interest to understand precisely the range of application of our analysis.

Bibliography

M. Abadi, A. Agarwal, P. Barham, and E. Brevdo et al. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. URL: http://tensorflow.org/.

S. Alelyani, Z. Zhao, and H. Liu (2011). "A dilemma in assessing stability of feature selection algorithms". In: *13th IEEE International Conference on High Performance Computing & Communication*. Piscataway: IEEE, pp. 701–707.

Y. Amit and D. Geman (1997). "Shape quantization and recognition with randomized trees". In: *Neural Computation* 9, pp. 1545–1588.

S. Arlot and R. Genuer (2014). "Analysis of purely random forests bias". In: *arXiv:1407.3939*.

S. Athey, J. Tibshirani, and S. Wager (2019). "Generalized random forests". In: *The Annals of Statistics* 47.2, pp. 1148–1178.

M. Banerjee and I.W. McKeague (2007). "Confidence sets for split points in decision trees". In: *The Annals of Statistics* 35, pp. 543–574.

C. Bénard, G. Biau, S. Da Veiga, and E. Scornet (2019). "SIRUS: making random forests interpretable". In: *arXiv:1908.06852*.

C. Bénard, G. Biau, S. Da Veiga, and E. Scornet (2020). "Interpretable Random Forests via Rule Extraction". In: *arXiv:2004.14841*.

E. Bernard, Y. Jiao, E. Scornet, V. Stoven, T. Walter, and J.-P. Vert (2017). "Kernel multitask regression for toxicogenetics". In: *Molecular informatics* 36.10, p. 1700053.

S. Bernard, L. Heutte, and S. Adam (2008). "Forest-RK: A new random forest induction method". In: pp. 430–437.

G. Biau (2012). "Analysis of a random forests model". In: *Journal of Machine Learning Research* 13, pp. 1063–1095.

G. Biau, F. Cérou, and A. Guyader (2010). "On the rate of convergence of the bagged nearest neighbor estimate". In: *Journal of Machine Learning Research* 11, pp. 687–712.

G. Biau and L. Devroye (2010). "On the layered nearest neighbour estimate, the bagged nearest neighbour estimate and the random forest method in regression and classification". In: *Journal of Multivariate Analysis* 101, pp. 2499–2518.

G. Biau, L. Devroye, and G. Lugosi (2008). "Consistency of random forests and other averaging classifiers". In: *Journal of Machine Learning Research* 9, pp. 2015–2033.

G. Biau and E. Scornet (2016). "A random forest guided tour (with comments and a rejoinder by the authors)". In: *Test* 25.2, pp. 197–227.

G. Biau, E. Scornet, and J. Welbl (2019). "Neural random forests". In: *Sankhya A* 81.2, pp. 347–386.

S. Boucheron, G. Lugosi, and P. Massart (2013). *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press.

A.-L. Boulesteix, A. Bender, J. Lorenzo Bermejo, and C. Strobl (2012). "Random forest Gini importance favours SNPs with large minor allele frequency: impact, sources and recommendations". In: *Briefings in Bioinformatics* 13.3, pp. 292–304.

A.-L. Boulesteix, S. Janitza, J. Kruppa, and I.R. König (2012). "Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, pp. 493–507.

A.-L. Boulesteix and M. Slawski (2009). "Stability and aggregation of ranked gene lists". In: *Briefings in Bioinformatics* 10, pp. 556–568.

L. Breiman (1996). "Bagging predictors". In: *Machine Learning* 24, pp. 123–140.

L. Breiman (2000). *Some infinity theory for predictor ensembles*. Technical Report 577, University of California, Berkeley.

L. Breiman (2001a). "Random forests". In: *Machine Learning* 45, pp. 5–32.

L. Breiman (2001b). "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical Science* 16, pp. 199–231.

L. Breiman (2002). *Setting up, using, and understanding random forests v3.1*, p. 58. URL: https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf.

L. Breiman (2004). *Consistency for a simple model of random forests*. Technical Report 670, University of California, Berkeley.

L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone (1984). *Classification and Regression Trees*. Wadsworth. ISBN: 0-534-98053-8.

R.P. Brent (1991). "Fast training algorithms for multi-layer neural nets". In: *IEEE Transactions on Neural Networks* 2, pp. 346–354.

P. Bühlmann and B. Yu (2002). "Analyzing bagging". In: *The Annals of Statistics* 30, pp. 927–961.

S. van Buuren (2018). *Flexible Imputation of Missing Data*. Chapman & Hall/CRC Interdisciplinary Statistics. CRC Press LLC.

A. Chao, R.L. Chazdon, R.K. Colwell, and T.-J. Shen (2006). "Abundance-based similarity indices and their estimation when there are unseen species in samples". In: *Biometrics* 62, pp. 361–371.

T. Chen and C. Guestrin (2016). "Xgboost: A scalable tree boosting system". In: *sigkdd international conference on knowledge discovery and data mining*. ACM, p. 785.

H.A. Chipman, E.I. George, and R.E. McCulloch (2010). "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* 4, pp. 266–298.

S. Clémençon, M. Depecker, and N. Vayatis (2013). "Ranking Forests". In: *Journal of Machine Learning Research* 14, pp. 39–73.

A. Criminisi, J. Shotton, and E. Konukoglu (2011). "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning". In: *Foundations and Trends in Computer Graphics and Vision* 7, pp. 81–227.

D.R. Cutler, T.C. Edwards Jr., K.H. Beard, A. Cutler, K.T. Hess, J. Gibson, and J.J. Lawler (2007). "Random forests for classification in ecology". In: *Ecology* 88, pp. 2783–2792.

A.P. Dempster, N.M. Laird, and D.B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38.

M. Denil, D. Matheson, and N. de Freitas (2013). "Consistency of online random forests". In: *International Conference on Machine Learning (ICML)*.

L. Devroye, L. Györfi, and G. Lugosi (1996). *A Probabilistic Theory of Pattern Recognition*. New York: Springer.

R. Díaz-Uriarte and S. Alvarez de Andrés (2006). "Gene selection and classification of microarray data using random forest". In: *BMC Bioinformatics* 7, pp. 1–13.

T.G. Dietterich (2000). "Ensemble methods in machine learning". In: *Multiple Classifier Systems*. Springer, pp. 1–15.

F. Doshi-Velez and B. Kim (2017). "Towards a rigorous science of interpretable machine learning". In: *arXiv:1702.08608*.

D. Dua and C. Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

R. Duroux and E. Scornet (2018). "Impact of subsampling and tree depth on random forests". In: *ESAIM: Probability and Statistics* 22, pp. 96–128.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim (2014). "Do we need hundreds of classifiers to solve real world classification problems?" In: *Journal of Machine Learning Research* 15.1, pp. 3133–3181.

A.A. Freitas (2014). "Comprehensible classification models: A position paper". In: *ACM SIGKDD Explorations Newsletter* 15, pp. 1–10.

J.H. Friedman and B.E. Popescu (2003). "Importance sampled learning ensembles". In: *Journal of Machine Learning Research* 94305.

J.H. Friedman and B.E. Popescu (2008). "Predictive learning via rule ensembles". In: *The Annals of Applied Statistics* 2, pp. 916–954.

R. Genuer (2012). "Variance reduction in purely random forests". In: *Journal of Nonparametric Statistics* 24, pp. 543–562.

R. Genuer and J.-M. Poggi (2019). *Les forêts aléatoires avec R*. Presses universitaires de Rennes.

R. Genuer, J.-M. Poggi, and C. Tuleau (2008). "Random Forests: some methodological insights". In: *arXiv preprint arXiv:0811.3619*.

R. Genuer, J.-M. Poggi, and C. Tuleau-Malot (2010). "Variable selection using Random Forests". In: *Pattern Recognition Letters* 31:2225-2236.

P. Geurts, D. Ernst, and L. Wehenkel (2006). "Extremely randomized trees". In: *Machine Learning* 63, pp. 3–42.

P. Geurts and L. Wehenkel (2005). "Closed-form dual perturb and combine for tree-based models". In: *Proceedings of the 22nd International Conference on Machine Learning*. New York: ACM, pp. 233–240.

L. Gondara and K. Wang (2018). "Mida: Multiple imputation using denoising autoencoders". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, p. 260.

I. Goodfellow, Y. Bengio, and A. Courville (2016). *Deep learning*. MIT press.

B. Gregorutti, B. Michel, and P. Saint-Pierre (2015). "Grouped variable importance with random forests and application to multiple functional data analysis". In: *Computational Statistics & Data Analysis* 90, pp. 15–35.

B. Gregorutti, B. Michel, and P. Saint-Pierre (2017). "Correlation and variable importance in random forests". In: *Statistics and Computing* 27.3, pp. 659–678.

L. Györfi, M. Kohler, A. Krzyżak, and H. Walk (2002). *A Distribution-Free Theory of Nonparametric Regression*. New York: Springer.

T. Hastie, R. Mazumder, J.D. Lee, and R. Zadeh (2015). "Matrix completion and low-rank SVD via fast alternating least squares". In: *Journal of Machine Learning Research* 16.1, pp. 3367–3402.

T. Hastie and R. Tibshirani (1986). "Generalized Additive Models". In: *Statistical Science* 1, pp. 297–310.

T. Hastie, R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. Springer New York Inc.

K. He, X. Zhang, S. Ren, and J. Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Z. He and W. Yu (2010). "Stable feature selection for biomarker discovery". In: *Computational Biology and Chemistry* 34, pp. 215–225.

T. Ho (1998). "The random subspace method for constructing decision forests". In: *Pattern Analysis and Machine Intelligence* 20, 832-844.

T. Hothorn and A. Zeileis (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R". In: *Journal of Machine Learning Research* 16, p. 3905.

Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi (2016). "Decision Forests, Convolutional Networks and the Models In-Between". In: *arXiv:1603.01250*.

H. Ishwaran (2007). "Variable importance in binary regression trees and forests". In: *Electronic Journal of Statistics* 1, pp. 519–537.

H. Ishwaran and U.B. Kogalur (2010). "Consistency of random survival forests". In: *Statistics & Probability Letters* 80, pp. 1056–1064.

H. Ishwaran, U.B. Kogalur, E.H. Blackstone, and M.S. Lauer (2008). "Random survival forests". In: *The annals of applied statistics* 2.3, pp. 841–860.

W. Jiang, J. Josse, and M. Lavielle (2019). "Logistic Regression with Missing Covariates–Parameter Estimation, Model Selection and Prediction". In: *Computational and Statistics Analysis*.

M.P. Jones (1996). "Indicator and stratification methods for missing explanatory variables in multiple linear regression". In: *Journal of the American statistical association* 91.433, pp. 222–230.

M.I. Jordan and R.A. Jacobs (1994). "Hierarchical mixtures of experts and the EM algorithm". In: *Neural Computation* 6, pp. 181–214.

J. Josse, N. Prost, E. Scornet, and G. Varoquaux (2019). "On the consistency of supervised learning with missing values". In: *arXiv:1902.06931, in revision in JMLR*.

J. Josse, S. Sardy, and S. Wager (2016). "denoiser: A package for low rank matrix estimation". In: *Journal of Statistical Software*.

D.P. Kingma and J. Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

J.M. Klusowski (2018). "Sharp analysis of a simple model for random forests". In: *arXiv preprint arXiv:1805.02587*.

J. Kruppa, A. Schwarz, G. Arminger, and A. Ziegler (2013). "Consumer credit risk: Individual probability estimates using machine learning". In: *Expert Systems with Applications* 40, pp. 5125–5131.

K. Lakshminarayan, S.A. Harp, R.P. Goldman, and T. Samad (1996). "Imputation of Missing Data Using Machine Learning Techniques." In: *KDD*, pp. 140–145.

B. Lakshminarayanan, D.M. Roy, and Y.W. Teh (2014). "Mondrian forests: Efficient online random forests". In: *Advances in neural information processing systems*, pp. 3140–3148.

B. Lakshminarayanan, D.M. Roy, and Y.W. Teh (2016). "Mondrian forests for large-scale regression when uncertainty matters". In: *Artificial Intelligence and Statistics*, pp. 1478–1487.

P. Latinne, O. Debeir, and C. Decaestecker (2001). "Limiting the number of trees in random forests". In: *Multiple Classifier Systems*. Berlin: Springer, pp. 178–187.

M. Le Morvan, N. Prost, J. Josse, E. Scornet, and G. Varoquaux (2020). "Linear predictor on linearly-generated data with missing values: non consistency and solutions". In: *AISTAT 2020*.

B. Letham (2015). "Statistical learning for decision making: Interpretability, uncertainty, and inference". PhD thesis. Massachusetts Institute of Technology.

B. Letham, C. Rudin, T.H. McCormick, and D. Madigan (2015). "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9, pp. 1350–1371.

Y. Li, S. Liu, J. Yang, and M.-H. Yang (2017). "Generative face completion". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3911–3919.

Z.C. Lipton (2016). "The mythos of model interpretability". In: *arXiv:1606.03490*.

R.J.A. Little (1992). "Regression with missing X's: a review". In: *Journal of the American Statistical Association* 87.420, pp. 1227–1237.

R.J.A. Little (1993). "Pattern-mixture models for multivariate incomplete data". In: *Journal of the American Statistical Association* 88.421, pp. 125–134.

R.J.A. Little and D.B. Rubin (2019). *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons.

Y. Liu, Y. Wang, Y. Feng, and M.M. Wall (2016). "Variable selection and prediction with incomplete high-dimensional data". In: *Annals of Applied Statistics* 10.1, pp. 418–450.

W.-Y. Loh (2011). "Classification and regression trees". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1, pp. 14–23.

G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts (2013). "Understanding variable importances in forests of randomized trees". In: *Advances in Neural Information Processing Systems*, pp. 431–439.

I. Mayer, J. Josse, N. Tierney, and N. Vialaneix (2019). "R-miss-tastic: a unified platform for missing values methods and workflows". In: *arXiv preprint arXiv:1908.04822*.

N. Meinshausen (2006). "Quantile regression forests". In: *Journal of Machine Learning Research* 7, pp. 983–999.

N. Meinshausen (2010). "Node harvest". In: *The Annals of Applied Statistics* 4, pp. 2049–2072.

L. Mentch and G. Hooker (2016). "Quantifying uncertainty in random forests via confidence intervals and hypothesis tests". In: *Journal of Machine Learning Research* 17, pp. 841–881.

B.H. Menze, B.M. Kelm, D.N. Splitthoff, U. Koethe, and F.A. Hamprecht (2011). "On oblique random forests". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 453–469.

K. Mohan and J. Pearl (2018). "Graphical models for processing missing data". In: *arXiv preprint arXiv:1801.03583*.

M. Le Morvan, J. Josse, T. Moreau, E. Scornet, and G. Varoquaux (2020). "Neumann networks: differential programming for supervised learning with missing values". In: *arXiv preprint arXiv:2007.01627*.

J. Mourtada, S. Gaiffas, and E. Scornet (2017). "Universal consistency and minimax rates for online Mondrian Forests". In: *Advances in Neural Information Processing Systems*, pp. 3758–3767.

J. Mourtada, S. Gaiffas, and E. Scornet (2019). "AMF: Aggregated Mondrian Forests for Online Learning". In: *arXiv:1906.10529, in revision in JRSSB*.

J. Mourtada, S. Gaıffas, and E. Scornet (2020). "Minimax optimal rates for Mondrian trees and forests". In: *Annals of Statistics* 48.4, pp. 2253–2276.

W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu (2019). "Interpretable machine learning: Definitions, methods, and applications". In: *arXiv:1901.04592*.

J.S. Murray (2018). "Multiple Imputation: A Review of Practical and Theoretical Findings". In: *Statist. Sci.* 33.2, pp. 142–159.

E.A. Nadaraya (1964). "On estimating regression". In: *Theory of Probability and Its Applications* 9, pp. 141–142.

A. Nemirovski (2000). "Topics in non-parametric". In: *Ecole d'Eté de Probabilités de Saint-Flour* 28, p. 85.

K.K. Nicodemus (2011). "Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures". In: *Briefings in bioinformatics* 12.4, pp. 369–373.

K.K. Nicodemus and J.D. Malley (2009). "Predictor correlation impacts machine learning algorithms: implications for genomic studies". In: *Bioinformatics* 25.15, pp. 1884–1890.

A. Nobel (1996). "Histogram regression estimation using data-dependent partitions". In: *The Annals of Statistics* 24, pp. 1084–1105.

C. Olaru and L. Wehenkel (2003). "A complete fuzzy decision tree technique". In: *Fuzzy Sets and Systems* 138, pp. 221–254.

P. Orbanz and D.M. Roy (2014). "Bayesian models of graphs, arrays and other exchangeable random structures". In: *IEEE transactions on pattern analysis and machine intelligence* 37.2, pp. 437–461.

R. Pascanu, G. Montufar, and Y. Bengio (2013). "On the number of response regions of deep feed forward networks with piece-wise linear activations". In: *arXiv:1312.6098*.

F. Pedregosa, G. Varoquaux, A. Gramfort, and V. Michel (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

K. Pelckmans, J. De Brabanter, J.A.K. Suykens, and B. De Moor (2005). "Handling missing values in support vector machine classifiers". In: *Neural Networks* 18.5-6, pp. 684–692.

A.M. Prasad, L.R. Iverson, and A. Liaw (2006). "Newer classification and regression tree techniques: Bagging and random forests for ecological prediction". In: *Ecosystems* 9, pp. 181–199.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

L. Raynal, J.-M. Marin, P. Pudlo, M. Ribatet, C.P. Robert, and A. Estoup (2019). "ABC random forests for Bayesian parameter inference". In: *Bioinformatics* 35.10, pp. 1720–1728.

M.T. Ribeiro, S. Singh, and C. Guestrin (2016). "Why should I trust you? Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM, pp. 1135–1144.

D.L. Richmond, D. Kainmueller, M.Y. Yang, E.W. Myers, and C. Rother (2015). "Relating cascaded random forests to deep convolutional neural networks for semantic segmentation". In: *arXiv:1507.07583*.

P.R. Rosenbaum and D.B. Rubin (1984). "Reducing bias in observational studies using subclassification on the propensity score". In: *Journal of the American statistical Association* 79.387, pp. 516–524.

D.M. Roy (2011). "Computability, inference and modeling in probabilistic programming". PhD thesis. Massachusetts Institute of Technology.

D.M. Roy and W.-T. Yee (2009). "The Mondrian Process". In: *Advances in Neural Information Processing Systems 21*, pp. 1377–1384.

D.B. Rubin (1976). "Inference and missing data". In: *Biometrika* 63.3, pp. 581–592.

D.B. Rubin (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley, p. 258.

S. Rüping (2006). "Learning interpretable models". PhD thesis. Universität Dortmund.

E. Scornet (2016a). "On the asymptotics of random forests". In: *Journal of Multivariate Analysis* 146, pp. 72–83.

E. Scornet (2016b). "Random forests and kernel methods". In: *IEEE Transactions on Information Theory* 62.3, pp. 1485–1500.

E. Scornet (2017). "Tuning parameters in random forests". In: *ESAIM: Proceedings and Surveys* 60, pp. 144–162.

E. Scornet (2020). "Trees, forests, and impurity-based variable importance". In: *arXiv:2001.04295*.

E. Scornet, G. Biau, and J.-P. Vert (2015a). "Consistency of random forests". In: *The Annals of Statistics* 43.4, pp. 1716–1741.

E. Scornet, G. Biau, and J.-P. Vert (2015b). "Supplementary materials for: Consistency of random forests". In: *The Annals of Statistics* 1510.

I.K. Sethi (1990). "Entropy nets: From decision trees to neural networks". In: *Proceedings of the IEEE* 78, pp. 1605–1613.

I.K. Sethi (1991). "Decision tree performance enhancement using an artificial neural network interpretation". In: *Artificial Neural Networks and Statistical Pattern Recognition*. Vol. 6912. Amsterdam: Elsevier, pp. 71–88.

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake (2011). "Real-time human pose recognition in parts from single depth images". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304.

D.J. Stekhoven and P. Bühlmann (2011). "MissForest—non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1, pp. 112–118.

C.J. Stone (1980). "Optimal rates of convergence for nonparametric estimators". In: *The Annals of Statistics* 8, pp. 1348–1360.

C.J. Stone (1982). "Optimal global rates of convergence for nonparametric regression". In: *The Annals of Statistics*, pp. 1040–1053.

C.J. Stone (1985). "Additive regression and other nonparametric models". In: *The Annals of Statistics*, pp. 689–705.

C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis (2008). "Conditional variable importance for random forests". In: *BMC Bioinformatics* 9, p. 307.

C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn (2007). "Bias in random forest variable importance measures: Illustrations, sources and a solution". In: *BMC bioinformatics* 8.1, p. 25.

C. Strobl, T. Hothorn, and A. Zeileis (2009). "Party on!" In:

C. Strobl and A. Zeileis (2008). "Danger: High power!–exploring the statistical properties of a test for random forest variable importance". In:

M. Sugiyama, N.D. Lawrence, and A. Schwaighofer (2017). *Dataset shift in machine learning*. The MIT Press.

A. Sutera, G. Louppe, V.A. Huynh-Thu, L. Wehenkel, and P. Geurts (2016). "Context-dependent feature analysis with random forests". In: *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*.

V. Svetnik, A. Liaw, C. Tong, J.C. Culberson, R.P. Sheridan, and B.P. Feuston (2003). "Random forest: A classification and regression tool for compound classification and QSAR modeling". In: *Journal of Chemical Information and Computer Sciences* 43, pp. 1947–1958.

T. Therneau and B. Atkinson (2018). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-13. URL: https://CRAN.R-project.org/package=rpart.

J. Tibshirani, S. Athey, and S. Wager (2020). *grf: Generalized Random Forests*. R package version 1.1.0. URL: https://CRAN.R-project.org/package=grf.

R. Tibshirani (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.

B.E.T.H. Twala, M.C. Jones, and D.J. Hand (2008). "Good Methods for Coping with Missing Data in Decision Trees". In: *Pattern Recognition Letters* 29.7, pp. 950–956.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008). "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 1096–1103.

S. Wager (2014). *Asymptotic Theory for Random Forests*.

S. Wager, T. Hastie, and B. Efron (2014). "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife". In: *The Journal of Machine Learning Research* 15, pp. 1625–1651.

S. Wager and G. Walther (2015). "Adaptive concentration of regression trees, with application to random forests". In: *arXiv:1503.06388*.

L. Wasserman (2006). *All of nonparametric statistics*. Springer Science & Business Media.

G.S. Watson (1964). "Smooth regression analysis". In: *Sankhyā Series A*, pp. 359–372.

J. Welbl (2014). "Casting random forests as artificial neural networks (and profiting from it)". In: *Pattern Recognition*. Ed. by X. Jiang, J. Hornegger, and R. Koch. Springer, pp. 765–771.

A.M. Wood, I.R. White, and P. Royston (2008). "How should variable selection be performed with multiply imputed data?" In: *Statistics in Medicine* 27.17, pp. 3227–3246.

M.N. Wright and A. Ziegler (2017). “ranger: A fast implementation of random forests for high dimensional data in C++ and R”. In: *Journal of Statistical Software* 77, pp. 1–17.

M.N. Wright, A. Ziegler, and I.R. König (2016). “Do little interactions get lost in dark random forests?” In: *BMC bioinformatics* 17.1, p. 145.

O.T. Yildiz and E. Alpaydin (2013). “Regularizing soft decision trees”. In: *Information Sciences and Systems 2013*. Cham: Springer, pp. 15–21.

J. Yoon, J. Jordon, and M. van der Schaar (2018). “GAIN: Missing Data Imputation using Generative Adversarial Nets”. In: *International Conference on Machine Learning*, pp. 5675–5684.

B. Yu (2013). “Stability”. In: *Bernoulli* 19, pp. 1484–1500.

B. Yu and K. Kumbier (2019). “Three principles of data science: Predictability, computability, and stability (PCS)”. In: *arXiv:1901.08152*.

S. Zhang, Z. Qin, C.X. Ling, and S. Sheng (2005). “" Missing is useful": missing values in cost-sensitive decision trees”. In: *IEEE transactions on knowledge and data engineering* 17.12, pp. 1689–1693.

Z.-H. Zhou and J. Feng (2017). “Deep forest”. In: *arXiv preprint arXiv:1702.08835*.

R. Zhu, D. Zeng, and M.R. Kosorok (2015). “Reinforcement Learning Trees”. In: *Journal of the American Statistical Association* 110.512, pp. 1770–1784.

M. Zucknick, S. Richardson, and E.A. Stronach (2008). “Comparing the characteristics of gene expression profiles derived by univariate and multivariate classification methods”. In: *Statistical Applications in Genetics and Molecular Biology* 7, pp. 1–34.