Mémoire présenté devant le jury de l'EURIA et de l'IMT Atlantique en vue de l'obtention du Diplôme d'Actuaire EURIA, du Diplôme d'Ingénieur IMT Atlantique et de l'admission à l'Institut des Actuaires

le 26 Septembre 2019

Par : Dimitri Delcaillau
Titre : Contrôle et Transparence des modèles complexes en actuariat

Confidentialité : Non

Les signataires s'engagent à respecter la confidentialité indiquée ci-dessus

Membres présents du jury de l'Institut des Actuaires :
Sophie Bourdet
Khalil Said
Signatures :

Entreprise :
MILLIMAN
Signature :

Membres présents du jury de l'EURIA ou de l'IMT Atlantique :
Daniel Boivin
Philippe Lenca

Directeur de mémoire en entreprise :
Antoine Ly
Signature :

Invité :

Signature :

Autorisation de publication et de mise en ligne sur un site de diffusion de documents actuariels
(après expiration de l'éventuel délai de confidentialité)

Signature du responsable entreprise :

Signature du candidat :

EURIA
EURo Institut
d'Actuariat

6, avenue le Gorgeu
CS 93837
29238 Brest Cedex 3

T +33 (0)2 98 01 66 55
euria@univ-brest.fr

Résumé

L'avènement du digital amène les industries à utiliser de manière systématique les données qu'elles collectent. A ces données, parfois d'un format non structuré : images, textes ou son, sont associées de nouvelles méthodologies : réseaux convolutifs, récurrents, méthodes par arbres agrégés, etc.

Souvent associées à des « boîtes-noires » ces méthodes questionnent sur leur transparence et leur interprétation. Plus particulièrement, en assurance – et même si leur utilisation reste modérée en France – le contexte réglementaire amène à s'interroger sur ces algorithmes afin de comprendre leurs risques et ainsi mieux les maîtriser.

Ainsi, dans ce contexte, les travaux porteront sur l'exploration de certains de ces algorithmes – utilisés en assurance - et la tentative de les expliquer tant au niveau de leurs résultats que sur la méthode qui les produit.

Mots clefs: Machine Learning, Interprétabilité, Explicabilité, Transparence, LIME, SHAP, GLM, XGBoost, Tarification Automobile, Prime Pure

Abstract

The digital revolution leads many industries and companies to exploit their data in an automated and systematic way. To deal with these data, often available in a non-structured format (images, texts, sound data), new methodologies exist : convolutional and recurrent neural networks, boosting trees etc.

Because these methods are often seen as "black boxes", questions about their transparency and interpretability arise. Particularly in the insurance field, the regulations require that actuaries understand and monitor the risks associated to these algorithms – even though their use is still moderated in France.

In this context, the study will focus on the exploration of specific algorithms used in insurance and the attempt to interpret them not only in terms of results, but also methodologies and implementation.

Keywords: Interpretability, Explicability, Transparency, XAI (Explainable Articial Intelligence), LIME, SHAP, IML (Interpretable Machine Learning), XGBoost, Pricing, Car Insurance, Pure Premium

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

En premier lieu, j'aimerais remercier Milliman Paris de m'avoir offert l'opportunité de réaliser ce stage de fin d'études.

Je remercie mon tuteur Antoine Ly d'avoir proposé ce sujet de mémoire et également pour la qualité de son encadrement tout au long du stage. Les nombreux points et échanges m'ont été d'une aide précieuse.

Je remercie l'équipe Non Vie et Analytics dans laquelle j'ai eu la chance d'évoluer, et tout particulièrement Rémi Bellina pour ses nombreux conseils et sa disponibilité.

Je remercie les enseignants de l'IMT Atlantique et de l'EURIA pour la qualité des cours dispensés au cours de mes études supérieures.

En dernier lieu, je tiens à remercier, pour son implication, mon tuteur Frank Vermet, directeur de l'EURIA.

A la mémoire de Dominique.

Note de synthèse

Contexte

Dans un contexte où les industries collectent de plus en plus de données, de nouvelles méthodes sont utilisées pour tirer profit au maximum des informations extraites. Des algorithmes comme les réseaux de neurones, forêts aléatoires ou les SVM (Support Vector Machine) sont désormais largement utilisés dans de nombreux domaines, y compris en actuariat. Ces modèles apportent, dans de nombreuses situations comme en tarification automobile, un apport non négligeable en terme de performance. Ce gain d'efficacité a un coût : il se fait au détriment de la lisibilité et de la transparence. Néanmoins, la compréhension du modèle demeure vitale dans de nombreux cas. Tout d'abord pour des raisons règlementaires, avec l'Autorité de Contrôle Prudentiel et de Résolution (ACPR), dans le monde de l'assurance, ou encore le Règlement Général sur la Protection des Données (RGPD), de manière plus générale, qui obligent de justifier les décisions prises par les algorithmes utilisés et refusent le transfert de la responsabilité sur la machine. L'interprétabilité est également essentielle pour pouvoir expliquer les résultats fournis aux intervenants concernés et pour générer de la confiance dans les outils utilisés. On peut enfin citer l'aspect éthique qui est indissociable de la transparence. Ainsi, de nombreuses raisons conduisent les acteurs à un désir d'une plus grande transparence des modèles utilisés : on ne les juge plus seulement par leur capacité à obtenir un taux d'erreur faible, mais également sur leur faculté à pouvoir s'expliquer de manière simple à un humain. Cette volonté de compréhension et d'interprétabilité des modèles a conduit à une prolifération des articles de recherches sur le sujet du machine learning interpretable (IML). De nombreuses méthodes sont alors proposées et nous en décrirons quelques unes qui semblent pertinentes. Nous nous sommes restreints aux méthodes d'interprétation post-hoc, i.e. une fois que le modèle a été ajusté, et agnostiques au modèle, i.e. indépendantes de l'algorithme à expliquer. Le schéma 1 résume les différentes catégories de l'interprétabilité et permet de comprendre où se situe notre étude.

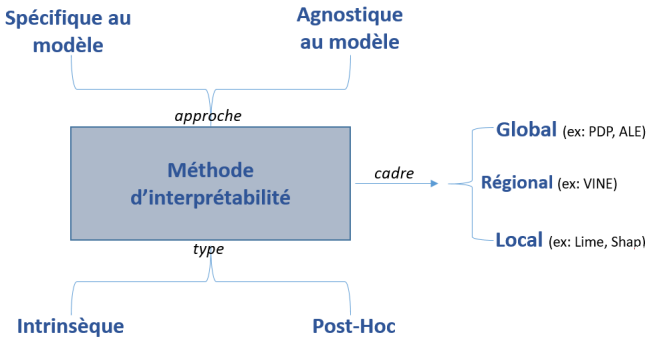

FIGURE *1: Différentes catégories d'interprétabilité des modèles*

Résumé de certaines méthodes d'interprétation

Résumons différentes méthodes d'interprétation qui ont été abordées au cours de ce mémoire. Notons que pour une explication plus complète, nous conseillons de se référer au chapitre correspondant.

PDP : graphique de dépendance partielle

Présentons tout d'abord une des méthodes les plus répandues, à savoir le graphique de dépendance partielle (PDP). Celui permet de montrer l'effet marginal d'une ou plusieurs variables explicatives (généralement pas plus de deux) sur la prédiction faite par un modèle. Si l'on note S l'ensemble des variables que l'on souhaite étudier, et ces C l'ensemble des autres variables explicatives du modèle, le graphique PDP se définit ainsi :

$$\hat{f}_{x_S}(x_S) = \mathbb{E}_{X_c}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_s, x_c) d\mathbb{P}_{X_c}(x_c) \tag{1}$$

Afin d'estimer cette dernière, nous nous basons sur les n observations à notre disposition et de la méthode de Monte Carlo :

$$\hat{f}_{x_S}(x_S) \simeq \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}) \tag{2}$$

Cette méthode consiste à moyenner les prédictions faites par le modèle lorsque l'on fixe les variables de l'ensemble S. Il en découle les avantages suivants de cette méthode, à savoir la simplicité d'interprétation des résultats, une facilité d'implémentation ainsi qu'une adaptation possible pour les variables catégorielles. Néanmoins, cette méthode présente la limite de ne fonctionner uniquement pour des variables indépendantes. Dans le cas contraire, au cours du calcul de la moyenne réalisée dans le PDP, on aboutirait à des points non désirés ou à des combinaisons de variables explicatives qui ne sont pas susceptibles de se produire. C'est pourquoi il est plus pertinent en pratique d'utiliser la méthode ALE, bien que peu répandue dans la littérature, qui est une adaptation du

PDP visant à corriger cette problématique de corrélations entre les différentes variables. Un autre problème inhérent aux méthodes globales d'interprétation, comme le PDP et l'ALE, est le fait de masquer des effets hétérogènes entre les variables, à cause de la moyenne réalisée. Une autre adaptation de la PDP, appelées courbes ICE, permettent de mettre en exergue ces effets hétérogènes, mais est plus coûteuse en terme de calcul. Pour une étude plus approfondie et une meilleur compréhension du modèle étudié, les méthodes d'interprétation locales, présentées ci-après, sont nécessaires.

LIME : Explications locales interprétables, agnostiques au modèle

La méthode LIME est l'une des premières approches locales apparues dans le domaine du machine learning interprétable et est à présent largement répandue. L'objectif de cette méthode est d'approcher le comportement local du modèle complexe étudié par un modèle plus simple et parcimonieux comme une régression linéaire par exemple. Ainsi, au niveau d'une instance donnée, la boîte-noire sera approximée par un modèle linéaire, et bénéficiera d'une interprétation aisée, notamment via les coefficients de la régression. Dans le cadre d'une application en tarification automobile, si le tarif est calculé via un modèle complexe, il serait alors possible via la méthode LIME de justifier la prédiction pour un assuré donné, au même titre qu'un GLM, à l'aide des coefficients. Cette méthode LIME présente l'avantage de fournir une explication concise et claire, via l'utilisation de régression linéaire de type Lasso par exemple. Elle est de plus, simple à mettre en ouvre, peu coûteuse en temps de calcul et fonctionne à la fois pour des données sous formes de tableaux, de textes ou d'images. L'un des inconvénients majeurs, comme le montrent différents articles, est l'instabilité des explications fournies dans certaines situations. Des méthodes alternatives sont proposées pour pallier ce problème et le sujet est en constante évolution.

SHAP

Une autre méthode locale couramment utilisée aujourd'hui pour l'interprétation des modèles de machine learning est SHAP. Quand un modèle réalise une prédiction, intuitivement nous savons que chaque variable ne joue pas le même rôle et que certaines n'ont quasiment aucun impact tandis que d'autres en ont un grand sur la décision prise par le modèle. L'objectif de SHAP est justement de quantifier le rôle de chaque variable dans la décision finale du modèle. Cette méthode repose sur la valeur de Shapley, issue de la théorie des jeux. Elle permet de donner la contribution de chaque variable dans la prédiction faite par un modèle pour une instance donnée, en attribuant un score juste, au sens mathématique, grâce à différentes propriétés vérifiées par la valeur de Shapley.

Au même titre que Lime, SHAP fournit des scores d'importance aux différentes variables explicatives, pour une instance donnée. Néanmoins l'interprétation résultante de ces scores est différente notamment car SHAP repose sur une notion de contribution. Ainsi, SHAP, en plus d'être simple d'interprétation, a l'avantage de posséder des propriétés de contributions justes, et repose sur une vraie théorique mathématique. Celle-ci semble donc pouvoir s'inscrire dans le cadre du "droit à l'explication" présent dans l'ar-

ticle 22 du RGPD, et son utilisation semble plus facilement justifiable que celle de LIME. Cependant, la formule théoriques des valeurs de Shapley ne permet pas, en pratique, son calcul exact, car trop coûteux en ressource. Il en résulte, comme le souligne différents articles, une instabilité due à l'échantillonnage réalisé. De plus, contrairement à Lime, la parcimonie ne peut être vérifiée, car SHAP nécessite l'utilisation de toutes les variables.

Application à la tarification automobile

Mise en place des modèles de calcul de la prime pure et analyse des résultats obtenus

Le coeur de notre sujet a été la mise en pratique de ces différentes méthodes d'interprétation sur une base de données réelle de tarification automobile, associée à la responsabilité civile moteur contenant plus de 400 000 polices. L'objectif est, à partir des variables explicatives comme la région, la puissance de la voiture ou encore l'âge du conducteur, de prédire le montant des sinistres pour pouvoir donner une estimation de la prime pure. La particularité des données de tarification est qu'elles sont très déséquilibrées, avec de nombreux zéros et une exposition qui varie du côté de la fréquence combinée avec des sinistres, rares, mais à queue lourde.

L'approche actuarielle classique est une modélisation indépendante de la fréquence (nombre de sinistres annuels) et de la sévérité (coût moyen d'un sinistre), pour former la prime pure en multipliant les prédictions de ces deux modèles. Avant de mettre en place les algorithmes répondant à ce problème, les traitements préliminaires, tentant de répliquer les pratiques opérationnelles, ont été effectués, comme l'analyse des valeurs aberrantes, des sinistres extrêmes (qui ont été écrêtés) et des retraitements en rendant toutes les variables catégorielles. Cette dernière étape est essentielle pour la mise en place d'un modèle linéaire généralisé (GLM), car sans retraiter les données, une monotonie est imposée pour les variables numériques de par la nature linéaire du modèle. Notre objectif dans cette partie était de mettre en place le modèle GLM classique, très souvent utilisé en actuariat, ainsi qu'un autre modèle, plus complexe donnant éventuellement des meilleures performances. Une fois cette étape réalisée, nous voulions montrer à l'aide des outils d'interprétation détaillés ci-avant qu'il était possible de comprendre le modèle de boîte noire implémenté et qu'il n'était pas nécessairement moins interprétable que le modèle GLM. Nous avons finalement opté pour le modèle eXtrem Gradient Boosting (XGBoost), pour lequel nous avons détaillé la théorie sous-jacente. A partir de validations croisées, nous avons cherché les paramètres optimaux du XGBoost. Ces derniers sont assez nombreux et représentent un des avantages du XGBoost et expliquent sa popularité dans de nombreuses compétitions de machine learning, comme Kaggle. Les résultats obtenus ne sont pas très satisfaisants, lorsque l'on se fie au critère de MSE que nous avons choisi comme métrique d'évaluation. En effet, les gains obtenus en fréquence et en coût sont proches de 0.1 %, alors que sur la modélisation totale (coût x fréquence), le gain est de l'ordre de 0.001 % en comparaison du meilleur modèle GLM. Les résultats sont indiqués dans le tableau 1.

	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
XGBoost total	13430342	288.442	3664.743	24.03593	0.0062510
GLM trivial total	13431960	278.8234	3664.964	24.03738	0.0062515
"meilleur" GLM total	13430428	281.2167	3664.755	24.0360	0.0062511
XGBoost coût	108541340	1936.062	10418.32	5.115432	0.034578
GLM trivial coût	108617958	1941.673	10421.99	5.117237	0.0345902
'meilleur' GLM coût	108565294	1938.289	10419.47	5.1160	0.034582
GLM trivial fréquence	0.1845885	0.1315303	0.4296377	6.011972	0.035803
"meilleur" GLM fréquence	0.1841617	0.1323143	0.429141	6.005016	0.035762
XGBoost fréquence	0.1840609	0.1361526	0.429023	6.003373	0.035752

TABLE *1: Résultats du XGBoost total et comparaison avec ceux du GLM*

En plus d'un gain très faible en MSE, notre approche fournit une perte au niveau du MAE proche de 0.5%. Nous avons conclu que cela vient du fait que notre métrique choisie pour l'optimisation est quadratique, comme le MSE, et n'est donc pas destinée à minimiser un critère comme le MAE. Il semblerait qu'obtenir à la fois un meilleur MSE et un meilleur MAE (que le GLM) sur nos données ne soit pas possible au vu de l'étude réalisée. A la lumière de ces résultats peu convaincants, nous avons souligné que ce problème est très courant avec les bases de données actuarielles, pour lesquels les modèles triviaux sont parfois équivalents voir meilleurs que des modèles complexes. De nombreuses études montrent qu'il est possible d'obtenir de bien meilleures performances, en tarification par exemple, avec des modèles comme le XGBoost ; cependant, il semblerait que nos données étudiées ne permettent pas d'obtenir un gain très élevé, sûrement dû à un manque de corrélation entre les variables explicatives et la variable cible. Étant données les performances peu satisfaisantes du XGBoost, nous avons remis en question le fait d'utiliser un unique critère pour juger de la qualité d'un modèle. En particulier, le fait d'avoir un gain en MSE minime, proche de 0.001 %, suffit-il pour conclure que le modèle XGBoost n'est pas satisfaisant pour notre étude ? Nous avons introduit des MSE dit "locaux", qui ne sont plus calculés sur la base de test entière mais sur certains assurés respectant des conditions définies. Quelques résultats sont donnés dans le tableau 2.

Condition	Nombre d'assurés concernés (Bt)	RMSE - GLM trivial	RMSE - GLM Best	RMSE - XGBoost
ClaimAmount = 0	59654 (96.3%)	**136.3**	148	153.9
ClaimAmount>0	2322 (3.7%)	18922	18918.4	**18917.1**
ClaimAmount >0 et ClaimAmount <10K	2278 (3.7%)	2652.3	2635.2	**2621.4**
ClaimAmount >10K	44 (0.071%)	86511.6	86502.75	**86500.75**

TABLE *2: RMSE locaux calculés sur la base de test pour comparer les modèles GLM et XGBoost qui modélisent la prime pure*

Nous restons lucides sur le fait que notre modèle de boîte noire semble être très proche du GLM classique et ne serait certainement pas déployé en pratique, mais nous trouvions intéressant d'introduire de nouveaux outils pour aider dans la prise de décision concernant le choix d'un modèle.

Interprétation des modèles GLM et XGBoost

Une fois les performances des deux modèles étudiées, notamment avec une analyse de la stabilité vis-à-vis de l'échantillonnage de la base de test et d'apprentissage, nous avons mis en pratique les méthodes d'interprétation développées précédemment, pour mieux comprendre les prédictions réalisées. Nous avons dans un premier temps analysé les prédictions du modèle GLM. Les propriétés de parcimonie et de simulabilité (détaillées dans ce mémoire) nous ont permis de comprendre directement le modèle à partir des coefficients de chaque variable. En particulier, nous avons décomposé le cheminement qui mène à une telle prédiction pour un assuré et nous avons également étudié le changement de prédiction lorsque les caractéristiques d'un assuré sont modifiées. Toute cette étude intrinsèque au GLM a été complétée par l'analyse utilisant les outils d'interprétation agnostiques au modèle. Nous avons montré à l'aide d'une analyse globale (PDP, ALE, importance des variables) et d'une analyse locale (courbes ICE, LIME, Shap) la cohérence des explications fournies. Dans un second temps, nous avons réalisé la même étude cette fois-ci sur le modèle XGBoost. Sa nature complexe, associée à une boîte noire, rend l'interprétation difficile et l'analyse intrinsèque, comme pour le GLM, n'est pas possible : nous avons donc recours aux outils d'interprétabilité des modèles. Nous avons pu, comme pour le GLM qui utilise la t-statistique, mesurer l'importance globale des variables à l'aide de la méthode PFI, basée sur les permutations (c.f figure 2). L'idée derrière cette méthode est de considérer qu'une variable est d'autant plus importante que l'erreur de prédiction du modèle augmente après avoir permuté les valeurs de cette variable, signe de la sensibilité du modèle à cette variable et donc de son importance. Cette mesure a été complétée par une analyse de la stabilité des résultats, en réalisant plusieurs simulations.

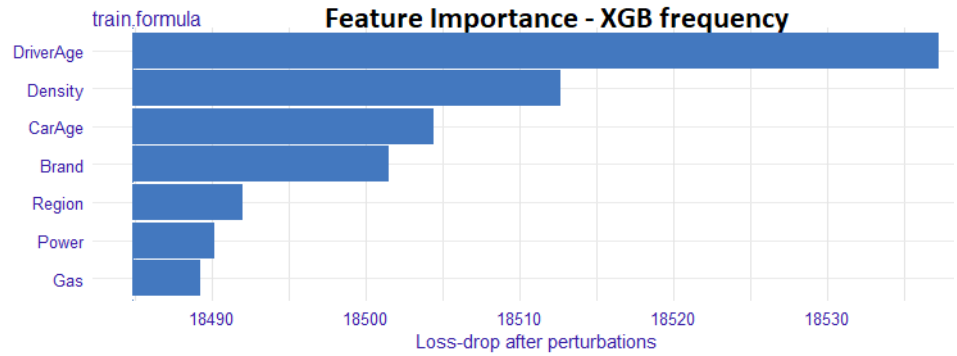

FIGURE *2: Importance des variables dans le modèle XGBoost fréquence, calculée à l'aide de la méthode PFI (Permutation Feature Importance)*

Ensuite, nous avons pu comprendre l'effet marginal moyen d'une variable sur la réponse fournie par le modèle, à l'aide des graphiques PDP et ALE. Ces deux derniers donnent des résultats très similaires, ce qui est cohérent avec le fait que les variables utilisées soient très peu corrélées. Le graphique des effets locaux accumulés, pour le XGBoost en fréquence, est représenté sur la figure 3. Une valeur positive d'une modalité

d'une variable sur cette courbe d'ALE signifie que la prédiction sur cette modalité est plus élevée en moyenne que la prédiction moyenne totale. C'est par exemple le cas pour la classe *18-25* de la variable *DriverAge*, indiquant qu'en moyenne les jeunes conducteurs ont plus d'accidents.

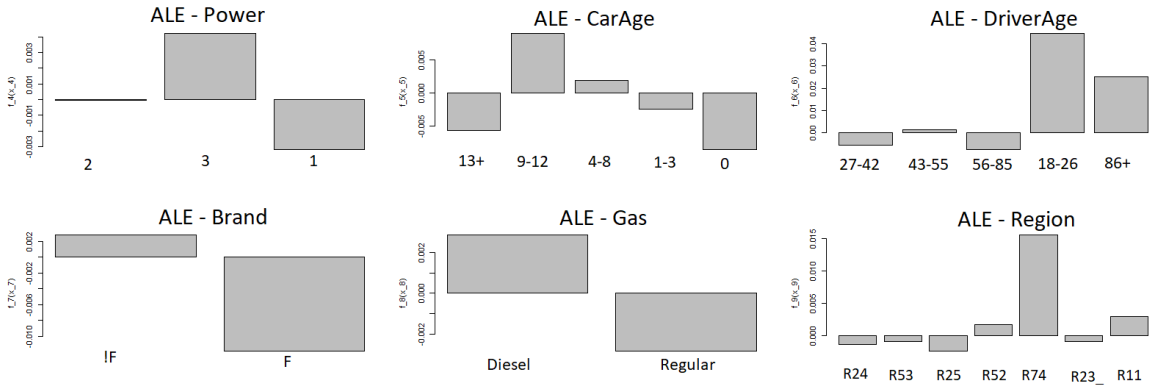

FIGURE *3: Graphique d'ALE (effets locaux accumulés) pour les 7 variables utilisées dans le modèle XGBoost fréquence*

A la différence du GLM pour lequel aucune interaction n'existe entre les variables, le XGBoost présente de nombreux effets hétérogènes. Ces interactions peuvent être détectées à l'aide de plusieurs outils développés avant, à savoir la H-statistique, les courbes ICE ou les graphiques de PDP (ou d'ALE) combinant deux variables. Il s'avère que toutes ces méthodes sont associées à un temps de calcul important, surtout la H-statistique. Nous avons conclu que lorsque les variables utilisées sont toutes catégorielles, l'usage de la H-statistique n'est pas pertinent. Nous avons finalement opté pour afficher quelques courbes ICE (c.f graphique 4), qui montrent des courbes non translatées, signe d'une hétérogénéité. Cette première analyse a permis de comprendre l'effet global d'une variable sur la prédiction du modèle XGBoost et également de mettre en exergue les interactions entre les différentes variables. Nous avons en particulier identifié l'effet combiné du couple de variables $(DriverAge, Power)$, pour laquelle le fait d'être à la fois jeune et d'avoir un véhicule puissant augmente considérablement le risque de sinistralité. Celui-ci s'observe sur la figure 4, avec une pentification plus importante entre les modalités 1826 et 2742 pour les courbes bleues relatives aux voitures puissantes, que pour le graphique de dépendance partielle en noir.

Pour mieux comprendre une prédiction donnée, nous avons utilisé les outils locaux, à savoir LIME et Shap. Nous avons complété cette étude avec l'analyse de la stabilité des interprétations fournies. Nous sommes arrivés à la conclusion que, pour des bases de données volumineuses, la méthode SHAP était moins appropriée. En effet, elle nécessite d'utiliser un échantillon de taille relativement faible si l'on veut des temps de calcul raisonnables, ce qui engendre une forte instabilité des explications fournies. Ceci ne semble pas être le cas pour LIME, pour laquelle les résultats sont obtenus bien plus rapidement et semblent également beaucoup plus stables, comme nous pouvons le voir sur la figure

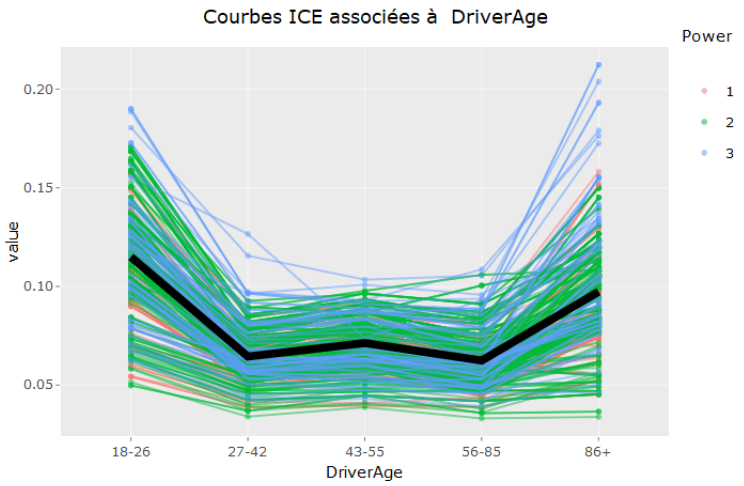

FIGURE *4: Graphique de PDP et courbes ICE obtenus pour le modèle XGBoost fréquence, associés à la variable DriverAge*

qui suit. Cet effet est d'autant plus marqué pour la variable *CarAge* avec la méthode SHAP pour laquelle la boîte à moustache est très étendue, signe d'une grande volatilité.

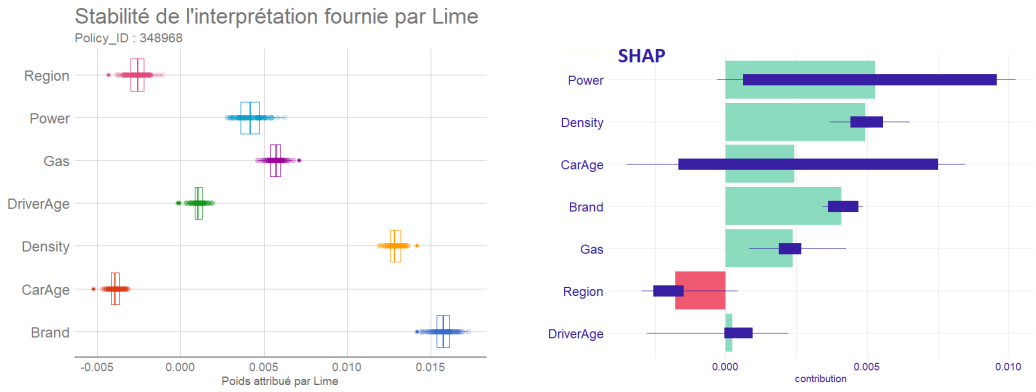

FIGURE *5: Boîtes à moustaches des résultats fournis par LIME et Shap. A droite : poids attribué à chaque variable par LIME pour la prédiction faîte pour un assuré (appelé assuré 2) par le modèle XGBoost fréquence, basé sur 100 simulations différentes. A Gauche : la contribution attribuée à chaque variable par la méthode Shap pour la prédiction faîte pour un assuré (appelé assuré 2) par le modèle XGBoost fréquence, basée sur 25 simulations différentes*

Assureur	Nombre de points utilisés par simulation	Nombre de simulations	Temps d'exécution (secondes)
SHAP	100	25	46
LIME	5000	100	14

TABLE *3: Temps d'exécution des méthodes LIME et Shap*

Conclusion

L'interprétabilité des modèles de machine learning est un sujet d'actualité qui devient peu à peu incontournable pour la majorité des data-scientists mais également des actuaires. Que ce soit pour des raisons éthiques - en s'assurant que le sexe ou l'origine ethnique ne soient pas des variables discriminantes par exemple -, réglementaires - notamment avec l'ACPR et le RGPD qui restent vigilants sur l'utilisation de boîtes noires - ou encore pour générer une plus grande confiance dans les outils utilisés, la question de la transparence et de la compréhension des modèles est aujourd'hui vitale.

De nombreuses méthodes d'interprétation existent pour comprendre le comportement des algorithmes qualifiés de boîtes noires, il convient alors de choisir les outils adaptés à l'étude réalisée. On peut citer les méthodes globales, comme le graphique de dépendance partielle (PDP) ou l'importance des variables et les méthodes locales, comme LIME et SHAP, qui sont les plus utilisées à l'heure actuelle. De nombreuses limites subsistent pour chacune de ces méthodes, comme l'incertitude des interprétations fournies qui demeure la plus problématique. Le sujet étant en constante évolution, des corrections et des adaptions de ces méthodes ne cessent d'être proposées, pour remédier à ces différentes limites. On peut notamment citer l'ALE qui permet de résoudre la problématique de corrélation des variables rencontrée avec le PDP.

A travers cette application en actuariat, et plus précisément en tarification automobile, nous avons montré la possibilité d'améliorer sensiblement les performances des modèles triviaux sur des données réelles, en utilisant des modèles complexes comme le XGBoost par exemple.

A ce gain de performance, est généralement associé une baisse de la transparence des résultats. Cependant, l'application de ces différentes méthodes d'interprétation ont permis de montrer qu'un modèle d'apprentissage, aussi complexe soit-il, peut s'interpréter à l'aide de ces outils et ainsi devenir « transparent » au même titre qu'un GLM. Il semble ainsi possible d'utiliser de tels modèles complexes pour de nombreux sujets actuariels, comme la tarification automobile, tout en respectant les contraintes réglementaires imposées par le RGPD et l'ACPR et en s'assurant d'une totale transparence.

Les recherches futures sur le thème du machine learning interprétable (IML) semble se diriger vers un formalisme plus clair, avec notamment des définitions non ambiguës des termes utilisés, un consensus général sur un cadre concernant la manière d'interpréter un modèle de machine learning et enfin sur la définition d'une métrique de qualité de ces explications, notamment par le biais de mesures de robustesse.

Summary

Context

In a context in which companies collect more and more data, new methods are used to take advantage of these extracted informations. Algorithms like neural networs, random forests or Support Vector Machine (SVM) are now widespread in many fields, including actuarial science. These models provide, in many situations, such as insurance car pricing, a significant contribution in terms of performance. This gain in efficiency has a cost : it is at the expense of readability and transparency. However, in many cases, understanding the model remains vital. First of all for regulatory reasons, with the french supervisory authority (ACPR), in the world of insurance, or the General Regulations on the Protection of Data (GDPR), more generally, which require the justification of decisions taken by the algorithms used and refuse the transfer of responsibility on the machine. Interpretability is also essential to explain the results provided to stakeholders and to build trust in the tools used. We can finally mention the ethical aspect which is inseparable from the transparency. Thus, many reasons lead actors to a desire for greater transparency of models used : they are no longer judged solely by their ability to get a high accuracy, but also on their ability to be simply explained to a human. This desire for understanding and interpretability of models led a proliferation of research articles on the subject of interpretable machine learning (IML). Many methods are then proposed and we describe some that seem relevant. We restricted ourselves to post-hoc interpretation methods, i.e. once the model has been fitted, and agnostic to the model, i.e. independent of the algorithm to be explained. Figure 6 summarizes the different categories of interpretability and allows us to understand where our study is.

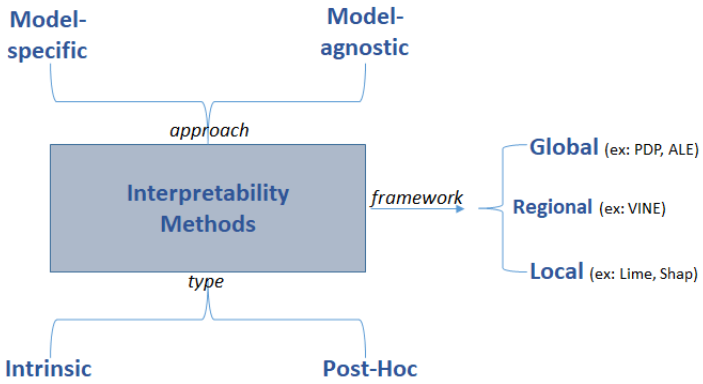

FIGURE *6: Categories of the interpretability methods*

Summary of few model-agnostic interpretation methods

In this part, we sumarise different interpretation methods which will be developed more precisely in this master thesis, with the chapter 4.

PDP : partial dependency plot

Let us first present one of the most widespread methods, the Partial Dependency Graph (PDP). The PDP allows us to show the marginal effect of one or more explanatory variables (usually no more than two) on the prediction made by a model. If one notes S the set of variables one wishes to study, and these C the set of other explanatory variables of the model, the *PDP* graph is defined as follows :

$$\hat{f}_{x_S}(x_S) = \mathbb{E}_{X_c}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_s, x_c) d\mathbb{P}_{X_c}(x_c) \tag{3}$$

In order to estimate the PDP, we use the n observations at our disposal and the Monte Carlo method :

$$\hat{f}_{x_S}(x_S) \simeq \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}) \tag{4}$$

This method consists of averaging the predictions made by the model when setting the variables in the S set. The following advantages of this method result : simplicity of interpretation of the results, ease of implementation, and possible adaptation for categorical variables. Nevertheless, this method has the limitation of working only for independent variables. Otherwise, during the averaging performed in the PDP, one would end up with unwanted points or combinations of explanatory variables that are unlikely to occur. This is why it is more relevant in practice to use the ALE method, although not widely used in the literature, which is an adaptation of the PDP aimed at correcting this problem of correlations between the different variables. Another inherent problem of global methods of interpretation, such as PDP and ALE, is the masking of heterogeneous effects between

variables, because of the mean that is produced. Another adaptation of the PDP, called ICE curves, allows to highlight these heterogeneous effects, but is more costly in terms of calculation. For a more in-depth study and a better understanding of the model under study, local interpretation methods, presented below, are necessary.

LIME : Interpretable local explanations, agnostic to the model

The LIME method is one of the first local approaches that appeared in the field of interpretable machine learning and is now widely used. The objective of this method is to approach the local behavior of the complex model studied by a simpler and parsimonious model such as a linear regression for example. Thus, at the level of a given instance, the black box will be approximated by a linear model, and will benefit from an easy interpretation, in particular via the regression coefficients. In the context of an application in automobile pricing, if the tariff is calculated using a complex model, it would then be possible to justify the prediction for a given insured, in the same way as a GLM, using the coefficients, using the LIME method. This LIME method has the advantage of providing a concise and clear explanation, using Lasso linear regression for example. Moreover, it is simple to implement, inexpensive in computing time and works for data in the form of tables, text or images. One of the major drawbacks, as shown in various articles, is the instability of the explanations provided in certain situations. Alternative methods are proposed to overcome this problem and the subject is constantly evolving.

SHAP

Another local method commonly used today for the interpretation of machine learning models is SHAP. When a model makes a prediction, intuitively we know that each variable does not play the same role and that some variables have almost no impact while others have a big impact on the decision made by the model. The objective of SHAP is precisely to quantify the role of each variable in the final decision of the model. This method is based on Shapley's value from game theory. It makes it possible to give the contribution of each variable in the prediction made by a model for a given instance, by attributing a fair score, in the mathematical sense, thanks to different properties that the Shapley value verifies.

Like LIME, SHAP provides importance scores to the different explanatory variables for a given instance. Nevertheless the resulting interpretation of these scores is different, because unlike LIME, SHAP's scores are based on a notion of contribution. SHAP, in addition to being simple to interpret, has the advantage of having properties of fair contributions, and is based on a true contribution theory.

Case study : Fitting models to compute the pure premium and analysis of the results

The main part of this study was the implementation of these different methods of interpretation on a real database of french insurance car pricing, associated with motor third party liability (MTPL) porfolio, containing more than 400 000 rows. The goal is, from features such as the region, the power of the car or the age of the driver, to predict the amount of the claims in order to give an estimate of the pure premium. The specific characteristics of insurance data is very unbalanced count data, with many zeros and a varying exposure on the frequency side combined with rare but heavy-tailed claims, on the severity side. The classical approach is an independent modeling of the frequency (number of annual claims) and the severity (average cost of a claim), to estimate the pure premium by multiplying the predictions of these two models. Before implementing the algorithms to solve this problem, preliminary processing was done, such as analysis of outliers or extreme claims (which were capped) and reprocessing by making all variables categorical. This last step is necessary to build the Generalized Linear Model (GLM), because a constrainst of monotony (growth or decay) is imposed on numerical variables because of the linear nature of this model. The purpose of this section was to fit the classical GLM, widely used in actuarial science, as well as an another more complex model, giving eventually better results. Once this step was done, we wanted to show, by using the interpretation tools detailed above, that it was possible to understand the black box fitted and it was not necessarily less interpretable than the GLM. We finally chose the eXtrem Gradient Boosting (XGBoost) algorithm, for which we detailed the underlying theory. Based on cross validations, we looked for the optimal parameters of the XGBoost. These are numerous which is one of the advantages of the XGBoost model and explain its popularity in many machine learning competitions, such as Kaggle. The performance results are not really satisfying, when we look à the criterion of MSE (Mean Square Error), which was chosen as evaluation metric. In fact, the relative gain in both frequency and severity modeling is near from 0.1 %, while the resulting model ($frequency \times severity$) is about 0.001% better than the GLM. The main results are showed in the table 4

	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
total XGBoost	13430342	288.442	3664.743	24.03593	0.0062510
trivial total GLM	13431960	278.8234	3664.964	24.03738	0.0062515
'best' GLM total	13430428	281.2167	3664.755	24.0360	0.0062511
XGBoost severity	108541340	1936.062	10418.32	5.115432	0.034578
trivial severity GLM	108617958	1941.673	10421.99	5.117237	0.0345902
'best' GLM coût	108565294	1938.289	10419.47	5.1160	0.034582
trivial frequency GLM	0.1845885	0.1315303	0.4296377	6.011972	0.035803
best frequency GLM	0.1841617	0.1323143	0.429141	6.005016	0.035762
XGBoost frequency	0.1840609	0.1361526	0.429023	6.003373	0.035752

TABLE *4: Results of the total XGBoost ($frequency \times severity$) and comparison with the best GLM*

In addition to a very low gain in terms of MSE, the XGBoost approach provides a loss of about 0.5% in terms of MAE (Mean Absolute Error). we concluded that it was due to the quadratic form of the metric chosen for optimization, just like the MSE, and so not intended to minimize the MAE criterion. Our study show that it seems difficult to obtain both a better MSE and a better MAE (than the GLM) on those data. in the light of the study. In the light of these unconvincing results, we have pointed out that this problem is very common with actuarial databases, for which trivial models are sometimes equivalent or even better than complex models. Numerous studies show that it is possible to get much better performance, for example in insurance pricing, with models such as the XGBoost ; however, it would appear that our studied data do not allow to get a high gain, probably due to a lack of correlation between the features and the target. In view of the unsatisfactory performance of the XGBoost, we questioned the fact of using a single criterion to judge the quality of a model. In particular, does having a minimal MSE gain, close to 0.001%, is enough to conclude that the XGBoost model is not satisfactory for our study ? We have introduced so-called "local" MSEs, which are no longer computed on the entire test base, but on chosen insureds group. Some results are given in the table 5.

Condition	Number of concerned insured people	RMSE - GLM trivial	RMSE - Best GLM	RMSE - XGBoost
ClaimAmount = 0	59654 (96.3%)	**136.3**	148	153.9
ClaimAmount>0	2322 (3.7%)	18922	18918.4	**18917.1**
ClaimAmount >0 and ClaimAmount <10K	2278 (3.7%)	2652.3	2635.2	**2621.4**
ClaimAmount >10K	44 (0.071%)	86511.6	86502.75	**86500.75**

TABLE *5: Local RMSE computed on the test base to compare the GLM and XGBoost models to predict the pure premium*

We remain lucid about the fact that our black box model seems very close to the classical GLM and would certainly not be deployed in practice, but we found interesting to introduce new tools to help in decision making about the choice of a predictive model.

GLM and XGBoost models interpretation

Once the performances of the two models were studied, in particular with a stability analysis with respect to the sampling of the test and learning database, we used the interpretation methods previously developed, to better understand the predictions made by them.

We first analyzed the predictions of the GLM. The properties of sparsity and simulability (detailed in this paper) allowed us to understand the model directly from the coefficients of each variable. In particular, we can easily explain the pathway that leads to such a prediction for an insured an also the change in prediction when an insured's characteristics are changed. To these intrinsic study of the GLM has been added a model-agnostic one using the interpretation methods seen before. We have shown using a global analysis (PDP, ALE and features importance) and a local analysis (ICE curves, LIME

and Shap) the coherence of the explanations provided by both of these two techniques.

Secondly, we did the same study on the XGBoost black-box model. His complex nature makes interpretation difficult and intrinsic analysis impossible : we need to use the previous tools of models interpretability. In the same way as the GLM using t-statistics, we were able to calculate the overall features importance, using the PFI - Permutation Feature Importance - method (see figure 7). Cette mesure a été complétée par une analyse de la stabilité des résultats, en réalisant plusieurs simulations.

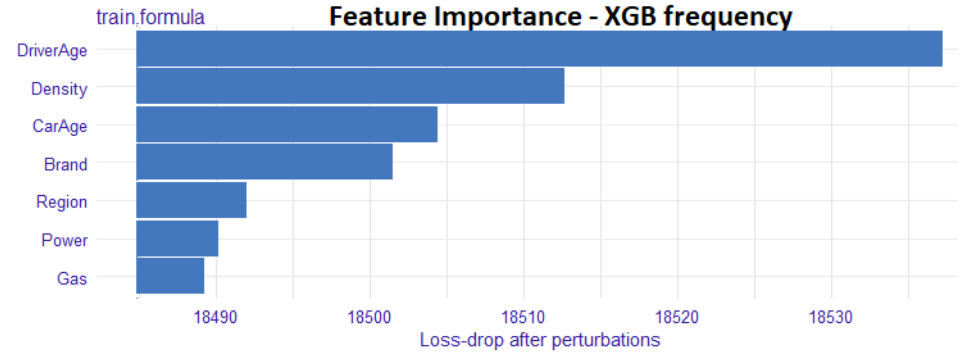

FIGURE *7: Features importance in the XGBoost frequency model, computed using PFI method*

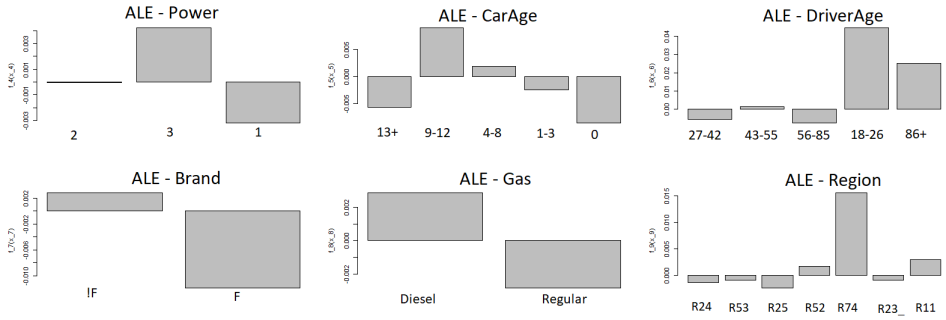

FIGURE *8: ALE plots computed for all seven features used in the frequency XGBoost model*

Next, we were able to understand the marginal average effect of a feature on the response provided by the model, using PDP and ALE. These last two methods give very similar results, which is consistent with the fact that the variables used are not very correlated. The accumulated local effects plot, for the XGBoost in frequency, is plotted in the figure 8. A positive value of a modality of a variable on this ALE curve means that the prediction on this modality is higher on average than the total average prediction. This is the case, for example, for the *18-25* class of the variable *DriverAge*, indicating that, on average, young drivers have more accidents.

Unlike the GLM for which no interaction exists between the features, the XGBoost has many heterogeneous effects. These interactions can be detected using several tools developed before, namely H-statistics, ICE curves or PDP (or ALE) plots combining two

features. It turns out that all these methods are associated with an huge computation time, especially the H-statistic. We concluded that when the variables used are all categorical, the use of the H-statistic is irrelevant. We finally chose to display some ICE curves (see figure 9), showing a non-translated curves, which is a sign of heterogeneity.

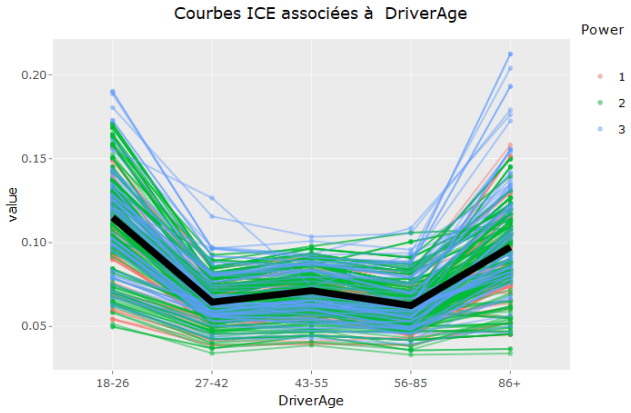

FIGURE *9: PDP and ICE curves of the feature DriverAge for the frequency XGBoost model*

This first analysis made possible to understand the overall effect of a feature on the XGBoost predictions and also to highlight the possible interactions between features. In particular, we have identified the cross effect of the $(DriverAge, Power)$ combination of variables, for which being both young and having a powerful vehicle greatly increases the risk of claims. We can see it in the figure 9, with a greater steepness between the modalities 18-26 and 27-42 for the blue curves, relative to the powerful cars, than for the graph of partial dependence in black. To get a better understanding of a single prediction,

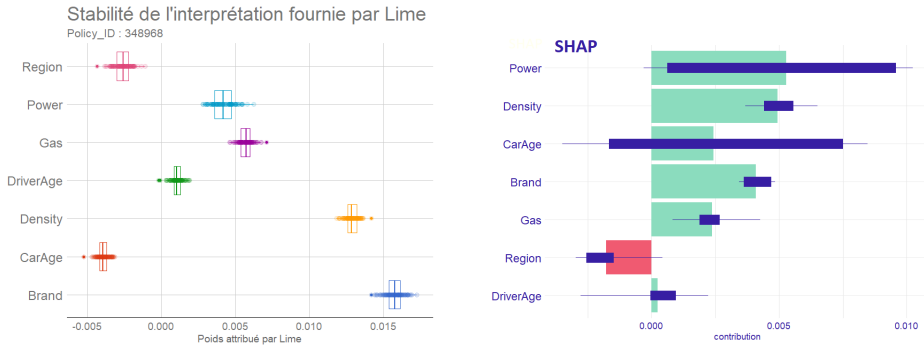

FIGURE *10: Boxplot of the results provided by LIME and Shap. On the left : the LIME weights for the prediction of the so-called insured 2, by the black-box model frequency XGBoost, based on 100 simulations. On the right : boxplot of the SHAP contributions for the prediction made on the so-called insured 2 by the frequency XGBoost model, based on 25 simulations*

we used local tools, namely LIME and Shap.

We completed this study with the analysis of the stability of the interpretations

provided by both of them. We concluded that for large databases, which is common in actuarial science, the SHAP method requires the use of a relatively small sample size to obtain reasonable computation times, resulting in high instability. This does not seem to be the case for LIME, for which the results are obtained much more quickly (see table 6) and also seem much more stable, as we can see in the figure 10. This effect is even more marked for the variable *CarAge* with the SHAP method for which the boxplot is very expanded, synonymous with high instability.

Assureur	Number of points used in each simulation	Number of simulations	Computation time (seconds)
SHAP	100	25	46
LIME	5000	100	14

TABLE *6: Computation time by LIME and Shap methods*

Conclusion

The interpretability of machine learning models is a hot topic that is gradually becoming an unavoidable issue for most data-scientists, but also for actuaries. Whether for ethical reasons - by ensuring that gender or ethnic origin are not discriminating variables, for example -, regulatory reasons - particularly with the ACPR and the RGPD who remain vigilant on the use of black boxes - or to generate greater confidence in the tools used, the question of transparency and understanding of models is vital today.

Numerous interpretation methods exist to understand the behavior of algorithms qualified as black boxes. It is then necessary to choose the tools adapted to the study carried out. These include global methods, such as the partial dependency graph (PDP) or the importance of variables, and local methods, such as LIME and SHAP, which are the most widely used at present. Many limitations remain for each of these methods, such as the uncertainty of the interpretations provided, which remains the most problematic. As the subject is in constant evolution, corrections and adaptations of these methods are constantly being proposed to remedy these different limitations. We can mention in particular the ALE which allows to solve the problem of correlation between the features encountered with the PDP.

Through this application in actuarial science, and more precisely in automobile pricing, we have shown the possibility of significantly improving the performance of trivial models on real data, using complex models such as the XGBoost for example.

This gain in performance is generally associated with a decrease in the transparency of the results. However, the application of these different methods of interpretation have shown that a learning model, however complex, can be interpreted using these tools and thus become "transparent" in the same way as a GLM. It thus seems possible to use such complex models for many actuarial subjects, such as automobile pricing, while respecting the regulatory constraints imposed by the GDPR and ACPR and ensuring total transparency.

Future research on the theme of interpretable machine learning (IML) seems to be moving towards a clearer formalism, with unambiguous definitions of the terms used, a general consensus on a framework for how to interpret a machine learning model and finally on the definition of a quality metric for these explanations, notably through robustness measures.

Table des matières

Table des figures

Liste des tableaux

Liste des sigles et acronymes

ACPR	Autorité de Contrôle Prudentiel et de Résolution
ALE	Effets Locaux Accumulés (Accumulated Local Effects)
CV	Validation croisée (Cross Validation)
GLM	Modèle Linéaire Généralisé (Generalized Linear Model)
IA	Intelligence Artificielle
ICE	Espérance Conditionnelle Individuelle (Individual Conditional Expectation)
IID	(Variables Aléatoires) Indépendantes et Identiquement Distribuées
IML	Machine Learning Interpretable (Interpretable Machine Learning)
LIME	Explication Locale et Interprétable, Indépendante du Modèle (Local Interpretable Model-Agnostic Explanations)
LM	Modèle Linéaire (Linear Model)
LS	Modèle de Substitution Local (Local Surrogate)
MAE	Erreur Moyenne Absolue (Mean Absolute Error)
ML	Machine Learning
MSE	Erreur Quadratique Moyenne (Mean Square Error)
NNET	Réseau de neurones
PD(P)	(Graphique de) Dépendance Partielle (Partial Dependance (Plot))
RF	Forêt Aléatoire (Random Forest)
RGPD	Règlement Général sur la Protection des Données
SVM	Support Vector Machine
VA	Variable Aléatoire
VINE	Visual INteraction Effect
XAI	eXplainable Artificial Intelligence
XGBoost	eXtrem Gradient Boosting

Introduction

Le machine learning explicable, l'interprétabilité des modèles ou encore la transparence des boîtes-noires sont désormais des termes couramment employés dans l'apprentissage statistique. Un algorithme n'est plus seulement jugé sur la performance qu'il produit mais également sur la possibilité de comprendre les prédictions qu'il réalise. Ce désir de transparence et d'interprétabilité vient à la fois des utilisateurs métiers qui ne veulent pas utiliser des outils qu'ils ne comprennent pas entièrement, mais également de la législation, avec notamment le Règlement général sur la protection des données (RGPD) et le "droit à l'explication".

Dans le domaine de l'assurance, l'Autorité de contrôle prudentiel et de résolution (ACPR) se doit d'assurer la protection des clients : l'assureur doit être capable de justifier toutes les décisions prises de manière détaillée et ne pas transférer la responsabilité à la machine. En particulier, le sexe ne peut plus être utilisé comme variable tarifaire et l'ACPR doit vérifier qu'aucune discrimination n'est faite par l'assureur.

Les modèles complexes, tels que les réseaux de neurones, le Support Vector Machine (SVM) ou le XGBoost détaillés dans ce mémoire, fournissent une alternative aux méthodes plus classiques utilisées en actuariat, comme le GLM en tarification, notamment grâce à leurs performances. Ce gain de précision se fait au détriment de la transparence et de l'interprétabilité : ces algorithmes n'offrent pas une interprétation claire de l'acheminement qui a mené à une prédiction. Un compromis s'opère généralement entre précision et interprétabilité.

Dans ce mémoire, après avoir introduit l'usage du machine learning en actuariat, nous tenterons de définir la notion d'interprétabilité d'un modèle.

Ensuite, nous verrons quelques modèles, considérés comme simples et transparents, couramment utilisés en actuariat pour ces raisons.

Après avoir détaillé quelques algorithmes complexes, nous verrons différentes méthodes d'interprétabilité des modèles, dont Lime et Shap qui sont les plus utilisées aujourd'hui, ainsi que les limites de ces différentes approches.

Enfin, nous mettons en place une tarification automobile à partir d'une base de données publique, en implémentant un GLM et un XGBoost. Nous utiliserons les différentes techniques vues dans ce mémoire pour interpréter le modèle de boîte noire.

Chapitre 1

Généralités sur l'apprentissage statistique et sur son utilisation en actuariat

1.1 Apprentissage statistique

Dans ce chapitre, nous voulons introduire les concepts généraux relatifs à l'apprentissage statistique, qui seront la base de nos travaux. Ensuite, nous voulons montrer l'importance du machine learning dans le métier d'actuaire, ainsi que la nécessité de transparence, qui s'illustre souvent par un compromis entre la précision et l'interprétabilité.

1.1.1 Approche générale de l'apprentissage supervisé

L'objectif principal de l'apprentissage statistique (aussi appelé machine learning) est de mettre en évidence de manière automatique des règles générales à partir d'exemples. Bien qu'il soit considéré aujourd'hui comme une branche de l'informatique et de la Data Science, l'apprentissage statistique est étroitement lié à la Statistique. Nous utiliserons indifféremment les termes *modèle* et *algorithme* au cours de ce mémoire, pour décrire les outils de l'apprentissage statistique comme les réseaux de neurones, SVM, etc., bien que certaines nuances peuvent être apportées. L'un des pionniers dans ce domaine est Vapnik, notamment grâce à ses apports fournis dans le livre *The Nature of Statistical Learning Theory* [70].

Nous considérerons que l'on dispose d'un échantillon de taille $n \in \mathbb{N}$ formé de couples de la forme : $Z_i = (X_i, Y_i),\ 1 \leq i \leq n$, supposés être des réalisations indépendantes d'une même loi de probabilité $\mathbb{P}$ inconnue. On écrit donc :

$$Zi = (X_i, Y_i) \overset{iid}{\sim} \mathbb{P},\ 1 \leq i \leq n$$

On note $x = (x_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq p}}$ et $y = (y_i)_{1 \leq i \leq n}$ les observations associées aux variables $(X_i)_{1 \leq i \leq n}$ et $(Y_i)_{1 \leq i \leq n}$. On suppose que les X_i appartiennent à un ensemble $\mathbb{X}$, appelé espace des

variables explicatives, entrées ou features. Généralement $\mathbb{X} = \mathbb{R}^p$ avec $p \in \mathbb{N}$. Les Y_i appartiennent eux à un ensemble $\mathbb{Y}$ appelé sorties, labels ou étiquettes. On est généralement confrontés à deux cas : $\mathbb{Y}$ un ensemble fini (classification) ou $\mathbb{Y}$ un sous-ensemble de $\mathbb{R}$ (régression). On se place donc dans le cas de l'apprentissage supervisé, c'est-à-dire que l'on veut prévoir une sortie Y associée à une nouvelle entrée X, où (X, Y) est une réalisation de la loi $\mathbb{P}$, supposée indépendante de celles observées auparavant. On suppose que toutes les quantités manipulées par la suite sont mesurables. On appelle fonction de prédiction associée à un modèle, une fonction mesurable de $\mathbb{X}$ dans $\mathbb{Y}$ et on note $F(\mathbb{X}, \mathbb{Y})$ l'ensemble des fonctions de prédictions. L'ensemble constitué de $Z_1 = (X_1, Y_1), ..., Z_n = (X_n, Y_n)$ est appelé la base d'apprentissage de notre algorithme, noté B_a par la suite. Un algorithme d'apprentissage (appelé simplement algorithme ou modèle par la suite) est une fonction qui à tout ensemble d'apprentissage renvoie une fonction de prédiction, c'est-à-dire qu'il s'agit d'une fonction de $\bigcup_{n=1}^{\infty} Z^n$ dans l'ensemble $F(\mathbb{X}, \mathbb{Y})$ avec $Z = \mathbb{X} \times \mathbb{Y}$. Cet algorithme est un estimateur de la "meilleure" fonction de prédiction. Pour expliquer le terme de "meilleure" fonction, nous devons définir une fonction de coût. Celle-ci a pour but de quantifier la perte entre la sortie réelle y et la sortie prédite par l'algorithme $\hat{y}$. On appelle fonction de perte, toute fonction $l : \mathbb{Y}^2 \to \mathbb{R}$. Les exemples les plus courants pour cette fonction de pertes sont :

- Pour la classification binaire : $\mathbb{Y} = \{0, 1\}$ et $\forall (y, \hat{y}) \in \mathbb{Y}^2, l(y, \hat{y}) = \mathbb{1}_{\{y \neq \hat{y}\}}$
- Pour la régression : $\mathbb{Y} = \mathbb{R}$ et $\forall (y, \hat{y}) \in \mathbb{Y}^2,\ l(y, \hat{y}) = |y - \hat{y}|^d$, $d \in \mathbb{N}^*$.
  Dans le cas courant $d = 2$, on parle de régression par moindres carrés.

A partir d'une fonction de coût l, on peut estimer la qualité d'une fonction de prédiction $g : \mathbb{X} \to \mathbb{Y}$ par son risque (appelé aussi erreur de généralisation) :

$$R_{\mathbb{P}}(g) = \mathbb{E}_{\mathbb{P}}[l(Y, g(X))] \tag{1.1}$$

On peut alors définir une "meilleure" fonction de prédiction $g^*_{\mathbb{P}}$ comme une fonction de $F(\mathbb{X}, \mathbb{Y})$ qui minimise le risque $R_{\mathbb{P}}$, c'est-à-dire :

$$g^*_{\mathbb{P}} \in \underset{g \in F(\mathbb{X}, \mathbb{Y})}{argmin} R_{\mathbb{P}}(g)$$

On appelle une fonction de la sorte oracle ou prédicteur de Bayes. Bien qu'une telle fonction n'existe pas nécessairement, nous nous plaçons dans le cas où celle-ci existe, ce qui est le cas pour les fonctions de pertes usuelles [22].

1.1.2 Compromis biais-variance et notion de surapprentissage

Comme nous ne disposons pas de la distribution de $\mathbb{P}$, nous ne pouvons pas connaître le risque $R_{\mathbb{P}}(g) = \mathbb{E}_{\mathbb{P}}[l(Y, g(X))]$. On peut cependant estimer le risque empirique $\hat{R}_n(g)$ à l'aide de l'échantillon $(Z_1, ..., Z_n)$ dont nous disposons :

$$\hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} l(Y_i, g(X_i))$$

En supposant $\mathbb{E}_{\mathbb{P}}[l(Y, g(X))^2] < +\infty$, nous déduisons du théorème central limite et de la loi forte des grands nombres que :

$$\hat{R}_n(g) \underset{n\to+\infty}{\overset{p.s}{\to}} R_{\mathbb{P}}(g), \ et \ \sqrt{n}[\hat{R}_n(g) - R_{\mathbb{P}}(g)] \underset{n\to+\infty}{\overset{D}{\to}} N(0, Var[l(Y, g(X))])$$

Il pourrait paraître alors logique de chercher comme algorithme g celui qui minimise $\hat{R}_n(g)$ sur l'ensemble $F(\mathbb{X}, \mathbb{Y})$ de toutes les fonctions de prédiction. Néanmoins, en le cherchant sur l'ensemble entier des fonctions de prédiction on s'expose à deux problèmes majeurs. Tout d'abord pour tout ensemble d'apprentissage, il existe une infinité de fonctions de prédiction minimisant le risque empirique. De plus, ce choix peut mener à un surapprentissage, c'est-à-dire que le risque réel $R_{\mathbb{P}}(g)$ sera bien supérieur au risque empirique $\hat{R}_n(g)$. Ainsi, notre choix d'algorithme g sera toujours fait en minimisant $\hat{R}_n(g)$ mais pas sur l'espace $F(\mathbb{X}, \mathbb{Y})$ entier mais sur un sous ensemble, noté G, de cet espace. En notant $g^*_{p,G} = \underset{g\in G}{argmin} R_{\mathbb{P}}(g)$ (prédicteur de Bayes sur le sous-ensemble G) et $\hat{g}_{n,G} = \underset{g\in G}{argmin} \hat{R}_n(g)$, on a l'inégalité :

$$R_{\mathbb{P}}(\hat{g}_{n,G}) \geq R_{\mathbb{P}}(g^*_{p,G}) \geq R_{\mathbb{P}}(g^*_{\mathbb{P}})$$

On peut alors décomposer l'excès de risque de $\hat{g}_{n,G}$ par rapport au prédicteur de Bayes $g^*_{\mathbb{P}}$, en deux termes :

$$R_{\mathbb{P}}(\hat{g}_{n,G}) - R_{\mathbb{P}}(g^*_{\mathbb{P}}) = \underbrace{(R_{\mathbb{P}}(\hat{g}_{n,G}) - R_{\mathbb{P}}(g^*_{p,G}))}_{erreur \ d'estimation} + \underbrace{(R_{\mathbb{P}}(g^*_{p,G}) - R_{\mathbb{P}}(g^*_{\mathbb{P}}))}_{erreur \ d'approximation}$$

Le premier terme $R_{\mathbb{P}}(\hat{g}_{n,G}) - R_{\mathbb{P}}(g^*_{\mathbb{E}_P,G})$ est appelé erreur stochastique ou erreur d'estimation, tandis que le deuxième $R_{\mathbb{P}}(g^*_{\mathbb{E}_P,G}) - R_{\mathbb{P}}(g^*_{\mathbb{P}})$ s'appelle erreur systématique ou erreur d'approximation ou encore biais [22]. Il découle que plus G va être grand plus l'erreur d'approximation sera faible (biais fiable), mais plus l'erreur d'estimation sera grande (variance élevée). Plus G sera petit, nous aurons alors les conclusions inverses, c'est-à-dire plus le biais sera élevé mais plus la variance sera faible. Nous avons donc un compromis à réaliser, appelé dilemme "biais-variance", représenté sur la figure 1.1.

Nous pouvons également définir ce dilemme biais-variance d'une autre manière. Considérons notre ensemble d'apprentissage :

$$\{x_1, ..., x_n\} \ et \ \{y_1, ...y_n\}, \ avec \ \forall i \in \{1, ..., n\} \ x_i \in \mathbb{R}^p \ et \ y_i \in \mathbb{R}$$

On suppose que l'on peut écrire :

$$\forall i \in \{1, ..., n\}, y_i = f(x_i) + \epsilon_i$$

Avec ϵ_i centré et de variance σ^2 ($\sigma > 0$). Soit $\hat{f}$ la fonction associée au modèle qu'on utilise, alors l'erreur attendue est : $\mathbb{E}_{\mathbb{P}}[(Y - \hat{f}(X))^2]$. On peut alors montrer que cette erreur se décompose en trois termes, à savoir :

$$\mathbb{E}_{\mathbb{P}}[(Y - \hat{f}(X))^2] = B[\hat{f}(X)]^2 + Var[\hat{f}(X)] + \sigma^2$$

Les différents termes présents dans cette équation sont :

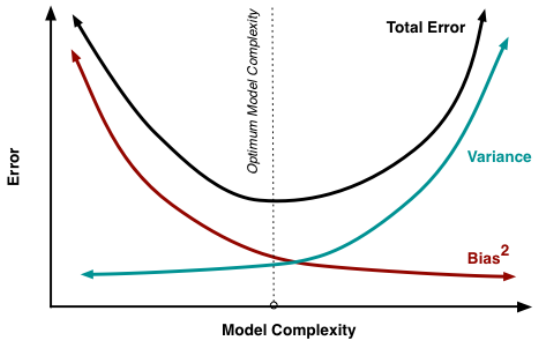

FIGURE *1.1: Compromis biais-variance (issu du site [63])*

— Le *biais* :

$$B[\hat{f}(X)] = \mathbb{E}_{\mathbb{P}}[\hat{f}(X) - f(X)]$$

Il peut être interprété comme l'erreur due au modèle simplifié utilisé.

— La *variance* :

$$\text{Var}\left[\hat{f}(X)\right] = \mathbb{E}_{\mathbb{P}}\left[\left(\hat{f}(X) - \mathbb{E}_{\mathbb{P}}[\hat{f}(X)]\right)^2\right]$$

Elle peut être vue comme l'erreur due à la sensibilité aux petites fluctuations de l'échantillon d'apprentissage.

— L'*erreur irréductible* σ^2, résultant du bruit lui-même.

La recherche de la fonction $\hat{f}$ la plus pertinente par rapport à nos données se fera dans une classe de modèles choisie telle que les modèles linéaires, les réseaux de neurones à une couche cachée, etc... Pour trouver le "meilleur" modèle, on doit également tenir compte de la complexité de celui-ci : à taux d'erreur équivalent, un modèle moins complexe sera privilégié. Il existe de nombreux critères qui pénalisent les modèles complexes (avec beaucoup de paramètres par exemple) tels que le R^2 ajusté, le BIC et AIC pour les modèles linéaires. Le modèle $\hat{f}$ retenu à partir de l'échantillon $(x_i, y_i)_{1\leq i\leq n}$ peut représenter un biais d'échantillon, c'est-à-dire qu'il "surapprend" les données de la base d'apprentissage. Cela vient du fait que les modèles complexes sont capables de modéliser presque parfaitement des données, sans pour autant avoir une bonne généralisation. C'est pourquoi il est courant en apprentissage statistique de mesurer la qualité de prédiction d'un modèle, sur un échantillon indépendant de la base d'apprentissage, appelé base de test (notée B_t). Sur la figure 1.2, on peut voir sur le graphique de gauche que le modèle en bleu colle parfaitement aux données d'apprentissage (en rouge) et aura donc un taux d'erreur nul sur cette base d'apprentissage. Cependant, le modèle ne va sûrement pas bien se généraliser et va fournir une erreur élevée sur la base de test : on a un phénomène de surapprentissage. Sur le graphique de gauche, le modèle ne colle pas parfaitement aux données et aura une erreur sur la base d'apprentissage plus élevée que le modèle précédent. Cependant, son erreur de test sera bien plus faible et donc celui-ci sera retenu par rapport au modèle précédent.

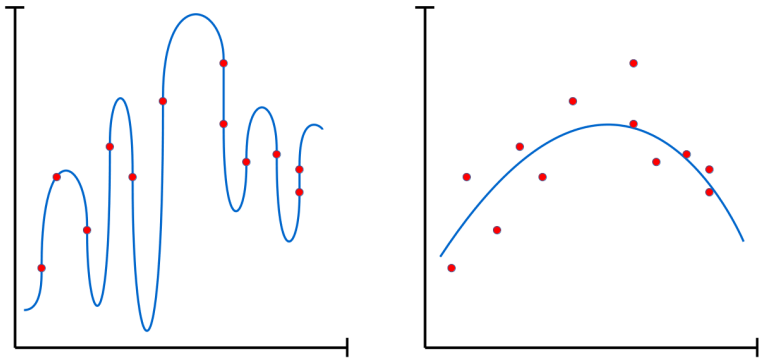

FIGURE *1.2: Comparaison entre surapprentissage (à gauche) et bon apprentissage (à droite) (issu du site [63])*

Une manière de représenter le biais et la variance d'un modèle est une cible de fléchettes, pour laquelle on veut viser au plus près du centre. Si en en moyenne, les prédictions du modèle (correspondant aux tirs d'un joueur) sont au centre, alors le modèle est sans biais. Si les tirs sont très éparpillés sur la cible, alors la variance sera élevée. Ceci est représenté sur la figure 1.3.

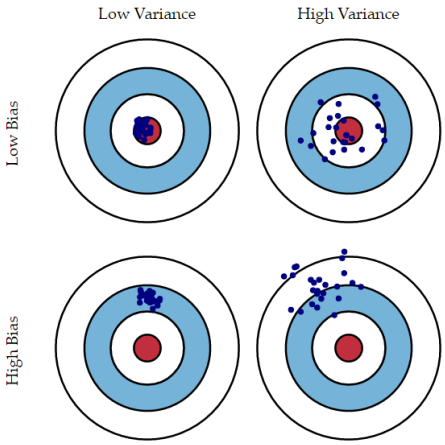

FIGURE *1.3: Représentation du biais et de la variance d'un modèle d'apprentissage statistique par l'intermédiaire d'une cible (issu du site [63])*

1.1.3 Validation croisée

Avant d'analyser les résultats fournis par notre algorithme sur une base de test, il est courant d'avoir recours à une validation croisée pour optimiser les paramètres disponibles dans le modèle (par exemple le nombre d'arbres utilisés pour un algorithme de

forêt aléatoire ou un XGBoost). La validation croisée consiste à entraîner le modèle avec différents paramètres (par exemple de 1 à 100 arbres pour la forêt aléatoire) sur une base d'apprentissage, et de tester les résultats fournis par ce modèle sur un échantillon indépendant des données d'entraînement. Le but est de trouver les paramètres optimaux du modèle, c'est-à-dire ceux qui minimisent l'erreur sur l'échantillon de validation, notamment pour éviter le surapprentissage décrit ci-dessus. On trace généralement les courbes d'erreurs obtenues sur la base d'apprentissage et la base de validation en fonction du paramètre étudié. La courbe d'erreur par rapport à la base d'apprentissage est couramment décroissante, tandis que la courbe d'erreur sur la base de validation est décroissante, puis croissante, comme on peut l'observer sur la figure 1.4. Le point à partir duquel la courbe de validation devient croissante correspond à la valeur du paramètre pour laquelle le surapprentissage apparaît. Dans l'exemple de la figure 1.4, on note que le nombre optimal d'arbres semble être 11.

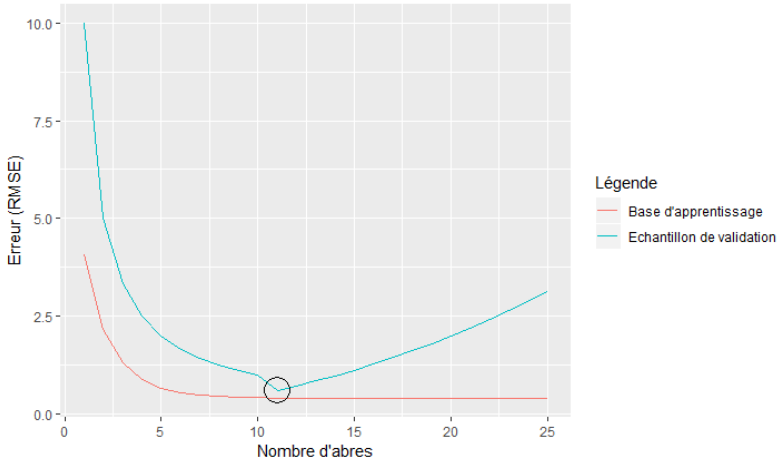

FIGURE *1.4: Exemple de courbe de validation croisée obtenue pour un algorithme de forêt aléatoire associée à un problème de régression, avec comme métrique retenue le RMSE*

Il existe différentes méthodes de validation croisée, les trois les plus utilisées sont les suivantes :

— *Méthode Holdout* : On divise l'échantillon à notre disposition en 2 sous-échantillons, l'un d'apprentissage (généralement de taille supérieure à 60%) et l'autre de test. L'idée est d'ajuster le modèle, en faisant varier le paramètre d'intérêt, sur la base d'apprentissage et de calculer l'erreur résultante sur la base de test. L'erreur couramment utilisée est le RMSE (l'erreur moyenne quadratique) pour la régression et le taux d'erreur (nombre de prédictions correctes faites par le modèle divisé par le nombre d'individus) pour la classification.

— *k-fold cross-validation* : l'idée est de séparer notre jeu de données en $k \in \{1, ..., n\}$ sous-échantillons. Chaque sous-échantillon créé servira de base de test, associé au modèle ajusté sur les $k-1$ échantillons restants. On répète ainsi cette procédure k fois et on obtient k scores d'erreur. Finalement l'erreur globale estimée sera la

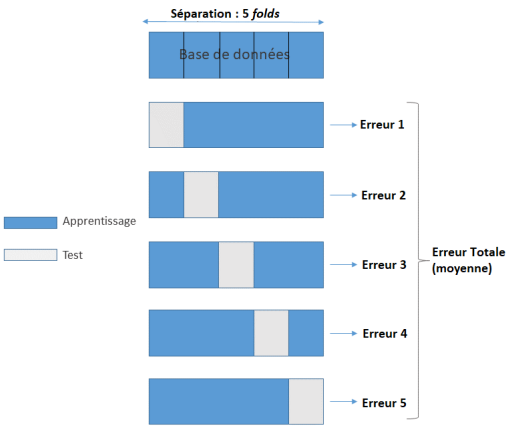

FIGURE *1.5: Explication de la k-fold cross validation dans le cas k=5*

moyenne de ses k erreurs précédemment calculées.

— *Leave One Out Cross Validation* : il s'agit du cas particulier de la *k-fold cross-validation* lorsque le nombre de sous-échantillons créés est le nombre de données à dispostion, i.e. : $k = n$.

Au cours des travaux réalisés, nous avons le plus souvent utilisé l'algorithme de *k-fold cross-validation* avec k=5, détaillée sur la figure 1.5.

1.2 Convergence de l'actuariat vers la Data Science

Aujourd'hui, la Data Science et le machine learning prennent une place prépondérante dans de nombreux secteurs y compris l'assurance. Revenons tout d'abord sur l'histoire du machine learning et des termes associés. Dans les années 1940-1950, les travaux d'Alan Turing et les premiers modèles neuronaux mis en place forment les débuts de l'intelligence artificielle (IA). Dans les années 50, le Workshop de Darmout College officialise la discipline de l'IA et donne naissance au machine learning pour l'apprentissage à partir de données. Entre 1960 et 1980, le machine learning se développe et à partir des années 80, le Statistical Learning prend place, avec notamment le début des GLM en assurance et le développement des méthodes par arbre comme CART. En 2001, W. CLeveland définit la Data Science comme "l'extension du domaine de la statistique à divers champs d'application". Les GAFA introduisent le terme de Big Data et de Deep Learning pour le traitement et l'analyse des données massives [12]. Comme indiqué précédemment, il est difficile de dissocier l'actuaire d'un data scientists. En effet, dans la majorité des travaux actuariels, que ce soit la tarification, le provisionnement ou l'évaluation du capital, utilisent des méthodes d'apprentissage statistique qui sont proches des missions réalisées par un data scientist. Pour David Dubois, président de l'Institut des Actuaires, "l'actuaire est un data scientist mais un data scientist n'est pas, inversement, un actuaire" [46]. Actuellement en tarification non-vie, le modèle utilisé le plus abondamment est le modèle GLM

(en coût-fréquence). Cependant, l'actuaire a également à sa disposition de nombreux modèles complexes tels que les forêts aléatoires, XGBoost ou SVM qui peuvent compléter l'analyse de la sinistralité. Ces derniers peuvent permettre une explication plus fine et sont capables de mieux capter la complexité du phénomène que l'on cherche à expliquer, ce qui ne sera pas le cas avec les modèles linéaires classiques. Des études ont montré que l'utilisation de méthodes construites à partir d'arbres pouvait donner de meilleurs performances pour la modélisation de la prime pure en tarification automobile. On peut citer l'article [55] de Paglia et Al. (2010) qui a été un des premiers mémoires à utiliser des arbres pour remplacer les GLM, et également le mémoire [8] de R. Bellina qui utilise des méthodes d'agrégation d'arbres.

D'autres études ont montré que l'utilisation de méthodes de machine learning, comme le XGBoost par exemple, permet une meilleur estimation de provisions dans le cas de l'activité Garantie contre les accidents de la vie (GAV), comme il est souligné dans le mémoire [54] de P. Ottou.

Le mémoire [4] de Aouizerate (2010) fut l'un des premiers à proposer une approche par réseaux de neurones en tarification santé, avec de meilleures performances que les modèles plus traditionnels.

Enfin, R. Gauville dans son mémoire d'actuariat [28], a étudié une méthode alternative à la méthode SdS (simulations dans les simulations), utilisée pour le calcul du SCR en modèle interne. Cette nouvelle méthode repose sur une approche machine learning, reposant sur les modèles SVM, Random Forest et XGBoost, fournit des résultats bien meilleurs que la méthode classique SdS, à savoir un taux d'erreur très faible, une stabilité temporelle et enfin un temps d'exécution réduit.

1.3 Compromis entre précision et interprétabilité

Nous avons donc vu que les modèles de machine learning fournissent une alternative aux méthodes plus classiques utilisées en actuariat, notamment grâce à leurs performances. Cependant ce gain de précision par une approche de type machine learning a un coût : il se fait au détriment de l'interprétabilité et de la transparence. En effet, les modèles complexes comme le XGBoost n'offrent pas une interprétation claire de l'acheminement qui a mené à une prédiction. Ainsi, un compromis entre la précision et l'interprétabilité semble nécessaire [60] : il s'agit de trouver un équilibre entre un modèle complexe capable d'apprendre en profondeur un phénomène et la capacité de comprendre comment ces tâches ont été accomplies. Ce compromis pourrait également être nommé connaissance contre contrôle ou efficacité contre simplicité. Ainsi lorsque l'on met en place un modèle d'apprentissage statistique, il est nécessaire de se demander si l'on souhaite obtenir les meilleurs résultats ou avoir une compréhension de comment ces résultats ont été fournis. Dans leur livre [42], Kuhn et Johnson se préoccupent principalement de la précision d'un modèle au détriment de son interprétation. En effet, pour eux "tant que les modèles complexes sont correctement validés, l'utilisation d'un modèle construit pour l'interprétation plutôt que pour la performance prédictive peut s'avérer inappropriée". Ils s'appuient sur les exemples de détection de spam dans les mails et d'évaluation du prix

d'une maison pour lesquels la compréhension de ce qui a mené à la prédiction n'est pas nécessaire. Cependant, dans de nombreux autres cas, il semble important d'avoir à la fois une bonne précision et une bonne interprétabilité, notamment pour le diagnostic médical d'un patient. On peut également citer la détection de fraude, pour laquelle il n'est pas très utile pour un utilisateur de disposer uniquement d'une liste de tentatives de fraudes potentielles sans avoir la raison qui a mené à cette hypothèse. L'utilisateur voudra par exemple savoir pourquoi le système pense qu'il s'agit d'une fraude, par exemple parce que la carte de crédit a été utilisée pour effectuer un grand nombre de transactions d'une somme inférieure au montant habituel.

Le graphique 1.6 résume l'importance de l'interprétabilité d'un modèle. Les notions abordées sur cette figure seront développées plus précisément dans le chapitre suivant.

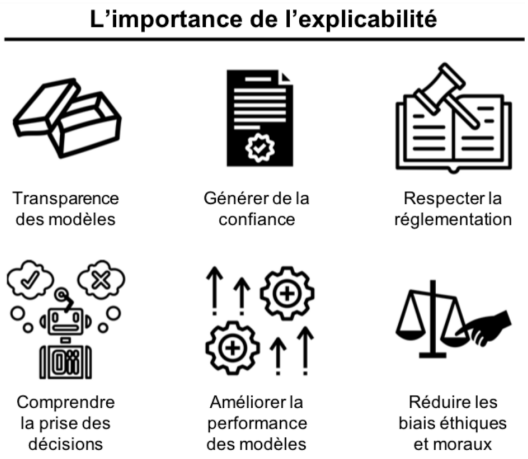

FIGURE *1.6: Schéma résumant l'importance de l'interprétabilité d'un modèle de machine learning (issu de l'article [21])*

1.4 Transparence en actuariat

Le gain d'efficacité obtenu à l'aide des modèles de machine learning a engendré une grande perte de l'interprétabilité des modèles. En effet les décisions prises par ces modèles complexes semblent difficilement lisibles pour un humain qui ne semblent même plus en capacité d'expliquer les prédictions : c'est pourquoi on qualifie ces modèles de boîtes noires.

1.4.1 Précision : un besoin majeur face à l'émergence d'acteurs sur le marché de l'assurance

L'utilisation des modèles complexes "boîtes noires" devient quasiment indispensable pour les acteurs de l'assurance qui doivent faire face à une concurrence accrue depuis quelques années. Notons cependant que ces propos peuvent être nuancés : utiliser des méthodes plus complexes et segmenter davantage ne présentent pas nécessairement que

des avantages et des interrogations existent toujous à ce sujet, comme le souligne Planchet [58]. On peut citer tout d'abord les GAFA qui souhaite intégrer le marché de l'assurance et leur présence dans un futur proche n'est pas à exclure [20]. D'autres acteurs, tels que les Assurtechs, viennent bouleverser le monde assurantiel, en proposant des produits innovants et une simplification de soucription pour les clients. Ces derniers semblent favorables aux Assurtechs et aux services qu'ils proposent [57].

Les GAFA et les Insurtechs sont enclins à utiliser les méthodes de "boîtes noires" car ils maîtrisent les technologies les plus avancées, notamment Google qui utilise déjà ce type d'algorithme dans son activité, avec la voiture autonome par exemple. Ainsi, l'utilisation des méthodes les plus performantes, telles que XGBoost, semble inévitable pour les Assurances, en particulier pour éviter le risque d'anti-sélection. En effet, si les assureurs continuent d'utiliser des méthodes anciennes, comme le GLM, contrairement aux acteurs émergents qui utilisent eux les techniques les plus fines, alors ils s'exposent au risque de récupérer les mauvais risques, à cause de prix moins attractifs.

1.4.2 Contraintes réglementaires et métiers

Face à la nécessité d'utilisation des méthodes complexes se posent des contraintes réglementaires et métiers. Tout d'abord, le RGPD impose une nouvelle exigence sur la transparence, obligeant à justifier l'usage de modèle "boîte noire". Le RGPD autorise aux utilisateurs le droit de regard sur la collecte et l'usage des données dans les articles 16 à 20 sur la rectification et l'effacement. L'article 22 donne le "droit de ne pas faire l'objet de décisions entièrement automatisées", c'est-à-dire que la machine ne peut pas remplacer totalement l'humain dans la prise de décision [33].

En assurance spécifiquement, l'ACPR se doit d'assurer la protection des clients : l'assureur doit être capable de justifier toutes les décisions prises de manière détaillée et ne pas transférer la responsabilité à la machine ou à la "boîte noire" [72]. L'ACPR vérifie en particulier qu'il n'y a pas de discrimination et que l'assureur n'introduit pas un biais (volontaire ou involontaire) en utilisant des variables interdites telles que le sexe.

D'un côté opérationnel, on observe une certaine réticence des utilisateurs "métiers". En général, celui qui calibre le modèle n'est pas celui qui l'utilise *in fine* et l'utilisateur n'aura pas intérêt à ce qu'il soit peu expliqué ou difficilement compréhensible. La moindre erreur de décision commise par le modèle va tendre à le discréditer et donc à augmenter le risque de réputation du fait de ce manque de compréhension. De plus, la mise en place d'un modèle complexe nécessite la maîtrise totale des décisions qu'il prend afin de pouvoir garantir son utilisation sur le long terme. Il doit pouvoir être facilement modifié et adapté à la situation actuelle.

Chapitre 2

Propriétés et définition du machine learning interprétable

Le machine learning a été particulièrement encouragé pour sa capacité à prédire des phénomènes complexes. En plus des prédictions réalisées, ces modèles peuvent nous fournir de la connaissance, notamment sur les relations entres les différentes données, ce qui est généralement qualifié d'interprétation des modèles. Ces méthodes ont connu un essor considérable au cours des dernières années. Les auteurs de l'article [2] ont conduit une étude afin de montrer le gain d'intérêt pour l'interprétation des modèles de machine learning. Leur idée était de s'appuyer sur les mots-clés suivants : "interpretable", "intelligible", "transparency" "black box", "understandable" and "explainable" afin de trouver les articles en lien avec ce sujet sur les sites comme Google Scholar ou arXiv. Afin de s'assurer du lien avec l'apprentissage statistique, les mots clés tels que "machine learning", "Deep Learning" ou "Decision Tree" ont été ajoutés. Ils ont finalement obtenus la courbe de la figure 2.1, montrant l'augmentation exponentielle du nombre de papiers sur ce sujet ces dernières années.

Cependant, on peut noter une absence de consensus général sur la définition et la mesure de l'interprétabilité d'un modèle de machine learning : aucune définition mathématique rigoureuse ne peut être trouvée [51]. En effet, de nombreuses méthodes (graphiques, mathématiques, etc.) ont été rangées dans la catégorie d'interprétation des modèles ce qui a entraîné une certaine confusion sur la notion d'interprétabilité. Par exemple, il semble difficile de définir clairement la notion d'interprétation d'un modèle ainsi que de choisir la méthode d'interprétation adéquate pour un problème donné. L'article [52] *Interpretable machine learning : Definitions, Methods, and Applications* de Murdoch et Al. (2019) tente de donner une définition précise à l'interprétabilité dans le cadre d'un modèle de machine learning et également de fournir un cadre appelé PDR, construit sur trois propriétés souhaitées pour l'évaluation et la construction d'une interprétation : la précision prédictive (P), la précision descriptive (D) et la pertinence (R). Ceci permet entre autre de classer les différentes méthodes existantes et d'utiliser un vocabulaire commun entre les différents acteurs du domaine de l'apprentissage statistique.

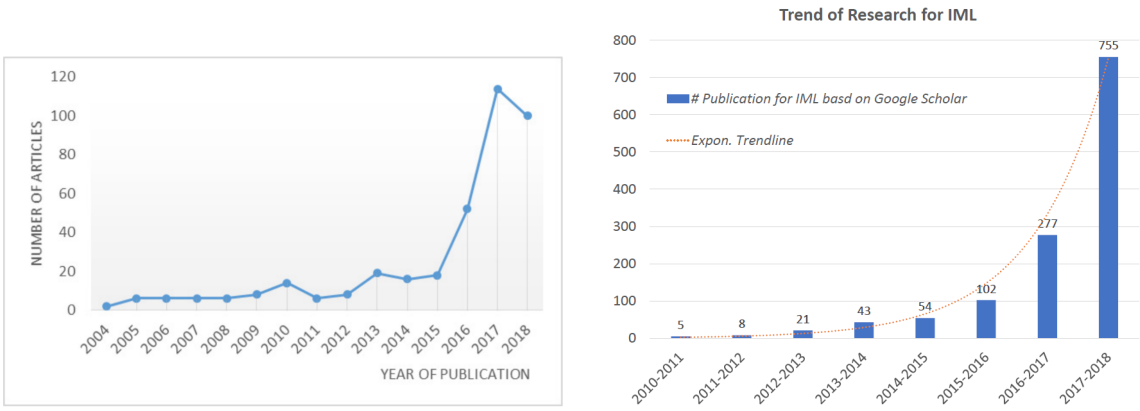

FIGURE *2.1: Nombre d'articles publiés en lien avec l'interprétabilité des modèles de machine learning au cours des 15 dernières années (issu de l'article 2)*

2.1 Définir le machine learning interprétable

Comme indiqué dans l'introduction, il n'existe pas réellement de définition commune et précise de l'inteprétabilité. L'article [52] suggère tout d'abord qu'interpréter quelque chose fait référence à la notion d'extraction d'informations. Miller propose dans l'article [50] de définir l'interprétabilité comme le degré à partir duquel un humain peu comprendre la cause de la décision. Une définition alternative proposée par C. Molnar [51] est : l'interprétabilité est le degré à partir duquel un humain peut régulièrement prédire le résultat du modèle. Enfin, l'article 52 propose de définir l'interpretabilité comme l'utilisation du machine learning pour extraire des connaissances pertinentes sur les relations contenues dans les données. Pour préciser cette définition, une connaissance est dite pertinente si elle fournit une information pour un public particulier et un problème d'un domaine choisi [52]. Ces informations ainsi obtenues, grâce aux méthodes d'interprétabilité des modèles, seront alors des vecteurs de communication, d'action et de découverte.

2.2 Propriétés souhaitées de la recherche d'interprétabilité

Avant de détailler le cadre PDR introduit précédemment, déterminons les objectifs de la recherche d'interprétabilité en répondant à la question : pourquoi souhaitons-nous qu'un modèle soit interprétable ? La figure 2.2 résume les différentes étapes qui ont lieu lors de la mise en place d'un modèle de machine learning, avec notamment une partie consacrée à l'interprétation de l'algorithme implémenté.

Pendant longtemps, les modèles de machine learning étaient jugés par leur pouvoir prédictif, par exemple avec le taux d'erreur pour la classification ou le MSE pour la régression. Dans certains cas, un modèle performant n'est pas suffisant pour le public concerné et une interprétation des décisions prises par l'algorithme est requise. L'article

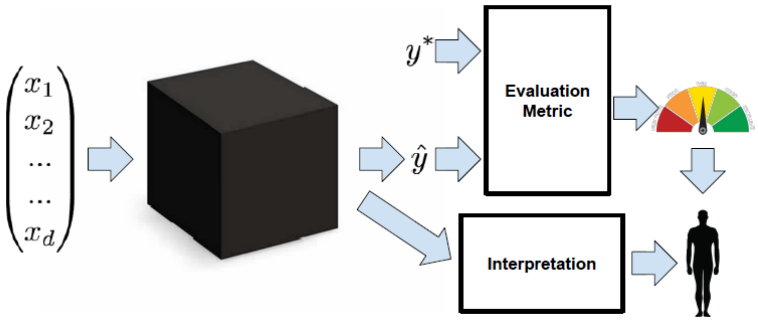

FIGURE *2.2: Schéma résumant les différentes étapes lors de la mise en place d'un modèle de type boîte-noire (issu de l'article [47])*

[47] essaie de définir quelles propriétés doivent alors vérifier ces interprétations et dans quel contexte elles sont nécessaires.

2.2.1 Confiance

Un terme couramment employé lorsque l'on essaie de définir l'interprétabilité est la confiance, comme dans l'article [61] *Why Should I Trust You ?* de Ribeiro et Al. (2016). Cependant le terme "confiance" reste assez vague : ne s'applique-t-il qu'aux modèles avec de bonnes performances ? Il semble qu'une part de subjectivité est à prendre en compte lorsque l'on définit un modèle auquel on peut faire confiance. Par exemple, dans de nombreux domaines, le client se sentira plus à l'aise s'il est capable de comprendre l'algorithme, même si sa compréhension n'est pas indispensable au bon déroulement du projet. Le terme client réfère ici à l'utilisateur concerné par l'algorithme en question, qui dans le cadre de l'application actuarielle réalisée dans ce mémoire (5) est l'assuré ayant souscrit une assurance pour son véhicule.

Dans d'autres situations, par exemple pour la prédiction du taux de criminalité, il semble primordial de comprendre ce qui a conduit le modèle à une telle prédiction. Si un biais selon la couleur de peau a été introduit dans l'apprentissage du modèle il est important de le détecter. On peut donc définir un modèle auquel on fait confiance lorsque l'on se sent capable d'abandonner le contrôle sur celui-ci. On ne va plus s'intéresser seulement à la fréquence à laquelle un modèle a raison mais aussi pour quels exemples il prédit correctement. En effet, si l'algorithme a tendance à faire des erreurs dans les régions où l'humain se trompe également, on peut facilement abandonner son contrôle. A l'inverse, si le modèle réalise des erreurs là où l'humain n'en fait pas, il semble nécessaire de maintenir une supervision de l'algorithme.

2.2.2 Causalité

Bien que l'un des objectifs des algorithmes d'apprentissage statistique soit de réaliser des associations, les chercheurs les utilisent également pour en déduire des propriétés

particulières ou générer des hypothèses sur le monde réel, comme le fait de fumer peut provoquer des cancers du poumon.

Il faut cependant se méfier des possibles associations apprises par le modèle : la corrélation n'implique pas la causalité. Un exemple classique pour l'illustrer est que 57 % des décès ont lieu à l'hôpital : on a une probabilité plus grande de mourir à l'hôpital que dans son lit. Pour autant, l'hôpital ne peut pas être considéré comme un lieu dangereux. C'est parce qu'on est malade que l'on se rend à l'hôpital et c'est quand on est malade qu'on risque plus de mourir.

2.2.3 Transférabilité

Pour étudier les performances d'un modèle d'apprentissage statistique, on considère généralement l'erreur de généralisation (c.f 1.1). Pour se faire, on regarde la différence de performance sur la base d'apprentissage et la base de test. L'erreur commise sur la base de test, sauf cas très rare, est plus élevée du fait que ces données sont totalement nouvelles pour l'algorithme mis en place. Cependant, comme le souligne l'article [47], l'humain possède une meilleure capacité à généraliser que les modèles, en transférant ses compétences apprises à des situations non familières. Par exemple, lorsque l'environnement, dans lequel on réalise l'étude, n'est pas stationnaire, une telle compétence est nécessaire. Il arrive aussi que, dans certaines situations, la mise en place d'un modèle sur une population donnée, peut modifier le comportement de celle-ci, invalidant alors le modèle. Un exemple, fourni par Caruana et Al. (2015) indique une probabilité de décès à cause de la pneumonie plus faible lorsque le patient est atteint d'asthme. Cela vient du fait que les patients atteints d'asthme reçoivent un traitement plus agressif. Si on suivait les préconisations données par l'algorithme, c'est-à-dire de rendre le traitement moins agressif pour les personnes atteintes d'asthme, le modèle deviendrait faux. Ainsi, la transférabilité, définie comme la capacité au modèle de s'adapter à des situations légèrement différentes, est une des propriétés souhaitées dans la recherche d'interprétabilité.

2.2.4 Informativité

Un autre élément que l'on recherche dans l'interprétabilité d'un modèle est le fait de fournir des informations concernant les décisions prises par l'algorithme. Il ne s'agit pas seulement de minimiser le taux d'erreur comme cela est couramment fait notamment sur le site de compétition Kaggle, mais également de fournir des informations utiles dans le monde réel. En particulier, les relations de causalité, évoquées précédemment, font partie de ces informations utiles à l'interprétation d'un modèle.

2.2.5 Prise de décision juste et éthique

Un autre élément qui a conduit à introduire l'interprétabilité des modèles est la législation et le désir de transparence des utilisateurs. Dans la société actuelle, la question d'éthique est prépondérante et la prise de décision automatisée inquiète une partie de la population [33]. L'article [47] cite l'utilisation de modèles de machine learning pour

l'assignation d'un score de crédit, le filtrage des offres d'emploi ou encore pour prédire le risque de récidive après une condamnation. Une question qui se pose alors est l'absence de biais dans la prise de décision de l'algorithme concernant la couleur de peau. Comme indiqué précédemment, la législation européenne renforce cette transparence, ainsi que la protection des données à caractère personnel, avec le RGPD. Une partie est consacrée à la non-discrimination tandis qu'une autre concerne le *droit à l'explication* (c.f. article [33]). Dans le domaine actuariel, le sexe ne peut plus être utilisé comme variable discriminante. L'interprétabilité peut alors être un outil pour s'assurer du respect de cette règle, notamment pour l'ACPR.

2.3 Cadre PDR et différents types d'interprétabilité

2.3.1 Interprétabilité dans le cycle de vie de la Data Science

L'article [52] propose un cycle de vie associé à la réalisation d'un projet de Data Science (c.f. figure 2.3).

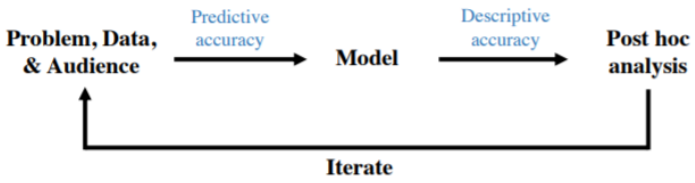

FIGURE *2.3: Cycle de vie d'un projet de machine learning (issu de l'article [52])*

Au début du cycle, le praticien doit définir le problème associé au domaine considéré comme par exemple le calcul d'un score pour la demande de crédit, ou la prédiction d'un mail frauduleux. Après l'avoir bien définir, il collecte les données nécessaires et effectue les retraitements tels que le tri ou le nettoyage des données. L'étape suivante est le choix du modèle. Comme nous l'avons mentionné dans la partie précédente, l'objectif est de trouver la meilleure fonction, appelée oracle, c'est-à-dire celle minimisant l'erreur de généralisation (c.f 1.1). Nous avons également souligné qu'une infinité de fonctions peuvent répondre à ce problème, c'est pourquoi un choix d'algorithmes est à réaliser. On cherche généralement la famille de modèles donnant les meilleures performances. Néanmoins, il se peut que l'on se restreigne à des catégories d'algorithmes jugées interprétables - notion que l'on définira par la suite, comme les modèles linéaires ou les arbres de décision par exemple. Cette restriction empêche l'utilisation de boîtes-noires, comme le SVM ou le XGBoost détaillés dans ce mémoire, qui potentiellement modélisent mieux le phénomène étudié pouvant ainsi engendrer une diminution de la précision prédictive (c.f *predictive accuracy* sur la figure 2.3).

Une fois le(s) modèle(s) ajusté(s), il doit le (ou les) analyser pour répondre au problème posé initialement : on parle d'analyse post-hoc. Pour cela, des informations sont extraites du modèle, comme des histogrammes ou des scatter-plot par exemple. La ca-

pacité des interprétations à décrire proprement ce que le modèle a appris est appelée la précision descriptive (*descriptive accuracy* sur la figure 2.3).

2.3.2 Cadre PDR : précision Prédictive, précision Descriptive et Pertinence

L'article [52] a construit un cadre, appelé PDR, pour sélectionner et évaluer les différentes méthodes utilisées pour l'interprétation d'un modèle pour un problème et un public particulier. Celui-ci repose sur trois critères fondamentaux : la précision descriptive, la précision prédictive et la pertinence. On distingue deux types d'interprétabilité, une basée sur le modèle (*model-based interpretability*) et l'autre post-hoc (*post-hoc interpretability*), une fois que le modèle a été ajusté. La figure 2.4 résume l'impact de ces deux-types d'interprétabilité sur la précision prédictive et la précision descriptive dans le cadre du PDR.

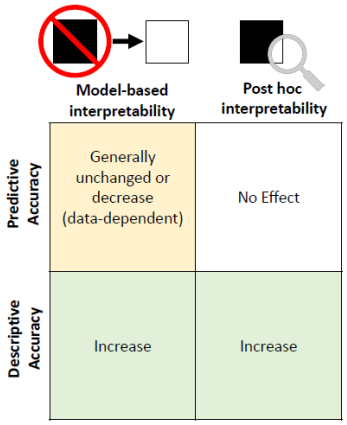

FIGURE *2.4: Impact des méthodes d'interprétabilité sur les précisions prédictive et descriptive dans le cadre du PDR (issu de l'article [52])*

2.3.2.1 Précision prédictive

La précision prédictive est la première source d'erreur lors de la mise en place d'un modèle. Les critères généralement utilisés sont le taux d'erreur pour la classification et l'erreur quadratique pour la régression. Si le modèle apprend mal les relations au sein des données, toutes les informations extraites de celui-ci seront probablement imprécises. C'est pour cela que lorsqu'on cherche à interpréter un modèle, la précision prédictive est requise afin d'obtenir des résultats pertinents. La distribution des prédictions est également importante : il peut être problématique que l'erreur de prédiction soit bien plus élevée pour une classe spécifique. De plus, on souhaite avoir une précision prédictive stable selon de petites perturbations au niveau des données. En particulier, si le modèle apprend sur un échantillon légèrement plus petit et que les résultats associés sont très nettement différents, il sera difficile de faire confiance à l'interprétation fournie initialement.

2.3.2.2 Précision descriptive

La deuxième source d'erreur survient lors de l'analyse post-hoc, une fois que le modèle a été mis en place. Généralement, les méthodes d'interprétation fournissent une représentation imparfaite des relations apprises par le modèle. Ceci est notamment le cas lorsque la boîte noire est très complexe, comme les réseaux de neurones profonds pour lesquels les relations sont clairement non linéaires. L'article [52] propose la définition suivante pour la précision descriptive : *degré à partir duquel une méthode d'interprétation capture objectivement les relations apprises par les modèles de machine learning.* Lorsque nous sommes confrontés au choix du modèle de machine learning pour un problème donné, un compromis est à réaliser entre précision prédictive et précision descriptive. L'intérêt des méthodes d'interprétation basées sur le modèle est la simplicité qui en découle et donc une grande précision descriptive, qui néanmoins se fait généralement au détriment de la précision prédictive sur les jeux de données complexes. D'un autre côté les modèles à grand pouvoir prédicitif, comme les méthodes de Deep Learning pour les images, sont plus complexes à analyser et possèdent une faible précision descriptive.

2.3.2.3 Pertinence

Quand on sélectionne une méthode d'interprétation (basée sur le modèle ou post-hoc), cela n'est pas suffisant qu'elle possède une grande précision : l'information extraite doit aussi être pertinente. Cette notion est définie dans l'article [52] par : *une interprétation est dite pertinente si elle fournit des informations pour un public particulier et un domaine choisi.* Un exemple, dans le contexte de la génomique, les interprétations souhaitées ne seront pas les mêmes si le public visé est un patient, un biologiste ou un statisticien par exemple. La pertinence peut alors jouer un rôle majeur pour réaliser le compromis entre la précision descriptive et prédictive. s'il s'agit d'un problème pour lequel l'éthique et la justesse des décisions prises est importante : la précision prédictive sera privilégiée. Inversement, si l'algorithme mis en place a seulement pour but d'améliorer les performances sans avoir besoin de comprendre le cheminement qui a mené à ces prédictions, la précision prédictive sera favorisée.

A présent, détaillons les différentes formes d'interprétabilité, l'une basée sur le modèle et l'autre post-hoc.

2.3.3 Interprétabilité basée sur le modèle

L'interprétabilité basée sur le modèle (notée IBM), qui se situe au niveau de la phase *Model* sur le graphique 2.3, se définit comme la construction de modèles qui fournissent facilement un aperçu des relations qu'ils ont apprises. Différentes méthodes d'IBM offrent la possibilité d'augmenter la précision descriptive en construisant des modèles plus faciles à comprendre mais peut entraîner une perte de précision prédictive. Ceci nous conduit au compromis de l'inteprétabilité basée sur le modèle, entre un modèle assez simple pour être compréhensible par le public visé et un modèle assez sophistiqué pour refléter correctement le phénomène (éventuellement complexe) modélisé. Si la précision prédictive est

trop basse (i.e. le modèle choisi possède un faible pouvoir prédictif) alors toute l'analyse qui en découlera sera jugée "suspecte" ou inadaptée. Essayons de définir différentes notions pour juger de l'IBM d'un modèle. Pour cela, posons-nous la question suivante : les modèles jugés simples et interprétables (comme notamment les arbres de décision ou les GLM) par la majorité sont-ils nécessairement si faciles à interpréter ? Prenons l'exemple d'une tarification automobile classique, construit à partir d'un GLM. Les variables utilisées dans la modélisation de la prime pure sont discrétisées (comme l'âge du conducteur par exemple) et le modèle final peut posséder plus de 200 coefficients. On peut se demander alors s'il est réellement plus facile d'interpréter un tel modèle GLM par rapport à un XGBoost, considéré comme une boîte noire. Cette interrogation soulevée nous conduit au premier type d'IBM, appelé la parcimonie.

2.3.3.1 Parcimonie

La parcimonie est étroitement liée au principe du rasoir d'Ockham, qui stipule que "les multiples ne doivent pas être utilisés sans nécessité". Dans le cas d'un modèle de machine learning, imposer que le modèle soit parcimonieux revient à limiter le nombre de paramètres non nuls. Quand le nombre de paramètres non nuls est suffisamment petit, le praticien peut alors interpréter les variables correspondantes aux paramètres comme étant reliés à la sortie et peut également interpréter l'amplitude et le signe de ces paramètres. L'article [52] rappelle que pour interpréter un modèle parcimonieux, il est nécessaire de s'assurer de la stabilité des coefficients obtenus. Pour se faire, de légères perturbations du dataset initial peuvent être appliquées afin de vérifier que les coefficients obtenus sont sensiblement proches aux précédents. Pour mettre en place la parcimonie, les outils généralement utilisés sont la régularisation (de type Lasso par exemple) ou les indices AIC et BIC, qui pénalisent les modèles complexes.

La parcimonie (stable) peut alors présenter trois avantages : en réduisant le nombre de paramètres, les modèles parcimonieux sont plus faciles à comprendre ce qui fournit une précision descriptive plus élevée. De plus, elle peut permettre d'atteindre une précision prédictive plus élevée, notamment en évitant le sur-apprentissage. Enfin, un modèle parcimonieux fournit des informations plus pertinentes.

Un exemple montrant l'importance de la parcimonie est détaillé dans l'article [52]. Il s'agit de l'identification d'interactions entre différentes biomolécules dans le domaine de la génomique. Dans ce contexte, les bases de données à disposition sont généralement massives et peuvent posséder des millions de colonnes. Appliquer des pénalités de parcimonie s'avère alors essentiel pour que les biologistes puissent identifier les candidats prometteurs. Une étude a montré que l'utilisation de modèle parcimonieux pour l'identification d'interactions entre les variables décrivant le génome des drosophiles permettait à la fois d'obtenir des résultats plus stables (avec des hyperparamètres robustes aux perturbations) et également d'identifier les interactions majeures, contenant assez peu de variables pour être analysées par un humain.

2.3.3.2 Simulabilité

L'article [52] dit qu'un modèle est simulable si un humain, à qui on demande l'interprétation, est capable de simuler le process de décision global de l'algorithme. L'article [47] souligne que la simulabilité réfère à une transparence totale du modèle : un humain devrait être capable, à partir des inputs et des paramètres du modèle, de réaliser l'ensemble des calculs, en temps raisonnable, pour donner la prédiction faite par le modèle. En ce sens, les arbres de décision sont généralement cités comme des algorithmes simulables, étant donné leur architecture pour la prise de décision. De même, les règles de décision sont rangées dans cette catégorie. En plus de la simplicité du modèle sous-jacent, d'autres contraintes doivent être placées, notamment sur le nombre de paramètres, ce qui rejoint le principe de parcimonie évoqué juste avant. En effet, un modèle linéaire parcimonieux semble bien plus facile à interpréter qu'un modèle avec des centaines de coefficients. En définissant la notion d'interprétabilité par le biais du terme "raisonnable" on introduit de la subjectivité. Cependant, étant donné les limites de la capacité cognitive humaine, cette ambiguïté ne couvre seulement que quelques ordres de grandeur. Ainsi, l'article [47] suggère que ni les modèles linéaires, ni les règles de décisions ou les arbres de décisions ne sont réellement intrinsèquement interprétables. Par exemple, des règles de décision trop lourdes ou des arbres de décisions trop profonds pourraient alors être considérés comme moins transparents que des réseaux de neurones relativement compacts. On peut citer également les modèles GLM, qui sont couramment déployés en actuariat, notamment en tarification comme nous le verrons dans la partie application de ce mémoire. Néanmoins, ces derniers peuvent posséder plus de 200 coefficients pouvant rendre leur interprétation compliquée.

2.3.3.3 Modularité

L'article [52] définit un modèle de machine learning modulaire si une portion significative du process de prédiction peut être interprétée indépendamment. Ainsi, un modèle modulaire ne sera pas aussi facile à comprendre qu'un modèle parcimonieux ou simulable mais peut s'avérer très utile pour augmenter la précision descriptive en fournissant des relations apprises par l'algorithme. Un exemple classique de modèle considéré comme modulaire est la famille des GAM (modèles additifs généralisés), dont les GLM sont une sous-famille. Dans ce type de modèles, la relation entre les variables est forcément additive et les coefficients trouvés permettent une interprétation relativement facile du modèle. Par opposition, les réseaux de neurones profonds sont eux considérés comme peu modulaires, étant donné le peu d'informations fournis par les coefficients de chaque couche. L'exemple cité précédemment sur la prédiction des décès à cause de la pneumonie montre l'intérêt de la modularité pour produire des interprétations pertinentes, pour détecter des biais dans la base d'apprentissage. En effet, en analysant le modèle (GAM avec interaction) mis en place, les chercheurs ont trouvé des propriétés contre-intuitives, avec notamment un risque de décès moins élevé lorsque le patient est atteint d'asthme. Cela venait du fait que les patients atteints d'asthme recevaient un traitement plus agressif, réduisant le risque de mortalité, ce qui introduit un biais dans la base d'apprentissage.

Sans cette analyse du modèle, les médecins auraient eu tendance à ne pas privilégier les patients asthmatiques ce qui aurait augmenter leur risque de mourir.

2.3.4 Interprétabilité Post-Hoc

L'interprétabilité post-hoc, à la différence de l'interprétabilité basée sur le modèle, correspond à l'analyse une fois que le modèle ait été ajusté dans le but de fournir des informations sur les relations apprises. Il s'agit de la forme d'interprétabilité que l'on étudiera en majorité au cours de ce mémoire. Ce type d'interprétations s'avère particulièrement important lorsque les coefficients du modèle ne montrent pas clairement ce que celui-ci a appris (modèle non modulaire). Pour se faire, de nombreux méthodes d'interprétabilité existent et certaines seront détaillées dans les parties suivantes. Comme le modèle a déjà été ajusté à ce stade, la précision prédictive est fixée. Ainsi, dans le cadre du PDR, seules la précision descriptive et la pertinence doivent être considérées. L'article [52] propose de classer les méthodes d'interprétation post-hoc en deux catégories : l'interprétation au niveau du jeu de données et l'interprétation au niveau de la prédiction. La deuxième se concentre sur l'explication des prédictions individuelles faites par le modèle, en analysant par exemple les variables ou les interactions qui ont mené à une telle prédiction. La première s'intéresse principalement aux relations globales apprises par le modèle, en extrayant par exemple une représentation graphique associée à une certaine prédiction. Ces deux types de méthodes sont étroitement liés : l'approche au niveau du jeu de données fournit généralement des informations au niveau de la prédiction.

2.3.4.1 Interprétation au niveau du jeu de données

L'interprétation au niveau du jeu de données est utilisée lorsque nous nous intéressons aux relations générales apprises par le modèle, c'est-à-dire aux relations pertinentes pour une classe particulière de réponses ou une sous-population.

Interaction et importance des variables

Le score d'importance d'une variable, au niveau du jeu de données, estime la contribution d'une variable, à travers le dataset entier, concernant une prédiction. Ces scores permettent d'extraire des informations sur le modèle et notamment sur les variables qu'il a identifié comme importantes selon la sortie. De nombreuses recherches ont été faites concernant l'importance des variables. L'une des premières, par Breiman et Al. (2001), portait sur les modèles utilisant des arbres, comme les forêts aléatoires, comme nous le verrons dans la partie sur les arbres de décision du chapitre 3). Depuis, d'autres méthodes ont été développées pour les réseaux de neurones par Olden et AL. (2004) mais également des méthodes indépendantes du modèle ont été mises en place, comme le souligne l'article [25], détaillé dans la suite du mémoire, notamment dans le chapitre 4.

Les scores d'importance ont été suivis par des mesures de l'interaction entre les variables. Celles-ci semblent primordiales pour interpréter un modèle de machine learning étant donné que les relations sont pour la plupart non linéaires avec de nombreuses

interactions complexes. Comme pour le score d'importance, il existe des méthodes dépendantes du modèle, notamment pour les forêts aléatoires et les réseaux de neurones, mais également des méthodes agnostiques comme la H-statistique de Friedman.

Importance statistique des variables

Dans certains cas, il est possible de calculer des mesures statistiques de confiance comme les scores d'importance de variables. Pour cela, il est nécessaire de réaliser des hypothèses à propos de la structure des données, comme pour les modèles linéaires ou la régression logistique par exemple, pour lesquels on peut citer la mesure de la t-statistique, qui est approfondie dans le chapitre 3. Grâce à ces hypothèses, on peut tester si les coefficients calculés au sein du modèle sont réellement significatifs ou non. Tout ces résultats peuvent être utilisés pour l'interprétation du modèle à condition que toutes les hypothèses sous-jacentes à celui-ci soient vérifiées.

Visualisation

Les études de machine learning utilisent couramment un grand nombre de variables explicatives. Des données multi-dimensionnelles rendent très difficiles la compréhension des relations complexes entre les variables apprises par le modèle. C'est pourquoi disposer d'outils de visualisation résumant ce que le modèle a appris est très intéressant dans le domaine de l'interprétabilité. De nombreuses méthodes ont été développées par les chercheurs, notamment avec des graphiques de dépendance partielle (PDP) ou des effets locaux accumulés (ALE), développés dans le chapitre 4.

Analyse des résidus et des points aberrants

Pour interpréter les résultats d'un modèle de machine learning, on regarde généralement la précision moyenne, comme le taux d'erreur ou le MSE. Cependant, on ne considérant que cette valeur, on perd des informations sur notre modèle : il peut être intéressant d'analyser la distribution des prédictions et des erreurs. Dans la partie application du chapitre 5, on propose d'autres critères tels que les MSE locaux, calculés sur des assurés bien précis et non la base de test toute entière. Le graphique des résidus permet d'identifier l'hétérogénéité des prédictions, notamment les points aberrants ou les points mal prédits par le modèle. Cette analyse est possible à l'aide du package $DALEX$, disponible sous R ; son contenu est détaillé dans l'annexe E. Une autre méthode, construit à partir des fonctions d'influence, qui sont détaillées en annexe B.5, permet d'identifier efficacement les points mal classifiés (c.f 41). A l'aide de celle-ci, il est alors possible d'améliorer la précision sur la base de test.

2.3.4.2 Interprétation au niveau de la prédiction

Une autre approche de l'interprétation post-hoc repose sur l'analyse des prédictions individuelles réalisées par le modèle.

Score de contribution des variables dans la prédiction

La technique d'interprétation au niveau de la prédiction la plus utilisée est l'assignation de scores de contribution (ou d'importance) aux différentes variables. Plus le score de contribution (en valeur absolue) est élevé, plus la variable aura contribué, négativement ou positivement suivant le signe, à la prédiction réalisée. De nombreuses approches reposant sur cette méthode ont été proposées agnostiques ou non au modèle. Les plus couramment mentionnées sont la méthode LIME, qui ajuste un modèle linéaire (LASSO) au niveau d'une observation, et la méthode Shap, qui utilise la théorie des jeux pour calculer une contribution "juste" de chaque variable dans la prédiction d'une instance. Celles-ci sont détaillés dans le chapitre 4 consacré aux méthodes d'interprétations des modèles post-hoc et agnostiques aux modèles considérés.

Alternatives aux scores d'importance

Les méthodes par calcul de contribution des variables sont les plus couramment citées lorsque l'on étudie l'interprétabilité au niveau d'une prédiction. Il en existe néanmoins d'autres, à savoir les explications basées sur les exemples -comme les explications contrefactuelles et les exemples contradictoires - ou les fonctions d'influence (détaillées en annexe B.5 et B.4). Celles-ci peuvent être utilisées pour détecter les faiblesses d'un modèle et pour les corriger.

Chapitre 3

Modèles couramment utilisés en Data Science et en Actuariat

3.1 Des exemples de modèles nativement interprétables

Ce chapitre a pour objectif de présenter différents modèles d'apprentissage statistique très répandus et utilisés en actuariat. Nous étudierons tout d'abord les modèles linéaires, dont la régression linéaire et le GLM, puis nous verrons les arbres de décision avant d'expliquer le principe de l'algorithme XGBoost. Cette partie permettra de mieux appréhender les outils utilisés au cours de l'application du chapitre 5, au cours duquel nous mettrons en place un GLM, dans le cadre de la tarification de l'assurance automobile, qui s'avère être le modèle le plus classique. Nous comprendrons à l'aide de cette partie pourquoi le GLM est tant utilisé et considéré interprétable, selon les critères définis dans le chapitre 2, à savoir la parcimonie, la modularité et la simulabilité. Au cours de notre application actuarielle, nous mettrons également en place un XGBoost, considéré comme une boîte-noire. Cette partie permettra de comprendre son fonctionnement et également ce qui le rend si complexe à expliquer et donc la nécessité d'utiliser les outils du chapitre 4, permettant l'interprétabilité de n'importe quel modèle d'apprentissage statistique.

3.1.1 Régression Linéaire

3.1.1.1 Principe général de la régression linéaire

Décrivons succinctement le modèle de régression linéaire, couramment utilisé en actuariat notamment pour sa simplicité. Ce modèle se place dans le cas d'une variable quantitative à expliquer, à partir de variables explicatives (réelles ou catégorielles). Soit y une variable quantitative que l'on cherche à prédire, et $(x_i)_{1\leq i\leq p}$ p variables explicatives, supposées quantitatives pour le moment. L'objectif est de trouver des coefficients constants $\beta = (\beta_i)_{0\leq i\leq p}$ de sorte à ce que $\beta_0 + \sum_{i=1}^{p} \beta_i x_i$ approxime au mieux la variable y. Ainsi le modèle se présente sous la forme : $y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \epsilon$ où ϵ est l'erreur commise

par le modèle, c'est-à-dire la différence entre la prédiction et la véritable valeur prise par y. On suppose généralement que ces erreurs suivent une loi normale centrée, de variance constante (homoscédastique) σ^2. Soit $Y = (y^{(j)})_{1\leq j\leq n}$ et $X = (x_i^{(j)})_{1\leq j\leq n, 1\leq i\leq p}$ nos données pour nos n individus. Sous forme matricielle, notre modèle de régression linéaire se présente sous la forme : $Y = X\beta + \epsilon$. On trouve les coefficients $(\beta)_{0\leq i\leq p}$ du modèle par la méthode des moindres carrés, i.e. en minimisant l'erreur commise par le modèle.

$$\beta = \underset{(\beta_i)_{0\leq i\leq p}}{argmin} \sum_{j=1}^{n} \left(y^{(j)} - \beta_0 - \sum_{i=1}^{p} \beta_i x_i^{(j)} \right)^2 \tag{3.1}$$

On trouve finalement [3] :

$$\hat{\beta} = (X^t X)^{-1} X^t Y \tag{3.2}$$

On note par la suite : $\hat{y}^{(j)} = \sum_{i=1}^{p} \beta_i x_i^{(j)},\ j \in \{1, ..., n\}$ la prédiction faite par le modèle linéaire pour $y^{(j)}$. Nous prendrons cet exemple, afin d'interpréter notre modèle : on suppose que la variable y à expliquer est le prix d'une maison (en euros), et que l'on a deux variables explicatives x_1 et x_2 correspondant respectivement à la superficie en mètres carrés de la maison et à la présence ou non d'un jardin (valant 1 si il y en a un, 0 sinon). Ainsi x_1 est quantitative (ou numérique) et x_2 qualitative.

3.1.1.2 Interprétation du modèle de régression linéaire

Comme il a été indiqué en introduction, le modèle de régression linéaire est abondamment utilisé pour sa simplicité, mais également pour sa facilité d'interprétation. En plus de la parcimonie possible à l'aide de la régularisation, les critères de modularité et de simulabilité détaillés dans le chapitre 2 pour définir un modèle intrinsèquement interprétables. On utilise notamment les arguments qui suivent comme preuve de la lisibilité et transparence du modèle [3] :

- Pour les variables explicatives numériques : l'augmentation d'une unité change l'estimation finale du poids associé à cette variable. Dans notre exemple, si la superficie augmente d'un mètre carré, alors l'estimation de prix de la maison augmentera de β_1 euros.
- Pour les variables explicatives qualitatives (binaires) : le passage d'une catégorie à l'autre change l'estimation finale du poids associé à la variable considérée. Dans notre exemple, si la maison possède un jardin, son prix estimé augmentera de β_2 par rapport à l'estimation da la même maison sans jardin. Dans le cas de variables qualitatives à L modalités ($L > 2$), il faut recoder notre variable en L-1 colonnes contenant que des 0 si il s'agit de la première modalité, et contenant un unique 1 sur la colonne correspond à la modalité de la variable, et des zéros ailleurs.
- L'intercept β_0 représente la valeur de y estimée par notre modèle lorsque toutes les variables explicatives sont à 0. L'intercept n'a en général pas de sens et ne peut être interprété que si les variables explicatives ont été centrées réduites. Dans ce cas-ci, il représente la variable estimée de y lorsque toutes les variables explicatives sont à leur valeur moyenne.

D'autres éléments d'interprétation du modèle linéaire existent à savoir :

— Le R^2 (ou R^2 ajusté) : il mesure la quantité de variance expliquée par le modèle. Pour le calculer, nous devons d'abord définir les termes SSE (somme des carrés des résidus) et SST (somme totale des carrés) :

$$SSE = \sum_{j=1}^{n} (y^{(j)} - \hat{y}^{(j)})^2 \; et \; SST = \sum_{j=1}^{n} (y^{(j)} - \bar{y})^2 \tag{3.3}$$

avec $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ la moyenne empirique des sorties observées. Alors le coefficient R^2, appelé coefficient de détermination se calcule par :

$$R^2 = 1 - \frac{SSE}{SST} = \frac{SST - SSE}{SST} \tag{3.4}$$

Le R^2 est compris entre 0 et 1 : plus il est proche de 0, moins le modèle expliquera les données. Inversement, plus il est proche de 1, plus le modèle ajuste correctement les données. Le problème de ce coefficient est qu'il est croissant par rapport au nombre p de paramètres dans le modèle, même si les paramètres ajoutés ne contiennent aucune information sur la sortie. C'est pourquoi l'outil le plus couramment utilisé est le R^2 ajusté, noté R^2_{adj}, qui pénalise les modèles avec beaucoup de paramètres. Celui-ci est défini par :

$$R^2_{adj} = R^2 - (1 - R^2)\frac{p}{n - p - 1} \tag{3.5}$$

— L'importance des variables. Cette notion sera également vue dans la suite du mémoire, d'une manière différente, à savoir par un calcul indépendant du modèle considéré. Dans le cas de la régression linéaire, l'importance d'une variable x_j est définie par la t-statistique : $t_{\beta_j} = |\frac{\beta_j}{\sigma_{\beta_j}}|$, rapport entre le poids estimé et son écart-type. Ainsi plus le poids d'une variable est grand, plus la variable est importante. De même, plus la variance du poids estimé est faible, c'est-à-dire plus on est sûr de sa valeur réelle, plus la variable associée est importante.

3.1.1.3 Régularisation

La régularisation consiste à ajouter un terme de pénalité aux valeurs fortes des paramètres afin d'éviter le surapprentissage [53]. Elle permet également de rendre les modèles parcimonieux, en imposant un nombre de coefficients nuls, facilitant ainsi leur interprétation. Rappelons que la parcimonie est l'un des critères permettant de considérer un modèle interprétable, comme nous l'avons vu au cours du chapitre 2. Dans le cas de la régression notre fonction de coût L à minimiser est l'erreur quadratique, à savoir :

$$\forall (y, \hat{y}) \in \mathbb{R}^2, \; L(y, \hat{y}) = \frac{1}{2n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 = \frac{1}{2n} \sum_{i=1}^{n} (y^{(i)} - \sum_{j=1}^{p} \beta_j x_j^{(i)})^2 \tag{3.6}$$

La régularisation va ajouter un terme de correction dans la fonction de coût, qui est proportionnel à la norme des coefficients. On distingue notamment la régularisation de type Lasso (Least Absolute Shrinkage and Selection Operator) lorsque la norme utilisée est $\mathbb{L}^1$ et la régularisation ridge, lorsqu'il s'agit de la norme $\mathbb{L}^2$: Les fonctions de coût dans les deux cas sont alors :

— Régularisation $\mathbb{L}^1$ (Lasso) :

$$\forall (y,\hat{y}) \in \mathbb{R}^2,\ L(y,\hat{y}) = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)}-\hat{y}^{(i)})^2+\lambda||\beta||_1 = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)}-\sum_{j=1}^{p}\beta_j x_j^{(i)})^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

— Régularisation $\mathbb{L}^2$ (Ridge) :

$$\forall (y,\hat{y}) \in \mathbb{R}^2,\ L(y,\hat{y}) = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)}-\hat{y}^{(i)})^2+\lambda||\beta||_2 = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)}-\sum_{j=1}^{p}\beta_j x_j^{(i)})^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

Le paramètre $\lambda \in \mathbb{R}$ introduit dans les formules précédentes est un paramètre de contrôle sur la force de la régularisation. Il est souvent choisi à l'aide de la méthode de validation croisée. Plus λ sera grand, plus on aura de paramètres nuls.

La régularisation a été ici présentée pour la régression linéaire, mais elle s'applique à la majorité des algorithmes de Machine Learning, notamment ceux qui utilisent une fonction de coût.

La figure 3.1 montre la représentation usuelle des régularisations $\mathbb{L}^1$ et $\mathbb{L}^2$ dans le cas à deux dimensions, en affichant les points de normes 1 dans $\mathbb{R}^2$. Des études ont montré que la régularisation $\mathbb{L}^1$ favorisait les modèles parcimonieux, en fixant de nombreux coefficients nuls et en les rendant plus lisibles et interprétables [53].

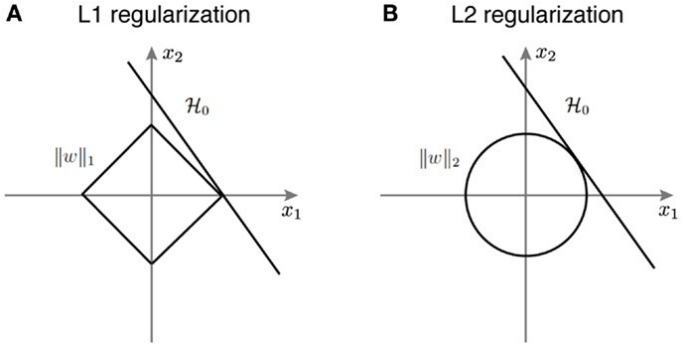

FIGURE *3.1: Régularisation* $\mathbb{L}^1$ *et* $\mathbb{L}^2$

3.1.2 Modèle linéaire généralisé (GLM)

3.1.2.1 GLM

Principe du GLM

Nous avons vu précédemment le modèle de régression linéaire qui consiste à approcher

une valeur cible réelle par une somme pondérée des caractéristiques. Le GLM repose sur une idée similaire, en supposant également que la variable cible conditionnellement aux variables explicatives suit une certaine loi. Le GLM est mis en place pour des variables cibles dont le support est un sous-espace $\mathbb{E}$ de $\mathbb{R}$. Par exemple :

- $\mathbb{E} = \{0, 1\}$: on trouve la régression logistique, approfondie ci-après, pour laquelle on utilise une loi de Bernouilli.
- $\mathbb{E} = \{0, ..., K\}$: on parle de régression multinomiale, pour laquelle on utilise une loi binomiale. ($K \in \mathbb{N}$)
- $\mathbb{E} = \mathbb{R}^+$, on utilise généralement une loi exponentielle, une loi gamma ou une loi inverse-normale.
- $\mathbb{E} = \mathbb{N}$, on utilise une loi de Poisson.

Le principe du GLM est alors le suivant : à partir des variables explicatives $X = (X_1, ..., X_p)$, on suppose que Y conditionnellement à X suit une certaine loi dépendant de la combinaison linéaire des X_i. On relie alors les paramètres à l'aide de la fonction lien Φ par la formule :

$$\Phi(\mathbb{E}[Y|X]) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j \tag{3.7}$$

A partir de ces définitions, nous pouvons trouver les paramètres β optimaux par maximum de vraisemblance.

Interprétation du modèle GLM

De même que pour la régression linéaire, nous pouvons facilement interpréter ce modèle, et notamment l'impact d'un changement d'une unité d'une caractéristique sur la prédiction. Cette facilité de compréhension font que les critères de modularité et simulabilité, détaillés dans le chapitre 2, sont respectés, rendant le modèle intrinsèquement interprétable. De plus, le critère de parcimonie peut être obtenu, en réalisant une sélection de variables, une fois le modèle complet ajusté, notamment à l'aide de critères comme l'AIC permettant de pénaliser les modèles trop complexes. Ce dernier permettra de savoir si le gain (éventuel) obtenu par le nouveau modèle est suffisant pour compenser la complexité induite par l'ajout d'un coefficient. Il participe au compromis précision interprétabilité vu dans le chapitre 1.

Considérons la prédiction y faite pour un certain $x = (x_1, ..., x_p) \in \mathbb{R}^p$. On a :

$$y = \Phi^{-1}(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)$$

Donc, notant $\tilde{x}$ le même vecteur que x sauf pour un certain $j \in \{1, ..., p\}$ où $\tilde{x_j} = x_j + 1$, i.e. :

$$\forall i \in \{1, ..., p\}, \tilde{x_l} = x_l + \mathbb{1}_{\{j=l\}}$$

Alors la prédiction $\tilde{y}$, faite pour $\tilde{x}$ est :

$$\tilde{y} = \Phi^{-1}(\beta_0 + \sum_{l=1}^{p} \beta_l x_l + \beta_j)$$

Donc l'écart de prédiction faite par le modèle lorsque la variable x_j augmente de une unité est :

$$\tilde{y} - y = \Phi^{-1}(\beta_0 + \sum_{l=1}^{p} \beta_l x_l + \beta_j) - \Phi^{-1}(\beta_0 + \sum_{l=1}^{p} \beta_l x_l)$$

Dans les cas usuels suivants, on trouve :

— Si $\Phi = Id$ (fonction identité) alors : $\tilde{y} - y = \beta_j$ (effet additif)
— Si $\Phi = log$ alors : $\frac{\tilde{y}}{y} = e^{\beta_j}$ (effet multiplicatif)

3.1.2.2 Un cas particulier du GLM : la régression logistique

Principe général

Nous avons vu ci-avant le modèle général du GLM, intéressons nous au cas où nous souhaitons prédire une variable binaire $Y \in \{0, 1\}$. Nous disposons de p variables explicatives $X = (X_1, ..., X_p)$. Notons $p(1|X) = p(Y = 1|X)$ la probabilité que la variable Y soit égale à 1 conditionnellement aux valeurs prises par les variables du vecteur X. L'idée de la régression logistique est de considérer que la combinaison des variables explicatives est fonction de cette probabilité, c'est-à-dire qu'on a la relation :

$$\Phi(p(1|X)) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i \tag{3.8}$$

où Φ est une fonction lien. Généralement en régression logistique, la fonction lien utilisée est *logit*, c'est-à-dire :

$$logit(p(1|X)) = ln(\frac{p(1|X)}{1 - p(1|X)}) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i \tag{3.9}$$

Il en découle la relation :

$$p(1|X) = \frac{1}{1 + exp(-\beta_0 - \sum_{i=1}^{p} \beta_i X_i)} \tag{3.10}$$

Finalement, la régression logistique revient à supposer que la loi de Y, conditionnellement à X, est une loi de Bernouilli de paramètre $p(Y = 1|X) : L(Y|X) \sim \mathbb{B}(p(Y = 1|X))$. En pratique lorsque nous disposons d'un échantillon de taille n $y = (y^{(j)})_{1 \leq j \leq n} \in \{0, 1\}^n$ et $x = (x_i^{(j)})_{1 \leq j \leq n, 1 \leq i \leq p} \in (\mathbb{R}^p)^n$, la méthode utilisée pour estimer les paramètres $(\beta_i)_{0 \leq i \leq p}$ est le maximum de vraisemblance. c'est-à-dire que l'on veut maximiser, par rapport aux paramètres $\beta_0, ..., \beta_p$, la fonction de vraisemblance suivante :

$$L(y, x; \beta) = \prod_{i=1}^{n} P[Y_i = 1|X_i = x^{(i)}]^{Y^{(i)}}(1 - P[Y_i = 1|X_i = x^{(i)}])^{1-Y^{(i)}} \tag{3.11}$$

$$= \prod_{i=1}^{n} \left(\frac{1}{1 + exp(-\beta_0 - \sum_{j=1}^{p} \beta_j x_j^{(i)})} \right)^{y^{(i)}} \left(1 - \frac{1}{1 + exp(-\beta_0 - \sum_{j=1}^{p} \beta_j x_j^{(i)})} \right)^{1-y^{(i)}} \tag{3.12}$$

On utilise généralement alors l'algorithme de descente du gradient (détaillé dans les parties suivantes) pour trouver les paramètres optimaux.

Interprétation du modèle de régression logistique

Nous voulons à présent pouvoir interpréter le modèle de régression logistique tout comme nous l'avons fait pour la régression linéaire. Cette partie sert à montrer que les critères de modularité et de simulabilité (vu dans le chapitre 2) sont respectés rendant le modèle intrinsèquement interprétables. Nous voyons tout de suite que l'interprétation des poids β_j sera différente étant donnée que la sortie est à présent une probabilité entre 0 et 1. De plus, du fait de la fonction lien *logit*, les poids n'affectent plus linéairement la sortie. Notons $odds_X = \frac{p(Y=1|X)}{p(Y=0|X)}$ le rapport des probabilités de Y conditionnellement à la valeur X des variables explicatives. Ce terme, couramment appelé l'odds, sera central pour l'interprétation du modèle. Ainsi la régression logistique s'écrit à présent : $ln(odds_X) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$. On retrouve le modèle de régression linéaire pour le terme $ln(odds_X)$. A présent, observons l'impact dans la prédiction du modèle, lorsque la variable x_j pour $j \in \{1, ..., p\}$ change d'une unité. Nous voyons que le rapport des termes $odds_X$ sera alors de e^{β_j}. Ainsi le changement d'une unité de x_j multiplie le ratio $odds_X$ par un facteur e^{β_j}. L'interprétation a été faite ici pour des variables numériques. Dans le cas de x_j variable catégorielle binaire, le changement de la variable x_j de la catégorie de référence à l'autre catégorie change l'estimation du ratio $odds_X$ par un facteur e^{β_j}. De même que pour la régression linéaire, l'intercept β_0 n'a pas réellement d'interprétation. Il correspond à la valeur du log-ratio $log(odds_X)$ lorsque les variables numériques sont à zéro et les variables catégorielles dans leur classe de référence. Ainsi, nous observons que la régression logistique s'interprète facilement, tout comme la régression linéaire, ce qui en fait un modèle très couramment utilisé de par sa simplicité et son interprétabilité.

3.1.3 Arbres de décision

Dans cette partie, nous détaillons une famille d'algorithmes également très répandue, à savoir les arbres de décision, pouvant servir à la fois en classification et en régression. Cette partie permettra dans un premier temps de comprendre comment les arbres sont formés et ce qui les rend lisibles et interprétables, selon les critères du chapitre 2. De plus,

l'arbre de décision est l'élément de base du modèle plus complexe XGBoost, détaillé dans la partie suivante, qui nous servira au cours de l'application actuarielle du chapitre 5.

3.1.3.1 Algorithme CART

Pour expliquer les arbres de décision de manière globale, nous nous plaçons dans le cas simplifié d'arbres de classification binaires. La méthode présentée se généralise pour les arbres de régression. Nous présentons l'algorithme le plus couramment utilisé, c'est-à-dire celui présenté par Breiman et Al. en 1984 [14].
Soit $Y \in \{1, ..., K\}$ la variable catégorielle que l'on cherche à expliquer et $X \in \mathbb{R}^p$ ($p, K \in \mathbb{N}$). On suppose que l'on dispose d'un échantillon : $y = (y_1, ..., y_n) \in \mathbb{R}^n$ correspondant aux valeurs cibles et $x = (x_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$ correspondant aux variables explicatives (avec $n \in \mathbb{N}$ le nombre d'individus).

Principe général

Un arbre binaire est obtenu en séparant les individus successivement en deux sous-ensembles. L'objectif à chaque étape est de trouver la variable explicative qui sépare le mieux possible les données en deux sous-groupes. On s'arrête lorsque une nouvelle division ne nous apporte pas d'amélioration ou lorsqu'un sous-groupe est trop petit. On note déjà que la construction d'un arbre nécessite : un processus de sélection des divisions successives de l'arbre, d'un critère pour décider de réaliser une nouvelle coupe ou de considérer le noeud comme terminal et d'un processus d'attribution de classe à chaque noeud terminal (ou feuille).

Notations utilisées

Nous utiliserons pour la suite les notations suivantes :

- I l'ensemble des individus et $i \in I$ un individu.
- C une partition de l'échantillon, et $t \in C$ un noeud
- $N(t)$: le nombre d'individus dans un noeud t.
- $N_k(t)$: le nombre d'individus de classe k dans le noeud t.
- $p(k|t) = \frac{N_k(t)}{N(t)}$ la proportion d'individus de classe k dans le noeud t.
- $y(t)$ la prédiction faite pour les individus du noeud t : on choisit la classe majoritaire, i.e : $y(t) = j <=> \forall k \in K, p(k|t) \leq p(j|t)$ (en régression on utiliserait une moyenne pondérée).

Construction de l'arbre maximal

La première partie de l'algorithme CART consiste à créer un arbre, dit maximal.

Fonction d'impureté : Définissons tout d'abord la notion d'impureté. Un noeud est dit pur si une seule classe est représentée dans le noeud. L'objectif est d'introduire une fonction qui renvoyant un réel positif, donnant le degré d'impureté du noeud. Ainsi pour un noeud t donné, on définit la fonction Imp telle que :

$$\phi : [0, 1]^K \to \mathbb{R}, Imp(t) = \phi(\{p(k|t), k \in K\}) \tag{3.13}$$

Cette fonction ϕ, positive, doit vérifier les conditions suivantes :

- elle prend son maximum uniquement lorsque les classes sont équiréparties, c'est-à-dire au point : $(\frac{1}{K}, ..., \frac{1}{K})$
- elle prend son minimum lorsque le noeud est pur, c'est-à-dire pour tous les points contenant K-1 zéro et un unique 1.
- elle est symétrique, c'est-à-dire qu'elle ne privilégie pas une classe plus qu'une autre

L'exemple le plus classique d'impureté est la fonction d'entropie définie par :

$$Ent(t) = \sum_{k \in K} p(k|t)\ log\ p(k|t) \tag{3.14}$$

$$Gini(t) = \sum_{k=1}^{K} p(k|t)(1 - p(k|t)) \tag{3.15}$$

Division d'un noeud : Soit $x_j (j \in \{1, ..., p\})$la variable que l'on souhaite utiliser pour créer une coupe à partir d'un noeud t donné, pour former deux sous-noeuds : t_G et t_D. Dans le cas de variables quantitatives,on chercherait un réel z qui sépare au mieux le noeud en deux sous-noeuds : $t_G(z) = \{w \in t, x_j(w) \leq z\}$ et $t_D(z) = \{w \in t, x_j(w) > z\}$. On cherche z qui minimise la fonction :

$$z \mapsto \Delta Imp(d_z(t), t) = Imp(t) - \frac{N(t_G(z))}{N(t)} Imp(t_G(z)) - \frac{N(t_D(z))}{N(t)} Imp(t_D(z)) \tag{3.16}$$

où $d_z(t)$ est la division du noeud t associée au réel z. On cherche à maximiser l'homogénéité des noeuds obtenus, un noeud étant parfaitement homogène s'il ne contient que des individus de la même classe. Dans le cas de la régression, on cherche à créer des découpes qui tendent à diminuer la variance des noeuds. Nous disposons à présent de l'algorithme pour créer l'arbre maximal : à partir de la racine de l'arbre (un arbre vide initialement), on réalise une découpe en utilisant le critère de division de noeud vu précédemment. Sur chacun des sous-noeuds ainsi formés on met en place le processus, et ainsi de suite. Deux conditions supplémentaires sont ajoutées pour fixer l'arrêt de l'algorithme :

- On introduit un critère d'arrêt qui impose de ne pas découper un noeud qui contient moins d'un certain nombre d'individu.
- On ne découpe pas un noeud pur, c'est-à-dire qu'un noeud pur est nécessairement une feuille de l'arbre.

Ainsi à la fin de cet algorithme, on a créé l'arbre maximal noté T_{max}. Cependant, ce n'est pas nécessairement cet arbre que l'on va utiliser pour la prédiction, mais un sous-arbre obtenu en élaguant T_{max}.

Élagage de l'arbre maximal

Présentons à présent comment se réalise l'élagage de l'arbre T_{max} construit précédemment.

Taux d'erreur : Définissons pour cela le taux d'erreur, qui sera une notion fondamentale pour comprendre l'élagage. Au sein d'un noeud t, on le définit ainsi :

$$R(t) = \sum_{k \in K, k \neq y(t)} p(k|t) \tag{3.17}$$

Il s'agit de la proportion d'individus présents dans le noeud t qui n'ont pas la même valeur cible que la prédiction faite par le noeud t.

On peut alors généraliser cette définition à un arbre et non plus à un unique noeud. Pour cela, on note τ l'ensemble des noeuds terminaux (i.e feuilles) de l'arbre T. On définit alors le taux d'erreur de cet arbre par la formule :

$$R(T) = \sum_{t \in \tau} \frac{N(t)R(t)}{n} = 1 - \frac{1}{n} \sum_{t \in \tau} N_{y(t)}(t) \tag{3.18}$$

Il s'agit de la somme des taux d'erreurs de chaque noeuds terminaux, pondérés par la proportion d'individus au sein de ce noeud.

Paramètre de complexité et taux d'erreur Le taux d'erreur précédemment introduit ne prend pas en compte la complexité de l'arbre. En général, on cherche à ne pas avoir un arbre trop profond, pour éviter le surapprentissage, c'est pourquoi on introduit un terme de pénalité pour favoriser les arbres peu complexes. On note $\alpha \in \mathbb{R}^+$ le paramètre de complexité, et on appelle taux d'erreur pénalisé d'ordre α, la fonction R_α définie pour un arbre T par :

$$R_\alpha(T) = R(T) + \alpha|T| \tag{3.19}$$

Ainsi, ce nouveau taux pénalise les arbres avec beaucoup de feuilles (paramètre quantifiant la complexité du modèle).

Choix de division ou de noeud terminal A présent, il nous faut comprendre comment on élague l'arbre. Soit T un arbre, t un noeud de cet arbre, et $T(t)$ le sous-arbre de T issu du noeud t. On a nécessairement $R(T(t)) \leq R(t)$. Pour savoir si on élague, pour un paramètre de complexité α donné, l'élagage est utile si le taux d'erreur (pénalisé) du sous-arbre $T(t)$ est plus grand que le taux d'erreur (pénalisé) du noeud t, c'est-à-dire si : $R_\alpha(T(t)) \geq R_\alpha(t)$. De plus,

$$R_\alpha(T(t)) \geq R_\alpha(t) \iff R(T(t)) + \alpha|T(t)| \geq R(t) + \alpha \iff \alpha \leq \frac{R(t) - R(T(t))}{|T(t)| - 1}$$

On note la fonction introduite ci-dessus g : $g(t,T) = \frac{R(t)-R(T(t))}{|T(t)|-1}$. On appelle alors maillon faible t_{min}, le noeud qui minimise cette fonction : $t_{min} = \underset{t \in T}{argmin}\ g(t,T)$. Le paramètre de complexité choisi est alors : $\alpha = g(t_{min}, T)$.

Algorithme d'élagage La paragraphe précédent nous fournit alors un algorithme bottom-up d'élagage, qui va renvoyer une liste d'arbres élagués issus de T_{max} :

- Entrée : arbre maximal T_{max} construit précédemment
- Initialisation : $\alpha_0 = 0$, $T_0 = Tmax, k = 0$
- Itération : Tant que $|T_{max}| > 1$:
  - $t_{min}^{(k+1)} = \underset{t \in T_k}{argmin}\ g(t, T_k)$ (noeud, correspondant au maillon faible de l'arbre T_k)
  - $\alpha_{k+1} = g(t_{min}, T_k)$ (paramètre de complexité)
  - construction du nouvel arbre T_{k+1} à partir de T_k en élaguant toutes les branches issues de t_{min}^{k+1}
- Outputs : $\{\alpha_1, ..., \alpha_N\}$ et $\{T_{max}, T_1, ..., T_N\}$ avec $N \in \mathbb{N}$ le nombre d'itérations de l'algorithme.

L'arbre que l'on retient en général pour la prédiction, est l'arbre T_k pour lequel son taux erreur pénalisé $R_{\alpha_k}(T_k)$ est minimum.

3.1.3.2 Importance des variables

L'importance des variables a été l'une des premières méthodes faisant écho à l'interprétabilité et la transparence d'un modèle. En effet, une question que l'on se pose lorsque l'on ajuste un algorithme de machine learning est : quelles variables ont réellement servi pour la prédiction et quelles sont celles la plus grande contribution ? La réponse à cette question permet de mieux comprendre le modèle d'une part et de supprimer les variables jugées inutiles d'autre part, en rendant ainsi le modèle plus interprétable par le biais du critère de parcimonie, vu dans le chapitre 2. Brieman fut l'un des premiers à répondre à cette question d'importance des variables, pour le cas des arbres de décision [14]. Nous verrons dans le chapitre 4 une généralisation indépendante du modèle de l'importance des variables, appelée PFI, qui repose sur des permutations entre les différentes variables utilisées dans le modèle.

Division de substitution

Notons pour tout noeud t, $d^*(t)$ la division optimale à partir d'un noeud t, c'est-à-dire la division faite au niveau du noeud t au cours de l'algorithme de construction de l'arbre maximal. On note t_g et t_d les sous-noeuds ainsi obtenus. Définissons également pour tout $m \in \{1, ..., p\}$, $Dm(t)$ l'ensemble des divisions possibles à partir du noeud, en utilisant la variable explicative x_m pour séparer, et $\bar{D_m}(t)$ son complémentaire. Dans le cas de la régression, on aurait des divisions de la forme : $x_m \leq \alpha,\ \alpha \in \mathbb{R}$. Pour une division $d_m(t) \in D_m(t)$, on t'_g et t'_d les sous-noeuds ainsi obtenus. On note pour tout $j \in \{1, ..., K\}$, $N_j(t_g \cap t'_g)$ le nombre d'individus de la classe j du noeud t envoyés dans le sous-noeud de gauche par $d_m(t)$ et $d(t)^*$. On utilise les mêmes notations avec l'indice pour les individus envoyés à droite. Ainsi la probabilité que les divisions $d_m(t)$ et $d^*(t)$

envoient un individu du noeud t dans le sous-noeud de gauche est donnée par :

$$p_{gg}(d_m(t), d^*(t)) = \sum_{k=1}^{K} \frac{p(k|t)N_j(t_g \cap t'_g)}{p(t)N_j(t)} \tag{3.20}$$

On a la même définition pour les sous-noeuds droits $p_{dd}(d_m(t), d^*(t))$, avec les indices d à la place de g. On peut alors définir la probabilité que la division $d_m(t)$ prédise correctement la meilleure division $d^*(t)$, par : $p(d_m(t), d^*(t)) = p_{gg}(d_m(t), d^*(t)) + p_{dd}(d_m(t), d^*(t))$
La division de substitution $\tilde{d_m}(t)$ associée à une variable explicative $x_m,\ m \in \{1, ..., p\}$ est alors définie comme la division $d_m(t) \in D_m(t)$ qui prédit au mieux le partage fait par la division optimale d^*, c'est-à-dire :

$$\tilde{d_m}(t) = \underset{d_m(t) \in D_m(t)}{argmax}\ p(d_m(t), d^*(t)) \tag{3.21}$$

Définition de l'importance d'une variable

A l'aide des divisions de substitution introduites ci-avant, on peut alors définir l'importance I_{x_m} d'une variable $x_m,\ m \in \{1, ..., p\}$, associée à un arbre T par :

$$I(x_m) = \sum_{t \in T} \Delta Imp(\tilde{d_m}(t), t) \tag{3.22}$$

Il s'agit de la somme des diminutions de déviance provoquées à chaque noeud t de l'arbre, si on remplaçait la division optimale $d^*(t)$ par la division de substitution $\tilde{d_m}(t)$. On la normalise en général de sorte à avoir une valeur comprise entre 0 et 1, avec la variable la plus importante fixée à 1.

Stabilité de l'importance des variables CART

Des études empiriques ont été menées par Breiman (1996) et Ghattas (1999) pour montrer l'instabilité des arbres de classification et de régression par rapport à de faibles perturbations des données initiales. Pour étudier la stabilité de l'importance des variables, on peut utiliser des échantillons *bootstrap* des données d'origine et voir l'impact sur l'importance des variables données pour chaque algorithme CART ajusté. Ghattas a réalisé cette étude à partir d'une base de données de la station de Vitrolles, afin de prédire le maximum d'ozone dans la station. Les variables explicatives sont nombreuses (39) avec notamment la température, la nébulosité ou encore la quantité de SO2 au sein de la station. 40 échantillons *bootstrap* des données disponibles ont été mis en place, et pour chacun de ces échantillons, un arbre CART de taille fixe (10 feuilles) a été construit. Pour chaque arbre, l'importance des variables a été calculée, et l'étude de la distribution associée a été réalisée, sous la forme d'une boîte à moustaches, comme nous pouvons le voir sur la figure 3.2.

La variable représentant le maximum d'ozone observé la veille demeure la plus importante sur 39 des 40 échantillons créés, ce qui démontre une stabilité, malgré la perturbation des données. Ce n'est pas le cas pour les autres variables importantes telles que le

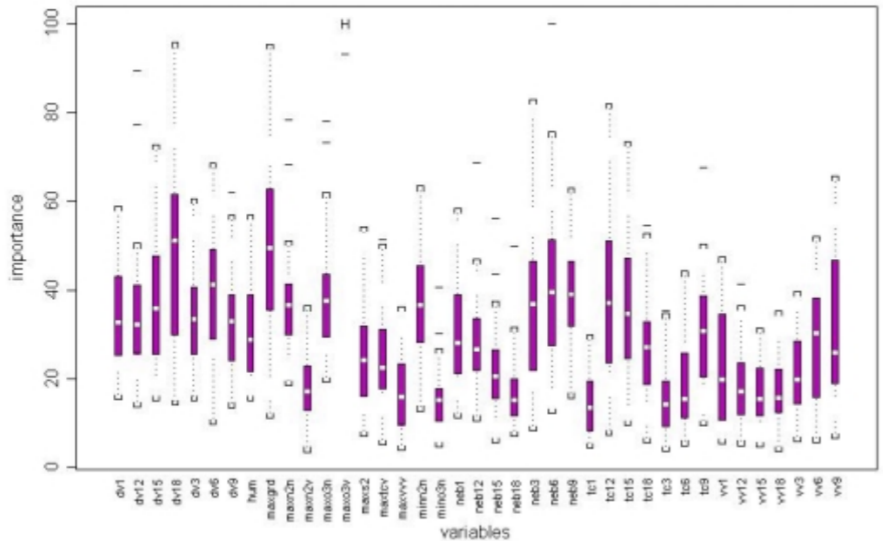

FIGURE *3.2: Boxplot d'importance de chaque variable, obtenu à partir de 40 arbres construits sur des échantillons* bootstrap *(figure issue de l'article [29])*

gradient ou la nébulosité qui ont une dispersion significative. Les variables les moins importantes semblent elles conserver leur position. Finalement, cette analyse a montré une instabilité des variables intermédiaires mais une stabilité des variables peu importantes et très importantes.

Limites de l'importance des variables d'un arbre de décision

L'importance des variables a l'avantage d'être un indicateur simple et facilement compréhensible du comportement du modèle. Cependant, il devient vite limité et peut donner des résultats extrêmement variables. Pour mettre en évidence cette limite de l'importance des variables, considérons un problème de classification binaire. On veut mesurer l'importance de deux variables x_1 et x_2 sur la variable à expliquer $y \in \{0, 1\}$. On construit la règle de calcul suivante : Si $x_1 < 0.5$ et $x_2 < 0.5$ ou $x_1 \geq 0.5$ et $x_2 \geq 0.5$: $y = 0$, sinon $y = 1$. On construit un échantillon de taille $n = 1000$ défini suivant cette règle, avec x_1 et x_2 suivant une loi uniforme sur $[0, 1]$,

Pour cet échantillon de taille 1000, nous ajustons un arbre de décision, avec la fonction *rpart* de R, en utilisant les paramètres de base. A l'aide de cet ajustement, nous calculons l'importance des variables x_1 et x_2. Nous répétons cette opération $N = 1000$ fois et nous analysons la distribution de l'importance des variables observées (pour la variable x_1 par exemple). Pour se faire, nous normalisons l'importance des variables de sorte à avoir une valeur entre 0 et 100.

Nous observons très clairement que dans la plupart des cas, l'importance estimée par le modèle n'est pas de 50%, comme on aurait pu le penser intuitivement. Dans la majorité des cas, il surestime l'importance d'une variable au détriment de l'autre. Ainsi, il faut se méfier de la notion d'importance des variables, et ne s'interprète pas aussi intuitivement

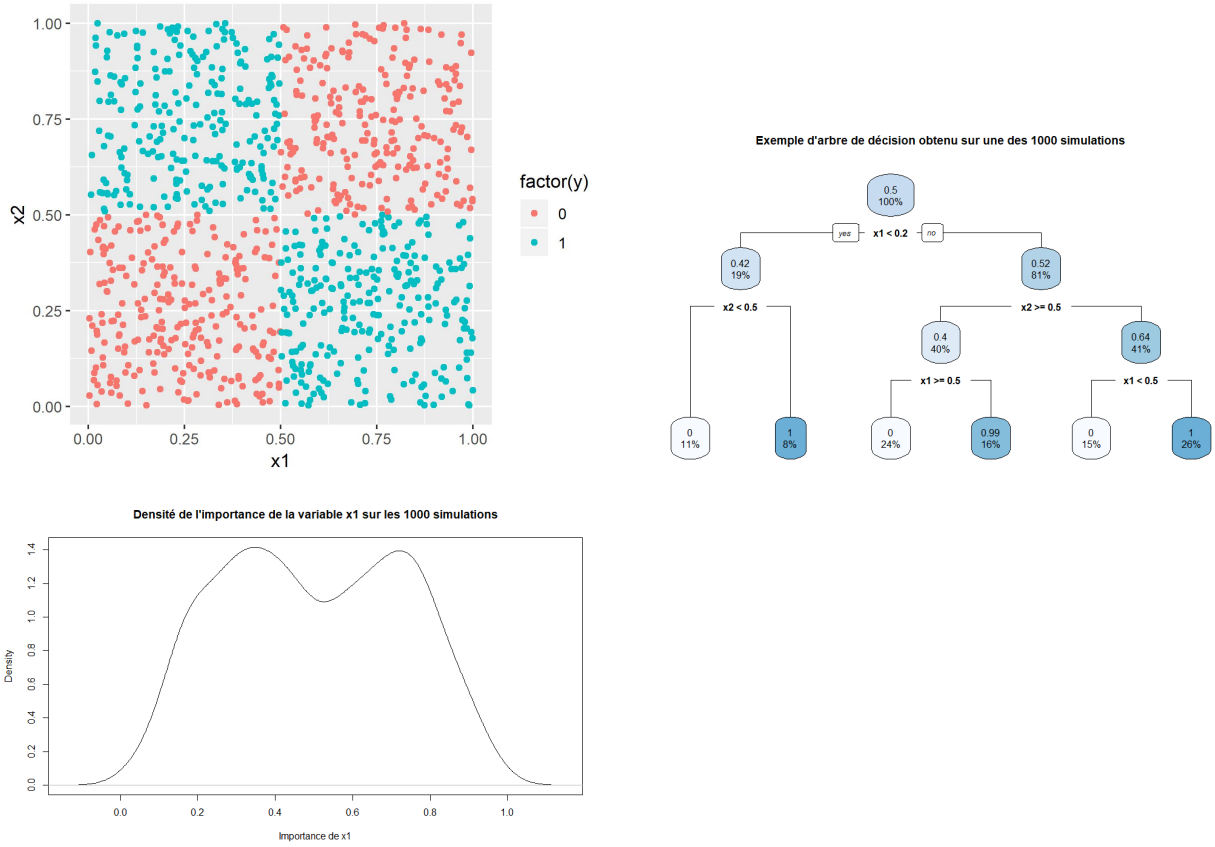

FIGURE *3.3: Mise en place d'un modèle CART sur des données simulées afin de montrer la limite de l'importance des variables*

qu'on ne le croit.

3.1.3.3 Interprétabilité des arbres de décision

Les arbres de décision font partie de la classe d'algorithmes considérés comme "interprétables". En effet, leur structure simple d'arbre leur permet d'expliquer facilement les prédictions réalisées. L'arborescence nous invite à raisonner de manière "contrefactuelle", c'est-à-dire que pour comprendre une prédiction d'une certaine instance, on se demande l'impact sur la décision si une variable avait été inférieure ou supérieure à une certaine valeur, ou si elle changeait de catégorie. Ceci peut être vu sur l'exemple simple des données Titanic [1] dont l'objectif est de prédire la survie ou non d'un individu en fonction de ses caractéristiques, dont l'âge. Supposons que l'on obtient le modèle décrit sur la figure 3.4. Alors il est facile d'interpréter les prédictions faites. En effet, si un individu est un

1. Données Titanic disponibles sur : `https://gist.github.com/michhar/2dfd2de0d4f8727f873422c5d959fff5`.

homme de 30 ans, alors sa probabilité de survie estimée par le modèle est de 17 %. Si cet individu avait été une femme alors sa probabilité de survie estimée par le modèle aurait été de 74 %, tandis que si il avait été âgé de moins de 6 ans et demi, alors elle aurait été de 67 %.

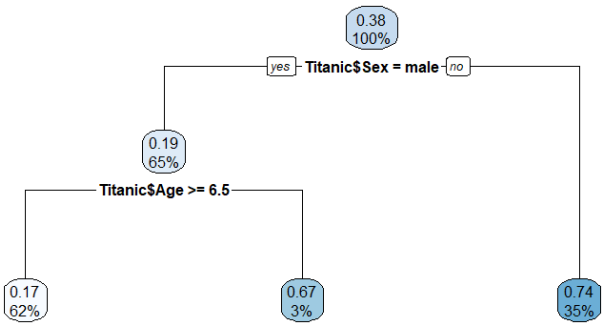

FIGURE *3.4: Modèle d'arbre de décision sur la probabilité de survie d'un passager du Titanic en fonction de son âge et de son sexe*

Cette simplicité d'interprétation en fait l'un des modèles de data science les plus prisés comme alternative aux méthodes traditionnelles (statistiques univariées, bivariées, etc.). Il peut également servir comme modèle de substitution à un modèle plus complexe comme nous le verrons dans l'annexe B.3.

3.2 Un exemple de modèle complexe : XGBoost

Dans cette partie, on s'intéresse à un modèle de Machine Learning en particulier : le XGBoost. Celui-ci utilise de manière judicieuse les arbres de décisions, détaillés ci-avant, pour former un modèle plus complexe et généralement plus performant. Il est désormais largement utilisé dans de nombreux domaines, y compris en actuariat, notamment pour ses nombreux hyperparamètres que l'on peut régler le plus finement possible pour améliorer la précision. Ce modèle sera utilisé dans notre application actuarielle du chapitre 5, notamment pour la modélisation fréquentielle en tarification automobile. Considéré comme boîte-noire, le XGBoost sera alors interprété à l'aide des outils développés au cours du chapitre 4.

D'autres algorithmes d'apprentissage statistiques jugés complexes ont été envisagés lors de notre application et sont détaillés en annexe A.

3.2.1 Boosting

Pour comprendre l'algorithme XGBoost, il est nécessaire de rappeler le Boosting et notamment Adaboost, ainsi que l'algorithme de Gradient Boosting. Le Boosting est une

technique ensembliste qui consiste à agréger des classifieurs faibles [2] qui sont construits étape par étape, en corrigeant au fur et à mesure les poids des individus [24]. L'objectif est alors d'obtenir un classifieur fort [3].

Description de l'algorithme de Boosting (Adaboost) : Soit B le nombre de modèles utilisés, $(ALGO)$ l'algorithme utilisé (par exemple un arbre de décision), y la variable binaire cible : $y \in \{-1, 1\}$, X la matrice des variables explicatives. On considère l'ensemble d'apprentissage que l'on a à notre disposition : $(x_1, y_1), ..., (x_n, y_n)$, c'est-à-dire : $x = (x_{i,j})_{1 \leq i \leq n, 1 \leq j \leq p}$ et $(y_i)_{1 \leq i \leq n}$

- $LISTE$, une liste qui contiendra les différents modèles et initialement vide.
- les individus sont pondérés uniformément : $w_i^1 = 1/n$
- pour $b = 1$ à B :
  - construire le modèle M_b reposant sur les données pondérées par $w^b = (w_i^b)_{\{1 \leq i \leq n\}}$ à l'aide de $(ALGO)$. On note $\hat{y}_i = M_b(x_i)$ la prédiction faite par le modèle M_b.
  - ajouter M_b dans $LISTE$
  - calcul du taux d'erreur commis par M_b : $\epsilon_b = \sum_{i=1}^{n} w_i \mathbb{I}_{\{y_i \neq \hat{y}_i\}}$
  - si $\epsilon_b > 0.5$ ou $\epsilon_b = 0$: arrêt de l'algorithme
  - sinon :
    - calcul du poids du classifieur : $\alpha_b = ln(\frac{1-\epsilon_b}{\epsilon_b})$
    - calcul du poids suivant : $w_i^{b+1} = w_i^b e^{\alpha_b \mathbb{I}_{\{y_i \neq \hat{y}_i\}}}$

A la fin de cet algorithme, un vote pondéré par $(\alpha_b)_{1 \leq b \leq B}$ est ensuite effectué et finalement :

$$f(x) = sign(\sum_{b=1}^{B} \alpha_b M_b(x)) \tag{3.23}$$

où x est le vecteur des variables explicatives d'entrée, et $f(x)$ la sortie estimée par notre modèle de Boosting. Il existe de nombreuses variantes sur le choix des poids α des classifieurs. Nous voyons ici que nous avons le choix sur le type de modèle utilisé (c.f. $(ALGO)$). En général, ce sont des arbres de décision qui sont utilisés car ils permettent de remplir le critère de classifieur faible.

3.2.1.1 Algorithme d'optimisation de descente de gradient

Pour faire le lien avec l'algorithme de gradient boosting, il est nécessaire de rappeler l'algorithme de descente de gradient. Ce dernier est une technique itérative qui permet d'approcher la solution d'un problème d'optimisation convexe . Considérons une fonction deux fois différentiable f, d'espace de départ $\mathbb{R}^p$ (muni de la norme $\| \cdot \|$ et du produit scalaire $< \cdot >$) et d'espace d'arrivée $\mathbb{R}$, qu'on souhaite minimiser. Pour $x \in \mathbb{E}$, on note $\nabla f(x)$ le gradient de f en x : $\nabla f(x) = (\frac{\partial f}{\partial x_i}(x))_{1 \leq i \leq p}$. On choisit une valeur initiale

2. Le terme de classifieur faible (*weak classifier*) proposé par Shapire [24] fait référence à un modèle légèrement plus performant que le modèle trivial.

3. Par opposition au classifieur faible, un classifieur fort fait écho à un modèle bien plus performant que le modèle trivial. [24]

$x_0 \in \mathbb{R}^p$ pour l'algorithme, ainsi qu'un seuil (critère d'arrêt) $\epsilon > 0$. L'algorithme va alors calculer de manière itérative des valeurs $x_1, x_2, ...$ jusqu'à la convergence vers l'optimum. L'algorithme est le suivant :

— k=0
— tant que $\| \nabla f(x_k) \| > \epsilon : x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, avec α_k un paramètre calculé à chaque itération ou bien un paramètre constant α.

Cet algorithme utilise le fait qu'une fonction décroît plus fortement autour d'un point x dans le sens opposé au gradient, ce qui justifie le signe moins dans la formule itérative précédente.

Appliquons à présent cet algorithme d'optimisation à la fonction coût du modèle considéré. Notons j notre fonction de coût utilisée pour comparer la valeur $f(x_i) = \hat{y_i}$ prédite par notre modèle et la valeur y_i réelle (par exemple $j(y_i, f(x_i)) = (y_i - f(x_i))^2$ couramment utilisée en régression). Notons J la fonction de coût global, définie comme la somme des fonctions de coût individuelles : $J(y, f) = \sum_{i=1}^{n} j(y_i, f(x_i)$ L'objectif va être alors de minimiser cette fonction de coût, par rapport aux paramètres de la fonction f associée à notre modèle, à l'aide de la méthode de descente du gradient vu précédemment. Ainsi, si on note f_b notre classifieur à l'étape b de l'algorithme, nous obtenons la formule suivante :

$$f_b(x) = f_{b-1}(x) - \alpha \nabla j(y, f(x)) \tag{3.24}$$

$$= f_{b-1}(x_i) - \alpha \frac{\partial j(y_i, f(x_i))}{\partial f(x_i)}, 1 \leq i \leq n \tag{3.25}$$

On remarque alors que l'algorithme de Boosting présenté au-dessus consiste à optimiser une fonction de coût, à l'aide de la méthode de descente de gradient avec une fonction de coût exponentielle. En reprenant notre exemple avec $y \in \{-1, 1\}$ notre variable à expliquer, et en notant f notre classifieur agrégé à partir des modèles individuels $M_1, M_2, ... : J(f) = \sum_{i=1}^{n} exp(-y_i f(x_i))$ (notre fonction de coût à minimiser).

3.2.1.2 Algorithme de Gradient Boosting

Cas de la régression

Nous pouvons à présent définir la méthode de Gradient Boosting, qui généralise l'approche précédente, avec d'autres fonctions de coût. Les notations précédentes sont conservées, à savoir f_i pour les fonctions associées au modèle de l'itération i, J la fonction de coût. L'algorithme se présente ainsi :

— initialisation par un modèle constant : $f_0(x) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} J(y_i, \theta)$ (correspondant à la moyenne des observations de y pour une fonction de coût correspondant à la norme euclidienne)
— pour m allant de 1 à M (on pourrait aussi introduire un critère d'arrêt) :
  — calul des pseudo résidus : $r_{i,m} = -\frac{\partial J(y_i, f(x_i)}{\partial f(x_i)}|_{f(x)=f_{m-1}(x)}, 1 \leq i \leq n$

— ajustement d'un classifieur $h_m(x)$ (faible, comme un arbre par exemple), entraîné sur les données $(x_i, r_{i,m})_{1\leq i\leq n}$, où $(x_i)_{1\leq i\leq n}$ sont les variables explicatives et $(r_{i,m})_{1\leq i\leq n}$ les sorties.
— calcul du prochain classifieur : $f_m(x) = f_{m-1}(x) + \theta_m h_m(x)$, où θ_m est obtenu en résolvant le problème d'optimisation : $\theta_m = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} J(y_i, f_{m-1})(x) + \theta h_m(x_i)$

A la sortie de cet algorithme, on obtient : $f_M(x)$ qui est notre classifieur final. Les fonctions de coûts généralement choisies pour la régression sont :

— *écart quadratique* : $j(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$. Cette fonction présente l'avantage d'être sensible aux écarts mais l'inconvénient d'être peu robuste vis à vis des points aberrants
— *valeur absolue* : $j(y_i, f(x_i)) = |y_i - f(x_i)|$. L'avantage de ce choix réside dans sa robustesse aux points abérrants mais est moins sensible aux écarts
— *fonction Huber* : $j(y_i, f(x_i)) = \begin{cases} y_i - f(x_i) \ si \ |y_i - f(x_i)| < \delta_q \\ \delta_q sign(y_i - f(x_i)) \ sinon \end{cases}$ où δ_q est le quantile d'ordre q de $(|y_i - f(x_i)|)_{1\leq i\leq n}$. Ceci permet d'obtenir les avantages de la valeur absolue (sensibilité aux valeurs faibles du gradient) et de l'écart quadratique (sensibilité aux valeurs élevées du gradient).

Cas de variable cible catégorielle

Dans le cas où la variable cible est catégorielle avec K modalités (de 1 à K), l'algorithme reste très proche de celui présenté précédemment, avec néanmoins une fonction de coût différente, adaptée au classement. Pour cela on définit :

$$y_i^k = \begin{cases} 1 \ si \ y_i = k \\ 0 \ sinon \end{cases}, 1 \leq i \leq n, 1 \leq k \leq K$$

La nouvelle fonction de coût est alors définie par :

$$j(y_i, f(x_i)) = -\sum_{k=1}^{K} y_i^k log(p^k(x_i)) \tag{3.26}$$

où $p_k(x_i)$ correspond à la probabilité conditionnelle d'appartenir à la classe k, i.e :

$$p_k(x_i) = \frac{exp(f^k(x_i))}{\sum_{k=1}^{K} exp(f^k(x_i))} \tag{3.27}$$

On travaille donc à présent sur les indicatrices (y^k), et on construit des modèles (généralement des arbres) sur les gradients négatifs pour les K classes. f^k est ainsi le modèle additif issu des K modèles précédents.

3.2.1.3 Tree Gradient Boosting

Le Tree Gradient Boosting est le cas particulier du Gradient Boosting où les modèles h_m choisis sont des arbres de décision. Les paramètres à optimiser peuvent alors être

les poids w_j à chaque feuille, ainsi que le nombre de feuilles T dans chaque arbre. Le XGBoost présenté ensuite est une implémentation plus rapide de cet algorithme.

3.2.2 XGBoost

XGBoost est une manière optimisée et plus rapide de réaliser l'algorithme de Tree Gradient Boosting. Ce dernier est un algorithme d'apprentissage supervisé qui tente de prédire le plus précisément possible une variable cible, en combinant de manière séquentielle les prédictions de plusieurs algorithmes dits faibles. Le plus souvent, on utilise des arbres de décisions avec peu de noeuds comme apprenants faibles (*weak learners*). Le fait de travailler de manière séquentielle le différencie des forêts aléatoires et le rend plus lent mais lui permet de s'améliorer au fur et à mesure. L'algorithme tente d'optimiser à chaque itération une fonction objective régularisée (en norme $\mathbb{L}^1$ et $\mathbb{L}^2$), constituée d'une fonction de coût et d'un terme de pénalité pour tenir compte de la complexité du modèle. A chaque itération, le nouveau modèle prédit la partie résiduelle que l'on n'a pas su expliquer à l'itération précédente. L'output est alors la somme des prédictions faites par chaque modèle. Le terme de Gradient Boosting vient du fait que la méthode utilisée pour l'optimisation à chaque étape est une descente de gradient.

L'intérêt du XGBoost réside dans la correction d'un problème majeur du Tree Gradient Boosting, à savoir le fait de considérer la perte potentielle pour chaque séparation possible (*split*) à un noeud donné pour créer une nouvelle branche. En effet, dans le cas où nous disposons de beaucoup de variables explicatives, il y a un grand nombre de splits possibles, ce qui engendre des temps de calcul élevés.

Notons que nous allons présenter l'algorithme uniquement dans le cadre de la régression, mais celui-ci s'adapte pour la classification.

3.2.2.1 Algorithme

Considérons un problème de régression pour lequel nous voulons prédire la variable $Y \in \mathbb{R}$, à partir d'un vecteur X de p variables explicatives : $X \in \mathbb{R}$ ($p \in \mathbb{N}$). Nous reprenons les idées du Boosting, à savoir agréger les prédictions de plusieurs classifieurs dits faibles pour former un classifieur fort.

Entraînement séquentiel

Soit B le nombre d'étapes de notre algorithme. Notons pour $b \in \{1, ..., B\}$:

- H l'ensemble des modèles sur lequel on va réaliser l'optimisation
- j : $\mathbb{R}$x$\mathbb{R} \to \mathbb{R}$ une fonction de coût choisie au préalable (comme le MSE par exemple).
- $\hat{h}_b$: $\mathbb{R}^p$ x $\mathbb{R}^n \to \mathbb{R}^n$ la fonction associée au modèle M_b de l'étape b de l'algorithme, renvoyant le vecteur de prédiction de l'input. Notons bien que le modèle M_b apprend sur les valeurs résiduelles à l'étape b et non la valeur de y.
- $\hat{y}^{(b)}$ notre estimation de y à l'étape b.

— $r^{(b)}$ le vecteur de taille n des pseudo-résidus au début de l'étape b, c'est-à-dire la partie résiduelle de la valeur cible initiale que nous n'avons pas encore réussi à expliquer avant l'itération b : $r^{(b)} = y - \hat{y}^{(b-1)}$ et $r^{(0)} = \hat{y}^{(0)}$
— $\hat{r}^{(b)}$ l'estimation du vecteur $r^{(b)}$ faite par l'algorithme M_b à l'étape b, soit : $\hat{r}^{(b)} = \hat{h}_b(x)$.

On peut résumer ceci par les relations suivantes :

$$\hat{y}_i^{(b)} = \sum_{k=0}^{b} \hat{r}_i^{(k)} = \hat{y}_i^{(b-1)} + \hat{r}_i^{(b)} \; et \; \hat{y}_i^{(0)} = \hat{h}(x_i), \; 1 \leq b \leq B, 1 \leq i \leq n \tag{3.28}$$

On définit également Ω une fonction qui prend en entrée un modèle (un arbre de décision par exemple) et renvoie une mesure de la complexité du modèle. L'algorithme se présente comme ceci :

— Initialisation : on définit un modèle constant $\hat{h_0}$. Par exemple : $\hat{h_0}(x_i) = C, 1 \leq i \leq n$, avec $C \in \mathbb{R}$. Par exemple $C = 0$. On obtient $\hat{y}^{(0)} = \hat{r}^{(0)} = C$
— Pour b allant de 1 à B :
  — Calibrage du modèle M_b et de la fonction $\hat{h_b}$ associée, ajustés sur les données $(x, y - \hat{y}^{(b-1)})$ avec $y - \hat{y}^{(b-1)} = \hat{r}^{(b)}$. c'est-à-dire que $\hat{h}_b(x_i)$ donne une estimation de $y_i - \hat{y_i}^{(b-1)}, 1 \leq i \leq n$ en optimisant la fonction de coût :

$$\hat{h_b} = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} j(y_i, \hat{y_i}^{(b)}) + \sum_{l=1}^{b} \Omega(h_l) \tag{3.29}$$

$$= \underset{h_b \in H}{argmin} \sum_{i=1}^{n} j(y_i, \hat{y_i}^{(b-1)} + h_b(x_i)) + \Omega(h_b) + C_b \tag{3.30}$$

avec C_b une constante qui dépend seulement des itérations avant b. Dans le cas d'une fonction de coût MSE, on obtient :

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} (y_i - (\hat{y_i}^{(b-1)} + h_b(x_i))^2 + \Omega(h_b) + C_b \tag{3.31}$$

Soit :

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} [2(\hat{y_i}^{(b-1)} - y_i) h_b(x_i) + h_b(x_i)^2] + \Omega(h_b) + C_b \tag{3.32}$$

La forme obtenue, avec un terme d'ordre 1 et d'ordre 2, peut également être retrouvée dans le cas de fonctions de coût différentes de la MSE, en utilisant son développement de Taylor à l'ordre 2 :

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(b-1)}) + \alpha_i^{(b)} h_b(x_i) + \frac{1}{2} \beta_i^{(b)} h_b(x_i)^2 + \Omega(h_b) + C_b \tag{3.33}$$

avec :

— $\alpha_i^{(b)} = \frac{\partial j}{\partial y}(y_i, \hat{y}^{(b-1)})$
— $\beta_i^{(b)} = \frac{\partial^2 j}{\partial y^2}(y_i, \hat{y}^{(b-1)})$
Il s'agit d'une approximation licite car les termes $h_b(x_i)$ sont supposés de taille négligeable étant donné qu'il s'agit de classifieurs faibles.
Ainsi, en supprimant toutes les constantes (qui dépendent des étapes précédant l'étape b), on doit résoudre le problème d'optimisation suivant :

$$\hat{h}_b = \underset{h_b \in H}{argmin} \sum_{i=1}^{n} \alpha_i^{(b)} h_b(x_i) + \frac{1}{2}\beta_i^{(b)} h_b(x_i)^2 + \Omega(h_b) \tag{3.34}$$

C'est comme ceci que XGBoost fonctionne : il peut optimiser chaque fonction de perte choisie, en utilisant le même solveur en prenant les coefficients $\alpha_i^{(b)}$ et $\beta_i^{(b)}$ comme input.
— Finalement, à la fin de notre algorithme, notre estimation de y est donnée par $\hat{y}^{(B)}$: il s'agit de notre output.

Complexité du modèle et régularisation

Nous avons présenté dans l'algorithme une fonction de complexité Ω. Cette fonction contient un terme de régularisation afin d'éviter le surapprentissage. Plaçons nous dans le cas où les classifieurs faibles utilisés sont des arbres. Notons T le nombre de feuilles au dernier niveau de l'arbre, w_t le score prédit par la feuille t ($1 \leq t \leq T$). La fonction h_b associée à l'arbre de l'itération b ($1 \leq b \leq B$) est définie par : $h_b(x_i) = w_{s(x_i)}$, avec $s : \mathbb{R}^p \to \{1, ..., T\}$ une fonction renvoyant le numéro de la feuille associée au point considéré. Avec une écriture vectorielle, on a : $h_b(x) = w_{s(x)}$. La fonction de complexité du XGBoost est alors généralement définie par :

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} w_k^2 \tag{3.35}$$

où γ et λ sont des paramètres de régularisation à choisir.

Algorithme de construction des arbres

En reprenant, notre formule d'optimisation : $\sum_{i=1}^{n} \alpha_i^{(b)} h_b(x_i) + \frac{1}{2}\beta_i^{(b)} h_b(x_i)^2 + \Omega(h_b)$ et en injectant notre fonction de complexité définie juste avant, on obtient :

$$\sum_{i=1}^{n} \alpha_i^{(b)} h_b(x_i) + \frac{1}{2}\beta_i^{(b)} h_b(x_i)^2 + \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} w_k^2 \tag{3.36}$$

Soit, en intervertissant les sommes :

$$\sum_{j=1}^{T} \left[(\sum_{i \in I_j} \alpha_i^{(b)}) w_j + \tfrac{1}{2}(\sum_{i \in I_j} \beta_i^{(b)} + \lambda) w_j^2 \right] + \gamma T \tag{3.37}$$

avec $I_j = \{i \in \{1, ..., T\} | s(x_i) = j\}$ l'ensemble contenant tous les indices des points qui ont été conduits à la feuille j. En notant $A_j = \sum_{i \in I_j} \alpha_i^{(b)}$ et $B_j = \sum_{i \in I_j} \beta_i^{(b)}$, on obtient :

$$\sum_{j=1}^{T} [A_j w_j + \frac{1}{2}(B_j + \lambda) w_j^2] + \gamma T \tag{3.38}$$

Dans cette équation les scores de prédiction w_j n'interagissent pas entre-eux et sont donc indépendants. Pour une fonction $s(x)$ donnée, on peut optimiser chaque terme de la somme indépendamment, à savoir : $A_j w_j + \frac{1}{2}(B_j + \lambda) w_j^2$.
Finalement on obtient, en excluant la solution $w_j = 0$:

$$\forall j \in \{1, ..., T\}, w_j = -\frac{A_j}{B_j + \lambda} \tag{3.39}$$

et une fonction objectif associée valant alors : $-\frac{1}{2} \sum_{j=1}^{T} \frac{A_j^2}{B_j + \lambda} + \gamma T$, qui mesure à quel point l'arbre associé à la fonction s, est bon (en terme de prédiction et de complexité). Il faut à présent trouver le meilleur arbre, avec la fonction de tri s la plus performante associée. Il faudrait en théorie énumérer tous les arbres possibles et choisir le meilleur mais cela demanderait trop de temps. Une méthode utilisée en pratique est une optimisation étage par étage de l'arbre. L'idée est de regarder si l'ajout de nouvelles feuilles à un niveau donné fournit un gain comparé à l'arbre initial, tout en tenant compte de la régularisation. Dans le cas de l'ajout de deux feuilles à un niveau donné, il faut que le gain associé (régularisé) soit positif :

$$Gain = \frac{1}{2} \left[\frac{A_G^2}{B_G + \lambda} + \frac{A_L^2}{B_L + \lambda} - \frac{(A_G + A_L)^2}{B_G + B_L + \lambda} \right] - \gamma \tag{3.40}$$

L'indice G réfère à la feuille gauche ajoutée, et D à la feuille droite. Finalement il faut que le gain d'ajout de ces deux feuilles, comparé à la situation précédente, soit supérieur au coefficient γ de régularisation. Ainsi, pour construire un arbre, on cherche récursivement la meilleure division possible jusqu'à ce qu'on atteigne la profondeur maximale. Ensuite, on élague les noeuds avec un gain négatif avec une approche du bas vers le haut (bottom-up).

3.2.2.2 Avantages du XGBoost

Les avantages de l'algorithme XGBoost sont nombreux, et peuvent expliquer pourquoi il est tant utilisé dans les compétitions de Machine Learning.

— Il existe de nombreux hyperparamètres que l'on peut régler au plus fin pour optimiser ses performances. Ils permettent de garder un contrôle sur le compromis biais/variance. On peut citer notamment :
  - Paramètres de contrôle de la complexité du modèle :
    · Le nombre de sous-arbres h_m entraînés
    · La profondeur maximale de chaque arbre h_m

        - Gain minimum γ requis pour autoriser une division supplémentaire
    - Paramètres de robustesse au bruit :
        - Ratio de sous échantillonnage de la base d'apprentissage, compris entre 0 et 1
        - Ratio de sous-échantillonnage des colonnes lors de la construction de chaque arbre
    - Le taux d'apprentissage (*learning rate*). Il s'agit du coefficient utilisé dans la méthode de descente du gradient pour contrôler la vitesse de convergence de l'algorithme. Si il est trop bas, le temps de convergence peut être long, et si il est trop élevé, l'algorithme risque de ne pas converger.
    - Les paramètres γ et λ de régularisation $\mathbb{L}^1$ et $\mathbb{L}^2$.
    - Ajustement possible dans le cas d'une classification avec des données déséquilibrées entre les classes (en utilisant par exemple la métrique "AUC").
- Rapidité de calcul, en comparaison du Tree Gradient Boosting classique.

Chapitre 4

Interprétabilité post-hoc, indépendante du modèle

Dans ce chapitre, nous nous intéresserons à des méthodes d'interprétation des modèles de type boîte-noire. Ces outils nous permettrons d'interpréter le modèle XGBoost, vu dans le chapitre précédent et implémenté au cours de l'application actuarielle de tarification automobile du chapitre 4. Comme nous l'avons rappelé dans le chapitre 2, il existe de nombreuses méthodes d'interprétation. La figure 4.1 indique les différentes catégories dans lesquelles nous pouvons les ranger. On distingue tout d'abord différents cadres d'application de ces méthodes. Les deux grandes familles présentées dans la littérature sont les méthodes globales et locales. Ces dernières reposent sur la compréhension de la prédiction de la boîte-noire d'une observation en particulier alors que l'approche globale essaie de comprendre le modèle dans son intégralité. A mi-chemin entre ces deux familles, le cadre régional, qui essaie d'expliquer le comportement du modèle pour un groupe d'individus similaires, par exemple à partir de clusters. On peut également séparer les méthodes d'interprétation en deux types, à savoir spécifique ou agnostique au modèle. L'approche spécifique repose essentiellement sur la structure du modèle étudié. Par exemple, la méthode d'importance des variables introduites par Brieman pour les modèles construits à partir d'arbres est spécifique à cette classe de modèles, c'est-à-dire qu'elle ne pourra pas s'adapter pour d'autres méthodes, comme un SVM (détaillé en annexe A) par exemple. A l'inverse une méthode agnostique au modèle, est indépendante de celui-ci et peut donc s'appliquer à n'importe quelle famille d'algorithmes. Enfin, on distingue les méthodes intrinsèques, qui reposent essentiellement sur la structure du modèle (par exemple un arbre de décision ou des règles de décision) et les méthodes Post-Hoc qui ont lieu une fois que le modèle ait été ajusté. Ces dernières sont pour la majorité agnostiques au modèle. Dans ce chapitre, on s'intéresse uniquement aux méthodes Post-Hoc, et agnostiques au modèle, car il s'agit de la classe la plus populaire et la plus utilisée à l'heure actuelle, notamment grâce à la possibilité d'interpréter n'importe quel algorithme avec les mêmes outils, et donc de pouvoir comparer leurs comportements.

Nous commençons notre étude avec des méthodes visuelles d'interprétabilité.

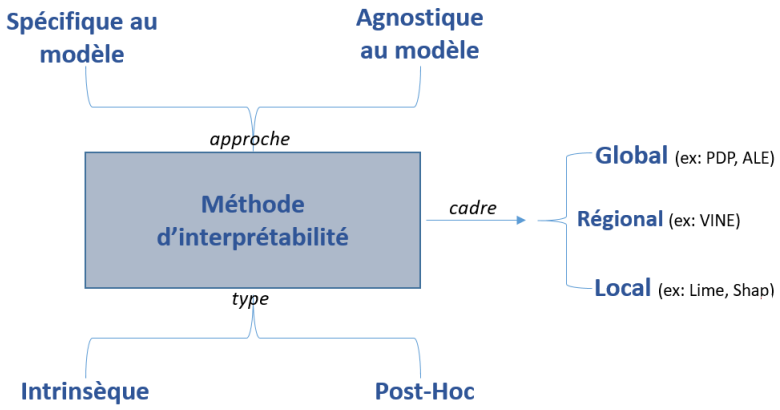

FIGURE *4.1: Différentes catégories d'interprétabilité des modèles*

4.1 Méthodes visuelles d'interprétabilité

4.1.1 Graphique de dépendance partielle (PDP) et Prédiction individuelle conditionnelle (ICE)

4.1.1.1 PDP

Principe général

Présentons une première méthode, appelée PDP (Partial Dependance Plot), introduite par Friedman en 2001 dans l'article [26] *A Greedy Function Approximation : A Gradient Boosting Machine*. Il s'agit sans doute de la méthode la plus ancienne d'interprétation des modèles de boîtes noires, au regard des publications de ces 3 dernières années (c.f figure 2.1 montrant l'augmentation exponentielle du nombre d'articles sur le sujet du machine learning interprétable). Il s'agit également d'une des méthodes les plus citées et les plus utilisées lorsque l'on souhaite interpréter un modèle de Machine Learning.

Ce graphique de dépendance partielle a pour objectif de montrer l'effet marginal d'une ou plusieurs variables explicatives (généralement pas plus de deux) sur la prédiction faite par un modèle. Considérons un modèle $\hat{f}$ entraîné sur une base d'apprentissage $Z^{(i)} = (X^{(i)}, Y^{(i)}) \in \mathbb{R}^p \text{x} \mathbb{R}$, associées aux observations $(x^{(i)}, y^{(i)})$ avec $x^{(i)} = (x_j^{(i)})_{1 \leq j \leq p}$ pour $1 \leq i \leq n$ (où $p, n \in \mathbb{N}$).

Notons X_S l'ensemble des variables pour lesquelles on veut connaître l'effet sur la prédiction et X_C les variables explicatives restantes. Par exemple $X_S = (X_1, X_2)$ et $X_C = (X_3, ..., X_p)$. Ainsi avec $X = (X_S, X_C)$, nous avons l'ensemble des variables explicatives utilisées par notre modèle pour apprendre. On définit la fonction de dépendance partielle par la formule :

$$\hat{f}_{x_S}(x_S) = \mathbb{E}_{X_c}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_s, x_c) d\mathbb{P}_{X_c}(x_c) \tag{4.1}$$

Notons bien que cette formule diffère de l'espérance conditionnelle de X_S. Afin d'esti-

mer cette dernière, nous nous basons sur les n observations à notre disposition et de la méthode de Monte Carlo :

$$\hat{f}_{x_S}(x_S) \simeq \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}) \tag{4.2}$$

Nous pouvons d'ores et déjà remarquer que le graphique de dépendance partielle est une méthode globale d'interprétation (c.f figure 4.1).

Algorithme d'estimation de la PDP

Une procédure pour estimer la PDP est donnée dans l'algorithme suivant :

- Input : la base d'apprentissage $(x_j^{(i)})_{1\leq i\leq n, 1\leq j\leq p}$, le modèle $\hat{f}$, une variable à expliquer supposée être x_1 ici pour simplifier, i.e. $S = 1$.
  c'est-à-dire que : $(x_j^{(i)})_{1\leq i\leq n, 1\leq j\leq p} = (x_1^{(i)}, x_C^{(i)})_{1\leq i\leq n}$
- Pour i allant de 1 à n :
  - Copier la base d'apprentissage, en remplaçant la valeur de la variable x_1 par la valeur constante $x_1^{(i)}$: $(x_1^{(i)}, x_C^{(k)})_{1\leq k\leq n}$
  - Calculer le vecteur de prédiction par $\hat{f}$ des données précédemment définies : $\hat{f}(x_1^{(i)}, x_C^{(k)})$ pour $k = 1, ..., n$
  - Calcul de $\hat{f}_{x_1}(x_1^{(i)})$ par la formule : $\hat{f}_{x_1}(x_1^{(i)}) \simeq \frac{1}{n}\sum_{k=1}^{n} \hat{f}(x_1^{(i)}, x_C^{(k)})$
- Output : le graphique des points $(x_1^{(i)}, \hat{f}_{x_1}(x_1^{(i)}))$ pour $i = 1, ..., n$, appelé graphique de dépendance partielle (PDP).

Cette méthode repose sur l'hypothèse de non corrélation entre les variables de (C) et les variables de (S), ce qui ne sera pas nécessairement le cas en pratique. Cela pose un problème, car dans le calcul de la moyenne précédente, on aboutira à des points non désirés ou à des combinaisons de variables explicatives qui ne sont pas susceptibles de se produire. Par exemple, si le poids et la taille sont des variables explicatives et que l'on souhaite comprendre l'effet marginal du poids, on va fixer la taille d'un individu et faire varier le poids suivant sa distribution dans la base d'apprentissage. On pourra alors avoir dans le graphique de la PDP un individu de taille 2m et de poids 30kg, ce qui n'est pas réaliste en pratique.

Pour le cas de la classification (binaire), on utilise les probabilités de sortie fournies par l'algorithme. Ainsi le PDP fournira des probabilités d'être dans une classe selon la valeur des variables de S. Dans le cas de variables catégorielles à prédire à plusieurs classes, on utilise un graphique par classe. Si nous disposons de variables explicatives catégorielles, nous obtiendrons un graphique pour chaque modalité de la variable considérée.

Exemple

Considérons un modèle simple, où les données suivent cette relation :

$$Y = 0.2X_1 - 5X_2 + 10X_2\mathbb{1}_{\{X_3\geq 0\}} + \epsilon$$

Où $X_1, X_2, X_3 \overset{iid}{\sim} U(-1,1), \epsilon \sim N(0,1)$ avec ϵ indépendant de X_1, X_2, X_3. On suppose que l'on a un échantillon de taille 1000 à disposition. Le nuage de points (*scatter-plot*) associé à X_2 et Y de cet échantillon est représenté à gauche de la figure 4.2. En supposant que l'on ne connaît pas le véritable lien entre les variables mais que l'on dispose de l'échantillon précédent, nous mettons en place un modèle de forêt aléatoire sur ces 1000 observations. Le graphique de PDP de la variable X_2 associé au modèle de Random Forest mis en place est représenté à droite de la figure 4.2.

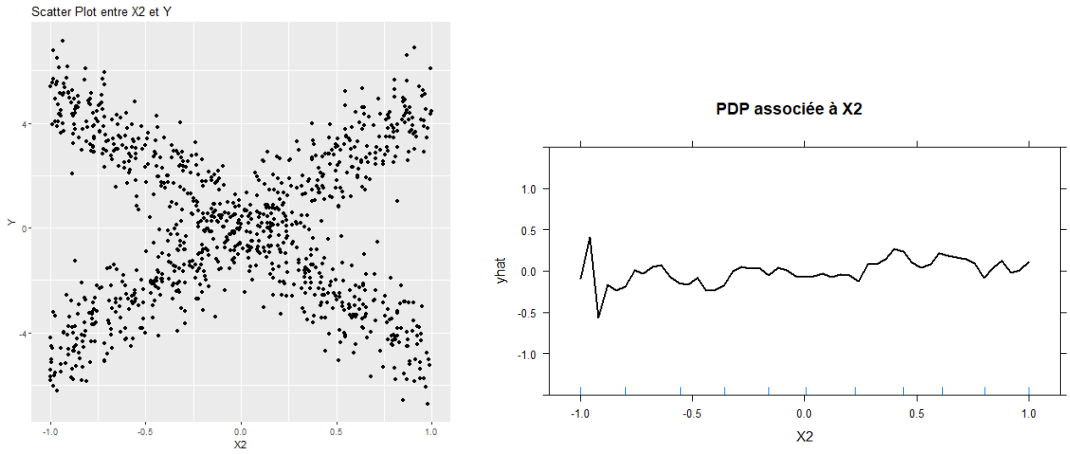

FIGURE *4.2: Scatter Plot de X_2 et Y (à gauche) et graphique PDP de X_2 (à droite)*

Le graphique de PDP laisse suggérer qu'en moyenne la variable X_2 n'est pas significative dans la prédiction de Y par le modèle de forêt aléatoire. Le nuage de points donne lui une conclusion inverse. Le fait que le PDP réalise une moyenne de l'influence de X_2 sur la prédiction fausse l'analyse qu'on souhaiterait en faire et il faut donc se méfier des conclusions que l'on peut donner. Il faut garder en tête que le PDP peut être un bon résumé de l'influence de certaines variables sur la prédiction du modèle lorsque celles-ci ne sont pas trop corrélées aux variables restantes. Une alternative à cette méthode PDP est le graphique ICE, détaillé dans la partie suivante.

Utilisation de la PDP pour une mesure d'importance de variable (IPD)

En plus de donner l'effet marginal moyen d'une variable, les graphiques de PDP semblent pouvoir fournir une information sur l'importance d'une variable dans la prédiction faite par un modèle. En effet, lorsque le graphe de PDP associé à la variable X_1 (par exemple) est relativement plat, il semble naturel de penser que cette variable n'a pas beaucoup d'influence sur la prédiction de Y. L'idée introduite par l'article [34] est de définir une fonction $Flat$ qui mesure la "platitude" de la courbe de PDP : pour une observation x, $i(x) = Flat(\hat{f}_{x_s}(x_s))$. Une mesure simple et efficace proposée dans l'article est : la variance empirique lorsque les variables x_S sont continues et la statistique d'intervalle divisée par 4 pour les variables catégorielles à $K \in \mathbb{N}$ niveaux. Dans le cas

où S=1, on obtient ainsi les formules :

$$i(x_1) = \begin{cases} \frac{1}{n-1}\sum_{i=1}^{n}[\hat{f}_{x_1}(x_1^{(i)}) - \frac{1}{n}\sum_{i=1}^{n}\hat{f}_{x_1}(x_1^{(i)})]^2 \ si\ x_1\ est\ continue \\ [\underset{1\leq i\leq n}{max}\hat{f}_{x_1}(x_1^{(i)}) - \underset{1\leq i\leq n}{min}\hat{f}_{x_1}(x_1^{(i)})]/4 \ si\ x_1\ est\ qualitative \end{cases}$$

Nous notons cette technique IPD (Importance Based On Partial Dependance) par la suite. Illustrons cette méthode sur un exemple simple de modèle linéaire. Nous avons vu que l'importance des variables en régression linéaire est calculée via la t-statistique, c'est-à-dire le rapport entre la valeur absolue du coefficient estimé, associée à une variable, divisée par l'écart-type du coefficient. Dans le cas où les variables explicatives sont indépendantes et distribuées uniformément sur le même intervalle, cette mesure d'importance via la t-statistique est équivalente à la fonction i définie ci-dessus à l'aide de la fonction de dépendance partielle. Considérons le modèle :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon,\ avec\ \beta_0, \beta_1, \beta_2 \in \mathbb{R},\ X_1, X_2 \overset{iid}{\sim} U(0,1), \epsilon \sim N(0, \sigma^2), \sigma > 0$$

On peut facilement calculer les fonctions de dépendance partielle associées à X_1 et X_2 :

$$f_{x_1}(X_1) = \int_0^1 \mathbb{E}[Y|X_1, X_2]dX_2 = \beta_0 + \beta_2/2 + \beta_1 X_1$$

$$f_{x_2}(X_2) = \int_0^1 \mathbb{E}[Y|X_1, X_2]dX_1 = \beta_0 + \beta_1/2 + \beta_2 X_2$$

On peut également calculer la variance de chaque fonction de dépendance partielle, à savoir :

$$\mathbb{V}[f_{x_1}(X_1)] = \frac{\beta_1^2}{12}\ et\ \mathbb{V}[f_{x_2}(X_2)] = \frac{\beta_2^2}{12}$$

Considérons un échantillon de taille $n = 1000$ observations du modèle :

$$Y = 1 + 3X_1 - 5X_2 + \epsilon,\ avec\ \sigma = 0.01$$

Alors :

$$f_{x_1}(X_1) = -\frac{3}{2} + 3X_1\ et\ f_{x_2}(X_2) = \frac{5}{2} - 5X_2$$

Ainsi, la PDP montre que X_2 a plus d'influence que X_1 du fait que le rapport (en valeur absolue) des pentes est de $5/3 \simeq 1.67$: X_2 est 1.67 fois plus influent que X_1 sur la prédiction de Y faite par le modèle. En utilisant la formule ci-dessus de la fonction d'importance i, on obtient numériquement : $i(X_1) \simeq 1.5$ et $i(X_2) \simeq 0.9$, ce qui donne un ratio $\frac{i(X_2)}{i(X_21}$ $\simeq 1.65$ proche de la valeur théorique :

$$\sqrt{\frac{\mathbb{V}[f_{x_2}(X_2)]}{\mathbb{V}[f_{x_1}(X_1)])}} = 5/3 \simeq 1.67$$

L'utilisation de la t-statistique devient moins pertinente dans les modèles linéaires lorsqu'une variable apparaît plusieurs fois dans l'équation, par exemple si il y a une interaction entre les variables ou des termes polynomiaux. L'approche de calcul d'importance par la fonction de dépendance ne présente pas cet inconvénient.

Utilisation de la PDP pour détecter des effets d'interaction

La mesure d'importance i définie précédemment peut également être utilisée pour mesurer les effets potentiels d'interaction entre les variables. Notons $i(x_i, x_j)$ $(i \neq j)$ l'écart type standard de la fonction de dépendance $f_{x_i,x_j}(x_i^{(i')}, x_j^{(j')})$, $1 \leq i', j' \leq n$. Une faible interaction de x_i et x_j suggère une faible variation de $i(x_i, x_j)$ quand une des variables varie et l'autre reste constante. Pour mesurer l'interaction, on construit la fonction de dépendance f_{x_i,x_j} et on calcule pour chaque valeur de x_j, $i(x_i)$, que l'on note $i(x_i|x_j)$ et on prend l'écart-type du score d'importance résultant. On fait de même pour la variable x_j et on réalise une moyenne des deux résultats. Une grande valeur peut alors indiquer une possible interaction entre les variables.

Avantages et inconvénients du graphique PDP

Le graphique PDP est souvent utilisé notamment pour sa simplicité d'interprétation et sa facilité d'implémentation. De plus, ce graphique est un outil pour estimer l'importance des variables et les interactions entre celles-ci au sein d'un modèle.

Cependant, ce graphique seul ne suffira pas à expliquer toute la complexité du modèle étudié. Le premier problème déjà cité vient du fait que le calcul réalisé repose sur une hypothèse d'indépendance entre les variables, ce qui ne sera pas nécessairement le cas en pratique. L'autre problème majeur du PDP est le fait de masquer les effets hétérogènes, comme nous pouvons le voir sur la figure 4.2. En général, on utilise ce graphique en association avec d'autres courbes, appelées courbes ICE, qui prennent en compte les effets hétérogènes.

4.1.1.2 ICE

L'approche par les courbes ICE fournit un graphique avec une ligne pour chaque instance, qui montre comment la prédiction change quand une caractéristique change. A la place de la moyenne réalisée lors du calcul de la PDP, le calcul de l'ICE est réalisé pour chaque instance. Ainsi, on obtient un graphique ICE, contenant autant de courbes que d'observations. Cette méthode a été introduite par Goldstein et al. en 2017 dans l'article [31] *Peeking Inside the Black Box : Visualizing Statistical Learning with Plots of Individual Conditional Expectation.* Contrairement au graphique de dépendance partielle qui est une approche globale, les courbes ICE sont locales (c.f 4.1). L'algorithme utilisé pour estimer l'ICE est le suivant :

- Entrée : les données d'entraînement : $(x_j^{(i)})_{1\leq i\leq n, 1\leq j\leq p}$, le modèle ajusté $\hat{f}$, S un sous-ensemble de $\{1, ...p\}$ et C le complémentaire de S dans $\{1, ..., p\}$.
- Pour i=1,...,n :
  - $\hat{f}_S^{(i)} = 0_{n\times 1}$

- $x_C = x_C^{(i)}$: on fixe les colonnes C à la i-ième observation.
- Pour l=1,...n :
  - $x_S = x_S^{(l)}$
  - $\hat{f}_{S,l}^{(i)} = \hat{f}(x_S, x_C)$
- Output : $\hat{f}_S^{(1)} = (\hat{f}_{S,1}^{(1)}, ..., \hat{f}_{S,n}^{(1)}), ..., \hat{f}_S^{(n)} = (\hat{f}_{S,1}^{(n)}, ..., \hat{f}_{S,n}^{(n)})$

Reprenons l'exemple de la partie sur le graphique de PDP. Dans celui-ci, on a observé que la PDP, en réalisant une moyenne, ne capte pas toute la dépendance d'une variable sur la prédiction. Affichons à présent le graphique d'ICE associé à la PDP. Cette fois, l'ICE

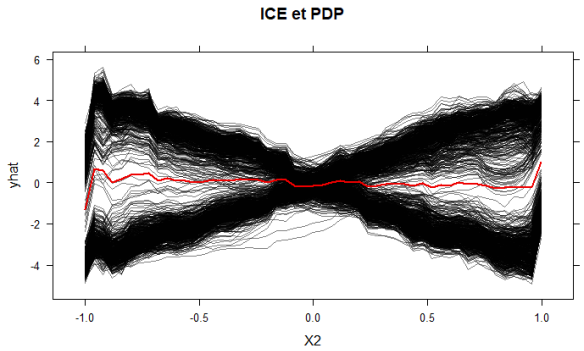

FIGURE *4.3: Graphique de PDP (en rouge) et ICE (en noir)*

capte l'effet de X_2 sur la prédiction pour chaque instance. On observe qu'en moyenne le graphique d'ICE est proche de 0, ce qui correspond à la PDP, mais que le graphique d'ICE complet est proche du nuages de points représenté sur la figure 4.2. Il existe une adaptation de la méthode ICE, appelée c-ICE, permettant de centrer les courbes et ainsi de mieux voir les effets de chaque variable sur la prédiction. Celle-ci est proposée dans l'annexe B.1

d-ICE

Un autre graphique utilisant l'ICE s'appelle le d-ICE, qui estime la dérivée partielle de chaque courbe du graphique ICE par rapport à x_S. Ceci permet de mesurer l'interaction entre les variables. Considérons le cas où x_S et x_C n'interagissent pas, c'est-à-dire que l'on peut écrire pour toute instance x : $\hat{f}(x) = \hat{f}(x_S, x_C) = g(x_s) + h(x_c)$ avec g et h deux fonctions. Il en découle que $\frac{\partial \hat{f}}{\partial x_S}(x) = g'(x_S)$. La relation entre $\hat{f}$ et x_S ne dépend pas de x_C. Ainsi, les n courbes du graphique d'ICE obtenu pour x_S ont donc la même forme mais sont juste décalées suivant la valeur de x_C. Comme il est difficile d'estimer les dérivés de chaque courbe à partir de l'ICE, il est utile d'avoir une estimation de la dérivée directement : c'est ce que fait la courbe d-ICE. La procédure d'estimation de la $d-ICE$ est donnée par l'algorithme suivant :

- Entrée : les données d'entraînement : $(x_j^{(i)})_{1 \leq i \leq n, 1 \leq j \leq p}$, le modèle ajusté $\hat{f}$, les estimations $f_S^{(1)}, ..., f_S^{(n)}$ données par l'algorithme d'ICE, et D une fonction donnant une estimation numérique des dérivées partielles.

- Pour i=1,...,n :
  - $d\hat{f}_S^{(i)} = 0_{n\times 1}$
  - $x_C = x_C^{(i)}$: on fixe les colonnes C à la i-ième observation.
  - Pour l=1,...n :
    - $x_S = x_S^{(l)}$
    - $d\hat{f}_{S,l}^{(i)} = D(\hat{f}(x_S, x_C))$
- Output : $d\hat{f}_S^{(1)}, ..., d\hat{f}_S^{(n)}$ (estimations des dérivées partielles par rapport à x_S de chaque courbe ICE)

Ainsi, l'interprétation d'un graphique d-ICE est la suivante : si il n'y a pas d'interaction entre les variables x_S et x_C dans $\hat{f}$, toutes les courbes sont équivalentes et une seule ligne est présente tandis qu'en cas d'interaction : les lignes de dérivées seront hétérogènes. On peut alors détecter des possibles interactions entre les variables à l'aide de ce graphique, en observant les régions d'hétérogénéité dans les dérivées estimées.

Exemple précédent

Dans l'exemple précédent, comme il n'y pas d'interaction entre les variables X_1 et X_2, le d-ICE devrait présenter qu'une seule ligne. Ceci est bien confirmé sur la figure 4.4.

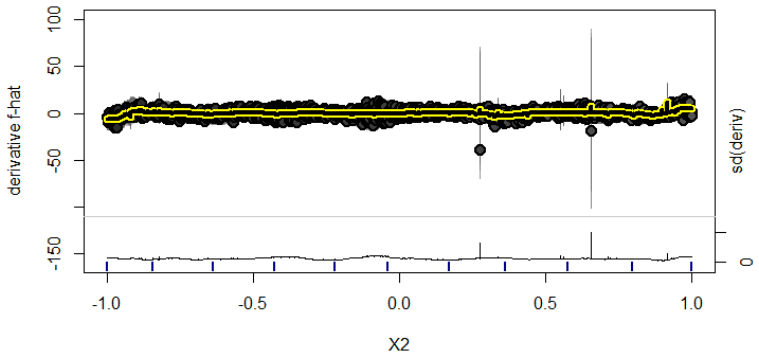

FIGURE *4.4: Graphique de d-ICE*

Autre exemple sur des données réelles

Considérons un autre exemple sur la base de données *Pima* de la librairie MASS de R. La variable à expliquer est la présence ou non de diabète chez l'individu concerné, à partir de variables explicatives de données corporelles telles que la pression artérielle ou la concentration en glucose. La base d'apprentissage contient 332 individus dont 109 diagnostiqués avec du diabète. On considère que $Y = 1$ correspond au cas d'un individu présentant du diabète et $Y = 0$ lorsque celui-ci n'en a pas. On ajuste un modèle de forêt aléatoire sur ces données. Afin de mieux comprendre la sortie délivrée par le modèle boîte noire, on utilise le graphique ICE et d-ICE. Ces derniers sont représentés sur les figures

4.5 et 4.6. Notons bien que l'ordonnée est le log-odds, c'est-à-dire : $ln(\frac{p}{1-p})$ où p est la probabilité d'être dans la classe 1, i.e. d'être diabétique. La variable *skin* correspond à l'épaisseur de pli de peau du triceps (en millimètres). Ces graphiques permettent une

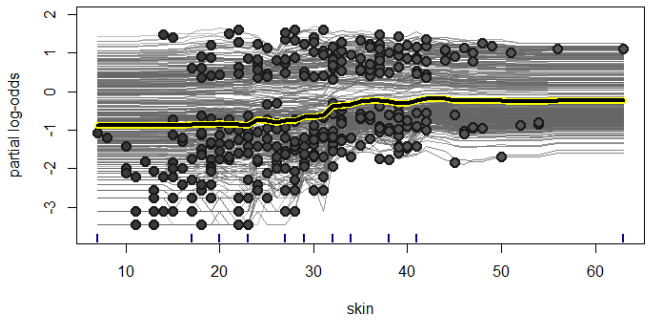

FIGURE *4.5: Graphiques ICE de la variable* skin *du modèle de Random Forest ajusté sur les données Pima*

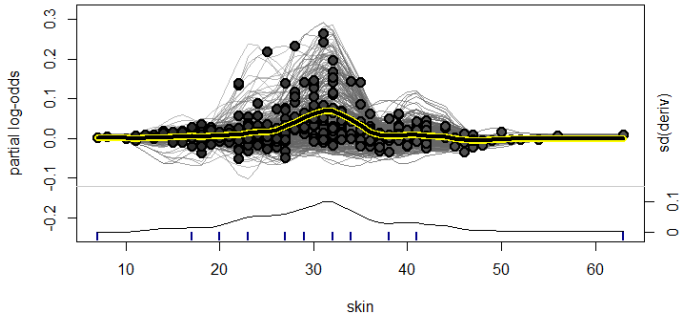

FIGURE *4.6: Graphiques d-ICE de la variable skin du modèle de Random Forest ajusté sur les données Pima*

meilleure compréhension de l'output du modèle. Par exemple, le PDP (ligne jaune de la figure 4.5) nous indique clairement que plus notre variable *skin* prend une valeur élevée plus le risque d'avoir du diabète est élevé. Ceci est cohérent avec le fait que cette variable est un *proxy* de l'obésité, qui est très corrélé avec le diabète. L'ICE nous fournit une explication plus détaillée que le PDP. On observe que beaucoup d'individus avec une variable *skin* élevée ont une probabilité de diabète qui stagne, c'est-à-dire que la courbe d'ICE devient plate à partir d'un certain seuil. Ceci est confirmé sur le graphique d-ICE de la figure 4.6, sur lequel on observe un pic de la fonction log odds, lorsque le paramètre *skin* vaut environ 28.

4.1.1.3 Avantages et inconvénients

Les courbes ICE présentent globalement les mêmes intérêts que le graphique PDP, notamment la simplicité d'implémentation et d'interprétation. L'autre avantage important est qu'elles ne masquent pas les effets hétérogènes du modèle considéré. Ainsi, en utilisant le graphique PDP, qui fournit un résumé de l'impact d'une variable sur la prédiction du modèle, et les courbes ICE, qui le précisent, on obtient une bonne explication globale des prédictions.

Cependant, tout comme le PDP, les courbes ICE reposent sur une hypothèse d'indépendance entre les différentes variables, et ne tient pas compte de leur distribution réelle. Un autre inconvénient est le fait de ne pouvoir représenter ses courbes qu'en 2 ou 3 dimensions, du fait que l'humain ne sait pas se représenter des dimensions supérieures. De plus, le graphique contenant toutes les courbes ICE est vite surchargé lorsque le nombre d'individus étudié est grand. Une solution à ce dernier problème est présentée dans la partie qui suit.

4.1.1.4 VINE : Visual INteraction Effect

Principe général

L'article [15] *VINE : Visualizing Statistical Interactions in Black Box Models* de M. Britton propose une alternative aux graphiques de PDP et ICE. L'idée est de fournir une explication régionale (c.f [4.1]), c'est-à-dire à cheval entre une explication locale (comme l'ICE) et une explication globale (comme le PDP). Pour cela, on rassemble les courbes ICE similaires dans des clusters, et on n'affiche qu'une courbe par cluster. Ceci permet d'avoir un graphique moins chargé que l'ICE tout en ayant une meilleure compréhension de la complexité du modèle que le PDP. Les contributions apportées par cet article sont :

- Un nouvel algorithme afin de générer des explications régionales des prédictions d'un modèle, en se basant sur les courbes de dépendance partielles.
- Une visualisation interactive, appelée VINE, qui permet à l'utilisateur d'explorer ces explications.
- Une méthode d'évaluation de la fidélité de l'explication visuelle fournie du modèle considéré.

Approche

L'approche VINE est constituée principalement de trois étapes, à savoir le calcul des clusters, la génération des explications des clusters et la fusion de ces derniers. On peut résumer VINE par cet algorithme : Pour chaque variable explicative x_j :

- Clusteriser les données, en considérant les pentes de chaque courbe ICE comme caractéristique de représentation.
- Générer un prédicat (par exemple : $\{taille < 1.7m\}$ pour chaque cluster, en construisant un arbre de décision à une couche.
- Fusionner les clusters ayant des explications similaires

Un exemple de cet algorithme est fourni sur la figure 4.7.

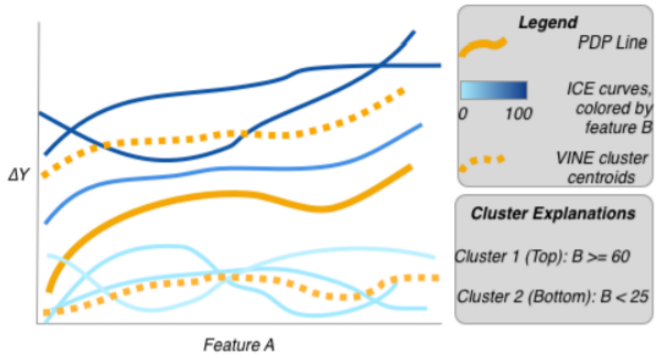

FIGURE *4.7: Courbes fournies par l'algorithme VINE avec deux clusters, superposées aux graphiques de PDP et d'ICE (issus de l'article [15])*

Calcul des clusters La première étape de l'algorithme est le calcul de clusters. Ceci a pour objectif d'éviter d'avoir des graphiques surchargés comme avec la méthode ICE. Pour se faire VINE essaie de clusteriser les courbes similaires et de visualiser une courbe "centroïde" représentative de chaque cluster. On est donc dans de l'apprentissage supervisé, et les données utilisées ne sont pas les variables explicatives mais les courbes ICE calculées auparavant. Il reste alors à choisir l'algorithme pour calculer les clusters. L'article a essayé les méthodes DBSCAN,K-Means, Affinity Propagation, Agglomerative Clustering, et Birch. Ils sont arrivés à la conclusion que l'algorithme Agglomerative Clustering proposé par J. H. Ward Jr. dans l'article *Hierarchical grouping to optimize an objective function* et l'algorithme Birch introduit dans l'article *Birch : an efficient data clustering method for very large databases* par T. Zhang, sont les plus performants. Un autre choix difficile à réaliser était celui de la métrique pour mesurer la distance entre deux courbes ICE. L'article indique que la distance euclidienne est un mauvais choix alors que la mesure de Slope Similarity (similarité de pente) semble le meilleur choix, avec un temps d'exécution similaire au calcul de la distance euclidienne.

Génération des explications pour chaque clusters Après avoir calculé les clusters des courbes ICE, l'idée est de fournir une explication interprétable par humain de chaque cluster. Pour cela, on cherche les points communs entre ces courbes clusterisées notamment par le biais de ce qui les différencie des autres courbes.

Pour se faire, on construit un arbre de décision, avec un noeud (et deux feuilles) pour chaque cluster créé par l'étape précédente. Si un point se retrouve dans la feuille de gauche, alors il appartient au cluster considéré, sinon il appartient à un autre cluster (one vs all). Ainsi, pour chaque cluster C_k, $1 \leq k \leq n_{clust}$, on dispose d'un arbre entièrement défini par : la caractéristique f_k, la direction Dir_k ("$\leq$" ou " $>$ ") et la valeur V_k du noeud de séparation. Par exemple, considérons que l'arbre créé pour le cluster C_k est donné par la figure 4.8. Alors la caractéristique f_k est la *taille*, la direction Dir_k est " $>=$ " et la valeur V_k est 149 cm. Ainsi un homme de taille supérieure à 1.49m, se

trouvera dans le cluster k, sinon il se retrouvera dans un autre cluster.

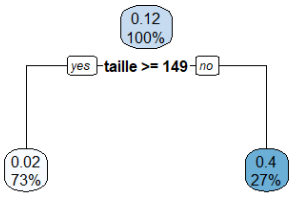

FIGURE *4.8: Exemple d'arbre de décision de profondeur 1*

L'arbre de décision ainsi créé, et le noeud de séparation associé, représentent une bonne explication des caractéristiques rendant le cluster "unique".

Fusion des clusters On dispose à cette étape de l'algorithme, de différents clusters et d'arbres de décision associés. Comme indiqué précédemment, nous ne voulons pas avoir trop de clusters pour ne pas surcharger le graphique que l'on va fournir. C'est pourquoi le choix de clusters est assez difficile en pratique. L'idée à présent est d'opérer à une fusion de clusters, en rassemblant ceux qui présentent des caractéristiques similaires. On considère que l'on s'intéresse à une certaine variable, pour laquelle F_{max} et F_{min} sont respectivement la valeur maximale et la valeur minimale de cette variable dans la base d'apprentissage. L'algorithme proposé par l'auteur de l'article sur VINE est le suivant :

- Pour tout cluster C_i de $C_1, ..., C_{n_{clust}}$:
  - Pour tout cluster C_j de $C_{i+1}, ..., C_{n_{clust}}$:
    - Si $f_i = f_j$ et $Dir_i = Dir_j$ et $\frac{V_i - V_j}{F_{max} - F_{min}} \leq 0.05$: alors on fusionne les clusters C_i et C_j.

Exemple

Reprenons l'exemple des données de Boston ajustées avec un modèle de forêt aléatoire. Les résultats obtenus avec deux et trois clusters se trouvent sur les figures 4.9 et 4.10.

4.1.2 Graphique des effets locaux accumulés (ALE)

4.1.2.1 Principe général

Le graphique des effets locaux accumulés (Accumulated Local Effects Plot) a pour objectif de corriger le PDP, notamment lorsque l'on possède des variables explicatives corrélées entre elles. Elle a été introduite dans l'article [5] *Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models*. L'ALE va alors décrire

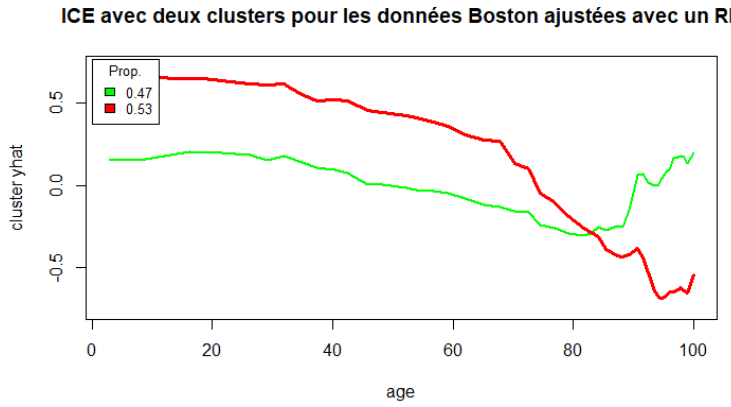

FIGURE *4.9: ICE associé à la variable* age *pour le modèle de RF ajusté sur les données Boston, avec deux clusters*

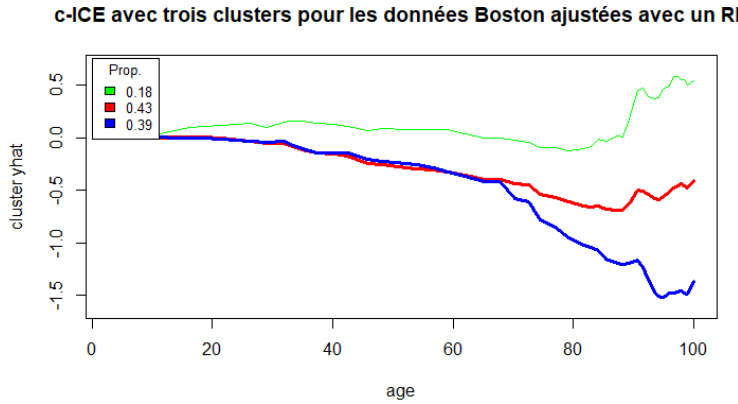

FIGURE *4.10: c-ICE associé à la variable* age *pour le modèle de RF ajusté sur les données Boston, avec trois clusters*

comment les variables influent sur la prédiction du modèle en moyenne, sans biais et plus rapidement que le PDP. Tout comme ce dernier, l'ALE est une approche globale (c.f schéma 4.1).

Comme indiqué ci-dessus, le problème de la PDP est lorsque l'on dispose de variables fortement corrélées entre elles, car une moyenne des prédictions est faite à partir de combinaisons de variables qui peuvent être irréelles. On peut reprendre l'exemple de la taille et du poids comme variables explicatives : on pourrait avec la PDP considérer des individus de 30kg et faisant plus de 2 mètres, ce qui n'est pas réellement possible. Ceci introduit un biais sur l'effet estimé des variables. Le graphique 4.11 montre que la PDP dans le cas de variables très corrélées ne tient pas compte de la distribution.

Une première idée pour éviter ce problème serait de moyenner à partir de la distribu-

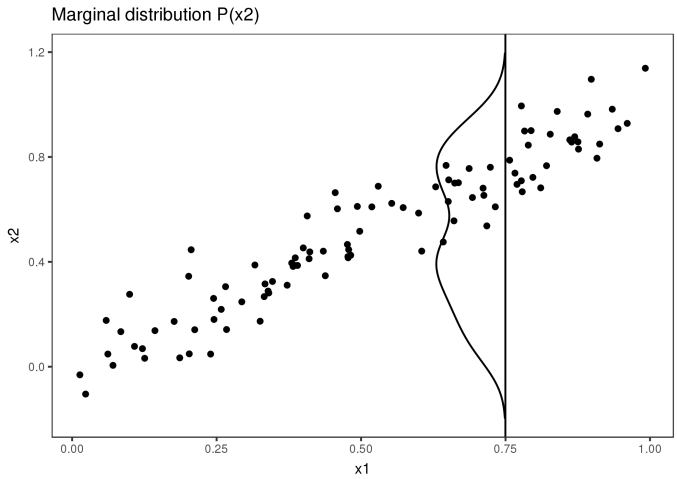

FIGURE *4.11: Cas du calcul de la PDP avec des variables très corrélées lorsque l'on fixe x_1=0.75 (issu du site [51])*

tion conditionnelle, ce qui signifie que pour une valeur x_1 donnée, on réalise la moyenne des instances avec des valeurs similaires (en terme de corrélation) à x_1. Le graphique 4.12 résume cette idée.

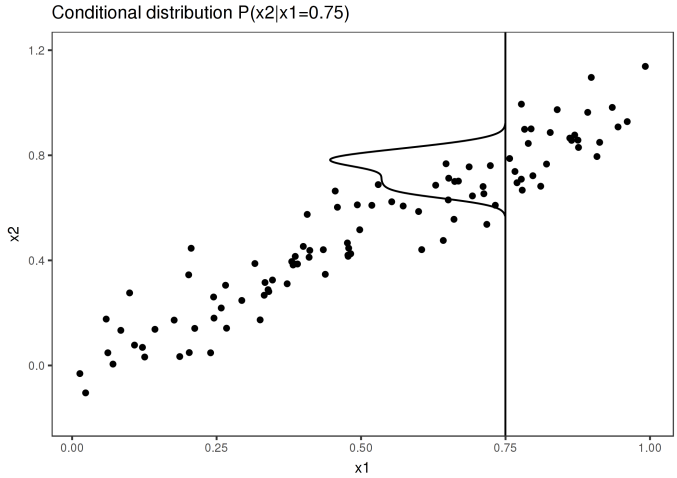

FIGURE *4.12: M-plot dans le cas de deux variables très corrélées en utilisant la distribution conditionnelle de x_2 sachant $x_1 = 0.75$ (issu du site [51])*

Le problème auquel on fait face en faisant cette moyenne des prédictions à partir du M-plot est d'estimer l'effet combiné des deux variables et non d'une seule, à cause de leur corrélation. Pour illustrer ceci, considérons l'exemple où l'on souhaite prédire Y le prix d'une maison, à partir des variables X_1 (surface de la maison) et X_2 (nombre de chambres). En supposant que la variable de surface de la maison n'a pas d'effet

sur la prédiction mais que seul le nombre de chambre en a un. Comme le nombre de chambre augmente avec la surface, le M-plot montrera alors que la surface de la maison fera augmenter son prix. Il nous faut alors définir un nouveau graphique, qui résout ce problème : il s'agit de ALE (le graphique des effets locaux accumulés). Ce dernier repose également sur la distribution conditionnelle des variables mais calcule les différences en prédiction à la place de moyennes. En reprenant l'exemple de prédiction du prix d'une maison : si on veut comprendre l'effet associé à une surface de 30 m^2, la méthode ALE va utiliser toutes les instances (maisons) de $30m^2$ et regarder la différence en prédiction lorsqu'on change sa surface de $31m^2$ à $29m^2$. Ceci va alors nous donner l'effet de la variable de surface et non l'effet combiné avec les variables corrélées comme dans le M-plot, grâce à la différence qui est réalisée. Le graphique 4.13 résume l'idée de calcul de l'ALE. On divise tout d'abord la variable X_1 en intervalles. Pour chaque instance dans un intervalle, on calcule la différence en prédiction lorsqu'on remplace la valeur de X_1 par la borne supérieure et inférieure de l'intervalle considéré. Ensuite, toutes ces différences sont accumulées et centrées, ce qui donne la courbe d'ALE.

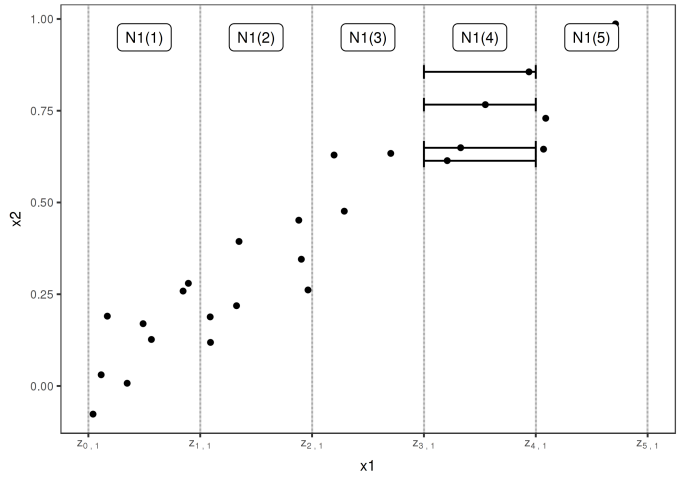

FIGURE *4.13: Explication du calcul de l'ALE avec des variables X_1 et X_2 très corrélées (issu du site [51])*

4.1.2.2 Formalisme mathématique

Pour bien comprendre l'ALE, reprenons la formule de la PDP. La PDP associée aux variables X_S repose sur le calcul de :

$$\hat{f}_{X_S,PDP}(x_S) = \mathbb{E}_{X_C}[\hat{f}(x_S, X_C)] = \int_{x_C} \hat{f}(x_S, x_C)\mathbb{P}_{X_c}(x_C)dx_C \tag{4.3}$$

pour chaque point x_S de la distribution marginale de la variable X_S. Le M-plot, lui repose sur le calcul de la moyenne des prédictions sur la distribution conditionnelle, à savoir :

$$\hat{f}_{X_S,M-plot}(x_S) = \mathbb{E}_{X_C|X_S}[\hat{f}(X_S, X_C)|X_S = x_S] = \int_{x_C} \hat{f}(x_S, x_C)\mathbb{P}(x_C|x_S)dx_C \tag{4.4}$$

Finalement, pour le graphique ALE, il nous faut définir une borne $z_{0,1}$ pour délimiter l'intervalle sur lequel on va faire la moyenne des différences de prédiction. Le calcul est alors le suivant (avant de centrer le résultat) :

$$\hat{f}_{X_S,ALE}(x_S) = \int_{z_{0,1}}^{x_S} \mathbb{E}_{X_C|X_S}[\hat{f}^{(S)}(X_S, X_C)|X_S = x_S]dz_S = \int_{z_{0,1}}^{x_S}\int_{x_C} \hat{f}^{(S)}(x_S, x_C)\mathbb{P}(x_C|x_S)dx_C dz_S \tag{4.5}$$

où $\hat{f}^{(S)}(x_S, x_C) = \frac{\partial \hat{f}(x_S,x_C)}{\partial x_S}$ est le gradient.

4.1.2.3 Estimation de l'ALE

Décrivons à présent comment l'ALE est estimée en pratique, dans le cas où l'on veut comprendre le comportement d'une seule variable numérique x_j ($S = \{j\}$) où $j \in \mathbb{N}$. Soit $x \in \mathbb{X}$, on divise la variable x_j en plusieurs intervalles, à savoir : $[z_{0,j}, z_{1,j}]$, $[z_{1,j}, z_{2,j}]$,...,$[z_{k_j(x)-1,j}, z_{k_j(x),j}]$, avec $k_j(x)$ le nombre d'intervalles et $z_{0,j} < z_{1,j} < ... < z_{k_j(x),j}$. Pour $k \in \{1, ..., k_j(x)\}$, on note $N_j(k)$ l'ensemble des individus de la base d'apprentissage pour lesquels la variable x_j est dans l'intervalle numéro k : $[z_{k-1,j}, z_{k,j}]$, et $n_j(k)$ le nombre d'individus dans $N_j(k)$. Alors l'ALE au point $x \in \mathbb{X}$ associée à la variable j est estimée par la formule :

$$\hat{\tilde{f}}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \left[f(z_{k,j}, x_{\setminus j}^{(i)}) - f(z_{k-1,j}, x_{\setminus j}^{(i)})\right] \tag{4.6}$$

Le terme d'effets locaux accumulés s'expliquent clairement sur cette formule : sur chaque intervalle on mesure la différence de prédiction "locale" puis on somme sur tous les intervalles, afin d'avoir l'effet "accumulé". En réalité, la véritable définition de l'ALE centre le terme précédent afin d'avoir un effet moyen à 0, il en découle la formule suivante :

$$\hat{f}_{j,ALE}(x) = \hat{\tilde{f}}_{l,ALE}(x) - \frac{1}{n}\sum_{l=1}^{n} \hat{\tilde{f}}_{l,ALE}(x) \tag{4.7}$$

$$= \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \left[f(z_{k,j}, x_{\setminus j}^{(i)}) - f(z_{k-1,j}, x_{\setminus j}^{(i)})\right] - \frac{1}{n}\sum_{l=1}^{n} \hat{\tilde{f}}_{l,ALE}(x) \tag{4.8}$$

Le fait de centrer la formule permet d'interpréter l'ALE comme l'effet d'une variable sur la prédiction en comparaison de la prédiction moyenne sur la base de données d'apprentissage. Ainsi, si on obtient une ALE de -2 à un certain point x lorsque $x_j = 3$, cela signifie que lorsque la j-ième variable vaut 3, alors la valeur de prédiction est inférieure de 2, en comparaison de la prédiction moyenne.

Les intervalles dans la formule de l'ALE 4.6 sont souvent choisis comme différents quantiles de la distribution de la variable x_j qu'on étudie. Cela permet d'avoir autant d'individus dans chaque intervalle, mais présente l'inconvénient d'avoir des intervalles de tailles très variables, notamment si la queue de la distribution est lourde.

4.1.2.4 Exemples montrant l'intérêt de l'ALE face au PDP

Exemple 1

Reprenons l'exemple présenté par Molnar sur son site *Interpretable machine learning* [51], en le modifiant légèrement. On souhaite mettre en évidence l'intérêt de l'ALE en comparaison de la PDP lorsque les variables explicatives sont très corrélées. Pour se faire, considérons par exemple un couple de variables aléatoires $U = (X1, X2)$ qui suit une copule de Gumbel de paramètre $\alpha = 6$, c'est-à-dire que la fonction de répartition C de la loi du couple U est définie par :

$$\forall (u_1, u_2) \in [0,1]^2, C(u_1, u_2) = exp(-[(-ln\ u_1)^\alpha + (-ln\ u_2)^\alpha]^{\frac{1}{\alpha}}) \tag{4.9}$$

Celle-ci est une forme particulière de la classe des copules archimédiennes. Le Tau de Kendall, une forme de corrélation ne dépendant que du rang des observations, est donné par : $\tau(\alpha) = 1 - \frac{1}{\alpha}$. Ici, $\alpha = 6$, ce qui donne un taux de Kendall d'environ 0.83 signifiant une forte corrélation entre les deux variables X_1 et X_2. Considérons le modèle de prédiction suivante :

$$\hat{Y} = \begin{cases} 2\ si\ X_1 > 0.7\ et\ X_2 < 0.3 \\ X_1 + X_2\ sinon \end{cases} \tag{4.10}$$

Le graphique 4.14 représente les prédictions faites par le modèle sur un échantillon de taille 25. On observe sur ce graphique que la zone $X_1 > 0.7$ $X_2 < 0.3$ n'est représenté par aucun point de la base d'apprentissage. A présent, nous représentons sur le graphique 4.15 le PDP et l'ALE ainsi obtenus. Sur celui-ci on observe bien une relation linéaire entre X_1 et $\hat{Y}$ pour l'ALE. Cela vient du fait que la zone (jaune) non représentée sur le graphique 4.14 n'est pas prise en compte dans le calcul de l'ALE, car celui-ci n'utilise que des points de la distribution observée des variables. Ce n'est pas le cas pour le PDP, sur lequel on observe que la relation linéaire est rompue lorsque $X_1 = 0.7$ sur le graphique en haut à gauche (PDP associé à la variable X_1) et lorsque $X_2 = 0.3$ sur le graphique en haut à droite (PDP associé à la variable X_2). Ceci est cohérent avec le fait que le calcul du PDP repose sur une moyenne de plusieurs points, indépendamment de la distribution réelle des variables.

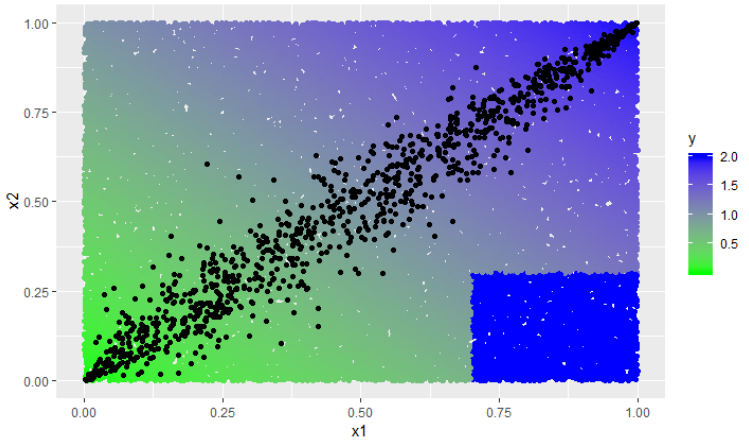

FIGURE *4.14: Modèle de prédiction suivant la valeur des variables X_1 et X_2 sur un échantillon de taille 1000*

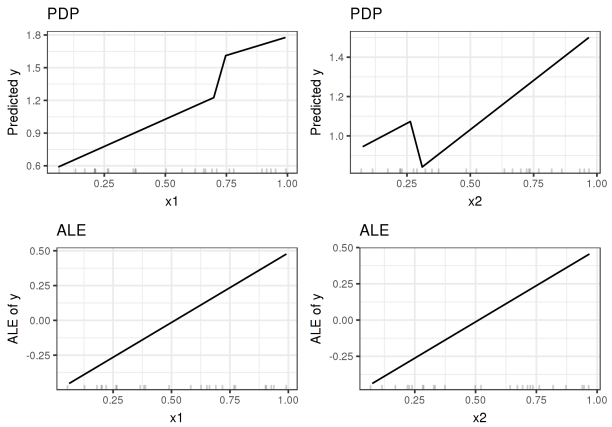

FIGURE *4.15: PDP et ALE associés aux variables X_1 et X_2 obtenus à partir de l'échantillon de la figure 4.14*

Exemple 2

Considérons un exemple introduit dans l'article [5]. Soit X_1 et X_2 deux variables définies par : $X_1 = U + \epsilon_1$ et $X_2 = U + \epsilon_2$ avec $U \sim U[0,1]$ et $\epsilon_1, \epsilon_2 \overset{iid}{\sim} N(0, 0.05^2)$ avec U indépendant de ϵ_1 et ϵ_2. On définit la variable réponse que l'on souhaite prédire par : $Y = X_1 + X_2^2$. On simule un échantillon de taille $n = 200$ $(x_1^{(i)}, x_2^{(i)}, y^{(i)})_{1 \leq i \leq n}$ pour lequel on ajuste un arbre de décision avec un paramètre de complexité $cp = 0.0001$. Le nuages de points entre les variables X_1 et X_2 est donné sur la figure 4.16 tandis que l'arbre de décision créé est sur la figure 4.17. De la même manière que dans l'exemple 1, la dépendance entre les variables X_1 et X_2 va rendre le graphique de PDP erroné, alors que l'ALE traduit la relation entre les variables avec une grande précision. Nous pouvons observer ceci sur la figure 4.18 , sur laquelle est superposé le graphique de PDP,

d'ALE et l'effet réel des variables X_1, X_2 sur la réponse fournie par le modèle d'arbre de décision. La ligne continue noire représente l'effet réel, la ligne bleue le graphique ALE, la ligne rouge le graphique PDP et la ligne noire en pointillée le M-plot. On remarque la proximité entre les courbes de l'effet réel et d'ALE.

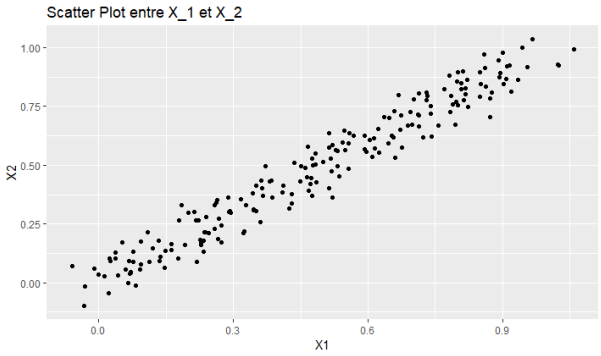

FIGURE *4.16: Scatter Plot entre X_1 et X_2 définis dans l'exemple 2*

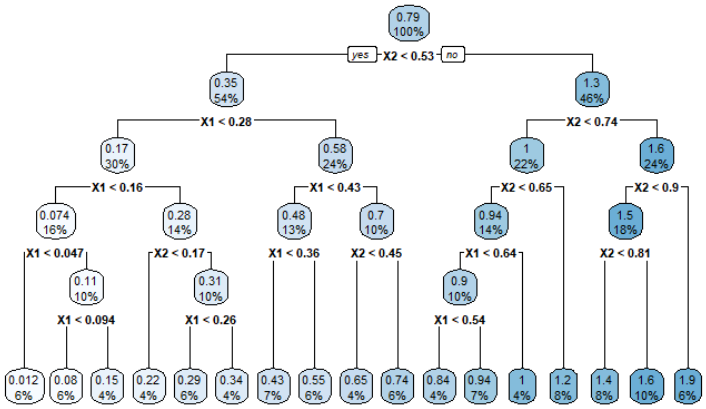

FIGURE *4.17: Arbre de décision ajusté sur les données simulées de (X_1, X_2, Y) de l'exemple 2*

Exemple 3

L'exemple 3 est assez proche de l'exemple 2 dans la construction des variables d'intérêt. En effet X_1 et X_2 sont définies de la même manière tandis qu'un bruit indépendant de X_1 et X_2 est ajouté à Y, à savoir : $Y = X_1 + X_2^2 + \epsilon$ avec $\epsilon \sim N(0, 0.1^2)$. On réalise toujours une simulation de taille $n = 200$ et on ajuste cette fois-ci un réseau de neurones, ajusté à l'aide d'une validation croisée, avec 10 couches couchées. Cette fois-ci, on répète notre ajustement 50 fois, avec de nouvelles simulations et on superpose les différents graphiques de PDP et d'ALE obtenus. Cet exemple a pour objectif de montrer la stabilité

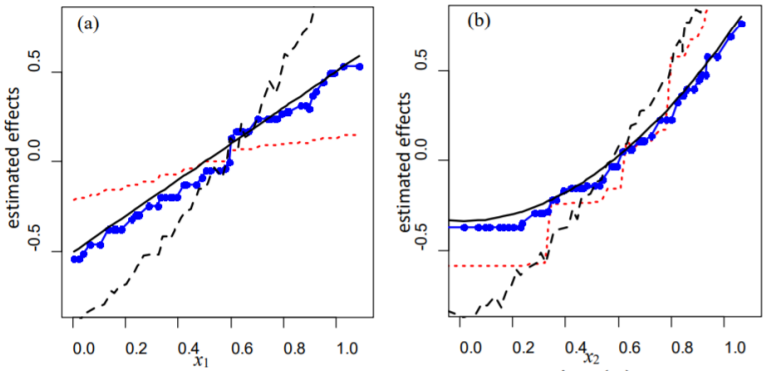

FIGURE *4.18: Comparaison des graphiques de PDP et d'ALE par rapport à l'effet réel des variables sur la réponse du modèle d'arbre de décision mis en place dans l'exemple 2*

du graphique ALE à travers différentes simulations bruitées, ce qui n'est pas le cas pour le graphique de PDP. Les résultats ainsi obtenus sont donnés par la figure 4.19.

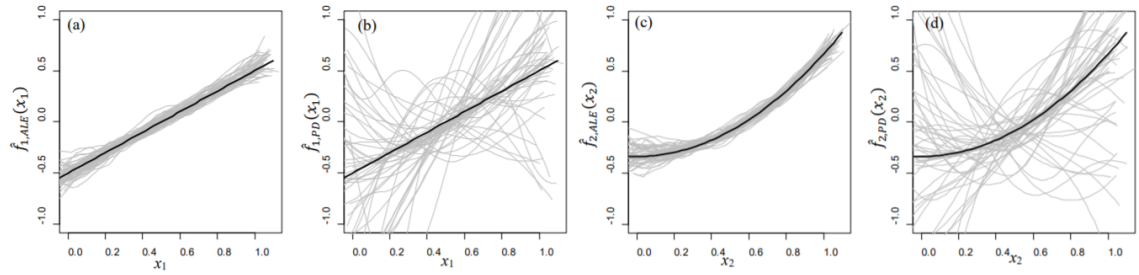

FIGURE *4.19: Superposition des différents graphiques de PDP (b et d) et d'ALE (a et c), associés respectivement aux variables x_1 et x_2 obtenus pour 50 simulations différentes de l'exemple 3*

4.1.2.5 Exemple sur des données réelles

Appliquons à présent la méthode ALE sur des données réelles, à savoir celles de Boston, du package "MASS" du logiciel R vu précédemment. On rappelle que la variable Y à prédire est la valeur médiane des logements occupés par leurs propriétaires, en milliers de dollars. On dispose de plusieurs variables explicatives dont, une variable binaire *chas* valant 1 si le tracé est lié à la rivière, 0 sinon, et la variable *lstat* donnant le statut inférieur de la population, en pourcentage. Pour le cas de la variable catégorielle *chas*, il existe une adaptation de la formule d'ALE vue précédemment pour les variables numériques. On met en place un modèle de forêt aléatoire, avec 50 arbres. Le graphique 4.20 affiche l'ALE obtenue respectivement pour les variables *chas* et *lstat*.

On peut également calculer l'ALE pour un ensemble de variables S. Affichons les résultats obtenus lorsque $S = \{lstat, crim\}$ (figure 4.21).

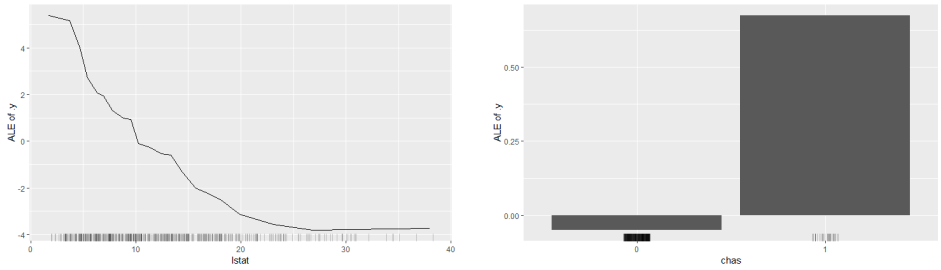

FIGURE *4.20: ALE associée aux variables* lstat *et* chas *obtenues pour le modèle de forêt aléatoire afin de prédire la variable* medv *à partir de 13 variables explicatives*

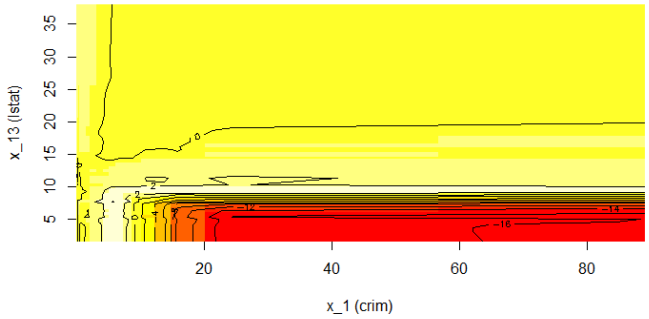

FIGURE *4.21: ALE pour l'ensemble de deux variables* $S = \{lstat, crim\}$ *pour les données de Boston, ajustées avec une forêt aléatoire de 50 arbres*

4.2 Importance des variables et interaction entre les variables

4.2.1 Importance des variables

La notion d'importance des variables dans un modèle a été l'objet de nombreuses définitions différentes. Nous avons dans la partie précédente, deux définitions dans le cas de la régression linéaire et dans le cas des arbres de décisions. Ici, nous nous intéressons à une nouvelle définition de l'importance des variables, calculée indépendamment du modèle considéré que l'on note PFI (*Permutation Feature Importance*). Elle a été introduite par A. Fisher et Al. en 2018 dans l'article [25]. L'idée est de considérer qu'une variable est d'autant plus importante que l'erreur de prédiction du modèle augmente après avoir permuté les valeurs de cette variable considérée. En effet, si la prédiction d'un modèle est grandement modifiée lorsque l'on mélange les valeurs d'une variable, cela signifie que le modèle est sensible aux variations de cette variable et donc qu'elle joue un rôle prépondérant dans le modèle. Inversement, une variable qui pour laquelle une modification de ses valeurs n'impactera que peu la prédiction du modèle ne sera pas

considérée comme importante. Soit f la fonction associée au modèle que l'on souhaite interpréter. On se place toujours dans le cas où nous disposons de $n \in \mathbb{N}$ observations : $(x^{(1)}, ..., x^{(n)}) \in (\mathbb{R}^p)^n$ et $(y^{(1)}, ..., y^{(n)}) \in \mathbb{R}^n$. On note L la fonction d'erreur utilisée, par exemple : $L(y, f(x)) = \frac{1}{2n} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)})^2$ La procédure pour le calcul de l'importance des p variables du modèle est alors détaillé dans l'algorithme suivant :

- Calcul de l'erreur d'origine du modèle : $err_1 = L(y, f(x))$
- pour $j = 1$ à p :
  - On choisit aléatoirement une permutation σ de $\{1, ..., n\}$ dans $\{1, ..., n\}$. On définit une nouvelle matrice $(\tilde{x}_j^{(i)})_{\substack{1 \leq j \leq p \\ 1 \leq i \leq n}}$ de variables d'entrées, par la formule : $\forall i \in \{1, ..., n\}, \forall k \in \{1, ..., p\}, \tilde{x}_k^{(i)} = \begin{cases} x_k^{(i)} \ si \ k \neq j \\ x_j^{(\sigma(i))} \ si \ k = j \end{cases}$. c'est-à-dire que l'on permute (avec σ) les n observations de la variable x_j et on laisse les autres observations inchangées.
  - On estime l'erreur commise par le modèle sur cette nouvelle matrice d'entrée, à savoir : $err_j = L(y, f(\tilde{x}))$.
  - On calcule l'importance de la variable x_j par la formule : $FI^{(j)} = err_j / err_1$
- Sortie de l'algorithme : $FI^{(1)}, ..., FI^{(p)}$, triées par ordre décroissant.

Une variante de cet algorithme consiste à considérer plusieurs permutations à chaque itération, plutôt qu'une seule, mais cela ajoute du temps de calcul.

La question que l'on peut se poser est de réaliser le calcul de l'importance des variables sur la base d'apprentissage B_a ou la base de test B_t. Le calcul sur B_a permet de savoir à quel point le modèle compte sur chaque variable pour faire une prédiction. Le choix de la base de test permet de savoir à quel point une variable contribue à la performance du modèle sur des données non entraînées.

4.2.1.1 Avantages

Les avantages de cette méthode d'importance des variables sont nombreux, on peut citer notamment :

- Interprétation facile : plus l'erreur est grande plus l'information est détériorée quand on modifie la valeur de la variable considérée.
- Donne un aperçu global du comportement du modèle, tout comme avec les coefficients dans le modèle de régression linéaire.
- Critère comparable entre différents modèles.
- C'est une méthode qui prend en compte à la fois les effets de la variable et les effets d'interaction entre celle-ci et les autres variables. Ceci peut également être vu comme un inconvénient.
- Le calcul ne nécessite pas de ré-entraîner le modèle : on a donc un gain de temps en comparaison avec d'autres méthodes.

4.2.1.2 Inconvénients

On peut citer des inconvénients à cette méthode, à savoir :
— Le choix entre les bases d'apprentissage et de test n'est pas très clair.
— Le résultat fournit par l'algorithme peut varier grandement, du fait du hasard introduit par les permutations.
— L'ajout d'une variable corrélée à une autre diminue l'importance de la variable considérée.
— Les permutations peuvent fournir des instances irréelles. En effet, lorsque l'on permute une variable au sein d'une instance, on ne fait pas attention au fait que la nouvelle instance soit réellement observable. Ceci est le même problème que celui observé avec la PDP. Considérons par exemple le cas où on dispose des variables explicatives de poids et de taille d'un homme. Si on réalise une permutation comme dans l'algorithme ci-dessus, on peut se retrouver avec un individu de taille 2 mètres et de poids 30kg, ce qui n'est pas possible en réalité.

4.2.1.3 Exemple sur des données réelles

Reprenons l'exemple de l'ajustement d'un modèle de forêt aléatoire sur les données Boston de la librairie MASS de R. L'importance des variables est calculée comme indiqué précédemment et les résultats obtenus sont donnés dans le graphique 4.22.

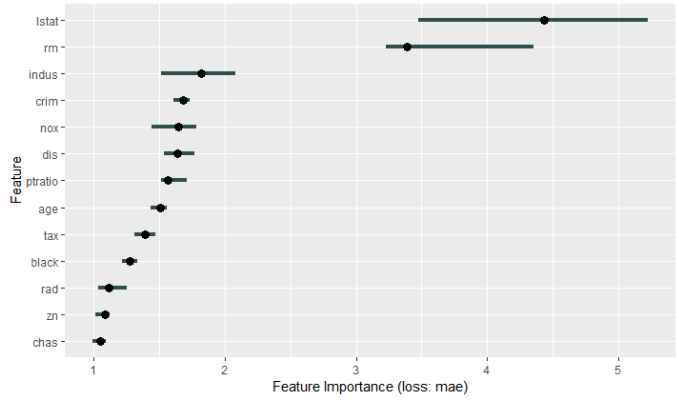

FIGURE *4.22: Importance des variables obtenues pour le modèle de RF ajusté sur les données Boston*

4.2.2 Importance partielle des variables (PI), Importance Conditionnelle Individuelle (ICI) et Importance de Shapley (SFIMP)

Dans cette partie, nous reprenons les concepts introduits dans l'article [17] *Visualizing the Feature Importance for Black Box Models* de Casalicchio et Al. (2018) qui fournit de nouveaux outils pour mesurer l'importance des variables. Tout d'abord, les auteurs de cet article rappelle les différentes méthodes déjà proposées pour le calcul de l'importance des

variables. Ils citent notamment le graphique de dépendance partielle (PDP), le graphique d'espérance conditionnelle individuelle ICE, les valeurs de Shapley, ainsi que la méthode d'importance des variables basée sur des permutations (PFI). Toutes ces méthodes serviront de point de départ aux nouvelles proposées par cet article : le PDP permettra d'introduire le graphique d'importance partielle (PI), l'ICE le graphique d'importance conditionnelle individuelle (ICI) et SHAP l'importance des valeurs de Shapley basée sur les permutations (SFIMP).

4.2.2.1 Notations et rappels

Notations

Notons $\mathbb{X}_P = \mathbb{X}_1 \times ... \times \mathbb{X}_p$ l'espace des p variables explicatives ($p \in \mathbb{N}$), où $P = \{1, ..., p\}$, $\mathbb{Y}$ l'espace de sortie (par exemple $\mathbb{R}$ pour la régression). On note $\mathbb{P}$ la distribution associée à l'espace $\mathbb{X}_P \times \mathbb{Y}$. On considère un modèle $\hat{f}$ qui essaie d'approcher la relation f entre les espaces $\mathbb{X}_P$ et $\mathbb{Y}$ On considère une base de test de taille $n \in \mathbb{N}$: $D = \{(x^{(i)}, y^{(i)})\}_{1 \leq i \leq n}$, constituée d'échantillons supposés iid. On note $X = (X_1, ..., X_p)$ les variables aléatoires associées à l'espace de départ et Y la variable aléatoire de l'espace de sortie. L'erreur de généralisation associée à notre modèle, pour une fonction de perte L choisie, est définie par :

$$GE(\hat{f}, \mathbb{P}) = \mathbb{E}_{\mathbb{P}}[L(\hat{f}(X), Y)]$$

L'estimation de l'erreur de généralisation associée à la base de test D, par la méthode Monte Carlo :

$$\hat{GE}(\hat{f}, \mathbb{P}) = \frac{1}{n} \sum_{i=1}^{n} L(\hat{f}(x^{(i)}), y^{(i)}) \tag{4.11}$$

Rappels

— **Graphique dépendance partielle (PD plot)** Ce graphique permet de mesurer l'effet marginal moyen d'un ensemble de variables sur la prédiction du modèle. Notons $S \subseteq P$ l'ensemble des indices associé aux variables étudiées et $C = P \backslash S$ le complémentaire de S. On alors $X = (X_S, X_C)$ et $\mathbb{X}_P = \mathbb{X}_S \times \mathbb{X}_C$. Pour chaque observation $x \in \mathbb{X}_P$, on note $x_S \in \mathbb{X}_S$ et $x_C \in \mathbb{X}_C$, contenant respectivement les variables de S et de C. La fonction de dépendance partielle (PD) est alors définie par : $\forall x_S \in \mathbb{X}_S,\ f_S(x_S) = \mathbb{E}_{\mathbb{X}_{\mathbb{C}}}[\hat{f}(x_s, x_c)]$ L'estimation de Monte Carlo de la fonction PD est donnée par :

$$\forall x_S \in \mathbb{X}_S,\ \hat{f}_S(x_S) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}_S^{(i)}(x_S) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}) \tag{4.12}$$

Le graphique PDP s'obtient alors en affichant le nuage de points $(x_S^{*(k)}, \hat{f}_S(x_S^{*(k)}))_{1 \leq k \leq m}$, pour certains points $x_S^{*(k)}$ choisis (où $m \in \mathbb{N}$). Un graphique résumant le calcul du graphique de PDP est proposé dans l'article et est donné par la figure 4.23.

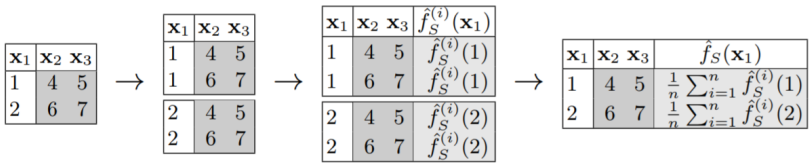

FIGURE *4.23: Calcul du graphique PDP sur un exemple simple*

— **Graphique ICE**. L'idée est la même que le graphique de PDP, sauf qu'au lieu de réaliser une moyenne sur toutes les observations, on réalise un graphique par observation. Ceci permet notamment de ne pas masquer les effets hétérogènes entres les variables. Le graphique s'obtient à l'aide d'un nuage de points $(x_S^{*(k)}, \hat{f}_S^{(i)}(x_S^{*(k)}))_{1\leq k\leq m}$ pour chaque observation $i \in \{1, ..., n\}$ où $\hat{f}_S^{(i)}(x_S) = \hat{f}(x_S, x_C^{(i)})$

— **Valeurs de Shapley** (vu dans les parties suivantes). On se place dans la théorie des jeux. On considère un jeu de coalition avec P joueurs, formant une coalition $S \subseteq$. Chaque coalition a une valeur de paiement (payout). On note $v : 2^P \to \mathbb{R}$, la fonction qui aux 2^p coalitions possibles associe leur payout. La valeur de Shapley est la seule qui assigne de manière "juste" un score à chaque joueur suivant leur contribution dans toutes les coalitions possibles. Dans le contexte de la prédiction d'une variable à partir d'une observation x, les joueurs sont les variables explicatives, le jeu est la prédiction et la fonction de payout est la dépendance partielle, que l'on décale par rapport à la prédiction moyenne i.e :

$$v(x_S) = \mathbb{E}_{X_C}[\hat{f}(x_S, X_C)] - \mathbb{E}_X[\hat{f}(X)] = f_S(x_S) - f_\emptyset(x_\emptyset) \tag{4.13}$$

Ainsi, pour chaque coalition S on définit la contribution marginale de la variable $j \in \{1, ..., p\}$ par :

$$\Delta_j(x_S) = v(x_{S\cup\{j\}}) - v(x_S) = f_{S\cup\{j\}}(x_{S\cup\{j\}}) - f_S(x_S) \tag{4.14}$$

Soit $\Sigma = (\sigma_1, ..., \sigma_{p!})$ l'ensemble des permutations de $\{1..., n\}$, alors pour $\sigma \in \Sigma$, on note $B_j(\sigma)$ tous les indices dans l'ordre qui arrive avant j dans la permutation σ. Si on considère $p = 5$, $j = 3$ et la permutation $\sigma = \{1, 4, 3, 5, 2\}$, alors : $B_j(\sigma) = \{1, 4\}$. Pour une observation x, la valeur de Shapley peut être estimée via :

$$\hat{\phi}_j(x) = \frac{1}{p!}\sum_{\sigma\in\Sigma}\hat{\Delta}_j(x_{B_j(\sigma)}) = \frac{1}{p!}\sum_{\sigma\in\Sigma}\hat{f}_{B_j(\sigma)\cup\{j\}}(x_{B_j(\sigma)\cup\{j\}}) - \hat{f}_{B_j(\sigma)}(x_{B_j(\sigma)}) \tag{4.15}$$

$$= \frac{1}{n.p!}\sum_{\sigma\in\Sigma}\sum_{i=1}^{n}\hat{f}^{(i)}_{B_j(\sigma)\cup\{j\}}(x_{B_j(\sigma)\cup\{j\}}) - \hat{f}^{(i)}_{B_j(\sigma)}(x_{B_j(\sigma)}) \tag{4.16}$$

L'article [66] *A General Method for Visualizing and Explaining Black-Box Regression Models* de Strumbelj et Al. (2011) propose une implémentation rapide en $O(m^2)$ plutôt que $O(n.p!)$.

— **Calcul de l'importance des variables basée sur les permutations**. L'approche pour le calcul de l'importance des variables proposée dans l'article [25] de Fisher et Al. (2018) repose sur une méthode indépendante du modèle étudié et repose sur des permutations des variables explicatives (notée PFI). La formule de PFI, associée à un ensemble d'indices S, est donnée par :

$$PFI_S = \mathbb{E}[L(\hat{f}(\tilde{X_S}, X_C), Y)] - \mathbb{E}[L(\hat{f}(X, Y)] \tag{4.17}$$

On trouve également dans la littérature une autre définition utilisant un quotient :

$$PFI_S = \frac{\mathbb{E}[L(\hat{f}(\tilde{X_S}, X_C), Y)]}{\mathbb{E}[L(\hat{f}(X, Y)]}$$

Dans cette définition $\tilde{X_S}$ est une réplication indépendante de X_S, et donc indépendante de X_C et Y. Le terme $\mathbb{E}[L(\hat{f}(\tilde{X_S}, X_C), Y)]$ correspond à l'erreur de généralisation moyenne lorsque l'on a pertubé X_S par $\tilde{X_S}$. On peut simplifier ce terme en écrivant :*

$$\mathbb{E}[L(\hat{f}(\tilde{X_S}, X_C), Y)] = \mathbb{E}_{(X_C,Y)}\left[\mathbb{E}_{\tilde{X_S}|(X_C,Y)}[L(\hat{f}(\tilde{X_S}, X_C), Y)]\right] \tag{4.18}$$

$$= \mathbb{E}_{(X_C,Y)}\left[\mathbb{E}_{\tilde{X_S}}[L(\hat{f}(\tilde{X_S}, X_C), Y)]\right] \tag{4.19}$$

$$= \mathbb{E}_{(X_C,Y)}\left[\mathbb{E}_{X_S}[L(\hat{f}(X_S, X_C), Y)]\right] \tag{4.20}$$

On en déduit un estimateur par la méthode Monte Carlo, à savoir :

$$\hat{GE}_C(\hat{f}, D) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n}\sum_{k=1}^{n} L\left(\hat{f}(x_S^{(k)}, x_C^{(i)}), y(i)\right) \tag{4.21}$$

Le temps de calcul associé à l'équation 4.21 est $O(n^2)$, ce qui peut devenir très coûteux lorsque n est grand. Une manière équivalente d'écrire l'équation 4.21 repose sur l'ensemble des permutations $\Sigma = (\sigma_1, ..., \sigma_{n!})$de $\{1, ..., n\}$ et s'écrit :

$$\hat{GE}_{C,perm}(\hat{f}, D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{n!} L\left(\hat{f}(x_S^{(\sigma_k^{(i)})}, x_C^{(i)}), y(i)\right) \tag{4.22}$$

L'intérêt de cette formule, bien qu'elle soit de complexité gigantesque $O(n.n!)$, réside dans l'approximation que l'on peut en faire, en utilisant seulement $m \in \mathbb{N}$ permutation sur les $n!$ possibles.

$$\hat{GE}_{C,approx}(\hat{f}, D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{m}\frac{1}{m} L\left(\hat{f}(x_S^{(\sigma_k^{(i)})}, x_C^{(i)}), y(i)\right) \tag{4.23}$$

Ceci consiste à permuter m fois les valeurs prises par les variables associées aux indices de S et moyenner les performances du modèle. On obtient alors une complexité en $O(n.m)$, et peut s'avérer avantageux lorsque n est grand. On dispose à présent de deux manières pour estimer PFI_S de l'équation 4.17, en utilisant soit l'équation 4.21 ou l'équation 4.23 pour l'estimation du premier terme ($\mathbb{E}[L(\hat{f}(\tilde{X_S}, X_C), Y)]$). Pour le deuxième terme($\mathbb{E}[L(\hat{f}(X, Y)]$), on utilise l'équation 4.11. Ainsi on obtient :

$$P\hat{F}I_S = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{k=1}^{n}\Big(L(\hat{f}(x_S^{(k)}, x_C^{(i)}), y(i)) - L(\hat{f}(x^{(i)}), y^{(i)})\Big) \tag{4.24}$$

$$P\hat{F}I_{S,approx} = \frac{1}{n.m}\sum_{i=1}^{n}\sum_{k=1}^{m}\Big(L(\hat{f}(x_S^{(\sigma_k^{(i)})}, x_C^{(i)}), y(i)) - L(\hat{f}(x^{(i)}), y^{(i)})\Big) \tag{4.25}$$

Ce dernier résultat est intéressant car il généralise la formulation de l'importance des variables pour le modèle de forêts aléatoires, si m est le nombre d'arbres utilisés, n le nombre d'échantillons OOB (out of bag) par arbre et $\hat{f}$ devient $\hat{f}_k$ la fonction de prédiction associée à chaque arbre de la forêt (pour $1 \leq k \leq m$).

4.2.2.2 Importance partielle (PI) des variables

L'apport essentiel de cet article est d'adapter les formules d'importance des variables et de valeurs de Shapley à l'échelle locale, c'est-à-dire au niveau d'une observation. Tout comme le graphique ICE qui correspond au graphique de dépendance partielle (PDP) au niveau d'une observation, l'importance conditionnelle locale (ICI) correspond aux termes de la somme introduite dans la formule de PFI de l'équation 4.24. On définit alors :

$$\forall i \in \{1, ..., n\}, \forall x_S \in \mathbb{X}_S, \Delta L^{(i)}(x_S) = L(\hat{f}(x_S, x_C^{(i)}), y(i)) - \underbrace{L(\hat{f}(x^{(i)}), y^{(i)})}_{=L(\hat{f}((x_S^{(i)}, x_C^{(i)}), y^{(i)})} \tag{4.26}$$

Le graphique ICI associé à l'observation $i \in \{1, ..., n\}$ est alors le nuage de points : $(x_S^{(k)}, \Delta L^{(i)}(x_S^{(k)}))_{1 \leq k \leq n}$.

De la même manière que le PDP qui correspond à la moyenne des courbes ICE, on définit la PFI, associée à une variable d'indice $i \in \{1, ..., p\}$, comme la moyenne des courbes ICI, c'est-à-dire :

$$P\hat{F}I_S^{(i)} = \frac{1}{n}\sum_{k=1}^{n}\Delta L^{(i)}(x_S^{(k)})$$

Ce terme peut être vu comme le changement de performance du modèle lorsque l'on modifie la i-ème variable et après avoir marginalisé les variables de l'espace S. On peut également le voir comme la contribution de la variable d'indice i à la PFI globale, c'est-à-dire :

$$P\hat{F}I_S = \frac{1}{n}\sum_{i=1}^{n}P\hat{F}I_S^{(i)}$$

De la même manière que la fonction de dépendance partielle (PD), on définit la fonction d'importance partielle (PI) comme le changement de performance pour une valeur x_S donnée et s'estime via la formule : $\hat{PI}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} \Delta L^{(i)}(x_S)$ Ainsi, le graphique d'importance partielle (PIP) correspond au nuage de points $(x_S^{(k)}, \hat{PI}_S(x_S^{(k)}))_{1\leq k\leq n}$. Le tableau 4.1 résume les notions de ICI et PI, et les similitudes avec les graphiques ICE et PD.

ICE	PD
$\hat{ICE}_S^{(i)}(x_S) = \hat{f}_S^{(i)}(x_S) = f(x_S, x_C^{(i)})$	$\hat{PD}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{ICE}_S^{(i)}(x_S)$
ICI	**PI**
$\hat{ICI}_S^{(i)}(x_S) = \Delta L^{(i)}(x_S) = L(\hat{f}(x_S, x_C^{(i)}), y(i)) - L(\hat{f}(x^{(i)}), y^{(i)})$	$\hat{PI}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{ICI}_S^{(i)}(x_S)$

TABLE *4.1: Résumé des notions de PD, ICE, ICI, PI*

L'article propose un exemple simple de calcul de l'importance partielle (PI) des variables, à partir des courbes ICI. Celui-ci est représenté sur la figure 4.24

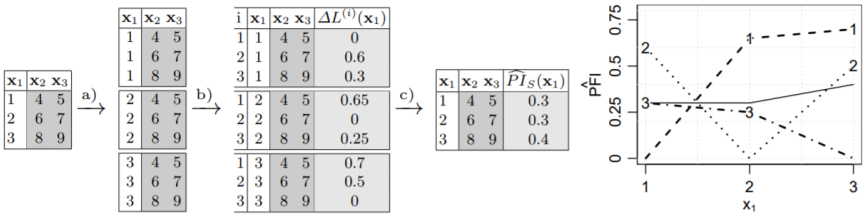

FIGURE *4.24: Exemple simple de calcul des courbes ICI (en pointillé) et PI (ligne pleine) (issu de l'article [17])*

4.2.2.3 Importance des variables basée sur les valeurs de Shapley

Comme nous le verrons dans les parties suivantes, les valeurs de Shapley, introduites dans la théorie des jeux initialement, ont été utilisées pour calculer la contribution des variables dans la prédiction d'un modèle de machine learning (c.f SHAP). Les auteurs de l'article [17] ont repris cette idée, en calculant cette fois-ci la contribution de chaque variable dans la performance du modèle, c'est-à-dire sur l'erreur de généralisation (GE). Cette mesure est appelée SFIMP, en référence à Shapley Feature IMPortance. Le but de ce calcul est de distribuer de manière juste la différence de performance associée à chaque variable entre le scénario où toutes les variables sont utilisées et aucune ne l'est. Ainsi, on définit la fonction de coalition d'un ensemble de variables $S \subseteq P$ comme la différence d'erreur de généralisation lorsque les variables S sont utilisées (les variables C sont marginalisées) et lorsque aucune n'est utilisée, soit :

$$v_{GE}(S) = \hat{GE}_S(\hat{f}, D) - \hat{GE}_\emptyset(\hat{f}, D) \tag{4.27}$$

On utilise l'équation 4.21 pour l'estimation de l'erreur de généralisation. On définit alors la contribution marginale d'une variable $j \in \{1, ..., p\}$ associée à la coalition S le changement de performance induit par l'ajout de cette variable :

$$\Delta_j(S) = v_{GE}(S \cup \{j\}) - v_{GE}(S) = \hat{GE}_{S\cup\{j\}}(\hat{f}, D) - \hat{GE}_S(\hat{f}, D) \tag{4.28}$$

De la même manière que l'équation **??** rappelée ci-dessus, on peut estimer la valeur de Shapley (SFIMP) à l'aide de la formule :

$$SFIMP(j) = \hat{\phi}_j(v_{GE}) = \frac{1}{p!}\sum_{\sigma\in\Sigma}\Delta_j(B_j(\sigma)) = \hat{\phi}_j(v_{GE}) \tag{4.29}$$

$$= \frac{1}{p!}\sum_{\sigma\in\Sigma}\hat{GE}_{B_j(\sigma)\cup\{j\}}(\hat{f}, D) - \hat{GE}_{B_j(\sigma)}(\hat{f}, D) \tag{4.30}$$

Étant donné que cela est coûteux en temps de calcul, l'approximation de l'équation 4.23 peut être utilisée, notamment lorsque p est grand. Rappelons les différentes propriétés vérifiées par la valeur de Shapley théorique ϕ_j :

- *Efficacité* : $\sum_{j=1}^{p}\phi_j = v_{GE}(P)$. La somme des valeurs de Shapley (SFIMP) donne la différence de performance lorsque toutes les variables sont utilisées et lorsque aucune n'est utilisée. On peut alors définir la proportion d'importance expliquée par chaque variable $j \in \{1, ..., p\}$ par : $prop_j = \frac{\phi_j}{\sum_{l=1}^{p}\phi_l}$
- *Symétrie* : Si : $\forall(j,k) \in \{1, ..., p\}^2, \forall S \subseteq \{1, ..., p\}\backslash\{j,k\} v_{GE}(S \cup \{j\}) = v_{GE}(S \cup \{k\})$ alors $\phi_j = \phi_k$. En d'autres termes, si deux variables ont la même contribution marginale pour chaque coalition possible, alors ils ont la même valeur de Shapley.
- *Dummy* : Si $\forall j \in \{1, ..., p\}, \forall S \subseteq P, v_{GE}(S \cup \{j\}) = v_{GE}(S)$ alors : $\phi_j = 0$. En d'autres termes, si une variable n'a aucune contribution marginale sur chaque coalition alors sa valeur de Shapley est nulle.
- *Additivité* et *Linéarité* : Pour deux fonctions de payout v_{GE} et w_{GE}, la valeur de Shapley associée à $v_{GE} + w_{GE}$ est la somme des valeurs de Shapley individuelles. De même la valeur de Shapley associée à une fonction de payout v_{GE} multipliée par une constante $\lambda \in \mathbb{R}$ est le produit de cette constante par la valeur de Shapley du payout $\phi_j(v_{GE})$: $\phi_j(\lambda.v_{GE} + w_{GE}) = \lambda.\phi_j(v_{GE}) + \phi_j(w_{GE})$

On peut alors remarquer qu'utiliser le PFI ou l'erreur de généralisation pour la fonction associée à une coalition fournit les mêmes valeur de Shapley (au signe près). Cela vient de la relation : $\hat{PFI} = \hat{GE}_C(\hat{f}, D) - \hat{GE}_P(\hat{f}, D)$. On peut donc considérer la SFIMP comme une mesure d'importance de variable tout comme la PFI. Cette dernière pose un problème lorsque les variables dans C et dans S sont corrélées. En effet, lorsque la PFI est calculée, les valeurs des variables dans S sont permutées, sans tenir compte de la valeur des variables de C. On peut reprendre l'exemple où l'on possède dans S la taille d'un individu et dans C son poids : lors du calcul de la PFI, des individus de grande taille et de poids léger peuvent alors être construits. De plus, la PFI aura tendance à surestimer l'importance des variables qui intéragissent avec d'autres. L'importance des variables

basée sur la PFI résout ce problème en distribuant de manière équitable l'importance des interactions entre les différentes variables. Ceci permet en outre la comparaison de l'importance des variables pour différents modèles.

4.2.2.4 Exemples

Exemple 1

Considérons un premier exemple pour illustrer l'intérêt des courbes PI et ICI en comparaison de la méthode PFI, qui mesure l'importance des variables de manière globale. Pour cela, les auteurs de l'article suggère l'exemple :

$$Y = X_1 + X_2 + 10X_1\mathbb{1}_{\{X_3=0\}} + 10X_2\mathbb{1}_{\{X_3=1\}} + \epsilon \tag{4.31}$$

Où : $X_1, X_2 \overset{iid}{\sim} N(0,1), X3 \sim B(0.5), \epsilon \sim N(0, 0.5)$. L'idée est de réaliser 10000 simulations et d'ajuster un modèle de forêt aléatoire pour prédire la variable Y à partir de X_1, X_2 et X_3. L'équation 4.31 nous indique que les variables X_1 et X_2 ont autant d'importance étant donné leur symétrie. Ainsi, les résultat fournis par la PFI devraient être semblables pour X_1 et X_2, ce que nous vérifierons ci-dessous. Cependant, la PFI fournit une unique valeur pour chaque variable et ne donne pas d'information en fonction de la valeur des variables. Par exemple, on remarque que lorsque $X_3 = 0$, la variable X_1 va jouer un rôle majeur contrairement à X_2. Lorsque $X_3 = 1$, nous aurons le constat inverse, c'est-à-dire que X_2 jouera un rôle prépondérant et non X_2. Ceci va pouvoir être observé à l'aide des courbes ICI. Celles-ci généralisent les courbes ICE pour l'importance des variables, qui elles permettaient de mettre en évidence les effets hétérogènes entre les variables à la différence du graphique PDP. Cette analyse est bien vérifiée dans le tableau 4.2.2.4 et sur la figure 4.25 qui indiquent la valeur des PFI selon la valeur de X_3, ainsi que les courbes PI.

	PFI pour $X_3 = 0$	PFI pour $X_3 = 1$	PFI globale
X_1	80.80	0.77	40.38
X_2	0.84	81.12	41.2

Un autre exemple, que nous ne détaillons pas ici, est proposé dans l'article [17], cette fois-ci pour mettre en exergue les limites, évoquées ci-dessus, des graphiques PFI et PI dans le cas d'interaction entre les variables. Celui-ci permet de montrer l'intérêt de la SFIMP, en utilisant comme équation 4.32 pour simuler des données, avec interaction.

$$Y = X_1 + X_2 + X3 + X1X2 + \epsilon,\ avec : X_1, X_2, X_2 \overset{iid}{\sim} N(0,1), \epsilon \sim N(0, 0.5) \tag{4.32}$$

4.2.3 Interaction entre les variables

4.2.3.1 Principe général

L'interaction entre les variables (ou feature interaction) apparaît lorsque les prédictions ne sont pas seulement composées de la somme des effets individuels de chaque

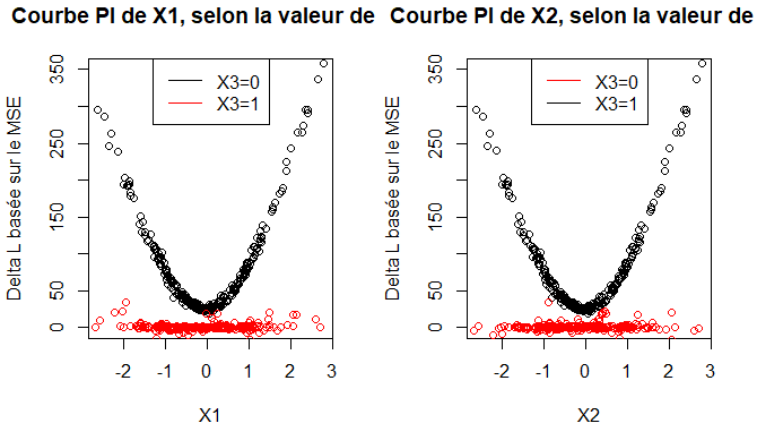

FIGURE *4.25: Courbes PI obtenues pour X_1 et X_2 suivant la valeur de X_3*

variable, mais aussi de termes supplémentaires, correspondant au fait que la valeur d'une variable dépend également de la valeur de l'autre variable. C'est par exemple le cas lorsque nous mettons en place un modèle de régression linéaire "avec interaction" :

- $Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon$ est sans interaction entre X_1 et X_2
- $Y = \beta_1 X_1 + \beta_2 X_2 + \beta_{1,2} X_1 X_2 + \epsilon$ possède une interaction entre les variables explicatives X_1 et X_2

Un autre exemple proposé sur le site [51] de C. Molnar. Si on considère le modèle pour lequel on veut prédire la valeur d'une maison à partir de sa taille (petite ou grande) et de sa localisation (bonne au mauvaise). On dispose des prédictions suivantes : Sur ce

Localisation	Taille	Prédiction (prix de la maison)
Bonne	Grande	300 000
Bonne	Petite	200 000
Mauvaise	Grande	250 000
Mauvaise	Petite	150 000

TABLE *4.2: Tableau de prédiction du modèle 1*

modèle très simple, on peut décomposer la prédiction du modèle de la manière suivante :

- un terme constant (intercept) de 150 000
- un terme d'effet de la taille de 50 000 (0 si la maison est petite, + 50 000 si elle est grande)
- un terme d'effet de la localisation de 100 000 (0 si la localisation est mauvaise, + 100 000 si elle est bonne)

On n'observe pas de terme d'interaction. Considérons un autre exemple. Les prédictions sont les suivantes : Sur ce nouveau modèle on peut décomposer la prédiction de cette manière :

- un terme constant (intercept) de 150 000

Localisation	Taille	Prédiction (prix de la maison)
Bonne	Grande	400 000
Bonne	Petite	200 000
Mauvaise	Grande	250 000
Mauvaise	Petite	150 000

TABLE *4.3: Tableau de prédiction du modèle 2*

— un terme d'effet de la taille de 50 000 (0 si la maison est petite, + 50 000 si elle est grande)
— un terme d'effet de la localisation de 100 000 (0 si la localisation est mauvaise, + 100 000 si elle est bonne)
— un terme d'interaction entre la variable de localisation et de taille de 100 000 (+100 000 si la maison est à la fois grande et bien placée, 0 sinon)

L'idée est à présent de mesurer l'interaction entre les variables en utilisant la théorie de la H-statistique introduite par Friedman et Popescu dans l'article [27] *Predictive Learning via Rule Ensembles.*

4.2.3.2 H-statistique de Friedman

On se place avec les mêmes notations que pour la partie sur le PDP, à savoir : x_S représente le sous-ensemble de variables que l'on étudie, x_C le reste des variables ($C = \{1, ..., n\} \backslash S$), $\hat{f}$ le modèle de boîte noire que l'on étudie. Pour tout $j \in \{1, ..., p\}$, on note PD_j la fonction de dépendance associée à la variable x_j et PD_{-j} la fonction de dépendance associée à toutes les variables sauf x_j. On note également, pour $j, k \in \{1, ..., p\}$, $PD_{j,k}$ la fonction de dépendance associée aux variables x_j et x_k. On fixe $j, k \in \{1, ..., p\}$. Rappelons que nous estimons la fonction de dépendance à l'aide de la relation : $PD_S \simeq \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)})$. Dans le cas d'absence d'interaction entre les variables x_j et x_k, on a la relation :

$$PD_{j,k}(x_j, x_k) = PD_j(x_j) + PD_j(x_j, x_k) \tag{4.33}$$

Si x_j n'a d'interaction avec aucune des autres variables, la prédiction du modèle d'une entrée x vérifie :

$$\hat{f}(x) = PD_j(x_j) + PD_{-j}(x_{-j}) \tag{4.34}$$

A présent, présentons les coefficients introduits par Friedman. Le premier noté $H_{j,k}$, mesure la quantité de variance expliquée par l'interaction entre x_j et x_k. La formule est alors la suivante :

$$H_{j,k}^2 = \frac{\sum_{i=1}^{n} \left[PD_{j,k}(x_j^{(i)}, x_k^{(i)}) - PD_j(x_j^{(i)}) - PD_k(x_k^{(i)}) \right]^2}{\sum_{i=1}^{n} PD_{j,k}^2(x_j^{(i)}, x_k^{(i)})} \tag{4.35}$$

Dans le cas de non interaction entre x_j et x_k, la H-statistique va valoir zéro, tandis que si toute la variance de $PD_{j,k}$ est expliquée par la somme des fonctions de dépendance individuelle alors elle vaudra 1.

Une autre statistique a été introduite par Friedman, pour mesurer l'effet d'une variable avec toutes les autres utilisées pour ajuster le modèle $\hat{f}$. Celle-ci est définie par :

$$H_j^2 = \frac{\sum_{i=1}^{n} \left[\hat{f}(x^{(i)}) - PD_j(x_j^{(i)}) - PD_{-j}(x_{-j}^{(i)}) \right]^2}{\sum_{i=1}^{n} \hat{f}^2(x^{(i)})} \tag{4.36}$$

L'un des problèmes que pose le calcul des H-statistique est le temps de calcul, qui s'exprime en $O(n^2)$ et devient vite impossible lorsque le nombre de données est important. On peut alors sous-échantillonner les n données disponibles, mais cela augmente la variance de l'estimation et rend la H-statistique instable.

Un test statistique est associé à la H-statistique précédemment définie avec l'hypothèse nulle d'absence d'interaction entre les variables.

4.2.3.3 Exemples sur la base de données "Boston"

Considérons la base de données "Boston" disponible dans le package *MASS* du logiciel R. La variable Y à expliquer est la valeur médiane des logements occupés par leurs propriétaires, en milliers de dollars. On dispose de 13 variables explicatives, dont la variable *crim* donnant le taux de criminalité par habitant par ville, et la variable. On ajuste un modèle de forêt aléatoire à partir de toutes les variables explicatives avec 50 arbres. Les résultats de la H-statistique pour l'interaction entre une variable et toutes les autres sont donnés par le graphique 4.26. Les résultats de la H-statistique entre la

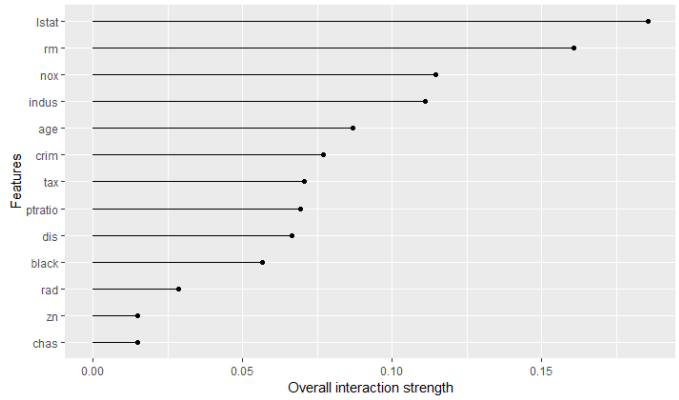

FIGURE *4.26: Interaction d'une variable avec toutes les autres (H-statistique) pour la base de données Boston ajustée avec un modèle de forêt aléatoire avec 50 arbres*

variable *crim* et toutes les autres sont présentés dans le graphique 4.27.

4.3 Modèles de substitution locaux

Cette partie s'intéresse aux méthodes locales d'interprétation des modèles de machine learning, à partir de modèles de substitution. Pour rappel, les modèles de substitution

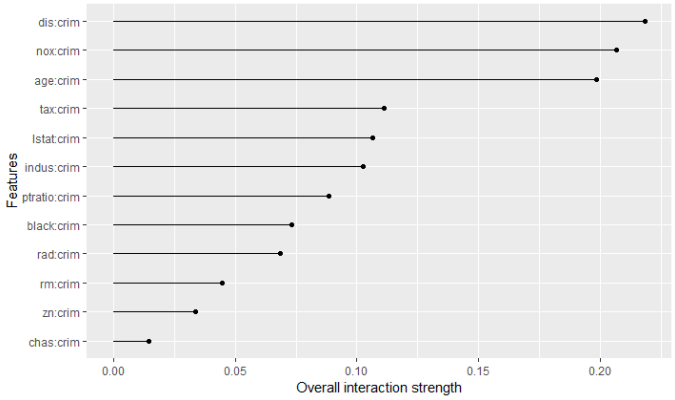

FIGURE *4.27: Interaction de la variable* crim *avec les autres variables (H-statistique) pour la base de données Boston ajustée avec un modèle de forêt aléatoire avec 50 arbres*

globaux, vus en annexe B.3, ont pour objectif d'approcher le comportement général de la boîte noire étudiée par un modèle simplifié comme un arbre de décision peu profond ou un modèle linéaire parcimonieux. L'approche locale réalise la même opération mais au niveau d'une observation donnée : on approche le comportement complexe de la boîte noire localement par un modèle simplifié.

4.3.1 LIME

Commençons par détailler la méthode LIME, qui est l'une des premières approches locales apparues dans le domaine du machine learning interprétable et qui est largement utilisée aujourd'hui.

4.3.1.1 Principe de LIME

Pour essayer d'expliquer les prédictions individuelles des modèles de machine learning, une autre méthode, appelée LIME, existe. Elle a été introduite dans l'article [61] *Why Should I trust You ?*. Cette méthode consiste également à utiliser un modèle de substitution (noté M_2) qui approche au mieux le modèle de machine learning (noté M_1) plus complexe, mais cette fois-ci de manière locale. En appliquant une légère perturbation au dataset initial X, on en crée un nouveau, noté $\tilde{X}$. Avec ce dernier, on construit les prédictions faites par le modèle $M_1 : \hat{y_1} = f_1(\tilde{X})$. On pondère le dataset $\tilde{X}$ en fonction de sa proximité avec le dataset initial : plus celui-ci est proche, plus son poids est important et vice-versa. A l'aide de ce dataset pondéré, on construit les prédictions faites par le modèle M_2. Celui-ci est généralement de type Lasso pour la régression et un arbre de décision pour la classification. Notons que cette fois-ci le modèle M_2 fournit une bonne approximation locale mais pas nécessairement une bonne approximation globale. Ainsi la fonction $\hat{g}$ associée au modèle M_2 est trouvée en résolvant ce problème d'optimisation :

$$\hat{g} = \underset{g \in G}{argmin}[J(f, g, \pi_x) + \Omega(g)] \tag{4.37}$$

avec J la fonction de coût, f la fonction associée au modèle M_1, g la fonction associée au modèle M_2 qu'on souhaite optimiser, appartenant à la classe de modèle G, π_x une mesure de proximité définissant la taille du voisinage autour de x que nous considérons pour l'interprétation du modèle et Ω une fonction traduisant la complexité d'un modèle. En pratique, LIME n'optimise que le terme associée à la fonction de coût. Il convient à l'utilisateur de choisir un modèle peu complexe, comme par exemple pour la régression, un modèle avec un nombre limité de variables explicatives (c.f critère de parcimonie vu dans la chapitre 2). Finalement, on peut résumer la méthode LIME par les étapes suivantes :

- pour chaque prédiction que l'on souhaite expliquer, on permute plusieurs fois les variables explicatives initiales
- pour chaque permutation, on utilise la prédiction faite par le modèle complexe M_2
- on calcule la distance entre toutes les permutations et l'observation initiale. Cette distance est alors convertie en un score de similarité
- on sélectionne m variables explicatives qui décrivent le mieux les prédictions faites par le modèle complexe sur les données permutées
- on ajuste un modèle simple M_1, qui explique la sortie du modèle complexe, à l'aide des m variables explicatives précédemment choisies, pondérées par leurs similarités avec l'observation d'origine
- on extrait les poids des variables explicatives du modèle M_1 et on les utilise comme explication pour le comportement local du modèle complexe M_2

La figure 4.28, introduite dans l'article [61], résume le fonctionnement de LIME dans le cas d'un modèle de classification binaire (classe 0 ou 1), avec deux variables explicatives. La zone en bleu représente les points prédits dans la classe 1 par le modèle étudié et en rose clair lorsque la classe prédite est 0. Les croix roses et les points bleus représentent les points simulés pour l'apprentissage du modèle de substitution. La taille du motif représente le poids du point considéré, suivant sa distance à l'observation d'intérêt, représenté par la croix rouge. La droite grise en pointillés est la limite de décision obtenue par l'algorithme LIME.

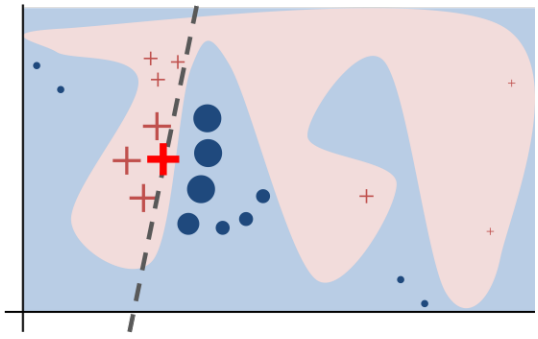

FIGURE *4.28: Principe de LIME*

Le choix du noyau dans l'algorithme LIME est primordial car a un impact majeur sur

la fidélité et la précision de l'explication qui en découle. Pour illustrer ceci, considérons un exemple avec une variable explicative X et un modèle de décision représenté par le trait noir sur la figure 4.29. Notre objectif est de comprendre localement la prédiction faite par le modèle au niveau de l'instance $x = 1.6$ (représentée par la croix noire). Les lignes tracées de différentes couleurs (jaune, vert et violet) correspondent aux différents modèles linéaires obtenus par la méthode LIME suivant le choix de la largeur σ du noyau. On observe que les lignes jaune (σ=2) et verte (σ=0.75) ne semblent pas bien approcher le comportement local de notre modèle, tandis que la courbe violette (σ=0.1) semble convenir.

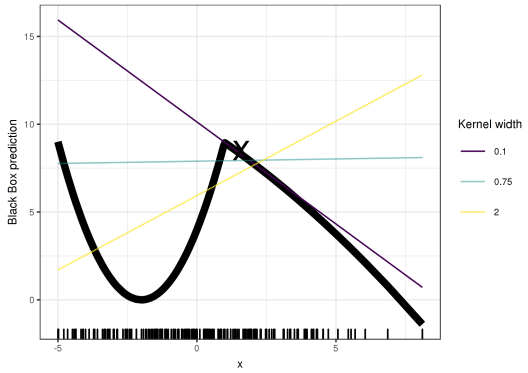

FIGURE *4.29: Choix du paramètre du noyau pour la mesure de proximité dans l'algorithme LIME*

4.3.1.2 Exemple sur les données réelles de Boston

Pour comprendre le résultat fourni par l'algorithme LIME, étudions le sur un exemple concret, à savoir les données Boston précédemment vues. Nous avons mis en place un modèle de forêt aléatoire, qui agit comme une boîte noire. Nous voulons comprendre la prédiction réalisée par ce modèle pour l'instance $x_{interest}$ suivante :

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	pratio	black	lstat
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98

TABLE *4.4: Observation de la base de données Boston dont on cherche à expliquer la prédiction* $\hat{medv} = 25.37863$ *par le modèle de boîte noire, alors que la valeur réelle est :* $medv = 24$

On utilise pour cela la fonction *LocalModel* de la librairie *iml* de R. L'algorithme utilisé est celui de LIME à quelques différences près :

— la mesure de distance utilisée pour calculer la proximité entre deux instances est par défaut la distance *gower* qui permet de ne pas avoir à réaliser de choix des paramètres du noyau. On peut cependant retrouver le cas exact de LIME, en choisissant une distance *dist* et un paramètre $\sigma > 0$, alors la mesure de proximité entre deux instances x et x' est définie par : $exp(-\frac{dist(x,x')^2}{2\sigma^2})$.

— Les données utilisées pour ajuster le modèle de substitution ne suivent pas une loi normale comme dans LIME, mais sont directement les données d'orgine, qui seront ensuite pondérées par leur distance à l'instance d'intérêt lors de l'ajustement du modèle de substitution.

Les résultats obtenus à l'aide de cet algorithme sont résumés dans le tableau 4.5. On a imposé un nombre maximum de *features* utilisés dans le modèle de substitution de 3. Le graphique associé des effets sur la prédiction de l'instance $x_{interest}$ est donné sur la figure 4.30.

Variable	β	$x_{interest}$	Effet
rm	4.4836483	6.575	29.479987
ptratio	-0.5244767	15.300	-8.024493
lstat	-0.4348698	4.980	-2.165652

TABLE *4.5: Paramètres du modèle de substitution ajusté et effets sur la prédiction*

Ainsi, le modèle de substitution utilisé pour expliquer l'instance $x_{interest}$ est le modèle de régression linéaire suivant : $s_x = \beta_0 + \beta_1\ rm + \beta_2\ pratio + \beta_3\ lstat$, avec $\beta = 0$, utilisant les trois variables *rm*, *pratio* et *lstat*.

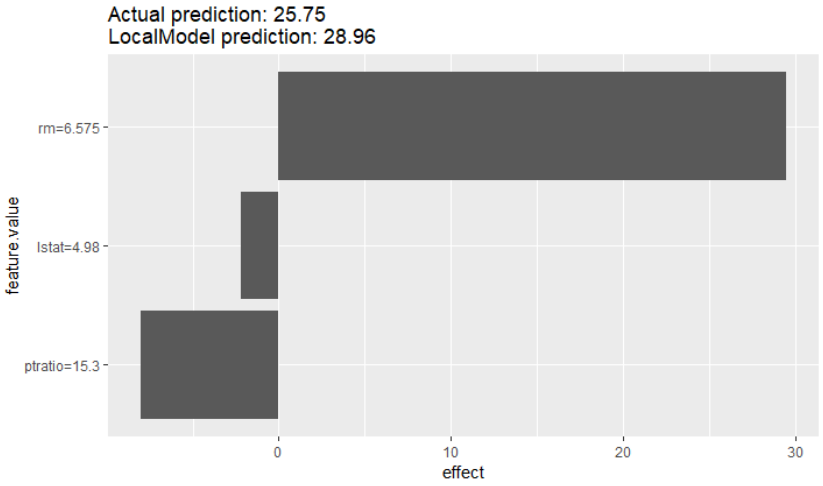

FIGURE *4.30: Résultats obtenus par une méthode proche de LIME, sur les données de Boston, avec une forêt aléatoire ajustée, sur la première instance*

4.3.1.3 Limites de LIME

L'article [68] *Why Should You Trust My Interpretation" Understanding Uncertainty in LIME Explanations* de Zang et Al. (2019) souligne les incertitudes liées aux méthodes d'interprétation des modèles de machine learning, dont LIME. Ceci engendre une inquié-

tude concernant leur robustesse et la confiance que l'on peut en avoir. Les trois sources majeures d'incertitudes détaillées dans l'article sont :

— *Le hasard dû à l'échantillonnage des données* : il s'agit de la variance d'échantillonnage pour expliquer un unique point de la base.
— *La sensibilité selon le choix des paramètres* : taille de l'échantillon et la proximité d'échantillonnage.
— *La variation de la crédibilité d'interprétation selon les points étudiés.*

Afin de mettre en exergue ces défauts liés à la méthode LIME, les auteurs de l'article ont utilisé deux jeux de données différents : le premier issu de simulation avec des données synthétiques générées à partir d'arbres, et le deuxième est le jeu de données réel COMPAS dont la variable cible est le risque de récidive d'un criminel. La classe des modèles de substitution utilisés pour expliquer les prédictions individuelles de la boîte-noire initiale est K-Lasso, c'est-à-dire une régression linéaire régularisée, de sorte à n'avoir qu'un nombre $K \in \mathbb{N}$ de coefficients, fixé auparavant.

Détaillons le principe de génération des simulations. Soit $N \in \mathbb{N}$, le nombre de variables explicatives choisies. Les bases d'apprentissage et de test sont générées à partir de modèles linéaires parcimonieux sur des entrées uniformément distribuées sur $[-1, 1]^N$. Afin d'expliquer le comportement local de LIME sur différents points, on les partitionne à l'aide d'un arbre de décision connu. Pour chaque partition, on assigne des sorties binaires (0 ou 1) sur chaque point X basées sur une classification linéaire avec un vecteur de coefficients β connu, comme indiqué sur l'équation 4.38 :

$$y(x) = \begin{cases} 1 \ si \ \ < x, \beta > \geq 0 \\ 0 \ si \ \ < x, \beta > < 0 \end{cases} \tag{4.38}$$

Prenons le cas particulier $N = 8$, on obtient par exemple l'arbre de décision de la figure 4.31, sur lequel les coefficients locaux sont indiqués sous les feuilles. Cet exemple sépare les données en six feuilles, avec seul trois des huit coefficients de β valent 1, et les cinq restants valent 0.

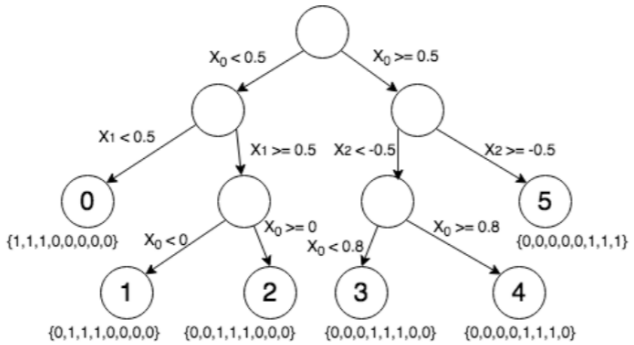

FIGURE *4.31: Exemple de partition de l'arbre de décision pour huit variables explicatives et coefficients locaux choisis selon la feuille (issu de l'article [68])*

Les données simulées à partir de ce processus sont ajustées à l'aide d'un Random Forest. L'idée est d'alors d'appliquer LIME sur un point de chaque feuille de l'arbre de

la figure 4.31, avec un $3-Lasso$. Les coefficients renvoyés par LIME devraient alors être ceux indiqués sous les feuilles, à savoir : $(1,1,1,0,0,0,0,0)$ pour le point de la feuille 0, $(0,1,1,1,0,0,0,0)$ pour le point de la feuille 1, ... et $(0,0,0,0,0,1,1,1)$ pour le point de la feuille 5. A partir de 1000 simulations différentes, on observe la probabilité estimée qu'un coefficient ait été choisi par LIME, selon la feuille dans laquelle se trouvait le point. La valeur de la largeur du noyau σ a été fixée dans un premier temps à 1, puis à 0.1. Les résultats obtenus pour $\sigma = 1$ sont représentés sur la figure 4.32.

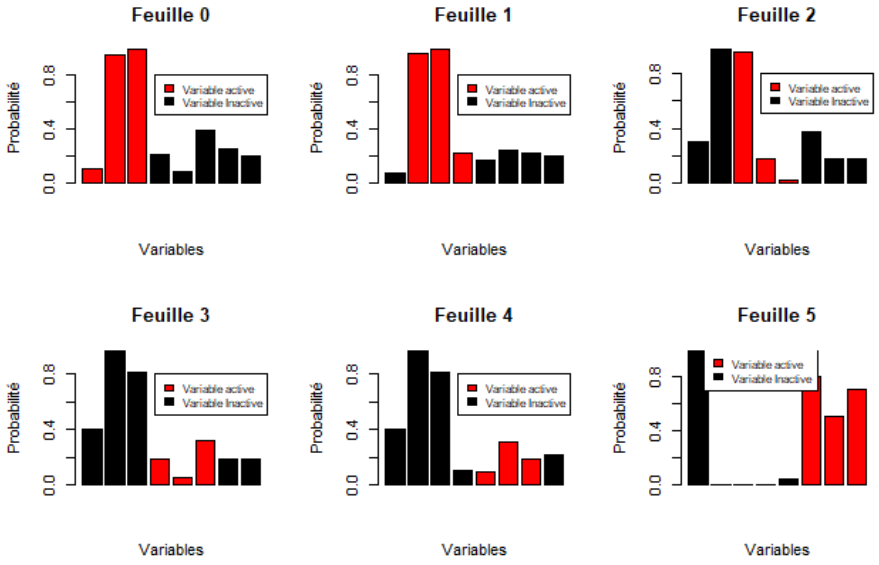

FIGURE *4.32: Probabilité qu'un coefficient ait été utilisé par le modèle de substitution 3-LASSO de l'interprétation fournie par LIME, dans le cas où* $\sigma = 1$

Dans le cas où $\sigma = 0.1$, on observe des résultats sensiblement différents, notamment pour les points de la feuille 5, comme indiqué sur la figure 4.33. En effet, on observe que pour $\sigma = 0.1$, LIME trouve pour chaque simulation les coefficients pertinents.

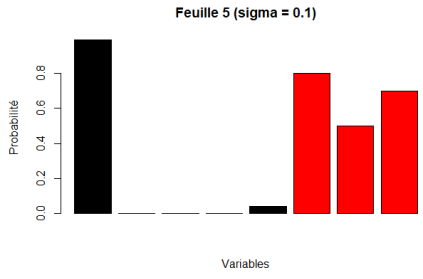

FIGURE *4.33: Probabilité qu'un coefficient ait été utilisé par le modèle de substitution 3-LASSO de l'interprétation fournie par LIME, pour la feuille 5, dans le cas où* $\sigma = 0.1$

Outre le choix de la largeur du noyau qui est source d'incertitude, on observe des résultats fournis par LIME différents selon la simulation considérée. Cela nous montre le problème de la variance d'échantillonnage. On observe également une instabilité de l'interprétation donnée par LIME : selon la feuille choisie, il n'arrive pas à refléter le comportement local du modèle. Une alternative à LIME est proposée en annexe B.2.

4.3.2 Limites de LIME et nouvelle méthode LS

Dans cette partie, nous reprenons les concepts introduits dans l'article [45] *Defining Locality for Surrogates in Post-hoc Interpretablity*, qui montre les limites de LIME et propose une nouvelle méthode.

4.3.2.1 Introduction

Les approches (comme LIME) par modèles de substitution locaux (*local surrogates*) font parties des méthodes pour comprendre les prédictions faites par un modèle dit boîte noire. Cet article souligne l'importance de définir la "bonne" localité, c'est-à-dire le voisinage sur lequel le modèle de substitution local va apprendre, afin d'approcher précisément les limites de décision locales de la boîte noire. Le problème souligné dans cet article, est qu'il ne s'agit pas uniquement d'un problème de paramètre ou de distribution de l'échantillon. Cela a donc un impact sur la pertinence et la qualité de l'approximation des limites de décision locales de la boîte noire et ainsi sur le sens et la précision des explications fournies par les méthodes comme LIME. Pour surmonter les problèmes identifiés, quantifiés avec une mesure et une procédure adaptées, l'article propose de générer des explications, toujours basées sur des modèles de substitution, avec des échantillons centrés sur un endroit particulier de la limite de décision (boundary limit), utile pour la prédiction à donner, plutôt que sur la prédiction elle-même comme c'est classiquement fait. Cette nouvelle approche est ensuite comparée aux méthodes habituelles, et nous verrrons qu'elles apportent une amélioration sur la qualité d'interprétation fournie.

4.3.2.2 Contexte

On considère un modèle de type boîte noire (SVM, Random Forest...) : $b : \mathbb{X} \to \mathbb{Y}$. On veut expliquer la prédiction $b(x)$ du modèle pour une instance $x \in \mathbb{X}$. Pour se faire, on met en place un modèle de substitution local s_x (surrogate local model) pour expliquer les limites de décision (decision boundary) de b. Pour se faire, il y a 3 étapes :

- Le choix d'une base d'apprentissage (X_{s_x}, Y_{s_x}) pour s_x, à partir de la base d'apprentissage initiale B_a.
- A partir de cet échantillon l'ajustement du modèle s_x peut-être mis en place dans le but d'expliquer le comportement local de b autour de l'instance x. On peut ajouter des contraintes pour contrôler la complexité du modèle de substitution ou sa localité.
- Utilisation du modèle s_x pour expliquer la prédiction $b(x)$ faite par le modèle boîte noire.

Il se pose alors la question du choix et de la manière dé générer la base d'apprentissage X_{s_x} de modèle de substitution. On pourrait utiliser directement la base d'apprentissage utilisée pour entraîner b, mais il a été montré par Craven et Shavlik en 1996 qu'augmenter localement la densité des instances est bénéfique pour la précision du modèle de substitution. Ainsi, il nous faut trouver une autre base d'apprentissage. Etant donné que l'on cherche un modèle local, il nous faut définir un voisinage. Analysons le choix fait par LIME et proposons ensuite une alternative.

4.3.2.3 LIME et le choix du voisinage d'un modèle de substitution local

La procédure de LIME pour expliquer la prédiction faite par b pour l'instance x repose sur les étapes détaillées dans l'algorithme :

— La base d'apprentissage X_{s_x} pour calibrer s_x est construite en simulant des échantillons d'une loi normale pour chaque variable explicative x_i, avec la même moyenne et le même écart-type que les variables utilisées pour ajuster b. On note pour tout $j \in \{1, ..., p\}$, $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ (moyenne empirique de la variable x_j dans la base d'apprentissage initiale) et $\sigma_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \mu_j)^2$ (variance empirique de la variable x_j dans la base d'apprentissage initiale). On note aussi n_{sim} le nombre d'individus utilisés pour ajuster s_x. Alors pour tout $j \in \{1, ..., p\}$, on simule n_{sim} échantillons indépendants de loi $N(\mu_j, \sigma_j^2)$. Ceci forme notre base X_{s_x}. Dans le cas de variables catégorielles, LIME travaille avec les probabilités de prédiction renvoyées par b.

— On ajuste le surrogate s_x en utilisant un modèle linéaire avec régularisation de type ridge, c'est-à-dire qu'on ajoute un paramètre de pénalité pour contrôler la variance du modèle. Chaque instance $\tilde{x}$ de X_{s_x} se voit attribuer un poids calculé par rapport à sa distance avec x, à l'aide d'une fonction noyau comme le noyau RBF défini par : $RBF(x, \tilde{x}) = exp(-\frac{||x-\tilde{x}||^2}{2\sigma^2})$, avec $\sigma > 0$ un paramètre à définir. Cette pondération a pour objectif de favoriser les points proches du point x que l'on cherche à expliquer, afin d'avoir une interprétation locale.

— On génère des explications de la boîte noire b en extrayant les coefficients de la régression linéaire mise en place avec s_x.

L'article montre alors que l'approche mise en place par LIME pour définir son voisinage a tendance à cacher les *features* avec une influence locale au profit des *features* avec une influence globale. Ceci a été mis en évidence sur le dataset "half moon" disponible dans le package "scikit-learn" de Python. C'est suite à cette découverte qu'une nouvelle approche, dans la continuité de LIME, a été introduite. Pour se faire, il nous faut un critère numérique pour évaluer la localité d'un point. On définit, à partir d'une mesure de précision Acc (telle que le score AUC), la fidélité locale $LocFid$ d'un modèle de substitution s_x pour la boîte noire b dans le voisiniage V_x autour de l'instance x que l'on souhaite définir : $LocFid(x, s_x) = \underset{x_i \in V_x}{Acc}\,(b(x_i), s_x(x_i))$. Il nous faut à présent définir le voisinage V_x choisi. L'article suggère que ce soit l'hypersphère l_2 centrée sur x de rayon

$r_{fid} > 0$. Ceci permet de considérer r_{fid} comme un proxy du degré de localité considéré. Étant donné que la valeur du rayon est étroitement liée à la dimension et à la densité de l'espace de départ $\mathbb{X}$, r_{fid} sera exprimé comme un pourcentage de la distance maximale entre les instances de la base de données initiales et l'instance x que l'on souhaite interpréter. L'étude a montré qu'avec LIME, la fidélité à l'échelle locale (r_{fid} petit) est moins bonne qu'à l'échelle globale (r_{fid} élevé), ce qui confirme ce qui a été dit précédemment. Afin de remédier à ce problème de fidélité à l'échelle locale, l'idée est d'ajuster le modèle de substitution sur un voisinage de la limite de décision à approcher, plutôt que sur un voisinage de l'instance x que l'on cherche à expliquer. En effet, LIME réalise l'ajustement du modèle de substitution sur toutes les données de la base d'apprentissage, en pondérant chaque instance par rapport à sa distance à l'instance x à expliquer.

4.3.2.4 Méthode LS (Local Surrogate)

La méthode LS (Local Surrogate) utilise la remarque faite précédemment. Son principe est très similaire à LIME mais diffère dans la première étape. En effet, la méthode LS calcule tout d'abord l'instance x_{bord}, la plus proche de l'instance à expliquer x telle que : $b(x) \neq b(x_{bord})$. Pour la calculer, la partie "Génération" de l'algorithme *GrowingSpheres* introduit par Laugel et al. (2018). Le principe est de générer des instances dans une hypersphère de rayons croissants, centrées sur x, jusqu'à atteindre la limite de décision de b. Une fois que x_{border} a été trouvé, les instances pour ajuster le modèle local sont tirées uniformément dans une hypersphère S de rayon r_{s_x}, centrée sur x_{bord} : $X_{s_x} \sim \mathbb{U}_{S(x_{bord}, r_{s_x})}$. On peut résumer la procédure LS par les étapes détaillées dans l'algorithme :

- Entrée : $x \in \mathbb{X}$, $b : \mathbb{X} \to \mathbb{Y}$, $r_{s_x} \in \mathbb{R}$, $N \in \mathbb{N}$
- Calculer de x_{bord} à l'aide de l'algorithme GrowingSpheres avec le modèle b et l'instance x
- Calcul de la base d'apprentissage X_{s_x} pour ajuster le modèle de substitution : tirage de N instances indépendantes dans l'hypersphère de rayon x_{bord} de rayon r_{s_x}.
- Calculer de la sortie de la boîte noire sur la base X_{s_x} : $Y_{s_x} = b(X_{s_x})$.
- Entraîner le modèle de substitution sur la base (X_{s_x}, Y_{s_x})
- Sortie : le modèle de substitution s_x

4.4 SHAP

4.4.1 Valeur de Shapley en théorie des jeux

Quand un modèle réalise une prédiction, intuitivement nous savons que chaque variable ne joue pas le même rôle et même que certaines n'ont quasiment aucun impact tandis que d'autres en ont un grand sur la décision prise par le modèle. L'objectif de Shap et de la valeur de Shapley est justement de quantifier le rôle de chaque variable dans la décision finale du modèle. Pour comprendre comment cette valeur fonctionne, rappelons le principe de la valeur de Shapley dans la théorie des jeux.

Considérons un jeu J de coalition (ou de coopération). Celui-ci est un 2-uplet : $J = (\{1, ..., p\}, v)$ où $P = (\{1, ..., p\}$ est un ensemble de p joueurs, $p \in \mathbb{N}^*$), et $v : S(P) \to \mathbb{R}$ est une fonction caractéristique telle que : $v(\emptyset) = 0$ (où $S(P)$ est l'ensemble des sous-ensembles de P) Un sous-ensemble de joueurs $S \in S(P)$ est appelé coalition et l'ensemble $\{1, ..., p\}$ de tous les joueurs est appelé la grande coalition. La fonction caractéristique v décrit l'importance de chaque coalition. L'objectif de notre jeu est de répartir l'importance de chaque joueur dans le gain total, de la manière la plus "juste" possible. Ainsi, on cherche un opérateur ϕ, qui assigne au jeu $J = (\{1, ..., p\}, v)$, un vecteur $\phi = (\phi_1, ..., \phi_p)$ de payoffs. Comment définir la notion de répartition juste entre les joueurs ? Elle a été introduite par Lloyd Shapley en 1953, à travers quatre axiomes :

— Efficacité : $\sum_{i=1}^{p} \phi_i(v) = v(\{1, ..., p\})$
— Symétrie : Pour tout couple de joueurs $(i, j) \in \{1, ..., p\}^2$, si $\forall S \in S(\{1, ..., p\} \backslash \{i, j\})$, $v(S \cup i) = v(S \cup j)$, alors $\phi_i(v) = \phi_j(v)$
— Facticité : Soit $i \in \{1, ..., p\}$ un joueur. Si $\forall S \in S(\{1, ..., p\} \backslash \{i\}), v(S \cup \{i\})$, alors : $\phi_i(v) = 0$.
— Additivité : Pour tout jeu, $v : S(P) \to \mathbb{R}$, $w : S(P) \to \mathbb{R}$, $\phi(v + w) = \phi(v) + \phi(w)$ avec : $\forall S \in S(\{1, ...p\}), (v + w)(S) = v(S) + w(S)$

La valeur de Shapley ϕ est alors l'unique valeur "juste" qui distribue le gain total $v(\{1, ..., p\})$, c'est-à-dire celle qui respecte les quatre conditions précédentes. Il s'agit d'un théorème démontré par Shapley, pour lequel nous avons également une formule explicite pour cette valeur de Shapley, à savoir :

$$\forall i \in \{1, ..., p\},\ \phi_i(v) = \sum_{S \in S(\{1,...,p\}) \backslash \{i\}} \frac{(p - |S| - 1)!|S|!}{p!} (v(S \cup \{i\}) - v(S)) \tag{4.39}$$

On peut également définir cette valeur de Shapley d'une autre manière :

$$\forall i \in \{1, ..., p\},\ \phi_i(v) = \frac{1}{p!} \sum_{O \in Perm(P)} v(Pre^i(O) \cup \{i\}) - v(Pre^i(O)) \tag{4.40}$$

avec : $Perm(P)$ l'ensemble des permutations de $P = \{1, ..., p\}$ et $Pre^i(O)$ l'ensemble des joueurs qui sont prédécesseurs du joueur i dans la permutation $O \in Perm(P)$ (il s'agit du nombre qui apparaît avant le nombre i dans la permutation O).

4.4.2 Valeur de Shapley appliquée à l'interprétabilité des modèles

L'idée derrière Shap est d'utiliser la valeur de Shapley pour expliquer les prédictions faites par un modèle complexe de machine learning. Pour se faire, nous devons relier l'interprétabilité qu'on souhaite réaliser avec la théorie des jeux. On se place toujours dans le cas où on veut prédire une variable numérique $Y \in \mathbb{R}$, à partir d'un vecteur $X \in \mathbb{R}^p$ de $p \in \mathbb{N}$ variables explicatives. On suppose que l'on dispose d'un échantillon : $y = (y_1, ..., y_n) \in \mathbb{R}^n$ correspondant aux valeurs cibles et $x = (x_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$ correspondant aux variables explicatives (avec $n \in \mathbb{N}$ le nombre d'individus). Notre modèle M de

machine learning apprend sur cet échantillon et on note $\hat{g}$ la fonction associée au modèle, c'est-à-dire la fonction qui renvoie la prédiction $\hat{y}$ de y faite par le modèle à partir de l'input x : $\hat{y} = g(x)$. Nous avons ici :

- le jeu : la tâche de prédiction pour une instance $\tilde{x} \in \mathbb{R}^p$ du dataset.
- le gain : la prédiction actuelle de cette instance moins la prédiction moyenne de toutes les instances du dataset
- les joueurs : les valeurs des caractéristiques $x_j,\ j \in \{1, ..., p\}$, qui collaborent pour recevoir le gain (ici il s'agit de prédire une certaine valeur)

Prenons un exemple concret pour mieux appréhender le jeu que l'on considère. Supposons que notre variable Y à expliquer est le prix d'une voiture en euros. Nos variables explicatives sont x_1 et x_2, respectivement le nombre de chevaux de la voiture et le nombre de portes. Supposons que pour $x_1 = 150$ et $x_2 = 4$, nous avons un prix estimé par le modèle $\hat{g}$ de : $y = 150000$. Nous savons également qu'à partir de notre dataset initial complet (constitué de plusieurs prix de voitures et des variables explicatives associées), nous avons une prédiction moyenne de 170000 euros. L'objectif de notre jeu est d'expliquer cette différence de -20000 euros, entre la prédiction faite par le modèle et la prédiction moyenne. On pourrait par exemple, obtenir le résultat suivant : x_1 a contribué pour $+10000$ euros et x_2 pour -30000 euros (par rapport à la valeur moyenne prédite) et justifierait donc la différence de -20000 euros observée. Finalement, on peut définir la valeur de Shapley comme la contribution marginale moyenne d'une variable (explicative) sur toutes les coalitions possibles.

4.4.2.1 Valeur de Shapley dans le cas de la régression linéaire

On considère le modèle linéaire : $\hat{g}(x) = \beta_0 + \sum_{i=1}^{p} \beta_i\ x_i$, avec $(\beta_i)_{0 \leq i \leq p} \in \mathbb{R}^p$. On définit alors la valeur de Shapley de la variable $j \in 1, ..., p$ associée à la prédiction $\hat{g}(x)$: $\phi_j(\hat{g}) = \beta_j x_j - \mathbb{E}[\beta_j Xj] = \beta_j(x_j - \mathbb{E}[Xj])$ (avec $\mathbb{E}[\beta_j Xj]$ l'effet moyen de la variable x_j). On parle aussi de contribution de la variable x_j dans la prédiction de $\hat{g}(x)$, car il s'agit de la différence entre l'effet de la variable et l'effet moyen. On peut remarquer que la somme des contributions de toutes les variables explicatives donnent la différence entre la valeur prédite pour x et la valeur de prédiction moyenne. En effet : $\sum_{j=1}^{p} \phi_j(\hat{f}) = \sum_{j=1}^{p} (\beta_j x_j - \mathbb{E}[\beta_j X_j]) = (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} \mathbb{E}[\beta_j X_j]) = \hat{g}(x) - \mathbb{E}[\hat{g}(X)]$ On peut à présent généraliser ceci pour tout modèle, à l'aide de la valeur de Shapley.

4.4.2.2 Valeur de Shapley dans le cas général

On se replace dans le cas d'un modèle boîte noire, avec $\hat{g}$ la fonction associée. on note $X = X_1$ x X_2 x ... x X_p l'espace des caractéristiques, représenté sur l'ensemble $\{1, ..., p\}$. Considérons $\tilde{x} = (\tilde{x_1}, ..., \tilde{x_p})$ l'instance pour laquelle on veut expliquer la prédiction. Définissons la différence en prédiction d'un sous-ensemble des valeurs des caractéristiques dans une instance particulière $\tilde{x}$, introduite par Strumbelj et Kononenko. Il s'agit du chan-

gement dans la prédiction causé par l'observation de ces valeurs des variables explicatives. Formellement, soit $S = \{i_1, ..., i_s\} \subset \{1, ..., p\}$ (avec $s \in \{1, ..., p\}$) un sous-ensemble des variables explicatives. Notons $\Delta^{\tilde{x}}$ cette différence de prédiction, associée au sous-ensemble S : $\Delta^{\tilde{x}}(S) = \mathbb{E}[\hat{g}(X_1, ..., X_p)|X_{i_1} = x_{i_1}, ..., X_{i_s} = x_{i_s}] - \mathbb{E}[\hat{g}(X_1, ..., X_p)]$ Cette différence de prédiction correspond à notre fonction de coût, qu'on note également $v_{\tilde{x}}$. On a bien la propriété $v_{\tilde{x}} = 0$ de vérifiée. Ainsi $(\{1, ..., p\}, \Delta^{\tilde{x}})$ forme un jeu de coalition tel qu'il est défini dans la partie précédente.

La contribution de la variable explicative $x_j,\ j \in \{1, ..., p\}$, est définie comme la valeur de Shapley de ce jeu de coopération $(\{1, ..., p\}, \Delta^{\tilde{x}})$:

$$\phi_j(\Delta^{\tilde{x}}) = \sum_{S \in S(\{x_1,...,x_p\} \backslash \{x_j\})} \frac{|S|!(p - |S - 1|)!}{p!}(\Delta^{\tilde{x}}(S \cup \{x_j\}) - \Delta^{\tilde{x}}(S)) \tag{4.41}$$

Dans cette formule : $S(\{x_1, ..., x_p\} \backslash \{x_j\})$ l'ensemble des permutations de cet ensemble. En utilisant la formule alternative équivalente, on a également :

$$\forall i \in \{1, ..., p\},\ \phi_i(\Delta^{\tilde{x}}) = \frac{1}{p!} \sum_{O \in Perm(P)} \Delta^{\tilde{x}}(Pre^i(O) \cup \{i\}) - \Delta^{\tilde{x}}(Pre^i(O)) \tag{4.42}$$

Avec : $Perm(P)$ l'ensemble des permutations de $P = \{1, ..., p\}$.

Prenons un exemple simple pour comprendre comment la valeur de Shapley fonctionne. Considérons un jeu avec trois joueurs $\{1, 2, 3\}$. On compte alors $2^3 = 8$ sous-ensembles possibles, à savoir : $\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}$ et $\{1, 2, 3\}$. En utilisant la formule de l'équation 4.42, on obtient :

$$\phi_1 = \frac{1}{3}(v(\{1, 2, 3\}) - \{2, 3\}) + \frac{1}{6}(v(\{1, 2\}) - \{2\}) + \frac{1}{6}(v(\{1, 3\}) - \{3\}) + \frac{1}{3}(v(\{1\}) - v(\emptyset))$$
$$\phi_2 = \frac{1}{3}(v(\{1, 2, 3\}) - \{1, 3\}) + \frac{1}{6}(v(\{1, 2\}) - \{1\}) + \frac{1}{6}(v(\{2, 3\}) - \{3\}) + \frac{1}{3}(v(\{2\}) - v(\emptyset))$$
$$\phi_3 = \frac{1}{3}(v(\{1, 2, 3\}) - \{1, 2\}) + \frac{1}{6}(v(\{1, 3\}) - \{1\}) + \frac{1}{6}(v(\{2, 3\}) - \{2\}) + \frac{1}{3}(v(\{3\}) - v(\emptyset))$$

En définissant le gain "non distribué" $\phi_0 = v(\emptyset)$, qui correspond au payoff fixé qui n'est pas associé aux actions des joueurs, la propriété d'additivité est bien respectée, à savoir : $\phi_0 + \phi_1 + \phi_2 + \phi_3 = v(\{1, 2, 3\})$.

Dans le cas général, on retrouve alors les propriétés vues précédemment à savoir :

— Efficacité : $\sum_{i=1}^{p} \phi_i(\Delta^{\tilde{x}}) = \Delta^{\tilde{x}}(\{1, ..., p\}) = \hat{g}(\tilde{x}) - \mathbb{E}[\hat{g}(X)]$. On retrouve alors la propriété que l'on a observée pour le modèle linéaire, à savoir que la somme des p contributions pour l'explication d'une observation est égale à la différence entre la prédiction faite par le modèle pour cette observation et la prédiction (moyenne) du modèle si on ne connaissait aucune information sur la valeur des variables explicatives $x_j,\ j \in \{1, ..., p\}$.

— Symétrie : deux variables explicatives qui ont une influence identique sur la prédiction auront des valeurs de contributions identiques

— Facticité : une variable qui a une contribution de 0 n'aura aucune influence sur la prédiction.
— Additivité : Si le modèle qu'on utilise repose sur la moyenne de plusieurs modèles (comme les forêts aléatoires qui utilisent des arbres de décision) alors la contribution de ce modèle sera la moyenne des contributions de chaque modèle pris seul.

4.4.2.3 Algorithme de calcul approché de la valeur de Shapley

Le problème en pratique est le temps d'exécution pour le calcul de cette valeur de Shapley, du fait de la complexité de calcul. En effet, pour se faire, nous devons calculer toutes les coalitions possibles avec ou sans la variable que l'on souhaite expliquer : il s'agit alors d'une complexité exponentielle. Pour remédier à ce problème, Strumbelj et Kononenko dans *Explaining prediction models and individual predictions with feature contributions* ont proposé une approximation utilisant le principe de Monte Carlo, à savoir : $\hat{\phi_j} = \frac{1}{M} \sum_{m=1}^{M} \hat{g}(x_{+j}^m) - \hat{g}(x_{-j}^m)$, où : $j \in \{1, ..., p\}$ le numéro de la variable qu'on souhaite expliquer, $M \in \mathbb{N}$ est le nombre d'itérations choisi, $\hat{g}(x_{+j}^m)$: la prédiction pour le vecteur $x = (x_1, ..., x_p)$ de p variables explicatives, mais avec un nombre aléatoire de caractéristiques remplacées par un point z aléatoire, excepté pour la valeur de la caractéristique j choisie. $\hat{g}(x_{-j}^m)$ est quasiment identique à $\hat{g}(x_{+j}^m)$ sauf que la valeur x_j^m est aussi prise à partir de l'échantillon de x.

On en déduit la procédure proposée par Strumbelj et Kononenko pour approcher la valeur de Shapley $\phi_j(\Delta^{\tilde{x}})$ associée à la variable x_j pour $j \in \{1, ..., p\}$ à l'aide de l'algorithme suivant :

— Entrée : modèle $\hat{g}$, l'instance $\tilde{x}$ qu'on cherche à expliquer, M le nombre d'itérations de l'algorithme
— $\phi_j = 0$
— pour i allant de 1 à M, faire :
  — choisir une permutation aléatoire $O \in Perm(P)$
  — choisir une instance $z = (z_1, ..., z_p)$ du dataset initial
  — $\forall k \in \{1, ..., p\}$, $x_k^+ = \begin{cases} \tilde{x}_k \text{ si } k \in Pre^i(O) \cup \{j\} \\ z_k \text{ sinon} \end{cases}$
  — $\forall k \in \{1, ..., p\}, x_k^- = \begin{cases} \tilde{x}_k \text{ si } k \in Pre^i(O) \\ z_k \text{ sinon} \end{cases}$
  — $\phi_j = \phi_j + (\hat{g}(x^+) - \hat{g}(x^-))$
— Output : $\hat{\phi_j}^{(M)} = \frac{\phi_M}{M}$

Notons bien qu'à chaque itération, les calculs des termes $\hat{g}(x^+)$ et $\hat{g}(x^-)$ reposent sur des observations qui sont identiques à l'exception de la variable $\tilde{x}_j$. Ils sont construits en prenant l'instance z et en changeant la valeur de chaque variable apparaissant avant la j-ième variable dans l'ordre de la permutation O (pour x^- la valeur de $\tilde{x}_j$ est également changée) par la valeur des caractéristiques de l'instance pour laquelle on désire expliquer y.

4.4.2.4 Propriétés de la méthode SHAP

Rappelons tout d'abord qu'il ne faut pas confondre Shap et LIME qui expliquent deux choses différentes : Shap explique la différence entre la prédiction et la prédiction moyenne globale, tandis que LIME explique la différence entre la prédiction et une prédiction moyenne locale.

Avantages

La méthode Shap a la seule méthode d'interprétabilité, à ce jour, avec un fondement mathématique solide. En effet, la différence entre la prédiction et la prédiction moyenne est distribuée de manière "juste" entre les différentes variables utilisées par le modèle, grâce à la propriété d'efficacité de la valeur de Shapley. Ceci n'est pas le cas de LIME, qui repose sur un principe qui semble cohérent mais n'a pas de justification mathématique du résultat fourni. Dans le cadre du RGPD et du "droit à l'explication", Shap pourrait être la seule méthode d'interprétabilité des modèles répondant à ces exigences.

La méthode Shap fournit une explication de la prédiction faite par la boîte noire en attribuant une valeur de contribution à chaque variable utilisée, contrairement à LIME qui renvoie une réponse plus concise, en pénalisant les modèles complexes. On peut alors considérer que Shap réalise moins d'approximations que LIME et de ce fait fournit une explication plus précise. Le fait d'utiliser toutes les variables dans l'explication peut également être vu comme un inconvénient comme nous le verrons dans le paragraphe suivant.

Inconvénients

Lorsque la boîte noire est entraînée avec un grand nombre de variables, l'interprétation renvoyée par Shap ne respecte pas le principe de parcimonie. Celle-ci devient alors difficilement lisible et interprétable. Pour contourner ce problème, une adaptation de Shap, appelée Kernel Shap (Linear LIME + Shapley Values), a été proposée dans l'article [48] *A Unified Approach to Interpreting Model Prediction* de Lundberg et Al. (2017). L'idée est de relier les équations 4.37 (de LIME) et 4.41 (de Shap). Bien que celles-ci aient l'air très différentes à première vue, en effectuant le bon choix de la fonction de coût J, de la mesure de proximité $\Pi_{x'}$ et du terme de régularisation Ω, il est possible d'écrire la valeur de Shapley comme solution du problème d'optimisation posé par LIME dans l'équation 4.37. Une fois la valeur de Shapley écrite comme solution d'un problème d'optimisation, il est possible de fournir des explications parcimonieuses, avec beaucoup de coefficients nuls.

Outre le temps de calcul de la valeur de Shapley théorique et de l'obligation d'avoir recours à des méthodes d'approximations, un autre problème se pose avec Shap et les méthodes classiques utilisées pour l'estimer : la prise en compte de la dépendance entre les variables. Par exemple, la méthode Kernel Shap, qui est la plus utilisée en pratique, repose sur l'hypothèse d'indépendance entre les variables. Dans la majorité des problèmes de machine learning, il est rare d'avoir des *features* statistiquement indépendants. Si une forte dépendance est présente au sein des données, les explications fournies par ces méthodes peuvent être totalement erronées, même si un modèle linéaire simple est utilisé.

En reprenant l'exemple cité tout au long de ce mémoire, avec des variables explicatives de poids et de taille d'un individu, on peut se retrouver dans les coalitions utilisées par Shap avec des individus de 5kg et de taille 2m, qui sont des instances irréalistes. L'article [1] *Explaining Individual Predictions When* features *Are Dependant : More Accurate Approximations To Shapley Values* de Aas. et Al (2019) propose d'incorporer la dépendance dans la méthode Kernel Shap. L'idée est de supposer que les variables explicatives suivent une distribution gaussienne multivariée et d'utiliser la théorie des copules pour y inclure la dépendance entre celles-ci.

Un dernier inconvénient qui peut être cité concernant Shap est que l'interprétation renvoyée est seulement un coefficient par variable et non un modèle complet comme LIME. Comme le souligne le site [51], on ne peut pas statuer sur le changement dans la prédiction si on change les entrées, par exemple : "Si je gagnais 300 euros de plus par an, mon score de crédit serait 5 points supérieurs".

4.4.2.5 Exemples sur les données Boston

Reprenons notre exemple de forêt aléatoire sur les données de Boston. On souhaite expliquer la prédiction réalisée par le modèle sur la première instance. En appliquant l'algorithme Shap, les résultats que l'on obtient sont résumés dans le tableau 4.6.

Feature	Phi	Phi.var	Valeur du *feature*
crim	-0.35591233	1.0353958657	crim=0.00632
zn	-0.03449733	0.0219707902	zn=18
indus	0.90981567	0.5831363873	indus=2.31
chas	-0.00475400	0.0008187619	chas=0
nox	-0.29788283	0.6333483584	nox=0.538
rm	-0.17674181	9.9415235348	rm=6.575
age	-0.23921066	4.973854e-01	age=65.2
dis	0.12372495	2.565502e-01	dis=4.09
rad	-0.31490883	5.935128e-02	rad=1
tax	-0.28033734	8.296632e-01	tax=296
ptratio	0.72933779	5.544890e-01	ptratio=15.3
black	0.07849862	1.625523e-01	black=396.9
lstat	3.05900193	1.456410e+01	lstat=4.98

TABLE *4.6: Résultats fournis par l'algorithme Shap sur la première instance des données de Boston, ajustées avec une Random Forest*

Graphiquement, la contribution de chaque variable à la prédiction est donnée par la figure 4.34. La somme de toutes les contributions, donne l'écart entre la prédiction réalisée et la moyenne des prédictions du modèle, c'est-à-dire ici : 25.75-22.56=3.19.

4.4.3 BreakDown : une variante de Shap

Nous présenterons dans cette partie une alternative à la méthode Shap, qui utilise une approche similaire. Celle-ci est présentée dans l'article [10] *Explanations of model predictions with live and breakDown packages* et est appelée BreakDown. Alors que la

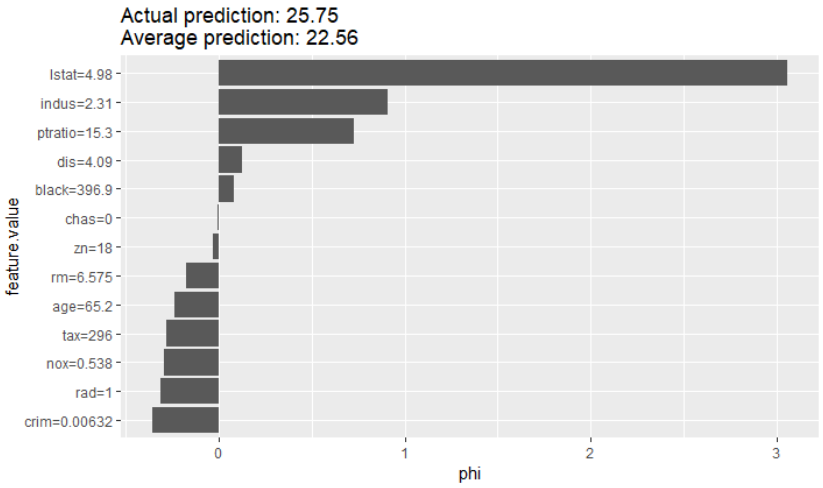

FIGURE *4.34: Résultats obtenus à l'aide de la méthode Shap pour expliquer la prédiction de la première instance des données Boston avec un modèle de Random Forest*

première méthode proposée par cet article (Live) reposait sur l'ajustement d'un modèle linéaire local autour d'une instance donnée, BreakDown décompose la prédiction du modèle en plusieurs parties, en attribuant des contributions aux variables utilisées par le modèle. Cette approche est triviale lorsque l'on s'intéresse à un modèle linéaire, l'article propose une méthode indépendante du modèle. Pour se faire, considérons les notations suivantes : $x = (x_1, ..., x_p) \in \mathbb{X} \subset \mathbb{R}^p$ un vecteur de l'espace des variables explicatives $\mathbb{X}$, $f : \mathbb{X} \to \mathbb{R}$ la fonction de prédiction associée au modèle étudié, et $X^{train} = (x^{(i)})_{1 \leq i \leq n}$ la base d'apprentissage utilisée, constituée de $n \in \mathbb{N}$ observations. Ainsi, pour une observation donnée x^{new}, $f(x^{new})$ correspond à la prédiction faite par le modèle f. Comme indiqué précédemment, l'objectif de la méthode BreakDown est d'affecter une part de la valeur $f(x^{new})$ aux variables explicatives, suivant leur contribution dans la prédiction.

4.4.3.1 Cas du modèle linéaire

Rappelons tout d'abord comment est calculé la contribution d'une variable lorsque l'on étudie un modèle de régression linéaire. Celui est défini par $f(x^{new}) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$, où $(\beta_i)_{0 \leq i \leq p}$ sont des réels. Il est alors facile de déterminer l'impact d'une variable x_i sur la prédiction $f(x^{new})$: il s'agit du terme $x_i^{new}\beta_i$. Pour avoir un terme plus facilement interprétable, il est intéressant qu'il vérifie des propriétés d'invariance par translation (lorsqu'il y a changement d'unité ou d'origine par exemple). Pour cela, on définit la contribution de la variables x_i par : $contrib(x_i) = (x_i^{new} - \bar{x_i})\beta_i$, avec $\bar{x_i} = \frac{1}{n}\sum_{k=1}^{n} x_i^{(k)}$ Ainsi, la prédiction faite par le modèle linéaire peut être décomposée ainsi : $f(x^{new}) = baseline + \sum_{i=1}^{p} contrib(x_i)$, avec le terme $baseline$, un terme constant défini

par : $baseline = \beta_0 + \sum_{i=1}^{p} \beta_i \bar{x_i}$.

4.4.3.2 Cas général

L'idée derrière la méthode BreakDown est de généraliser cette notion de contribution de variables pour des modèles non-additifs. Ainsi, l'idée est de décomposer la prédiction (locale) d'un modèle non-additif par une structure additive, ce qui va nécessairement engendrer une perte d'information sur la structure générale du modèle. Définissons tout d'abord, la notion de prédiction relâchée d'un modèle, pour un sous-ensemble $IndSet$ d'indices de $\{1, ..., p\}$, comme l'espérance de prédiction, lorsque l'on fige les valeurs des variables associées à cet ensemble, à savoir :

$$f^{IndSet}(x^{new}) = \mathbb{E}[f(x)|x_{IndSet} = x^{new}_{IndSet}] \tag{4.43}$$

Le terme "relâché" pour une variable correspond au fait que l'on ne fixe pas sa valeur, mais qu'elle suit la distribution de la population. Comme la distribution jointe réelle n'est pas connue, on l'estime empiriquement, c'est-à-dire on approche $f^{IndSet}(x^{new})$ par :

$$\hat{f^{IndSet}}(x^{new}) = \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)}_{\overline{IndSet}}, x^{new}_{Indset}) \tag{4.44}$$

On remarque deux cas extrêmes dans la définition précédente :
— Si $IndSet = \{1, ..., p\}$: $f^{\{1,...,p\}}(x^{new}) = f(x^{new})$
— Si $IndSet = \emptyset$: $f^{\emptyset}(x^{new}) = \mathbb{E}[f(x)]$.
Définissons à présent, la distance entre la prédiction d'un modèle et la prédiction relâchée associée à un sous-ensemble $IndSet$, par la formule :

$$d(x^{new}, IndSet) = |f^{IndSet}(x^{new}) - f(x^{new})| \tag{4.45}$$

Plus cette distance est petite, moins les variables de l'ensemble $IndSet$ sont importantes. Enfin, définissons la contribution d'ajout d'une variable x_j à l'ensemble $IndSet$, par la formule :

$$contrib^{IndSet}(j) = f^{IndSet \cup j}(x^{new}) - f^{Indset}(x^{new}) \tag{4.46}$$

L'approche BreakDown, aussi appelée ag-Break, repose sur le calcul de distances sur les prédictions relâchées du modèle. L'idée est de rechercher une série de variables pouvant être relâchées de sorte à faire passer la prédiction du modèle de $f(x^{new})$ (prédiction originale) à $\mathbb{E}[f(x)]$ (prédiction totalement relâchée).

Deux approches sont alors possibles : la première en considérant toutes les variables, et à retirer à chaque itération la variable que l'on peut relâcher en minimisant la perte (step-down), l'autre en partant avec une liste de variables vide à laquelle on ajoute, à chaque itération, la variable qui maximise la perte (step-up).

L'algorithme suivant résume les opérations à réaliser lors de l'approche step-down de BreakDown :
— $p \leftarrow$ nombre de variables

— $IndSet \leftarrow \{1, ..., p\}$: initialisation avec toutes les variables
— pour $i = 1, ..., n$:
  — $j_{min} \leftarrow \underset{j \in \{1,...,p\}}{argmin}\ d(x^{new}, IndSet \backslash \{j\})$: on trouve la variable que l'on peut relâcher avec la plus petite perte.
  — $contrib^{IndSet}(i) \leftarrow f^{IndSet}(x^{new}) - f^{IndSet \backslash \{j_{min}\}}(x^{new})$
  — $variables(i) \leftarrow j_{min}$
  — $IndSet \leftarrow IndSet \backslash \{j_{min}\}$
— Output : le vecteur de contributions $contrib^{IndSet}$ associé au vecteur de variables $variables$, classées par importance croissante

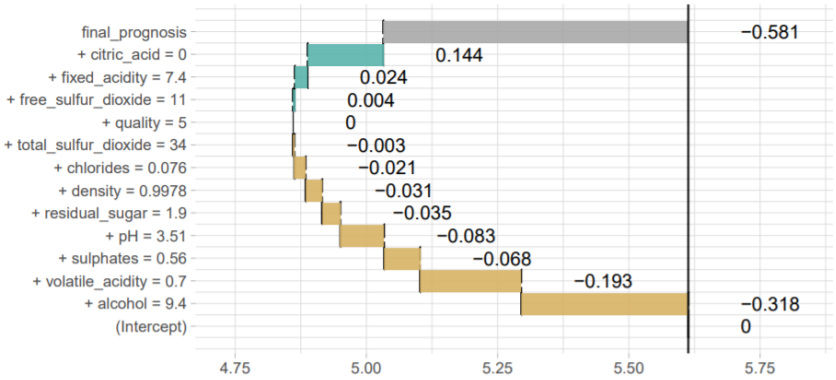

FIGURE *4.35: Résultat de l'approche step-down de BreakDown, pour expliquer la prédiction d'un modèle SVM radial sur la 5 i-ème observation de la base de données Wine*

L'approche step-up est détaillée dans l'algorithme suivant :

— $p \leftarrow$ nombre de variables
— $IndSet \leftarrow$: initialisation avec aucune variable
— pour $i = 1, ..., n$:
  — $j_{max} \leftarrow \underset{j \in \{1,...,p\}}{argmax}\ d(x^{new}, IndSet \cup \{j\})$: on trouve la variable à ajouter qui maximise la perte
  — $contrib^{IndSet}(i) \leftarrow f^{IndSet \cup \{j_{max}\}}(x^{new}) - f^{IndSet}(x^{new})$
  — $variables(i) \leftarrow j_{max}$
  — $IndSet \leftarrow IndSet \cup \{j_{max}\}$
— Output : le vecteur de contributions $contrib^{IndSet}$ associé au vecteur de variables $variables$, classées par importance décroissante

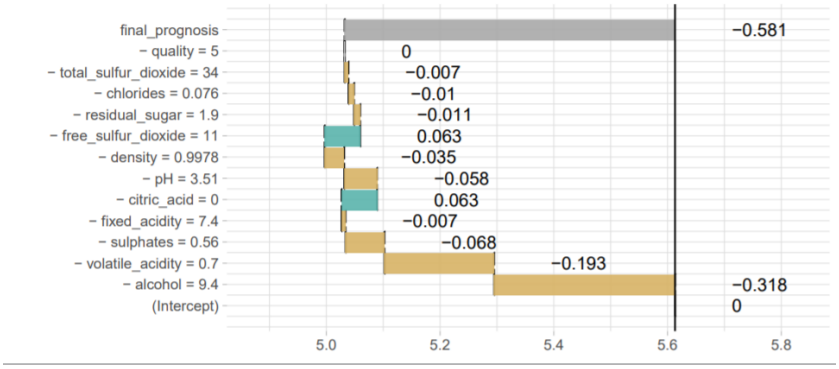

FIGURE *4.36: Résultat de l'approche step-up de BreakDown, pour expliquer la prédiction d'un modèle SVM radial sur la 5 i-ème observation de la base de données Wine*

Reprenons l'exemple de la base de données "Wine", étudiée dans la partie sur la méthode Live. Le modèle de boîte noire mis en place sur ces données est toujours un SVM avec un noyau radial. Une fois l'ajustement réalisé, on s'intéresse à la prédiction faite par ce modèle sur la 5 i-ème observation, définie par les variables explicatives de la figure B.2. La qualité réelle du vin est 5, tandis que la prédiction faite par le modèle est 5,03, ce qui est inférieur de 0,581 point par rapport à la prédiction moyenne du modèle. En appliquant la méthode BreakDown, par les deux approches vues précédemment, on obtient les résultats donnés par les figures 4.35 et 4.36.

4.5 Cartographie des différentes méthodes d'interprétabilité agnostiques au modèle

Cette section a pour objectif de résumer les différentes méthodes d'interprétation des modèles de machine learning que j'ai pu étudier au cours de la rédaction de ce mémoire, à l'aide d'un tableau récapitulatif. Dans celui-ci, j'ai indiqué la catégorie sur la colonne tout à gauche (comme l'importance des variables ou les méthodes basées sur l'effet des variables), l'approche locale ou globale de la méthode, les packages disponibles sous *R* et enfin la référence associée, qui est généralement l'article original.

	Méthodes	Local ou Global ?	Packages R	Référence(s)
Explications basées sur l'effet des variables	PDP	G	pdp, iml,DALEX	[26]
	ICE c-ICE	L	ICEbox, iml, condvis, pdp	[31]
	ALE	G	ALEplot, iml, DALEX	[5]
	VINE	L, G	ICEbox	[15]
	MPP	G	factorMerger, DALEX	[9]
Importance des Variables	PFI	G	iml, DALEX, PIMP, vip	[34]
	IPD	G	vip	[34]
	PI, ICI, SFIMP	L	featureImportance	[17]
Interaction entre les variables	H-statistics	G	iml, pre, gbm, vip	[27]
	VIN	G	/	[37]
	d-ICE	L	ICEbox	[31]
Global Surrogate	Simple Global Surrogate	G	/	[51]
	Model Extraction	G	/	[7]
Local Surrogate	LIME	L	LIME, iml	[61]
	LIVE	L	live	[10]
	LocalModel	L	LocalModel	[10]
	LS	L	/	[45]
Contribution des variables	SHAP	L	shapper, ShapleyR, iml	[48]
	BreakDown	L	breakDown, DALEX	[10]
Explications basées sur les exemples	Explications contrefactuelles	L	/	[73], [44], [62]
	Exemples contradictoires	L	/	[67], [32]
	Prototypes et Critiques	L, G	/	[39]
	Instances influentes	L	/	[41]

TABLE *4.7: Résumé (non exhaustif) des différentes méthodes d'interprétabilité post-hoc indépendantes du modèle*

Chapitre 5

Interprétabilité d'un modèle boîte noire utilisé pour la tarification automobile et comparaison avec un GLM classique

Dans cette partie, nous nous placerons dans un contexte de tarification automobile. L'objectif est de calculer la prime pure à partir des données disponibles. Pour cela, nous allons considérer deux modèles : un modèle GLM coût-fréquence et un modèle complexe (tel que SVM ou XGBoost). Le modèle GLM est le plus classiquement utilisé, de par sa simplicité et son interprétabilité comme nous avons pu le voir dans les parties précédentes. Nous voulons mettre en place un modèle complexe, plus performant que le GLM classique, et utiliser les outils vus dans le chapitre précédent afin d'interpréter les décisions prises par cette boîte noire. L'idée est d'alors de montrer que n'importe quel modèle, bien que très complexe, peut s'expliquer de manière similaire à un GLM. De plus, nous voulons souligner qu'un GLM supposé interprétable par la majorité est finalement difficilement lisible dans le cadre de la tarification automobile ; en effet, les nombreux coefficients (parfois plus de 100) ne permettent pas une vision claire de l'algorithme et l'idée de rejeter les modèles de machine learning, comme les forêts aléatoires, à cause de leur complexité semble surprenante étant donné que l'on accepte les GLM.

5.1 Généralités sur la tarification automobile et présentation des données

Dans ce chapitre, notre objectif est la tarification d'un contrat d'assurance automobile. Afin de la mettre en place, il est nécessaire d'introduire la notion de prime pure (PP). Si on note S la variable aléatoire correspondant à la charge totale de l'assureur, alors $\mathbb{E}[S]$ est la prime pure. Le modèle généralement mis en place pour modéliser S en assurance non vie (automobile, assurance habitation, etc.) est une approche fréquence/sévérité, éga-

lement appelée fréquence/coût moyen. On définit une variable aléatoire N à valeur dans $\mathbb{N}$, correspondant au nombre de sinistres survenus sur une période (par exemple un an) et des variables aléatoires iid $(W_i)_{1\leq i\leq N}$ donnant la sévérité du sinistre, c'est-à-dire le montant à la charge de l'assureur. L'hypothèse classique qui est réalisée est l'indépendance entre N et W_i pour tout i. On obtient alors le modèle :

$$S = \sum_{i=1}^{N} W_i \tag{5.1}$$

5.1.1 Tarification automobile

On peut alors facilement montrer, sous les hypothèses d'existence d'espérance et de variance des lois considérées, les formules :

$$\mathbb{E}[S] = \mathbb{E}[N]\mathbb{E}[W_1] \; et \; \mathbb{V}[S] = \mathbb{E}[N]\mathbb{V}[W_1] + \mathbb{E}[W_1]^2\mathbb{V}[N]$$

.

La formule de l'espérance donnée ci-dessus est l'élément de base du modèle coût-sévérité. En effet, la perte moyenne attendue (prime pure) est le produit de la fréquence moyenne $\mathbb{E}[N]$ et du coût moyen $\mathbb{E}[W_1]$: deux études indépendantes seront alors réalisées, une sur la fréquence et une autre sur la sévérité des sinistres. Afin de connaître la loi de S, à partir des lois de N et de W_1 plusieurs méthodes existent. La plus couramment utilisée est la formule de Panjer. Celle-ci néanmoins nécessite une hypothèse forte sur la loi de N, qui doit vérifier la propriété : $\forall k \in \mathbb{N}^*, P[N = k] = p_k = \left(a + \frac{b}{k}\right) p_{k-1}$ avec $(a, b) \in \mathbb{R}^2$. Néanmoins, on peut montrer que les deux lois usuelles pour la modélisation de la fréquence vérifient cette propriété, à savoir la loi de Poisson et la loi Binomiale Négative. Un théorème nous indique également que les deux lois citées précédemment ainsi que la loi Binomiale sont les seules lois vérifiant les hypothèses du théorème de Panjer.

Nous venons de présenter la notion de prime pure en tarification qui correspondent au montant moyen des sinistres attendus par l'assureur. Cependant, celle-ci n'est pas la prime réellement payée par l'assuré, il s'agit de la prime commerciale. Pour calculer cette dernière, on utilise la prime pure en lui ajoutant un terme de chargement et un terme de sécurité.

Les chargements (de gestion et d'acquisition) permettent de financer les coûts d'acquisition et d'administration de l'assureur. Le chargement de sécurité, généralement proportionnel à la variance des sinistres, est utilisé afin de se prémunir contre une mauvaise tarification du contrat d'assurance.

Nous nous plaçons dans le contexte de l'estimation de la prime pure, sans tenir compte des chargements ajoutés par la suite.

5.1.2 Jeu de données

Nous utilisons les jeux de données *freMTPLfreq* et *freMTPLsev* disponibles sous *R* dans le package *CASdatasets*. créé par C. Dutang et A. Charpentier. Les deux jeux de

données correspondent aux tables de fréquence et de sévérité, associées à la responsabilité civile moteur (observées sur une année) et des variables explicatives de risque sont collectées pour 413 169 polices.

Les variables à notre disposition, dans la base de données de fréquence sont :

— *PolicyID* : le numéro de police (pas explicative, sert à relier les deux bases de données)
— *ClaimNb* : le nombre de sinistres (ici entre 0 et 4)
— *Exposure* : l'exposition (en fraction d'années)
— *Power* : la puissance de la voiture (variable catégorielle ordonnée de "d" à "o")
— *CarAge* : l'âge du véhicule (en années)
— *DriverAge* : l'âge du conducteur (en années)
— *Brand* : la marque de la voiture, divisée en plusieurs groupes :
  • A - Renaut Nissan et Citroën,
  • B - Volkswagen, Audi, Skoda and Seat
  • C - Opel, General Motors et Ford
  • D - Fiat
  • E - Mercedes Chrysler et BMW
  • F - Japonaises (sauf Nissan) and Coréennes
  • G - Autres
— *Gas* : Gaz de la voiture : "Diesel" ou "Regular"
— *Region* : la région (en France) des polices, selon les standards français de classification. On dispose des régions suivantes :
  • *R11* : Ile-deFrance
  • *R23* : Haute-Normandie
  • *R24* : Centre-Val de Loire
  • *R25* : Basse-Normandie
  • *R31* : Nord-pas-de-Calais
  • *R52* : Pays de la Loire
  • *R53* : Bretagne
  • *R54* : Poitou-Charentes
  • *R72* : Aquitaine
  • *R74* : Limousin
— *Density* : nombre d'habitants par km^2 dans la ville où conduit le conducteur

Les variables dans la base de données de sévérité sont :

— *PolicyID* : le numéro de police
— *ClaimAmount* : le montant du sinistre

Une fusion des deux bases peut être réalisée en utilisant la clé d'identification *PolicyID*.

5.1.3 Analyse préliminaire

5.1.3.1 Données aberrantes

Avant de mettre en place la modélisation de la prime, il est nécessaire de bien connaître la base et d'effectuer des retraitements des données si cela est nécessaire. Une première

étape consiste à repérer les éventuelles données aberrantes ou manquantes. Notons par exemple que les données manquantes ne doivent pas être négligées étant donné qu'elles peuvent contenir des informations importantes sur l'assuré, bien qu'elles soient incomplètes. Dans le cas de nos données étudiées, aucune valeur manquante ou aberrante n'est à signaler.

Pour l'étude des valeurs aberrantes, on a par exemple étudié le nombre de sinistres pour une même police : on observe qu'aucun assuré n'a eu plus de 4 sinistres ce qui semble être une valeur raisonnable.

5.1.3.2 Sinistres extrêmes

Après avoir analysé les données manquantes ou aberrantes, il est courant en tarification automobile d'analyser les sinistres graves. Par exemple, les sinistres corporels en responsabilité civile peuvent représenter des sommes importantes (queues lourdes). Ces derniers sont à faible fréquence mais de forte sévérité et peuvent alors perturber l'estimation que l'on souhaite réaliser. C'est pour cela qu'il est courant de traiter ces sinistres dits extrêmes à part.

Le problème majeur de ces sinistres est leur faible nombre ce qui rend difficile l'estimation de la queue de distribution du coût moyen des sinistres. Illustrons ceci sur nos données de sinistres. Le tableau 5.1 résume les informations concernant la sévérité des 16181 sinistres de notre base de données. On note en particulier que 99 % des sinistres

min	max	moyenne	quantile 25 %	médiane	quantile 95 %	quantile 99 %
2	2036833	1986	590	1143	4110	15090

TABLE *5.1: Informations sur les montants de sinistres de la base de données étudiées*

ont un montant inférieur à 15 000 euros, alors qu'on dispose d'un seul sinistre de plus de 2 millions d'euros. Nous avons représenté sur la figure 5.1 les différents montants de sinistres (strictement positifs).

L'outil qui nous permet de résoudre ce problème des sinistres rares est la théorie des valeurs extrêmes. Plaçons-nous dans le contexte suivant : on dispose d'un échantillon i.i.d. de variables aléatoires $X_1, ..., X_n$ réelles positives, de fonction de répartition F (et de fonction de survie $\bar{F}$). Le problème qui se pose est l'estimation de la queue de distribution de la loi F. Comme indiqué précédemment, on dispose de peu de points dans cette queue de distribution et il n'est pas possible d'utiliser les méthodes classiques d'estimation de quantiles. Définissons tout d'abord la distribution GPD (*Generalized Pareto Distribution*). Une variable aléatoire X suit une loi GPD et on note $X \sim GPD(\sigma, k)$ avec $\sigma \geq 0$ et $k \in \mathbb{R}$, si sa fonction de répartition vérifie :

$$\mathbb{P}[X \leq x] = \begin{cases} G(x;\sigma,k) = 1 - \left[1 + \frac{kx}{\sigma}\right]^{-1/k} & si\ k \neq 0,\quad x \geq 0\ \ et\ \ 1 + \frac{kx}{\sigma} \geq 0 \\ G(x;\sigma,0) = 1 - exp(-\frac{x}{\sigma})\ si\ \ k = 0\ \ et\ \ x \geq 0 \end{cases} \tag{5.2}$$

On remarque que dans le cas particulier $k = 0$, on retrouve une loi exponentielle. Définissons également la loi d'excès au-dessus d'un seuil $u > 0$ par la fonction de survie

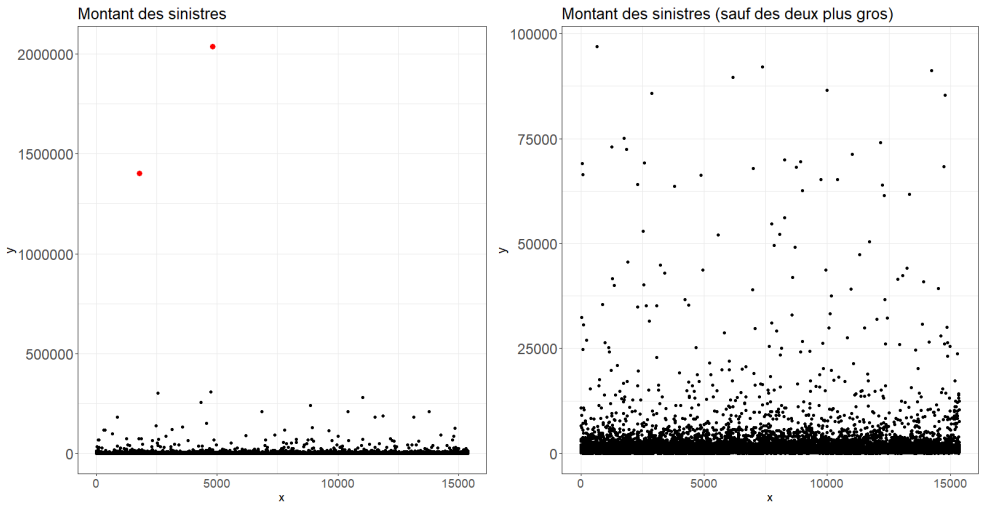

FIGURE *5.1: Montant des sinistres des différents assurés*

suivante :

$$\forall x > 0,\ \bar{F}_u(x) = \mathbb{P}[X - u > x | X > u] \tag{5.3}$$

$$= \frac{\mathbb{P}[X > x + u, X > u]}{\mathbb{P}[X > u]} \tag{5.4}$$

$$= \frac{\bar{F}(x+u)}{\bar{F}(u)} \tag{5.5}$$

Le théorème fondamental de la théorie des valeurs extrêmes est introduit par Pickands, Balkema et de Haan et relit la loi d'excès au-dessus d'un seuil à la distribution GPD. Sous certaines conditions, que nous ne spécifierons pas ici, la distribution d'excès au-dessus d'un seuil tend vers une loi GPD lorsque le seuil tend vers l'infini :

$$\exists \sigma > 0, \exists k \in \mathbb{R}, \forall x > 0\ F_u(x) \underset{u \to \infty}{\to} G(x; \sigma, k)$$

Ainsi, pour un seuil $u > 0$ assez grand, on a l'approximation suivante :

$$\frac{\bar{F}(x+u)}{\bar{F}(u)} \sim G(x; \sigma, k)$$

Soit par changement de variable :

$$\frac{\bar{F}(x)}{\bar{F}(u)} \sim G(x - u; \sigma, k)$$

Nous avons donc une approximation de la loi d'excès au-dessus d'un seuil par une GPD ce qui va nous permettre d'étudier les quantiles de niveau élevé comme nous le souhaitions. Pour que cette approximation soit valable, il faut choisir un seuil u assez élevé.

Néanmoins, si on choisit un seuil trop élevé, nous ne disposerons que de peu de données au-dessus de ce seuil : un compromis est à réaliser. Un résultat intéressant concernant la GPD nous dit que la loi des excès d'une GPD au-dessus d'un seuil u est donc une loi GPD avec le même paramètre de forme k et un paramètre d'échelle qui évolue linéairement avec u. En particulier, si $k = 0$ on retrouve la propriété d'absence de mémoire de la loi exponentielle. On a la relation suivante pour $X \sim GPD(\sigma, k)$:

$$\mathbb{E}[X - u | X > u] = \frac{\sigma}{1-k} + \frac{k}{1-k}u$$

Cette relation linéaire avec le seuil nous conduit à analyser la courbe d'excès moyen (*mean excess plot*) et de choisir le seuil u à partir duquel la courbe devient affine, afin que l'approximation avec une loi GPD soit valable. Le seuil ainsi choisi devient notre critère de gravité ou non d'un sinistre. La courbe de *mean excess* que l'on obtient avec nos données de sinistres est représentée sur la figure 5.2. Sur cette figure 5.2, il est

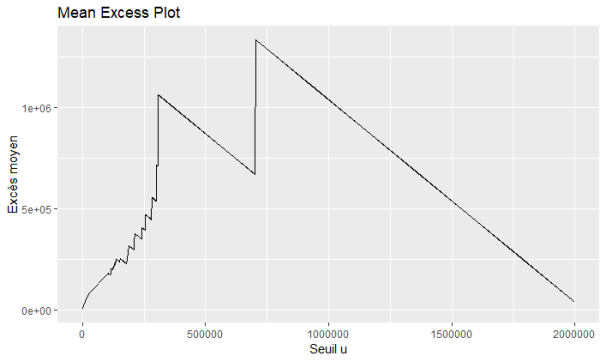

FIGURE *5.2: Mean Excess Plot de nos données de sinistres*

difficile de conclure sur le seuil à retenir. En effet, les valeurs marquantes de palier semblent être graphiquement 300 000 et 700 000. Choisir un tel seuil reviendrait à garder de nombreux sinistres importants, ce qui n'est pas souhaitable en pratique. Un seuil qui semble finalement acceptable est 10 000. Étant donné que seulement 245 sinistres se situent au-dessus de ce seuil, soit moins de 1% des sinistres, on décide de seulement les écrêter, c'est-à-dire de les répartir sur les autres sinistres de manière uniforme. Cela revient à "mutualiser" les sinistres extrêmes entre les différents assurés. Nous représentons sur le graphique 5.3 la densité des montants des sinistres, après avoir réalisé l'écrêtement.

5.1.3.3 Test d'indépendance

L'idée à présent est de réaliser des tests pour savoir si les variables explicatives sont dépendantes de la variable cible de présence de sinistre ou non. Ce test permet de faire un premier tri des variables qui ne nous servirons pas dans l'étude. Pour se faire, il existe le test d'indépendance du Khi-Deux, proposé par K. Pearson en 1900 [56]. Ce test permet

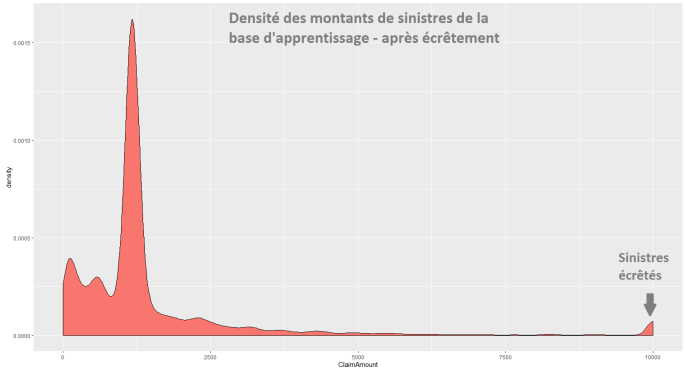

FIGURE *5.3: Densité des sinistres de la base d'apprentissage (Ba) après retraitement*

de vérifier l'absence de lien statistique (indépendance) entre deux variables aléatoires X et Y. L'hypothèse nulle (H0) de ce test est : les deux variables X et Y sont indépendantes. On rejette l'hypothèse nulle lorsque la p-value renvoyée par le test est inférieure à 0.05. Dans le cas de données, voici les résultats obtenus :

Power	CarAge	DriverAge	Brand	Gas	Region	Density
$1.06e10^{-13}$	$< 2.2e10^{-16}$	$< 2.2e10^{-16}$	$< 2.2e10^{-16}$	$1.14e10{-}9$	$< 2.2e10^{-16}$	$< 2.2e10^{-16}$

Toutes les p-value sont inférieures à 0.05, on rejette l'hypothèse nulle et on décide donc de garder toutes les variables explicatives pour la mise en place du modèle coût-fréquence.

5.1.3.4 Analyse graphique des données

Analyse avant regroupement

Analysons de manière univariée les données à disposition. Cette partie a pour objectif de mieux appréhender les variables explicatives et de réaliser les regroupements qui semblent les plus pertinents. Lors de la mise en place d'un modèle GLM pour la tarification, on ne garde jamais de variables numériques : toutes les données sont catégorisées. Cela vient du fait que le modèle GLM implique une monotonie de l'effet d'une variable sur la prédiction. Prenons l'exemple de la variable d'âge du conducteur : si la fréquence de sinistres est très élevée chez les jeunes conducteurs (18-25 ans) et chez les personnes âgées (60-90 ans) mais plus faible chez 26-59 ans, le GLM ne pourra pas traduire cet effet, à cause de cette monotonie induite.

Les figures 5.4 et 5.5 résument la sinistralité selon les différentes modalités de chaque variable présente dans la base. On peut faire les remarques suivantes : on observe une fréquence de sinistres plus élevée pour les assurés avec une voiture utilisant du Diesel, alors que le coût moyen est plus faible chez les assurés ne roulant pas au Diesel. On remarque que la sinistralité chez les jeunes conducteurs (18-26 ans) et les personnes âgées (plus de 85 ans) est plus élevée que pour la tranche d'âges 27-85 ans. Pour la

densité, nous avons enlevé les villes avec plus de 7500 habitants par kilomètre carré, pour une meilleure lisibilité des figures.

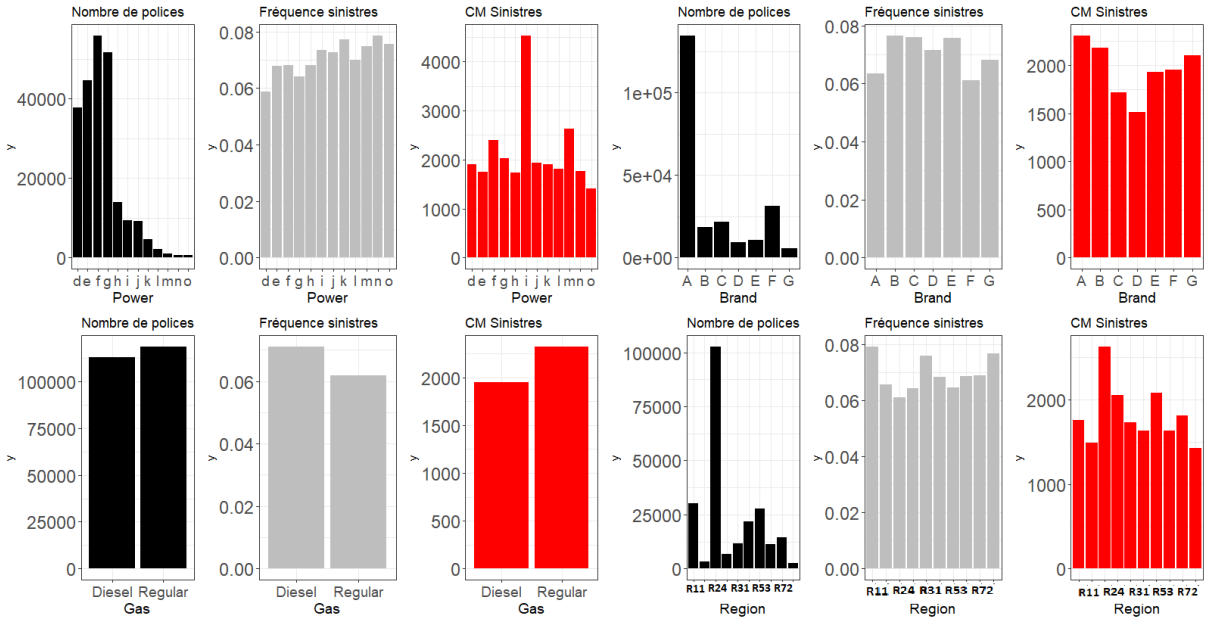

FIGURE *5.4: Analyse de la sinistralité selon les variables Gas, Power, Brand et Region avant regroupement*

La figure 5.6 indique la sinistralité moyenne (en terme de fréquence) au sein des différentes régions de France, en calculant le rapport entre le nombre total de sinistres par région et l'exposition totale des assurés de cette même région. Notons que les cases blanches correspondent aux régions non représentées dans notre base. On observe que la fréquence de sinistres est la plus élevée en Île-de-France et dans le Limousin alors qu'elle est minimale dans le Centre et en Bretagne.

Analyse après regroupement

On effectue les regroupements suivants :

— *Power* : ne contient désormais que 3 catégories (1,2 ou 3) :
- 1 : "d"
- 2 : "e", "f", "g" et "h"
- 3 : "i", "j", "k", "l", "m", "n" et "o"

— *CarAge* : ne contient désormais que 5 catégories :
- 0
- 1-3
- 4-8
- 9-12
- 13+

— DriverAge : ne contient désormais que 5 catégories :

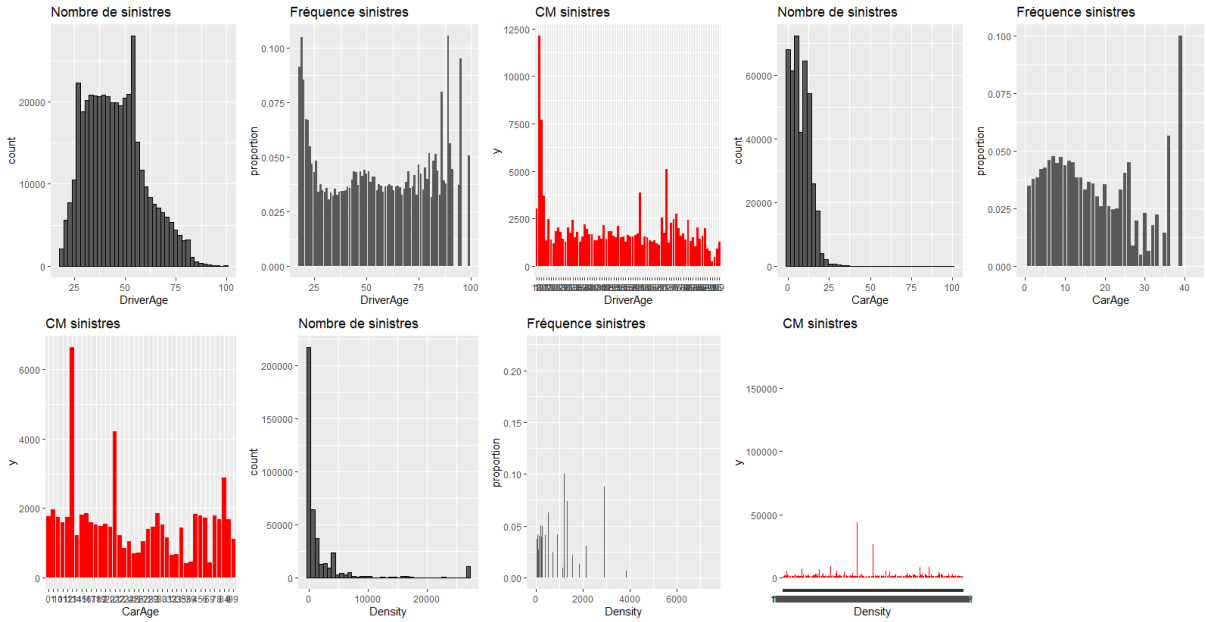

FIGURE *5.5: Analyse de la sinistralité selon les variables CarAge, Region et Density avant regroupement*

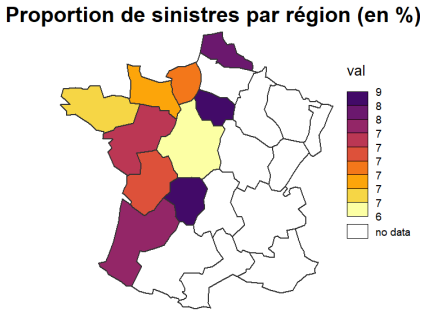

FIGURE *5.6: Analyse de la sinistralité dans les différentes régions*

- 18-26
- 27-42
- 43-55
- 56-85
- 85+

— *Brand* : ne contient désormais que 2 catégories :
- F
- !F : A, B, C, D, E et G

— *Gas* est inchangé, avec deux catégories :
- Diesel
- Regular

— *Region* ne contient désormais que 7 catégories :
  - R11
  - R23-31-72-54 (regroupement des régions R23, R31, R72 et R54)
  - R24
  - R25
  - R52
  - R53
  - R74

— *Density* ne contient désormais que 3 catégories :
  - 0-100
  - 100-19999
  - 20000+

Les graphiques de la figure 5.7 analysent la sinistralité selon les modalités des différentes variables, une fois le retraitement effectué.

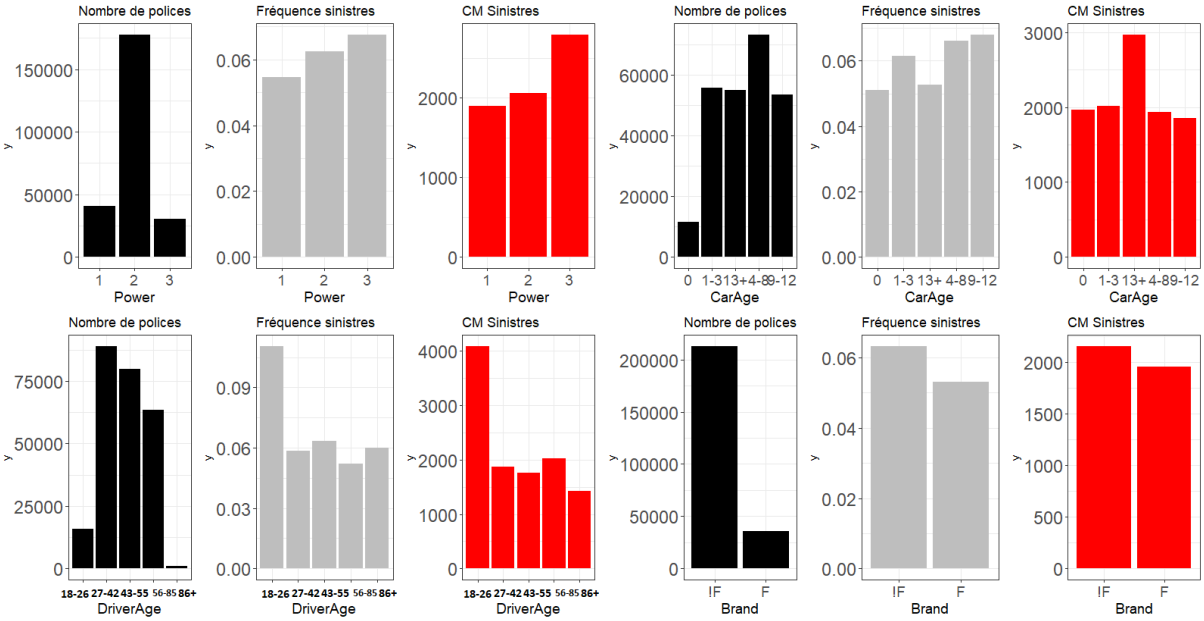

FIGURE *5.7: Analyse de la sinistralité selon les variables Power, CarAge, DriverAge et Brand après regroupement. En noir : le nombre de polices par modalité, en gris : la fréquence de sinistre par modalité et en rouge : le coût moyen des sinistres par modalité*

5.1.3.5 Découpage de la base de données

Pour mettre en place notre modèle afin d'estimer la prime pure, nous devons séparer notre base de données en deux : une base d'apprentissage et une base de test. Nous choisissons les proportions suivantes : 85 % des données pour l'entraînement (et la validation) et 15 % pour le test. Les données de validation nous serviront à optimiser les paramètres

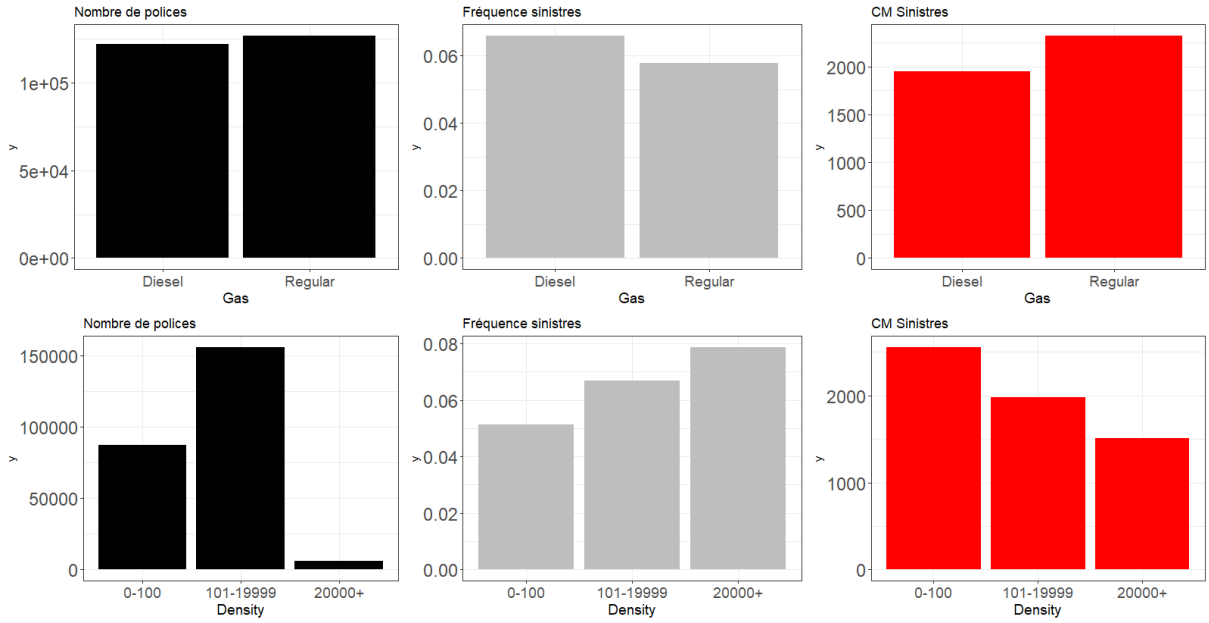

FIGURE *5.8: Analyse de la sinistralité selon les variables Gas et Density après regroupement*

associés au modèle considéré, avec notamment une validation-croisée. On s'assure également que les proportions de sinistres sont respectées au sein des bases d'apprentissage et de test afin de ne pas introduire un biais lors de l'apprentissage. Notons que nous avons effectuer un retraitement sur l'exposition des assurés. Afin d'éviter des valeurs irréalistes du nombre de sinistres annuels, nous avons considéré un seuil sur l'exposition à 0.25, c'est-à-dire que toute exposition inférieure à 0.25 (un quart d'année) sera considérée égale à 0.25. Cela vient du fait que la variable modélisée est $ClaimNb/Exposure$ et le fait de diviser par l'exposition peut donner des valeurs extrêmement élevées.

5.1.3.6 Corrélation entre les variables de la base de données

Généralement, lors de la mise en place de modèles d'apprentissage, nous ne souhaitons pas utiliser des variables trop corrélées entre elles. On préfère avoir des variables totalement décorrélées, ce qui est rarement le cas en pratique. C'est pourquoi une analyse préliminaire est nécessaire avant l'implémentation. Étant donné que nous disposons de plusieurs variables catégorielles, il n'est pas possible d'utiliser la corrélation classique de Pearson. Une variante de ce coefficient s'appelle le V de Cramer (ou parfois le phi de Cramer ϕ_c). Il a été introduit par Cramer en 1946 et permet de mesurer l'association entre deux variables nominales, en renvoyant non plus une valeur entre -1 et 1 mais dans l'intervalle [0,1]. Dans le cadre de nos données, nous savons déjà que les variables de densité et de région ne sont pas indépendantes, étant donné que la majorité des villes à forte densité (supérieure à 20000 habitants/km^2) se trouve en Île-de-France. La matrice de corrélation est donnée sur le graphique 5.9. Nous observons finalement que nos variables

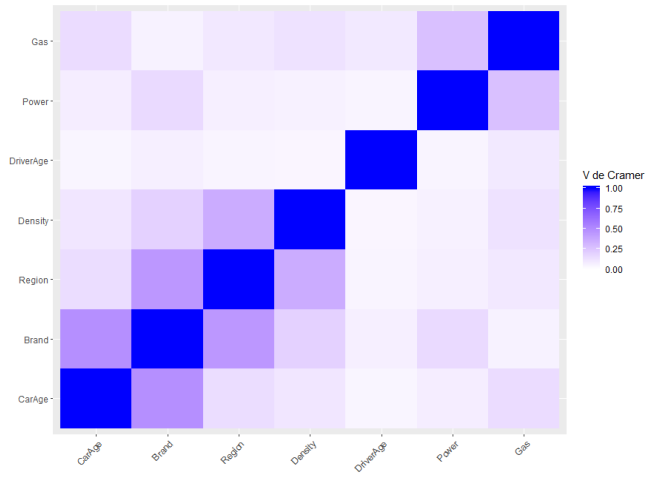

FIGURE *5.9: Matrice de corrélation, calculée à l'aide de Cramer V, sur la base de données après retraitement*

ne sont pas très corrélées entre elles, mais le sont tout de même ce qui devra être pris en compte lors de la mise en place des méthodes d'interprétation des modèles.

5.1.4 Choix de la métrique d'évaluation

Une fois nos modèles ajustés, une comparaison sera nécessaire afin de conclure sur le "meilleur" modèle. Celle-ci se fera sur la base de test à l'aide d'une métrique d'évaluation, à choisir. Les plus classiques sont le MSE (Mean Square Error) et le MAE (Mean Absolute Error). Dans le cas de la fréquence (ou du coût), nous évaluerons les performances de notre modèle en comparant la valeur y_{pred} prédite par le modèle avec y_{theo} la valeur cible théorique, à savoir $ClaimNb/Exposure$ (nombre de sinistres annuels). Lorsque l'on parle de MSE et MAE, on utilise les formules :

$$mse = rmse^2 = \frac{1}{n}\sum_{i=1}^{n}(y_{pred}^{(i)} - y_{theo}^{(i)})^2 \; et \; mae = \frac{1}{n}\sum_{i=1}^{n}|y_{pred}^{(i)} - y_{theo}^{(i)}|$$

Afin de faciliter les comparaisons des résultats, il est courant de normaliser le MSE. Une première approche courante est d'utiliser le RMSE (ou RMAE) relatif qui permet de mesurer directement le pourcentage d'erreur en prédiction. Dans le cadre de nos données, celui-ci n'est pas utilisable, car la variable cible (nombre de sinistres annuels) contient de nombreux zéros et il n'est alors pas possible de diviser par la valeur théorique. D'autres approches existent et reposent toutes sur le calcul du RMSE, divisé par un terme, dépendant des valeurs théoriques de la variable cible que l'on cherche à estimer. Les trois formules les plus répandues dans la littérature sont les suivantes :

— Normalisation par la valeur moyenne :

$$RMSE_{mean} = \frac{RMSE}{\bar{y}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{pred}^{(i)} - y_{theo}^{(i)})^2}}{\frac{1}{n}\sum_{i=1}^{n} y_{theo}^{(i)}}$$

— Normalisation par l'écart entre la valeur maximale et minimale :

$$RMSE_{min,max} = \frac{RMSE}{y_{theo}^{max} - y_{theo}^{min}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{pred}^{(i)} - y_{theo}^{(i)})^2}}{max_{1\leq i\leq n} y_{theo}^{(i)} - min_{1\leq i\leq n} y_{theo}^{(i)}}$$

— Normalisation par l'écart interquantile :

$$RMSE_Q = \frac{RMSE}{Q3(y_{theo}) - Q1(y_{theo})} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{pred}^{(i)} - y_{theo}^{(i)})^2}}{Q3(y_{theo}) - Q1(y_{theo})}$$

où $Q3(y_{theo}) - Q1(y_{theo})$ est la différence entre le quantile d'ordre 0.75 et 0.25 des valeurs théoriques de la variable cible.

Pour les données de fréquence, la présence de zéros empêche l'utilisation de cette dernière normalisation ($Q3(y_{theo}) - Q1(y_{theo}) = 0$), c'est pourquoi nous retenons seulement les deux premières.

Notons que les métriques utilisées pour mesurer l'erreur de nos modèles reposent toutes sur le MSE et MAE. Cependant, comme nous le verrons dans les parties suivantes, il n'est pas toujours possible d'optimiser ces grandeurs en même temps. En effet, il est possible qu'un modèle présente un meilleur MSE qu'un autre, mais un moins bon MAE. C'est pourquoi, nous devons définir auparavant quel critère nous retenons, afin de savoir comment optimiser nos paramètres en conséquence. Le MSE et le MAE sont sujets à de nombreuses recherches et aucun critère ne semble être meilleur qu'un autre dans toutes les situations : un choix subjectif est à réaliser. Tout d'abord, on peut noter que le MSE permet d'éviter l'utilisation de valeurs absolues contrairement au MAE, ce qui est une propriété très souvent désirée. De plus, l'avantage principal du RMSE (ou MSE) est qu'il pénalise fortement les larges erreurs. Par exemple, un modèle se trompant de 10 sur la valeur théorique sera pénalisé exactement deux fois plus par le MAE, en comparaison d'un modèle se trompant de 5 alors qu'avec le MSE la pénalisation sera plus importante que ce facteur 2. Dans le cadre de notre modélisation, il semble intéressant de pénaliser les erreurs élevées, d'où le choix du MSE comme critère pour la suite de l'étude. Ainsi, lors de la construction des modèles, la mesure qui devra être utilisée est celle de la norme 2 (MSE), étant donné qu'il s'agit de la grandeur que l'on souhaite optimiser.

5.2 Mise en place du modèle GLM

5.2.1 GLM fréquence

La première étape de notre étude consiste en la mise en place de notre modèle de fréquence de sinistres. La variable que l'on cherche à prédire ici est le rapport entre

le nombre de sinistres et l'exposition de l'assuré. La variable d'exposition *Exposure* est alors un *offset*. Avant d'inclure les variables explicatives, nous allons ajuster un modèle de fréquence, dit trivial, qui sera notre modèle de référence. Les résultats obtenus sont résumés dans le tableau 5.2.

AIC	Déviance	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
115449	88907	0.18459	0.13153	0.42964	6.01197	0.03580

TABLE *5.2: Résultats du GLM fréquence "trivial"*

Le modèle GLM retenu ici est poissonnien, avec une fonction de lien logarithmique : il s'agit de l'approche la plus couramment utilisée pour la modélisation de la fréquence des sinistres en tarification. Les approches par la loi binomiale négative et la loi quasi-poisson donnent des résultats sensiblement proches et ne sont donc pas détaillés ici.

Afin de trouver le "meilleur" modèle GLM, nous mettons en place une approche backward, c'est-à-dire que l'on part du modèle complet avec toutes les variables explicatives dont nous disposons et nous retirons à chaque étape la variable qui est la moins significative. Nous jugerons alors de la qualité du modèle à l'aide des critères d'AIC, de déviance et de MSE.

Commençons donc par ajuster le modèle complet, utilisant les variables *Power*, *Brand*, *CarAge*, *DriverAge*, *Gas*, *Density* et *Region* après regroupement. Les résultats obtenus sont indiqués sur la figure 5.3. Il s'agit de la sortie délivrée par *R* lors de l'ajustement à l'aide de la fonction *glm* du package *MASS*.

	Estimate	Std. Error	z value	*pvalue*
(Intercept)	-2.3104	0.0640	-36.08	0.0000
Power2	0.0956	0.0257	3.73	0.0002
Power3	0.2465	0.0327	7.54	0.0000
CarAge1-3	0.1808	0.0475	3.81	0.0001
CarAge4-8	0.2182	0.0479	4.56	0.0000
CarAge9-12	0.2408	0.0491	4.91	0.0000
CarAge13+	0.0072	0.0503	0.14	0.8855
DriverAge27-42	-0.7032	0.0290	-24.25	0.0000
DriverAge43-55	-0.6187	0.0292	-21.21	0.0000
DriverAge56-85	-0.7509	0.0310	-24.20	0.0000
DriverAge86+	-0.4118	0.1339	-3.08	0.0021
GasRegular	-0.1462	0.0182	-8.02	0.0000
BrandF	-0.2470	0.0301	-8.20	0.0000
RegionR23_31_72_54	-0.1274	0.0318	-4.01	0.0001
RegionR24	-0.2019	0.0305	-6.62	0.0000
RegionR25	-0.2014	0.0578	-3.49	0.0005
RegionR52	-0.1206	0.0374	-3.23	0.0012
RegionR53	-0.1815	0.0360	-5.04	0.0000
RegionR74	0.0935	0.0820	1.14	0.2544
Density101-19999	0.2680	0.0202	13.27	0.0000
Density20000+	0.4236	0.0578	7.33	0.0000

TABLE *5.3: Résumé du GLM fréquence "complet" (utilisant toutes les variables explicatives disponibles)*

Les informations qui nous intéressent pour juger de la qualité du modèle sont données dans le tableau 5.4. Notons bien que le MSE et le MAE indiqués sont calculés sur la base de test.

AIC	Déviance	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
114255.5	87673.1	0.18416	0.13231	0.429140	6.005016	0.035762

TABLE *5.4: Résultats du GLM fréquence complet*

La deuxième étape consiste à ajuster tous les modèles en retirant une variable, c'est-à-dire tous les modèles utilisant 6 variables dans notre cas. A la fin de cette étape, l'idée est de sélectionner celui avec le meilleur AIC (ou la meilleure déviance) puis à partir de celui-ci, réitérer le processus en enlevant une variable et en sélectionnant le nouveau "meilleur" modèle. Dans le cadre de notre GLM fréquence, les résultats de cette première étape sont présentés dans le tableau 5.5.

On observe qu'aucun modèle ne fait mieux, en terme d'AIC et de déviance, que celui utilisant toutes les variables. L'algorithme backward s'arrête donc à la fin de la première étape et le modèle complet est finalement retenu.

	Déviance	AIC
Complet	87673.09	114255.5
-Power	87730.55	114308.96
-CarAge	87783.89	114358.31
-DriverAge	88270.33	114844.75
-Brand	87742.54	114322.95
-Gas	87737.61	114318.03
-Region	87728.94	114299.35
-Density	87862.57	114440.99

TABLE *5.5: Résultats des modèles GLM fréquence lorsque l'on a retiré une variable du modèle complet*

D'autres classes de GLM ont été mises en oeuvre, à savoir le modèle quasi-poisson et le modèle binomial-négatif. Les différences de performance avec le GLM Poisson sont extrêmement faibles et nous avons décidé de les écarter de notre étude.

5.2.2 GLM sévérité

Pour la mise en place du modèle GLM-coût, la méthodologie est similaire à celle du GLM fréquence. Tout d'abord, nous ajustons un modèle trivial, c'est-à-dire sans variable explicative. La base de données utilisée est la base d'apprentissage définie auparavant, en gardant uniquement les assurés avec un montant de sinistres strictement positif, ce qui représente 13798 lignes. La base utilisée pour tester nos modèles sera également la base de test définie précédemment, en conservant les montants de sinistres strictement positifs, ce qui représente 2383 lignes. La variable cible est alors le rapport entre le montant des sinistres et le nombre de sinistres ($ClaimAmount/ClaimNb$), en utilisant le nombre de sinistres comme poids. En d'autres termes, lors de l'ajustement du GLM, pour une observation donnée, on accorde un poids proportionnel aux nombres de sinistres subis par l'assuré. Cela revient à écrire $weights = ClaimNb$ dans la fonction *glm* de R.

Les résultas d'AIC, de déviance, de MSE, de MAE et de RMSE du modèle de coût trivial sont donnés dans le tableau 5.6.

AIC	Déviance	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
226488.4	11746.1	$1.086180.10^8$	1941.673	$1.042199.10^4$	5.117237	0.03459

TABLE *5.6: Résultats du GLM coût trivial*

Une fois le modèle trivial ajusté comme référence, l'idée est de trouver le meilleur modèle GLM en incluant les variables explicatives. Pour cela, nous mettons en place un modèle GLM Gamma, avec une fonction de lien logarithmique. Plutôt que de réaliser une approche "backward", nous décidons d'appliquer une méthode "forward", en partant du modèle vide (i.e le modèle trivial) et d'ajouter une à une les variables les plus significatives. Ce choix vient du fait que dans la majorité des GLM coût mis en place, moins de variables sont significatives dans l'ajustement. Une fois ces différentes étapes réalisées, le modèle finalement retenu n'utilise que cinq variables sur les sept à disposition, à savoir : Power, CarAge, Brand, Gas et Density. Le résumé de ce modèle est donné dans le tableau 5.7.

	Estimate	Std. Error	t value	*p-value*
(Intercept)	7.36	0.06	113.32	0.00
Power2	0.04	0.03	1.25	0.21
Power3	0.10	0.04	2.64	0.01
CarAge1-3	-0.00	0.05	-0.04	0.97
CarAge4-8	-0.02	0.05	-0.29	0.77
CarAge9-12	-0.11	0.05	-1.96	0.05
CarAge13+	-0.10	0.06	-1.74	0.08
DriverAge27-42	-0.13	0.03	-3.87	0.00
DriverAge43-55	-0.10	0.03	-3.05	0.00
DriverAge56-85	-0.11	0.03	-3.25	0.00
DriverAge86+	-0.06	0.15	-0.37	0.71
BrandF	0.06	0.03	1.99	0.05
GasRegular	-0.00	0.02	-0.07	0.95
Density101-19999	-0.00	0.02	-0.00	1.00
Density20000+	-0.17	0.06	-2.79	0.01

TABLE *5.7: Résumé du meilleur GLM pour modéliser la sévérité, obtenu à l'aide d'une approche forward*

Les informations qui nous intéressent pour juger de la qualité du modèle sont résumées dans le tableau 5.8. On observe un AIC et une déviance plus faibles pour le modèle obtenu à l'aide de l'approche forward que le modèle moyen.

AIC	Déviance	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
226419	11672.95	108565294	1938.289	10419.47	5.1160	0.034582

TABLE *5.8: Résultats du GLM coût finalement retenu*

5.2.3 Modèle GLM final : coût x fréquence

La variable finale que l'on souhaite modéliser est le rapport entre le montant des sinistres et l'exposition de l'assuré. Pour ce faire, il suffit de multiplier les prédictions

faites par le GLM fréquence et le GLM de coût moyen. En effet, ces derniers prédisent respectivement $ClaimNb/Exposure$ et $ClaimAmount/ClaimNb$, et donc leur produit correspond bien à notre variable cible. Étant donné que la sévérité et la fréquence ont été étudiées séparément, il n'est pas assuré que le résultat combiné des deux soient plus performants que le modèle moyen. Nous analysons toujours les performances du modèle à travers le MSE et MAE, comme il est indiqué dans le tableau 5.9. Ce tableau nous indique le modèle issu de la combinaison de nos "meilleurs" GLM fréquence et GLM coût est légèrement plus performant en terme de MSE que le modèle "moyen" mais moins performant en terme de MAE.

	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
GLM "moyen"	13431960	278.8234	3664.964	24.03738	0.0062515
"meilleur" GLM	13430428	281.2167	3664.755	24.0360	0.0062511

TABLE *5.9: Résultats de la combinaison du GLM fréquence et du GLM coût*

5.3 Mise en place du modèle de boîte noire (XGBoost)

Dans cette partie, nous tenterons une approche de type machine learning, alternative au modèle GLM, pour modéliser la sinistralité.

5.3.1 Modélisation de la fréquence (XGBoost fréquence)

Comme indiqué précédemment, nous nous intéresserons principalement à la fréquence des sinistres (i.e. à la variable $ClaimNb/Exposure$). Le modèle que l'on retient ici est le XGBoost, véritable star des compétitions de machine learning, que nous avons détaillé dans une partie antérieure. La principale force du XGBoost réside dans ses nombreux hyper-paramètres qui permettent d'ajuster un modèle le plus finement possible. L'objet de cette partie, qui est d'ajuster le "meilleur" XGBoost qui modélise la fréquence, sera basé essentiellement sur le choix optimal de ces paramètres, à l'aide de validation-croisée. L'idéal serait d'utiliser une validation de type Leave-One-Out Cross-Validation (LOOCV) mais étant donné la taille de notre base de données, nous nous limiterons à une k-fold Cross-Validation avec $k = 5$ folds. Trois types de paramètres seront à choisir : les paramètres généraux, les paramètres du booster (ceux-ci dépendent des choix précédents) et enfin les paramètres d'apprentissage. Les paramètres que l'on décide d'ajuster sont les suivants :

- $nrounds$: nombre maximum d'itérations du Boosting.
- max_depth : profondeur maximale des arbres utilisés. La valeur par défaut dans la fonction xgboost de R est : 6
- eta (η) : contrôle le taux d'apprentissage en modifiant la contribution de chaque arbre d'un facteur $\eta \in [0, 1]$ lorsque celui est ajouté à l'approximation courante. Ce paramètre permet en particulier d'éviter le surapprentissage. En général, une

valeur faible de η implique une valeur plus élevée pour *nrounds*. La valeur par défaut dans la fonction xgboost de R est 0.3.

- *gamma* (γ) : correspond à la réduction de perte minimale requise pour créer une partition supplémentaire sur un noeud feuille de l'arborescence. Plus γ est grand, plus l'algorithme sera conservateur. Par défaut, il vaut 0.
- *colsample_bytree* : ratio de sous-échantillonnage des colonnes lors de la construction de chaque arbre. Par défaut, il vaut 0.
- *min_child_weight* : il s'agit de la somme minimale de poids d'instance nécessaire dans un noeud fils. Si la partition conduit à une somme de poids d'instance inférieure à ce paramètre, le processus de construction abandonnera toute partition supplémentaire. Plus ce paramètre est grand, plus l'algorithme est conservateur. Par défaut, il vaut 1.
- *subsample* : ratio de sous-échantillonnage de l'instance d'entraînement. Par exemple, si celui-ci vaut 0.5, l'algorithme collectera aléatoirement la moitié des données de la base d'apprentissage pour faire grandir les arbres. Ceci permet également d'éviter le surapprentissage et diminue le temps d'exécution. Par défaut, il vaut 1.

Pour choisir les meilleurs paramètres de notre modèle XGBoost, la technique idéale serait de tester toutes les combinaisons possibles de paramètres et de regarder à l'aide de validation croisée, celle qui donne la meilleure performance en terme de MSE. Deux problèmes se posent alors. Tout d'abord, notre base de données étant relativement massive, les temps de calcul sont longs. De plus, tester toutes les combinaisons n'est pas possible. En effet, même si l'on choisissait une plage de seulement 5 valeurs par paramètre (sur les 7 paramètres choisis ci-dessus), nous aurions $5^7 = 78125$ modèles différents à ajuster, et à valider à l'aide d'une validation croisée (5 folds), ce qui ferait plus de 300 000 ajustements de modèles différents. Il faut donc raisonner différemment et choisir judicieusement quel(s) paramètre(s) ajuster les uns après les autres.

Notre première étape sur le choix du nombre d'itérations *nrounds* et du taux d'apprentissage *eta*, qui sont intimement liés. Dans notre cas, nous choisissons un nombre maximal d'arbres de 250. Dans la majorité des algorithmes XGBoost implémentés en pratique (dans les compétitions Kaggle par exemple), ce paramètre est proche de 1000. Dans le cas de notre étude, nous nous sommes aperçus qu'utiliser une valeur supérieure à 250 conduisait de manière quasi systématique à du surapprentissage. C'est pourquoi nous avons décidé de fixer ce seuil, en plus d'éviter un temps d'exécution trop conséquent. Ensuite, nous voulons trouver un taux d'apprentissage assez bon associé à ce nombre d'arbres, sachant que pour un nombre d'itérations inférieur à 250, celui-ci ne sera peut-être pas suffisant. Nous nous restreignons à la plage $\eta \in [0.01, 0.3]$. Sachant que la profondeur maximale des arbres (*max_depth*) dépend également des paramètres *nrounds* et *eta*, nous décidons d'analyser l'effet combiné des 3 paramètres. La plage utilisée pour *max_depth* est $[3, 6]$, comme cela est couramment fait dans la littérature. Les autres paramètres décrits ci-dessus sont fixés à leur valeur par défaut. Nous obtenons alors la courbe de validation croisée de la figure 5.10, basée sur le RMSE moyen dans les 5 folds.

Pour 250 itérations au maximum, un taux d'apprentissage de 0.3 semble être un bon

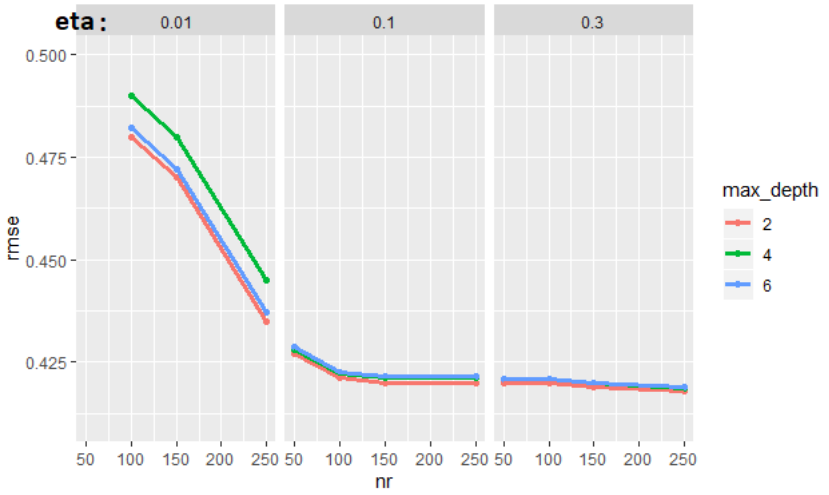

FIGURE *5.10: Première étape dans le choix des paramètres du XGBoost fréquence à l'aide d'une validation croisée (5-fold CV) : choix de eta*

point de départ.

Après avoir fixé *eta* à 0.3, nous voulons optimiser les paramètres max_depth et min_child_weight. Nous décidons de fixer max_depth à 2 comme le suggère le graphique précédent et de faire varier min_child_weight entre 1 et 3. Nous obtenons alors les résultats de la figure 5.11. Nous remarquons que les courbes sont quasiment superposées et donc que ce paramètre influe peu sur les résultats obtenus.

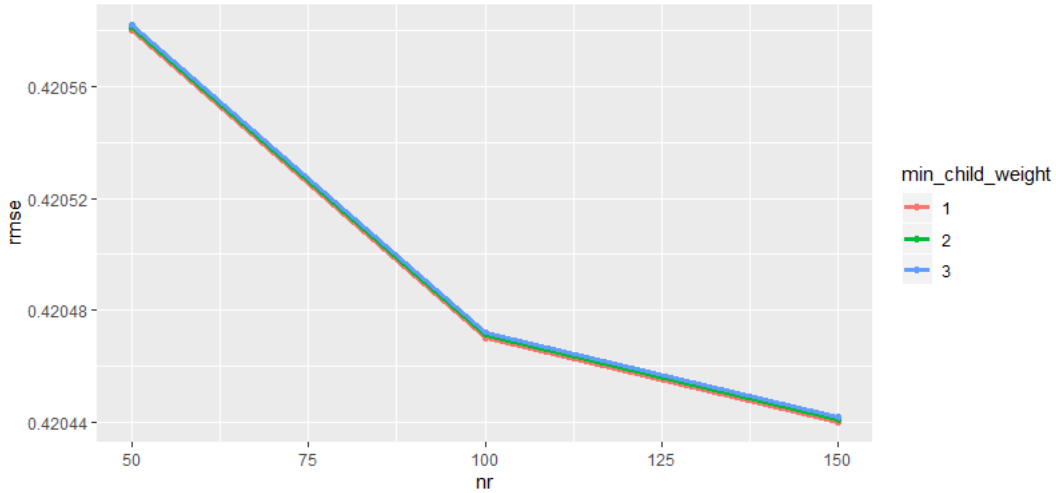

FIGURE *5.11: Deuxième étape dans le choix des paramètres du XGBoost fréquence à l'aide d'une validation croisée (5-fold CV) : choix de max_depth et min_child_weight*

Nous optons alors pour les paramètres suivants : $min\_child\_weight = 1$ et $max\_depth =$

2. A présent, nous pouvons essayer différentes valeurs de sous-échantillonnage pour les lignes et les colonnes. Nous choisissons les plages $subsample \in [0.4, 1]$ et $colsample\_bytree \in [0.5, 1]$. Les résultats de la validation croisée sont indiqués sur la figure 5.12. Ils nous conduisent à choisir les valeurs suivantes : $colsample\_bytree = 1$ et $subsample = 1$.

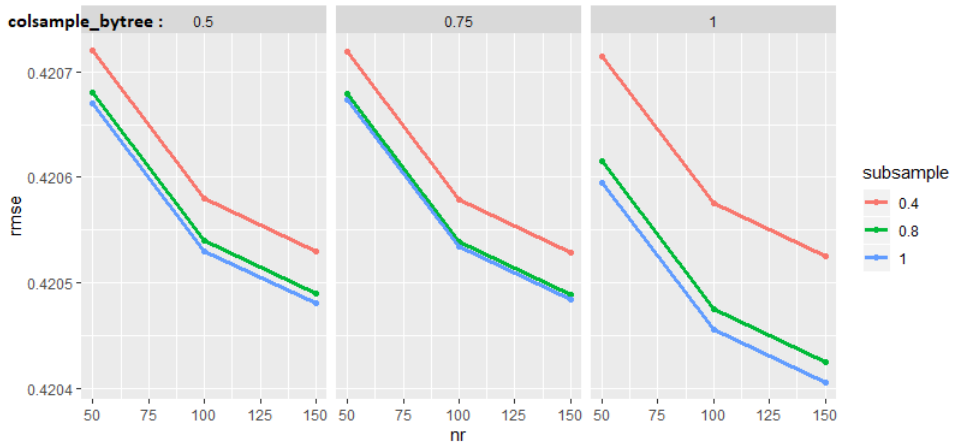

FIGURE *5.12: Troisième étape dans le choix des paramètres du XGBoost fréquence à l'aide d'une validation croisée (5-fold CV) : choix de colsample_bytree et subsample*

A présent, nous voulons analyser l'impact de *gamma* sur l'ajustement du modèle. La procédure repose toujours sur le RMSE dans les différents folds de la validation croisée. Les résultats sont affichés sur la figure 5.13.

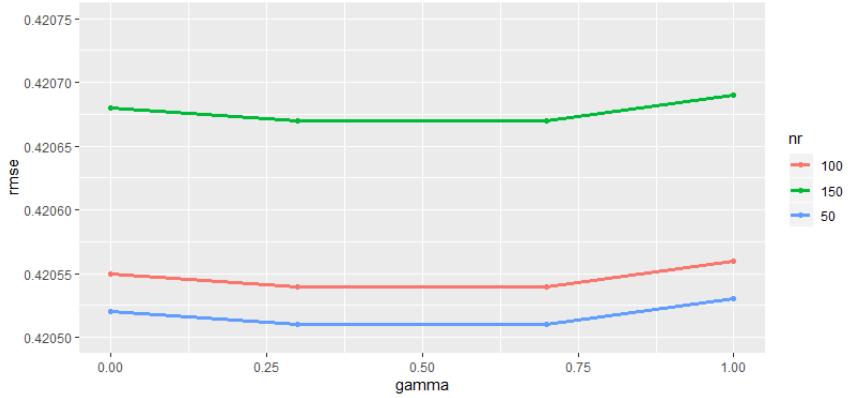

FIGURE *5.13: Quatrième étape dans le choix des paramètres du XGBoost à l'aide d'une validation croisée (5-fold CV) : choix de gamma*

Finalement, le paramètre $gamma = 0.7$ est retenu. La cinquième et dernière étape consiste à réduire le paramètre *eta* du taux d'apprentissage, en augmentant éventuellement le nombre d'itérations *nrounds*.

Cette dernière figure nous conduit à réaliser le choix suivant : $nrounds = 250$ et

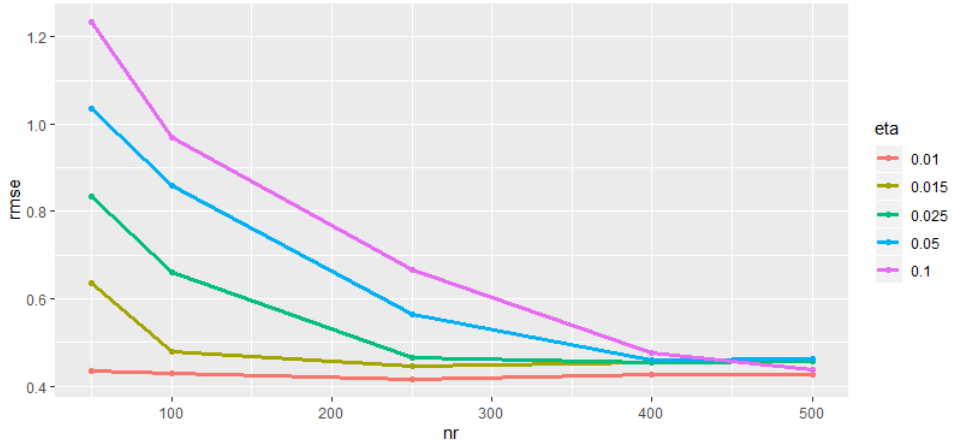

FIGURE *5.14: Cinquième étape dans le choix des paramètres du XGBoost fréquence à l'aide d'une validation croisée (5-fold CV) : choix combiné de eta et nrounds*

$eta = 0.1$.

Nous avons à présent trouvé les différents hyper-paramètres de notre XGBoost. Ceux-ci sont résumés dans le tableau 5.10.

nrounds	*max_depth*	*eta*	*gamma*	*colsample_bytree*	*min_child_weight*	*subsample*
250	2	0.1	0.7	1	1	1

TABLE *5.10: Paramètres finaux du modèle XGBoost fréquence*

Au cours de cette étape, nous avons seulement utilisé la base d'apprentissage (et de validation). A présent, nous testons les performances du modèle en analysant les prédictions qu'il réalise sur la base de test, toujours par le biais du MSE.

	MSE	MAE
GLM "moyen"	0.1845885	0.1315303
"meilleur" GLM	0.1841617	0.1323143
XGBoost final	0.1840609	0.1361526
XGBoost 2	0.1838469	0.1350176

TABLE *5.11: Comparaison, sur la base de test, des différents modèles mis en place pour modéliser la fréquence*

Ajustement d'un XGBoost fréquence en gardant les variables numériques inchangées

Au cours de cette partie, nous avons ajusté un modèle XGBoost en gardant les variables transformées de la même manière que pour le GLM. Le regroupement des données qui a été réalisé précédemment était indispensable dans le cadre de la mise en place du

GLM. En effet, un modèle GLM induit nécessairement une monotonie des différentes variables numériques sur la variable cible. Par exemple, dans le cas de notre étude avec la variable d'âge du conducteur, si nous n'avions par créé les buckets "18-26", "27-42", "43-55", "56-85" et "86+" nous aurions forcément eu une fréquence de sinistres, estimée par le modèle GLM, croissante (ou décroissante) avec l'âge. Or, nous avons constaté que l'impact de l'âge sur les sinistres ne semble pas monotone : elle est élevée pour les jeunes conducteurs, relativement faible pour les âges intermédiaires et de nouveau élevée pour les conducteurs âgés. Un assureur ne peut se permettre d'ignorer cette information en laissant les variables inchangées. La première création de buckets qui portait sur les variables numériques *CarAge*, *DriverAge* et *Density* était donc nécessaire dans le cadre du GLM. Le deuxième retraitement que nous avons réalisé portait sur les variables déjà catégorielles auparavant, à savoir *Power*, *Brand* et *Region*. La variable binaire *Gas* a elle été inchangée. Ce retraitement avait pour objectif de réduire le nombre de modalités par variable, notamment pour éviter le surapprentissage mais aussi pour s'assurer que chaque modalité était suffisamment représentée dans la base. C'est pourquoi la variable *Power* est passée de 12 à 3 modalités, *Brand* de 7 à 2 modalités et enfin *Region* de 10 à 7 modalités. Dans le cadre de la modélisation via l'algorithme XGBoost, aucune monotonie n'est imposée (nous pourrons d'ailleurs le vérifier à l'aide des graphiques de dépendance partielle). C'est pourquoi le premier retraitement n'est pas indispensable et peut éventuellement être mis de côté. Nous décidons d'ajuster un second modèle XGBoost (que l'on notera XGBoost fréquence 2 par la suite), sur la base d'apprentissage ayant subie uniquement le deuxième retraitement, mais en gardant les variables numériques *CarAge*, *DriverAge* et *Density* inchangées. Afin de trouver les paramètres optimaux, nous optons pour la même stratégie que précédemment, en reprenant les 5 étapes, pour choisir *max_depth*, *eta*, *gamma*, *colsample_bytree*, *min_child_weight* et *subsample*. Celles-ci sont représentées sur le graphique 5.15.

Ces différentes courbes obtenues nous conduisent à réaliser le choix de paramètres donné dans le tableau 5.12.

nrounds	*max_depth*	*eta*	*gamma*	*colsample_bytree*	*min_child_weight*	*subsample*
500	2	0.05	0.3	1	2	0.5

TABLE *5.12: Paramètres finaux du modèle XGBoost fréquence 2 (en gardant les variables numériques inchangées)*

Nous remarquons dans le tableau 5.11 (sur la ligne XGBoost 2) que nous obtenons de meilleurs résultats, en terme de MSE, avec ce modèle. Cela permet en outre d'utiliser les différentes méthodes d'interprétation sur des données mixtes, à la fois catégorielles et numériques.

5.3.2 Modélisation de la sévérité (XGBoost coût)

Après avoir modélisé la fréquence, il nous reste la sévérité des sinistres pour obtenir la prime pure, qui sera la combinaison de ces deux modèles. Rappelons que cette fois-ci, nous n'utilisons que les données de la base d'apprentissage pour lesquelles les montants de

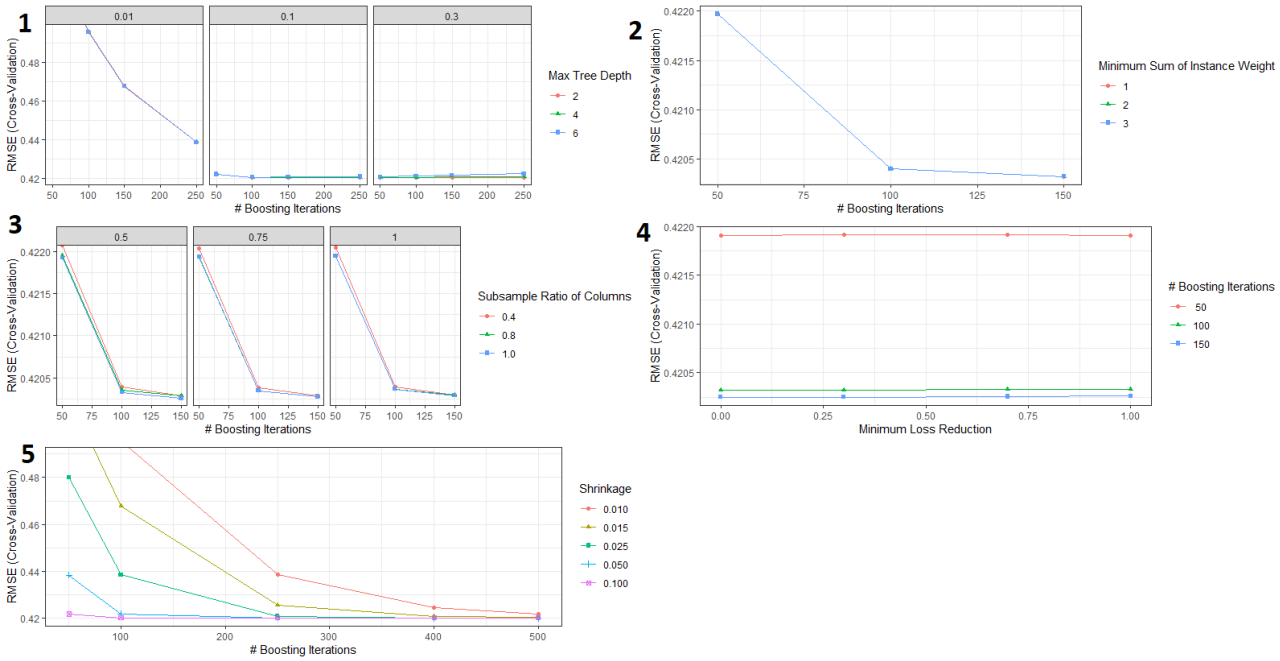

FIGURE *5.15: 5 étapes de validation croisée réalisées pour trouver les meilleurs paramètres du XGBoost de fréquence, en gardant les variables numériques non retraitées*

sinistres sont strictement positifs. La variable à prédire est $ClaimAmount/ClaimNb$, qui correspond au montant moyen d'un sinistre. Comme pour la modélisation fréquentielle, nous optons pour une validation croisée avec $k = 5$ folds, pour trouver les paramètres $nrounds, max\_depth, eta, gamma, colsample\_bytree, min\_child\_weight, subsample$ optimaux. Nous avons réalisé cette étude pour les données après retraitement, et aussi pour les données gardant les variables numériques inchangées. Nous ne présentons ici que les résultats obtenus pour les données retraitées, car nous obtenons des résultats légèrement meilleurs que pour le GLM, ce qui n'est pas le cas lorsque l'on ne retraite pas les variables quantitatives. Les résultats de performance du tableau 5.13 sont donnés pour le même découpage base de test - base d'apprentissage que pour la modélisation en fréquence, obtenus en ajustant un XGBoost avec les paramètres décrits dans le tableau 5.14. La stabilité vis à vis de ce découpage est discutée dans une partie ultérieure.

	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
XGBoost coût	108541340	1936.062	10418.32	5.115432	0.034578
'meilleur' GLM coût	108565294	1938.289	10419.47	5.1160	0.034582

TABLE *5.13: Résultats du XGBoost coût*

nrounds	eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample
171	0.1	2	0.20	0.90	4	0.40

TABLE *5.14: Paramètres retenus pour le XGBoost coût, obtenus à l'aide d'une validation croisée*

5.3.3 Modélisation finale : coût-fréquence

Dans cette partie, on analyse les résultats finaux du modèle total obtenus en rassemblant les deux modèles XGBoost fréquence et coût. Pour cela, la prédiction finale, correspondant à la variable cible $ClaimAmount/Exposure$, qui correspond au montant moyen annuel de sinistres des assurés, est obtenue en multipliant la prédiction du XGBoost fréquence et celle du XGBoost coût.

	MSE	MAE	RMSE	RMSE_{mean}	$\text{RMSE}_{min,max}$
XGBoost total	13430342	288.442	3664.743	24.03593	0.0062510
GLM "moyen"	13431960	278.8234	3664.964	24.03738	0.0062515
"meilleur" GLM	13430428	281.2167	3664.755	24.0360	0.0062511

TABLE *5.15: Résultats du XGBoost total et comparaison avec ceux du GLM*

Ces résultats seront commentés dans une partie ultérieure.

5.4 Analyse de la stabilité des modèles vis à vis des bases de test et apprentissage

Au cours des parties précédentes, nous avons ajusté un modèle à partir d'une base d'apprentissage, obtenue en échantillonnant 85% de la base de données initiale. Au sein de cette base d'apprentissage (qui nous sert également de base de validation), nous avons obtenu alors les paramètres "optimaux", à l'aide de validation croisée (définie dans le chapitre 1). Tous les résultats de performance ainsi calculés ont été réalisés sur la base de test, à savoir sur les données des 15% restants. Bien que la séparation entre l'entraînement et le test ait été faite de manière aléatoire, rien ne nous garantit que sur un nouvel échantillon, les modèles mis en place et les paramètres choisis soient toujours aussi performants. Un des critères primordiaux lors de l'ajustement de modèle d'apprentissage statistique est la stabilité et la non dépendance des résultats selon la base d'apprentissage choisie. C'est pourquoi dans cette partie, nous réalisons la même étude que précédemment en choisissant, toujours de manière aléatoire, une nouvelle base d'apprentissage et une nouvelle base de test, afin de vérifier la stabilité des résultats. Pour ce faire, nous prenons quatre nouveaux échantillonnages de la base initiale. Notre critère de comparaison des résultats se fera toujours par le biais du MSE et du MAE. Dans le tableau 5.16, nous avons indiqué à gauche les valeurs des critères retenus (MSE, MAE etc.) pour différents échantillonnages des bases d'apprentissage et de test, et à droite les gains relatifs (ou perte si la valeur indiquée est négative) par rapport au GLM trivial de référence. Sur la figure 5.16, nous avons représenté la valeur du MSE et du MAE sur la base de test pour les 5 échantillonnages différents et pour les différents modèles (XGBoost, GLM trivial

et "meilleur" GLM). Pour la modélisation de la fréquence, on remarque une certaine stabilité des résultats, avec des courbes à peu près translatées les unes des autres. Pour la modélisation de sévérité, on note une grande instabilité des résultats. Par exemple, le $RMSE_{mean}$ est 5 fois plus élevé entre le premier échantillonnage réalisé et le deuxième. Néanmoins, les gains et pertes relatifs, en comparaison du GLM, sont globalement les mêmes pour tous les échantillons, avec une perte de l'ordre de moins de 1% en MAE, et un gain très faible en MSE.

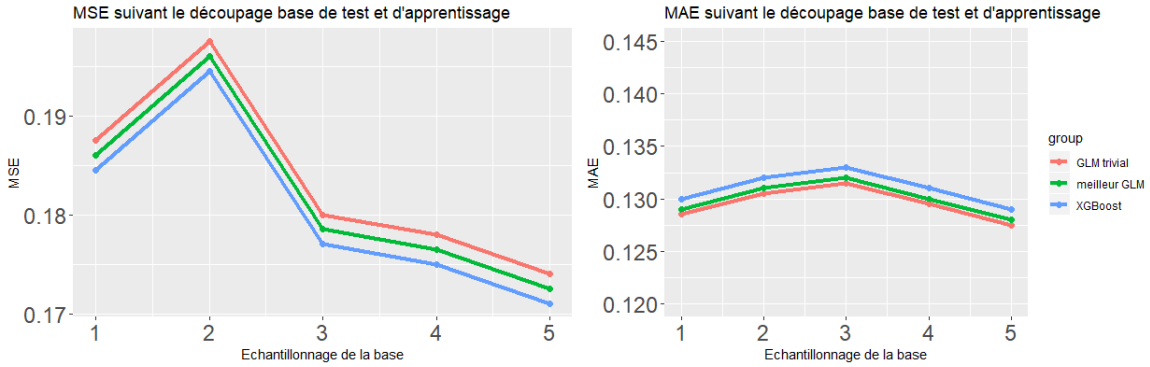

FIGURE *5.16: Analyse de la stabilité des modèles mis en place pour la fréquence résultats sur différents échantillonnages des bases d'apprentissage et de test*

5.5 Analyse critique des résultats et comparaison des deux modèles

Dans cette partie, on reprend les résultats des parties 5.3.3 et 5.4 avec pour objectif d'analyser les performances du modèle XGBoost mis en place et d'avoir un regard critique sur celui-ci. Nous utilisons également les figures 5.17 et 5.18 qui permettent d'analyser la répartition des prédictions des modèles GLM et XGBoost, que ce soit en fréquence, en coût ou au total. Sur la figure 5.18, on a représenté les prédictions de la prime pure du GLM, ordonnées de manière croissante (courbe bleue) auxquelles sont superposées les prédictions du XGBoost (points rouges). A l'aide de toutes ces courbes et également des résultats de performance (MSE, MAE etc.), nous pouvons conclure que les performances des deux modèles sont très similaires.

On observe très clairement sur le tableau 5.15 que le gain apporté par le XGBoost est très limité, voir inexistant. En effet, le critère que l'on a retenu au cours de notre étude est le MSE, et le gain relatif associé en comparaison du meilleur modèle est de l'ordre de 0.001 %. Comme mentionné auparavant, les modélisations de la fréquence et du coût sont réalisées indépendamment. Bien que les résultats soient plus performants pour les deux modèles (gain d'environ 0.3 % pour la fréquence et de 0.07% pour le coût), il n'est pas impossible que la combinaison des deux ne le soit pas, ou le soit dans une moindre mesure comme ici.

	MAE	RMSE	RMSE_mean	RMSE_min_max
XGB freq	0.1344	0.4223	6.1011	0.0352
GLM freq Trivial	0.1294	0.4231	6.1120	0.0353
GLM freq Best	0.1303	0.4224	6.1023	0.0352
XGB Coût	1674.9913	5177.1334	3.0029	0.0378
GLM Coût Trivial	1683.0576	5177.2505	3.0029	0.0378
GLM Coût Best	1677.6616	5178.0482	3.0034	0.0378
XGB Total	272.6917	2454.7075	18.5621	0.0094
GLM Total Trivial	262.2297	2454.9316	18.5638	0.0094
GLM Total Best	264.7175	2454.6745	18.5618	0.0094

	MAE	MSE	RMSE
XGB freq	-3.64%	0.31%	0.15%
GLM freq Trivial	0.00%	0.00%	0.00%
GLM freq Best	-0.58%	0.27%	0.14%
XGB Coût	0.30%	0.01%	0.00%
GLM Coût Trivial	0.00%	0.00%	0.00%
GLM Coût Best	0.12%	0.01%	0.00%
XGB Total	-2.17%	0.00%	0.00%
GLM Total Trivial	0.00%	0.00%	0.00%
GLM Total Best	-0.50%	0.00%	0.00%

	MAE	RMSE	RMSE_mean	RMSE_min_max
XGB freq	0.1342	0.4240	6.1513	0.0286
GLM freq Trivial	0.1293	0.4246	6.1608	0.0287
GLM freq Best	0.1301	0.4240	6.1525	0.0286
XGB Coût	2774.4294	43981	14.8333	0.0216
GLM Coût Trivial	2782.8687	43983	14.8339	0.0216
GLM Coût Best	2779.5004	43981	14.8332	0.0216
XGB Total	421.0532	33070	112.2808	0.0041
GLM Total Trivial	411.9326	33071	112.2820	0.0041
GLM Total Best	414.0132	33070	112.2809	0.0041

	MAE	MSE	RMSE
XGB freq	-3.60%	0.31%	0.15%
GLM freq Trivial	0.00%	0.00%	0.00%
GLM freq Best	-0.47%	0.27%	0.14%
XGB Coût	0.32%	-0.04%	-0.02%
GLM Coût Trivial	0.00%	0.00%	0.00%
GLM Coût Best	0.07%	-0.03%	-0.01%
XGB Total	-3.20%	0.01%	0.00%
GLM Total Trivial	0.00%	0.00%	0.00%
GLM Total Best	-0.62%	0.00%	0.00%

	MAE	RMSE	RMSE_mean	RMSE_min_max
XGB freq	0.1357	0.4241	5.9945	0.0530
GLM freq Trivial	0.1307	0.4247	6.0025	0.0531
GLM freq Best	0.1315	0.4240	5.9940	0.0530
XGB Coût	1813.4742	8522.1759	4.4490	0.0278
GLM Coût Trivial	1820.4263	8526	4.4510	0.0278
GLM Coût Best	1819.5475	8523	4.4493	0.0278
XGB Total	283.9587	3161	21.5868	0.0088
GLM Total Trivial	273.9296	3162	21.5889	0.0088
GLM Total Best	276.2919	3161	21.5857	0.0088

	MAE	MSE	RMSE
XGB freq	-3.75%	0.23%	0.11%
GLM freq Trivial	0.00%	0.00%	0.00%
GLM freq Best	-0.55%	0.21%	0.10%
XGB Coût	0.29%	0.02%	0.01%
GLM Coût Trivial	0.00%	0.00%	0.00%
GLM Coût Best	0.08%	0.01%	0.01%
XGB Total	-2.38%	0.00%	0.00%
GLM Total Trivial	0.00%	0.00%	0.00%
GLM Total Best	-0.52%	0.00%	0.00%

	MAE	RMSE	RMSE_mean	RMSE_min_max
XGB freq	0.1357	0.4253	5.9725	0.0266
GLM freq Trivial	0.1309	0.4260	5.9817	0.0266
GLM freq Best	0.1315	0.4254	5.9735	0.0266
XGB Coût	2046.9673	11077	5.1068	0.0368
GLM Coût Trivial	2053.4730	11074	5.1057	0.0368
GLM Coût Best	2052.0523	11076	5.1065	0.0368
XGB Total	310.0428	5492	31.2018	0.0065
GLM Total Trivial	300.1204	5493	31.2030	0.0065
GLM Total Best	301.9963	5493	31.2027	0.0065

	MAE	MSE	RMSE
XGB freq	-4.01%	0.31%	0.15%
GLM freq Trivial	0.00%	0.00%	0.00%
GLM freq Best	-0.66%	0.25%	0.13%
XGB Coût	0.30%	-0.00%	-0.00%
GLM Coût Trivial	0.00%	0.00%	0.00%
GLM Coût Best	0.22%	0.02%	0.01%
XGB Total	-3.25%	0.01%	0.00%
GLM Total Trivial	0.00%	0.00%	0.00%
GLM Total Best	-0.73%	0.01%	0.00%

TABLE *5.16: Analyse des résultats (MSE, MAE etc.) sur différents échantillonnages de la base d'apprentissage (Ba) et de la base de test (Bt). Sur le tableau de gauche sont représentés les indicateurs de performance sur la base de test pour 5 échantillonnages différents. A droite sont représentés les gains relatifs par rapport au modèle GLM trivial.*

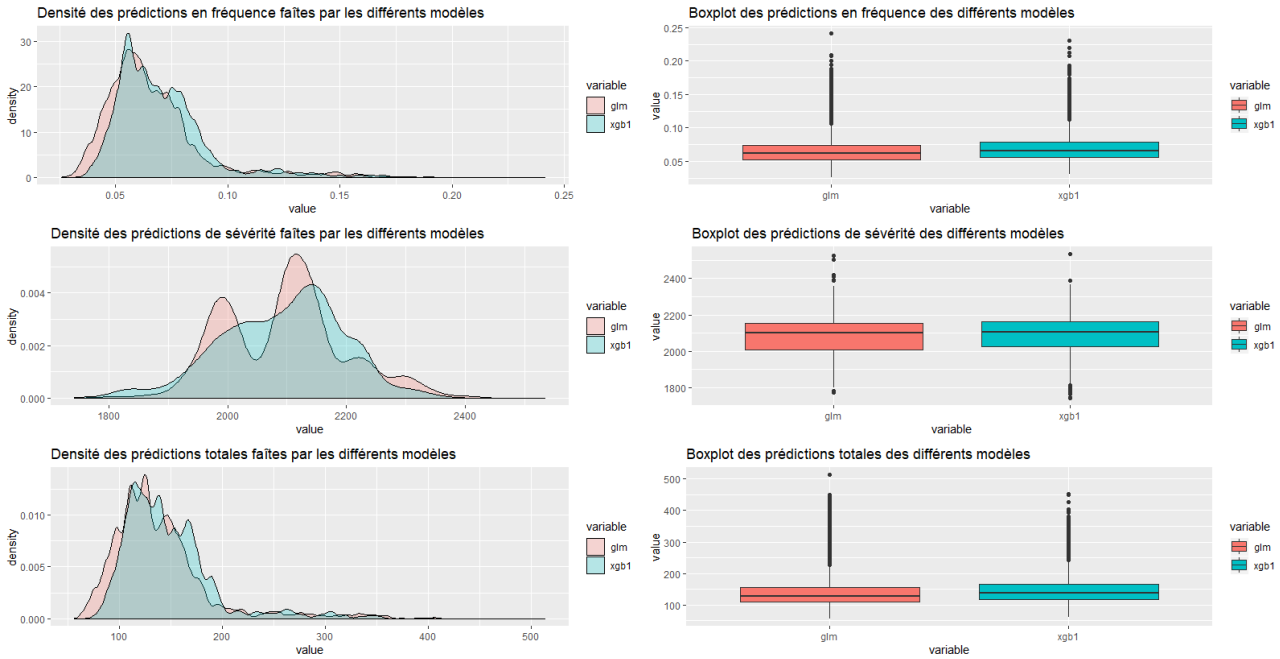

FIGURE *5.17: Densité et Boxplot des prédictions réalisées par le modèle GLM et le modèle XGBoost, en fréquence, coût et au total*

En plus d'un gain infime sur le MSE, notre modèle XGBoost réalise une perte d'environ 3.5 % sur le MAE. Cela peut s'expliquer par le fait que la métrique d'évaluation utilisée par le XGBoost est quadratique et donc rien ne peut indiquer un gain au niveau de ce critère.

Au vu de ces résultats, il semblerait que le modèle XGBoost ne soit pas pertinent dans le cas de notre étude, étant donné le faible gain réalisé. En pratique, il ne serait certainement pas déployé, notamment à cause de la simplicité d'un GLM et de la proximité des résultats entre les deux modèles.

Ce problème rencontré très souvent lors de la mise en place de modèles complexes sur des données actuarielles pourrait faire l'objet d'une étude plus aboutie. En effet, il est courant de voir que les méthodes censées être plus performantes sur les données étudiées, comme celles de tarification automobile, ne soient en réalité équivalentes voire pires que le modèle trivial. Ce problème peut venir du fait que l'on utilise des données publiques, ayant peu de variables explicatives (7 dans notre cas) et n'étant pas très corrélées à notre variable cible. A la lumière des nombreux articles étudiés évoquant des performances bien meilleures avec des modèles complexes comme le XGBoost ([8], [36]), ces faibles performances sembleraient inhérentes aux données modélisées et difficilement améliorables.

Bien que ces résultats soient très décevants en terme de performance, l'étude réalisée demeure réutilisable avec d'autres données. En optant pour la même méthodologie sur une nouvelle base, avec des variables explicatives éventuellement plus corrélées à la variable

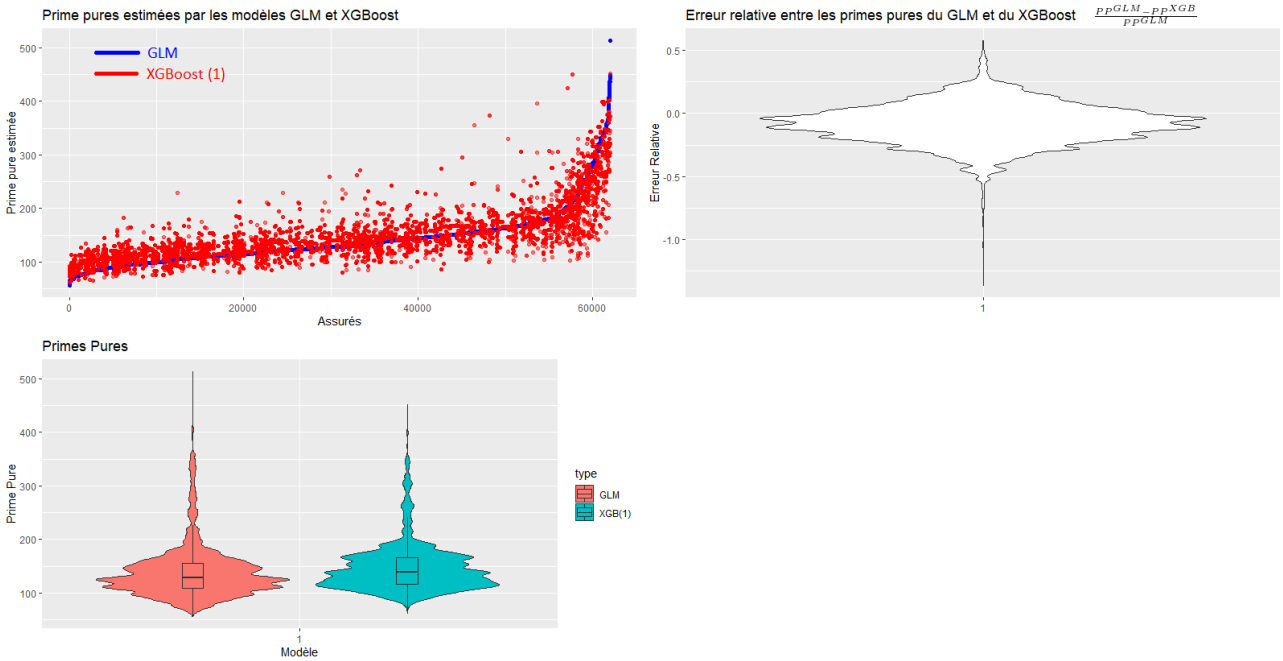

FIGURE *5.18: Analyses de la distribution des primes pures estimées par les deux modèles GLM et XGBoost (1)*

cible, de meilleures performances sont attendues. En particulier, la variable associée au bonus-malus serait intéressante.

De plus, les faibles performances ne changent pas le propos du mémoire, qui veut montrer qu'un modèle aussi complexe soit-il peut s'interpréter à l'aide des outils développés dans le chapitre 4, ce qui sera l'objet de la partie suivante.

Autres outils d'aide à la décision

Nous voulons également mettre l'accent sur un deuxième point. Les modèles d'apprentissage statistique sont généralement jugés sur un unique critère qui est supposé mesurer sa performance ; il s'agit dans notre étude du MSE, pour des raisons justifiées précédemment. Cependant, cette unique valeur est-elle suffisante pour juger de la qualité d'un modèle ? En particulier, le fait d'avoir un gain en MSE minime, proche de 0.001%, suffit-il pour conclure que le modèle XGBoost n'est pas satisfaisant pour notre étude ? Nous n'avons pas la réponse à cette question, mais nous proposons d'autres critères qui nous semblent pertinents ici pour comparer les différents modèles. Nous introduisons la notion de MSE, dit locaux, c'est-à-dire calculés non pas sur la base de test entière mais sur certaines assurés vérifiant des critères définis. Cela permettra de voir pour quels types d'assurés les modèles sont les plus (ou moins) performants, pouvant ainsi servir d'aide à la décision pour le choix du modèle. Nous avons retenu les critères suivants :

— Les assurés n'ayant pas eu de sinistres
— Les assurés ayant eu au moins un sinistre

— Les assurés ayant eu au moins un sinistre et dont le montant total n'excède pas le seuil d'écrêtement de 10000 . Les assurés ayant eu au moins un sinistre et dont le montant total dépasse le seuil d'écrêtement de 10000

On observe que le modèle GLM est plus précis (en terme de MSE) sur les assurés n'ayant pas eu de sinistres. Étant donné que cela représente plus de 96 % de la base, cela peut expliquer pourquoi il est difficile pour le XGBoost de surpasser le GLM. Cependant, on remarque que sur les assurés ayant au moins eu un sinistre, la modélisation par le XGBoost est plus précise en terme de MSE. Il semble que les assurés sinistrés soient les plus importants pour un assureur, étant donné qu'ils représentent le plus grand risque. Ainsi, on peut se demander si il n'est pas préférable de se tromper légèrement plus sur les assurés n'ayant pas de sinistres, et d'être plus précis sur ceux ayant eu un sinistre. Dans le cas où cette remarque est prise en compte dans la décision de l'assureur, le XGBoost pourrait sembler plus pertinent à utiliser.

Condition	Nombre d'assurés concernés (Bt)	RMSE - GLM trivial	RMSE - GLM Best	RMSE - XGBoost
ClaimAmount = 0	59654 (96.3%)	**136.3**	148	153.9
ClaimAmount>0	2322 (3.7%)	18922	18918.4	**18917.1**
ClaimAmount >0 et ClaimAmount <10K	2278 (3.7%)	2652.3	2635.2	**2621.4**
ClaimAmount >10K	44 (0.071%)	86511.6	86502.75	**86500.75**

TABLE *5.17: RMSE locaux calculés sur la base de test pour comparer les modèles GLM et XGBoost qui modélisent la prime pure*

On peut également analyser les biais "locaux", c'est-à-dire la différence en moyenne entre la valeur théorique et la valeur prédite par le modèle considéré, sur un groupe d'individus défini. Nous reprenons les mêmes conditions que celles introduites précédemment. Pour un assureur, il semble légitime de penser que sur-estimer les sinistres d'un assuré est moins grave que les sous-estimer. Bien que dans un milieu concurrentiel, on modélise souvent que l'assuré va chez l'assureur le moins cher, le fait de sous-estimer la sinistralité peut avoir un impact énorme sur la santé financière de la compagnie. Ainsi le biais entre la valeur théorique et la valeur prédite sera considéré meilleur lorsqu'il sera proche de zéro ou négatif.

Condition	Nombre d'assurés	Prédiction Moyenne Théorique	Prédiction Moyenne GLM trivial	Prédiction Moyenne GLM Best	Prédiction Moyenne XGBoost
Aucune	61976 (100%)	152.47	136.27	139.5	**147.37**
ClaimAmount = 0	59654 (96.3%)	0	**136.274**	139.14	146.97
ClaimAmount>0	2322 (3.7%)	4069.53	136.274	148.435	**157.75**
ClaimAmount >0 et ClaimAmount <10K	2278 (3.7%)	2067	136.274	148.1	**157.71**
ClaimAmount >10K	44 (0.071%)	44730	86511.6	**155.88**	158.60

TABLE *5.18: Moyenne des prédictions sur différentes classes d'assurés de la base de test, pour les modèles GLM et XGBoost*

Mise en place d'un modèle concurrentiel entre nos deux modèles

Dans ce paragraphe, nous nous plaçons dans un contexte de concurrence entre deux assureurs. On suppose qu'ils sont seuls sur le marché et que l'un utilise le modèle GLM défini ci-avant pour tarifier l'assurance automobile et l'autre utilise le modèle XGBoost. On suppose que la prime commerciale correspond à la prime pure, c'est-à-dire que l'on ignore la marge mise en place par chaque assureur ainsi que le chargement pour les frais administratifs et de gestion, les taxes et le chargement de sécurité. Il s'agit d'un modèle très simplifié qui a pour objectif de montrer que dans un marché concurrentiel, le bénéfice réalisé par les assureurs est très sensible aux prix donnés et donc au modèle utilisé. Nous devons alors définir un critère pour modéliser le choix de l'assureur chez lequel l'assuré va aller. Comme nous l'avons souligné précédemment, une hypothèse souvent réalisée est que l'assuré va chez l'assureur le moins cher. Il s'agit bien évidemment d'une hypothèse très forte qui n'est pas vérifiée dans la pratique. D'autres hypothèses aurait pu être considérées comme par exemple, la moitié des assurés va aléatoirement chez les assureurs 1 et 2, et l'autre moitié va chez le moins onéreux.

Une fois ce marché concurrentiel mis en place, il est alors possible de calculer de nombreuses informations, comme le chiffre d'affaires, les bénéfices, le *loss ratio* etc.

Tout d'abord la répartition des assurés est donnée sur la figure 5.19.

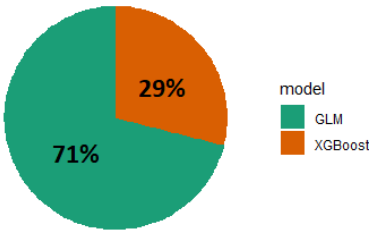

FIGURE *5.19: Part de marché de chaque assureur dans le cas d'un marché concurrentiel très simplifié. L'assureur 1 utilise un modèle GLM et l'assureur 2 utilise un XGBoost pour modéliser la prime pure. Les assurés vont chez l'assureur le moins onéreux*

Assureur	Chiffre d'affaires (Primes)	Sinistres (S)	Résultat (=S-P)	Loss Ratio (S/P)
1 (GLM)	5.65M	6.45M	-800K	1.14
2 (XGBoost)	2.67M	3.0M	-329K	1.12

TABLE *5.19: Analyse du chiffre d'affaire, du résultat et du Loss Ratio des deux assureurs (GLM et XGBoost) dans le cas d'un marché concurrentiel très simplifié*

On remarque tout d'abord que l'assureur 1, utilisant le GLM, a une part de marché bien plus élevée (environ 71%). Cela vient du fait que le GLM sous-tarifie de manière générale, comme nous l'avons vu dans les tableaux précédents, et récupère donc plus d'assurés. Comme nous l'avions souligné, le fait d'avoir une part de marché élevée ne signifie pas que le résultat de l'entreprise est plus important. Ce phénomène est bien constaté dans le tableau 5.19. Bien que le chiffre d'affaires de l'assureur 1 soit deux fois

plus important, son résultat est nettement inférieur à celui de l'assureur 2. De plus, le facteur qui est généralement étudié est le *loss ratio* (aussi noté S/P), qui correspond au ratio des sinistres des assurés divisés par le montant total des primes. L'objectif pour un assureur est bien évidemment de le minimiser et surtout qu'il soit inférieur à 1. Pour les deux assureurs, celui-ci est supérieur à 1 signifiant que le résultat des deux entreprises est négatif. Cela peut venir du fait que seule la prime pure a été prise en compte et qu'en ajoutant les marges et les chargements de sécurité, les résultats devraient être sensiblement différents. On remarque que le S/P est inférieur pour l'assureur utilisant le XGBoost, signifiant que dans le jeu de concurrence avec l'assureur 1, il est globalement gagnant.

Cet exemple très simplifié n'a pas vocation à être détaillé plus que cela. Il permet juste de montrer de manière très schématique que le choix du modèle peut avoir un impact important pour l'assureur et que modéliser la sinistralité le plus finement possible peut s'avérer déterminant.

5.6 Interprétabilité des deux modèles

Nous avons désormais mis deux modèles en place : un GLM et un XGBoost et nous avons conclu que ce dernier est très légèrement plus performant par rapport aux critères que nous avons retenu. Notre idée à présent est d'interpréter les deux modèles. Dans le cas du XGBoost, il s'agit d'un modèle de boîte noire à part entière et nos seuls moyens de l'interpréter seront les différentes méthodes vues dans le chapitre 4. Étant donné que le modèle GLM mis en place est relativement parcimonieux (avec 7 variables et 20 coefficients) et qu'il se situe dans la classe des algorithmes modulaires et simulables (c.f chapitre 2), il sera relativement facile à interpréter. Rappelons que cela signifie que l'on est capable de comprendre la prise de décision, les poids, les paramètres et la structure globale du modèle. Les méthodes du chapitre 4 ne sont donc nécessaires mais pourront également être utilisées afin de vérifier la cohérence des résultats.

5.6.1 Interprétation du GLM fréquence

5.6.1.1 Interprétation intrinsèque du modèle GLM fréquence

Intéressons-nous tout d'abord au modèle GLM. Essayons de comprendre comment l'algorithme aboutit à une telle prédiction. Prenons un assuré en particulier qui par exemple possède les caractéristiques du tableau 5.6.1.1.

A l'aide des différents coefficients du modèle GLM fréquence mis en place (résumés dans le tableau précédent), il est facile de comprendre la prédiction faite par le modèle. En effet, elle se décompose comme une somme des coefficients associés à chaque variable à laquelle on applique la fonction de lien. Ainsi, notant $link$ la fonction de lien (ici log) et $\beta_{intercept}$, β_{CarAge}, $\beta_{DriverAge}$, β_{Brand}, β_{Gas}, β_{Region} et $\beta_{Density}$ les différents coefficients

de cet assuré, nous avons la prédiction suivante faite par le GLM :

$$\hat{y}_1 = link^{-1}(\beta_{intercept} + \beta_{CarAge} + \beta_{DriverAge} + \beta_{Brand} + \beta_{Gas} + \beta_{Region} + \beta_{Density}) \tag{5.6}$$

$$= exp(-2.310407 + 0.095596 + 0.180802 - 0.618698 + 0 + 0 - 0.127440 + 0.268020) \tag{5.7}$$

$$= exp(-2.512127) = 0.08109555 \tag{5.8}$$

En omettant la fonction de lien inverse, qui est ici la fonction croissante exponentielle, on peut interpréter la prédiction du modèle de la manière suivante : l'assuré commence avec un coefficient initial de -2.310407 (l'intercept) qui correspond à la valeur prédite par le GLM si l'assuré se situait dans la modalité de référence pour chaque variable, c'est-à-dire si on avait un assuré avec Power=1, CarAge=0, DriverAge=18-26, Brand= !F, Gas=Diesel, Region=R11, Density=0-100. A ce coefficient on ajoute 0.095596 (car l'assuré a une voiture de puissance 2 et non 1), on ajoute 0.180802 (car l'assuré a une voiture âgée de 1 à 3 ans et non de 0), on retranche 0.618698 (car l'assuré est âgé entre 43 et 55 ans et non entre 18 et 26), on ajoute 0 (car la marque est de catégorie " !F" et *Gas* de type "Diesel"), on retranche 0.127440 (car l'assuré vit dans une des régions de la catégorie R23-31-72-54 et non dans la région R11) et enfin on ajoute 0.268020 (car la densité de la ville où habite l'assuré est entre 101 et 19999 et non inférieure à 100).

$$\hat{y}_1 = \underbrace{e^{-2.310407}}_{intercept} \times \underbrace{e^{0.095596}}_{Power=2} \times \underbrace{e^{0.180802}}_{CarAge=1-3} \times \underbrace{e^{-0.618698}}_{DriverAge=43-55} \times \underbrace{e^{0}}_{Brand=!F} \times \underbrace{e^{0}}_{Gas=Diesel} \times \underbrace{e^{-0.127440}}_{Region=R23-31-72-54} \times \underbrace{e^{0.268020}}_{Density=101-19999}$$

On peut alors facilement comprendre comment change la prédiction si une (ou plusieurs) des variables est modifiée. Considérons un deuxième assuré qui a les mêmes caractéristiques que l'assuré précédent, à la seule différence que la puissance du véhicule est de catégorie 3 et non 2. Le coefficient associé à la variable *Power* est alors 0.246528 et non plus 0.095596. Nous avons représenté dans le tableau 5.21 les prédictions faites par nos différents modèles mis en place pour ses deux assurés. Notons que le XGBoost 1 réfère au modèle ajusté avec les variables retraitées, alors que pour le XGBoost 2, les variables numériques (*Density*, *CarAge* et *DriverAge*) ont été inchangées. Aucune valeur n'a été donnée pour le XGBoost 2 sur le modèle de sévérité car aucun ne fournissait de meilleurs résultats que le GLM classique.

On peut par le même raisonnement établir que la prédiction du GLM fréquence pour l'assuré 2 est la suivante :

$$\hat{y}_2 = \underbrace{e^{-2.310407}}_{intercept} \times \underbrace{e^{0.246528}}_{Power=3} \times \underbrace{e^{0.180802}}_{CarAge=1-3} \times \underbrace{e^{-0.618698}}_{DriverAge=43-55} \times \underbrace{e^{0}}_{Brand=!F} \times \underbrace{e^{0}}_{Gas=Diesel} \times \underbrace{e^{-0.127440}}_{Region=R23-31-72-54} \times \underbrace{e^{0.268020}}_{Density=101-19999}$$

	Prédictions	Assuré 1	Assuré 2
Meilleur GLM	Fréquence (=*ClaimNb*/Exposure)	0.08109555	0.09430749
	Coût (=ClaimAmount/*ClaimNb*)	1506.905	1602.303
	Total (=ClaimAmount/Exposure)	122.2033	151.1091
XGBoost 1	Fréquence (=*ClaimNb*/Exposure)	0.07695238	0.08792618
	Coût (=ClaimAmount/*ClaimNb*)	1546.624	1662.131
	Total (=ClaimAmount/Exposure)	119.0164	146.1448
XGBoost 2	Fréquence (=*ClaimNb*/Exposure)	0.08173109	0.0854457
	Coût (=ClaimAmount/*ClaimNb*)	/	/
	Total (=ClaimAmount/Exposure)	/	/

TABLE *5.21: Prédictions réalisées par les différents modèles ajustés sur les assurés 1 et 2 étudiés*

On peut également raisonner différemment, en repartant de la prédiction faite pour le premier assuré. La seule différence est que la catégorie de puissance a été modifiée, engendrant un changement de coefficient de $0.246528 - 0.095596 = 0.150932$. La prédiction $\hat{y}_2$ est alors facile à exprimer en fonction de $\hat{y}_1$:

$$\hat{y}_2 = \hat{y}_1 \times e^{\beta_{\{Power=3\}} - \beta_{\{Power=2\}}} = \hat{y}_1 \times e^{0.150932} = 0.08109555 \times e^{0.150932} = 0.09430749$$

A travers ces deux exemples, on souligne la facilité d'interprétation d'un modèle GLM parcimonieux. Nous nous sommes intéressés particulièrement aux prédictions faites sur la fréquence, mais le raisonnement est identique pour la sévérité.

Dans le cas du XGBoost, on ne peut pas directement comprendre le cheminement qui a conduit à une telle prédiction pour ces deux assurés, d'où le terme de boîte-noire couramment employé. Nous avons alors recours aux différentes méthodes vues dans le chapitre 4.

5.6.1.2 Vérification de la cohérence avec les méthodes d'interprétation du chapitre 4

Avant d'interpréter le modèle XGBoost, nous allons vérifier que les outils vus dans le chapitre 4 sont cohérents avec l'interprétation réalisée ci-dessus.

Analyse globale

Nous commençons par une analyse globale du modèle GLM, avant de passer à une étude locale, au cours de laquelle nous verrons que les résultats sont identiques avec la première analyse.

Importance des variables

Une des premières analyses que l'on souhaite réaliser une fois notre modèle ajusté est

l'importance des variables, afin de comprendre quelles variables sont les plus utilisées par le modèle et celles qui influent le plus, de manière globale, sur les prédictions.

Nous voulons vérifier que les méthodes vues dans le chapitre 4 pour mesurer cette importance des variables sont conformes avec les résultats fournis par la t-statistique. Rappelons que dans le cas du GLM, on mesure l'importance d'une variable $j \in \{1, ..., p\}$ en calculant la t-statistique définie par : $t_j = \frac{|\beta_j|}{\sigma_j}$ où β_j est le coefficient de la variable j et σ_j l'écart-type associé (c.f chapitre 3) Pour notre meilleur GLM fréquence, les résultats sont indiqués dans le tableau 5.3 dans la colonne *z value*. Dans le chapitre 4, nous avons étudié plusieurs techniques indépendantes du modèle pour estimer l'importance d'une variable. En particulier, nous avons retenu la méthode *PFI* (Permutation Feature Importance) qui repose sur le fait que : si la modification de la valeur d'une variable augmente drastiquement l'erreur de prédiction, alors cette variable est considérée importante, car le modèle compte beaucoup sur elle. Nous voulons vérifier que les résultats fournis par cette méthode sont conformes à la t-statistique. Les résultats obtenus sont donnés sur la figure 5.20. Nous observons une réelle similitude entre les deux graphiques de coefficients. Les légères différences observées proviennent certainement des erreurs dues au nombre de permutations utilisées dans la méthode PFI, et à l'incertitude qui en découle.

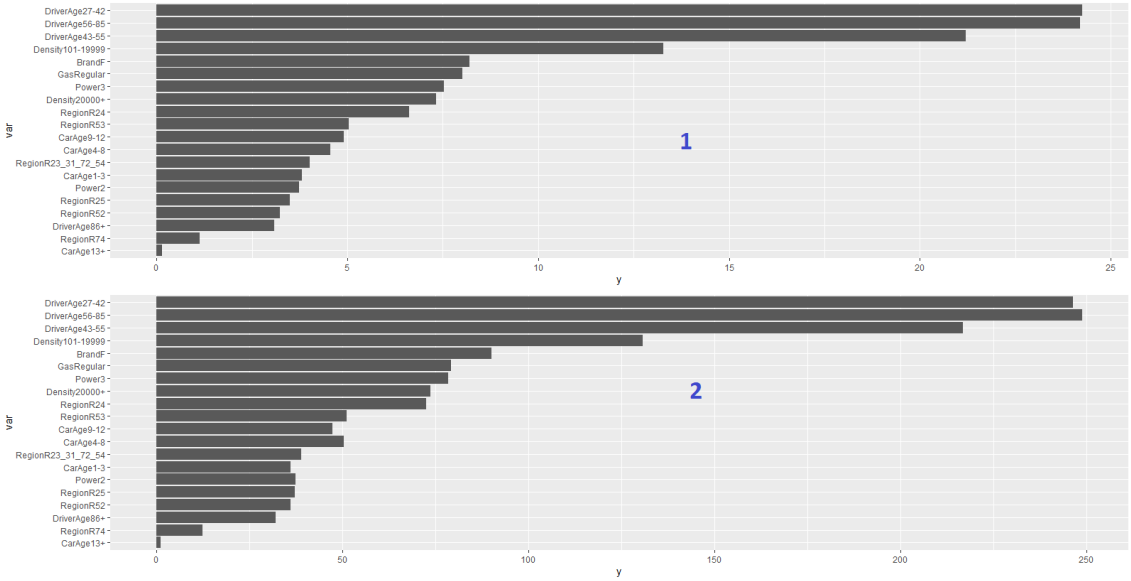

FIGURE *5.20: Importance des variables dans le GLM fréquence, basée sur la t-statistique (en haut) et sur la méthode PFI (en bas)*

Graphique de dépendance partielle

Analysons à présent le graphique de dépendance partielle, noté PDP tout au long du mémoire. Rappelons que celui-ci permet de mesurer l'effet marginal moyen d'une (ou de plusieurs) variable(s) sur la prédiction du modèle. Dans le cas où les variables ne sont pas (ou très peu) corrélées, le graphique de dépendance partielle traduit exactement la

relation en moyenne entre une variable et la sortie du modèle. Ici, nous devrions observer une certaine cohérence avec les coefficients du GLM. Vérifions cela graphiquement sur les figures 5.21 et 5.23, qui représentent respectivement les graphiques de dépendance partielle et les coefficients du GLM. Les courbes obtenues sont identiques, à un changement d'échelle près : cela est bien cohérent avec la théorie.

Courbes ICE

On peut également étudier les courbes ICE (courbes d'espérance conditionnelle individuelles), qui permettent d'avoir une vision au niveau d'une instance et non globale du modèle. Dans le cas du modèle GLM fréquence mis en place, il y a une absence d'interaction entre les variables et dont aucun effet hétérogène ne devrait être présent sur les courbes ICE, comme nous l'avons vu dans le chapitre 4. Cela est bien observé sur la figure 5.22, sur laquelle chaque courbe est translatée de l'autre, et également translatée du graphique de dépendance partielle, qui est une moyenne de toutes les courbes ICE.

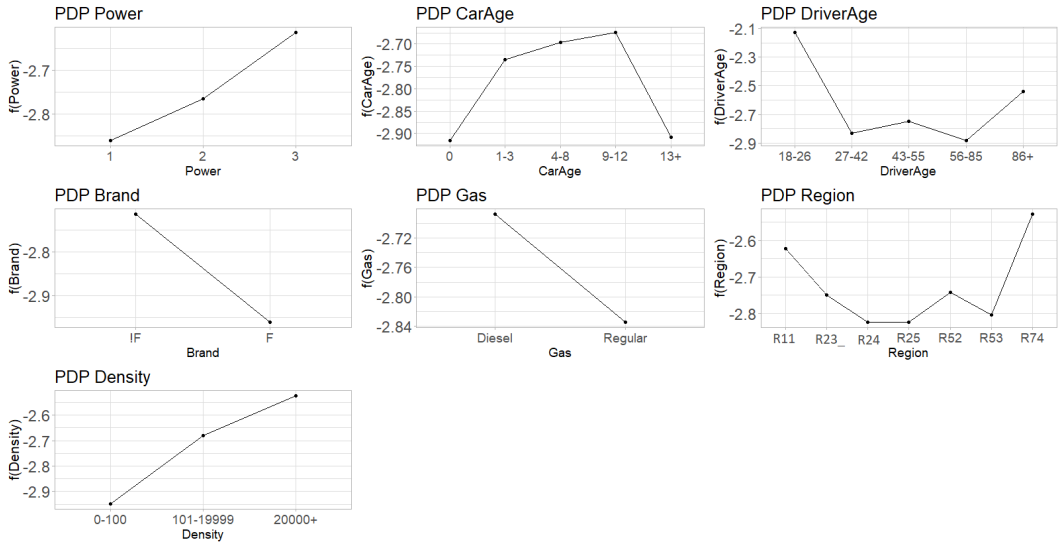

FIGURE *5.21: Graphiques de dépendance partielle (PDP) des 7 variables du GLM fréquence*

ALE

Nous pouvons réaliser la même étude avec les graphiques d'effets locaux accumulés (ALE). Étant donné que les variables sont très peu corrélées dans notre étude, les graphiques ALE devraient être sensiblement identiques aux graphiques de PDP. Ceci est bien vérifié sur les graphiques de la figure 5.24.

Les graphiques précédents, de PDP, d'ALE et des coefficients du modèle GLM, nous expliquent pourquoi la prédiction pour l'assuré 2 est plus élevée que pour l'assuré 1. En effet, ils montrent que lorsque la variable *Power* est de catégorie 2, la fréquence de sinistralité moyenne est plus faible que lorsqu'elle est de catégorie 3. Cette première étude repose sur une analyse globale du modèle. Nous pouvons à présent étudier le comportement local des deux assurés. Étant donné la structure du modèle GLM, le comportement global est le même que le comportement local. C'est ce que nous allons

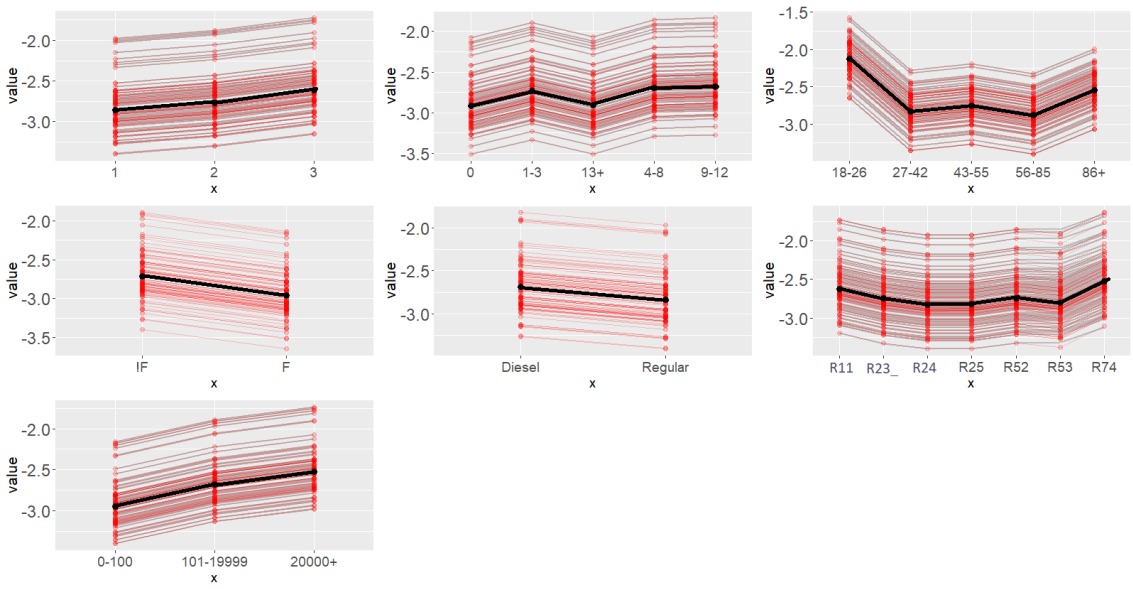

FIGURE *5.22: Quelques courbes ICES des 7 variables du GLM fréquence, avec en noir la courbe de dépendance partielle (PDP)*

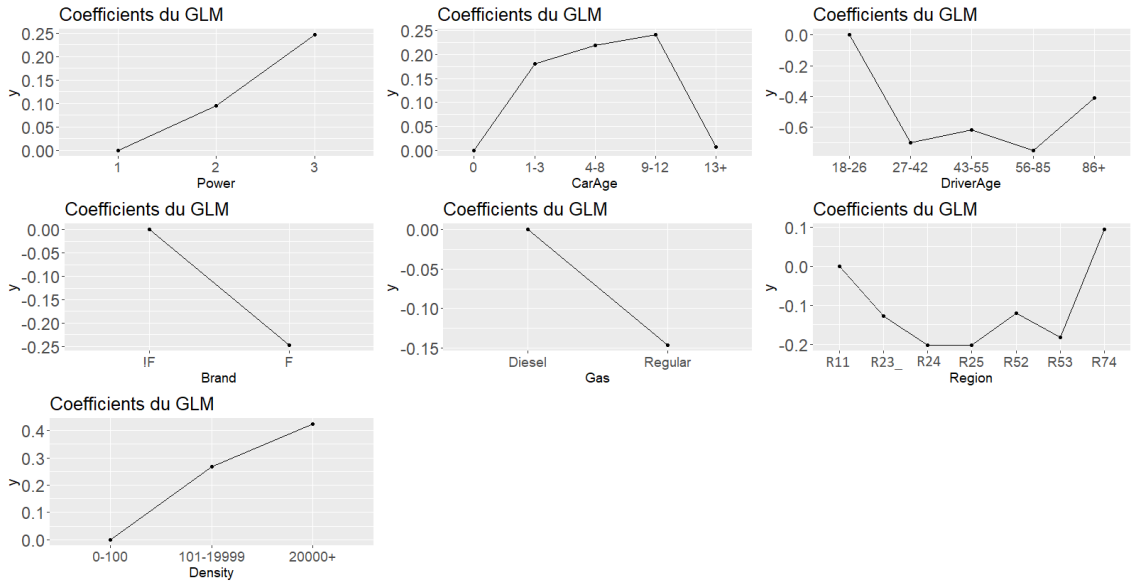

FIGURE *5.23: Coefficients des différentes variables utilisées dans le GLM fréquence*

vérifier avec le paragraphe suivant.

Analyse locale

Dans cette partie, nous nous intéressons au comportement local du modèle GLM fré-

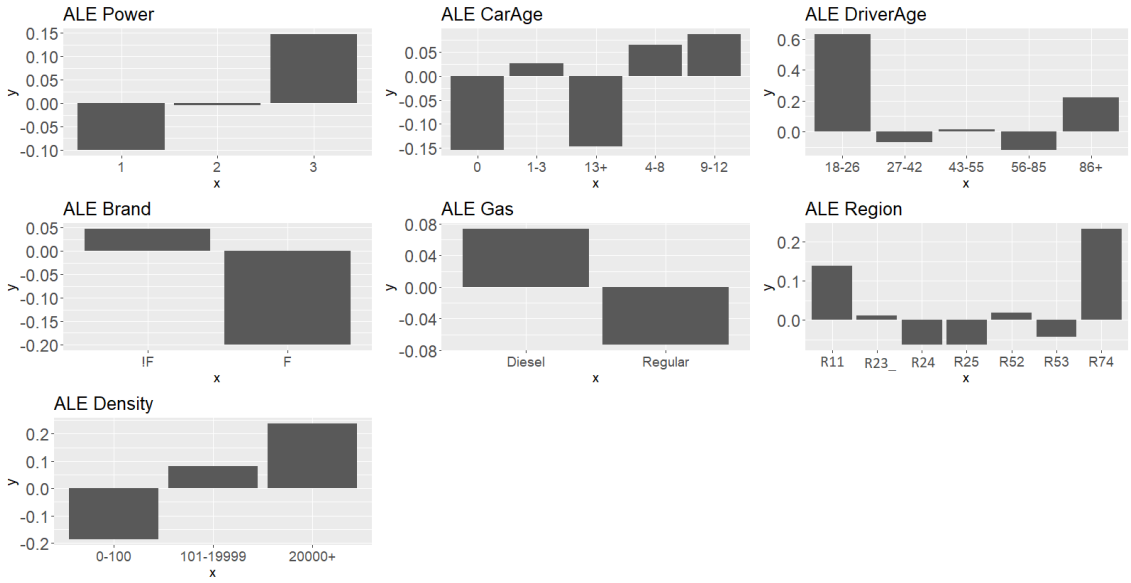

FIGURE *5.24: Graphiques d'effets locaux accumulés (ALE) des 7 variables du GLM fréquence*

quence. Rappelons qu'une interprétation locale réfère à une interprétation au niveau d'une instance particulière. Nous prenons l'exemple des assurés 1 et 2, déjà étudié auparavant. Les outils développés dans le chapitre 4 qui vont nous être utiles sont les méthodes LIME et SHAP, ainsi que leurs variantes (Live, BreakDown etc.).

Méthodes LIME et LIVE

Commençons par la méthode LIME, pour laquelle les résultats obtenus sont donnés sur la figure 5.25. Comme indiqué précédemment, le comportement local des deux assurés est le même lorsqu'ils ont les mêmes caractéristiques. Ici, leur seule différence est la variable *Power*, qui vaut 2 pour l'assuré 1 et 3 pour l'assuré 2. Nous remarquons que pour l'assuré 2, l'effet de la variable *Power* est bien supérieur que pour l'assuré 1, et tous les autres coefficients sont inchangés, ce qui explique cette prédiction plus élevée ($\hat{y}_1 = 0.08109555$ et $\hat{y}_2 = 0.09430749$). Il est important de remarquer que les valeurs inscrites sur le graphique au nom de "Actual prediction" correspondent aux valeurs des prédictions avant d'avoir appliqué la fonction de lien inverse du GLM, à savoir la fonction exponentielle ici. On retrouve bien : $\hat{y}_1 = e^{-2.51} = 0.0811$ et $\hat{y}_2 = e^{-2.36} = 0.0943$.

Méthodes SHAP et BreakDown

Analysons à présent les résultats obtenus à l'aide des méthodes SHAP et breakDown. Celles-ci donnent la contribution de chaque variable dans la prédiction faite par le modèle considéré. Nous remarquons qu'au sein d'une même méthode, les contributions sont identiques pour chaque variable, sauf *Power* qui diffère pour les deux assurés (c.f figures 5.26 et 5.28). Plus précisément, lorsque *Power* vaut 2 pour l'assuré 1, il s'agit de la variable qui contribue le moins dans la prédiction du modèle alors que lorsque *Power* vaut 3 pour l'assuré 2, il s'agit de celle qui contribue le plus, ce qui justifie l'écart entre les deux prédictions. Nous analysons également la stabilité des interprétations fournies par

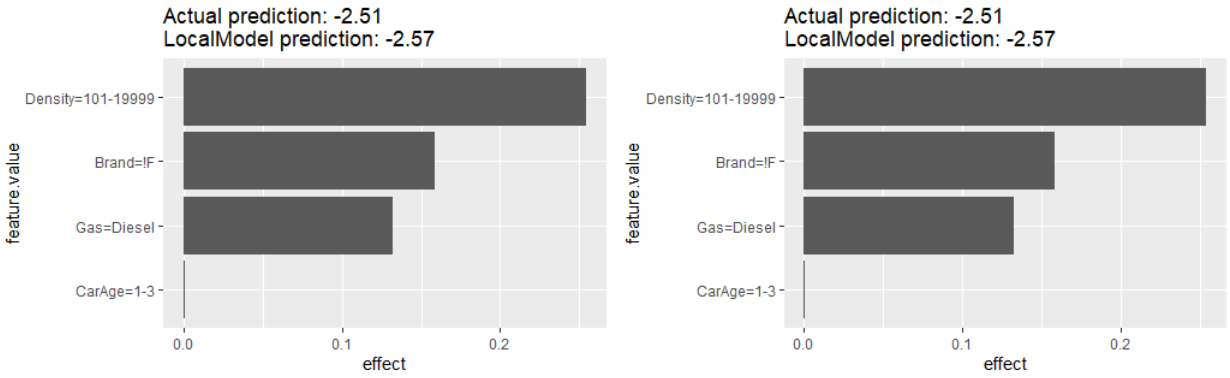

FIGURE *5.25: Résultats fournis par LIME pour les assurés 1 et 2 dans le modèle GLM fréquence, avec 4 variables retenues dans la régression LASSO*

les méthodes présentées ci-dessus. Pour ce faire, nous les appliquons un grand nombre de fois et on affiche la boîte à moustaches des valeurs obtenues pour chaque coefficient. Nous présentons sur la figure 5.27 l'analyse réalisée avec la méthode SHAP, pour laquelle des échantillons de taille 100 ont été utilisés ainsi que 100 simulations. On observe une certaine stabilité, bien que la taille des échantillons soit très faible, pour éviter un temps de calcul trop important.

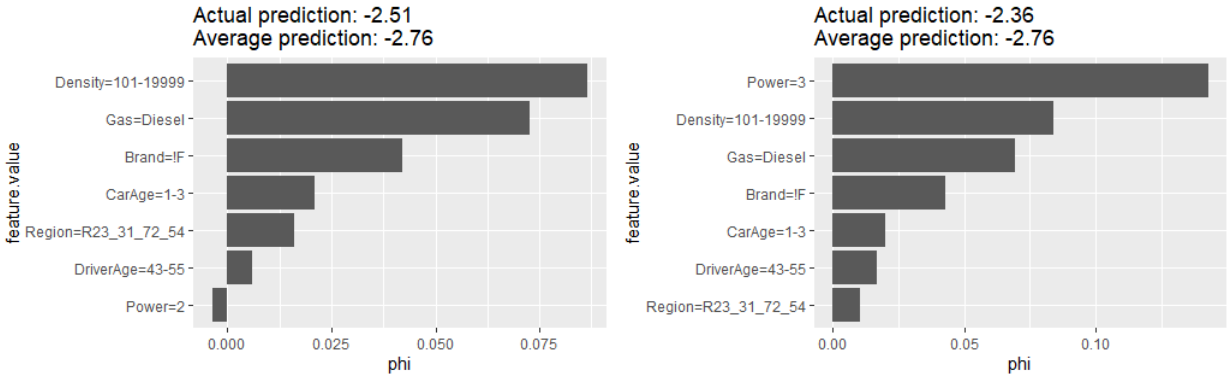

FIGURE *5.26: Contributions de chaque variable dans la prédiction du modèle GLM fréquence pour les assurés 1 et 2 à l'aide de la méthode SHAP*

Finalement, les analyses locales et globales du GLM sont conformes à la théorie et reflètent bien l'impact des variables sur la prédiction tout comme les coefficients du GLM.

5.6.2 Interprétation du XGBoost fréquence

Nous savons désormais parfaitement comprendre les prédictions réalisées par notre GLM fréquence, que ce soit directement à l'aide des coefficients du modèle, ou grâce

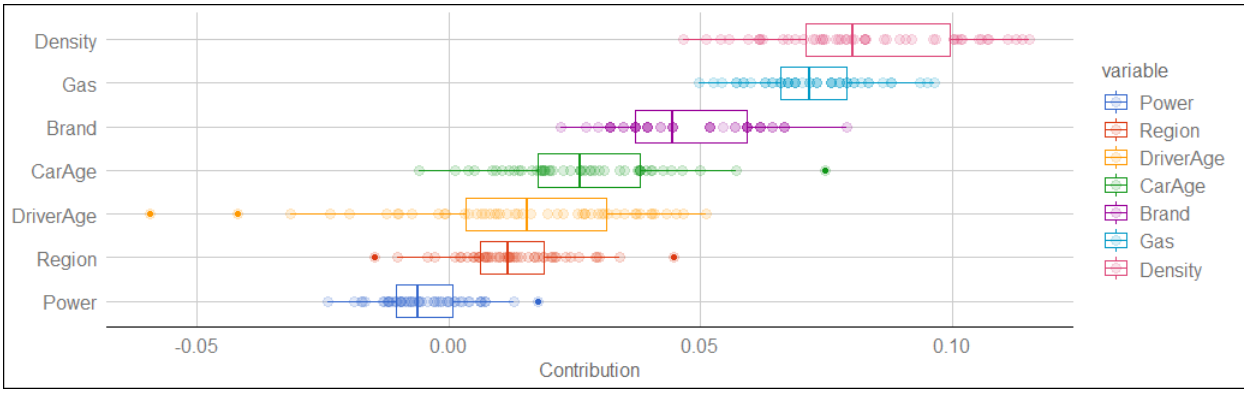

FIGURE *5.27: Stabilité des contributions de chaque variable dans la prédiction du modèle GLM fréquence pour l'assuré 1 avec la méthode SHAP*

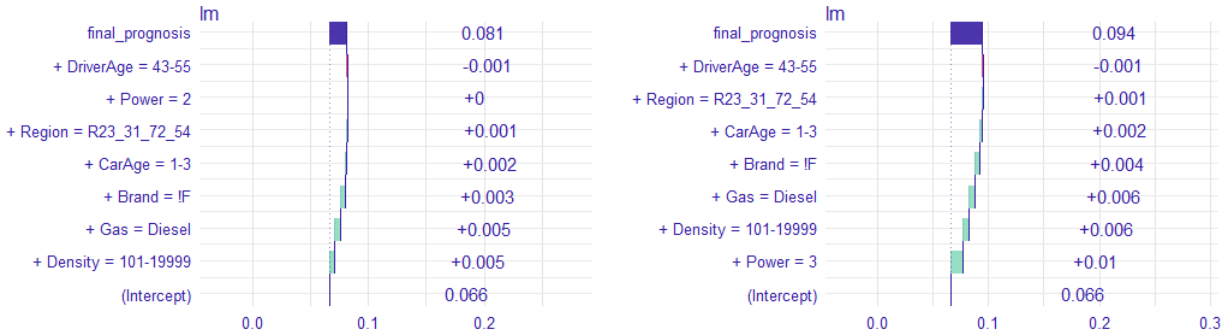

FIGURE *5.28: Contributions de chaque variable dans la prédiction du modèle GLM fréquence pour les assurés 1 et 2 à l'aide de la méthode Breakdown (direction "up")*

aux différentes méthodes d'interprétations vues dans le chapitre 4. Les résultats sont similaires et très facilement compréhensibles, notamment pour expliquer le changement de prédiction entre deux assurés. Essayons à présent de réaliser la même étude pour le modèle XGBoost implémenté. Contrairement au GLM, cette famille d'algorithmes ne bénéficie pas des propriétés de modularité ou de simulabilité vues dans le chapitre 2 et ne pouvons donc comprendre son comportement directement à partir de coefficients définissant le modèle. L'utilisation des outils d'interprétation développés dans ce mémoire sont alors inévitables.

Commençons par essayer d'expliquer le comportement global du modèle.

5.6.2.1 Analyse globale

Importance des variables

Une première étude qui est généralement réalisée lorsque l'on souhaite interpréter un modèle est l'importance des variables. Cette étape permet de comprendre les variables qui sont indispensables pour prédire la variable cible et éventuellement de retirer celles

qui ne le sont pas, afin de rendre le modèle plus compact, ou parcimonieux. Dans la majorité des travaux d'apprentissage statistique, les méthodes employées pour estimer l'importance des variables étaient intrinsèques au modèle considéré. Par exemple, pour le GLM la t-statistique, détaillée ci-avant, est utilisée alors que pour la méthode CART (c.f chapitre 3), on calcule une somme des diminutions de déviance provoquées à chaque noeud de l'arbre, si on remplaçait la division optimale par une division de substitution. Cette dernière repose également sur la structure interne de l'algorithme utilisé. Grâce au chapitre 4 nous sommes désormais en mesure de généraliser ce calcul d'importance des variables à n'importe quelle famille d'algorithmes et en particulier pour le XGBoost implémenté. Nous avons vu plusieurs techniques différentes au cours de ce chapitre, celle que nous retenons ici est PFI (Permutation Feature Importance). Les résultats sont affichés sur la figure 5.29. Commençons par interpréter les résultats obtenus pour les modèles de fréquence. On observe que les variables ont quasiment toutes le même ordre d'importance que ce soit pour le GLM ou le XGBoost. On observe néanmoins une inversion d'importance entre *Brand* et *CarAge* qui peut venir soit du fait que les modèles n'utilisent pas les variables de la même manière et n'ont donc pas nécessairement la même importance, soit du fait que cette mesure est parfois instable pour les variables "intermédiaires", comme nous l'avions observé dans le chapitre 3 pour les arbres de décision. La deuxième remarque que l'on peut faire concernant le XGBoost fréquence est qu'une certaine stabilité des résultats est présente sur la figure en haut à droite. En effet, on distingue clairement trois groupes de variables : le premier, composé de DriverAge domine largement tous les autres et sa présence au sein du modèle semble primordiale. Un deuxième groupe, composé de *CarAge*, *Density* et *Brand*, qui reste également essentiel au modèle, mais dans une moindre mesure en comparaison du groupe précédent. Enfin, les variables *Power*, *Region* et *Gas* semblent être moins importantes dans les prédictions faites par le XGBoost fréquence. Globalement, cette stabilité tend à faire confiance aux résultats fournis. Notons cependant que l'analyse réalisée ici est globale et donc ne donne qu'un aperçu général de l'intérêt de chaque variable dans le modèle. Au niveau d'une observation particulière, des variables jugées peu importantes par cette méthode peuvent être finalement essentielles. Ceci pourra être observé à l'aide des méthodes locales, comme LIME ou SHAP par exemple.

Etudions à présent les modèles de sévérité. Notons tout d'abord que contrairement à la modélisation de la fréquence, l'ordre d'importance des variables est totalement différent pour les deux algorithmes. Cette différence s'explique tout d'abord par le fait que la variable *Region* qui est la plus importante pour le XGBoost n'est pas utilisée par le GLM. En effet, après avoir réalisé une sélection de variables nous avons conclu que retirer la variable *Region* augmentait les performances du modèle, en terme d'AIC et de BIC. Le fait de ne pas utiliser les mêmes variables peut bouleverser l'ordre d'importance, car certaines peu importantes de base peuvent devenir essentielles en compensant l'effet de la variable retirée. Cela semble être par exemple le cas de *Brand*, relativement peu importante pour le XGBoost mais qui est classée première pour le GLM sévérité. Notons également que pour les deux modèles, la variable *Gas* est peu influente sur les prédictions. Enfin, il semble que comme dans le cas de la fréquence, on note une certaine stabilité

Concernant les scores d'importance du XGBoost. Le classement des variables semblent bien établi, et on fait confiance aux résultats. L'ordre (décroissant) semble être le suivant : $Region > CarAge > Power > Density > Brand > DriverAge > Gas$.

Enfin, nous avons représenté sur la figure 5.30 l'importance des variables pour les deux modèles XGBoost fréquence, l'un utilisant les variables retraitées, l'autre ayant gardé les variables numériques inchangées. On remarque que l'importance des variables au sein des deux modèles sont très similaires.

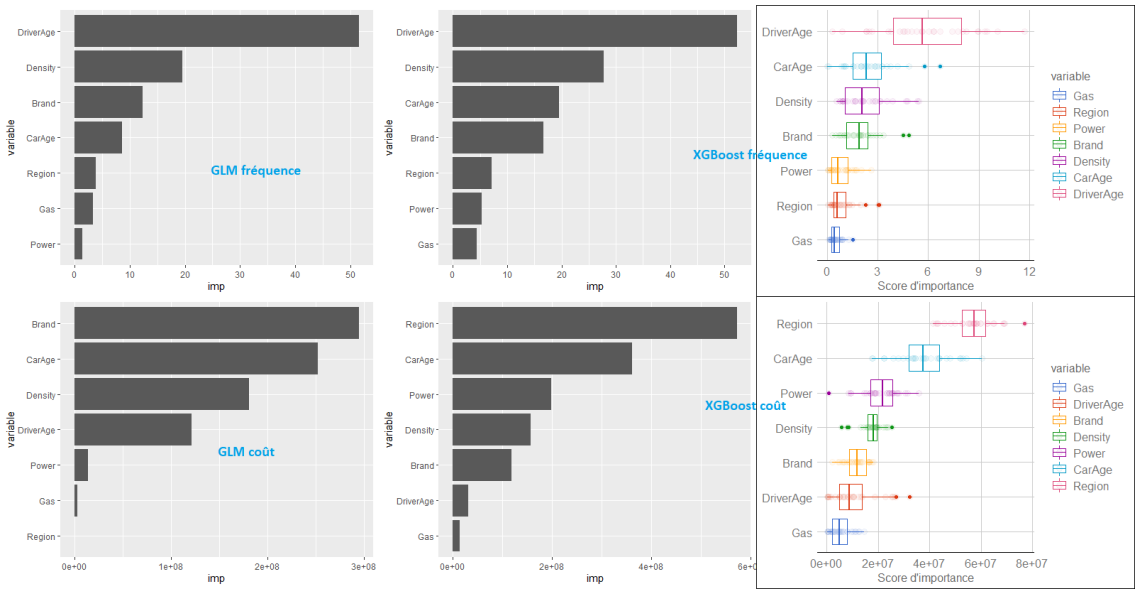

FIGURE *5.29: Importance des variables obtenue par la méthode - agnostique au modèle - PFI, pour les GLM fréquence et sévérité (à gauche) et pour les modèles XGBoost fréquence 1 et sévérité (au milieu). Les deux graphiques de droite correspondent à des boîtes à moustaches obtenues sur plusieurs simulations de calcul d'importance des variables des modèles XGBoost.*

PDP (dépendance partielle) et ALE (effets locaux accumulés)

Analysons à présent le graphique de dépendance partielle (PDP). Nous avons superposé les graphiques de dépendances obtenus pour les deux modèles XGBoost ajustés (c.f figure 5.31), l'un a retraité les variables numériques alors que les autres les a laissé inchangées. C'est pour cela que nous observons sur les PDP associés à *Density*, *CarAge* et *Driver* des paliers pour le modèle 1 alors qu'on observe une courbe "continue" pour le XGBoost 2.

Globalement, l'effet de chaque variable est très similaire dans les deux modèles et l'évolution des courbes est dans le même sens, surtout pour les variables catégorielles (*Brand*, *Power*, *Gas* et *Region*).

On observe néanmoins que sur la courbe du modèle 2, pour la variable *DriverAge* (en haut à droite de la figure 5.31), l'effet de la jeunesse du conducteur sur la sinistralité est très marquée, alors qu'elle est atténuée sur le modèle 1, bien que toujours significative. Cela vient du fait qu'en créant la catégorie des "18-26 ans" dans le modèle 1, la forte

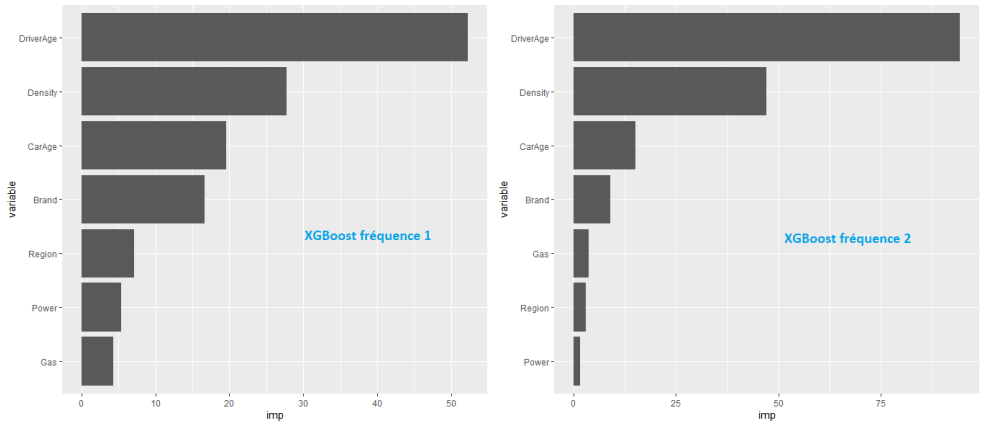

FIGURE *5.30: Importance des variables estimées via la méthode PFI (Permutation Feature Importance) pour le modèle XGBoost fréquence (à gauche) et XGBoost fréquence 2 (à droite)*

décroissance observée entre les âges 18 et 26 ans a été moyennée, formant une seule valeur pour ce groupe.

On remarque également, sur le deuxième graphique de la figure 5.31, que le modèle 2 va estimer une sinistralité plus faible pour les véhicules de plus de 20 ans. Cela vient du fait que peu de données sont disponibles pour les véhicules âgés de plus de 20 ans, à savoir moins de 1.5% de la base. Les résultats affichés sur cette courbe au delà de l'abscisse 20 ne sont pas nécessairement significatifs et doivent être interprétés avec précaution.

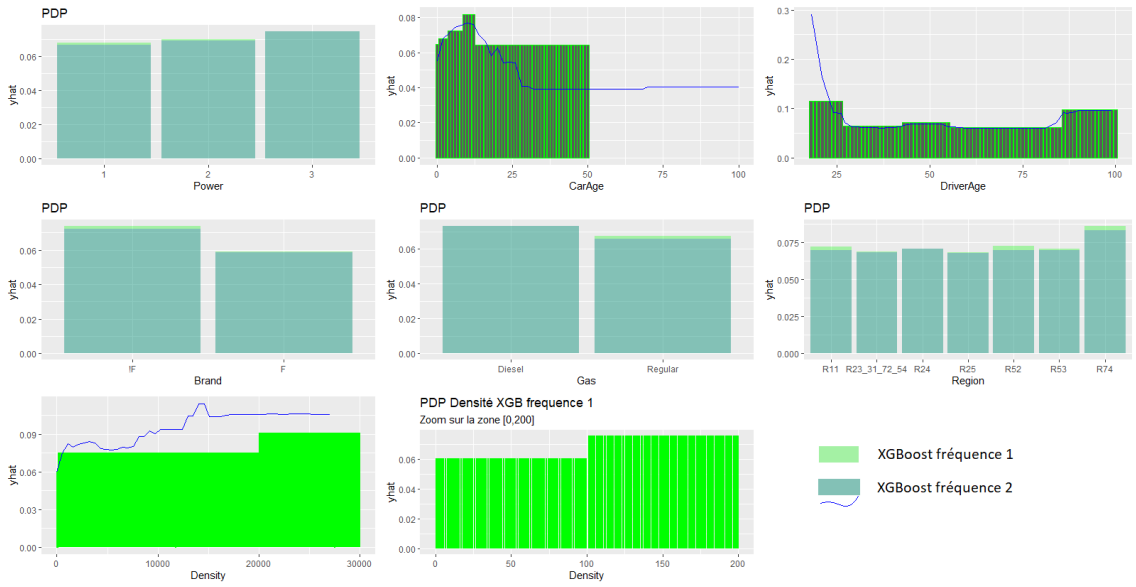

FIGURE *5.31: Graphiques de dépendance partielle (PDP) du modèle XGBoost 1 (après retraitement des variables numériques) et du XGBoost fréquence 2 (sans retraitement des variables numériques)*

Sur la base des deux modèles, nous pouvons faire les remarques suivantes, tout en gardant à l'esprit que corrélation n'est pas synonyme de causalité :

- Power : plus la puissance est élevée plus la sinistralité est élevée (en accord avec la figure 5.7).
- *CarAge* : on observe une croissance de l'impact de l'âge du véhicule sur la sinistralité, à l'exception des véhicules âgés de plus de 13 ans pour lesquels la sinistralité est minimale. Cela peut venir du fait que lorsque l'on détient une voiture depuis très longtemps, nous sommes d'autant plus prudent avec, et donc moins d'accidents en moyenne.
- *DriverAge* : comme nous l'avions observé en analyse préliminaire, la sinistralité est plus élevée chez les jeunes (18-26) et chez les personnes âgées (86+), et relativement faible et stable dans les classes intermédiaires.
- *Brand* et *Gas* : la sinistralité est plus élevée pour les voitures de marque de la catégorie F (voitures coréennes et japonaises, à l'exception de Nissan), et pour celles ne roulant pas au Diesel.
- *Region* : la région $R74$ (Limousin) semble la plus sinistrée, ce qui est cohérent avec l'analyse préliminaire que l'on avait réalisée sur la figure 5.6.
- *Density* : on observe une croissance ; plus le nombre d'habitants par km^2 est élevé, plus la fréquence de sinistralité est importante. Cela est également en adéquation avec l'analyse initiale (c.f 5.8).

Finalement, le modèle XGBoost semble reproduire le comportement univarié que l'on avait observé entre les modalités des variables et la variable cible $ClaimNb/Exposure$ de fréquence de sinistres. Notons que l'on a analysé ici par le biais du graphique de dépendance partielle, comme il est couramment réalisé dans la littérature. Cependant, comme nous l'avons illustré dans le chapitre 4, la méthode ALE est une alternative à la fois plus rapide mais également non biaisée du PDP. La différence de vitesse d'exécution des deux méthodes est d'autant plus marquée lorsque des variables numériques sont utilisées dans le modèle. Prenons l'exemple du XGBoost fréquence 2 (celui laissant les variables numériques inchangées). Alors l'estimation du graphique de dépendance partielle (PDP) pour la variable *CarAge* se fait en environ 100 secondes, alors qu'il faut moins de 5 secondes pour le graphique d'effets locaux accumulés (ALE) associé à la même variable. Comme suggéré précédemment dans ce mémoire, nous recommandons toujours l'ALE en comparaison du PDP. Dans notre cas présent, comme les variables sont peu corrélées, les résultats de ces deux graphiques devraient être sensiblement proches. Ceci est bien vérifié sur la figure 5.32. Nous avons représenté à la fois l'ALE associé au XGBoost 1 (avec les variables retraitées) et le XGBoost 2 (avec les variables numériques inchangées). Il faut bien faire attention à l'ordre des modalités des variables sur les deux graphiques de cette figure, qui ne sont pas nécessairement dans le même ordre, notamment pour les variables numériques *CarAge*, *DriverAge* et *Density*.

Analyse de l'interaction entre les variables

Cette analyse univariée, à l'aide des graphiques de PDP ou d'ALE, ne montre pas les effets hétérogènes au sein de la base. En effet, comme nous étudions un modèle

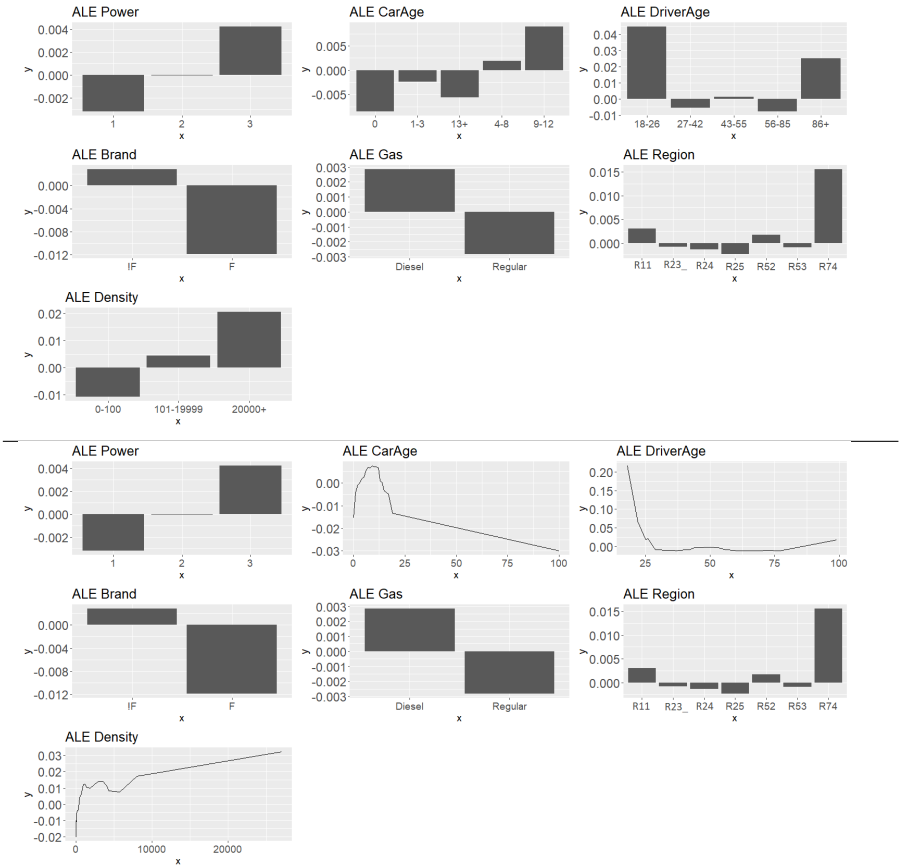

FIGURE *5.32: Graphiques d'ALE des différentes variables utilisées dans l'ajustement du XGBoost fréquence 1 (figure du haut) et du XGBoost fréquence 2 (figure du bas)*

XGBoost, les prédictions ne sont pas aussi simples à expliquer que pour le GLM. En effet, la prédiction du modèle ne pourra pas être exprimée comme la somme des effets individuels de chaque variable, car l'effet d'une variable dépend de la valeur des autres variables. L'article 16 montre que les méthodes construites à partir d'arbres - comme le XGBoost - sont vantées pour leur capacité à modéliser l'interaction entre différentes variables.

H-statistique

Pour mettre en évidence ces possibles interactions, le calcul de la H-statistique est une solution. Celle-ci, étudiée dans le chapitre 4, estime la force d'interaction en mesurant la part de la variance due à l'effet d'interaction entre plusieurs variables. Son calcul repose en grande partie sur la dépendance partielle, avec un ratio de la variance due à l'interaction et de la variance totale. La valeur de la H-statistique est comprise entre 0 et 1 avec 0 référant à l'absence d'interaction et 1 indiquant que la prédiction est purement guidée par l'interaction étudiée. Dans notre cas, nous nous restreignons à l'étude conjointe de deux

variables. Comme la formule de la H-statistique repose sur des dépendances partielles, le temps de calcul est très élevé et des approximations doivent être réalisées. De plus, dans notre étude seules des variables catégorielles sont utilisées ce qui rend l'utilisation de la H-statistique peu pertinente. En effet, comme nous pouvons le voir dans le tableau 5.22, les valeurs de la H-statistique sont anormalement élevées, avec une moyenne supérieure à 0.8.

	Power	CarAge	DriverAge	Brand	Gas	Region	Density
Power		0.95	0.80	0.94	0.99	1.01	0.90
CarAge			0.75	0.95	1.01	0.95	0.96
DriverAge				0.70	0.70	0.73	0.61
Brand					1.07	1.10	1.14
Gas						0.96	0.85
Region							0.78
Density							

TABLE *5.22: H-statistique du modèle XGBoost 1 fréquence pour mesurer l'interaction entre les variables*

Ceci vient du fait que la H-statistique surestime l'effet des interactions à cause des variables catégorielles, comme c'est le cas avec l'exemple de la combinaison $(Power, Gas)$, ayant respectivement 3 et 2 modalités. De plus, même si les valeurs de la H-statistique étaient cohérentes, cela ne nous aurait pas donné d'information concernant la nature de l'interaction entre les deux variables choisies.

Courbes ICE

Pour avoir une idée plus précise de comment l'interaction joue un rôle dans les prédictions, nous pouvons par exemple étudier les courbes ICE (*Individual Conditional Expectation* : espérance conditionnelle individuelle), qui généralisent le graphique de dépendance partielle pour chaque observation (c.f chapitre 4). Notons bien que les courbes ICE est une méthode d'interprétation locale, mais nous les avons tout de même conservées au sein de cette partie car elle s'inscrivent dans la continuité des graphiques de dépendance partielle précédents.

Contrairement au GLM (c.f figure 5.22), les courbes de la figure 5.33 ne sont pas translatées entre elles, signe d'une hétérogénéité.

Nous avons également représenté, sur la figure 5.34, différentes courbes ICE associées à une variable, en ajoutant un code couleur référant à la modalité d'une autre variable choisie. Ceci permet de mettre en exergue la manière dont l'interaction joue un rôle sur la prédiction du modèle. Un effet important et très bien connu en tarification automobile est l'interaction entre l'âge de l'assuré et la puissance du véhicule. En effet, plusieurs études ont montré que l'effet couplé d'un conducteur jeune et d'un véhicule de puissance élevée augmentait drastiquement le risque de sinistres (en terme de fréquence). Ce phénomène peut-être observé sur la figure 5.34, sur le graphique du milieu en haut où sont représentées les courbes ICE associées à la variable $DriverAge$, en affichant égale-

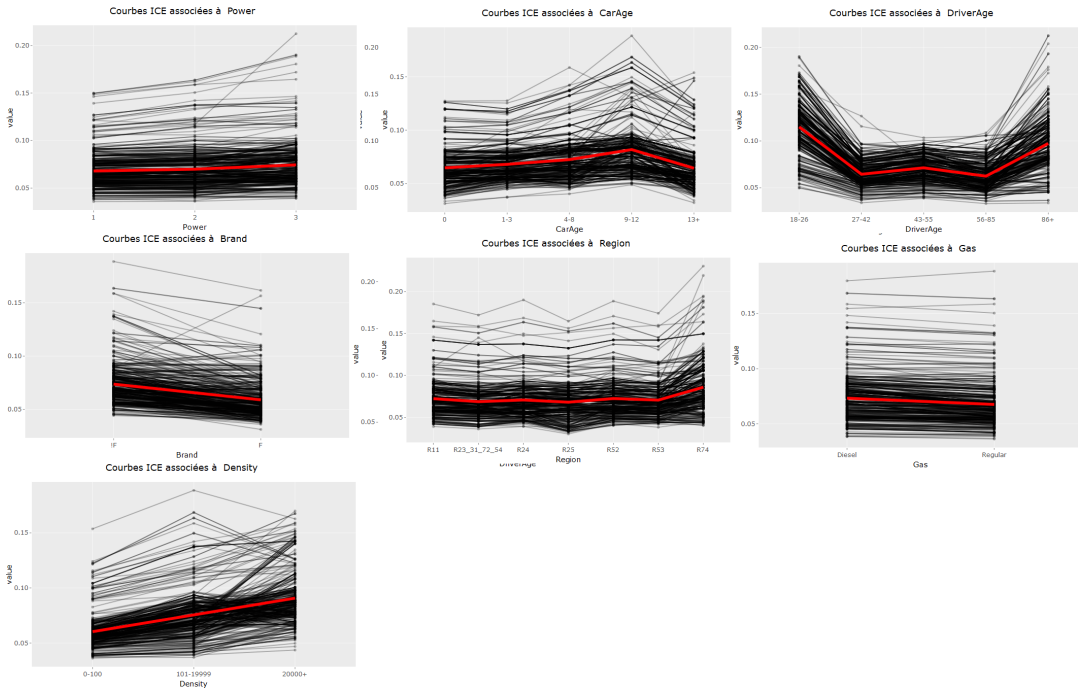

FIGURE *5.33: Courbes ICE affichées pour 500 observations différentes, pour les 7 variables explicatives du modèle XGBoost fréquence 1*

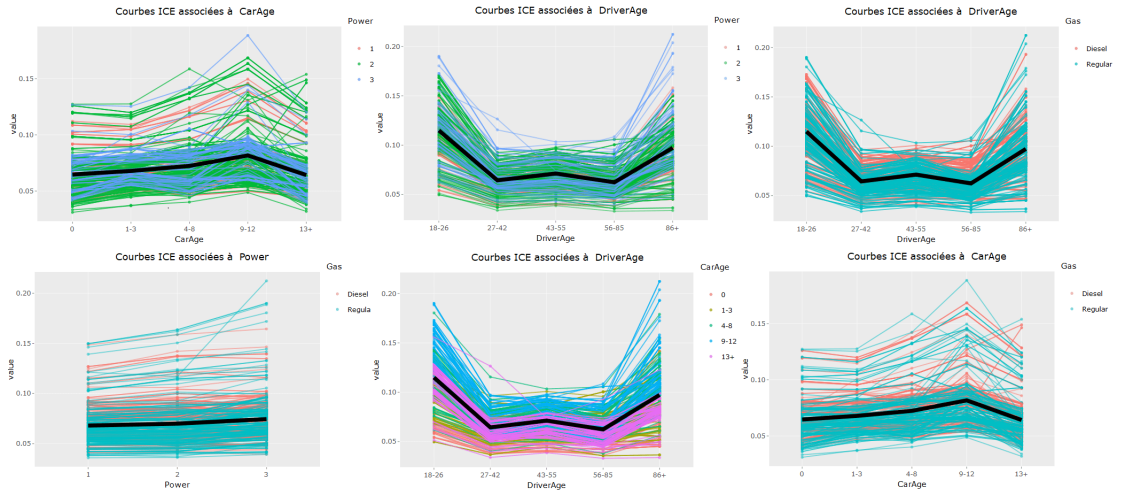

FIGURE *5.34: Courbes ICE du modèle XGBoost fréquence 1. En noir est représentée la courbe de dépendance partielle (PDP) qui est la moyenne de toutes les courbes ICE.*

ment la puissance du véhicule. En rouge, il s'agit des véhicules de catégorie 1, en vert de catégorie 2 et en bleu de catégorie 3. Pour mettre en évidence la remarque précédente sur l'effet combiné de (*DriverAge*,*Power*), il faut comparer les courbes bleues, pour

lesquelles la puissance du véhicule est la plus élevée, à la courbe noire de dépendance partielle représentant l'effet moyen de l'âge du conducteur sur la prédiction. On remarque de nombreuses courbes bleues ayant une pentification plus importante entre les modalités $18-26$ et $27-42$, que le PDP, signe d'un risque supplémentaire de sinistres lorsque l'assuré est jeune et qu'il possède une voiture puissante. La même remarque peut être faite lorsque l'assuré est à la fois vieux (plus de 86 ans) et possède une voiture puissante (de catégorie 3).

PDP et ALE associées à deux variables

Pour étudier l'effet sur les prédictions des interactions entre les variables nous pouvons également étudier les graphiques PDP (ou ALE), en analysant différents croisements de variables. Comme nous l'avons souligné précédemment, ces courbes sont des translations en cas de non interaction, ce qui n'est pas le cas avec l'utilisation d'un modèle complexe comme le XGBoost.

Nous pouvons représenter de nombreuses combinaisons possibles de croisements entre les variables ; pour garder en lisibilité nous décidons d'en afficher que quelques-un.

Sur le graphique 5.36, la variable *CarAge* est couplée avec toutes les autres variables pour voir l'effet sur la prédiction de la combinaison des deux variables. La même étude est réalisée sur la figure 5.35 avec la variable *Density*.

Commençons par analyser la variable *Density*. En regardant, l'ALE (ou le graphique PDP) de cette variable seule (c.f 5.32, on observe une croissance de la variable avec la densité de population. Cette croissance de courbe est sur une moyenne des prédictions, sans tenir compte de la valeur des autres variables. A l'aide du graphique 5.35, on voit l'effet de la densité sur la sinistralité, conditionnellement aux valeurs prises par une autre variable. Commençons par le graphique en haut à gauche, correspondant à l'effet *Density* et *Power*. On remarque que le comportement vis à vis de la valeur de la densité est totalement différent, suivant la valeur prise par *Power*, contrairement au modèle GLM pour lequel chaque courbe serait translatée les unes des autres. En effet, on remarque que si $Power = 1$ ou $Power = 2$, l'effet de la densité est globalement assez faible sur la prédiction. On observe même une décroissance sur les faibles valeurs de densité, ce que l'on ne pouvait imaginer avec le graphique d'ALE associé à *Density* seul. Pour $Power = 3$, le comportement de la variable cible par rapport à la densité est très proche de l'effet moyen observé sur la courbe 5.32.

Concernant le croisement *Density* et *Brand*, lorsque $Brand = !F$, on remarque que l'impact de la valeur de la densité sur la réponse est globalement constant, alors que lorsque $Brand = F$, on observe une forte décroissance pour les valeurs faibles de densité, puis une croissance à partir de 2000 $habitants/km^2$.

On note également que suivant que $Gas = Diesel$ ou $Gas = Regular$, l'effet de la densité sur la fréquence de sinistres est contraire : croissante dans le premier cas, et décroissante dans l'autre.

A propos des régions, seule la modalité $R23\_31\_72\_54$ (notée $R23\_$) semble se comporter différemment des autres. En effet, on remarque qu'à partir d'une densité supérieure à 4000, une croissance est observée.

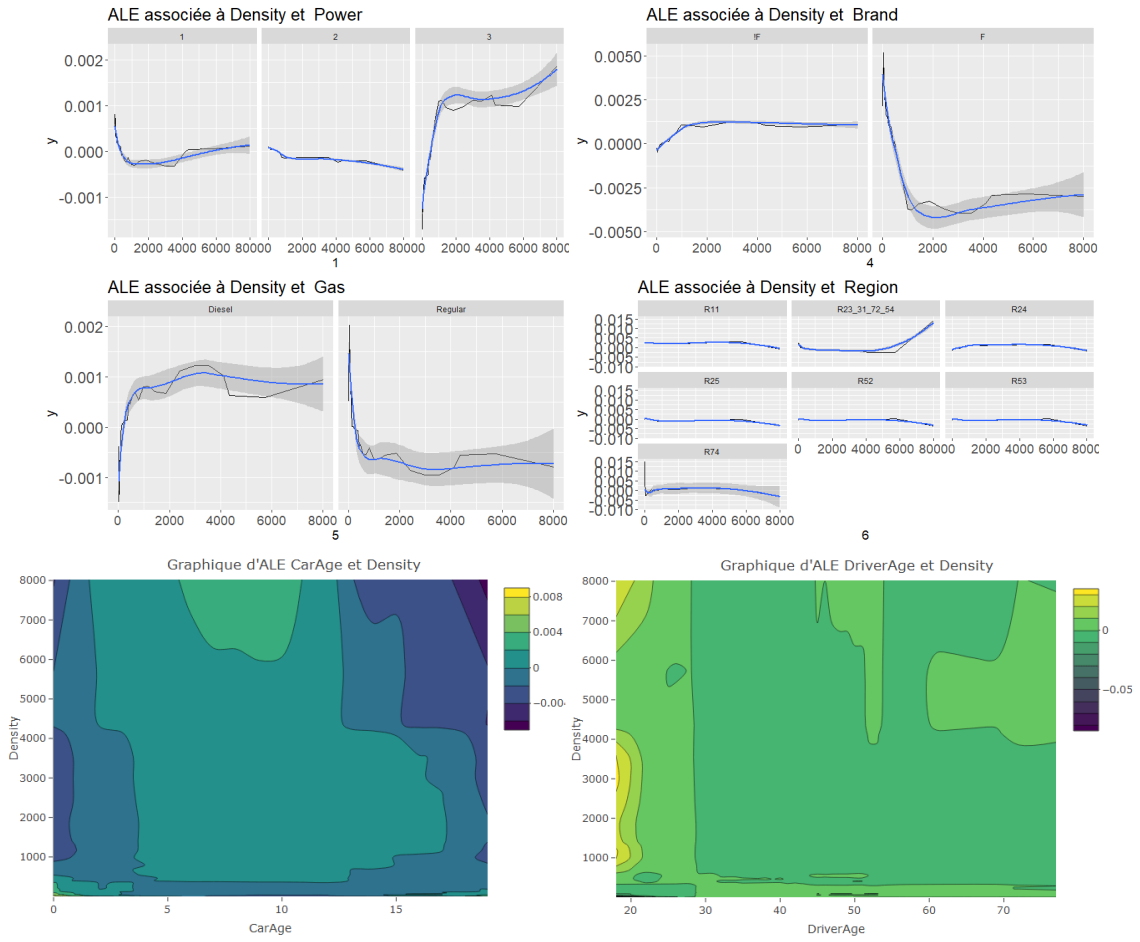

FIGURE *5.35: Graphiques d'ALE associés à la variable Density avec toutes les autres variables utilisées dans le modèle XGBoost fréquence 2*

Concernant les croisements de *Density* avec les autres variables numériques (*CarAge* et *DriverAge*), l'interprétation demeure légèrement plus difficile. Tout d'abord, notons que nous avons réduit la zone des graphiques pour une meilleur lisibilité des résultats :

- *Density* est réduit sur [0,8000], car peu d'individus se trouve en dehors de cet intervalle.
- *DriverAge* est limitée à l'intervalle [18,76].
- *CarAge* est limitée à l'intervalle [0,20].

Ceci permet d'avoir une plus grande significativité des résultats donnés par ces figures. On remarque l'effet combiné d'une voiture très récente et d'une densité faible semble impacter de manière significative la sinistralité. Cet effet peut paraître étrange et semble difficile à expliquer. On note également un pic de sinistralité pour des voitures âgées de 5 à 10 ans, dans des villes de densité supérieure à 6 000. Sur la figure 5.36, on remarque également un effet combiné du couple *(DriverAge,CarAge)*, notamment avec une plus

grande sinistralité lorsque la voiture est âgée de 6 à 12 ans et que le conducteur a moins de 20 ans. On aurait pu attendre également un effet décuplé sur la sinistralité d'un conducteur jeune et d'une voiture récente mais cela n'est pas observée sur cette figure.

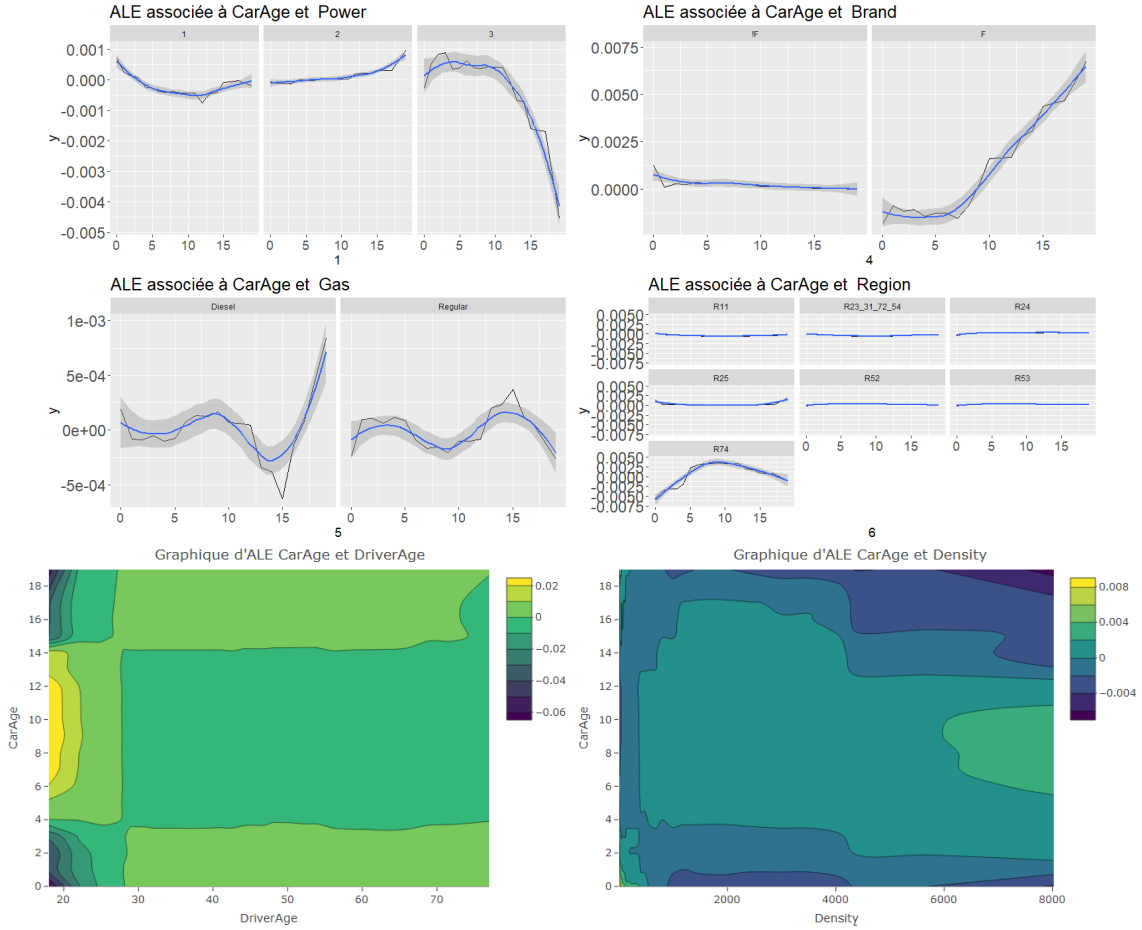

FIGURE *5.36: Graphiques d'ALE associés à la variable CarAge avec toutes les autres variables utilisées dans le modèle XGBoost fréquence 2*

5.6.2.2 Analyse locale

Reprenons à présent l'étude locale des deux assurés étudiés dans le cadre du GLM. Ceux-ci sont détaillés au niveau des tableaux 5.6.1.1 et 5.21. Pour le GLM nous avons pu facilement comprendre le cheminement menant à une telle prédiction par le modèle, d'autant plus qu'une analyse locale n'était pas nécessaire mais l'analyse intrinsèque au modèle se suffisait à elle-même. Comme nous l'avions évoqué à de nombreuses reprises, la nature complexe de l'algorithme (ou de tout autre modèle considéré "boîte noire") rend l'interprétation compliquée. Nous avons recours aux outils évoqués dans le chapitre 4.

Les analyses globales précédentes donnent une idée générale du comportement du modèle mais ne peuvent pas nécessairement expliquer la prédiction réalisée pour un assuré donné. Pour cela, on utilise les outils locaux, dont LIME et SHAP.

LIME

Commençons avec la méthode LIME. Comme nous l'avons vu dans le chapitre consacré aux différentes méthodes d'interprétation des modèles, LIME utilise une approche locale, pour comprendre la prédiction faite pour un individu donné. L'hypothèse sous-jacente est que le comportement de tout modèle, aussi complexe soit-il, peut s'approcher localement par une régression linéaire (de type K-Lasso). Ainsi, dans le cas de notre modèle XGBoost fréquence, on va construire deux modèles qui approchent le comportement local de l'algorithme, au niveau des assurés 1 et 2. Comme nous l'avons évoqué précédemment, bien que LIME soit une des méthodes les plus couramment utilisées pour interpréter un modèle, elle en possède de nombreuses comme l'instabilité des résultats produits dans grand nombre de situations. Pour s'assurer que l'on peut faire confiance à l'explication fournie par LIME pour nos deux assurés (i.e. s'assurer de la stabilité), nous réalisons différentes simulations de l'algorithme et nous représentons les résultats sous la forme d'une boîte à moustaches (c.f. graphiques en bas à droite des figures 5.37 et 5.38). Dans le cas des deux figures évoquées ci-avant, une régression K-Lasso avec 4 variables explicatives a été utilisée comme modèle de substitution dans la méthode LIME. Les résultats semblent assez stables pour les assurés 1 et 2, avec des boîtes à moustaches assez resserrées, d'autant plus pour l'assuré 2. Une autre courbe pour juger de la stabilité des informations fournies par LIME ainsi que de l'importance de chaque variable dans la prédiction est celle en bas à gauche des figures 5.37 et 5.38. Cette courbe montre le pourcentage d'utilisation de chaque variable au cours des différentes simulations de la méthode LIME. Pour l'assuré 1, on remarque que les variables *Brand*, *Density* et *Gas* sont utilisées au cours de chaque simulation (100 % indiqué sur la courbe). Ceci nous montre que ces variables semblent être les plus influentes sur la prédiction de 0.07695 faite par notre modèle XGBoost fréquence 1 pour l'assuré 1. La boîte à moustaches associée nous montre que ses variables jouent une influence positive sur la prédiction, c'est-à-dire ont tendance à faire augmenter la valeur prédite. Les deux autres variables semblant avoir un impact sur la prédiction pour l'assuré 1 sont *Power* et *DriverAge* qui sont utilisées dans environ 50% des simulations. Enfin, la variable *Region*, utilisée par LIME dans seulement 1% des simulations, et la variable *CarAge* ne semblent pas avoir un rôle majeur. Pour conclure sur l'assuré 1, une méthode semblable à LIME, appelée Live et détaillée dans le chapitre 4 a été utilisée, comme nous pouvons le voir sur le graphique en haut à droite de la figure 5.37. Les résultats semblent cohérents avec ceux fournis par LIME : les variables *Brand*, *Density* et *Gas* sont les plus importantes dans la prédiction faite par le XGBoost fréquence. Bien que la quatrième variable utilisée par Live est ici *CarAge*, et qu'elle n'est pas présente dans LIME, son rôle est négligeable devant celui des variables citées juste avant et ne remet pas en cause les explications fournies par ces deux méthodes.

Concernant l'assuré 2, les résultats sont d'autant plus stables car au cours de chaque simulation, les quatre mêmes variables ont été utilisées à chaque fois, à savoir *Brand*,

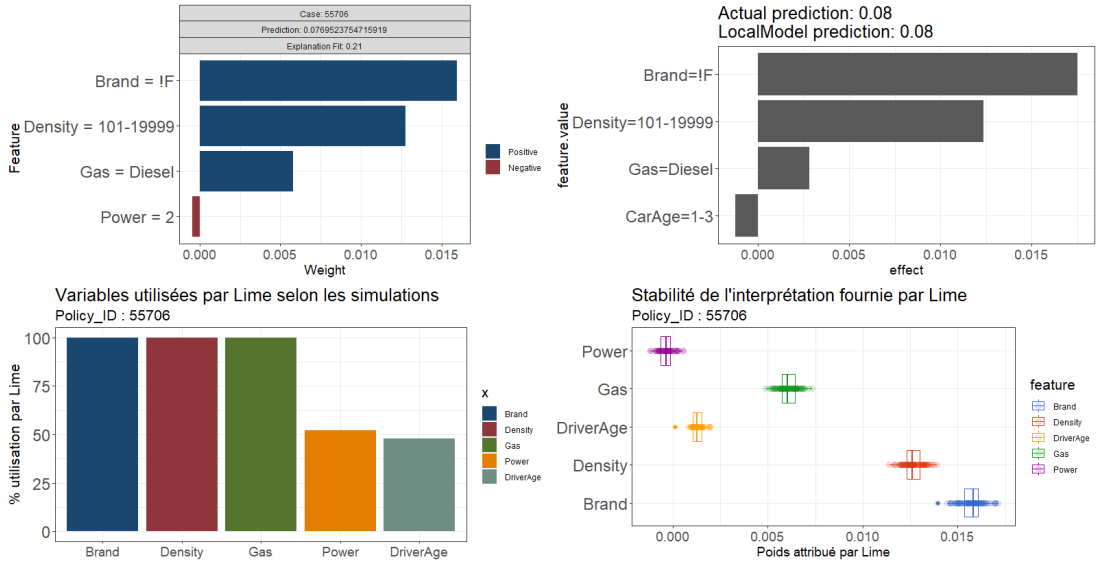

FIGURE *5.37: Résultats fournis par la méthode de substitution locale pour l'assuré 1 avec le modèle XGBoost fréquence 1. En haut à droite, la méthode Live est utilisée, en haut à gauche, la méthode LIME est utilisée. Les deux schémas du bas correspondent à l'analyse de la stabilité de la méthode LIME sur 100 simulations différentes.*

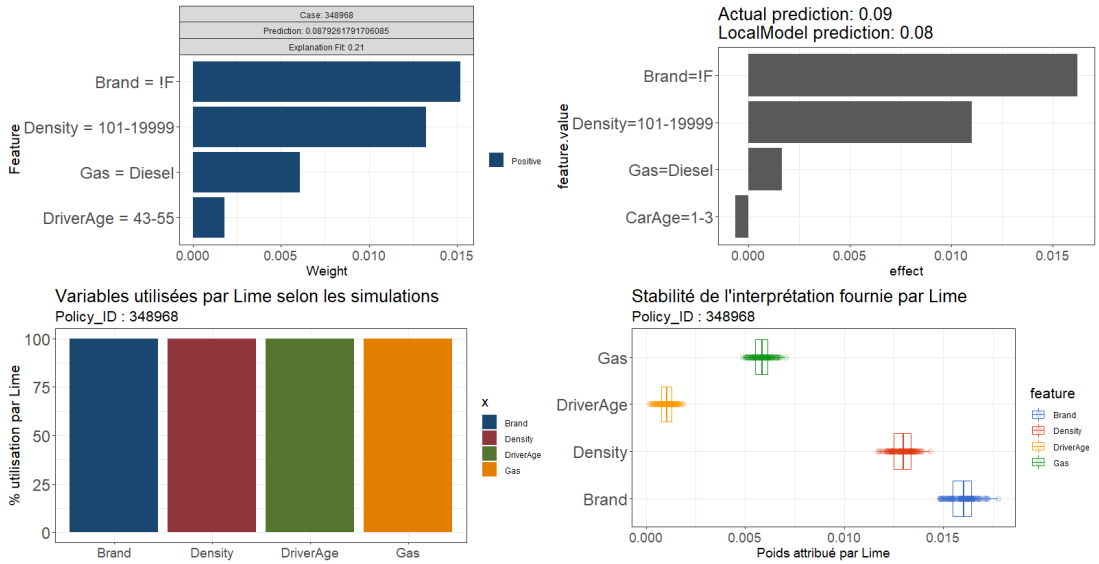

FIGURE *5.38: Résultats fournis par la méthode de substitution locale pour l'assuré 2 avec le modèle XGBoost fréquence 1. En haut à droite, la méthode Live est utilisée, en haut à gauche, la méthode LIME est utilisée. Les deux schémas du bas correspondent à l'analyse de la stabilité de la méthode LIME sur 100 simulations différentes. 4 variables sont utilisées pour les deux méthodes*

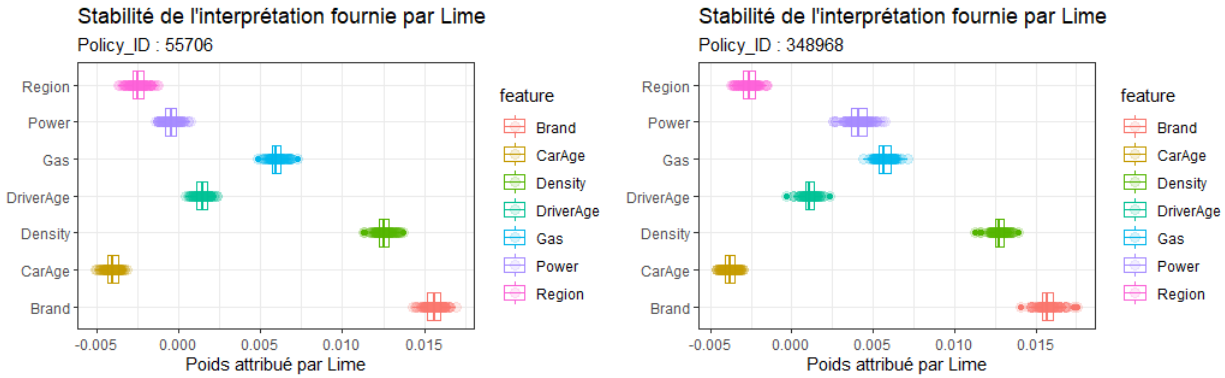

FIGURE *5.39: Stabilité des résultats fournis par LIME pour expliquer la prédiction de l'assuré 1 (à gauche) et de l'assuré 2 (à droite) en utilisant les 7 variables dans le modèle du substitution K-Lasso*

Density, *DriverAge* et *Gas*. De même l'ordre d'importance des variables est clairement visible sur la boîte à moustache de la figure 5.38, avec notamment *Brand* et *Density* qui semblent être les plus influentes. De même que pour l'assuré 1, la méthode Live a été également appliquée (en haut à droite de la figure) et les résultats sont eux aussi cohérents, à l'exception toujours de la variable *CarAge* qui est utilisée, avec néanmoins une influence très limitée. Des graphiques complémentaires sont représentés sur la figure 5.39 pour lesquels toutes les variables - les sept - ont été utilisées dans le modèle de substitution mis en place par LIME. En général, on privilégie des modèles parcimonieux, avec très peu de variables pour mieux comprendre comment le modèle étudié est arrivé à telle prédiction, mais cette étude montre toujours une stabilité des résultats et également le rôle de chaque variable. Ceci peut nous permettre d'expliquer pourquoi la prédiction faite par le XGBoost fréquence est plus élevée pour l'assuré 2 que pour l'assuré 1 (prédiction de 0.087926 contre 0.076952) alors que la seule différence entre ces deux assurés est la puissance du véhicule, à savoir catégorie 2 pour l'assuré 1 et catégorie 3 pour l'assuré 1. En effet, on observe sur la figure 5.39 que la contribution de la variable *Power* est négative (proche de 0) pour l'assuré 1 alors qu'elle est positive pour l'assuré 2. Il s'agit en réalité de la seule différence notable entre les résultats fournis par LIME pour ces deux assurés et nous permet de mieux comprendre la différence de prédiction. Nous aurions également pu raisonner différemment. Étant donné que la sinistralité moyenne estimée par le XGboost fréquence est plus élevée pour les assurés du groupe $Power = 3$ que pour ceux du groupe $Power = 2$ (c.f. graphique de dépendance partielle ou graphique ALE des figures 5.31 et 5.32), nous aurions pu conclure qu'il paraît logique que la prédiction pour l'assuré 2 soit plus élevée. Néanmoins, cette analyse n'est pas correcte et aurait pu s'avérer fausse dans d'autres situations. En effet, du fait des interactions entre les variables, nous ne pouvons conclure aussi rapidement.

Illustrons ce propos par le biais d'un exemple, avec deux assurés ayant eux également des caractéristiques très proches et pour lesquels les prédictions réalisées par le modèle XGBoost fréquence ne sont pas en accord avec l'analyse globale. On considère l'assuré

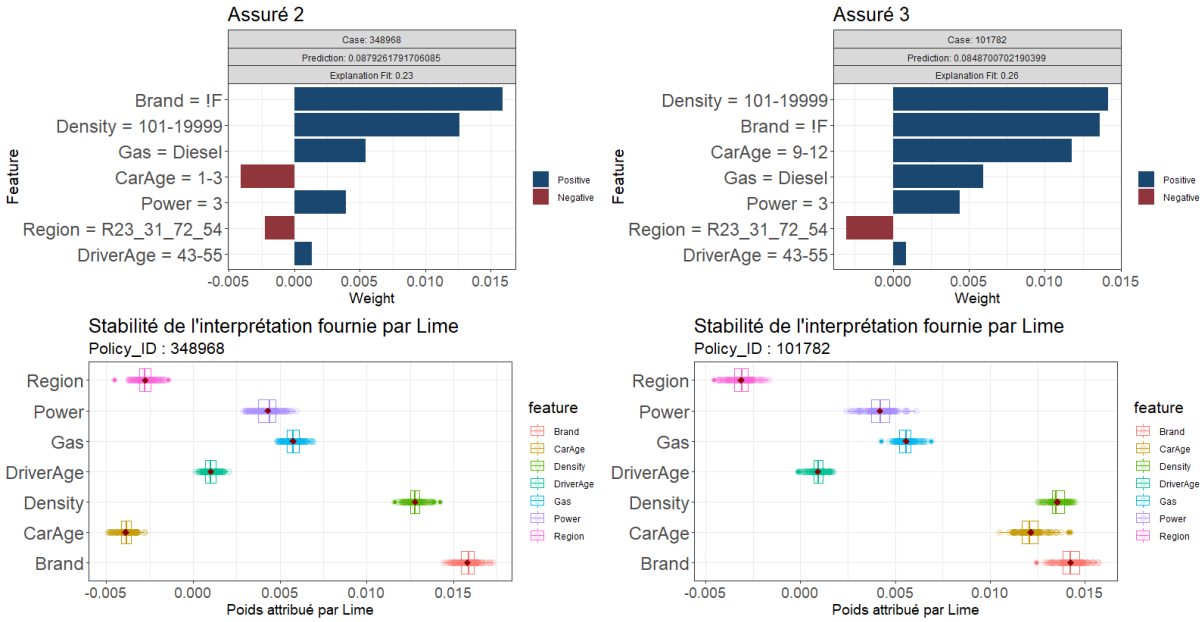

FIGURE *5.40: Méthode LIME appliquée sur les assurés 2 et 3*

2 précédent, et un autre assuré, appelé assuré 3 par la suite, qui a les mêmes variables explicatives, sauf l'âge de la voiture, qui appartient à la catégorie *9-12* et non *1-3* (c.f 5.21). Le graphique de dépendance partielle indique qu'en moyenne, la fréquence de

	Power	CarAge	DriverAge	Brand	Gas	Region	Density	Exposure	Prédiction XGBoost
348968	3	1-3	43-55	!F	Diesel	R23_31_72_54	101-19999	1.0000	0.0879
101782	3	9-12	43-55	!F	Diesel	R23_31_72_54	101-19999	1.000	0.0849

TABLE *5.23: Caractéristiques des assurés 2 et 3, et prédictions du XGBoost fréquence*

sinistres prédite par le modèle XGBoost est plus élevée dans la classe *9-12* que dans *1-3* (valeurs de la fonction de dépendance partielle de 0.0817 contre 0.0679). Ainsi comme les assurés ont toutes les autres caractéristiques égales, il semblerait logique que l'assuré 2 est une valeur prédite par le XGBoost plus élevée . Ceci n'est pas vérifié, comme nous pouvons le voir dans le tableau 5.23, avec des prédictions de 0.0879 et 0.0849 respectivement pour les assurés 2 et 3. Nous voyons bien à travers cet exemple l'importance d'une analyse locale, pour mieux comprendre pourquoi la prédiction est plus élevée pour l'assuré 2. Celle-ci est réalisée sur la figure 5.40, sur laquelle est présentée les résultats de la méthode LIME, avec une simulation en particulier sur les graphiques du haut, et des boîtes à moustaches, basées sur 100 simulations, en bas pour vérifier la stabilité des explications. Comme nous l'avions vu avec le graphique de dépendance partielle, l'effet de *CarAge* sur la prédiction est négatif pour l'assuré 2 (modalité *1-3*) alors qu'il est positif pour l'assuré 3 (modalité *9-12*), ce qui pourrait nous faire croire que la prédiction serait plus élevée pour l'assuré 3. Cependant, même si toutes les autres variables sont égales par ailleurs, leurs effets ne sont pas les mêmes pour les deux assurés, à cause des interactions, comme

nous pouvons le voir sur la figure 5.40, avec notamment $Brand =!F$ qui a une influence plus élevée chez l'assuré 2, tout comme $DriverAge = 43 - 55$. Ceci permet d'expliquer finalement pourquoi la prédiction de l'assuré 3 est plus faible.

SHAP

A présent, nous allons utiliser la deuxième méthode locale la plus répandue pour comprendre les prédictions faites pour les assurés 1 et 2, à savoir la méthode SHAP. Celle-ci longuement évoquée dans le chapitre 4 utilise la théorie des jeux pour calculer une contribution de chaque variable sur la prédiction réalisée moins la prédiction moyenne. Elle présente l'avantage de reposer sur une vraie théorique mathématique, et donc de pouvoir éventuellement être utilisée pour justifier l'utilisation de boîtes noires dans le cadre du RGPD, et également de réaliser une distribution "juste" entre les différentes variables. Notons bien que les méthodes LIME et SHAP n'utilisent pas les mêmes outils et rien ne nous dit que les résultats vont être identiques, bien qu'ils devraient être sensiblement proches. Ceci est bien vérifié pour l'assuré 1, pour lequel l'ordre d'importance des variables est identique ainsi que le sens (positif ou négatif) de la contribution. Les boîtes à moustaches de la figure 5.41 n'ont pu être représentées que pour un petit nombre de simulations (10), étant donné le temps d'exécution de la méthode SHAP. De plus, la taille de l'échantillon utilisé est également très faible, avec seulement 100 valeurs utilisées. Bien que les valeurs précédentes soient très faibles, le temps d'exécution demeure bien plus élevé que pour LIME (c.f tableau 5.24), rendant l'utilisation de la méthode SHAP (et de la méthode BreakDown) pour des bases de données volumineuses limitée.

Assureur	Nombre de points utilisés par simulation	Nombre de simulations	Temps d'exécution (secondes)
SHAP	100	25	46
LIME	5000	100	14

TABLE *5.24: Temps d'exécution des méthodes LIME et SHAP*

On remarque pour l'assuré 2 une forte instabilité des coefficients fournis par SHAP. En plus d'un temps de calcul bien plus conséquent que LIME, les résultats dans le cas présent ne semblent pas fiables, ce qui nous conduit à privilégier la première approche. Le paragraphe suivant approfondit cette étude de la stabilité des deux méthodes ainsi que le temps de calcul associé.

5.6.2.3 Analyse de la stabilité et du temps de calcul de LIME et SHAP

Le paragraphe précédent nous laisse penser que LIME est à la fois plus rapide en temps de calcul et plus stable que SHAP pour notre base de données. L'étude précédente reposait seulement sur l'étude d'un assuré spécifique, nous proposons à présent de la réaliser pour 100 assurés pris aléatoirement dans la base de données, pour vérifier que les conclusions sont bien identiques. L'idée est pour chaque assuré de mettre en place

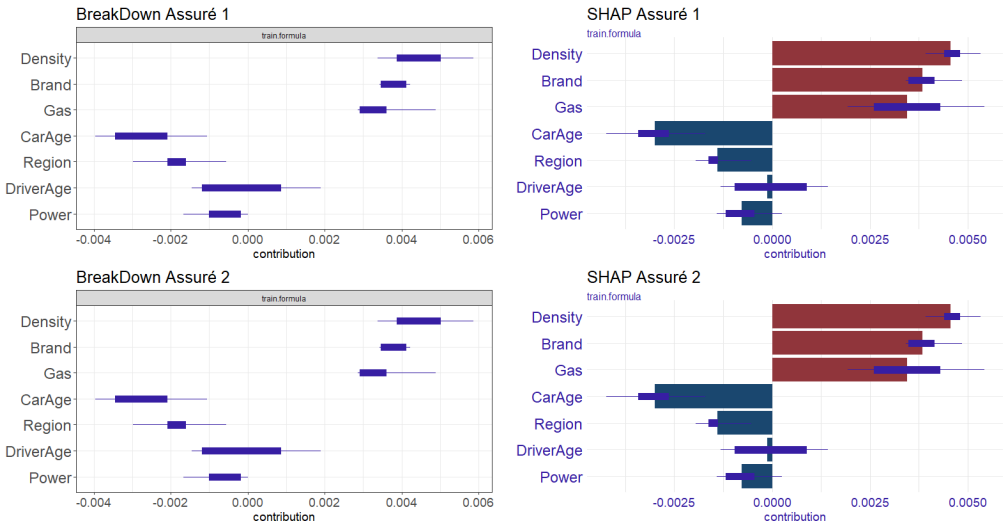

FIGURE *5.41: Stabilité des résultats fournis par les méthodes d'interprétation locales BreakDown (à gauche) et SHAP (à droite) pour les assurés 1 (en haut) et 2 (en bas) associés au modèle XGBoost fréquence 1*

les méthodes LIME et SHAP et d'analyser la stabilité des résultats fournis par chacune d'elles. Pour la méthode LIME, le nombre de permutations a été fixé à 5000, valeur par défaut dans la fonction *lime* du package R *lime*. Pour SHAP, nous avons utilisé 20 échantillons de Monte Carlo dans l'estimation de la valeur de Shapley. Ces valeurs ont été choisies pour éviter un temps de calcul trop important. En utilisant une seule simulation, le temps de calcul associé est donnée sur la figure 5.42. On remarque, comme nous l'avions développé dans la partie théorique sur ces deux méthodes, que LIME est bien moins coûteux que SHAP en temps de calcul. Comme nous étudions la stabilité des résultats fournis, nous allons effectuer plusieurs simulations. Nous utilisons 20 simulations différentes pour chaque méthode. c'est-à-dire que pour les 100 assurés, nous allons utilisés 5000 points pour LIME (et 20 pour SHAP), pour chacune des 20 simulations mises en place. Le temps de calcul total est donné dans le tableau qui suit. Nous voyons que même

Méthode	Nombre de points utilisés par simulation	Nombre de simulations	Nombre d'assurés étudiés	Temps total d'exécution
SHAP	20	20	100	45min
LIME	5 000	20	100	4min

TABLE *5.25: Temps de calcul des méthodes LIME et SHAP suivant le nombre de simulations, le nombre d'assurés étudiés et le nombre de points utilisés par simulation.*

pour peu de points utilisés avec SHAP, le temps de calcul est déjà très grand, proche de 45min ici.

Nous avons représenté les boîtes à moustaches des interprétations fournies par LIME

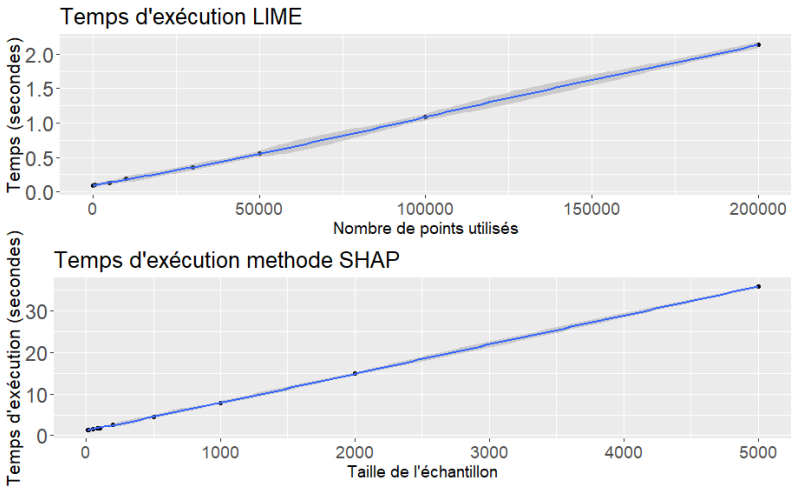

FIGURE *5.42: Comparaison des temps de calcul des méthodes LIME et SHAP suivant le nombre de points utilisés. Pour LIME, il s'agit du nombre de points simulés selon une loi normale pour ajuster le modèle de substitution de type K-Lasso. Pour SHAP, il s'agit du nombre de points utilisés dans l'estimation Monte Carlo de la valeur de Shapley.*

et SHAP pour 4 assurés parmi les 100 étudiés sur la figure 5.43. On remarque une forte instabilité de SHAP en comparaison de LIME.

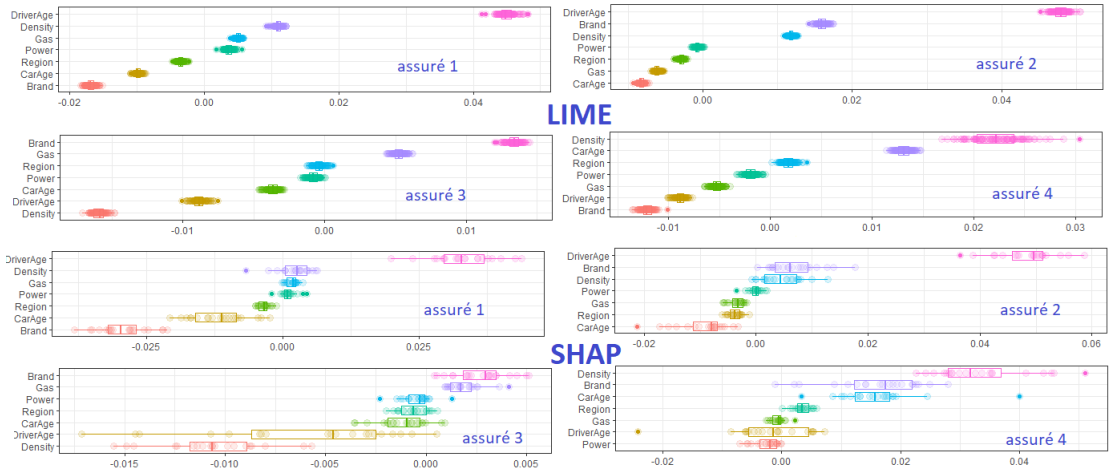

FIGURE *5.43: Analyse de la stabilité de LIME et SHAP sur 4 assurés pris aléatoirement. En haut, sont représentés les résultats pour la méthode LIME avec 5000 observations, et en bas ceux de SHAP avec 20 points.*

Pour vérifier par le calcul l'intuition graphique montrant la plus grande stabilité de LIME en comparaison de SHAP, nous avons calculé les écarts-types moyens de chaque méthode, ainsi que l'étendue normalisée et l'écart entre les valeurs maximales et minimales normalisées. Il en découle qu'en moyenne, la méthode SHAP est 4 fois plus instable

que LIME d'après les critères précédents, en plus d'un temps de calcul 10 fois plus élevé.

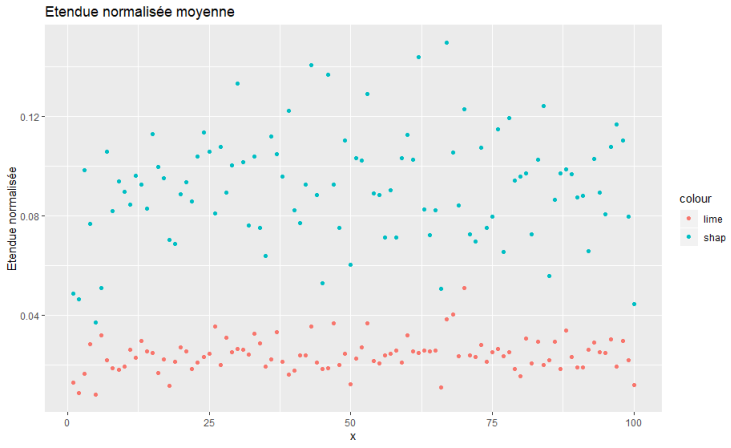

FIGURE *5.44: Étendue (différences des quantiles d'ordre 75% et 25%) normalisée*

Nous avons enfin analysé la stabilité des résultats de LIME et SHAP selon le nombre de points utilisés. Ceux-ci sont donnés sur les figures 5.45 et 5.46.

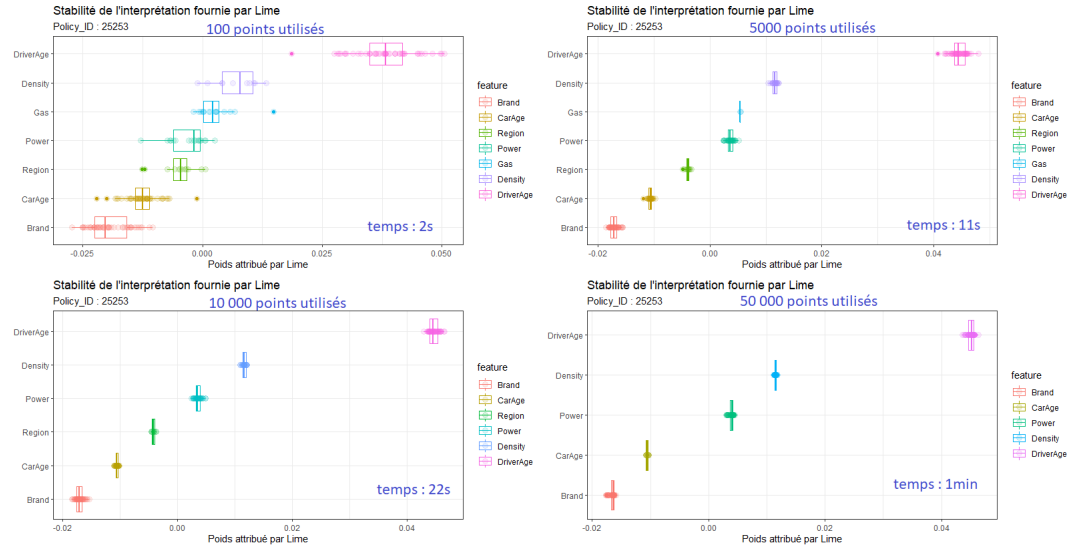

FIGURE *5.45: Stabilité de LIME suivant le nombre d'observations utilisées pour ajuster le modèle local*

Finalement, cette partie nous a permis de montrer que l'approche par un modèle simple comme le GLM peut facilement s'interpréter, que ce soit directement par le biais des coefficients du modèle ou par une analyse post-hoc avec les outils d'interprétation vus dans le chapitre 4, pour lesquels les résultats sont identiques. Cette analyse est facilité par le fait que le modèle est parcimonieux dans notre étude, avec peu de coefficients.

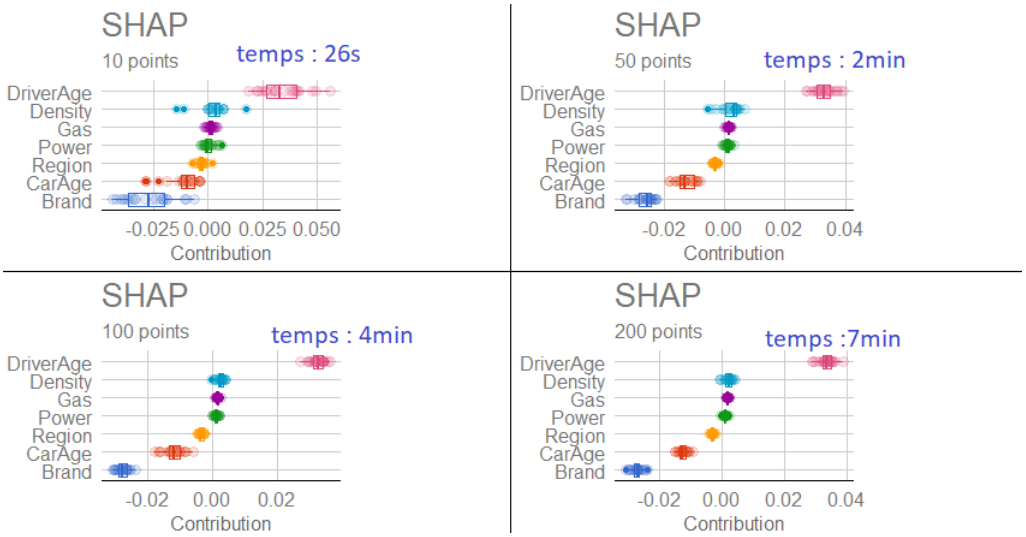

FIGURE *5.46: Stabilité de SHAP suivant le nombre de points utilisés dans la méthode Monte Carlo pour estimer les valeurs de Shapley*

Nous avons vu dans un second temps, la difficulté, à première vue, d'expliquer un modèle complexe comme le XGBoost, et la nécessité d'utiliser les outils d'interprétation précédemment introduits. Nous avons pu observer que l'analyse globale (PDP, ALE, importance des variables, etc.) donnait une idée générale sur le comportement du modèle, mais que pour comprendre en détail le cheminement qui a mené à une telle prédiction pour une observation donnée, une analyse locale était indispensable. A l'aide des méthodes LIME et SHAP (ainsi que leurs variantes) nous avons pu fournir des explications satisfaisantes, en particulier avec la méthode LIME qui s'avère globalement stable dans notre étude. Nous avons pu constater les inconvénients de la méthode SHAP, qui présente à la fois un temps de calcul très important, et également une incertitude élevée, avec un manque de stabilité, lorsque le nombre de points choisis est insuffisant. Il semblerait que pour des bases de données volumineuses, l'approche par LIME soit plus adaptée, si l'on souhaite privilégier un temps de calcul raisonnable.

Conclusion

L'interprétabilité des modèles de machine learning est un sujet d'actualité qui devient peu à peu incontournable pour la majorité des data-scientists mais également des actuaires. Que ce soit pour des raisons éthiques - en s'assurant que le sexe ou l'origine ethnique ne soient pas des variables discriminantes par exemple -, réglementaires - notamment avec l'ACPR et le RGPD qui restent vigilants sur l'utilisation de boîtes noires - ou encore pour générer une plus grande confiance dans les outils utilisés, la question de la transparence et de la compréhension des modèles est aujourd'hui vitale.

Comme nous avons pu le voir tout au long de ce mémoire, de nombreuses méthodes d'interprétation existent pour comprendre le comportement des algorithmes qualifiés de *boîtes noires*. On peut citer les méthodes globales, comme le graphique de dépendance partielle (PDP) ou l'importance des variables et les méthodes locales, comme LIME et SHAP, qui sont les plus utilisées à l'heure actuelle. Nous avons conclu que le graphique des effets locaux accumulés (ALE), bien que moins répandu, était une alternative de premier choix au PDP, notamment par sa rapidité de calcul et le fait de fonctionner lorsque les variables sont corrélées. Nous avons également vu que de nombreuses limites subsistent pour chacune des méthodes locales, dont l'incertitude des interprétations fournies qui demeure la plus problématique. Étant donné le caractère récent des recherches sur ce sujet, nous pouvons espérer des méthodes plus robustes et plus fiables dans un futur proche.

A travers notre application en tarification automobile, nous avons pu mettre en évidence la difficulté d'améliorer significativement les performances des modèles triviaux sur des données réelles, même en utilisant des modèles complexes comme le XGBoost. Ceci est un problème fréquemment rencontré en actuariat, y compris en tarification. Nous nous sommes également questionnés sur la pertinence de juger les résultats d'un modèle par le biais d'un unique critère comme le MSE ou le MAE. Nous avons proposé des critères locaux, comme le MSE calculé sur un sous-ensemble d'observations à définir, pouvant aider dans la prise de décision. Nous avons effectué la même étude sur une autre base de données, proche de celle analysée dans ce mémoire, pour laquelle nous obtenons des résultats sensiblement meilleurs, notamment sur la modélisation en fréquence, avec un gain en terme de MSE ou de déviance de Poisson de l'ordre de 3% avec le modèle XGBoost en comparaison du GLM. Ce gain de précision peut s'avérer essentiel dans un marché d'assurance concurrentiel pour se prévenir de l'anti-sélection notamment et pour mieux maîtriser ses risques.

L'utilisation des méthodes d'interprétation sur cette base de tarification automobile, nous a permis de mettre l'accent essentiellement sur deux points. Premièrement, un modèle d'apprentissage aussi complexe soit-il peut s'interpréter à l'aide de ces outils, malgré leurs limites évoquées ci-dessus. En particulier, la méthode LIME semble plus appropriée lorsque la base de données est volumineuse, contrairement à SHAP qui s'avère instable et coûteuse en terme de calcul. Deuxièmement, les modèles GLM ou GAM, utilisés en tarification avec parfois plus de cent coefficients, sont jugés simples et compréhensibles par la majorité mais sont-ils nécessairement plus interprétables et transparents qu'un réseau de neurones compact par exemple ? Bien que les résultats de l'application effectuée dans ce mémoire soient peu satisfaisants, dans de nombreuses situations les modèles complexes, comme le XGBoost, s'avèrent plus performants et semblent à privilégier étant donné les méthodes d'interprétation à disposition.

Les recherches futures sur le thème du machine learning interprétable (IML) semble se diriger vers un formalisme plus clair, avec notamment des définitions non ambiguës des termes utilisés, un consensus général sur un cadre concernant la manière d'interpréter un modèle de machine learning, comme celui introduit dans l'annexe C, et enfin sur la définition d'une métrique de qualité de ces explications, notamment par le biais de mesures de robustesse, comme celle vue en annexe D.

Annexe A

Deux exemples de modèles complexes : les réseaux de neurones et le SVM

Dans cette partie, nous détaillerons deux familles d'algorithmes d'apprentissage statistique très connues. Celles-ci ont été envisagées dans notre application du chapitre 5 mais n'ont pas conduit à des résultats satisfaisants pour notre base de données. Nous présenterons tout de même leur principe général, car elles peuvent s'avérer très utiles pour améliorer sensiblement les performances du problème considéré, notamment en tarification automobile [13], [38].

A.1 Réseaux de neurones

Les réseaux de neurones (ou méthodes connexionnistes) peuvent être définis comme l'ensemble des méthodes numériques de résolution de problème utilisant des modèles issus de la neurobiologie. Ces techniques ont fait leur apparition dans les années 40, avec notamment les travaux de Neumann, Turing, Wiener et Mac Culloch. Leur but était d'utiliser les progrès réalisés en sciences cognitives sur le cerveau pour les appliquer aux systèmes informatiques et à l'apprentissage statistique [71].

A.1.1 Perceptron simple

Les réseaux de neurones en Machine Learning sont inspirés des neurones réels de l'Homme qui fonctionnent schématiquement de la manière suivante :

- Sommation pondérée des influx nerveux en provenance des neurones auxquels les neurones sont reliés avec les dentrites et synapses.
- Emission dans l'axone d'un influx si la somme précédente dépasse un certain seuil, appelé seuil d'activation.

Le neurone formel de Mac Culloch et Pitts utilise ce principe. Considérons un vecteur d'entrée $x = (x_1, ..., x_p) \in \mathbb{R}^p$ ($p \in \mathbb{N}$), des poids synaptiques $(w_1, ..., w_p) \in \mathbb{R}^p$ et un

seuil d'activation $\theta \in \mathbb{R}$. L'objectif est de prédire la sortie $y(x) \in \{0, 1\}$. Alors les étapes précédentes deviennent :

— Sommation pondérée : $< w, x >= \sum_{i=1}^{p} w_i x_i$

— Sortie $y(x) = \mathbb{1}_{\{<w,x>\geq\theta\}} = \Phi(\sum_{i=1}^{p} w_i x_i + w_0)$, en notant : $w_0 = -\theta$.

Dans ce cas simple de classification binaire, la fonction Φ est une indicatrice, et on l'appelle fonction d'activation ou fonction de réponse. On peut généraliser ce principe, en utilisant d'autres fonctions Φ. On parle alors d'un Perceptron, formé d'une couche constituée de $p + 1$ neurones, dont un correspondant à l'unité de biais avec le terme w_0. Un perceptron prédit donc la sortie grâce à une fonction de décision f, définie par : $f(x) = \Phi(\sum_{i=1}^{p} w_i x_i + w_0)$. On remarque que la fonction a une forme explicite : il s'agit d'une forme paramétrique. Dans le cas de la régression, la fonction d'activation est généralement l'identité, à savoir $\forall u \in \mathbb{R}, \Phi(u) = u$. Pour la classification binaire, nous avons vu que l'on peut utiliser une indicatrice, renvoyant la classe prédite. On peut aussi utiliser une fonction sigmoïde pour prédire la probabilité d'appartenir à la classe 1 :

$$\forall u \in \mathbb{R}, \Phi(u) = \frac{1}{1 + exp(-u)} \; soit : \Phi(\sum_{i=1}^{p} w_i x_i + w_0) = \frac{1}{1 + exp(-\sum_{i=1}^{p} w_i x_i + w_0)} \quad \text{(A.1)}$$

La figure A.1 montre la représentation graphique usuelle d'un Perceptron à une couche, avec $p + 1$ neurones.

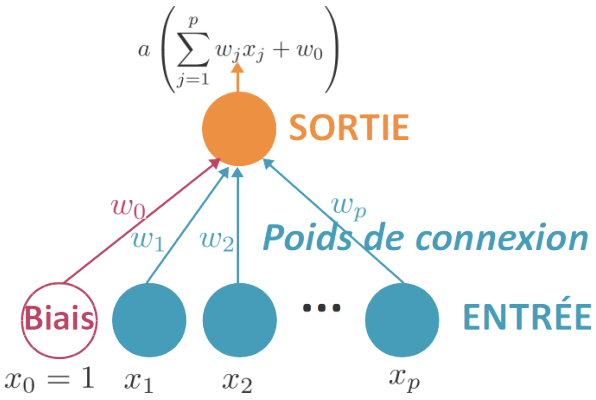

FIGURE *A.1: Architecture d'un Perceptron à une couche, avec a comme fonction d'activation*

Dans le cas d'un problème de classification avec plusieurs classes, on peut généraliser en utilisant autant d'unités de sortie que de classes possibles. Si on a $K \in \mathbb{N}$ classes, on aura donc $K(p+1)$ poids de connexion. La fonction d'activation utilisée est généralement la fonction softmax, qui généralise la sigmoïde vue précédemment, dans le cas de plusieurs

classes :

$$\forall k \in \{1, ..., K\}, \forall u_k \in \mathbb{R}, \sigma_k(u_k) = \frac{e^{u_k}}{\sum_{l=1}^{K} e^{u_l}} \tag{A.2}$$

L'architecture d'un Perceptron dans le cas de plusieurs classes est résumée sur la figure A.2.

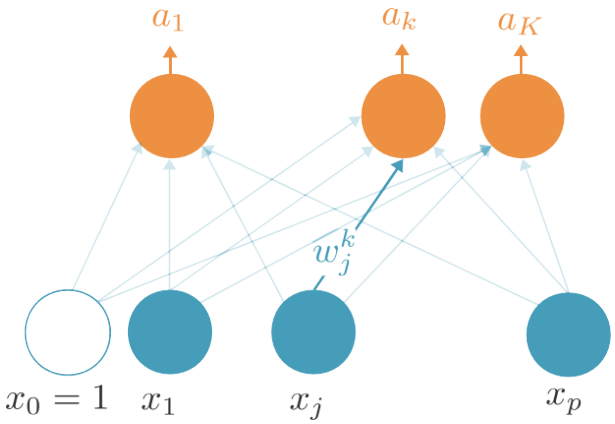

FIGURE *A.2: Architecture d'un Perceptron simple dans le cas d'une variable multi-classes à prédire*

A.1.2 Apprentissage d'un perceptron

Maintenant que l'on connaît la structure d'un perceptron simple, il nous faut comprendre comment déterminer ses paramètres, à savoir ses poids des connexion. $w_0, ..., w_p$. Considérons que l'on dispose de $n \in \mathbb{N}$ observations : $(x^{(1)}, ..., x^{(n)}) \in (\mathbb{R}^p)^n$ et $(y^{(1)}, ..., y^{(n)})$. Pour déterminer les paramètres optimaux, on pourrait utiliser la méthode des moindres carrés comme pour la régression linéaire. Cependant, l'hypothèse réalisée est que les n observations ne sont pas observées simultanément mais séquentiellement, c'est-à-dire l'une après l'autre : $x^{(1)}$, puis $x^{(2)}$,..., puis $x^{(n)}$. L'idée est alors d'ajuster les poids à chaque nouvelle observation, en utilisant l'algorithme de descente du gradient avec une fonction de coût L définie avant. L'idée est de se déplacer dans la direction opposée au gradient pour trouver le minimum d'une fonction. On note L la fonction de coût (ou d'erreur). Soit $I \in \mathbb{N}$ le nombre d'itérations choisi pour l'algorithme. On part avec des poids aléatoires $(w_j^{(0)})_{0 \leq j \leq p}$. Alors pour i allant de 0 à $I - 1$, on définit les nouveaux poids par la relation :

$$w_j^{(i+1)} = w_j^{(i)} - \eta \frac{\partial L}{\partial w_j} \left(y^{(i)}, f(x^{(i)}) \right) \tag{A.3}$$

$$= w_j^{(i)} - \eta \frac{\partial L}{\partial w_j} \left(y^{(i)}, \Phi(\sum_{k=1}^{p} w_k x_k^{(i)} + w_0) \right), \ 0 \leq j \leq p \tag{A.4}$$

Le paramètre $\eta \in \mathbb{R}$ est un paramètre du réseau de neurone, appelé vitesse d'apprentissage (ou learning rate en anglais).
Dans le cas de la régression, la fonction d'erreur la plus courante est l'erreur quadratique :

$$\forall (y, y') \in \mathbb{R}^2, L(y, y') = \frac{1}{2}(y - y')^2 \tag{A.5}$$

En classification, on utilise l'entropie croisée, à savoir :

$$\forall (y, y') \in \mathbb{R}^2, L(y, y') = -y\ log(y') - (1 - y)log(y') \tag{A.6}$$

En utilisant la fonction d'erreur définie par l'équation A.5 ou par l'équation A.6, la formule de mise à jour des poids de l'équation A.4 s'écrit alors de la même manière et est donnée par :

$$w_j^{(i+1)} = w_j^{(i)} - \eta(y^{(i)} - f(x^{(i)})x_j^{(i)},\ 0 \leq j \leq p \tag{A.7}$$

A.1.3 Perceptron multicouche

A.1.3.1 Architecture d'un Perceptron multi-couche

Bien que l'on applique des fonctions pas nécessairement linéaires au sein du réseau, la structure d'un Perceptron est globalement linéaire, étant donné qu'on réalise une combinaison linéaire des entrées. Ceci engendre une limitation dans sa capacité de modélisation de phénomènes complexes. Une idée fut alors d'empiler les couches intermédiaires (appelées par la suite couches cachées) entre la couche d'entrée et de sortie. On parle alors d'un Perceptron multi-couches. Un exemple d'architecture d'un tel réseau est représenté sur la figure A.3. Pour simplifier, considérons un Perceptron avec une seule

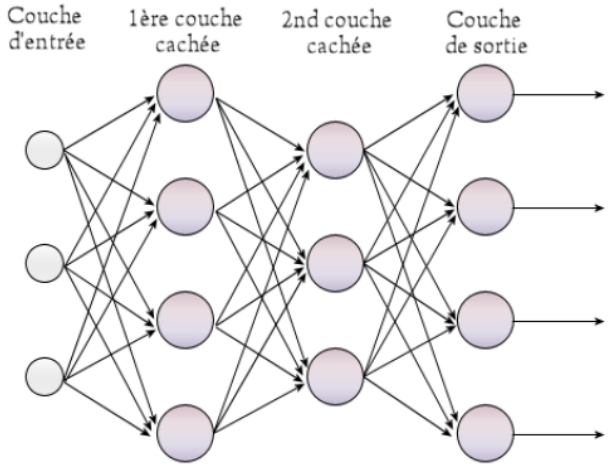

FIGURE *A.3: Architecture d'un Perceptron multicouche, avec 4 couches*

couche intermédiaire, dans le cas d'une classification binaire ou d'une régression. Considérons que cette couche cachée soit constituée de $H \in \mathbb{N}$ neurones. Notons pour tout

$h \in \{1, ..., H\}$, i_h la sortie du neurone h de la couche intermédiaire, et Φ_h la fonction d'activation associée à ce neurone. Notons également $(w_{j,h})_{1 \leq j \leq p}$ l'ensemble des poids associés. Ainsi, la sortie (intermédiaire) i_h est définie par : $i_h = \Phi_h(\sum_{j=1}^{p} w_{j,h} x_j)$. Pour chaque neurone h de la couche intermédiaire, on dispose d'une sortie i_h. La sortie finale du réseau est alors fonction d'une combinaison linéaire de ses sorties intermédiaires, c'est-à-dire : $f(x) = \Phi(\sum_{h=1}^{H} \nu_h i_h) = \Phi(\sum_{h=1}^{H} \nu_h \Phi_h(\sum_{l=1}^{p} w_{l,h} x_l))$ Avec Φ : la fonction d'activation de la sortie (finale), $(\nu_h)_{1 \leq h \leq H} \in \mathbb{R}^H$ les poids associés. On utilise généralement comme fonction d'activation pour les couches intermédiaires et la couche finale la fonction tangente hyperbolique, qui renvoie des valeurs dans l'intervalle [-1,1]. Notons que l'architecture mise en place peut alors posséder un nombre de paramètres gigantesque si plusieurs couches sont empilées. C'est de là que le Deep Learning, ou réseau profond trouve son origine. Ce grand nombre de paramètres nous expose alors à un risque de surapprentissage et il est nécessaire d'avoir un nombre de données conséquent pour entraîner le modèle.

A.1.3.2 Théorème d'approximation universelle

Un résultat fondamental démontré par Cybenko, puis précisé par Hornik, affirme que toute fonction continue de $\mathbb{R}^p$ peut-être approximée, avec une précision $\epsilon > 0$ arbitraire, par un Perceptron à une couche intermédiaire. Le nombre de neurones utilisés dans la couche intermédiaire n'est pas précisé et peut-être très élevé.

Soit $\phi : \mathbb{R} \to \mathbb{R}$ une fonction continue, bornée et non constante. Soit $p \in \mathbb{N}$, soit C_p, $Cont(C_p)$ l'ensemble des fonctions continues de C_p à valeurs réelles. Le théorème affirme que pour tout $\epsilon > 0$ et toute fonction $f \in Cont(C_p)$, il existe $v_i \in \mathbb{R}$, $b_i \in \mathbb{R}$, $w_i \in \mathbb{R}^p$ $(1 \leq i \leq p)$, telle que la fonction F, définie par $\forall x \in C_p,\ F(x) = \sum_{i=1}^{N} v_i \phi(< w_i, x > + b_i)$ vérifie : $|F(x) - f(x)| < \epsilon$. Autrement dit, l'ensemble des fonctions de la forme de F est dense dans $Cont(C_p)$.

A.1.3.3 Algorithme de rétropropagation pour trouver les poids optimaux d'un Perceptron multi-couche

A présent que la structure d'un réseau multicouche a été définie, il reste à trouver les poids optimaux. Pour se faire, on utilise toujours l'algorithme de descente de gradient, c'est-à-dire que l'on adapte l'équation A.4, pour i allant de 0 à $I - 1$, on définit les nouveaux poids par :

$$w_{j,h}^{(i+1)} = w_{j,h}^{(i)} - \eta \frac{\partial L}{\partial w_{j,h}} \left(y^{(i)}, f(x^{(i)}) \right) \tag{A.8}$$

$$= w_{j,h}^{(i)} - \eta \frac{\partial L}{\partial w_j} \left(y^{(i)}, \Phi(\sum_{h=1}^{H} \nu_h \Phi_h(\sum_{l=1}^{p} w_{l,h}^{(i)} x_l^{(i)})) \right),\ 0 \leq j \leq p\ ,\ 1 \leq h \leq H \tag{A.9}$$

Pour le calcul de la dérivée partielle $\frac{\partial L}{\partial w_{j,h}}\left(y^{(i)}, f(x^{(i)})\right)$, on utilise le théorème de dérivation des fonctions composées qui décompose cette dérivée en trois termes :

$$\frac{\partial L}{\partial w_{j,h}}\left(y^{(i)}, f(x^{(i)})\right) = \frac{\partial L}{\partial f(x^{(i)})}\frac{\partial f(x^{(i)})}{\partial i_h}\frac{\partial i_h}{\partial w_{j,h}} \tag{A.10}$$

Illustrons cette formule sur un exemple concret, où les fonctions d'activations utilisées sont toutes une sigmoïde, i.e. : $\forall u \in \mathbb{R}, \Phi(u) = \Phi_h(u) = \frac{e^u}{1+e^u}$ et la fonction de perte est l'erreur quadratique. Alors, les trois termes présents dans la décomposition de la dérivée partielle deviennent :

- $\frac{\partial L}{\partial f(x^{(i)}} = f(x^{(i)}) - y(i)$
- $\frac{\partial f(x^{(i)}}{\partial i_h} = f(x^{(i)})(1 - f(x^{(i)}))i_h$
- $\frac{\partial i_h}{\partial w_{j,h}} = i_h(1 - i_h)x_j^{(i)}$

A.2 Support Vector Machine (SVM)

L'algorithme de machines à vecteurs de support ou de séparateurs à vaste marge (Support Vector Machine en anglais) est une technique d'apprentissage supervisée qui généralise les classifieurs linéaires, dont le but est de prédire une variable binaire. Il existe désormais des adaptations de la méthode SVM originale pour la régression et la classification à plus de deux classes. Son principe repose sur la recherche d'un hyperplan optimal pour séparer les données, tout en étant le plus éloigné possible des observations. Le SVM a été développé dans les années 90, à partir des travaux de Vapnik et Lerner. On distingue les SVM linéaires des SVM non linéaires. Considérons y variable binaire cible valant -1 ou 1, x le vecteur d'entrée ($\in \mathbb{R}^p$), et h la fonction qu'on cherche à optimiser de sorte à avoir $h(x)$ qui approche au mieux y.

A.2.1 Séparateur linéaire

On traite un cas simplifié de classification pour comprendre le concept du SVM. L'objectif va être de fournir une fonction :

$$f_{\alpha,b} : \mathbb{R}^p \to \mathbb{R},\ f_{\alpha,b}(x) =< \alpha, x > +b = \alpha^T x + b \tag{A.11}$$

de sorte que : si $f_{\alpha,b}(x) \geq 0$ on prédit la valeur 1 pour y sinon la valeur -1.

A.2.1.1 Cas linéairement séparable

Comme on se place dans le cas linéairement séparable, on peut trouver un tel hyperplan $H_{\alpha,b} : f_{\alpha,b} = 0$, qui sépare parfaitement notre espace, à partir de notre échantillon $(y_1, ..., y_n) \in \{0, 1\}^n$ et $(x_1, ..., x_n) \in (\mathbb{R}^p)^n$, c'est-à-dire :

$$\forall i \in \{1, ...n\}, y_i f_{\alpha,b}(x_i) \geq 0 \tag{A.12}$$

la figure A.4 représente le cas linéairement séparable. Pour tout point $x_0 \in \mathbb{R}^p$, sa distance

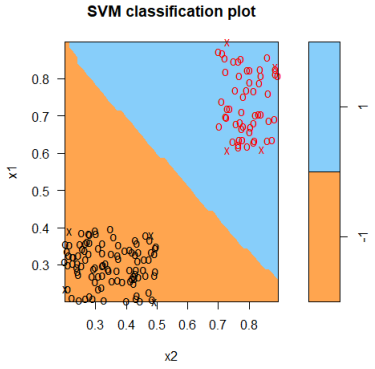

FIGURE *A.4: Illustration du cas linéairement séparable*

à l'hyperplan $H_{\alpha,b}$ est définie par :

$$d(x_0, H_{\alpha,b}) = \frac{| < \alpha, x > +b|}{||\alpha||} \tag{A.13}$$

Pour tous les points x^+_{marge} et x^-_{marge} les plus proches de l'hyperplan $H_{\alpha,b}$, respectivement d'output 1 et -1, on impose les conditions : $f_{\alpha,b}(x^+_{marge}) = +1$ et $f_{\alpha,b}(x^-_{marge}) = -1$. De sorte : $\forall i \in \{1, ...n\},\ y_i f_{\alpha,b}(x_i) \geq 1$ On peut trouver une infinité d'hyperplans qui vérifient de telles propriétés, il nous faut alors faire un choix. On introduit une condition d'optimalité, traduisant le fait que l'on veut l'hyperplan qui maximise la distance entre les points les plus proches des deux classes, c'est-à-dire qui maximise $\frac{1}{||\alpha||}$. On se retrouve finalement avec le problème d'optimisation suivant : Maximisation de $\frac{1}{||\alpha||}$ sous la contrainte : $\forall i \in \{1, ...n\},\ y_i f_{\alpha,b}(x_i) \geq 1$. Le problème est souvent reformulé ainsi : Minimisation de $\frac{1}{2}||\alpha||^2$ sous la contrainte :$\forall i \in \{1, ...n\},\ y_i f_{\alpha,b}(x_i) \geq 1$. On peut résoudre ce problème à l'aide des multiplicateurs de Lagrange, en définissant le lagrangien :

$$L(\alpha, b, \lambda) = \frac{1}{2}||\alpha||^2 - \sum_{k=1}^{n} \lambda_k (y_k (f_{\alpha,b}(x_k) - 1) \tag{A.14}$$

Avec : $\lambda \in \mathbb{R}^n,\ \alpha \in \mathbb{R}^p,\ b \in \mathbb{R}$. Pour résoudre ce problème d'optimisation, le lagrangien doit être minimisé par rapport à α et b et maximiser par rapport à λ. La solution $(\alpha^*, b^*, \lambda^*)$ est donc un point selle du lagrangien. Il vérifie en particulier :

$$\lambda^*_k (y_k (f_{\alpha^*,b^*}(x_k) - 1) = 0,\ \forall k \{1, ..., n\} \tag{A.15}$$

En annulant les dérivées partielles du lagrangien, sans prendre la solution avec $\lambda^*_i = 0$, on obtient les relations suivantes :

$$\alpha^* = \sum_{i=1}^{n} \lambda^*_i y_i x_i \ et \ \sum_{i=1}^{n} \lambda^*_i y_i = 0 \tag{A.16}$$

Ces contraintes permettent d'écrire la forme duale du lagrangien, à savoir :

$$W(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j < x_i, x_j > \tag{A.17}$$

Pour trouver le point selle, il suffit à présent de maximiser : W avec $\lambda_i \geq 0$, $\forall i \in \{1, ..., n\}$. La résolution de ce problème quadratique d'optimisation nous fournit l'équation vérifiée par l'hyperplan optimal recherché.

A.2.1.2 Cas non séparable

Dans la partie précédente, nous avons supposé que les observations étaient linéaires séparables, ce qui est une hypothèse très forte, rarement vérifiée en pratique. La figure A.5 montre l'exemple des fonctions "ou" (or), "et" (and) et "ou exclusif" (xor), qui sont linéairement séparables pour les deux premières mais pas pour la dernière citée. En considérant toujours un séparateur linéaire, nous allons voir comment déterminer

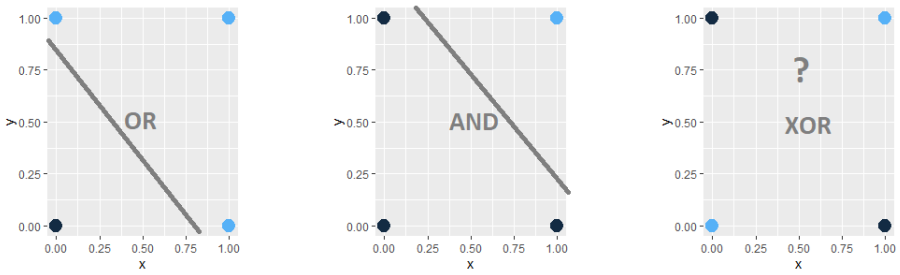

FIGURE *A.5: Fonctions OR, AND linéairement séparables et XOR qui n'est pas linéairement séparable*

l'hyperplan optimal, lorsque les données ne sont pas linéairement séparables. L'idée est d'assouplir les contraintes en introduisant un terme d'erreur ϵ_i $(1 \leq i \leq n)$, qui contrôle les dépassements :

$$y_i < w, x_i > +b \geq 1 - \epsilon_i, \ 1 \leq i \leq n$$

C'est-à-dire que l'on tolère que certains points ne respectent pas la condition souhaitée initialement. L'objectif est toujours le même, on veut l'hyperplan optimal qui sépare au mieux les données (de la même manière que dans la partie précédente) tout en minimisant également l'erreur introduite par les coefficients ϵ_i $(1 \leq i \leq n)$. On a alors affaire à un nouveau problème d'optimisation, Minimisation de $\frac{1}{2}||\alpha||^2 + C \sum_{i=1}^{n} \epsilon_i$ sous la contrainte : $\forall i \in \{1, ...n\}$, $y_i f_{\alpha,b}(x_i) \geq 1 - \epsilon_i$, où la constante $C \in \mathbb{R}$ est un paramètre qui contrôle la pénalisation des termes d'erreur : il permet de réaliser un compromis entre un bon ajustement et une bonne généralisation. Le lagrangien associé à cette équation est défini par :

$$L(\alpha, b, \epsilon, \lambda) = \frac{1}{2}||\alpha||^2 + C \sum_{i=1}^{n} \epsilon_i - \sum_{k=1}^{n} \lambda_k (y_k (f_{\alpha,b}(x_k) - 1 + \epsilon_k) \tag{A.18}$$

Avec : $\lambda \in \mathbb{R}^n,\ \alpha \in \mathbb{R}^p,\ b \in \mathbb{R}, \forall \epsilon \in \mathbb{R}^n$.

On peut également écrire ce problème sous la forme duale du lagrangien.

A.2.2 Séparateur non linéaire

Nous nous plaçons à présent dans le cas général. On cherche toujours à séparer de manière optimale nos observations. Afin de traiter les cas non linéairement séparables, on a recours à une astuce, appelée astuce du noyau (*kernel trick*), qui consiste à considérer le problème dans un espace de dimension supérieure. Dans ce dernier, il sera plus probable de trouver une séparation linéaire.

A.2.2.1 Astuce du noyau

Soit $\mathbb{X} = \mathbb{R}^p$ notre espace de départ de nos variables explicatives. On applique aux vecteurs d'entrée une transformation $x; \mathbb{X} \to \mathbb{H}$ non linéaire (où $\mathbb{H}$ est l'espace d'arrivée, de dimension supérieure : généralement $\mathbb{H} = \mathbb{R}^m$, avec $m > p$). On appelle $\phi(\mathbb{X})$ l'espace de redescription. On cherche alors dans cet espace un hyperplan de la forme : $h(x) = b + < \alpha, \phi(x) >,\ x \in \mathbb{R}^p$, tel que $\forall k \in \{1, ..., n\},\ y_k h(x_k) \geq 0$, c'est-à-dire qui sépare notre base d'apprentissage dans l'espace de redescription. A l'aide des calculs précédemment réalisés, on obtient des résultats similaires, à savoir la résolution du problème d'optimisation : Maximisation de

$$\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j < \phi(x_i), \phi(x_j) > \tag{A.19}$$

sous les contraintes : $\forall i \in \{1, ..., n\}, \lambda_i \geq 0$ et $\sum_{k=1}^{n} \lambda_k y_k = 0$. En général, il n'est pas possible non plus de trouver une séparation linéaire dans l'espace de redescription, c'est pourquoi on se ramène au cas non séparable de la partie précédente, en introduisant des termes d'erreurs ϵ_i $(1 \leq i \leq n)$. Cette technique a été proposée par C. Cortes et V. Vapnik en 1995 : on parle de marge souple. La formulation précédente pose un problème à cause du produit scalaire entre des vecteurs de l'espace de redescription, qui est de dimension élevée et donc ce sera une opération coûteuse en complexité. C'est ici que l'astuce du noyau intervient, avec l'introduction de la fonction noyau $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$, définie par :

$$\forall (x, x') \in \mathbb{X}^2, K(x, x') =< \phi(x), \phi(x') > \tag{A.20}$$

L'expression de l'hyperplan que l'on recherche sera alors de la forme :

$$h(x) = b + \sum_{i=1}^{n} \lambda_i y_i K(x_i, x),\ x \in \mathbb{R}^p \tag{A.21}$$

L'intérêt de noyau repose sur le fait que le calcul est alors effectué dans l'espace de départ, ce qui est moins coûteux que de le réaliser dans l'espace d'arrivée. De plus, la transformation ϕ n'a pas besoin d'être connue explicitement, car seul le noyau K associé apparaît dans les formules. Montrons ceci avec un exemple simple. On prend

$x = (x_1, x_2) \in \mathbb{R}^2$, et la fonction $\phi : \mathbb{R}^2 \to \mathbb{R}^3$, définie par : $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2$. Alors $\forall (x, x') \in \mathbb{R}^2 \text{x} \mathbb{R}^2, K(x, x') =< \phi(x), \phi(x') >=< x, x' >^2$. On voit alors que l'on n'a pas besoin d'évaluer de manière explicite la fonction ϕ.

A.2.2.2 Différentes fonctions noyaux

Énumérons les principales fonctions noyaux utilisées lors de la réalisation d'un SVM :

— Le noyau linéaire :

$$\forall (x, x') \in \mathbb{X}^2, \ K(x, x') =< x, x' > \tag{A.22}$$

— Le noyau polynômial, pour $c \in \mathbb{R}$, $d \in \mathbb{R}$:

$$\forall (x, x') \in \mathbb{X}^2, \ K(x, x') = (c+ < x, x' >)^d \tag{A.23}$$

— Le noyau radial (ou gaussien)„ pour $\sigma > 0$:

$$\forall (x, x') \in \mathbb{X}^2, \ K(x, x') = exp(-\frac{1}{2}\frac{||x - x'||^2}{\sigma^2}) \tag{A.24}$$

— Le noyau sigmoïde, pour $c \in \mathbb{R}$ et $d \in \mathbb{R}$:

$$\forall (x, x') \in \mathbb{X}^2, \ K(x, x') = tanh(c < x, x' > +d) \tag{A.25}$$

Annexe B

Autres méthodes d'interprétation des modèles

Cette annexe présente d'autres méthodes d'interprétation des modèles de machine learning que celles vues dans le chapitre 4. Ces méthodes ont été étudiées dans un premier temps mais n'ont finalement pas été utilisées pour l'application actuarielle du chapitre 5.

B.1 Une variante aux courbes ICE : les courbes c-ICE

Une adaptation du graphique ICE est le c-ICE, qui consiste à centrer chaque courbe du graphique de base. Lorsque les courbes sur l'ICE ont une grande gamme d'intercepts qui sont donc "tassés" entre eux, l'hétérogénéité du modèle tend à rendre difficile l'interprétation du graphique. L'idée est donc de retirer cet effet niveau dans le graphique c-ICE (*centered ICE*). L'idée est de considérer un point x^* appartenant à x_S comme point de jointure pour toutes les lignes de prédiction. En général, x^* est choisi comme le maximum (ou minimum des valeurs observées). Par exemple, si on note $x_S = (x_1, .., x_s)$ (avec $s \in \{1, ..., p\}$), on aura : $\forall j \{1, ..., s\}, x_j^* = \underset{i \in \{1,...,n\}}{max} x_j^{(i)}$. Ainsi pour chaque courbe $\hat{f}^{(i)}$ du graphique ICE, la c-ICE correspondante est donnée pour tout point x par la relation : $\hat{f}^{(i)}_{centered}(x) = \hat{f}^{(i)}(x) - \hat{f}(x^*, x_C^{(i)})$ En reprenant l'exemple vu pour la courbe ICE dans le chapitre 4, on obtient la courbe c-ICE de la figure B.1.

B.2 Live : une alternative à LIME

B.2.1 Principe général

L'article [10] *Explanations of model predictions with live and breakDown packages* propose deux alternatives aux méthodes LIME et Shap (vues dans le chapitre 4), appelées Live et breakDown. Nous nous concentrons dans cette partie sur la méthode Live (Local Interpretable Visual Explanations). L'objectif principal de cette méthode est le même que LIME, à savoir expliquer une prédiction particulière d'un modèle de type boîte-noire

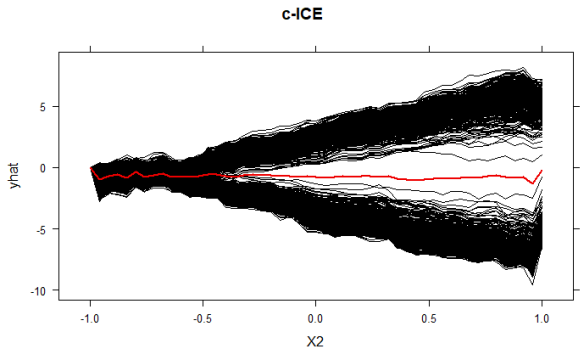

FIGURE *B.1: Graphique de c-ICE (en noir) et de PDP (en rouge)*

en utilisant un modèle de substitution local, mais en portant une grande importance à la visualisation du modèle pour comprendre le modèle complexe. Une première différence avec l'algorithme LIME original est le voisinage choisi pour ajuster le modèle de substitution. L'algorithme de génération du voisinage est le suivant :

- Input : n, le nombre d'observations à générer et p : le nombre de variables explicatives utilisées dans le modèle b de boîte noire.
- Dupliquer les observations données n fois.
- Si $p > n$, alors :
  - Choisir au hasard n variables explicatives
  - Pour $i \in \{1, ..., n\}$:
    - Tirer $k \in \{1, ..., n\}$ uniformément. Remplacer la k i-ème variable avec un tirage de la distribution empirique (dans la base d'apprentissage) de cette variable.
- Sinon :
  - Pour $i \in \{1, ..., p\}$:
    - Remplacer la valeur de la i i-ème variable dans l'observation i par un tirage aléatoire dans la distribution empirique de cette variable.
    - Pour les $(n - p)$ observations restantes, procéder comme dans le cas $p > n$.

Toutes les observations ainsi générées par l'algorithme ci-dessus sont considérées comme similaires à l'instance x d'intérêt et donc un noyau identité est utilisé. L'interprétation de la prédiction de l'instance x par le modèle b se fait alors en utilisant un modèle (simple) de substitution ajusté sur les prédictions faites par b sur les données du voisinage créé par l'algorithme précédent. On utilise généralement comme modèle de substitution (appelé également boîte blanche, en opposition avec le modèle de boîte noire étudié) un modèle linéaire ou un arbre de décision. Dans le cas d'un modèle linéaire, une sélection de variables peut être réalisée, pour avoir un nombre assez petit de variables afin d'interpréter facilement la boîte noire étudiée. Des graphiques peuvent également être réalisés, à savoir :

- Le "Waterfall Plot" qui présente la contribution de chaque variable à la prédiction faite par le modèle.

— Le "Forest Plot" qui résume la structure de l'approximation linéaire locale.

B.2.2 Exemple de Live sur les données "Wine"

Afin de comprendre les résultats fournis par la méthode, considérons un exemple sur la célèbre base de données "Wine", introduite par Cortez et al. (2009), sur laquelle on souhaite prédire la qualité du vin à partir de 11 propriétés chimiques. D'après l'article précédent, l'ajustement d'un SVM donne les meilleurs performances. On considère donc comme modèle de boîte noire à expliquer un SVM avec un noyau radial. Une fois l'ajustement réalisé, intéressons nous à la prédiction faite par ce modèle sur la 5 i-ème observation, définie par les variables explicatives de la figure B.2. La qualité réelle du vin est 5, tandis que la prédiction faite par le modèle est 5,03.

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality
1	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5

FIGURE *B.2: Cinquième observation de la base de données Wine*

On utilise comme nombre d'observations à générer $n = 500$, en gardant les notations de l'algorithme Live défini précédemment. Les résultats obtenus sont résumés par les graphiques Waterfall (figure B.3) et le Forest Plot (figure B.4), correspondant à l'ajustement d'un modèle de régression linéaire, comme modèle de substitution local (boîte blanche).

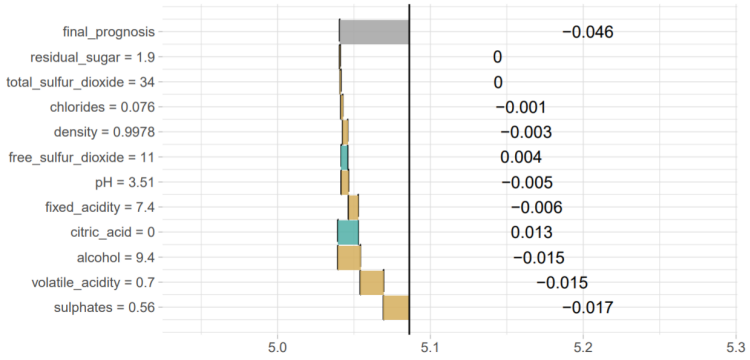

FIGURE *B.3: Waterfall plot utilisé par l'algorithme LIVE pour l'interprétation d'un modèle de type boîte-noire, sur les données "Wine"*

B.3 Modèles de substitution globaux

L'objectif des modèles de substitution globaux ("global surrogate models") est d'approcher aussi précisément que possible les prédictions d'un modèle de machine learning complexe par un modèle plus simple, que l'on sait interpréter. Les modèles que l'on

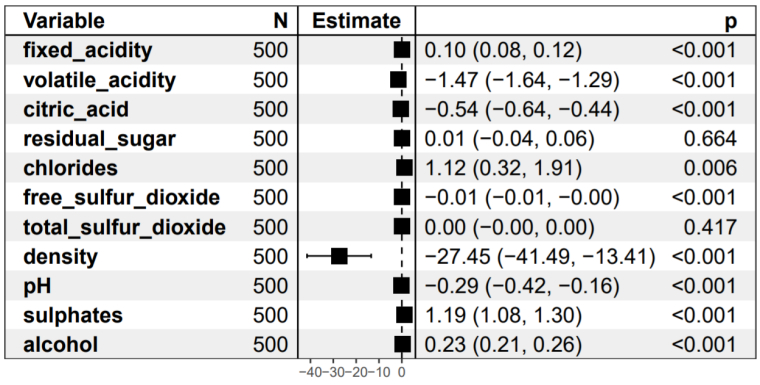

FIGURE *B.4: Forest Plot utilisé dans l'algorithme Live pour expliquer la prédiction d'un modèle*

peut choisir sont par exemple la régression linéaire ou les arbres de décision. Nous allons considérer différentes approches.

B.3.1 Première approche

Notons g la fonction associée au modèle de substitution. On a en particulier :

- Pour la régression linéaire : $g(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$
- Pour les arbres de décision : $g(x) = \sum_{m=1}^{M} c_m \mathbb{I}_{\{x \in F_m\}}$ où c_m est le poids associé à la feuille m, notée F_m, et M le nombre total de feuilles.

Comme il s'agit d'une méthode "agnostique", elle n'utilise que les prédictions faites par le modèle choisi, et non la structure interne du modèle. Les différentes étapes sont alors :

— choix du dataset sur lequel on travaille. On peut par exemple choisir le même dataset que celui utilisé initialement.
— application du modèle de machine learning de type boîte noire sur ce dataset. On appelle ce modèle M_1. On obtient une prédiction $\hat{y_1} = f_1(X)$
— on met en place le modèle de substitution choisi (régression linéaire, arbre de décision...), on calcule ses prédictions sur ce même dataset. On appelle ce modèle M_2. On obtient alors : $\hat{y_2} = f_2(X)$
— on compare les prédictions faites par les deux modèles M_1 et M_2 et on mesure à quel point les prédictions du modèle de substitution sont fidèles à celles du modèle initial.
— on interprète le modèle de substitution à l'aide des outils vus dans la partie précédente sur l'interprétabilité intrinsèque des modèles simples.

Considérons l'exemple de la base de données de Boston vu ci-dessus. On décide d'utiliser comme modèle de substitution un arbre de décision, avec une profondeur maximale de 2. Le modèle initial ajusté est le même que dans la partie précédente, à savoir un Random Forest avec 50 arbres. Les résultats obtenus sont donnés dans le graphique B.5 et sont très facilement interprétables en comparaison du modèle de forêt aléatoire initial.

Il s'agit néanmoins d'une méthode naïve, pour laquelle beaucoup d'informations sont perdues concernant le comportement de la boîte-noire.

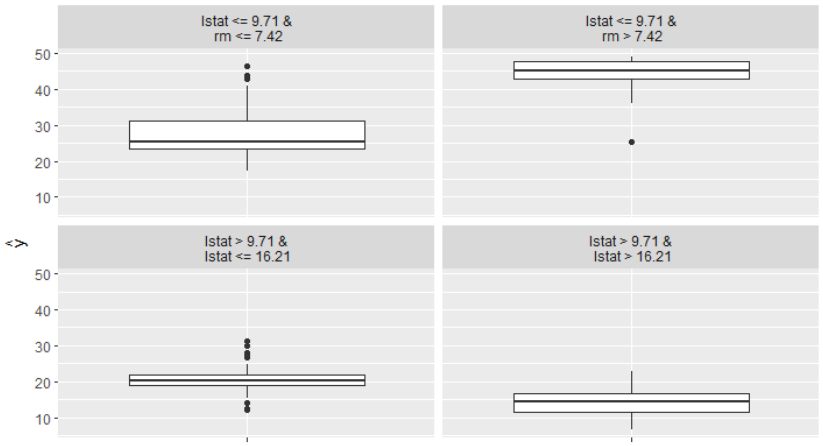

FIGURE *B.5: Arbre de décision comme modèle de substitution global au modèle de Random Forest ajusté sur les données de Boston*

B.3.2 Modèle de substitution global à partir d'une extraction de modèle

L'article [7] *Interpreting Blackbox Models via Model Extraction* de Bastani et Al. (2019) présente une nouvelle méthode de construction d'un modèle de substitution global, au modèle de boîte noire étudié, basée sur des arbres de décision. L'apport de cet article est essentiellement l'algorithme d'extraction des explications d'un arbre de décision, qui échantillonne activement des nouveaux points d'apprentissage afin d'éviter le surapprentissage.

B.3.2.1 Formulation du problème

On se place dans le contexte d'un algorithme de boîte noire (Random Forest, XGBoost, SVM, etc) associé à sa fonction de prédiction $f : \mathbb{X} \to \mathbb{Y}$. Supposons pour simplifier, que $\mathbb{X} = \mathbb{R}^p$ ($p \in \mathbb{N}$), et $\mathbb{Y} = \{1, ..., m\}$ ($m \in \mathbb{N}$) (problème classification). La méthode proposée dans l'article se généralise pour un problème de régression et lorsque l'on dispose de variables catégorielles. On suppose que l'algorithme f apprend sur une base d'apprentissage X_{train}, composée de $n \in \mathbb{N}$ individus. L'idée est de trouver un arbre de décision qui approche au mieux les prédictions de l'algorithme f.

Définissons tout d'abord les contraintes alignées sur l'axe (que l'on appellera plus simplement contraintes par la suite) : il s'agit d'un d'ensemble de la forme $C = (x_i \leq t)$, avec $i \in \{1, ...d\}$, $t \in \mathbb{R}$. Plus généralement, les contraintes se construisent à partir d'unions, d'intersections et de négations, respectivement de la forme : $C_1 \vee C_2$, $C_1 \wedge C_2$ et

$\neg C$ où C, C_1 et C_2 sont des contraintes. L'ensemble des possibles associé à une contrainte C est définie par : $\mathbb{F}(C) = \{X \in \mathbb{X} | x\ satisfait\ C\}$. Considérons un arbre de décision (binaire) $T : \mathbb{X} \to Y$, un noeud interne $N = (N_g, N_d, C)$, constitué d'un sous-noeud droit, un sous-noeud gauche et d'une contrainte. On note N_T la racine de l'arbre. Une feuille N de l'arbre T est assimilée à une fonction : $N : \mathbb{X} \to \mathbb{Y}$. De même pour un noeud :

$$\forall x \in \mathbb{X}, N(x) = \begin{cases} N_g(x)\ si\ x \in \mathbb{F}(C) \\ N_d(x)\ sinon \end{cases} avec\ N = (N_g, N_d, C) \tag{B.1}$$

On a en particulier pour la racine : $T(x) = N_T(x)$. Pour un noeud quelconque $N \in T$, on définit récursivement la conjonction de contraintes C_N de la racine N_T jusqu'au noeud N : $C_{N_T} = VRAI$ et pour tout noeud interne $N = (N_g, N_d, C), C_{N_g} = C_N \wedge C$ et $C_{N_d} = C_N \wedge \neg C$.

Pour mesurer la performance du modèle de substitution, on utilise un ensemble de test X_{test} et une fonction d'erreur $Err(X_{test})$. L'article considère trois cas :

— Pour la régression : l'erreur quadratique moyenne (MSE) définie par

$$Err(X_{test}) = \frac{1}{|X_{test}|} \sum_{x \in X_{test}} (T(x) - f(x))^2$$

— Pour la classification binaire (m = 2), on utilise le score F1. Celui-ci est défini par la moyenne harmonique de la précision (rapport des vrais positifs et du nombre de positifs au total) et de la sensibilité (rapport des vrais positifs sur la somme des vrais positifs et faux négatifs). En d'autres termes, si on note TP, FP, FN respectivement le nombre de vrais positifs, de faux positifs et de faux négatifs sur la base de test, on a la formule $precision = \frac{TP}{TP+FP}$, $sensibilite = \frac{TP}{TP+FN}$. Le score F1 est ainsi donné par la formule :

$$F1(X_{test}) = Err(X_{test}) = \left(\frac{precision^{-1} + sensibilite^{-1}}{2} \right)^{-1} = 2\frac{precision \times sensibilite}{precision + sensibilite}$$

.

— Pour la classification avec au moins 3 classes : on utilise le taux d'erreur classique, à savoir :

$$Err(X_{test}) = \frac{1}{|X_{test}|} \sum_{x \in X_{test}} \mathbb{1}_{\{T(X) \neq f(x)\}}$$

B.3.2.2 Algorithme d'extraction de l'arbre de décision

La première étape de l'algorithme est l'estimation de la distribution de probabilité $\mathbb{P}$ associée à l'ensemble $\mathbb{X}$. Pour cela, on ajuste un modèle de mélange de gaussiennes à partir de $\mathbb{X}_{train}$ en utilisant la méthode d'espérance-maximisation, proposée en 1977 par Dempster et Al. [23]. Soit $K \in \mathbb{N}$ le nombre de mélanges réalisés. La distribution de catégorie du mélange est définie pour $j \in \{1, ..., K\}$ par : $j \sim Categorical(\phi)$ où $\phi \in \mathbb{R}^K$. On rappelle qu'une distribution X de catégorie de paramètre ϕ est définie par : $\forall i \in \{1, ..., K\}, \mathbb{P}[X = i] = \phi_i$. Les distributions de mélange associées sont définies pour

tout $j \in \{1, ..., K\}$ par des lois normales $N(\mu_j, \Sigma_j)$ où $\mu_j \in \mathbb{R}^p$, $\Sigma_j \in \mathbb{R}^{p \times p}$ sont des matrices de variances-covariances diagonales.

Arbre de décision Glouton exact

L'article suggère de construire l'arbre de décision Glouton exact T^* (de taille k choisie initialement), associé au modèle de boîte noire f par une approche similaire à la méthode CART de Breiman et Al. (1984). Tout d'abord, T^* est initialisé par une seule feuille $N_{T^*} = (y)$, où $y \in \{1, ..., m\}$ est la classe majoritaire. Ensuite, on divise chaque feuille de T^* itérativement. Pour chaque feuille N de l'arbre à l'étape considérée, on le remplace par un noeud $N' = (N_g, N_d, C)$, où $N_g = (y_g)$ et $N_d = (y_d)$ sont des feuilles et $C = (x_{i^*} \leq t^*)$ tel que :

$$(i^*, t^*) = \underset{1 \leq i \leq d, t \in \mathbb{R}}{argmax} G(i, t) \tag{B.2}$$

où G est la fonction gain associée à la séparation $C = (x_i \leq t)$, définie par :

$$G(i, t) = -H(f, C_N \wedge (x_i \leq t) - H(f, C_N \wedge (x_i \leq t) + H(f, C_N) \tag{B.3}$$

H est la fonction d'impureté de Gini vue dans les parties précédentes, définie par :

$$H(f, C) = \left(1 - \sum_{y \in \mathbb{Y}} \mathbb{P}_P[f(x) = y|C]^2\right) \mathbb{P}_P[C]$$

Les labels y_g et y_d associés aux deux sous-noeuds créés sont les classes majoritaires conditionnellement aux contraintes de chaque noeuds, c'est-à-dire :

$$y_g = \underset{y \in \mathbb{Y}}{argmax}\ \mathbb{P}_P[f(x) = y|C_N \wedge (x_i \leq t)]\ y_d = \underset{y \in \mathbb{Y}}{argmax} \mathbb{P}_P[f(x) = y|C_N \wedge (x_i > t)] \tag{B.4}$$

On remplace le noeud $N \in T^*$ avec le plus grand gain défini par l'équation B.3.

Estimation de l'arbre de décision Glouton exact

Notons $\hat{T}$ l'arbre de décision Glouton que l'on va estimer. Son calcul repose sur les mêmes étapes que celles vues juste avant, à l'exception des équations B.3 et B.4 qui sont estimées à l'aide de simulation de $n \in \mathbb{N}$ échantillons i.i.d de loi $P|C_N$. En considérant une conjonction de contraintes C, en omettant les redondances possibles dans l'arbre, on peut l'écrire sous la forme : $C = (s_1 \leq x_1 \leq t_1) \wedge ... \wedge (s_p \leq x_p \leq t_p)$. Comme nous avons supposé que P suivait un mélange de K gaussiennes, de densité f_j : de paramètres respectifs $\theta_j = (\mu_j, \Sigma_j)$ et de poids ϕ_j pour $j \in \{1, ..., K\}$, on a la formule :

$$f_P(x) = \sum_{j=1}^{K} \phi_j f_j(x) = \sum_{j=1}^{K} \phi_j \prod_{i=1}^{p} f_{j,i}(x_i)$$

Avec pour tout $i \in \{1, .., p\}$: $f_{j,i} \sim N(\mu_{j,i}, \sigma_{j,i})$ où $\sigma_{j,i} = (\Sigma_j)_{ii}$. La distribution conditionnelle associée est alors :

$$f_{P|C}(x) = \sum_{j=1}^{K} \phi_j \prod_{i=1}^{p} f_{j,i|(s_i \leq x_i \leq t_i)}(x_i)$$

Les lois $f_{j,i|(s_i \leq x_i \leq t_i)}(x_i)$ sont des loi normales tronquées. Un algorithme de simulation de lois normales tronquées est proposé dans l'article [18], et est implémenté en R, dans le package *truncnorm* et la fonction éponyme associée. Comme les gaussiennes sont alignées sur l'axe (car les matrices de variances-covariances sont diagonales) il en résulte la probabilité (non normalisée) de chaque composante par la formule :

$$\tilde{\phi}'_j = \int_{\mathbb{R}} \phi_j \prod_{i=1}^{p} f_{j,i}(x_i) = \phi_j \prod_{i=1}^{p} \left(\Phi(\frac{t_i - \mu_{j,i}}{\sigma_{j,i}}) - \Phi(\frac{s_i - \mu_{j,i}}{\sigma_{j,i}}) \right), 1 \leq j \leq K \tag{B.5}$$

Avec Φ la fonction de répartition de la loi normale centrée réduite. La constante de normalisation est la somme des termes de l'équation B.5 : $Cste = \sum_{j=1}^{K} \tilde{\phi}'_j$, et finalement le vecteur de probabilité $\tilde{\phi}$ des composantes est donné par : $\tilde{\phi} = \frac{\tilde{\phi}'}{Cste}$. Ainsi, la simulation d'un vecteur x de loi $P|C$ est réalisée en deux étapes :

- Simulation de j par : $j \sim Categorical(\tilde{\Phi})$
- Simulation de x_i, issue d'une loi normale tronquée, par : $x_i \sim N(\mu_{j,i}, \sigma_{j,i})|(s_i \leq x_i \leq t_i)$ pour tout $i \in \{1, .., p\}$) (à l'aide de l'article de 2012 *Simulation d'une distribution gaussienne tronquée sur un intervalle fini* de Mazet)

Propriétés de notre arbre estimé

En adaptant les démonstrations utilisées par Domingos et Hulten en 2000, les auteurs de l'article [7] ont prouvé un résultat de convergence de l'arbre estimé $\hat{T}$, à savoir :

$$\hat{T} \underset{n \to \infty}{\to} T^*$$

Ils ont également démontré une propriété intéressante de l'arbre estimé. Pour cela, définissons les notions de ϵ-approximation et d'arbres (ϵ, δ)-exact. Soit T, T' deux arbres de décision, $\epsilon > 0$ et $\delta > 0$.
On dit alors que T est une ϵ-approximation de T' si : $\mathbb{P}_P[T(x) = T'(x)] \geq 1 - \epsilon$.
On dit que T est (ϵ, δ)-exact si $\mathbb{P}[T$ est une ϵ-approximation de $T^*] \geq 1 - \delta$ (probabilité sur la base d'apprentissage $x \sim P$).
Le résultat démontré par Bastani et Al. est : Pour tout $\epsilon > 0$ et $\delta > 0$, il existe un entier naturel n tel que l'arbre extrait $\hat{T}$ construit à partir de n échantillons i.i.d est (ϵ, δ)-exact.

B.3.2.3 Application : évaluation de la méthode

Afin de montrer l'intérêt de cette nouvelle méthode proposée, les auteurs de l'article proposent de comparer la fidélité (précision relative au modèle complexe) sur la base de données Diabetes. Celle-ci qui a pour objectif de prédire si le patient a un risque faible ou élevé d'avoir du diabète (de type II), en fonction de plusieurs caractéristiques telles que les médicaments prescrits ou la démographie. Le modèle de boîte noire utilisé est une forêt aléatoire, à partir duquel on extrait l'arbre de décision par le processus vu ci-dessus (avec $n = 1000$). La mesure de fidélité, qui est ici le score F1, permet de voir si l'arbre de décision extrait reflète le modèle complexe que l'on souhaite comprendre. L'idée est alors

de comparer ce score avec d'autres méthodes déjà existantes, à savoir les arbres CART, les arbres born-again, de Breiman et Al.(1996) et des listes de règles (rule lists). La figure B.6 montre que la méthode d'extraction est la plus performante en terme de fidélité, en comparaison des méthodes déjà existantes.

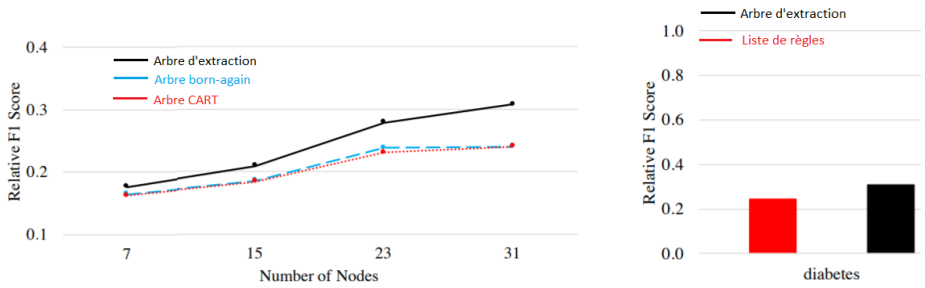

FIGURE *B.6: Fidélité des méthodes d'extraction, CART, born-again et des ensembles de règles sur la base de données de diabète*

Le théorème énoncé ci-dessus suggère que les arbres de décision deviennent de plus en plus stables lorsque le nombre d'échantillons n tend vers l'infini. Les auteurs de l'article ont alors voulu vérifier que ce résultat se vérifiait empiriquement. Leur idée est de toujours utiliser les algorithmes d'extraction et *born-again* puis d'extraire 10 arbres de décision aléatoirement $T_1, ..., T_{10}$. Pour chaque couple d'arbre (T_k, T_h) $(1 \leq k < h \leq 10)$, ils ont compté la fraction des noeuds correspondants de T_k et T_h qui sont égaux. Ceci permet de mesurer la similarité entre les deux arbres. Les résultats obtenus sont donnés dans le tableau B.1. Ils montrent la stabilité des arbres d'extraction en comparaison des arbres born-again.

	Arbre d'extraction	Arbre born-again
$n = 2000$	0.52	0.22
$n = 20000$	0.67	0.25

TABLE *B.1: Mesure de similarité entre les arbres d'extraction et les arbres born-again, suivant la taille n de l'échantillon*

B.4 Explications basées sur les exemples

Les explications basées sur les exemples sont des méthodes d'interprétabilité dont le but est d'expliquer le comportement des modèles de type boîtes-noires, en sélectionnant des instances particulières de la base de données utilisée pour ajuster le modèle. Cette méthode a un intérêt tout particulier lorsque l'on peut représenter les instances de manière à ce qu'elles soient compréhensibles pour un humain. C'est le cas lorsque l'on manipule des images par exemple mais deviennent plus complexes à interpréter pour des données sous forme de tableau par exemple. Pour comprendre les méthodes d'explication

basées sur les exemples, prenons le cas où l'on souhaite trouver la maladie d'un patient à partir de données le concernant. L'idée est alors d'identifier un autre patient avec des symptômes similaires, déjà diagnostiqué auparavant. Ainsi, on peut alors suggérer que le nouveau patient a une maladie proche de celle identifiée plus tôt et faire des tests pour s'en assurer.

B.4.1 Explications contrefactuelles

Une explication contrefactuelle d'une prédiction d'un modèle de boîte noire décrit le plus petit changement de la valeur des variables explicatives qui modifie significativement la prédiction réalisée avant ce changement. Nous expliciterons par la suite le terme "significativement" plus rigoureusement. Par exemple, si on s'intéresse au prix d'une maison à partir de sa surface et du nombre de pièces, on cherche quelle surface et quel nombre de chambres il faudrait pour faire augmenter le prix de la maison de 1000 euros par exemple. Ceci nous fournira alors une nouvelle instance, appelée instance contrefactuelle. Intuitivement, il semble que plusieurs exemples peuvent fonctionner, comme par exemple : une chambre supplémentaire et 3 mètres carrés de plus ou bien une chambre en moins et 15 mètres carrés de plus. Il nous faut alors choisir celui qui nous intéresse le plus. Pour cela, on souhaite que l'instance contrefactuelle vérifie certaines propriétés. Tout d'abord, celle-ci doit être la plus proche possible de l'instance à expliquer. C'est pourquoi il nous faut définir une fonction qui mesure la distance entre deux instances. On souhaite également que la nouvelle instance ait un nombre de features changées le plus faible possible. Enfin, on souhaite que l'instance contrefactuelle possède des valeurs de features réalistes. Plaçons nous dans le cas usuel où on cherche à expliquer une variable $Y \in \mathbb{Y} = \mathbb{R}$ à partir de variables explicatives $X \in \mathbb{X} = \mathbb{R}^p$ ($p \in \mathbb{N}$). Soit x une instance particulière : $x \in \mathbb{R}^p$, $\hat{f}$ la fonction associée au modèle qu'on étudie : $\hat{f}(x)$ fournit une estimation de la valeur de la variable cible y à partir des variables explicatives x. Supposons que l'on dispose d'un échantillon de taille $n \in \mathbb{N}$: $(x_{i,j})_{1\leq i\leq n, 1\leq j\leq p}, (y_i)_{1\leq i\leq n}$ On définit alors la fonction de perte, comme suggérée dans l'article [73] de Watcher et al. en 2017 :

$$L(x, x', y', \lambda) = \lambda(\hat{f}(x') - y')^2 + d(x, x'),\ x' \in \mathbb{X}, y' \in \mathbb{Y}, \lambda \in \mathbb{R} \tag{B.6}$$

où $d : \mathbb{X}^2 \to \mathbb{R}^+$ est une fonction de distance, y' la nouvelle valeur cible désirée, et λ un paramètre de contrôle. L'objectif est de trouver l'instance x' qui minimise cette fonction de perte, i.e.

$$x' = \underset{x' \in \mathbb{X}}{argmin} L(x, x', y', \lambda) \tag{B.7}$$

Le paramètre λ va définir si l'on souhaite favoriser les instances x' qui fournissent une sortie par $\hat{f}$ proche de y' choisi (λ grand) ou favoriser les instances x' proches de l'instance initiale x (λ petit). Une alternative proposée, pour ne pas avoir à choisir le paramètre λ est de résoudre le problème d'optimisation :

$$arg \min_{x' \in \mathbb{X}} \max_{\lambda \in \mathbb{R}} L(x, x', y', \lambda) \tag{B.8}$$

L'article indique que l'algorithme utilisé en pratique pour résoudre ce problème d'optimisation est ADAM (c.f. article [40]). Il faut à présent choisir une fonction de distance

adaptée. L'article suggère la fonction Manhattan pondérée par l'inverse de la médiane de la valeur absolue de la déviation, i.e. :

$$d(x, x') = \sum_{j=1}^{p} \frac{|x_j - x'_j|}{MAD_j} \tag{B.9}$$

où

$$\forall j \in \{1, ..., p\},\ MAD_j = \underset{i \in \{1,...,n\}}{mediane}(|x_{i,j} - \underset{l \in \{1,...,n\}}{mediane}(x_{l,j})|) \tag{B.10}$$

Ce choix est dû aux différentes propriétés vérifiées par cette distance, à savoir que deux points sont d'autant plus proches les uns des autres que le nombre de paramètres qui différent entre les deux est faible. De plus, cette distance est plus robuste face aux valeurs aberrantes. La procédure proposée par l'article pour fournir des instances contrefactuelles est détaillée dans l'algorithme suivant :

- Choisir une instance $x \in \mathbb{X}$ à expliquer, une output y' désirée, une tolérance ϵ, une valeur initiale (faible) de λ
- Tirer une instance x' aléatoirement comme instance contrefactuelle de départ.
- Optimiser la fonction de perte L avec l'instance contrefactuelle initiale comme point de départ.
- Tant que $|\hat{f}(x') - y'| > \epsilon$:
  - Augmenter la valeur de λ
  - Optimiser la fonction de perte avec l'instance contrefactuelle actuelle comme point de départ
  - x' devient la nouvelle instance contrefactuelle qui minimise la fonction de perte.
- Répéter les étapes 2 et 3 : l'algorithme renvoie alors la liste des instances contrefactuelles qui minimisent la perte L.

Une variante de cet algorithme est proposée dans l'article [44] de Laugel et Al. de 2017. Celui-ci utilise des sphères croissantes. L'idée est de définir une sphère la plus petite possible autour de l'instance d'intérêt x et de la faire grandir jusqu'à ce qu'elle contienne une instance x' qui fournit la prédiction désirée y'.

B.4.2 Prototypes et points critiques

B.4.2.1 Définition et méthode de calcul

Dans cette partie, on s'intéresse à la notion de points appelés "prototypes" ou "critiques". Un prototype est une instance d'une base de données utilisée pour ajuster notre modèle qui est représentative de la base entière tandis qu'une critique est une instance mal représentée par l'ensemble des prototypes de la base. Sur le graphique B.7, on donne un exemple de prototypes et de critiques dans le cas de données avec deux variables explicatives.

Les prototypes seuls sont rarement suffisants pour représenter l'essentiel de la complexité des données, c'est pourquoi on construit également des critiques pour avoir une meilleure compréhension des données qu'on analyse. L'algorithme des k-médoïdes était

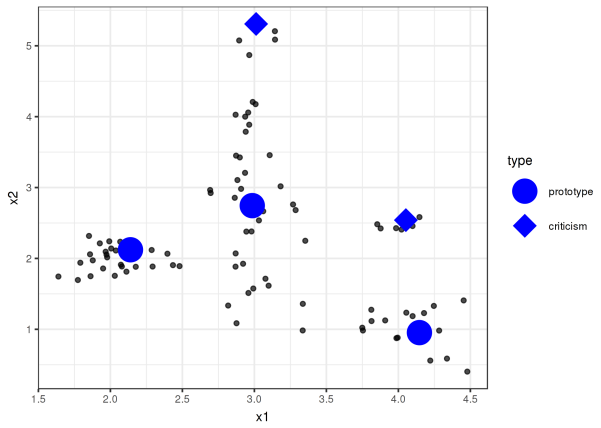

FIGURE *B.7: Exemples de prototypes et de critiques pour une distribution de données avec deux variables x_1 et x_2*

couramment utilisé pour trouver les prototypes, aussi appelés médoïdes, correspondant aux représentants les plus centraux d'une classe de points (clusters). Cet algorithme repose sur la minimisation de l'erreur quadratique moyenne qui est la distance entre les points de la classe et le point central. Il est très proche de l'algorithme des k-moyennes (ou k-means) mais est plus robuste vis à vis des données aberrantes. L'article [39] de Been et Al. (2016) propose une variante, appelée MMD-critic, qui combine la recherche de prototypes et de critiques. L'idée est de comparer la distribution des données avec les prototypes sélectionnés. Une fois ceux-ci trouvés, on cherche les critiques dont la distribution est la plus éloignée de celle des prototypes. Il nous faut alors introduire une mesure de la différence entre deux distributions de points. Soit $X = (x_i)_{1 \leq i \leq n} \in (\mathbb{R}^p)^n$ et $(y_i)_{1 \leq i \leq n}$ notre jeu de données de taille $n \in$. Soit $P = (P_i)_{1 \leq i \leq m}$ un ensemble de $m \in \mathbb{N}$ prototypes choisis. Alors on définit la fonction MMD (Maximum Mean Discrepancy) entre X et P par :

$$MMD^2(X, P) = \frac{1}{n} \sum_{i,j=1}^{n} k(x_i, x_j) - \frac{2}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, z_j) + \frac{1}{m} \sum_{i,j=1}^{m} k(z_i, z_j) \qquad \text{(B.11)}$$

Elle correspond à la moyenne maximale de contradiction. Dans cette définition, k : $\mathbb{R}^p \text{x} \mathbb{R}^p \to R^+$ est une fonction noyau, telle que RBF, définie par :

$$\forall (x, x') \in \mathbb{R}^p \times \mathbb{R}^p, k(x, x') = exp(-\frac{1}{2\sigma^2} ||x - x'||^2) \qquad \text{(B.12)}$$

avec un paramètre d'échelle $\sigma > 0$.

La première étape de l'algorithme est de trouver l'ensemble (de taille m choisie) de prototypes qui représente le mieux les données. Cela consiste à choisir l'ensemble $\tilde{P}$ qui minimise la fonction MMD :

$$\tilde{P} = \underset{P=(p_i)_{1 \leq i \leq m} \in (\mathbb{R}^p)^m}{argmin} MMD(X, P) \qquad \text{(B.13)}$$

On part donc d'une liste vide de prototypes, et tant qu'on a pas m prototypes, on ajoute à la liste le point qui minimise la MMD. Une fois cette liste de m prototypes trouvés, on cherche les critiques. Pour cela, on définit la fonction témoin, qui indique comment deux distributions varient à un point particulier :

$$\forall x \in \mathbb{R}^p,\ temoin(x) = \frac{1}{n}\sum_{i=1}^{n} k(x, x_i) - \frac{1}{m}\sum_{j=1}^{m} k(x, z_j) \tag{B.14}$$

Pour trouver les critiques, on cherche les valeurs extrêmes (en valeur absolue) de la fonction témoin, avec éventuellement un terme de régularisation à intégrer. Le cas où la fonction témoin est positive en un point x signifie que la distribution des prototypes surestime la distribution des données au point x tandis que le cas où elle est négative correspond à une sous-estimation.

B.4.2.2 Utilisation des prototypes et critiques pour l'interprétation d'un modèle

Nous pouvons à présent nous demander comment les prototypes et critiques peuvent servir à l'interprétabilité d'un modèle. Pour cela, nous utilisons la procédure suivante :

- Trouver les prototypes et les critiques avec l'algorithme MMD-critic.
- Entraîner un modèle de Machine Learning sur la base de données initiale.
- Prédire les sorties fournies par le modèle pour les prototypes et les critiques.
- Analyser les prédictions, notamment en trouvant les cas où l'algorithme se trompe. On obtient des exemples qui représentent bien nos données initiales et peuvent nous aider à trouver les faiblesses du modèle.

B.5 Instances influentes et fonction d'influence

Dans cette partie, nous détaillons une méthode qui utilise les fonctions d'influence (technique classique des statistiques robustes) pour expliquer les prédiction d'un modèle de type boîte noire. Cette méthode a été introduite dans l'article [41] de Koh et Al. (2017). Contrairement aux autres méthodes d'interprétation qui considère le modèle entraîné comme fixe, les fonctions d'influence le traitent comme une fonction des exemples d'entraînement qui ont servi à l'ajustement du modèle.

B.5.1 Calcul mathématique de la fonction d'influence

On considère un espace de départ $\mathbb{X} = \mathbb{R}^p$ et un espace de sortie $\mathbb{Y}$. On dispose de $n \in \mathbb{N}$ points d'entrainements $z_1, ..., z_n$ où $\forall i \in \{1, ..., n\}, z_i = (x_i, y_i) \in \mathbb{X} \times \mathbb{Y}$. On considère un modèle, pour lequel on cherche le paramètre $\hat{\theta} \in \Theta$ qui minimise le risque empirique, c'est-à-dire :

$$\hat{\theta} = \underset{\theta \in \Theta}{argmin} \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta) \tag{B.15}$$

où pour un point z et un paramètre θ, $L(z, \theta)$ est une fonction de perte. On suppose que le risque empirique est deux-fois différentiable et strictement convexe en θ. Notre but est de comprendre l'effet des points d'apprentissage sur la prédiction du modèle. Etudions d'abord le cas où l'on entraîne le modèle sur la même base de données, mais en retirant un point z. Notons alors $\hat{\theta}_{-z}$ les nouveaux paramètres estimés par le modèle, c'est-à-dire :

$$\hat{\theta}_{-z} = \underset{\theta \in \Theta}{argmin} \sum_{z_i \neq z} L(z_i, \theta) \tag{B.16}$$

Définissons également : $\hat{\theta}_{\epsilon,z}$ le paramètre estimé par le modèle lorsque l'on pondère le point z par un coefficient $\epsilon > 0$ en plus, c'est-à-dire :

$$\hat{\theta}_{\epsilon,z} = \underset{\theta \in \Theta}{argmin} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta) \tag{B.17}$$

Cook et Weisberg en 1982 dans l'article [19] *Residuals and influence in regression* ont défini la fonction d'influence de la surpondération de z sur le paramètre $\hat{\theta}$:

$$I_{up,params}(z) = \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}|_{\epsilon=0} \tag{B.18}$$

Ils ont également prouvé, sous les hypothèses de double-différentiabilité et de stricte convexité du risque empirique, la relation suivante :

$$I_{up,params}(z) = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{B.19}$$

où $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L(z_i,)$ est la matrice hessienne du risque empirique.
Prouvons ce résultat. Tout d'abord, la stricte convexité du risque empirique implique que $H_{\hat{\theta}}$ est une matrice définie positive (et donc inversible). Définissons : $\Delta\epsilon = \hat{\theta}_{\epsilon,z} - \hat{\theta}$. Comme $\hat{\theta}$ ne dépend pas de ϵ, on a la relation :

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta\epsilon}{d\epsilon} \tag{B.20}$$

Sachant que $\hat{\theta}_{\epsilon,z}$ minimise la fonction $\theta \mapsto R(\theta) + \epsilon L(z, \theta)$, on a : $\nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z}) = 0$. En utilisant le développement de Taylor à l'ordre 1, sachant que $\hat{\theta}_{\epsilon,z} \underset{\epsilon \to 0}{\to} \hat{\theta}$, on a l'approximation, lorsque $\epsilon \to 0$:

$$0 = [\nabla R(\hat{\theta}) + \epsilon \nabla L(z, \hat{\theta}] + [\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z, \hat{\theta})]\Delta\epsilon + o(||\Delta\epsilon||) \tag{B.21}$$

Comme $\hat{\theta}$ minimise R, $\nabla R(\hat{\theta}) = 0$ et en ne gardant que les termes en $O(\epsilon)$, il en découle l'approximation :

$$\Delta\epsilon = -[\nabla^2 R(\hat{\theta}) + \cancel{\epsilon \nabla^2 L(z, \hat{\theta})}]^{-1} \epsilon \Delta L(z, \hat{\theta}) \tag{B.22}$$

Soit, comme $H_{\hat{\theta}} = \nabla^2 R(\hat{\theta})$,

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}|_{\epsilon=0} = I_{up,params}(z) = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{B.23}$$

Remarquons que retirer le point z comme point d'observation revient à choisir la pondération $\epsilon = -\frac{1}{n}$. On peut alors approcher linéairement le changement des paramètres après avoir retiré le point z :

$$\hat{\theta}_{-z} - \hat{\theta} \simeq -\frac{1}{n} I_{up,params}(z) \tag{B.24}$$

sans avoir à ré-entraîner le modèle.
Définissons à présent l'influence de surpondérer z sur la fonction de perte à un point de test z_{test} par la formule :

$$I_{up,loss}(z, z_{test}) = \frac{dL(z_{test}, \hat{\theta}_{\epsilon,z}}{d\epsilon}|_{\epsilon=0} \tag{B.25}$$

En appliquant la règle de la chaîne, il en découle que :

$$I_{up,loss}(z, z_{test}) = \nabla_\theta L(z_{test}, \hat{\theta})^T \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}|_{\epsilon=0} \tag{B.26}$$

$$= -\nabla_\theta L(z_{test}, \hat{\theta})^T H_\theta^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{B.27}$$

B.5.2 Interprétation de la formule

Analysons la formule donnée par l'équation B.27. Tout d'abord, on observe que la fonction d'influence est proportionnelle à la norme du gradient. Cela semble cohérent avec le fait qu'un gradient plus élevé aura une plus grande influence sur les paramètres du modèle et donc sur la perte du test. On remarque également que la forme obtenue est assez familière en mathématiques, à savoir une matrice (définie positive) avec un vecteur de chaque côté. Celle-ci introduit une notion de "similarité" : plus les points sont proches plus ils seront influents. Cela est cohérent également avec le fait qu'un exemple de test sera plus influencé par des exemples qui sont proches.

B.5.3 Calcul numérique de la fonction d'influence

La formule donnée par l'équation B.27 entraîne pour n points d'entraînement et $\theta \in \mathbb{R}^p$ une complexité de calcul en $O(np^2 + p^3)$, ce qui est trop coûteux notamment pour des modèles de Deep Learning pouvant posséder plus d'un million de paramètres. Un deuxième problème se pose : on souhaite en général calculer $I_{up,loss}(z_i, z_{test})$ pour chaque point $z_i,\ 1 \leq i \leq n$ de la base. L'objectif est alors de trouver des méthodes d'approximation pour accélérer le calcul. L'idée dans l'article est d'éviter le calcul explicite de $H_{\hat{\theta}}^{-1}$ et d'utiliser le produit Hessienne-vecteur implicite (HVP) pour approcher précisément :

$$s_{test} = H_{\hat{\theta}}^{-1} \nabla_\theta L(z_{test}, \hat{\theta}) \tag{B.28}$$

pour ensuite calculer :

$$I_{up,loss}(z, z_{test}) = -s_{test} \nabla_\theta L(z, \hat{\theta}) \tag{B.29}$$

Cela permet de calculer s_{test} une seule fois et de l'utiliser pour calculer $-s_{test}\nabla_\theta L(z_i, \hat{\theta}), 1 \leq i \leq n$. Deux méthodes sont proposées dans l'article, permettant le calcul de la Hessienne en $O(p)$: la méthode du gradient conjugué et une estimation stochastique que nous ne détaillons par ici.

B.5.4 Cas général

Dans le cas le plus général, les hypothèses de double-différentialité du risque empirique et de matrice hessienne définie positive ne sont pas vérifiées et l'équation B.27 ne s'applique pas. Les auteurs de l'article utilisent des astuces afin de se ramener au cas où ces hypothèses sont vérifiées. Si la convexité n'est pas vérifiée, on modifie légèrement la matrice hessienne et on la remplace par la matrice $H_{\hat{\theta}} + \lambda I_p$. Ceci revient en réalité à ajouter un terme de régularisation $\mathbb{L}^2$. Dans le cas où la double-différentiabilité du risque empirique n'est pas vérifiée, on l'approxime par une fonction qui est elle différentiable. Par exemple, si la fonction à minimiser est $s \mapsto max(0, 1 - s)$ (non différentiable), semblable à la fonction ReLU utilisée dans les réseaux de neurones, on utilise un lissage de celle-ci, par exemple par la fonction $s \mapsto t\ log(1 + exp(\frac{1-s}{t}))$ (avec t un hyperparamètre à définir).

L'article montre que les résultats fournis par cette méthode de fonction d'influence sont également très bons lorsque les hypothèses du théorème ne sont pas vérifiées et peut donc s'appliquer dans le cas général. Finalement, la méthode d'interprétation par les fonctions d'influence permet, en plus de mieux comprendre le modèle, de trouver des labels mal étiquetés et de créer des exemples d'entraînement contradictoires.

Annexe C

Un exemple de cadre généralisé des méthodes d'interprétation des modèles : le cadre SIPA

La majorité des techniques d'interprétation des modèles de Machine Learning repose sur les mêmes principes. Au lieu d'essayer de comprendre le comportement interne des boîtes noires, généralement non linéaires, on évalue l'impact sur la prédiction d'un changement des entrées. L'article [64] *Sampling, Intervention, Prediction, Aggregation : A Generalized Frameword for Model Agnostic Interpretations* propose un cadré généralisé pour ces méthodes d'interprétation agnostiques au modèle : Sampling, Intervention, Prediction and Aggregation (SIPA). Comme dans la majorité des applications réelles, le nombre de données disponibles est grand et l'algorithme mis en place est coûteux en terme de calcul, la première étape qui est généralement réalisée est l'échantillonnage (*Sampling*). Celle-ci permet de réduire le temps de calcul en ne sélectionnant qu'une plage de données, plutôt que la base entière. L'étape qui suit est l'*Intervention* qui consiste à manipuler les données disponibles, pour changer les prédictions faites par la boîte noire. La valeur des variables est soit fixée grâce aux distributions marginales observées, comme pour les courbes ICE, PDP ou les valeurs de Shapley, soit fixée à des valeurs non observées, notamment pour les méthodes basées sur des différences finies comme la courbe ALE. Ensuite, l'étape *Prediction* consiste à prédire, à l'aide d'un modèle de boîte-noire déjà ajusté, sur les données issues de la phase Intervention précédente. Une fois ces prédictions réalisées, celles-ci sont agrégées lors de l'étape *Aggregation*. A l'issue de cette étape, l'interprétation qui en découle est généralement globale. Pour bénéficier d'interprétations locales, c'est-à-dire au niveau d'une observation particulière, il faut extraire les résultats de la phase *Prediction*, avant que l'agrégation ne soit réalisée. La figure C.1 résume ces différentes étapes de SIPA dans le cadre de méthodes d'interprétation basées sur l'effet des variables, comme le PDP, l'ALE ou les courbes ICE.

L'article montre que ce cadre SIPA s'applique également aux méthodes d'importance des variables.

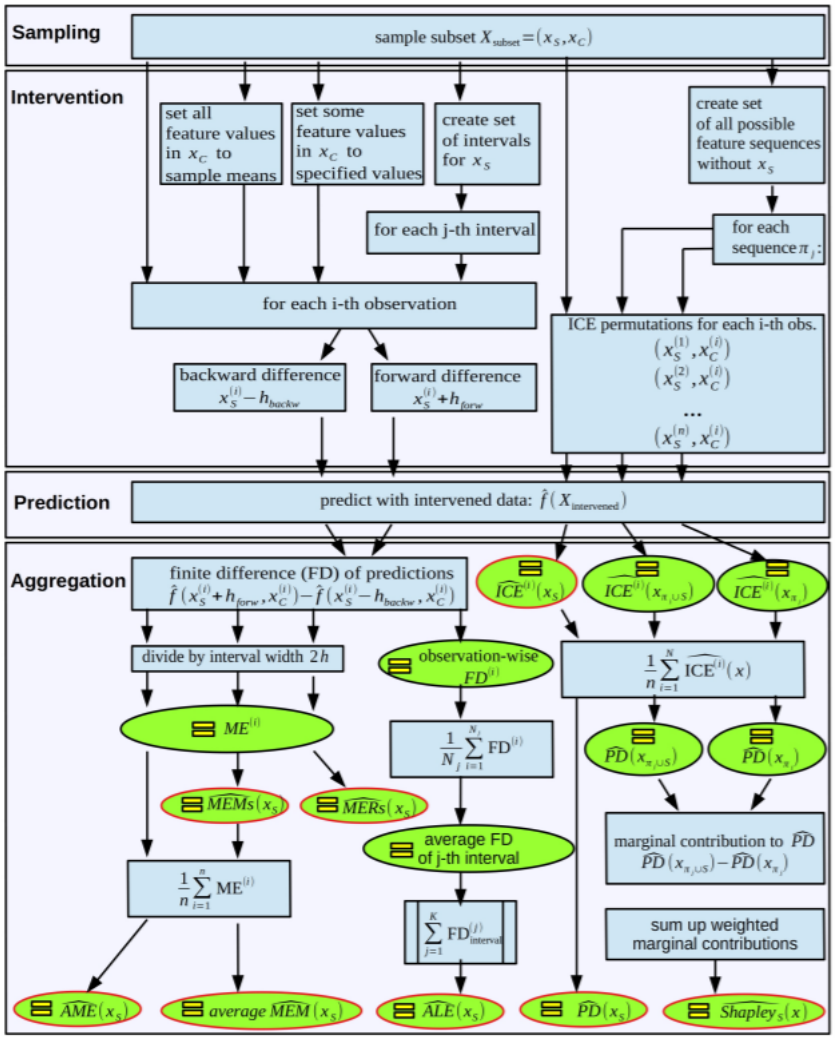

FIGURE *C.1: Cadre SIPA des méthodes d'interprétation basées sur l'effet des variables (issu de l'article [64])*

Annexe D

Mesure de robustesse des méthodes d'interprétation

La robustesse des explications fournies est une des propriétés désirées dans la recherche d'interprétabilité. Celle-ci peut être définie comme : des entrées similaires devraient être associées à des explications similaires. L'article [D] *On the Robutness of Interpretability Methods* part de ce postulat pour définir une métrique pour quantifier la robustesse des méthodes d'interprétation. La motivation de cet article provient essentiellement du fait que les méthodes usuelles, telles que LIME et Shap, ne sont pas nécessairement stables à de petites perturbations des données. L'article fournit l'exemple d'un SVM linéaire, qui est globalement stable et celui d'un réseau de neurones plus complexe qui varie considérablement selon le voisinage de l'entrée. Pour mesurer cette robustesse, les auteurs utilisent la continuité de Lipschitz, qui permet de quantifier les changements relatifs dans la sortie, suivant l'entrée. Etant donné que les méthodes d'interprétation les plus répandues sont locales, i.e. au niveau d'une instance, il faut définir une notion de continuité de Lipschitz locale. Ils proposent la définition suivante : Soit $n, m \in \mathbb{N}^*, \mathbb{X} \subset \mathbb{R}^n, f : \mathbb{X} \to \mathbb{R}^m$ est localement Lipschitz, si :

$$\forall x_0 \in , \exists \delta > 0, \exists L \in \mathbb{R}, ||x - x_0|| < \delta \implies ||f(x) - f(x_0)|| \leq L||x - x_0|| \tag{D.1}$$

L'article souligne que contrairement au critère de Lipschitz global, les quantités δ et L de la définition précédente sont dépendantes du point x_0. On applique alors ceci au cas où f est la fonction associée à la méthode d'interprétabilité étudiée. L'idée est de calculer la constante L "optimale", et cette grandeur nous donnera une indication sur la robustesse de l'explication fournie. Cette quantité est rarement connue à priori, c'est pourquoi il faut l'estimer à l'aide du problème d'optimisation suivant :

$$\hat{L}(x) = \underset{\tilde{x} \in B_\epsilon(x)}{argmax} \frac{||f(x) - f(\tilde{x})||_2}{||x - \tilde{x}||_2} \tag{D.2}$$

Certains problèmes peuvent alors être rencontrés, comme :

— La fonction f, associée à la méthode d'inteprétabilité étudiée, n'est pas nécessairement différentiable, empêchant le calcul du gradient.

— Le temps de calcul pour évaluer la fonction f peut être très long, ce qui impose de réaliser des approximations.
— Il faut réaliser un choix sur la valeur de ϵ.

Dans le cas d'une base de données $X = (x_i)_{1 \leq i \leq n}$, on peut définir une notion empirique de la stabilité, basée sur un voisinage de taille finie. Cette définition est alors plus faible que la précédente, mais permet d'être calculée facilement étant donné que l'on travaille sur un échantillon de taille finie. Pour cela, on définit alors, pour un $\epsilon > 0$ donné : $N_\epsilon(x) = \{\tilde{x} \in X, ||x - \tilde{x}|| \leq \epsilon\}$ et la quantité L précédente devient alors :

$$\hat{L}_X(x) = \underset{\tilde{x} \in N_\epsilon(x)}{argmax} \frac{||f(x) - f(\tilde{x})||_2}{||x - \tilde{x}||_2}$$

Il est clair que dans cette définition de la valeur L, plus celle-ci est faible, plus la méthode d'interprétabilité étudiée est considérée stable. Dans l'article [D], les auteurs ont étudié différentes méthodes d'interprétation des modèles, dont LIME et Shap, sur des bases de données de classification de l'UCI telles que *wine* ou *leukemia*, et également sur la base de données de régression *Boston*. Ils sont arrivés à la conclusion que même de petites perturbations, qui ont un effet minimal (voir inexistant) sur la prédiction du modèle étudié, ont un effet considérable dans la majorité des cas sur l'explication fournie par la méthode d'interprétation.

Annexe E

DALEX : package R et méthodologie pour interpréter un modèle

Dans cette partie, nous reprenons les idées introduites dans l'article [11] *DALEX : Explainers for Complex Predictive Models in R* de P. Biecek. L'idée de cet article est de présenter un nouveau package de R, appelé DALEX, qui fournit des outils pour l'interprétabilité des modèles de Machine Learning. L'auteur rappelle tout d'abord l'importance de l'interprétabilité des modèles de type boîte noire. En effet, les modèles prédictifs sont utilisés dans de nombreux domaines que ce soit, en médecine, banque ou assurance. Leur complexité et leur précision ne cesse d'augmenter au détriment de leur compréhension. Citons par exemple l'algorithme XGBoost, apparu récemment, qui est très répandu lors des compétitions Kaggle. Ses nombreux hyper-paramètres permettent d'obtenir les meilleures performances ce qui nuit à la compréhension du modèle. L'article de Johansson et Al. (2011) souligne justement le compromis nécessaire entre la précision du modèle et son interprétabilité.

Le créateur du package DALEX insiste sur les intérêts de l'interprétabilité des modèles, notamment avec les points suivants :

— *Amélioration du modèle* : les méthodes d'interprétabilité permettent de souligner les zones de faiblesse de l'algorithme, et donc d'ajuster le modèle en conséquence pour obtenir de meilleures performances.

— *Confiance* : Si l'algorithme est utilisé pour assister des personnes dans leur activité, comprendre les facteurs clés de la prédiction réalisée peut s'avérer très important

— *Dette cachée* : Le manque d'interprétabilité d'un modèle peut conduite à une dette cachée, comme indiqué dans l'article de Scully et Al. (2015 *Hidden Technical Debt in Machine Learning Systems.* En effet, bien que les performances initiales du modèle soient très élevées, les performances réelles du modèle peuvent se détériorer très vite. Ceci nous encourage de disposer d'outils pour expliquer le modèle.

— *Amélioration de la connaissance dans le domaine* : Bien qu'il puisse sembler suffisant de disposer d'un modèle avec un grand pouvoir prédicitif, son utilisation ne permet d'améliorer les connaissances dans le domaine sur lequel porte les données. Des modèles interprétables peuvent fournir des découvertes intéressantes.

Le Package DALEX propose des outils pour l'inteprétabilité des modèles à travers deux catégories. Tout d'abord, la première basée sur la compréhension du modèle et l'autre sur la compréhension des prédictions.

E.1 Compréhension du modèle

L'objectif de la première catégorie est de répondre aux questions suivantes : Notre modèle est-il bon ? Quelles sont les variables les plus importantes ? Quelles variables sont liées à la réponse du modèle ? Pour se faire, trois fonctions peuvent être utilisées.

— *Analyse de la performance des modèles* : en général, la performance d'un modèle est définie à l'aide d'un unique facteur, telle que l'erreur quadratique moyenne pour la régression, et le taux d'erreur pour la classification. Pour mieux comprendre le modèle, il est intéressant d'utiliser une statistique descriptive plus détaillée. Par exemple, la courbe ROC pour la classification binaire fournit davantage d'informations qu'un taux d'erreur. Il existe une adaptation de la courbe ROC pour des modèles de régression, proposée par J. Hernandez-Orallo dans l'article *ROC Curves for Regression*. DALEX propose une étude des résidus fournis par le modèle étudié. Par exemple, on peut comparer les résidus (en valeur absolue) entre deux modèles. Bien que deux algorithmes présentent une erreur quadratique moyenne égale, il se peut que la distribution des résidus soit totalement différente. L'article fournit l'exemple d'un modèle de régression linéaire et de forêt aléatoire qui fournissent la même erreur de prédiction. Cependant, en analysant les résidus, on remarque que 90 % des résidus du modèle Random Forest sont plus petits que ceux du modèle linéaire, ce qui nous conduira à choisir plutôt le modèle de forêt aléatoire. Ces conclusions ont été fournies à l'aide des graphiques de la figure E.1.

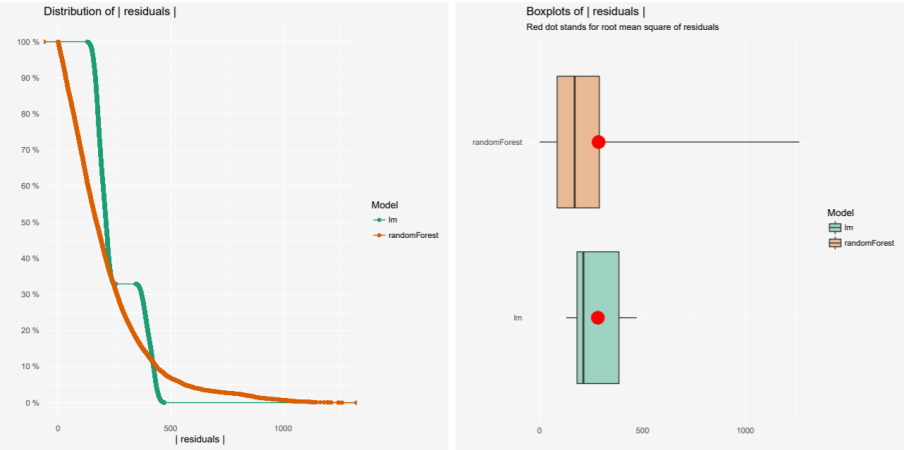

FIGURE *E.1: Analyse des résidus (en valeur absolue) pour un modèle de forêt aléatoire et un modèle linéaire. A gauche, on observe la distribution empirique et à droite la boîte à moustache.*

— *Effet d'une variable sur le modèle* Cette partie regroupe les méthodes d'analyse graphique détaillées dans les parties précédentes, à savoir le graphique PDP, l'ALE et également le Marging Path Plot (MPP) pour les variables catégorielles.
— *Importance des variables* Cette partie reprend la méthode indépendante du modèle de calcul d'importance des variables proposée dans l'article [25] par Fisher et Al. (2018).

E.2 Compréhension de la prédiction

La deuxième catégorie de méthodes pour l'interprétabilité des modèles repose sur la compréhension des prédictions. On peut séparer ces méthodes en deux sous-catégories :

— *Robustesse de la prédiction* : l'idée est de comprendre comment la réponse du modèle change par rapport à la valeur d'une seule variable. Cette méthode est proche du graphique de PDP, sauf que celui-ci repose sur une seule instance. Ce dernier s'appelle Ceteris Paribus Plot (CPP) ou What If Plot. Considérons l'exemple d'un client qui possède un mauvais score en vue de l'obtention d'un crédit et se demande pourquoi le score est si bas et voudrait comprendre comment augmenter son score. Le CPP fournit plusieurs scénarios possibles en changeant la valeur d'une variable, en gardant toutes les autres constantes.
— *Attribution des variables* : l'idée de cette catégorie de méthodes est de fournir un score à chaque variable correspondant à son poids dans la prédiction d'une instance donnée. Les techniques proposées sont BreakDown, LIME et Shap.

Un résumé des différentes méthodologies pour l'interprétabilité d'un modèle est donné par la figure E.2.

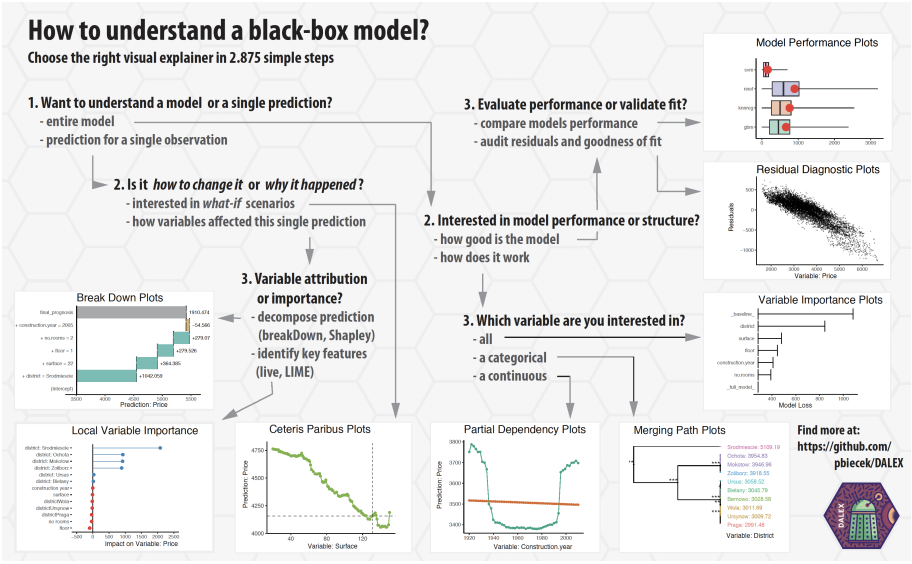

FIGURE *E.2: Méthodologie proposée par DALEX pour comprendre un modèle de type boîte noire*

Bibliographie

[1] K. Aas, M. Jullum, A. Loland *Explaining Individual Predicitions When Features Are Dependant : More Accurate Approximations To Shapley Values* arXiv :1903.10464, 2019

[2] A. Adadi, M. Berrada *Peeking Inside the Black-Box : A Survey on Explainable Artificial Intelligence (XAI)* IEEE Access, 2018. 6 : p. 52138-52160, 2018

[3] P. Ailliot. *Cours de Modèle Linéaire Gaussien (EURIA)* 2017

[4] J. M. Aouizerate. *Alternative neuronale en tarification santé.* Ressources Actuarielles, 2010

[5] D.W Apley. *Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.* arXiv :1612.08468, 2016

[6] P. L. Bartlett, D.J Foster, M. Telgarsky. *Spectrally-normalized margin bounds for neural networks.* arXiv :1706.08498, 2017

[7] A. Bastani, C. Kim, H. Bastani *Interpreting Blackbox Models via Model Extraction* arXiv :1705.08504, 2019

[8] R. Bellina. *Méthodes d'apprentissage appliquées à la tarification non-vie.* Ressources Actuarielles, 2014

[9] P. Biecek, A. Sitko *The Merging Path Plot : adaptive fusing of k-groups with likelihood-based model selection* arXiv :1709.04412, 2017

[10] P. Biecek, M. Staniak *Explanations of model predictions with live and breakDown packages* arXiv :1804.01955, 2018

[11] P. Biecek *DALEX : Explainers for Complex Predictive Models in R* arXiv :1806.08915, 2018

[12] V. Blanchot. *Brève histoire de l'intelligence artificielle.* Siècle Digital, 2018

[13] K. Boukhetala, M. Yahiaoui, T. Laadjel. *Une approche de tarification en assurance automobile par Réseaux de Neurones.* DMAS, 2007

[14] L. Breiman, J. Friedman, C. J. Stone, R.A. Olshen. *Classification and Regression Trees.* Mathematics, 1984

[15] M. Britton *VINE : Visualizing Statistical Interactions in Black Box Models* arXiv :1904.00561, 2019

[16] F. Buchner, J. Wasem, S. Schillo *Regression trees identify relevant interactions : can this improve the predictive performance of risk adjustment ?* Health economics, 2017

[17] G. Casalicchio, C. Molnar, B. Bisch *Visualizing the Feature Importance for Black Box Models* arXiv :1804.06620, 2018

[18] N. Chopin *Fast simulation of truncated Gaussian distributions* Statistics and Computing, vol. 21, p. 275–288, 2011

[19] R. D. Cook, S. Weisberg. *Residuals and influence in regression.* New York : Chapman and Hall, 1982

[20] D. Cuny. *Le digital et les Gafa, premier risque pour les assureurs.* La Tribune, 2017

[21] J. Cupe *L'interprétabilité de l'IA – Le nouveau défi des data scientists* ActuIA 2018

[22] A. S. Dalayan. *Cours d'Apprentissage et Data Mining (Ensae PariTech).*

[23] A.P. Dempster, N.M. Laird, D. Rubin *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society. Series B (Methodological), 1977

[24] Y. Freund, R. Schapire, *A desicion-theoretic generalization of on-line learning and an application to boosting.* 1997

[25] A. Fisher. *All Models are Wrong but Many are Useful : Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance.* arXiv :1801.01489, 2018

[26] J. H. Friedman. *Greedy Function Approximation : A Gradient Boosting Machine.* The Annals of Statistics, 2001

[27] J. H. Friedman, B. E. Popescu. *Predictive learning via rule ensembles.* arXiv :0811.1679, 2008

[28] R. Gauville. *Projection du ratio de solvabilité : des méthodes de machine learning pour contourner les contraintes opérationnelles de la méthode des SdS.* Ressources Actuarielles, 2017

[29] B. Ghattas *Prévisions des pics d'ozone par arbres de régression, simples et agrégés par bootstrap* Revue de statistique appliquée, 1999

[30] L. H. Gilpin *Explaining Explanations : An Overview of Interpretability of Machine Learning* arXiv :1806.00069, 2019

[31] A. Goldstein. *Peeking Inside the Black Box : Visualizing Statistical Learning with Plots of Individual Conditional Expectation.* arXiv :1309.6392, 2014

[32] I. J. Goodfellow, J. Shlens, C. Szegedy. *Explaining and Harnessing Adversial Examples.* arXiv :1412.6572, 2015

[33] B. Goodman, F. Flaxman *European Union regulations on algorithmic decision-making and a "right to explanation"* arXiv :1606.08813, 2016

[34] B. M. Greenwell. *A Simple and Effective Model-Based Variable Importance Measure.* arXiv :1805.04755, 2018

[35] R. Guidotti *A Survey Of Methods For Explaining Black Box Models* arXiv :1802.01933, 2018

[36] R. Henckaerts, M.P. Côté, K. Antonio, R. Verbelen *Boosting insights in insurance tariff plans with tree-based machine learning* arXiv :1904.10890, 2019

[37] G. Hooker *Discovering Additive Structure in Black Box Functions* Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004

[38] V. Kašćelan, L. Kašćelan, M. N. Burić *A nonparametric data mining approach for risk prediction in car insurance : a case study from the Montenegrin market* DOI :10.1080/1331677X.2016.1175729, 2016

[39] B. Kim. *Examples are not Enough, Learn to Criticize ! Criticism for Interpretability.* Advances in Neural Information Processing Systems, 2016

[40] D. P. Kingma, J. L. Ba. *ADAM : A Method For Stochastic Optimization.* arXiv :1412.6980, 2017

[41] P. W. Koh. *Understanding Black-box Predictions via Influence Functions.* arXiv :1703.04730, 2017

[42] M. Kuhn, K. Johnson. *Applied Predictive Modeling* Springer, 2013

[43] F. Lange. *Exploration de la valeur de Shapley et des indices d'interaction pour les jeux définis sur des ensembles ordonnés.* 2008

[44] T. Laugel. *Inverse Classification for Comparison-based Interpretability in Machine Learning.* arXiv :1712.08443, 2017

[45] T. Laugel, X. Renard. *Defining Locality for Surrogates in Post-hoc Interpretablity.* arXiv :1806.07498, 2018

[46] E. Le Goff. *« L'actuaire est un data scientist »* L'Argus de l'assurance, 2018

[47] Z. C. Lipton *The Mythos of Model Interpretability* arXiv.org :1606.03490, 2017

[48] S. M Lundberg, S. Lee. *A Unified Approach to Interpreting Model Predictions.* arXiv :1705.07874, 2017

[49] S. M. Lundberg, G. G. Erion, S. Lee. *Consistent Individualized Feature Attribution for Tree Ensembles.* arXiv :1802.03888, 2019

[50] T. Miller. *Explanation in artificial intelligence : Insights from the social sciences.* arXiv Preprint arXiv :1706.07269, 2017

[51] C. Molnar, *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable.* `https://christophm.github.io/interpretable-ml-book/`, 2019

[52] W. J Murdoch, C. Singh *Interpretable Machine Learning : Definitions, Methods, and Applications* arXiv :1901.04592, 2019

[53] A. Nagpal. *L1 and L2 Regularization Methods* Towards Data Science, 2017

[54] P. Ottou. *Méthodes d'apprentissage automatique appliquées au provisionnement ligne à ligne en assurance non-vie.* Ressources Actuarielles, 2017

[55] A. Pagliat, M. Phélippé-Guinvarc'H. *Tarification des risques en assurance non-vie, une approche par modèle d'apprentissage statistique.* Bulletin français d'actuariat, 2011

[56] K. Pearson. *On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling* Philosophical Magazine 1900

[57] G. Perrin. *Les assurtech, parties pour perdurer dans le paysage de l'assurance.* L'Argus de l'assurance, 2018

[58] F. Planchet. *Quelques réflexions sur la segmentation en assurance.* Conférences des 30 ans de l'EURIA, 2019

[59] N. Puri *MAGIX : Model Agnostic Globally Interpretable Explanations* arXiv :1706.07160, 2018

[60] S. Rane. *The balance : Accuracy vs. Interpretability* Towards Data Science, 2018

[61] M. T. Ribeiro, S. Singh, C. Guestrin. *"Why Should I Trust You ?" - Explaining the Predictions of Any Classifier.* arXiv :1602.04938, 2016

[62] M. T. Ribeiro. *Anchors - High-Precision Model-Agnostic Explanations.* AAAI Conference on Artificial Intelligence, 2018

[63] F. de Ryckel *Machine Learning with R*, GitHub 2019

[64] C. A. Scholbeck, C. Molnar, C. Heumann, B. Bischl, G. Casalicchio *Sampling, Intervention, Prediction, Aggregation : A Generalized Framework for Model Agnostic Interpretations* arXiv :1904.03959, 2019

[65] K. Simonyan. *Deep Inside Convolutional Network - Visualising Image Classification Models and Saliency Maps.* arXiv :1312.6034, 2014

[66] E. Strumbelj, I. Kononenko *A General Method for Visualizing and Explaining Black-Box Regression Models* In : Int. Conf. on Adaptive and Natural Computing Algorithms, 2011

[67] C. Szegedy. *Intriguing Properties of Neural Networks.* arXiv :1312.6199, 2014

[68] H. F. Tan, K. Song, M. Udell. *Why should you trust my interpretation ? Understanding uncertainty in LIME predictions.* arXiv :1904.12991, 2019

[69] H. F. Tan, K. Song, M. Udell. *Safe ML : Surrogate Assisted Feature Extraction For Model Learning.* arXiv :1902.11035, 2019

[70] V. Vapnik. *The nature of statistical learning theory.* Springer, 2000.

[71] F. Vermet. *Cours d'Apprentissage Statistique : une approche connexionniste (EURIA).*

[72] F. Villeroy de Galhau. *Rapport annuel de l'ACPR* . ACPR - Banque de France, 2017

[73] S. Wachter. *Counterfactual Explanations Without Opening the Black Box : Automated Decisions and the GDPR.* arXiv :1711.00399, 2017