NEXIALOG CONSULTING

NEXIASEARCH

Benchmark sur les algorithmes de ML supervisé interprétables

Ernesto LOPEZ FUNE
Valentin MESSINA
Amande EDO

THINK SMART ACT DIFFERENT

TABLE DES MATIÈRES

INTRODUCTION

Le domaine de l'intelligence artificielle (IA) est souvent perçu comme complexe et opaque par le grand public, principalement en raison de la sophistication des algorithmes sous-jacents, rendant leur interprétation difficile pour un public non spécialisé. Bien que ces algorithmes, alimentés par des bases de données de haute qualité, produisent des modèles performants, la compréhension des mécanismes décisionnels demeure souvent insuffisante pour convaincre les décideurs [1], qui recherchent des modèles plus transparents pour soutenir leurs décisions.

L'amélioration de la précision des prédictions conduit souvent à l'élaboration d'algorithmes de plus en plus complexes, utilisant des méthodes d'apprentissage avancées et des architectures profondes. Ces algorithmes capturent des motifs subtils dans des ensembles de données volumineux et variés. Cependant, cette complexité accrue se fait au détriment de l'interprétabilité, transformant les modèles en "boîtes noires" difficiles à comprendre, même pour les experts en IA.

Cette opacité pose des défis importants dans des domaines où la transparence et l'équité sont cruciales, comme la finance, la médecine et la justice [2, 3]. Il est donc essentiel de développer des modèles interprétables pour garantir leur acceptabilité et leur applicabilité dans ces contextes sensibles et régulés [4, 5]. Un modèle intrinsèquement interprétable permet de suivre le cheminement des données de l'entrée vers la sortie, rendant le processus décisionnel transparent et comparable à un processus décisionnel quotidien.

Dans cette étude, nous comparerons des algorithmes de Machine Learning (ML) supervisé allant des plus simples aux modérément complexes (sans aborder les détails de l'apprentissage profond), en expliquant les mécanismes décisionnels sous-jacents. Nous commencerons par les algorithmes paramétriques, dont les paramètres peuvent souvent s'interpréter comme des scores d'importance de variables. Ensuite, nous expliquerons les algorithmes de ML non-paramétriques et leur interprétation.

I. Les algorithmes paramétriques de ML supervisé interprétables

Les algorithmes de ML supervisé qui simplifient la fonction de décision à une forme connue sont appelés algorithmes paramétriques. Ils utilisent un ensemble fixe de *paramètres*, quel que soit le nombre d'exemples d'entraînement, ce qui simplifie l'apprentissage mais limite la complexité des modèles. Ces algorithmes suivent deux étapes : choisir une forme pour la fonction de décision, souvent linéaire ou polynomiale, puis apprendre les paramètres à partir des données d'entraînement. Leur simplicité les rend faciles à comprendre et à interpréter, et ils nécessitent peu de données d'entraînement. Cependant, leur capacité à s'adapter à des données complexes est restreinte. Leur interprétabilité est un atout majeur, car les paramètres peuvent être vus comme les poids de chaque variable dans le processus de décision. Nous présenterons ensuite les trois algorithmes paramétriques les plus connus pour leurs applications pratiques.

1.1 Régression Linéaire

L'algorithme de régression linéaire (RL) est un algorithme paramétrique qui fait des prédictions en ajustant un nombre fini de paramètres à partir des données d'entraînement. Il suppose que la variable cible continue est une combinaison linéaire des variables prédictives, formulée par l'équation : $y = \beta_0 + X\beta + \varepsilon$, où y est la variable cible, β_0 est le biais, X les variables prédictives, β les coefficients de la régression, et ε l'erreur résiduelle.

Il arrive souvent que certaines variables prédictives aient des relations spécifiques, comme une loi de puissance $y \sim X^{\alpha}$, avec la variable cible. Les modèles de régression linéaire généralisée (GLM) ou les modèles additifs généralisés (GAM) [6] permettent de prendre en compte ces relations en utilisant des fonctions de lien

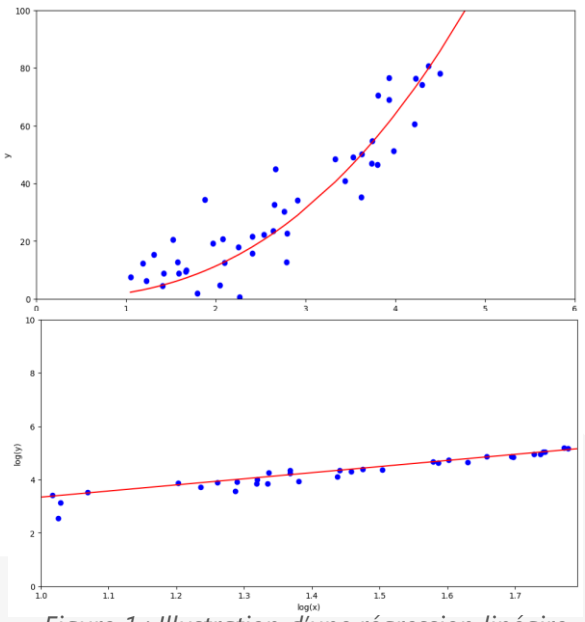

Figure 1 : Illustration d'une régression linéaire après transformation de la variable cible.

adaptées, offrant ainsi une modélisation plus flexible et précise des données. Par exemple, comme illustré dans la *Figure 1*, le passage aux logarithmes peut linéariser une relation du type $y \sim X^{\alpha}$.

Pour revenir au cas de la RL, son objectif est de trouver les valeurs optimales des paramètres β_0 et β en minimisant la norme L_2 du vecteur ε sous contraintes, avec la fonction:

$$L(\beta) = L_2[\varepsilon] + \lambda\left((1-\alpha)L_1[\beta] + \alpha\, L_2[\beta]\right),$$

où $L_2[z] = \sqrt{\sum_{i=1}^{n} z_i^2}$ et $L_1[z] = \sum_{i=1}^{n} |z_i|$ sont les normes vectorielles.

Les paramètres λ et α n'appartiennent pas à l'algorithme lui-même, mais servent à restreindre les solutions obtenues à une plage d'intérêt afin de réduire les erreurs et le surajustement. L'ajustement de λ et α avec les données est également connu sous le nom de pénalisation et régularisation, respectivement. On distingue trois types de régularisation : Least Absolute Shrinkage and Selection Operator ou LASSO ($\alpha = 0$), Ridge ($\alpha = 1$), et Elastic-Net ($0 < \alpha < 1$) qui combine les deux précédents.

Avantages : La RL est très transparente, et les signes des paramètres β indiquent les relations entre les variables prédictives et la variable cible : positif pour une relation directe, négatif pour une relation inverse. L'ampleur des paramètres représente l'impact du prédicteur sur la cible. Un autre avantage de la RL est sa facilité de mise en œuvre.

Désavantages : Le principal inconvénient de la RL est la multi-colinéarité des variables prédictives. Lorsqu'il y a des variables fortement corrélées, il est crucial de déterminer lesquelles inclure ou écarter avant de modéliser, en réalisant des tests d'hypothèse pour identifier celles ayant les associations les plus fortes avec la variable cible. Un autre inconvénient majeur, notamment pour l'estimation du paramètre β_0, est la présence de valeurs aberrantes qui peuvent affecter cette estimation.

1.2 Régression Logistique

La régression logistique (RG) est un algorithme paramétrique qui utilise une combinaison linéaire des variables prédictives pour estimer la probabilité qu'une variable cible binaire appartienne à l'une des deux classes (positive ou négative). Cette combinaison linéaire est transformée en une probabilité grâce à la fonction logistique, qui comprime la sortie en une valeur entre 0 (classe positive) et 1 (classe négative). Cette probabilité permet ensuite d'effectuer une classification binaire en utilisant un seuil, souvent fixé à 0.5, pour décider de la classe assignée.

Mathématiquement, cela s'exprime par la fonction logistique :

$$p(y) = \frac{1}{1 + e^{-y}}, y = \beta_0 + X\beta,$$

avec β_0, X et β comme dans la RL, mais contrairement à cela, les paramètres β_0 et β sont déterminés en maximisant la fonction :

$$L = \prod_{\{k:y_k=1\}} p_k \prod_{\{k:y_k=0\}} (1 - p_k),$$

représentant la probabilité d'appartenance de chaque instance de données à la classe correspondante. Cet algorithme peut aussi être régularisé en utilisant les méthodes LASSO, Ridge et Elastic-Net pour pénaliser les paramètres β.

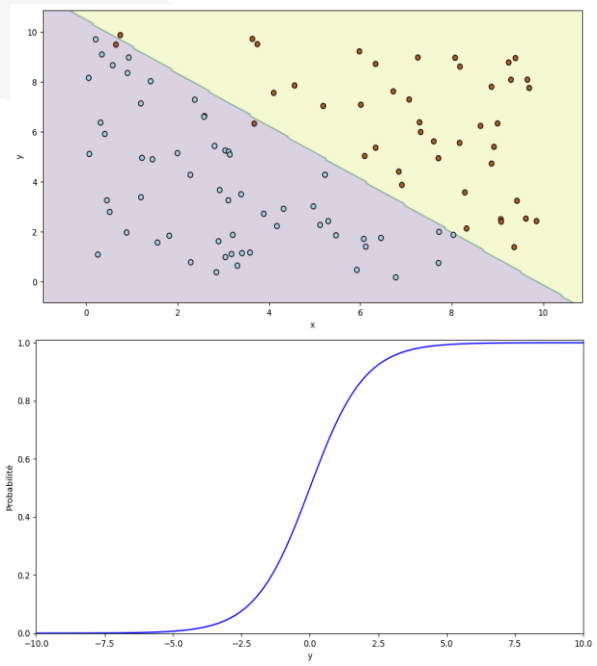

Figure 2 : Illustration de la régression logistique : scindage des données en deux classes (haut) et fonction logistique (bas).

Dans le panneau supérieur de la *Figure 2*, le processus de séparation des deux classes à l'aide d'une frontière de décision linéaire est illustré, tandis que dans le panneau inférieur, la conversion de chaque point en probabilité est représentée.

Avantages : La régression logistique (RG) est facile à interpréter, car un paramètre β positif ou négatif indique qu'une augmentation de la variable prédictive associée augmente la probabilité d'appartenance à la classe correspondante. Une valeur absolue plus élevée d'un paramètre β reflète un effet plus fort du prédicteur sur cette probabilité. Elle est facile à mettre en œuvre, s'entraîne efficacement, et peut être étendue aux GLM et aux GAM.

Elle permet de classifier rapidement les observations inconnues et est précise pour des ensembles de données simples et linéairement séparables. De plus, elle est moins sujette au surajustement, bien que des techniques de régularisation puissent être nécessaires pour les ensembles de données de grande dimension.

Désavantages : La RG présente plusieurs limitations importantes. Elle ne doit pas être utilisée lorsque le nombre d'observations est inférieur au nombre de classes, car cela peut entraîner un surajustement. Elle repose sur l'hypothèse de linéarité entre la variable cible et les variables prédictives, ce qui limite sa capacité à résoudre des problèmes non linéaires, qui sont courants dans le monde réel. Elle nécessite également une faible multi-colinéarité entre les variables indépendantes et ne peut prédire que des fonctions discrètes.

1.3 Perceptron Simple

Le perceptron simple (PS), proposé initialement comme algorithme de classification binaire supervisé [7], est une composante fondamentale de l'IA moderne [8]. Les blocs interconnectés de perceptrons simples forment des architectures ou réseaux neuronaux complexes, utilisés dans des domaines tels que la reconnaissance de formes, de sons et la vision par ordinateur [9].

Le PS se distingue par sa simplicité et son efficacité. Il modélise un neurone formel avec une règle d'apprentissage pour séparer les données linéairement séparables de manière optimale. En tant qu'algorithme paramétrique, il prend en entrée un vecteur $X = [X_1, X_2, \dots, X_n]$ et produit une sortie binaire $y = \theta(\beta_0 + \sum_{i=1}^{n} \beta_i X_i)$, comme schématisé dans la Figure 3. La fonction d'activation θ est souvent la fonction de Heaviside

$$\theta(x) = \begin{cases} 0, si\ x < 0, \\ \frac{1}{2}, si\ x = 0, \\ 1, si\ x > 0, \end{cases}$$

ou une fonction de la famille sigmoïde, à laquelle appartient la fonction logistique présentée dans la sous-section précédente.

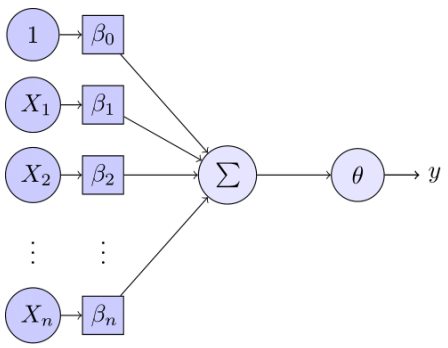

Figure 3 : Schéma de fonctionnement du PS.

L'apprentissage dans un PS consiste à optimiser les poids β_i et le biais β_0 afin de minimiser l'erreur entre les prédictions et la variable cible, avec la possibilité d'appliquer une régularisation.

Un PS avec une fonction d'activation logistique et la RG deviennent très similaires lorsqu'ils utilisent la même fonction de coût et méthode d'ajustement des poids. Leur principale différence réside dans leur contexte d'utilisation et leur historique, bien qu'ils soient presque interchangeables pour des tâches de classification binaire dans la pratique moderne.

Le PS peut également être utilisé pour la régression lorsque des fonctions d'activation comme la Rectified Linear Unit (ReLU) $ReLU(x) = \max(0, x)$ sont employées. L'ajustement des poids dans une RL est direct et continu, tandis que dans un PS avec ReLU, il dépend de l'activation des neurones, introduisant des non-linéarités et des sparsités dans l'apprentissage.

Avantages : Le PS offre une grande transparence grâce à sa structure linéaire, permettant de voir clairement l'influence de chaque entrée sur la sortie via les poids β_i. Facile à implémenter et à comprendre, il est rapide et efficace pour les tâches de classification binaire et de régression. De plus, il nécessite moins de ressources computationnelles que les algorithmes plus complexes, ce qui le rend adapté aux applications à faible puissance de calcul.

Désavantages : Le PS est limité aux problèmes linéairement séparables et présente les mêmes inconvénients que la RL et la RG.

II. Les algorithmes non-paramétriques de ML supervisé interprétables

Les algorithmes de ML supervisé non-paramétriques n'ont pas de fonction de décision prédéfinie. Ils s'adaptent aux ensembles d'entraînement en ajustant des *hyperparamètres* via des méthodes comme la recherche sur grille. Ces algorithmes, utiles avec de grands volumes de données, apprennent des formes fonctionnelles variées à partir des observations. Bien qu'ils soient flexibles et puissants, ils nécessitent beaucoup d'exemples, sont plus lents à entraîner et présentent un risque accru de surapprentissage. Contrairement aux algorithmes paramétriques, le poids de chaque variable sur la décision finale doit être déterminé par des méthodes comme la Permutation Feature Importance (PFI) [10], LIME [11], et les valeurs de Shapley [12], qui seront discutées séparément dans une autre note benchmark [5]. De plus, ils reflètent la prise de décision humaine en utilisant des processus adaptatifs basés sur des exemples similaires.

Cette section présente six algorithmes non-paramétriques de ML supervisé couramment utilisés.

2.1 Support Vector Machines et Regressors (SVM et SVR)

Les Support Vector Machines (Machines à Vecteurs de Support) ou SVM [13] sont des algorithmes puissants conçus pour la classification supervisée et aussi pour les tâches de détection de valeurs aberrantes, trouvant un hyperplan optimal dans un

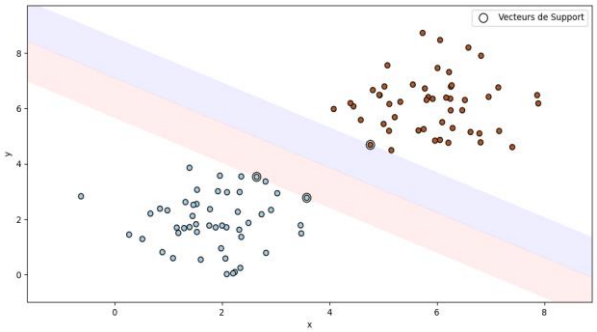

Figure 4 : Schéma de fonctionnement du SVM.

espace de dimension supérieure pour séparer les classes des données d'entraînement, comme illustré sur la *Figure 4*.

Les SVM peuvent être comparées à un recruteur expérimenté différenciant les candidats qualifiés des non qualifiés. Le recruteur évalue les caractéristiques essentielles, telles que l'expérience et les compétences, pour tracer une frontière de décision claire. De même, ils utilisent des informations pertinentes pour définir une séparation optimale entre différentes classes de candidats, permettant ainsi de prendre des décisions éclairées et précises pour garantir le meilleur choix possible.

L'utilisation des SVM reflète le processus de prise de décision humaine, où les informations pertinentes sont utilisées pour distinguer efficacement les différentes catégories d'individus. Cette approche reflète le fonctionnement de l'algorithme SVM, qui identifie la meilleure séparation possible entre différentes classes de données pour optimiser la précision des décisions.

La découverte de comportements non linéaires dans les données est courante, rendant les frontières de décision linéaires inadaptées.

Pour mieux traiter ces données, on utilise des techniques comme l'astuce du noyau [14] dans les SVM. Cette méthode consiste à mapper les données dans un espace de grande dimension où la frontière de décision devient linéaire, permettant l'application d'un SVM simple. Par exemple, des points sur des circonférences de rayons différents ne sont pas séparables linéairement. En ajoutant une coordonnée représentant le rayon, les points deviennent séparables dans un nouvel espace dimensionnel, comme illustré sur la *Figure 5*. Ainsi, les SVM peuvent gérer des problèmes de classification linéaires et non linéaires.

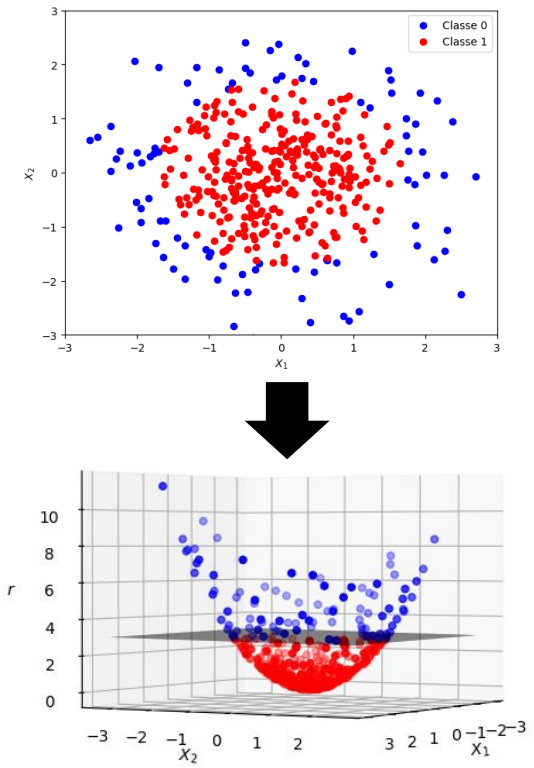

Figure 5 : Visualisation de l'astuce du noyau pour les SVM.

La version de SVM utilisée pour la régression est appelée Support Vector Regression (SVR). Les SVR cherchent à approximer la relation entre les variables prédictives et la cible en minimisant les écarts au-delà d'une marge de tolérance ϵ. En utilisant une fonction de perte insensible à ϵ, le SVR pénalise uniquement les erreurs dépassant cette marge. Comme les SVM, les SVR peuvent aussi utiliser l'astuce du noyau pour traiter les frontières de décision non linéaires.

Avantages : Les SVM/SVR peuvent être comparés à un processus de décision humain malgré leur complexité algorithmique. Elles sont efficaces et robustes face aux données bruitées et aux valeurs aberrantes grâce aux marges souples. En utilisant des noyaux, elles transforment les données non linéairement séparables en données linéairement séparables dans des dimensions supérieures, avec une interprétation géométrique simple. Enfin, les SVM/SVR ont une bonne capacité de généralisation et fonctionnent bien avec des petits et grands jeux de données.

Désavantages : Le principal désavantage des SVM/SVR est leur complexité computationnelle, surtout avec des jeux de données volumineux, rendant gourmand en ressources l'entraînement des modèles. De plus, le choix du noyau et des hyperparamètres peut être délicat et nécessite souvent une validation croisée approfondie pour des résultats optimaux.
Le poids de chaque variable dans la décision finale peut être difficile à interpréter en raison des transformations qui doivent être effectuées pour séparer efficacement les données, nécessitant le recours aux méthodes mentionnées au début de la section.

2.2 *k* plus proches voisins (*kNN*)

k plus proches voisins (***kNN***) est un algorithme non-paramétrique de ML supervisé utilisé pour les tâches de classification et de régression [15]. Cet algorithme se base sur le vote majoritaire des voisins les plus proches, similaire à la manière dont les gens prennent des décisions en fonction des recommandations de leurs pairs. Par exemple, on peut choisir un restaurant basé sur les avis des ***k*** amis ayant des goûts similaires. Pour éviter les égalités lors du vote majoritaire, le nombre d'amis ***k*** doit être impair.

Cet algorithme prédit la classe d'une observation donnée en fonction de la distance aux ***k*** voisins les plus proches. Si la majorité des ***k*** voisins appartiennent à une classe, l'observation est probablement de cette classe.

La *Figure 6* illustre cet algorithme pour ***k***=3 et deux classes (rouge et bleu) : où on cherche à classer l'observation représentée par la croix noire.

Comme la majorité de ses 3 plus proches voisins est rouge (ici ses 3 plus proches voisins sont rouges), alors on prédit une classe rouge pour cette observation.

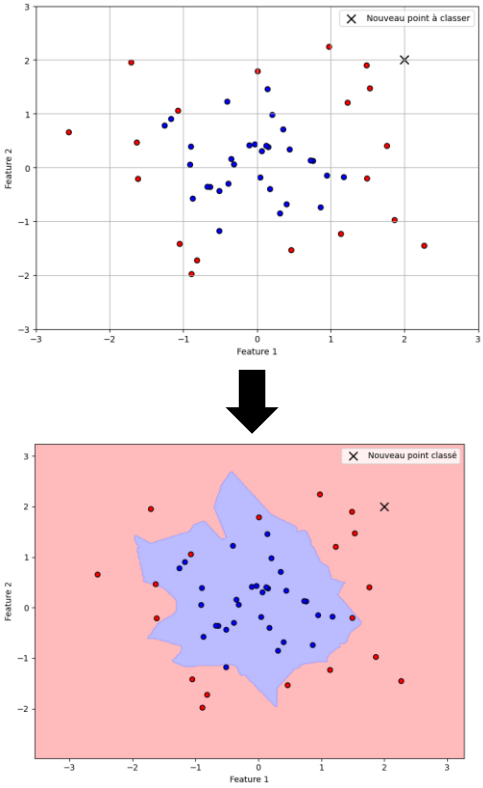

*Figure 6 : Visualisation de **kNN** pour **k** = 3.*

Le degré de proximité est généralement déterminé par une fonction de distance, souvent la distance euclidienne, mais d'autres fonctions de distances peuvent également être utilisées, comme les distances générées par les *p-normes* [16].

Dans le cas de régressions, une fois que les ***k*** plus proches voisins sont identifiés, l'algorithme agrège leurs valeurs cibles pour faire une prédiction pour la nouvelle observation donnée.

L'approche la plus courante consiste à prendre la moyenne (ou parfois la médiane) des valeurs cibles des ***k*** voisins. Enfin, la valeur agrégée est attribuée comme la valeur prédite.

Avantages : ***kNN*** est un modèle transparent, basé uniquement sur les données d'entrée et la fonction de distance utilisée, avec une interprétation géométrique claire. Il ne fait aucune hypothèse sur la distribution des données, ce qui le rend adaptable à divers types de problèmes. Il est efficace pour les petits ensembles de données et peut gérer des classes complexes en capturant des frontières de décision non linéaires.

Désavantages : ***kNN*** est coûteux en calcul, surtout pour les grands jeux de données, car il nécessite de calculer les distances entre les données d'entraînement et chaque nouvelle observation. Sa performance dépend du choix de ***k*** et de la méthode de mesure de distance, nécessitant souvent une optimisation et une validation croisée. Il est sensible aux échelles des variables prédictives, qui doivent être normalisées pour éviter des dominances. ***kNN*** peut être affecté par le bruit dans les données, introduisant des erreurs si les voisins proches sont mal étiquetés. Enfin, comme tout algorithme non-paramétrique, le poids de chaque variable sur la décision finale est difficile à déterminer sans méthodes spécifiques.

2.3 Arbres de décisions

Les arbres de décision sont des éléments fondamentaux du ML supervisé, reconnus pour leur interprétabilité et leur processus décisionnel intuitif [17]. Ils s'appliquent aussi bien en classification qu'en régression et offrent une grande transparence.

Fonctionnant comme un dialogue structuré, chaque nœud représente une question posée aux variables prédictives concernant la variable cible. Imaginons un groupe de spécialistes avec différentes expertises (les variables prédictives) et un problème complexe à résoudre (la variable cible). On commence par poser une question au spécialiste le plus expérimenté, représenté par la racine de l'arbre. En fonction de sa réponse, on interroge d'autres spécialistes, formant les nœuds internes. Chaque réponse guide vers des questions plus spécifiques, formant les branches de l'arbre. Finalement, après avoir posé suffisamment de questions précises, on arrive aux réponses finales, représentées par les feuilles de l'arbre.

Ce processus itératif sélectionne stratégiquement les variables les plus informatives, facilitant ainsi la compréhension du modèle, sa visualisation et la capture des relations complexes entre les variables prédictives et la variable cible.

On trouvera sur la *Figure 7* un exemple d'arbre de décisions qui permet de visualiser son fonctionnement, en observant comment il grandit hiérarchiquement à partir de la variable avec la plus grande information, exprimée quantitativement à partir de

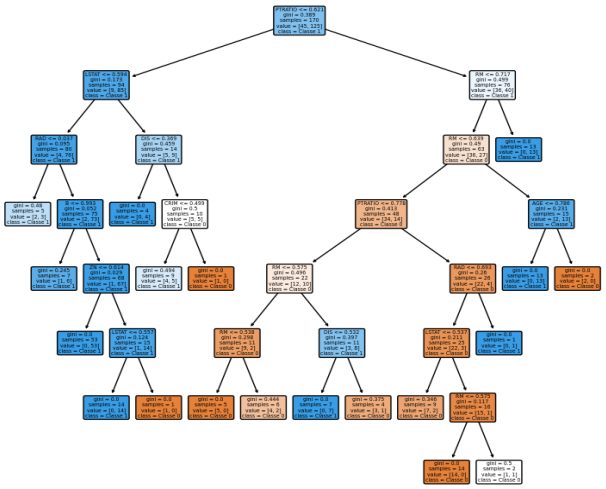

Figure 7 : Exemple d'arbre de décision.

l'indice de Gini ou de l'entropie de cette variable.

Avantages : Les arbres de décision sont faciles à interpréter et à visualiser, permettant une compréhension intuitive des décisions sans expertise en ML. Ils gèrent des données catégorielles et numériques et capturent les interactions non linéaires, les rendant adaptés à des problèmes complexes. De plus, dans le processus de formation basé sur les hiérarchies des variables les plus informatives, le poids de chacune d'elles est interprétable.

Désavantages : Le surajustement et le biais peuvent arriver, surtout s'ils sont trop profonds, réduisant leur performance sur de nouvelles données. Ils sont sensibles aux variations dans les données, et de petites modifications peuvent changer significativement leur structure.

2.4 Les méta-algorithmes

Les méta-algorithmes exploitent la sagesse collective pour construire des algorithmes plus complexes mais plus performants à partir d'algorithmes de base, également appelés *apprenants faibles*. Dans cette sous-section, nous montrerons les trois types de méta-algorithmes les plus utilisés en pratique : le bagging, le boosting et le stacking.

2.4.1 Méta-algorithmes de bagging

Les méta-algorithmes de bagging [18] ou d'agrégation de renforcement sont conçus pour améliorer la stabilité et la précision des résultats en classification et en régression, réduisant ainsi la variance et aidant à éviter le surajustement.

Pour mieux comprendre le concept du bagging, imaginons que l'on souhaite prédire la survenue d'un événement spécifique en consultant un panel de spécialistes dans des domaines connexes, comme dans l'exemple des arbres de décision. Ce panel est divisé en plusieurs groupes des spécialistes, et des questions similaires sont posées aléatoirement à chaque groupe, avec possibilité de répétition des questions. Les réponses indépendantes de chaque groupe sont ensuite combinées pour former une décision finale. Cette analogie illustre parfaitement le fonctionnement du bagging : plusieurs sous-échantillons de données sont créés par échantillonnage

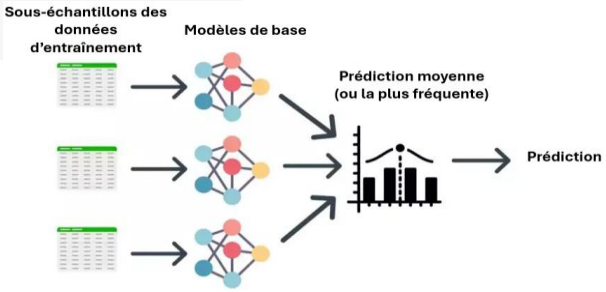

Figure 8 : Schéma de fonctionnement du bagging.

avec remplacement, des modèles de base sont entraînés sur chaque sous-ensemble, puis la prédiction finale s'obtient en agrégeant toutes les prédictions générées par les modèles de base entraînés, comme schématisé sur la *Figure 8*.

Bien que cette technique soit généralement appliquée aux arbres de décision, elle peut être utilisée avec n'importe quel type d'algorithme de ML supervisé. L'algorithme le plus connu qui exploite cette méthodologie d'apprentissage par des arbres de décision est le Random Forest [19].

Avantages : Le bagging crée un modèle plus robuste et efficace, moins sensible aux variations des données d'entraînement et moins sujet aux surajustements que les arbres de décision individuels. Le sous-échantillonnage aléatoire pour entraîner chaque modèle de base et l'agrégation finale des prédictions réduisent considérablement la variance.

Désavantages : Le bagging peut être coûteux en termes de calcul car il nécessite l'entraînement de plusieurs modèles sur différents échantillons. Bien que ce processus réduise la variance, il peut introduire un biais dans les prédictions, surtout si les données d'entraînement sont biaisées. De plus, étant un méta-algorithme combinant plusieurs modèles, l'interprétation du poids des variables dans le processus de décision n'est pas évidente et nécessite des méthodes spécifiques, comme celles mentionnées au début de la section.

2.4.2 Méta-algorithmes de boosting

Le boosting [20], similaire au bagging, est une technique d'ensemble puissante utilisée en apprentissage supervisé pour améliorer la performance des apprenants faibles. Son objectif est de combiner plusieurs modèles pour créer un modèle plus précis et moins biaisé que ceux produits par le bagging.

Cette méthode peut être comparée à une consultation séquentielle d'un panel de spécialistes, divisé en plusieurs groupes, pour comprendre un sujet complexe. Un premier groupe fournit sa meilleure explication, bien qu'imparfaite, puis chaque groupe suivant corrige les lacunes des précédents, affinant progressivement la compréhension du sujet, comme illustré à la *Figure 9*.

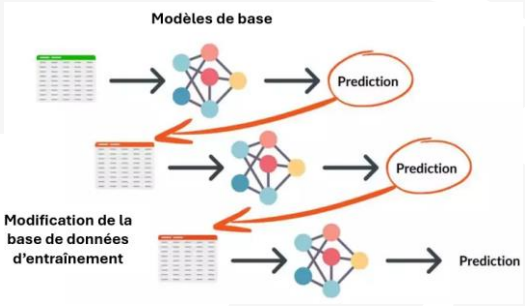

Figure 9 : Schéma de fonctionnement du boosting.

Au lieu de créer des sous-échantillons des questions, le boosting ajuste les poids des observations dans l'ensemble de données. Les observations mal prédites par les groupes précédents reçoivent un poids plus élevé, de sorte que les modèles ultérieurs se concentrent davantage sur ces observations difficiles.

Le boosting s'applique aussi bien aux tâches de classification qu'à celles de régression, tout comme le bagging. Les implémentations les plus connues des méta-algorithmes de boosting sont AdaBoost [21], XGBoost [22] et CatBoost [23], qui utilisent des arbres de décision comme modèles faibles.

AdaBoost améliore la précision des modèles faibles en ajustant les poids des instances mal classées, créant ainsi un classificateur robuste. XGBoost optimise le boosting de gradient avec des techniques de régularisation, le traitement parallèle et une gestion efficace des arbres, offrant des performances accrues. Il est reconnu pour sa scalabilité et sa capacité à traiter de grands ensembles de données à haute dimensionnalité, en faisant un choix privilégié dans les compétitions de ML. CatBoost gère les variables catégorielles sans prétraitement intensif, utilise un boosting ordonné et des méthodes pour réduire le surapprentissage, offrant d'excellentes performances sur des ensembles de données avec de nombreuses variables catégorielles.

Chacun de ces algorithmes exploite le boosting de manière unique pour améliorer la précision et l'efficacité des prédictions, ce qui en fait des outils précieux pour les data scientists.

Avantages : Le boosting améliore la performance en réduisant le biais et la variance, en se concentrant sur les erreurs des modèles précédents. Sa construction séquentielle permet à chaque nouveau modèle de corriger les erreurs des précédents, ce qui conduit à des prédictions très précises et permet de traiter efficacement les données déséquilibrées.

Désavantages : Le boosting peut être sujet au surajustement, surtout si le nombre d'apprenants faibles est trop élevé. Il est également sensible aux données bruitées et aux valeurs aberrantes, car il essaie de corriger les erreurs à chaque étape. Comme le bagging, le boosting est computationnellement intensif et nécessite un temps de calcul plus long. De plus, les poids finaux des variables dans le processus de prise de décision ne sont pas directement interprétables et nécessitent des méthodes spécifiques pour leur analyse.

2.4.3 Méta-algorithmes de stacking

Les méta-algorithmes de stacking suivent le même principe que ceux de bagging et de boosting, en créant une famille de d'apprenants faibles pour obtenir un algorithme plus robuste et aux performances améliorées [24].

Il fonctionne comme une assemblée nationale composée de plusieurs partis (apprenants faibles), chacun ayant ses propres députés. Le président de l'assemblée (métamodèle) propose un projet de loi (variable cible) et sollicite l'avis de chaque parti. Chaque parti, avec ses priorités spécifiques, évalue différents

aspects du projet et rend son verdict. Le président compile ensuite ces verdicts pour prononcer la décision finale avec son propre critère.

Le stacking se distingue principalement par son approche de l'entraînement et de la combinaison des apprenants faibles, sans recourir au sous-échantillonnage comme le bagging. En effet, les apprenants faibles sont entraînés sur l'ensemble de données complet ou via validation croisée. Leurs prédictions sont ensuite utilisées pour entraîner un métamodèle final, comme illustré à la *Figure 10*. Ce métamodèle combine les forces de chaque modèle individuel, offrant ainsi une prédiction plus précise et améliorant la performance globale.

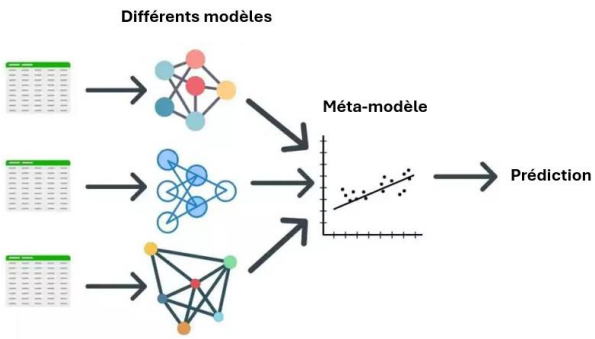

Figure 10 : Schéma de fonctionnement du stacking.

Le stacking est une approche flexible, applicable à diverses situations. Le méta-algorithme de stacking le plus connu est l'ExtraTrees [25], qui utilise des arbres de décision comme des apprenants faibles, puis combine les prédictions de chaque arbre dans un méta-arbre final. Il existe aussi une version pour la régression.

Avantages : Le stacking améliore les performances des apprenants faibles en combinant leurs prédictions, ce qui conduit à des résultats plus précis et robustes. En réduisant à la fois la variance et le biais, il exploite les points forts de différents algorithmes d'apprentissage supervisé, offrant ainsi une plus grande flexibilité. En utilisant des modèles hétérogènes, le stacking capture mieux les divers aspects des données, le rendant particulièrement efficace pour les problèmes complexes. De plus, cette méthode est moins susceptible de surajuster les données d'entraînement grâce à la combinaison judicieuse des prédictions des différents modèles.

Désavantages : Cependant, le stacking présente une complexité accrue par rapport à des méthodes d'ensemble plus simples comme le bagging ou le boosting. La gestion et la combinaison de plusieurs modèles nécessitent une expertise technique avancée et un temps de calcul plus long, surtout pour des ensembles de données volumineux. Il y a aussi un risque de surajustement si le métamodèle est trop complexe, et la validation croisée est nécessaire pour éviter les fuites de données, ajoutant une couche de complexité supplémentaire. En outre, les modèles empilés sont souvent plus difficiles à interpréter, rendant la compréhension des mécanismes internes et des décisions prises par le modèle final moins transparente. Comme dans le bagging et le boosting, les poids des variables ne sont pas directement interprétables.

CONCLUSION

La transparence et l'interprétabilité des algorithmes de ML supervisé sont cruciales pour leur acceptation dans des domaines sensibles comme la finance, la médecine, et la justice. Ce benchmark a comparé divers algorithmes paramétriques et non-paramétriques, en soulignant les mécanismes décisionnels de chacun, leurs avantages et leurs désavantages.

Les algorithmes paramétriques, tels que la RL, la RG et le PS, offrent une bonne interprétabilité grâce à leurs structures simples et aux significations claires des paramètres. Cependant, leur capacité à gérer des relations complexes et non linéaires est limitée. Les algorithmes non-paramétriques, comme les SVM, ***kNN*** et les arbres de décision, sont plus flexibles et puissants, mais leur interprétabilité dépend souvent de méthodes additionnelles comme les PFI ou les valeurs de Shapley. Enfin, les méta-algorithmes comme le bagging, le boosting et le stacking combinent plusieurs modèles pour améliorer la performance et la robustesse, au prix d'une complexité accrue et d'une interprétation plus difficile des poids des variables.

En conclusion, le choix de l'algorithme dépend du compromis entre la complexité du modèle et la nécessité d'interprétabilité, avec une préférence pour les modèles plus transparents dans des contextes où la compréhension et la justification des décisions sont essentielles.

RÉFÉRENCES

1. M. Il Idrissi, N. Bousquet, F. Gamboa, B. Iooss, J-M. Loubes. Understanding black-box models with dependent inputs through a generalization of hoeffding's decomposition, e-print: arXiv 2105.07190, 2023.
2. A. Barredo Arrieta, N. Diaz Rodriguez, J. Del Ser. Explainable artificial intelligence (XAI) : Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion,* 58 : 82-115, 2020.
3. C. Hurlin, C. Pérignon, S. Saurin. The fairness of credit scoring models, e-print: arXiv 2205.10200, 2024.
4. V. Nguyen, N. Kadio. Techniques d'interprétabilité des modèles de Machine Learning. *Production interne R&D*, Nexialog Consulting, 2022.
5. E. Lopez Fune, V. Messina, A. Edo. Corrélation et Importance des variables : guide des modèles de ML supervisé. *Production interne R&D*, Nexialog Consulting, 2024.
6. P. McCullagh, J. A. Nelder. Generalized Linear models. *Chapman and Hall/CRC*, 37, 1989.
7. F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 1958.
8. M. Minsky, S. Papert. Perceptrons: An Introduction to Computational Geometry. *MIT Press*, 1988.
9. R. Szeliski. Computer Vision: Algorithms and Applications. *Springer*, 2010.
10. C. Molnar. Interpretable Machine Learning A Guide for Making Black Box Model Explainable, 2023. https://christophm.github.io/interpretable-ml-book/
11. M.T. Ribeiro, S. Singh, C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, . e-print arXiv : 1602.04938, 2016.
12. L.S. Shapley. A Value for n-Person Games. *Contributions to the Theory of Games Princeton University Press*, 307-317, 1953.
13. A. Sucre. Support vector machines : A brief introduction. *Medium,* 2022. https://medium.com/the-beginners-guide/support-vector-machines-a-brief-introduction-784bbf97cdce
14. B. Schölkopf, A. J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. *MIT Press*, 2002.
15. X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J McLachlan, A. Ng, B. Liu, S Yu Philip, *et al.* Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1) : 1–37, 2008.
16. A. C Thompson. Minkowski Geometry. Encyclopedia of Mathematics and its Applications. *Cambridge University Press*, 1996.
17. L. Breiman, J. H Friedman, R. A Olshen, C. J Stone. Classification and Regression Trees. *Wadsworth International Group*, 1984.
18. L. Breiman. Bagging predictors. *Machine Learning*, 24(2) : 123–140, 1996.
19. L. Breiman. Random forests. *Machine Learning*, 45(1) : 5–32, 2001.

20. J. H Friedman. Greedy function approximation : a gradient boosting machine. *Annals of statistics*, 1189–1232, 2001.

21. R. E Schapire. Explaining AdaBoost. In Empirical inference, *Springer*, 37–52, 2013.

22. T. Chen, C. Guestrin. XGBoost : A scalable tree boosting system. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, 2016.

23. A. V Dorogush, V. Ershov, A. Gulin. CatBoost : gradient boosting with categorical features support, e-print: arXiv 1810.11363, 2018.

24. D. H Wolpert. Stacked generalization. *Neural Networks*, 5(2) : 241–259, 1992.

25. P. Geurts, D. Ernst, L. Wehenkel. Extremely Randomized Trees. *Machine Learning*, 36 : 3–42, 2006.

NEXIALOG CONSULTING

ACTUARIAT | **GESTION DES RISQUES** | **DATA** | **FINANCE DURABLE**

Nexialog Consulting est un cabinet de conseil spécialisé en Stratégie, Actuariat, Gestion des risques et Data qui dessert aujourd'hui les plus grands acteurs de la banque et de l'assurance. Nous aidons nos clients à améliorer de manière significative et durable leurs performances et à atteindre leurs objectifs les plus importants.

Les besoins de nos clients et les réglementations européennes et mondiales étant en perpétuelle évolution, nous recherchons continuellement de nouvelles et meilleures façons de les servir. Pour ce faire, nous recrutons nos consultants dans les meilleures écoles d'ingénieur et de commerce et nous investissons des ressources de notre entreprise chaque année dans la recherche, l'apprentissage et le renforcement des compétences.

Quel que soit le défi à relever, nous nous attachons à fournir des résultats pratiques et durables et à donner à nos clients les moyens de se développer.

CONTACTS

Retrouvez toutes nos publications sur Nexialog R&D

www.nexialog.com

ALI BEHBAHANI

Associé, Fondateur

+33 (0) 1 44 73 86 78

abehbahani@nexialog.com

ARESKI COUSIN

Directeur Scientifique

+33 (0) 7 88 03 51 87

acousin@nexialog.com

CHRISTELLE BONDOUX

Associée, Directrice Commerciale, Recrutement & Marketing

+33 (0) 1 44 73 75 67

cbondoux@nexialog.com

BAPTISTE BOBEL

Account Manager

+33 (0)6 64 59 12 48

bbobel@nexialog.com