CaMUS **2** (2012), 34–58

Les arbres de décision hybrides

Adam Salvail-Bérard

Résumé. Les bases de données, toujours grandissantes, renferment une variété d'informations qui n'attendent qu'à être extraites. Pour y parvenir, un éventail d'outils de forage de données a été inventé. Parmi ceux-ci, une méthode qui permet de créer de nouveaux algorithmes de forage est d'hybrider certains algorithmes existants. Cet article se penche sur les hybrides qui utilisent entre autre l'arbre de décision dans leur conception. L'arbre de décision est d'abord décrit en détail puis une exploration des méthodes d'hybridation est exposée. Enfin, un exemple d'arbre de décision hybride capable de prendre en charge les coûts de test et de mauvaise classification est présenté.

1. Introduction

Depuis quelques années, il est possible de constater une nette augmentation de l'information prête à la consommation et celle-ci ne cesse de croitre. Grandement aidé par les outils informatiques maintenant à notre disposition, l'entreposage de données numériques est aujourd'hui plus facile que jamais. Par conséquent, il est routinier de garder une quantité importante d'informations, même si elle ne semble pas être une source, à première vue, de connaissances autres que celles représentées par les données elles-mêmes.

Pourtant, ces énormes bases de données sont des mines d'informations d'une grande valeur pour ceux qui sont en mesure d'en extraire les secrets. La discipline qui se spécialise dans cette extraction s'appelle le forage de données (*data mining*) [1] [2].

L'une de ces techniques d'extraction, l'arbre de décision, retient l'attention tant par sa simplicité que par la facilité d'interprétation des modèles qu'il crée. Cet article, après une présentation des concepts utiles à la lecture de ce document,

J'aimerais remercier chaleureusement le CRSNG pour son appui financier et Jessica Lévesque pour ses conseils et son soutien au niveau de la rédaction.

1. Il est courant de rencontrer les termes « exploration des données », « fouille de données » ou même « extraction de connaissances ». En anglais, il est fréquent de rencontrer également « *Knowledge Discovery in Data*.

2. Le domaine du forage de données, comme bon nombre de domaines des technologies, utilise le plus souvent l'anglais pour transmettre ses informations, notamment dans le choix de nom des nouveaux concepts. C'est pourquoi ce rapport indique les termes anglophones en *italique* de la plupart des définitions. Prendre note également que certaines définitions francophones sont le fruit d'une traduction libre.

présente l'arbre de décision dans le but de l'améliorer en créant des techniques d'apprentissage hybrides. Finalement, un exemple d'algorithme hybride capable de prendre en charge les coûts de test est présenté.

2. Le forage de données

Cette section fait le point sur les connaissances requises afin de favoriser la compréhension des sections suivantes. Il est question de parcourir quelques concepts de base du forage de données et de se familiariser avec les notions et le vocabulaire propres à ce domaine.

2.1. Les méthodes d'apprentissage

Une base de données est une collection de mesures qui représentent un système. Ces mesures sont encodées sous forme de **données** qui sont la représentation de ces mesures sur un support informatique. Ces données sont entreposées dans une base de données sous forme d'**instances**[3] qui sont des vecteurs de valeurs fournies par des mesures. Chaque instance représente un élément de l'étude. Avec ces définitions, une **base de données** devient donc une collection d'instances.

Par exemple, pour une base de données contenant des informations sur un ensemble de figures géométriques, chaque figure est une instance de la base de données. De plus, chacune de ces figures (ou instances) possède des caractéristiques (hauteur, largeur...) appelées **attributs** ou **variables**. Ces valeurs sont les mesures qui représentent chaque instance. Dans un autre domaine, un foreur de données peut s'intéresser à l'influence de la météo sur l'achalandage d'un parc d'attractions : une instance pourrait être une journée d'activités et représentée par divers attributs concernant la météo, l'achalandage, le chiffre d'affaires... Ou encore, il est possible d'étudier la généalogie des vaches laitières : les vaches sont les instances et les attributs sont les informations qu'il est possible de recueillir sur chacune des vaches. Dans le domaine de la finance, il est possible de lister de nouvelles entreprises (les instances) selon plusieurs caractéristiques (les attributs) pour les classifier ou prédire leurs chances de réussite. Pour les banques, chaque client de leurs bases de données est un individu sur lequel ils possèdent une panoplie d'informations les aidant à faire des études pour, par exemple, attribuer les cotes de crédit.

Formellement, chaque instance est donc un vecteur de dimension égale au nombre d'attributs d'une base de données. Un attribut est dit **catégorique** ou **numérique**. De façon simple, si la moyenne des valeurs a un sens, il s'agit d'un attribut numérique. Sinon, l'attribut est catégorique. Par exemple, la couleur d'un objet est un attribut catégorique alors que sa hauteur en centimètres est un attribut numérique.

Les techniques de forage de données sont séparées en deux grands groupes. Le premier, **l'apprentissage non supervisé**, regroupe les procédures qui tentent de décrire l'organisation des données. Un exemple d'algorithme d'apprentissage non supervisé est **l'analyse de groupement** dont le but est de grouper les instances semblables selon leurs similarités.

3. Une instance est aussi souvent appelée un enregistrement, une observation ou un individu.

Le deuxième grand groupe de méthode d'apprentissage concerne l'**apprentissage supervisé**. Dans cette catégorie, le but est souvent de décrire l'information (graphiquement, à l'aide de règles...) en créant un **modèle** représentatif des données étudiées. Un tel modèle sert à établir des liens entre un attribut d'intérêt, la variable **dépendante**, et les autres attributs, les variables **indépendantes** [4].

Pour effectuer un apprentissage supervisé, le foreur de données dispose d'un certain nombre d'instances dont la valeur de la variable dépendante est déjà connue. C'est à partir de ces exemples que le modèle est construit et celui-ci sert ensuite à établir des prédictions. Partant du principe qu'il existe un modèle décrivant parfaitement la réalité, le but devient donc d'estimer le plus fidèlement possible ce modèle optimal.

De ces modèles découle le côté prédiction de l'apprentissage supervisé. Il fournit des méthodes pour estimer la valeur de la variable dépendante en se basant sur la valeur d'autres attributs et sur le modèle représentant les données. Ce pronostic est souvent le but ultime de l'utilisation des techniques d'apprentissage supervisé. C'est également ce qui est étudié dans cet article.

La prédiction de la variable dépendante est définie comme étant une **classification** ou une **estimation** si la nature de la variable dépendante est respectivement catégorique ou numérique. Dans le cas d'une classification, les valeurs possibles de la variable dépendante sont appelées les **classes**.

Un exemple de classification : il faut classer des figures géométriques selon leur forme (carré, rectangle, triangle...). Dans ce cas, chaque instance possède des attributs : nombre de côtés, longueur des côtés, aire de la surface, forme... Parmi ces attributs, la variable dépendante est la forme, car c'est sur la valeur de cet attribut que la technique d'apprentissage doit faire sa prédiction. Donc, « carré », « rectangle » et « cercle » sont tous des exemples de classes.

Pour le reste de l'article, l'accent est mis sur les méthodes de classification. Si une variable dépendante est numérique, elle serait au préalable **discrétisée**, c'est-à-dire que les valeurs de la variable dépendante seraient partitionnées, les parties devenant les nouvelles valeurs possibles de la variable transformée [5]. Par exemple, il est possible de discrétiser la variable « âge » en tranches d'âge ou le salaire d'un individu en classes salariales.

2.2. L'évaluation de l'apprentissage

Un modèle n'est pratiquement jamais parfait : il n'est pas en mesure de décrire entièrement le système sous-jacent. Les raisons sont multiples, mais le résultat est le même : une perte de fiabilité du modèle. Dans tous les cas, il faut pouvoir faire l'évaluation de la fiabilité du modèle créé. Pour ce faire, il existe différentes techniques.

La **performance** de l'apprentissage est donc une mesure qui représente la capacité du modèle créé à bien représenter un système. Pour estimer la performance, il faut procéder à certains tests et recueillir des mesures la décrivant. Une méthode populaire est de séparer les instances en deux groupes : les **données**

4. Par abus de langage, les variables indépendantes sont souvent appelées attributs sans autres précisions. Dans ce contexte, un modèle est formé d'attributs et d'une variable dépendante.

5. Pour plusieurs algorithmes, il est également nécessaire de discrétiser les attributs afin que l'algorithme puisse être appliqué.

d'apprentissage (ou **d'entrainement**), et les **données tests**. Les données d'apprentissage servent à la construction du modèle alors que les données test sont utilisées pour vérifier la fiabilité du modèle construit.

Les données d'apprentissage regroupent les instances desquelles l'algorithme choisi doit apprendre et construire son modèle. Une fois l'apprentissage terminé, une classification des données est accomplie. Cette classification s'opère en notant la valeur de la variable dépendante prédite par le modèle selon la valeur des variables indépendantes de l'instance à classer. Il est ensuite possible de déterminer un certain **taux d'erreur**, donné par le ratio du nombre d'instances mal classées sur le nombre d'instances étudiées.

Aussi, il peut être intéressant de calculer le taux d'erreur sur chacune des classes individuellement : peut-être que le modèle ne classe incorrectement que l'une d'elles. Cette information peut être particulièrement importante si le coût de mauvaise classification varie en fonction des classes.

Imputer des coûts aux bonnes et mauvaises classifications permet d'avoir un meilleur contrôle sur l'importance de celles-ci. Il est possible de définir une **fonction de coût** C qui renseigne sur le coût d'une classification et une autre fonction N qui indique le nombre d'instances classées d'une certaine façon. Par exemple, dans la table 1, le coût de la mauvaise classification d'une instance qui aurait dû être classée A mais qui a été classée B est donné par $C(A, B)$. Pour déterminer la performance d'un classificateur, il suffit de calculer le coût de classification d'un groupe d'instances :

$$\text{Cout de classification} = \sum_{i,j \in \{A,B,C\}} C(i,j) \cdot N(i,j)$$

où $N(i, j)$ représente le nombre d'instances ayant comme classe i et classées dans la classe j.

		Valeurs prédites		
		A	B	C
Valeurs réelles	A	0	5	10
	B	10	0	20
	C	15	30	0

TABLE 1. Exemple d'une matrice de coûts de dimension 3.

3. L'arbre de décision

Cette section présente en plus de détails le fonctionnement de l'algorithme qui sert à la création des arbres de décision. Avant d'entrer dans les détails, il faut d'abord se familiariser avec le vocabulaire lié aux arbres de décision. Ensuite, une présentation de l'algorithme général est donnée pour terminer avec les avantages et les inconvénients de cette méthode.

3.1. Définitions

Parmi les algorithmes de classification, l'un des plus simples d'utilisation et d'interprétation, tout en gardant des performances très respectables, est l'**arbre de décision** [**WF05**]. Existant sous plusieurs formes, l'arbre de décision est reconnu par le résultat de l'algorithme qui produit un modèle constitué d'un ensemble de règles de classification qu'il est possible de représenter sous forme d'**arbre** comme à la figure 1.

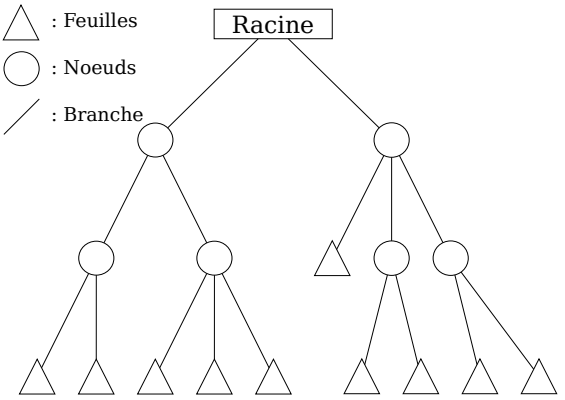

FIGURE 1. Un exemple d'arbre de décision qui illustre chaque composante de l'arbre.

Sur cette figure, il y a d'abord la **racine** de l'arbre qui se trouve tout en haut. Les instances à classifier sont insérées dans l'arbre à partir de la racine. Elle contient l'ensemble des données d'entrainement et doit donner une première séparation qui amènera à la classification.

Dans l'exemple de la figure 2, pour chaque instance à classer, il faut regarder d'abord le nombre de côtés. Selon la valeur du nombre de côtés (plus grand que $3,5$ ou plus petit ou égal à $3,5$), il faut suivre la **branche** correspondante. Par exemple, pour classer une figure à quatre côtés, il faut suivre la branche « $> 3,5$ » pour arriver au **noeud** « largeur ».

Un noeud contient l'ensemble des instances qui satisfont aux critères de séparation précédents. Ici, le noeud « hauteur » le plus élevé de l'arbre contient donc l'ensemble des instances qui ont une valeur « $> 3,5$ » à l'attribut « nbCotes » et une valeur « $> 9,71$ » à l'attribut « largeur ». Les noeuds, comme la racine, donnent d'autres critères de segmentation. En fait, la racine n'est qu'un noeud spécial : c'est le premier rencontré et il est le point d'entrée de toutes les instances.

Supposons maintenant que la figure, en plus de posséder quatre côtés, ait également une largeur de 4 et une aire de 8. Au bout du **chemin** construit à partir des branches qui satisfont aux caractéristiques de cette figure se trouve une **feuille**, nommée ainsi parce qu'elle-même n'a pas de branches. Une feuille est spéciale : chaque instance qui s'y trouve, ayant suivi le chemin de la racine jusqu'à la feuille, est classée de la même façon par l'arbre de décision.

La classification d'une nouvelle instance se fait à l'aide de l'arbre : il faut trouver la feuille de l'arbre qui contient l'instance à classer selon le critère de

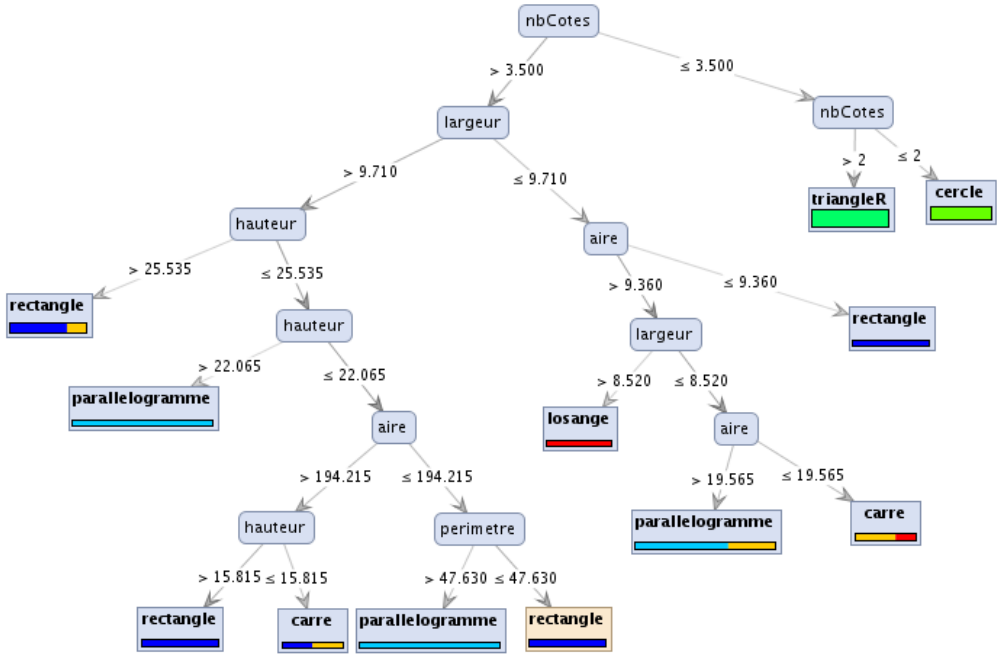

FIGURE 2. Exemple d'un arbre de décision .

segmentation et attribuer à l'instance la classe la plus fréquente des autres instances de cette feuille. Par conséquent, selon l'exemple, la figure décrite plus haut se retrouve dans la feuille étiquetée « rectangle » : elle est donc classée comme étant un rectangle.

La figure 3 donne un aperçu du vocabulaire lié aux arbres. Par exemple, le noeud A précède le noeud B et C, A est donc le **parent** de B et B est le **fils** de A. De plus, il est possible de considérer B comme étant la racine d'un **sous-arbre** de l'arbre principal. B et C sont également les **enfants** de A. Aussi, B est le frère de C.

3.2. Algorithme de création du modèle

Les arbres de décision sont apparus simultanément dans deux disciplines, soit en statistiques et en apprentissage automatique [**HK06**] [**WF05**]. J. Ross Quinlan fut celui qui travailla sur les arbres de décision, au cours des années 70 et au début des années 80, du point de vue de l'apprentissage automatique. Il donna naissance à l'algorithme *ID3* [**Qui86**] et à ses successeurs le *C4.5* [**Qui93**] et le *C5.0* qui devinrent une référence sur laquelle les autres se basent aux fins de comparaison [**HK06**]. C5.0 étant sous licence propriétaire et son code source n'étant pas disponible, les chercheurs continuent à se baser sur le C4.5 [6] [**WF05**].

Dans les mêmes années, L. Breiman, J. Friedman, R. Olshen et C. Stone ont développé la version statistique de l'arbre de décision en inventant l'algorithme *CART* [**BFOS84**] [7].

6. Plus particulièrement la révision 8 du C4.5 ou C4.5R8 publiée par Quinlan.
7. D'autres algorithmes d'arbre de décision d'intérêt : QUEST, CHAID, SLIQ et SPRINT.

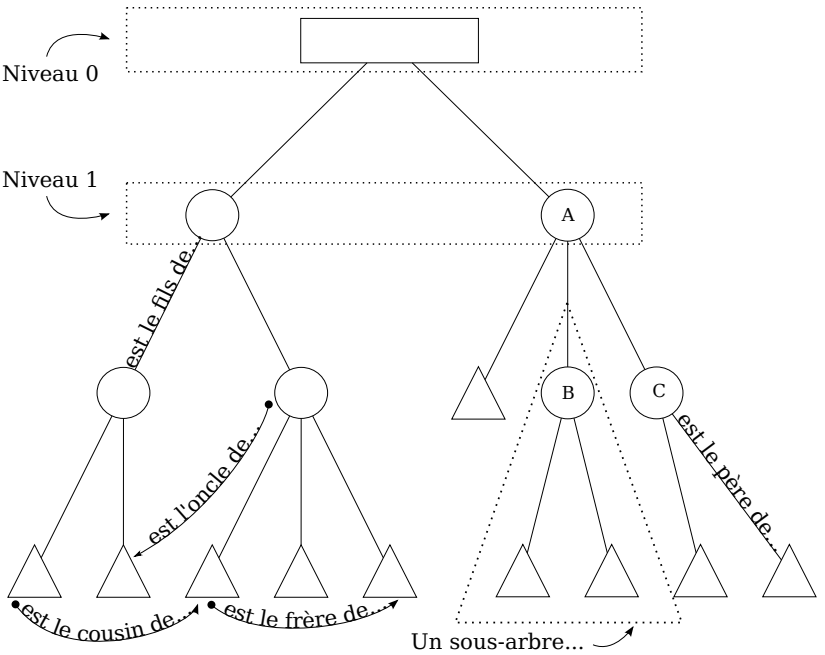

FIGURE 3. Illustration du vocabulaire relatif aux arbres.

Le fonctionnement de l'arbre de décision est simple, c'est ce qui le rend si facile d'interprétation et si populaire. Son apprentissage se fait par étape. À chaque étape, un noeud est divisé en plusieurs enfants selon une condition sur les valeurs d'un attribut. Pour ce faire, il faut avoir un moyen de trouver avec quel attribut créer la segmentation, c'est-à-dire la division du noeud. Pour ce faire, il existe plusieurs **critères de segmentation** qui servent à diviser les données qui se trouvent dans un noeud. Ce critère choisit de façon judicieuse un attribut dans le but d'avoir le meilleur modèle de classification possible. Dans l'exemple illustré à la figure 2, l'algorithme a décidé de séparer la racine selon l'attribut « NbCotes » en segmentant les figures ayant moins de $3,5$ côtés et ceux en ayant au moins $3,5$.

L'un de ces critères de segmentation est le gain en information basé sur l'entropie de Shannon [8] [9] décrit par Claude Shannon vers la fin des années 1940 et représente en quelque sorte l'information véhiculée par un certain support (ici un attribut) [**HK06**].

3.3. Avantages

Un principal avantage de l'arbre de décision est sa capacité à représenter un système sous forme d'arbre ou d'une suite de règles de classification des données. Cette représentation du modèle illustre, en plus d'une classification, la logique derrière le modèle qui est en elle-même une information qui peut s'avérer intéressante [**WF05**].

8. Sans autre précisions sur l'algorithme, cet article fait référence à l'algorithme C4.5 comme algorithme d'arbre de décision par défaut. Cet algorithme utilise le critère le l'entropie de Shannon pour son critère de segmentation.

9. Quelques autres critères intéressants : l'indice de Gini, le χ^2 et leurs dérivés.

De plus, il devient très facile de classer les nouvelles instances une fois l'arbre construit. En effet, comme l'apprentissage est effectué lors de la construction de l'arbre, une fois celui-ci terminé, chaque instance à classer n'a qu'à suivre un chemin de l'arbre, ce qui est très efficace [**WF05**].

Aussi, puisque l'algorithme reste assez simple, il est souvent privilégié par rapport aux autres méthodes plus lourdes à mettre en place, d'autant plus qu'un arbre bien construit donne des performances similaires à d'autres techniques plus complexes sur de petits ensembles de données [**RN94**] [**WF05**].

3.4. Inconvénients

La segmentation en de plus en plus petits ensembles de données pendant l'apprentissage amène un problème : la fragmentation des données. Une fois l'arbre rendu à quelques niveaux, il doit travailler sur un noeud contenant un nombre restreint d'instances, laissant place à un biais plus important dans ses prédictions [**RN94**].

Un autre désavantage est la tendance des algorithmes d'arbres de décision à favoriser la séparation des données en d'importants groupes en ignorant, ou reléguant à plus tard, la classification d'un groupe marginal de données. Or, ces groupes marginaux peuvent être tout aussi intéressants, sinon plus, que la structure générale des données [**CF00**].

Dans le même esprit, certaines fonctions de gain en information tendent à être biaisées envers les sous-ensembles de grosseurs semblables et multiples, ce qui ne reflète pas toujours la réalité [**RN94**].

Concernant la séparation d'arbres en sous-arbres, il peut se produire un problème de réplication des sous-arbres. Si un attribut sépare les données en deux groupes, il est possible, et même fréquent que chacun des sous-groupes puisse être redivisé selon une même série d'attributs, conduisant à deux sous-arbres avec la même structure et introduisant de la redondance dans le modèle [**RN94**].

Il est aussi possible d'imaginer comment un arbre pourrait être très différent si un attribut opérant une séparation au début de l'arbre était remplacé par un autre. Modifier quelques instances d'un échantillon d'entrainement peut avoir des effets importants sur la construction du modèle, par exemple en changeant le premier attribut choisit par l'arbre, ce qui le rend instable aux changements [**WF05**]. Le problème pourrait provenir seulement du choix des instances constituant les données d'entrainement. L'arbre est donc sensible à la façon d'échantillonner. Une façon de repérer ce problème est d'utiliser des techniques comme la validation croisée pour réduire l'effet de l'échantillonnage sur la mesure des performances de l'arbre ou une technique d'ensemble (voir section 4.1) pour diminuer l'importance de l'échantillonnage dans la classification des données.

L'un des pires défauts de l'arbre de décision est son incapacité de prendre plus d'un attribut en compte à la fois [**WF05**]. Bien qu'il existe des algorithmes d'arbre de décision permettant de découvrir des relations linéaires entre les variables, les algorithmes d'arbre de décision communément utilisés ne sont pas en mesure de répondre à ce problème. Par exemple, il est montré à la figure 4 un exemple où un arbre de décision complique beaucoup la classification comparé à la solution « simple » qui consisterait à tracer une droite séparant les données.

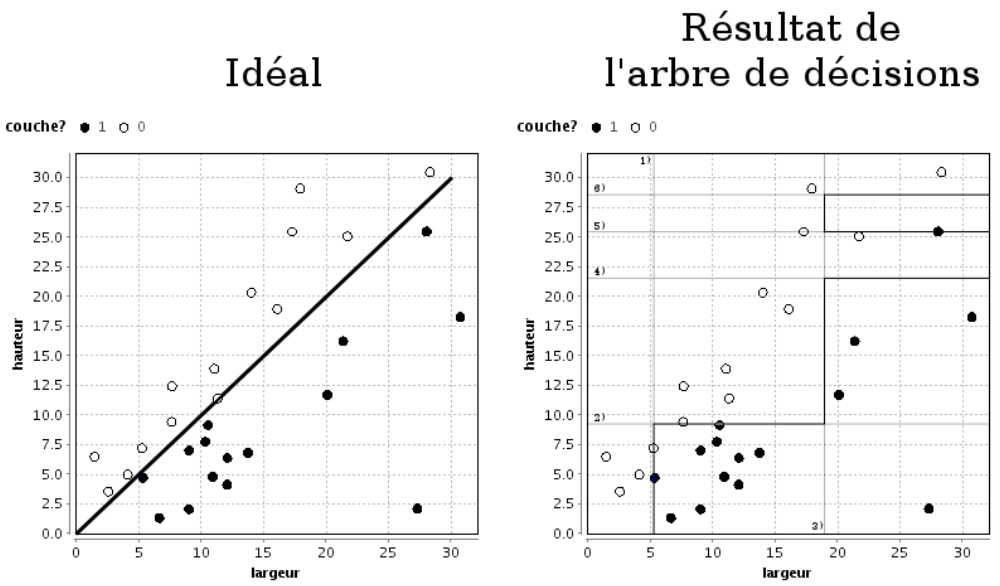

FIGURE 4. Cas où une séparation par une fonction linéaire serait plus appropriée, difficile à effectuer avec un arbre de décision.

Un algorithme, tel le C4.5, fait une segmentation ne s'appuyant que sur un attribut à la fois. Ceci donne des frontières de séparation constantes, c'est-à-dire une fonction constituée de lignes horizontales ou verticales seulement. Ceci n'est pas toujours idéal comme illustré à la figure 4 où une droite aurait suffit.

Ces inconvénients sont les principales motivations du reste de ce rapport. Les techniques hybrides ont été développées dans le but d'atténuer les conséquences négatives de l'arbre de décision tout en renforçant ses points positifs [**AH98**]. Plusieurs de ces défauts sont donc adressés dans les sections suivantes sur la combinaison de méthodes d'apprentissage.

4. Combiner l'arbre de décision à d'autres techniques

Alors que les arbres de décisions simples sont la plupart du temps basés sur les mêmes principes de segmentation des données, plusieurs lacunes au fil du temps sont ressorties. C'est dans l'objectif de réduire l'impact de ces défauts que les chercheurs ont voulu modifier de façon fondamentale l'algorithme, tout en essayant de garder la particularité qui le rend si populaire : sa facilité d'interprétation.

Pour y parvenir, quelques techniques ont été mises de l'avant. Parmi celles-ci, il est possible de retrouver les techniques d'apprentissage d'ensembles, et les algorithmes hybrides[10].

10. Plusieurs classifications des algorithmes hybrides sont proposées dans la littérature. Notamment, Goonatilake et Khebbal [**GK95**] proposent une classification issue du domaine de l'intelligence artificielle que Anand et Hughes [**AH98**] réfutent pour son application au forage de donnée.

La suite de l'article fait l'objet, entre autres, de techniques de combinaison entre un arbre de décision et une tout autre méthode de forage de données comme la machine à vecteurs de support, le réseau de neurones, l'algorithme bayésien naïf ou l'algorithme génétique. Ces autres méthodes sortant du cadre de ce document, le lecteur désireux d'en savoir plus est invité à consulter les références de l'article, tout particulièrement [**HK06**] et [**WF05**].

4.1. Les techniques d'ensembles

Une première tactique utilisée afin de combiner les résultats de plusieurs itérations d'un même algorithme ou même de plusieurs algorithmes différents consiste à faire l'agrégation de différents modèles, c'est-à-dire de tenter de les combiner en un plus robuste et performant. Cette technique est appelée l'**apprentissage par ensembles** (*ensemble learning*) [**WF05**].

Parmi les techniques d'apprentissage par ensembles les plus populaires se retrouvent l'**agrégation par rééchantillonage**(*bagging* qui vient de *bootstrap aggregating*) et le **renforcement**(*boosting*). Chacune de ces techniques fonctionne selon le principe de combinaison de plusieurs cycles d'apprentissage afin d'arriver à un modèle plus robuste. Voici une brève explication de leur fonctionnement.

L'agrégation par rééchantillonage utilise plusieurs itérations d'un même algorithme comme l'arbre de décision. Dans le but de créer des modèles différents, les arbres sont construits à partir des données d'entrainement initiales échantillonnées avec remise. Il résulte donc d'un certain nombre d'ensembles de données d'entrainement, tous ayant le même nombre d'instances que l'ensemble de données d'entrainement initial. L'arbre de décision étant un algorithme plutôt instable, les perturbations entre chacun des échantillons sont suffisantes pour créer des modèles différents d'une itération à l'autre [**WF05**]. C'est donc une façon de se servir d'une faiblesse de l'arbre de décision à son avantage. Lors de la classification des nouvelles instances, chaque instance est classée dans tous les modèles et la classe la plus fréquente (pondérée selon la performance des arbres individuels ou non) est choisie comme étant la classification finale [**WF05**].

De l'agrégation par rééchantillonage découle une technique très efficace et qui gagne en popularité appelée **forêt d'arbres décisionnels** (*random forests*). Cette technique, inventée par Breiman [**Bre96**] (l'inventeur du modèle CART, l'un des premiers arbres de décision), commence à faire sa place dans l'arsenal des foreurs de données.

L'algorithme de la forêt d'arbres décisionnels construit un nombre donné d'arbres de décision à l'aide d'un sous-ensemble restreint d'attributs pour chaque arbre dans le but d'insérer un effet aléatoire. Les instances sont toujours choisies au hasard et avec remise.

Ensuite, comme pour l'agrégation par rééchantillonage, une instance est classée par chacun des arbres et la classe résultante la plus fréquente est utilisée pour la classification.

La seconde technique d'apprentissage par ensemble est le **renforcement**. Cette technique tente d'améliorer chacune des itérations de l'algorithme en mettant un poids plus important sur les instances mal classées pour les itérations suivantes. De cette façon, chaque fois qu'un arbre est reconstruit, la priorité de

classification des données mal classées précédemment est plus élevée, ce qui force l'algorithme à tenter de classer ces instances correctement le plus tôt possible. Contrairement à l'agrégation par rééchantillonage où le nombre d'itérations est arbitraire, ici certaines conditions d'arrêts sont définies selon le taux d'erreur du modèle produit [**WF05**]. Le lecteur intéressé peut en apprendre davantage en s'informant sur l'algorithme AdaBoost [**FS95**], l'implantation la plus populaire du renforcement.

4.2. Tri des attributs

Le tri des attributs peut améliorer significativement la performance d'une méthode d'apprentissage. Il a été adopté par plusieurs auteurs pour l'hybridation d'un arbre de décision, dans le but d'améliorer les performances de classification.

Par exemple, dans [**WF05**] il est proposé d'utiliser un arbre de décision afin de faire un premier tri des attributs et de ne retenir que ceux utilisés par le modèle résultant. Bien sûr, il est inutile de faire cela si la méthode d'apprentissage est elle-même un arbre de décision.

C'est le modèle utilisé par Kissi et Ramdani [**KR10**] qui propose d'utiliser un arbre de décision pour faire une première sélection des attributs. Les attributs restants sont ensuite fournis à un réseau de neurones qui s'occupe de faire la classification de façon usuelle.

Alors qu'il est possible d'utiliser un arbre de décision avant un réseau de neurones, rien n'a empêché Fong et coll. [**FSBA09**] de faire l'inverse pour leur problème de prédiction de l'admission dans les universités. De plus, comme c'est l'arbre de décision qui s'occupe de fabriquer le modèle de classification, l'algorithme retient l'avantage principal de l'arbre de décisions : sa facilité d'interprétation. Leur algorithme, appelé RSAU, utilise encore une fois le principe de sélection des attributs pour améliorer la performance de leur modèle. Cette fois, c'est le réseau de neurones qui a pour objectif de faire un premier tri des attributs pour permettre au C4.5 de faire un modèle de prédiction plus performant.

Dans un article de Nair et coll. [**NMS10**], les auteurs utilisent un arbre de décision afin de faire un premier tri des attributs avant de laisser le soin aux esquisses d'ensemble (*rough set*) de faire la classification.

Le tri des attributs avant l'apprentissage est une étape importante. Par exemple, une bonne raison de procéder à un tel tri est qu'il a été démontré que l'ajout d'un attribut binaire provenant de tirs d'une pièce non biaisée fait décroitre les performances du modèle C4.5. Les performances résultantes affichent un taux d'erreur de 5 à 10 % plus élevé [**WF05**]. Le tri des attributs par un second algorithme d'apprentissage différent est donc une bonne façon d'éliminer le bruit causé par des attributs superflus.

Bala et coll. [**BHV+95**] introduisent une nouvelle technique pour faire un tri préventif des attributs fournis à l'algorithme de construction de l'arbre de décision à l'aide d'un algorithme génétique. L'idée est reprise par Stein et coll. [**SCWH05**] et Kornienko et coll. [**KB03**] mais avec quelques nuances.

L'analyse en composantes principales [11] peut également être utilisée conjointement avec l'arbre de décision afin de s'assurer que les attributs étudiés couvrent

11. Pour plus d'informations au sujet des analyses en composantes principales et leur importance dans le domaine du forage de données, une bonne introduction est donnée dans [**WF05**].

le plus de variance (donc d'information) possible afin de permettre à l'algorithme de l'arbre de décision de faire des segmentations qui ne sont pas nécessairement parallèles aux axes originaux. En reproduisant une analyse en composante principale à chaque noeud, il est possible d'arriver à un arbre de décision multivarié, c'est-à-dire qui prend compte de plus d'une variable à la fois [**WF05**].

4.3. Partition des instances

Un autre objectif recherché par les auteurs d'arbres de décision hybrides est d'effectuer une partition des instances, souvent pour faire en sorte qu'une partie des instances ne soit pas classée par le même algorithme. La séparation en multiples parties permet, par exemple, d'utiliser un algorithme rapide, mais moins performant en combinaison avec un algorithme performant, mais moins rapide.

C'est l'approche utilisée par Kumar et coll. [**KG10**] qui tentent d'améliorer le temps de classification d'une machine à support de vecteur en classant les instances facilement classables par l'arbre de décision et reléguant les cas plus délicats à la machine à vecteurs de support.

Comme il a déjà été mentionné, un arbre de décision est une méthode efficace si utilisée avec des variables catégoriques. Par contre, l'utiliser pour la classification d'une variable numérique implique une discrétisation de la variable au préalable pour l'arbre de décision de Quinlan [**Qui93**]. Or, c'est l'inverse pour un réseau de neurones : une variable numérique n'a aucun mal à être utilisée par le modèle, alors que pour les variables catégoriques l'algorithme doit utiliser des stratagèmes pas toujours efficaces pour la classification des données. Les mêmes restrictions s'appliquent sur les attributs utilisés pour l'apprentissage.

C'est en conséquence de ceci que Zhou et Chen [**ZC02**] ont inventé HDT, un algorithme hybride qui utilise un arbre de décision pour gérer l'information des variables indépendantes catégoriques et un réseau de neurones pour s'occuper des variables indépendantes numériques.

Une approche similaire a été adoptée par Arentze et Timmermans [**AT07**] qui utilisent un modèle logit [12] afin de faire la séparation d'un attribut numérique.

Tsai et Wang [**TW09**], quant à eux, filtrent les instances avec un réseau de neurones : seules les instances bien classées sont utilisées comme données d'apprentissage pour bâtir l'arbre de décision.

4.4. Modification du comportement des feuilles

Les hybrides de cette section visent à modifier la façon dont une instance est classée. Par exemple, l'algorithme C4.5 effectue le classement d'une instance en prenant, dans une feuille, la classe la plus fréquente. C'est ce principe qui est remplacé dans cette section.

Pour des raisons de performance, Kohavi [**Koh96**] et Wang et coll. [**WZX10**] ont décidé d'utiliser l'algorithme bayésien naïf pour continuer le travail de l'arbre de décision une fois l'arbre élagué. En effet, pendant le processus d'élagage, l'algorithme bayésien naïf est utilisé afin de tester si l'application de cet algorithme

---

Pour une analyse plus approfondie, il peut être intéressant de se référer à un ouvrage statistique sur le sujet.

12. Plus d'informations sur le modèle logit peuvent être obtenues en cherchant une description de la régression logistique, décrite entre autres dans [**WF05**].

dans un noeud précédant les feuilles améliore les performances du classificateur. Une fois l'arbre élagué, il y a donc certaines feuilles qui ont effectivement une autre technique totalement implantée à partir des instances disponibles à ce point.

Dans leur article, Seewald et coll. [**SPW01**] ont pris l'idée de Kohavi [**Koh96**] et l'ont étendue. Au lieu de seulement remplacer le classificateur standard par un classificateur bayésien, ils ont laissé le choix à l'utilisateur entre trois classificateurs : la méthode bayésienne naïve, IB1 (un algorithme du plus proche voisin) et la régression linéaire.

Bien que ce ne soit pas l'algorithme lui-même qui fait le choix de la méthode, il est concevable que chacune d'elles puisse être testée et que seule la plus performante dans chaque noeud soit retenue.

La classification par régression linéaire est décrite par Frank et coll. [**FWI+98**] dans leur article sur les arbres modèles (*model trees*).

Ce principe rappelle la régression logistique (une fonction qui minimise l'erreur entre les données et une courbe logistique) qui pourrait être mixée avec un arbre de décision. En effet, c'est ce que Landwehr et coll. [**LHF03**] ont utilisé dans leur article.

Cet algorithme a été produit dans le but de diminuer le biais de la classification en rendant possible la classification locale des données. Ici, l'algorithme C4.5 ne sert qu'à subdiviser l'espace des solutions en régions sur lesquelles est appliquée une méthode d'apprentissage locale.

Le raisonnement par cas, une autre méthode, peut être coûteux en ressources. Dans le but de vaincre ce problème, Bouyer et coll. [**BAM07**] utilisent un arbre de décision pour localiser l'apprentissage, ce qui le rend plus performant et plus rapide à exécuter.

Suivant une autre idée, il est possible de remarquer que lors de la construction d'un arbre de décision, l'algorithme a tendance à privilégier les séparations en groupes importants, c'est-à-dire qui contiennent beaucoup d'instances. Les plus petits groupes sont souvent mal classés puisque du point de vue de l'algorithme ils sont plus propices à refléter une classification erronée, surentrainée ou tout simplement causée par du bruit dans les données d'apprentissage.

Or, il se trouve que souvent l'information intéressante et inconnue se trouve être une conséquence de l'existence de petits groupes d'instances qui sont souvent mis de côté dans le processus de décision. Le problème est de savoir comment les reconnaitre de façon efficace.

Carvalho et Freitas [**CF00**] [**CF04**] se sont penchés sur la question et ont décidé de recourir à un algorithme génétique afin de régler le problème. L'idée est tout simplement de recourir au principe d'évolution de l'algorithme génétique sur l'ensemble des instances qui tombent dans une feuille avec un faible nombre d'instances afin de leur attribuer des règles de classification plutôt que de les élaguer hâtivement. L'esprit de cette méthode a été repris dans les travaux de Noumbi [**Nou**].

Sur un autre plan, l'utilisation de la conversion du modèle de l'arbre de décision en règles de classification est utilisée par Pistori et Neto [**PN03**] qui ont développé un hybride entre un arbre de décision et la théorie des automates. Une des caractéristiques de cet algorithme nommé *Adaptree* est qu'il tente de changer

l'ordre d'apparence des attributs dans les règles en utilisant quelques stratégies syntaxiques.

4.5. Altération de la gestion des coûts

Comme discuté dans la section 2.2 sur l'évaluation des performances des méthodes d'apprentissage, une pratique efficace est d'affecter des coûts aux différentes classifications correctes ou erronées. Une façon de pousser cette technique un peu plus loin est de demander à l'algorithme de se servir de cette information pour tenter de minimiser les coûts de classification et de test.

Cette modification, qui semble raisonnable, demande souvent des changements à l'algorithme de base assez importants. Par conséquent, élaborer un hybride qui combine une approche qui tient compte des coûts et une autre qui effectue la classification est plus complexe que les solutions présentées jusqu'à maintenant. Dans tous les cas, l'augmentation de la complexité est compensée par l'acquisition d'une méthode de classification très puissante qui peut s'adapter aux besoins de la personne nécessitant de l'information extraite de la base de données.

Une autre problématique concernant l'arbre de décision est attribuée à l'utilisation de moins en moins de données pour extraire ses règles de classification. Par conséquent, la signifiance des conditions de séparation plus loin dans les branches diminue en même temps que le nombre de données traitées à chaque noeud.

C'est le problème que Sheng et coll. [**SL05**] ont voulu régler. Ils ont eu l'idée de combiner un arbre de décision (C4.5) avec la méthode bayésienne naïve. Dans leur algorithme (*DTNB*), repris en détail dans la section 5, l'arbre de décision dicte quels tests doivent être effectués tandis que le classificateur bayésien effectue la classification en se basant sur l'ensemble des données disponibles.

Dans un autre ordre d'idée, Turney [**Tur95**] décrit une manière originale d'utiliser le principe d'évolution des algorithmes génétiques pour influencer la construction d'un arbre de décision. Cette méthode est appelée *ICET*.

Dans la méthode ICET, la fonction du coût en information est utilisée conjointement avec l'algorithme génétique qui insère des coûts supplémentaires à chaque noeud, ce qui permet de faire varier l'ordre d'utilisation des attributs dans l'arbre de décision. En introduisant un biais dans les coûts des attributs, l'arbre résultant a la chance d'avoir une meilleure vue d'ensemble du problème qui ne minimise peut-être pas le coût à court terme, mais qui permet d'avoir de meilleurs résultats à long terme en cherchant un minimum global plutôt que de s'attarder sur les minimums locaux.

4.6. Techniques diverses

Pratiquement tous les algorithmes d'apprentissage ont des paramètres qui peuvent être modifiés pour altérer la construction du modèle. Même les méthodes dites « non paramétriques », comme l'arbre de décision, possèdent certaines valeurs qui peuvent être modifiées comme la différence nécessaire entre l'erreur d'un noeud et l'erreur moyenne requise pour procéder à la construction d'un nouveau niveau.

Une méthode d'hybridation d'algorithmes employée afin de définir les paramètres d'un modèle est proposée par Remeikis et coll. [**RSM**] qui utilisent une

méthode similaire au DT-KBANN de Hewahi [**Hew09**]. Ils utilisent un arbre de décision pour d'abord étudier les données et en extraire des règles de décisions. Ces règles sont ensuite converties en une topologie pour un réseau de neurones dépendant des disjonctions et des conjonctions des règles.

Une nouvelle façon d'utiliser un algorithme génétique est donnée par Zorman et coll. [**ZČGM04**]. Ils se servent d'un algorithme génétique afin de trouver les meilleurs paramètres à utiliser pour la construction de leur arbre de décision.

Une autre technique d'hybridation est d'utiliser l'information découverte par un algorithme et de l'insérer sous forme d'un nouvel attribut dans le second algorithme pour l'appuyer dans sa classification. Comme le nouvel attribut créé est souvent déjà un bon indicateur de la classe à laquelle appartient une instance à classer, le second algorithme peut se concentrer sur les cas particuliers, un peu à l'image du renforcement présenté à la section 4.1.

Peddabachigari et coll. [**PAGT05**] discutent d'un hybride, appelé *DT-SVM*, qui mélange un arbre de décision à une machine à vecteurs de support. Leur idée consiste à construire un arbre de décision dont les n feuilles sont numérotées de 1 à n. Cette numérotation est ensuite convertie en un nouvel attribut (ayant n valeurs) qui devrait aider la machine à vecteurs de support à bâtir son modèle puisque les instances semblables, selon l'arbre de décision, auront des valeurs identiques.

Malheureusement, les auteurs ne sont pas arrivés à des résultats très concluants contrairement à Chong et coll. [**CAP05**]. Leur algorithme hybride, le *DTANN*, a donné de meilleurs résultats qu'un réseau de neurones, un arbre de décision ou une machine à vecteur de support employés séparément. Le principe du DTANN est le même que celui employé par Peddabachigari et coll. [**PAGT05**].

Une extension viable du DTANN est d'utiliser des techniques d'analyse de groupement des données, une technique faisant partie des apprentissages non supervisés, afin de créer des groupes semblables entre eux. C'est exactement ce qu'ont fait Bose et Chen [**BC09**] : ces groupes peuvent être convertis en un attribut qui peut ensuite aider à la construction d'un nouveau modèle. Cette façon de faire pourrait même s'étendre à tout les types d'apprentissage supervisé. En ajoutant cette étape, il est possible d'utiliser toute la puissance des algorithmes non supervisés au service des algorithmes supervisés.

5. Étude de cas

Cette section sert à illustrer la façon d'hybrider deux algorithmes pour arriver à en créer un meilleur. À cette fin, on reprend l'idée de Sheng et coll. [**SL05**] pour implanter un arbre de décision qui sait prendre en considération les coûts de classification et les **coûts de test** qui représentent la pénalité d'utilisation des valeurs d'un attribut.

En pratique, il arrive que l'étude d'une base de données puisse se faire directement avec les données disponibles, comme pour l'analyse des ventes faites durant une année. Il arrive également qu'il y ait un coût relié à l'acquisition de l'information pertinente à l'étude. Si l'algorithme utilise un certain attribut, le coût de test rattaché à cet attribut doit être payé. Le but de ces coûts de test est d'obliger l'algorithme à n'utiliser que les attributs les plus pertinents.

Ce genre de classificateur est employé en médecine où l'acquisition de résultats pour chaque test occasionne des frais et où un mauvais diagnostic peut entrainer des pertes financières et humaines.

Une première façon d'utiliser une fonction de coûts pour déterminer la performance d'un arbre de décision pourrait se faire en remplaçant la fonction de segmentation (originalement basée sur l'entropie de l'information) par la fonction de coût : le but deviendrait de minimiser les coûts de mauvaise classification. En ajoutant une fonction de coût de test, l'algorithme essaie de minimiser les coûts de test et de classification en parallèle, donc d'avoir un arbre compact qui ne cherche pas à tout prix à minimiser l'erreur de classification.

Il pourrait être intéressant d'avoir une classification plus nuancée surtout dans le cadre de tests médicaux. L'arbre de décision usuel ne donne qu'une classification booléenne : l'instance fait partie d'une classe ou elle n'en fait pas partie. Il est possible d'établir la probabilité qu'une instance dans une feuille appartienne à une certaine classe en gardant les ratios d'instances d'apprentissage affichant une classe. Par exemple, une feuille ayant quatre instances lors de l'apprentissage, dont trois sont présentes dans la classe A et une dans la classe B, indiquerait normalement que cette feuille classe les instances s'y trouvant dans la classe A. Or, il serait également possible d'indiquer qu'une instance appartenant à cette feuille a 75 % de chance d'appartenir à la classe A et 25 % de chance d'appartenir à la classe B. L'algorithme de classification bayésienne est un bon exemple d'algorithme qui effectue ce travail.

Malheureusement, cette technique présente quelques défauts qui peuvent être règlés grâce à l'algorithme *DTNB* de Sheng et coll. [**SL05**]. Cette façon de faire bâtit un arbre de décision prenant les coûts de test et de classification comme expliqué un peu plus tôt. Par contre, pour la classification, cette méthode bâtit également, en parallèle, un classificateur bayésien qui s'occupe de la classification. L'astuce réside dans le rôle joué par l'arbre de décision et le classificateur bayésien : l'arbre de décision sert à déterminer quels tests effectuer et le classificateur bayésien s'occupera de la classification en n'utilisant que certains tests, tel qu'indiqué par l'arbre.

Avant d'entrer dans les détails de cet algorithme, voici une introduction du classificateur bayésien utilisé par cette méthode d'apprentissage.

5.1. Application naïve de la théorie bayésienne

Pour la classification de données, le foreur de données dispose de fondements théoriques mathématiques qui peuvent s'incorporer à l'apprentissage comme le théorème de Bayes. Les développements en forage de données ont permis d'utiliser le théorème de Bayes afin de prendre en considération les données d'entrainement et produire la probabilité qu'une instance appartienne à une classe selon un groupe d'attributs.

Voici un rappel du théorème de Bayes. Soit A et B deux évènements, alors

$$\mathbb{P}(A|B)\mathbb{P}(B) = \mathbb{P}(A \cap B) = \mathbb{P}(B|A)\mathbb{P}(A) \Leftrightarrow \mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

où $\mathbb{P}(A|B)$ représente la probabilité d'un évènement A sachant que l'évènement B s'est réalisé. Dans cette formule, $\mathbb{P}(A|B)$ est la probabilité **à postériori** et $\mathbb{P}(A)$ est la probabilité **à priori**.

Soit $A = (A_1 A_2 \dots A_n)$ un vecteur de dimension n où les A_i ($1 \leq i \leq n$, $i, n \in \mathbb{N}_*$) sont les attributs d'une instance à distribuer selon les classes de la variable dépendante C. Or, selon le théorème de Bayes :

$$\mathbb{P}(C|A) = \frac{\mathbb{P}(A|C)\,\mathbb{P}(C)}{\mathbb{P}(A)}$$

où $\mathbb{P}(A)$ est la probabilité que tous les attributs aient une certaine valeur :

$$\mathbb{P}(A) = \mathbb{P}\left(\bigcap_{i=1}^{n} A_i\right),$$

ce qui mène à

$$\mathbb{P}(C|A) = \frac{\mathbb{P}(A|C)\,\mathbb{P}(C)}{\mathbb{P}(A)} = \frac{\mathbb{P}(\bigcap_{i=1}^{n} A|C)\,\mathbb{P}(C)}{\mathbb{P}(A)}. \tag{1}$$

Il suit, en utilisant la définition de la probabilité conditionnelle,

$$\begin{aligned}
\mathbb{P}(C)\,\mathbb{P}\left(\bigcap_{i=1}^{n} A \,\middle|\, C\right) &= \mathbb{P}(C)\frac{\mathbb{P}(A_1 \cap C)}{\mathbb{P}(C)}\frac{\mathbb{P}((\bigcap_{i=1}^{n} A_i) \cap C)}{\mathbb{P}(A_1 \cap C)} \\
&= \mathbb{P}(C)\,\mathbb{P}(A_1|C)\,\mathbb{P}\left(\bigcap_{i=2}^{n} A_i \,\middle|\, C \cap A_1\right) \\
&= \mathbb{P}(C)\,\mathbb{P}(A_1|C)\,\frac{\mathbb{P}(A_2 \cap C \cap A_1)}{\mathbb{P}(C \cap A_1)}\frac{\mathbb{P}((\bigcap_{i=1}^{n} A_i) \cap C)}{\mathbb{P}(A_2 \cap C \cap A_1)} \\
&= \mathbb{P}(C)\,\mathbb{P}(A_1|C)\,\mathbb{P}(A_2|C \cap A_1)\,\mathbb{P}\left(\bigcap_{i=3}^{n} A_i \,\middle|\, C \cap \left(\bigcap_{i=1}^{2} A_i\right)\right) \\
&= \dots \\
&= \mathbb{P}(C)\,\mathbb{P}(A_1|C)\,\prod_{i=2}^{n}\mathbb{P}\left(A_i \,\middle|\, C \cap \left(\bigcap_{j=1}^{i-1} A_j\right)\right). \qquad (2)
\end{aligned}$$

C'est ici où il faut poser l'hypothèse d'indépendance conditionnelle des évènements A_i entre eux sachant C [13], c'est-à-dire que $\mathbb{P}(A_i \cap A_j|C) = \mathbb{P}(A_i|C)\mathbb{P}(A_j|C)$. Il suit

$$\mathbb{P}(A_i \cap A_j|C) = \mathbb{P}(A_i|C)\,\mathbb{P}(A_j|C) \Leftrightarrow \mathbb{P}(A_i|C \cap A_j) = \mathbb{P}(A_i|C).$$

En appliquant cette propriété à répétition sur le résultat 2, il suit

$$\begin{aligned}
\mathbb{P}(C)\,\mathbb{P}\left(\bigcap_{i=1}^{n} A \,\middle|\, C\right) &= \mathbb{P}(C)\,\mathbb{P}(A_1|C)\,\prod_{i=2}^{n}\mathbb{P}\left(A_i \,\middle|\, C \cap \left(\bigcap_{j=1}^{i-1} A_i\right)\right) \\
&= \mathbb{P}(C)\prod_{i=1}^{n}\mathbb{P}(A_i|C)
\end{aligned}$$

13. À noter que l'indépendance n'implique pas l'indépendance conditionnelle, et vice versa.

et en remplaçant dans l'équation 1, il en découle

$$\mathbb{P}(C|A) = \mathbb{P}(C)\frac{\mathbb{P}(\bigcap_{i=1}^{n} A|C)}{\mathbb{P}(A)} = \frac{\mathbb{P}(C)\prod_{i=1}^{n}\mathbb{P}(A_i|C)}{\mathbb{P}(A)}.$$

Une remarque concernant la probabilité $\mathbb{P}(A)$ qui se trouve au dénominateur : en pratique, il faut classifier une instance selon la classe qui maximise $\mathbb{P}(C|A)$. Or, lorsqu'il vient le temps de faire la comparaison entre $\mathbb{P}(C_i|A)$ et $\mathbb{P}(C_j|A)$, les deux côtés de l'inéquation ont un dénominateur $\mathbb{P}(A)$ qui peut être simplifié. Pour cette raison, les calculs sont simplifiés dès le départ. Il faut donc trouver la classe de C qui maximise :

$$\mathbb{P}(C|A) \propto \mathbb{P}(C)\prod_{i=1}^{n}\mathbb{P}(A_i|C)$$

où $\mathbb{P}(C)$ est le ratio du nombre d'instances dans la classe c sur le nombre d'instances au total. De la même façon, $\mathbb{P}(A_i|C)$ est le nombre d'instances de la classe C qui ont A_i comme attribut sur le nombre d'instances de la classe C. Une modification intéressante est d'utiliser des courbes normales pour calculer la probabilité $\mathbb{P}(A_i|C)$ d'un attribut numérique.

Pour classifier une nouvelle instance, il suffit de trouver la classe c qui maximise la probabilité $\mathbb{P}(C = c|A)$ et de classer l'instance dans cette classe.

Malheureusement, en pratique il est bien rare que les attributs soient conditionnellement indépendants entre eux, d'où le nom d'application *naïve* du théorème de Bayes. Par contre, les chercheurs en apprentissage automatique ont constaté que même l'hypothèse d'indépendance conditionnelle naïvement faite, l'algorithme donnait de bons résultats. Pour ces raisons et pour ses racines dans la théorie mathématique, l'algorithme bayésien naïf est aujourd'hui très populaire, particulièrement dans le domaine de la reconnaissance du langage naturel [**WF05**].

5.2. Intégration de la théorie bayésienne dans l'arbre de décision

L'intégration d'un algorithme bayésien naïf dans les feuilles d'un arbre de décision poursuit un but en trois points :

- effectuer des classifications basées sur plusieurs variables à la fois ;
- prendre en compte les coûts de test et de classification ;
- décrire un modèle dynamique qui permet de minimiser les coûts de test.

Pour y arriver, voici comment l'algorithme d'arbre de décision est modifié :

- Tout d'abord, il faut spécifier les coûts de test (c'est-à-dire le coût d'acquisition des valeurs d'un attribut pour un enregistrement) de chaque attribut, en plus des coûts de mauvaise classification des instances.
- Une fois les coûts établis, un algorithme d'arbre de décision est employé sur les données. Cet arbre de décision fonctionne sur le même principe que le C4.5 à deux différences près :

  (1) La fonction de séparation des noeuds est une fonction de minimisation des coûts de mauvaise classification et de test ; cette fonction de coût est décrite plus loin.

  (2) Il n'y a pas d'élagage effectué sur l'arbre.

– De façon parallèle et indépendante de l'arbre de décision, un classificateur bayésien est créé avec l'ensemble des instances disponibles dans l'ensemble d'apprentissage.

À la fin, l'algorithme renvoie deux modèles : un arbre de décision et un classificateur bayésien.

Pour classer une nouvelle instance, l'arbre de décision est utilisé pour déterminer quels tests une nouvelle instance devrait passer afin d'être classée. Chaque test présent le long du chemin de la classification par l'arbre de décision devient un attribut pour lequel on accepte de payer le coût de test. C'est donc un processus dynamique : on fait un premier test, observe le résultat et détermine ensuite quel est le prochain test selon l'arbre de décision, jusqu'à ce que l'on parvienne à une feuille.

Une fois tous les tests d'un certain chemin effectués, l'instance peut être classée selon le modèle bayésien, en ne tenant compte que des attributs qui ont des valeurs (c'est-à-dire ceux pour lesquels un test a déjà été fait).

Bref, l'arbre de décision renseigne sur les tests (ou les attributs) qui sont utiles à la classification et le classificateur bayésien se charge de classer les instances en fonction des données disponibles.

La fonction qui sert de critère de segmentation est en fait l'espérance du coût de mauvaise classification. Soient :

$C(VP)$	le coût d'une classification positive alors qu'elle est positive ;
$C(FP)$	le coût d'une classification positive alors qu'elle est négative ;
$C(VN)$	le coût d'une classification négative alors qu'elle est négative ;
$C(FN)$	le coût d'une classification négative alors qu'elle est positive ;
$N(P)$	le nombre d'enregistrements positifs dans l'ensemble étudié ;
$N(N)$	le nombre d'enregistrements négatifs dans l'ensemble étudié.

À première vue, donner des coûts aux classifications correctes semble farfelu et est en effet souvent laissé à zéro. Par contre, il est utile de pouvoir leur donner un poids, même faible, afin de pouvoir établir des ratios entre les différentes bonnes classifications.

Le coût d'une classification positive $C_P = N(P) \cdot C(VP) + N(N) \cdot C(FP)$ et le coût d'une classification négative est $C_N = N(N) \cdot C(VN) + N(P) \cdot C(FN)$. Ce coût de classification sert de critère de segmentation dans chacun des noeuds et est appliqué sur les instances de chaque noeud qu'il faut séparer.

Puisqu'il faut minimiser le coût total, la probabilité de classer un élément comme étant positif est donnée par

$$\mathbb{P}(+) = 1 - \frac{C_P}{C_P + C_N} = \frac{C_N}{C_P + C_N}.$$

Ensuite, on cherche à déterminer le coût moyen d'une classification positive. Pour ce faire, on calcule l'espérance du coût de classer un élément comme étant positif :

$$E_P = \frac{C_N C_P}{C_P + C_N}.$$

L'espérance E_N des coûts d'une classification négative est obtenue de façon réciproque.

Afin de pouvoir effectuer des comparaisons, on calcule l'espérance E du coût de classification si l'ensemble (ou le noeud) courant n'est pas séparé. Cette espérance est définie comme étant

$$E = E_P + E_N = \frac{2C_P C_N}{C_P + C_N}$$

tandis que l'espérance E_A du coût si l'ensemble à l'étude est séparé selon l'attribut A possédant n valeurs est obtenu avec

$$E_A = 2\sum_{i=1}^{n} \frac{C_{P_i} C_{N_i}}{C_{P_i} + C_{N_i}}.$$

En calculant la valeur de E_A pour tous les attributs de la base de données, il est facile de pouvoir trouver l'attribut de séparation qui minimise le coût de classification. C'est de cette façon que la fonction de coût de classification remplace le critère de segmentation de l'arbre de décision.

Donc, contrairement à l'algorithme C4.5 qui calcule le gain en information, *DTNB* calcule la réduction R du coût de mauvaise classification qui est donnée par

$$R = E - E_A - T_A$$

où T_A est le coût du test à effectuer en lien avec l'attribut A.

Pour la classification d'un nouvel enregistrement, il faut passer d'abord par l'arbre de décision. Le nouvel enregistrement est introduit dans l'arbre. À chaque noeud, si l'attribut de séparation est manquant, un test est effectué. Une fois arrivé dans une feuille, l'instance (munie de ses nouvelles valeurs pour certains attributs qui étaient manquants) est soumise à la classification par le modèle bayésien. Celui-ci ignore les données manquantes et classe l'instance dans le but de minimiser le coût de mauvaise classification. L'algorithme classifie un enregistrement e comme étant positif si

$$C(FN) \cdot \mathbb{P}(+|e) < C(FP) \cdot \mathbb{P}(-|e)$$

ou négatif si l'expression est fausse.

Au niveau de la réduction des coûts, il a été montré par les auteurs que le *DTNB* est plus performant qu'un arbre de décision ou un modèle bayésien qui prennent en considération les coûts de mauvaise classification.

L'arbre utilise également l'ensemble des informations à sa disposition afin de prendre une décision quant à la classification d'un nouvel enregistrement, le but étant justement de minimiser les coûts de test pour obtenir les informations.

5.3. Et avec des données réelles...

Après avoir vu la théorie de ce nouvel hybride, il est intéressant d'observer comment il agit en pratique. La table 2 présente quelques instances sur lesquelles est appliqué le nouvel algorithme. Ensuite, il faut aussi établir des coûts de test à chacun des quatre attributs comme établis à la table 3. Finalement, il faut donner une matrice de coût de mauvaise classification telle celle présentée à la table 4. Les données présentées dans ces tableaux sont inspirées d'une batterie de tests qui pourraient être utilisée pour déterminer si oui (1) ou non (0) une personne est atteinte d'une certaine maladie, chaque test ayant un certain coût et donnant un résultat positif (1) ou négatif (0). Par conséquent, il est plus grave

de classer une personne malade comme étant saine que l'inverse. Ceci se reflète dans la matrice de coût de classification.

Instance#	A_1	A_2	A_3	A_4	Variable dépendante
1	0	0	1	0	1
2	0	1	0	0	0
3	1	1	1	0	0
4	0	1	1	0	0
5	1	1	0	1	0
6	1	1	1	0	1
7	0	0	1	0	1
8	0	1	1	1	0
9	1	1	0	0	1
10	1	1	0	1	1

TABLE 2. Base de données utilisée pour l'application de l'algorithme DTNB.

A_1	A_2	A_3	A_4
1	5	3	2

TABLE 3. Couts de test.

		Valeurs prédites	
		1	0
Valeurs réelles	1	$C(VP) = 0$	$C(FN) = 10$
	0	$C(FP) = 3$	$C(VN) = 0$

TABLE 4. Couts de classification.

Pour effectuer la classification selon l'algorithme du DTNB, il faut d'abord créer un arbre de décision en utilisant la fonction de réduction de coût de mauvaise classification $R = E - E_A - T_A$ comme critère de segmentation. Afin de simplifier les choses, il faut plutôt minimiser

$$E_A + T_A = 2\left(\frac{C_{P_0}C_{N_0}}{C_{P_0} + C_{N_0}} + \frac{C_{P_1}C_{N_1}}{C_{P_1} + C_{N_1}}\right) + T_A.$$

Par exemple, le calcul d'une première séparation par l'attribut A_1 donnerait un coût total d'environ $23,4138$. En effet,

$$C_{P_0} = N(P) \cdot C(VP) + N(N) \cdot C(FP) = 2 \cdot 0 + 3 \cdot 3 = 9$$

et

$$C_{N_0} = N(N) \cdot C(VN) + N(P) \cdot C(FN) = 3 \cdot 0 + 2 \cdot 10 = 20.$$

De la même façon, il est possible de trouver $C_{P_1} = 6$ et $C_{N_1} = 30$. Avec ces valeurs, il suit

$$E_{A_1} = 2\frac{C_{P_0}C_{N_0}}{C_{P_0} + C_{N_0}} + 2\frac{C_{P_1}C_{N_1}}{C_{P_1} + C_{N_1}} = 2\frac{9 \cdot 20}{9 + 20} + 2\frac{6 \cdot 30}{6 + 30} = 22,4138.$$

Sachant que le coût de test de A_1 est de 1, il suit que $E_{A_1} + T_{A_1} = 23,4138$. Suivant le même principe, il est possible de trouver $E_{A_2} - T_{A_2} = 25$, $E_{A_3} - T_{A_3} = 26.0769$ et $E_{A_4} - T_{A_4} = 24,1939$.

Suivant notre algorithme, notre première segmentation se fait selon l'attribut A_1 qui minimise $E_A - T_A$. La suite de notre arbre se construit selon un processus similaire et se termine quand $R < 0$, c'est-à-dire que le coût augmente si une nouvelle segmentation est créée ou s'il n'y a plus d'attribut pour faire une nouvelle segmentation.

En parallèle à la création de l'arbre, il faut créer un classificateur bayésien. Pour ce faire, il faut déterminer la probabilité qu'une instance soit positive sachant la valeur de ses attributs. En d'autres termes, on cherche :

$$\mathbb{P}(1|A) \approx \mathbb{P}(1)\prod_{i=1}^{n}\mathbb{P}(A_i|1) \text{ et } \mathbb{P}(0|A) \approx \mathbb{P}(0)\prod_{i=1}^{n}\mathbb{P}(A_i|0).$$

Par exemple, selon les données de la table 2 on peut trouver que la probabilité à priori qu'une valeur soit positive est de 5/10. Si on classe une instance avec les valeurs $A_1 = 1, A_2 = 1, A_3 = 0, A_4 = 1$, il suit

$$\begin{aligned}\mathbb{P}(1|A_i) &\approx \mathbb{P}(1) \cdot \mathbb{P}(A_1 = 1|1) \cdot \mathbb{P}(A_2 = 1|1) \cdot \mathbb{P}(A_3 = 0|1) \cdot \mathbb{P}(A_4 = 1|1) \\ &= \frac{1}{2} \cdot \frac{3}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} = 0,0144.\end{aligned}$$

Par le même processus, on trouve que $\mathbb{P}(0|A_i) \approx 0,032$. La probabilité que l'instance soit classée négative (0) étant supérieure à la probabilité que l'instance soit classée positive (1) amène le classificateur à assigner la classe négative à l'instance.

Or, l'algorithme hybride fait en sorte que l'ensemble des valeurs des attributs n'est pas disponible, tout dépendant de l'arbre de décision. Heureusement, le classificateur bayésien supporte très bien les valeurs manquantes : s'il y a des valeurs qui ne sont pas disponibles, il faut seulement ignorer les attributs correspondants.

Par exemple, si l'on classe une instance ayant comme valeur $A_1 = 1$, $A_2 = ?$, $A_3 = 0$, $A_4 = 1$, on obtient

$$\begin{aligned}\mathbb{P}(1|A_i) &\approx \mathbb{P}(1) \cdot \mathbb{P}(A_1 = 1|1) \cdot \mathbb{P}(A_3 = 0|1) \cdot \mathbb{P}(A_4 = 1|1) \\ &= \frac{1}{2} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} = 0,024 \text{ et} \\ \mathbb{P}(0|A_i) &= 0,032.\end{aligned}$$

Encore une fois, cette instance serait classée négative. Cette robustesse aux valeurs manquantes fait du classificateur bayésien un algorithme de choix pour cet hybride.

6. Conclusion

Dans cet article, il a été question de méthode d'hybridation des techniques d'apprentissage en forage de données, plus particulièrement des arbres de décision. Par exemple, un arbre de décision peut être combiné avec un algorithme bayésien naïf afin de prendre en compte la gestion des coûts de tests et de mauvaise classification.

Toutes les techniques présentées à la section 4 peuvent être utiles en situation pratique de forage de données et devraient être connues du foreur. Bien sûr, les techniques mentionnées peuvent très souvent être utilisées conjointement avec d'autres algorithmes que les arbres de décision. Notamment, beaucoup de recherches ont été faites sur les hybrides incluant un réseau de neurones.

D'une part, beaucoup de ces stratégies pourraient être appliquées de façon systématique comme le tri des attributs (section 4.2) ou les techniques d'ensembles (section 4.1).

D'autre part, il serait intéressant de faire ressortir les situations où certaines techniques sont à privilégier afin de bâtir un algorithme qui pourrait lui-même décider de ses stratégies d'hybridation. De cette façon, il serait possible d'établir des systèmes de forage de données qui, selon la nature des données, seraient en mesure de faire une batterie de tests complète et d'utiliser les informations de plusieurs de ces examens pour la construction d'un modèle efficace. Cette façon de faire pourrait unifier les cas spéciaux et transformer le forage de données en un outil utilisé de façon plus éclairée par les propriétaires de bases de données importantes.

Les informations enfouies dans les bases de données valent souvent beaucoup plus que les données qui les représentent. Il ne reste plus qu'à les forer pour en extraire toute la richesse.

Références

[AH98] S. Anand and J. Hughes. Hybrid data mining systems : The next generation. 1394 :13–24, 1998.

[AT07] T. Arentze and H. Timmermans. Parametric action decision trees : Incorporating continuous attribute variables into rule-based models of discrete choice. *Transportation Research Part B : Methodological*, 41(7) :772–783, 2007.

[BAM07] A. Bouyer, B. Arasteh, and A. Movaghar. A new Hybrid Model using Case-Based Reasoning and Decision Tree Methods for improving Speedup and Accuracy. In *Iadis International Conference Applied Computing*, pages 978–972, 2007.

[BC09] I. Bose and X. Chen. Hybrid models using unsupervised clustering for prediction of customer churn. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 1, 2009.

[BFOS84] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. Classification and regression trees. *Wadsworth, Belmont*, 1984.

[BHV+95] J. Bala, J. Huang, H. Vafaie, K. DeJong, and H. Wechsler. Hybrid learning using genetic algorithms and decision trees for pattern classification. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 719–724, 1995.

[Bre96] L. Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

[CAP05] M. Chong, A. Abraham, and M. Paprzycki. Traffic accident analysis using machine learning paradigms. *Special Issue : Computational Intelligence in Data mining Guest Editors : Janos Abonyi*, 29 :89, 2005.

[CF00] D.R. Carvalho and A.A. Freitas. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in data mining. In *Proc. Genetic and Evolutionary Computation Conf (GECCO-2000)*, pages 1061–1068, 2000.

[CF04] D.R. Carvalho and A.A. Freitas. A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences*, 163(1-3) :13–35, 2004.

[FS95] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[FSBA09] S. Fong, Y.W. Si, and R.P. Biuk-Aghai. Applying a hybrid model of neural network and decision tree classifier for predicting university admission. In *Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on*, pages 1–5. IEEE, 2009.

[FWI+98] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I.H. Witten. Using model trees for classification. *Machine Learning*, 32(1) :63–76, 1998.

[GK95] S. Goonatilake and S. Khebbal. *Intelligent hybrid systems*. John Wiley & Sons, 1995.

[Hew09] N.M. Hewahi. A Hybrid Architecture for a Decision Making System. *Journal of Artificial Intelligence*, 2(2) :73–80, 2009.

[HK06] J. Han and M. Kamber. *Data Mining : Concepts and Techniques*. Morgan Kaufmann, deuxième edition, 2006.

[KB03] Y. Kornienko and A. Borisov. Investigation of a hybrid algorithm for decision tree generation. In *Intelligent Data Acquisition and Advanced Computing Systems : Technology and Applications, 2003. Proceedings of the Second IEEE International Workshop on*, pages 63–68. IEEE, 2003.

[KG10] M.A. Kumar and M. Gopal. A hybrid SVM based decision tree. *Pattern Recognition*, 43(12) :3977–3987, 2010.

[Koh96] R. Kohavi. Scaling up the accuracy of naive bayes classifiers : a decision-tree hybrid. *KDD-96 Proceedings*, 1996.

[KR10] M. Kissi and M. Ramdani. A hybrid decision trees-adaptive neuro-fuzzy inference system in prediction of anti-HIV molecules. *Expert Systems With Applications*, 2010.

[LHF03] N. Landwehr, M. Hall, and E. Frank. Logistic model trees. *Machine Learning : ECML 2003*, pages 241–252, 2003.

[NMS10] B.B. Nair, VP Mohandas, and NR Sakthivel. A Decision Tree-Rough Set Hybrid System for Stock Market Trend Prediction. *International Journal of Computer Applications*, 6(9) :1–6, 2010.

[Nou] H.F.T. Noumbi. Feature Selection & Hybrid Decision Tree/Genetic Algorithm for Cancer Diagnosis based on Mass Spectrometry Proteomics.

[PAGT05] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas. Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*, 2005.

[PN03] H. Pistori and J.J. Neto. Decision tree induction using adaptive FSA. *CLEI Electron. J*, 6(1), 2003.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[Qui93] J.R. Quinlan. *C4. 5 : programs for machine learning*. Morgan Kaufmann, 1993.

[RN94] S.J. Russell and P. Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, 1994.

[RSM] N. Remeikis, I. Skučas, and V. Melninkaitè. Hybrid Machine Learning Approach for Text Categorization. *International Journal of Computational Intelligence*, 1(1).

[SCWH05] G. Stein, B. Chen, A.S. Wu, and K.A. Hua. Decision tree classifier for network intrusion detection with GA-based feature selection. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 136–141. ACM, 2005.

[SL05] S. Sheng and C. Ling. Hybrid cost-sensitive decision tree. *Knowledge Discovery in Databases*, 2005.

[SPW01] A. Seewald, J. Petrak, and G. Widmer. Hybrid decision tree learners with alternative leaf classifiers : An empirical study. *audiology*, 2001.

[Tur95] P. Turney. Cost-sensitive classification : Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research (JAIR)*, 2, 1995.

[TW09] CF Tsai and SP Wang. Stock price forecasting by hybrid machine learning techniques. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 1 :755–60, 2009.

[WF05] I. Witten and E. Frank. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, deuxième edition, 2005.

[WZX10] L. Wang, X. Zang, and P. Xu. Induction of a novel hybrid decision forest model based on information theory. *Journal of Software*, 5(11) :1195–1199, November 2010.

[ZC02] Z.H. Zhou and Z.Q. Chen. Hybrid decision tree. *Knowledge-based systems*, 15(8) :515–528, 2002.

[ZČGM04] M. Zorman, B. Černohorski, G. Goršek, and O. Milan. Hybrid evolutionary built decision trees for prediction of perspective cross-country skiers. In *Proceedings of the 4th WSEAS International Conference on Applied Informatics and Communications*, pages 1–6. World Scientific and Engineering Academy and Society (WSEAS), 2004.

ADAM SALVAIL-BÉRARD, DÉPARTEMENT D'INFORMATIQUE, UNIVERSITÉ DE SHERBROOKE
Courriel: Adam.Salvail-Berard@USherbrooke.ca