Table des matières

Table des figures

Liste des tableaux

La seule façon de renforcer notre intelligence est de n'avoir d'idées arrêtées sur rien, de laisser l'esprit accueillir toutes les pensées.

John Keats

La Science explique ce qui se passe tout le temps autour de nous. La religion aussi, mais la science marche mieux, parce qu'elle trouve des excuses plus crédibles quand elle se trompe.

Terry Pratchett

Chapitre 1

Introduction

La récente explosion des masses de données a créé un besoin toujours grandissant en analyse de données, mais, bien souvent, cette analyse est conduite par des experts de terrain et de différents domaines ayant peu d'expérience en science des données. Nous nous intéresserons donc à ce besoin d'*assistance* à l'analyse de données, qui commence tout juste à recevoir une certaine attention des communautés scientifiques, donnant naissance au domaine de la méta-analyse (Serban et al. 2013). Ce domaine relativement nouveau présente encore de nombreux verrous. Un des plus fondamentaux concerne les définitions de l'analyse de données, qui, dans la littérature, sont nombreuses et divergent selon les méthodes qu'elles englobent. De plus, l'effervescence de nouvelles approches d'analyse nécessite une adaptation constante, au risque d'une obsolescence rapide des méta-analyses incapables de prendre en compte les innovations du domaine. Enfin, l'interaction avec des utilisateurs d'autres disciplines se révèle souvent très complexe, nécessitant un important travail d'adaptation de la part de l'utilisateur pour comprendre les concepts d'analyse manipulés, souvent au détriment de la réalité de terrain qui l'intéresse.

Les premières approches d'assistance à l'analyse se révélant ainsi souvent similaires et peu abouties (aucune à notre connaissance n'a atteint un réel déploiement), nous tenterons en particulier d'étudier de nouvelles approches de méta-analyse pour adresser ce problème d'assistance à l'analyse de données. Notre objectif, issu d'un cheminement décrit au chapitre 2, peut se résumer comme suit :

> Proposer de nouvelles approches performantes de méta-analyse à des fins d'assistance à l'analyse de données adaptée à des utilisateurs non-experts.

En particulier, cela consistera à prendre avantage de la connaissance que l'on peut avoir du domaine pour recommander des processus d'analyse adaptés au contexte de l'utilisateur, comme illustré en Figure 1.1. Cette connaissance se présentant couramment sous

la forme d'une base d'expériences passées (partie inférieure de la Figure 1.1), il sera nécessaire de pouvoir évaluer la pertinence de processus d'analyses employés par le passé dans de nouveaux contextes (① en Figure 1.1). Au-delà de la recommandation du processus d'analyse, nombre d'utilisateurs devront être accompagnés depuis la formulation de leur besoin jusqu'à l'interprétation des résultats d'analyse. Il sera donc crucial que ces recommandation d'analyses par méta-analyse puissent s'inscrire dans un processus d'assistance global, capable d'accompagner l'utilisateur tout au long de son cheminement.

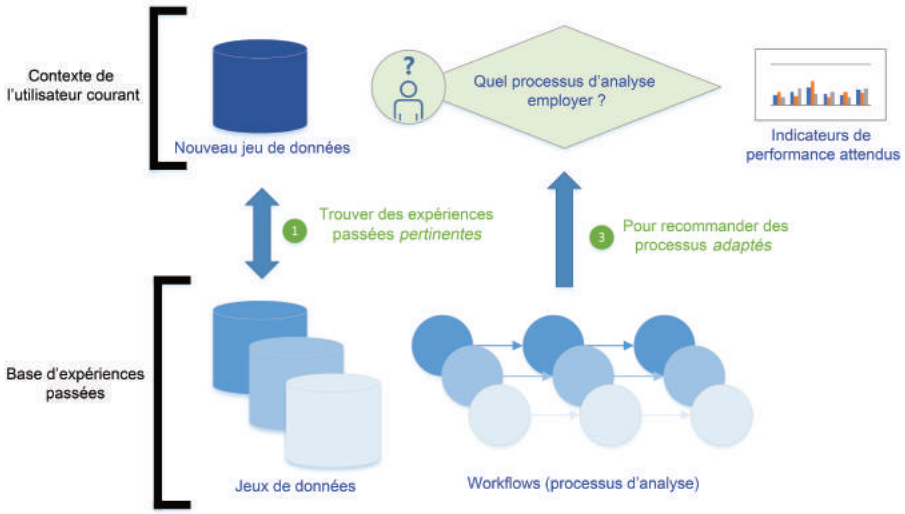

FIGURE 1.1 – Principe général de méta-analyse

Afin d'atteindre notre objectif, une première étape cruciale sera de déterminer ce qu'est une méta-analyse *performante*. Pour évaluer une analyse au méta-niveau, nous aurons besoin d'une procédure d'évaluation adaptée, aucun standard n'ayant encore été établi dans ce domaine relativement neuf. L'objet du chapitre 3 sera donc d'établir un **cadre d'évaluation** permettant la comparaison d'approches de méta-analyse potentiellement très différentes.

Ensuite, afin de pouvoir identifier les expériences pertinentes dans le contexte de l'utilisateur, nous nous intéresserons à un verrou majeur de la méta-analyse : la *caractérisation de jeu de données*. Il s'agit de plus d'un facteur de performance critique de la méta-analyse, dont les techniques les plus couramment employées conduisent à la perte d'une importante quantité d'information. Nous étudierons ainsi au chapitre 4 les propriétés désirables d'une méthode de caractérisation pour proposer une **dissimilarité entre jeux de données** faisant usage de toute l'information disponible et autorisant de nouvelles approches de méta-analyse reposant sur le principe exposé en Figure 1.1.

L'utilisation de cette caractérisation par dissimilarité permet alors de recommander des

processus d'analyse de données complets. Nous décrirons donc dans le chapitre 5 la forme des nouvelles approches de méta-analyses ainsi rendues possibles, ainsi que les processus afférents d'**assistance à l'analyse de données**. Nous nous intéresserons également à la définition du besoin de l'utilisateur en terme d'analyse, afin de pouvoir y répondre au mieux.

Enfin, le chapitre 6 présentera un bilan des contributions et travaux réalisés, suivi des nombreuses perspectives de recherche ouvertes. Le domaine étudié étant large et en grande partie inexploré, on pourra noter que chaque verrou levé apporte de nouvelles perspectives, chaque contribution soulève davantage de questions que de réponses. Ce sera l'un des obstacles majeur à notre progression, mais également ce qui fait l'intérêt du domaine. Aujourd'hui, toute la connaissance du domaine ne suffit pas à mesurer l'ampleur de la tâche, mais s'ouvrent les portes d'un monde de perspectives et d'innovations dont l'impact sociétal pourrait se révéler déterminant.

La connaissance scientifique possède en quelque sorte des propriétés fractales : nous aurons beau accroître notre savoir, le reste - si infime soit-il - sera toujours aussi infiniment complexe que l'ensemble de départ.

Isaac Asimov

Il existe deux sortes de progrès scientifique : l'expérimentation et la catégorisation méthodique qui repoussent progressivement les limites de la connaissance, et le jaillissement révolutionnaire du génie qui redéfinit et transcende ces limites.
Reconnaissant notre dette envers les premières, nous nous languissons néanmoins de ce dernier.

Prokhor Zakharov

Chapitre 2

État de l'art

De tous les champs de l'intelligence artificielle, l'apprentissage automatique a connu un essor des plus importants. De nombreuses approches aux fondements variés ont donné naissance à une profusion de méthodes et d'algorithmes, tandis que les théories sous-jacentes se sont étoffées rapidement. Il s'agit là d'un sujet très vaste, qui fait encore régulièrement l'objet de revues et critiques tentant de donner une vision d'ensemble des plus récentes avancées (Kotsiantis et al. 2007 ; Mitchell 1997). Nous nous attacherons dans un premier temps à donner un aperçu des principales approches d'apprentissage automatique depuis la perspective de leur applicabilité, c'est-à-dire des circonstances leur permettant d'afficher de bonnes performances. Ce point particulier a fait l'objet de nombreuses études et comparaisons empiriques que nous présenterons ensuite, mettant en évidence l'absence d'apprentissage "optimal". Ceci nous amènera à nous intéresser à la caractérisation et à l'apprentissage des zones de compétence des différents algorithmes d'apprentissage. Cette nouvelle perspective, consistant à sélectionner un algorithme adapté au problème à traiter, pose les fondations du méta-apprentissage, dont les multiples problématiques ont pour point commun une recherche d'adéquation entre la donnée à apprendre et l'algorithme à utiliser. Nombre de verrous du méta-apprentissage constituent encore des perspectives de recherche ouvertes, l'un des principaux consistant en la caractérisation des jeux de données. Il existe en effet une forte dépendance entre l'efficacité du méta-apprentissage et la caractérisation des jeux de données étudiés. Nous étudierons donc les méthodes permettant d'améliorer cette caractérisation, telles la sélection d'attributs au méta-niveau. Ceci implique l'utilisation de techniques classiques de sélection d'attributs. Serait-il alors possible de les intégrer au processus de méta-apprentissage ? Mais alors pourquoi s'arrêter à la sélection d'attributs ? Nombre de techniques d'analyse de données, telles les pré-traitements de la donnée ou l'optimisation des hyperparamètres d'apprentissage, sont connues pour

améliorer sensiblement les résultats de l'apprentissage. Nous verrons donc ensuite comment ces diverses techniques se sont intégrées à la boucle de méta-apprentissage pour donner naissance à cette nouvelle problématique de méta-analyse. Nous nous placerons enfin dans la continuité de cette problématique pour présenter le plan de cette thèse, et structurer la réponse envisagée à notre objectif initial.

2.1 Apprentissage automatique

Nous nous attacherons simplement ici à donner un aperçu aussi large que possible des différentes approches de l'apprentissage automatique. Leur objectif commun est la modélisation d'un *concept* (une *classe* en classification, une valeur en régression, etc...) à partir d'exemples (les *instances*) de ce dernier. Nombre de ces approches auront connu des développements majeurs depuis leurs premières parutions, mais nous discuterons en particulier les idées originales ayant donné lieu aux dites approches, nous intéressant à leurs avantages et limites, et laissant aux revues spécialisées le soin d'entrer dans les détails des développements individuels. La Table 2.1 en page 26 présente un récapitulatif de cette section. En présentant ces avantages et limites, on cherche en particulier à transmettre l'intuition du *biais* des algorithmes, c'est à dire de l'adéquation intrinsèque de leur principe de fonctionnement à certains problèmes, au détriment d'autres. On s'intéressera cependant également à des critères ne caractérisant pas leur *performance* au sens strict, comme la lisibilité des résultats (le concept est-il modélisé de façon intelligible pour un humain ?), ou la durée d'apprentissage.

Parmi les premières approches de l'apprentissage automatique, on trouve les arbres de décision (Safavian et Landgrebe 1991). Ces derniers découlent de méthodes préexistantes dans des domaines tels que les statistiques et le traitement du signal. Une des méthodes les plus populaires, toujours largement utilisée, est l'algorithme C4.5 (Quinlan 2014) dont le principe fondateur est la recherche de partitions les plus "pures" possibles, mais nombre d'autres travaux sur les arbres de décision sont par exemple recensés par Murthy (1998). Les systèmes à règles de décision (Fürnkranz 1999) constituent une approche assez semblable, présentant une définition compréhensible de la classe étudiée, mais ne pouvant souvent traiter différentes classes qu'isolément. Ces systèmes sont conceptuellement très proches des arbres de décision, et y sont souvent identifiables (Quinlan 1987). L'atout principal des arbres et règles de décision est leur compréhensibilité. Il est aisé de comprendre le raisonnement ayant amené à attribuer telle classe à telle instance, mais leur efficacité dans

le traitement de problèmes complexes s'est révélée faible par rapport à d'autres approches. Des techniques plus récentes réemploient les capacités des arbres de décision pour générer des modèles plus complexes. Par exemple, Frank et al. (1998) ajoutent des régressions linéaires aux feuilles d'un arbre de décision, améliorant sa performance tout en conservant une certaine interprétabilité.

Une autre approche est basée sur le concept de perceptron introduit par Rosenblatt (1962). Le perceptron permet un apprentissage en ligne, capable de produire une réponse indépendamment de l'état de son exécution, mais ne peut apprendre que des concepts linéairement séparables. *WINNOW* (Littlestone et Warmuth 1994) est un bon exemple d'algorithme construit sur un perceptron. Ce dernier exploite les propriétés du perceptron pour s'adapter rapidement aux dérives du concept cible, ce qui s'avère idéal dans des situations d'apprentissage de concept instable, telle la demande électrique d'une ville, variant dans le temps.

Découlant du perceptron, on retrouve l'une des approches les plus prolixes : les réseaux de neurones, popularisés par leurs nombreuses applications réelles. Leur force réside en leur capacité à caractériser les instances de manière efficace, épargnant ce problème à l'utilisateur, tout d'abord par rétropropagation des erreurs (Rumelhart et al. 1988), puis en particulier avec le récent essor de l'apprentissage profond, ou *Deep Learning* (Hinton et al. 2006 ; Schmidhuber 2015). Ceci s'est révélé particulièrement utile pour des instances difficiles à caractériser par un ensemble d'attributs simples (comme des images ou des sons), permettant d'extraires automatiquement les propriétés essentielles caractérisant le concept. Les réseaux de neurones se caractérisent en revanche souvent par une variance élevée (grande sensibilité à l'ensemble d'apprentissage) et une opacité des résultats les rendant difficiles à accepter pour certaines applications réelles. Zhang (2000) proposent une intéressante vision d'ensemble des développements des réseaux neuronaux.

L'apprentissage paresseux (Aha 2013), popularisé par l'algorithme des k plus proches voisins, ou *k-NN* (Cover et Hart 1967), reporte tout calcul à la phase de classification, présentant donc naturellement une durée d'apprentissage nettement moindre. Parmi les faiblesses de *k-NN*, on peut par exemple compter un espace d'exécution à croissance combinatoire, ou une importante dépendance des performances à la métrique utilisée pour évaluer les "*distances*" instances. Par exemple, K^* Cleary, Trigg et al. (1995) utilisent une distance information-théorétique basée sur la notion d'entropie qui lui permet d'éviter toute distinction entre attributs numériques et nominaux, améliorant sa performance sur les jeux de données employant des attributs des deux types.

Dans certains cas, on ne cherche pas à apprendre de concept particulier. On parle alors d'apprentissage non supervisé, pour lequel on pourra par exemple faire appel à des méthodes de partitionnement de données (*clustering*) (Jain et al. 1999), ou à des recherches de motifs fréquents (Brin et al. 1997), pour apprendre des structures intéressantes de la donnée. Ces approches ont révélé tout leur intérêt dans le contexte de l'exploration et de la compréhension de la donnée, et ce dans de nombreux domaines.

Si l'on cherche bien à apprendre un concept donné, mais que pour différentes raisons, on ne peut le connaitre que sur un nombre très limité d'instances (par exemple un test en laboratoire très couteux), on se trouve dans une situation d'apprentissage *semi-supervisé* (Chapelle et al. 2009 ; Zhu 2010). On y retrouve des techniques capables de prendre en compte la structure sous-jacente de la donnée (*manifold*) dans l'apprentissage du concept cible.

Basées sur "l'essai-et-erreur", les approches d'apprentissage par renforcement (Sutton et Barto 1998) se concentrent sur les situations d'interaction, que l'on apprend à résoudre dans l'optique de maximiser un objectif à plus long terme. Elles introduisent le dilemme entre apprentissage et exploration, se rapprochant de diverses méthodes heuristiques, car chaque étape présente un choix entre l'affinement d'une solution connue et la recherche potentiellement aveugle d'une nouvelle solution.

L'emploi d'heuristiques évolutionnaires pour l'apprentissage a quant à lui démontré une efficacité particulière dans le traitement d'importantes masses de données (Bacardit et Llorà 2013) grâce au parallélisme intrinsèque qui caractérise ces méthodes.

Ensuite, des approches basées sur la construction d'un modèle de probabilité, on retiendra en particulier les réseaux bayésiens. Introduits par Nilsson (1965), les réseaux bayésiens naïfs sont soumis à l'hypothèse d'indépendance des attributs. Mais bien que cette dernière soit en général fausse, les classifieurs bayésiens naïfs font montre de performances remarquables sur de nombreux problèmes. Des réseaux bayésiens plus avancés permettent de prendre en compte la connaissance préalable dont on peut disposer (Jensen 1996), mais restent sensibles à la surabondance d'attributs.

Construites à l'aide de la théorie de l'apprentissage statistique, et faisant suite au concept de *VC-dimension* qualifiant la complexité de problèmes d'apprentissage, sont ensuite introduites les machines à vecteurs de support, ou *SVM* (Burges 1998 ; Vapnik 2013). Construits sur de solides fondations théoriques, ces algorithmes ont fait la preuve d'excellentes performances, et apportent la garantie d'un optimum global dans leur marge de séparation des classes. Ces méthodes sont en revanche fondamentalement binaires, ren-

dant complexe le traitement de problèmes à nombreuses classes. Diverses techniques telles la "*Sequential minimal optimization*" (Platt et al. 1998) ou la régression par vecteurs de supports (Smola et Schölkopf 2004) s'attachent à améliorer le processus d'apprentissage des *SVM* pour mieux bénéficier de leurs garanties théoriques.

Enfin, la course à l'amélioration des performances a donné naissance à diverses méthodes composites, à commencer par la *Stacked Generalization* (D. Wolpert 1992), qui exploite les prédictions d'un ensemble de classifieurs. Les méthodes de *Bagging* (Breiman 1996) quant à elles se concentrent sur des perturbations de la donnée, permettant d'améliorer significativement la performance d'algorithmes trop sensibles au bruit. Le *Boosting*, illustré par le célèbre *Adaboost* (Freund et al. 1999), permet de réduire simultanément biais et variance par combinaison pondérée de différents classifieurs. Nombre de techniques de *Boosting* sont encore produites régulièrement, telle *XgBoost* (Chen et Guestrin 2016), système de *Boosting* d'arbres de décision affichant des performances remarquables. Le célèbre algorithme des *Random forests* (Breiman 2001) a démontré d'excellentes performances en combinant des arbres de décision construits sur des sous-ensembles aléatoires de la donnée, rapidement suivi par Dong et al. (2005) qui s'assure de la qualité des arbres utilisés. Ces différentes approches à multiples classifieurs présentent en général de meilleures performances que leurs pendants individuels (Ho et al. 1994), résultat souvent attribué à la diversité des biais des algorithmes combinés (Kuncheva et Whitaker 2003), mais sans fondation théorique forte. Cette incapacité à formaliser et expliquer la performance de ces approches peut représenter un obstacle théorique à leur évolution, mais leurs performances remarquables dans de nombreux contextes les placent souvent au premier plan de la recherche internationale.

On dispose ainsi de nombre de méthodes variées et performantes, mais dans de nombreuses applications, comme dans le contexte particulier de la biologie et de la santé, leur usage reste limité. En dépit de méthodologies pas-à-pas conçues pour guider l'utilisation d'algorithmes d'apprentissage (Palaniappan et Awang 2008), ou de vues d'ensemble vulgarisées présentant les avantages et inconvénients de diverses approches (Obenshain 2004), le choix et la calibration d'un algorithme pour résoudre un problème concret reste une tâche particulièrement ardue pour un non-expert, car reposant largement sur les expérimentations et l'expertise de l'utilisateur.

Approches	Limites	Références
Arbres et règles de décision	Instabilité et performances variables.	Frank et al. 1998 ; Fürnkranz 1999 ; Murthy 1998 ; Quinlan 1987, 2014 ; Safavian et Landgrebe 1991
Réseaux de neurones	Interprétation très complexe.	E. Frank 2014 ; Hinton et al. 2006 ; Littlestone et Warmuth 1994 ; Rosenblatt 1962 ; Rumelhart et al. 1988 ; Schmidhuber 2015 ; Zhang 2000
Apprentissage paresseux	Sensibilité aux attributs non pertinents.	Aha 2013 ; Cleary, Trigg et al. 1995
Apprentissage non supervisé	Forte dépendance de la méthode à l'objectif, faible pouvoir de généralisation.	Brin et al. 1997 ; Jain et al. 1999
Semi-supervisé	Risque de dégradation des performances par rapport au cas simplement supervisé.	Chapelle et al. 2009 ; Zhu 2010
Renforcement	Coût et complexité importants.	Sutton et Barto 1998
Heuristique évolutionnaire	Comportement stochastique.	Bacardit et Llorà 2013 ; Parthiban et Subramanian 2007
Réseaux bayésiens	Hypothèses en général non respectées et non testables.	Jensen 1996 ; Nilsson 1965
SVM	Classification fondamentalement binaire.	Burges 1998 ; Platt et al. 1998 ; Smola et Schölkopf 2004 ; Vapnik 2013
Ensembles	Grande complexité et difficulté d'interprétation.	Breiman 1996, 2001 ; Chen et Guestrin 2016 ; Dong et al. 2005 ; Freund et al. 1999 ; Ho et al. 1994 ; Kuncheva et Whitaker 2003 ; D. Wolpert 1992

TABLE 2.1 – Récapitulatif des approches d'apprentissage automatique

2.2 Évaluation de l'apprentissage

Devant la prolifération des techniques d'apprentissage, le besoin de pouvoir les évaluer et les comparer a conduit au développement de nombreux critères et méthodologies adaptés. Un des premiers besoins à ce niveau repose dans l'estimation de la fiabilité des prédictions d'un modèle appris. En effet, l'utilisation en situation réelle des techniques d'apprentissage nécessite une certaine confiance de la part de l'utilisateur final, qui doit parfois prendre des risques ou engager sa responsabilité sur la base des prédictions fournies. Les travaux de Strumbelj et al. (2010) et Strumbelj et Kononenko (2008) par exemple se sont attachés à faciliter ce rapport à l'utilisateur final en estimant la fiabilité des prédictions et en permettant d'expliquer les facteurs principaux y ayant conduit.

Une suite naturelle à l'estimation de fiabilité des prédictions est l'évaluation des modèles construits. Cette problématique reste intimement liée au développement des techniques d'apprentissage. En effet, la façon dont on juge la qualité des modèles va influencer la direction des recherches visant à proposer de nouvelles approches de construction. Divers types de critères d'évaluation ont été proposés par Kononenko (2001), incluant par exemple la transparence et compréhensibilité du modèle, ou la robustesse à la donnée bruitée ou incomplète, mais les critères de performance, de précision du modèle formé, sont restés prédominants. En classification, la précision, proportion d'instances bien classées par le modèle, est rapidement devenu un standard, malgré l'existence de résultats (Provost et al. 1998) mettant en doute son bien-fondé. Une recommandation alternative est l'utilisation du critère d'**aire sous la courbe de ROC** *(Receiver Operating Characteristic)* (Hanley et McNeil 1982), qui estime la qualité du modèle produit sur l'ensemble des conditions opérationnelles dans lesquelles il pourrait être utilisé. Construit spécifiquement pour répondre à l'insuffisance du critère de précision, l'**information score** de Kononenko et Bratko (1991) évalue par des méthodes information-théorétique la quantité d'information relative au concept produite par le modèle. De nombreux critères agrégés ou plus complexes ont vu le jour au fil des ans, comme le **Kappa** de Cohen (1968) ou les métriques multi-critères de Nakhaeizadeh et Schnabl (1997), conduisant à une confusion faisant écho à la multitude des techniques d'apprentissage (voir récapitulatif en Table 2.2 page 28).

De même, la procédure d'estimation de ces divers critères a fait l'objet de débats mettant en concurrence plusieurs approches. Une constante reste la nécessité de séparer les données utilisées pour le calcul du critère d'évaluation (*testing data*) de celles ayant

servi à construire le modèle (*training data*). Une dominance de la **validation croisée**, qui consiste à répéter les tests sur différents partitionnement *train/test* de la donnée, a néanmoins émergé dans les usages courants, validée par des résultats expérimentaux faisant état du faible biais qu'elle induit (Bailey et Elkan 1993 ; Kohavi et al. 1995). Il est donc commun d'évaluer nos critères de performance sur un ensemble de modèles construits en partitionnant la donnée disponible.

L'étude de ces critères sur des ensembles d'expériences nécessite une généralisation des processus de comparaisons, dont Aha (1992) proposent une première version. Des travaux plus récents (Vanschoren et al. 2012, 2013) proposent des bases d'expériences d'apprentissage évaluées de manière uniforme, fournissant un matériau précieux pour des comparaisons orientées vers une question de recherche particulière. Diverses méthodologies ont vu le jour pour assurer la consistance et la validité de telles comparaisons (Pizarro et al. 2002). Par exemple, Demšar (2006) présentent une méthodologie de test d'hypothèse statistique appliqué à la comparaison d'algorithmes d'apprentissage permettant de tirer des conclusions valides quant à la significativité des résultats obtenus. De même, Benavoli et al. (2015) présentent une procédure Bayésienne de comparaison permettant également de valider un résultat négatif.

Approches	Détails	Références
Précision	Proportion d'instances bien classées.	Provost et al. 1998
Aire sous la courbe de ROC	Estime la qualité du modèle produit sur l'espace complet des conditions opérationnelles.	Hanley et McNeil 1982
Information score	Estime la quantité d'information relative au concept produite par le modèle.	Kononenko et Bratko 1991
Kappa	Mesure l'accord entre les prédictions et la réalité.	Cohen 1968
Fiabilité des prédictions	Estime le risque d'erreur de prédictions individuelles.	Strumbelj et al. 2010 ; Strumbelj et Kononenko 2008
Transparence du modèle	Qualifie la compréhensibilité du modèle produit.	Kononenko 2001

TABLE 2.2 – Aperçu de quelques techniques d'évaluation

Ces critères et méthodologies ont rendu possible la comparaison directe d'algorithmes, donnant lieu à nombre d'études comparatives (Shavlik et al. 1991 ; Zheng et al. 2001) mettant en perspective les performances relatives de divers algorithmes sur des ensembles de jeux de données. Une des plus impactantes prit place dans le cadre du projet STATLOG

(Michie et al. 1994), qui en comparant une vingtaine des principaux algorithmes de l'état de l'art sur 12 jeux de données conclut en une forte dépendance entre les performances relatives des algorithmes et les jeux de données.

> Ce résultat met en valeur le cadre théorique général des **No Free Lunch Theorems** (Schaffer 1994 ; D. H. Wolpert 1996 ; Wolpert et Macready 1997) stipulant qu'aucun algorithme d'apprentissage ne peut présenter de performance optimale sur l'ensemble des problèmes d'apprentissage.

En particulier, toute performance positive dans un cas particulier implique une performance négative dans un autre. Les implications sont nombreuses, mais mettent avant tout en doute le bien-fondé de la comparaison directe d'algorithmes. En effet, quel sens y a-t-il à les comparer alors que l'absence d'optimum est établi ? La réponse passe par la caractérisation des zones d'expertise des algorithmes. Tout algorithme d'apprentissage va présenter un profil de performances propre sur l'espace des jeux de données possibles. Les régions de cet espace où il présente une performance *compétitive* va ainsi constituer sa zone d'expertise. La recherche de bonnes performances en apprentissage passe alors par **la recherche d'un algorithme adapté au problème étudié**, ce qui implique de pouvoir *caractériser* ces zones d'expertise. Ces tentatives de **caractérisation de l'adéquation donnée-algorithme** sont l'ébauche du **méta-apprentissage**.

2.3 Méta-apprentissage

> Le méta-apprentissage consiste donc en la recherche d'adéquation entre le concept à apprendre et l'algorithme utilisé pour ce faire.

De nombreuses approches à ce problème ont été proposées, engendrant bien sûr synthèses et tentatives de catégorisation (P. Brazdil et al. 2008 ; Giraud-Carrier et al. 2004 ; Vilalta et Drissi 2002). Nous les reprendrons ici pour un rapide panel des méthodes les plus emblématiques, avant d'en discuter les limites. Un récapitulatif est également présenté en Table 2.3 (page 31).

Une des premières approches s'est intéressée à la caractérisation du biais des algorithmes d'apprentissage. Le biais d'un algorithme représente ainsi les hypothèses qu'il émet quant au concept à apprendre (Gordon et Desjardins 1995). Ceci est par exemple assez intuitif sur une machine à vecteurs de support (SVM) qui va rechercher une séparation *linéaire* correspondant au mieux à la donnée : on comprend bien qu'il sera alors

plus difficile d'apprendre un concept non linéaire. Ces observations ont mené au développement d'approches de type système expert telles celles proposée par Brodley (1995), dont l'heuristique emploie des règles simples pour déterminer le biais adapté à un jeu de données, ou par Kalousis et Hilario (2000) qui introduisent une caractérisation précise du comportement des algorithmes. Tout se complique lorsque l'on considère des algorithmes plus complexes : le biais de méthodes ensemblistes sera par exemple plus difficile à exprimer. Dans une autre direction, Ganti et al. (1999) proposent d'identifier les algorithmes ayant des biais similaires en comparant les modèles qu'ils génèrent : des algorithmes faisant souvent les mêmes erreurs peuvent être considérés similaires. Une dernière possibilité serait de construire automatiquement des règles de sélection d'algorithme par des méthodes d'apprentissage de règles traditionnelles (P. Brazdil et al. 1994).

> C'est cette caractérisation de l'apprentissage par l'apprentissage qui a introduit le terme *méta-apprentissage*.

Une première variation a été introduite par Gama et Brazdil (1995), où l'on s'intéresse à caractériser les performances d'un algorithme au lieu de son biais intrinsèque. L'objectif est alors d'utiliser des techniques classiques d'apprentissage pour construire un modèle de la performance d'un algorithme sur des jeux de données. Cette approche a été largement utilisée et présente des résultats convaincants. Par exemple, dans le cadre du projet *Esprit Metal*, Köpf et al. (2000) utilisent des techniques de régression au méta-niveau pour prédire la performance des algorithmes disponibles et ainsi choisir le plus prometteur.

Une autre approche consiste à exploiter la connaissance d'expériences passées. Soares et Brazdil (2000) et P. B. Brazdil et al. (2003) proposent ainsi une méthode simple, dite de *"Zooming & Ranking"* : pour choisir quel algorithme appliquer à un nouveau jeu de données, il suffit de comparer les performances qu'ont présentées nos algorithmes sur des jeux de données semblables par le passé. On peut ainsi établir un classement des algorithmes par performance attendue sur le nouveau jeu de données, et donc potentiellement n'exécuter que les quelques meilleurs. Ceci présente de nombreux avantages, dont la simplicité d'interprétation et la prise en compte *online* de nouvelles informations. En revanche, ces méthodes reposent sur le choix des jeux de données "similaires" qui, comme nous le verrons plus loin, peut s'avérer complexe.

Toujours dans le but de sélectionner un algorithme performant pour un jeu de données, apparaissent ensuite des techniques plus élaborées. Kalousis et Hilario (2001a) ou Sun et Pfahringer (2013) par exemple, considèrent toutes les paires possibles d'algorithmes candidats, et construisent pour chaque paire un modèle de dominance capable de prédire

pour un nouveau jeu de données quel algorithme de la paire sera le plus performant. On obtient alors des performances intéressantes, mais la mise à jour des modèles pour prendre en compte de nouvelles données nécessite un réapprentissage intégral ! Leite et al. (2012) prennent un chemin différent et s'autorisent un certain nombre de tests dans sa recherche d'algorithmes performants. L'approche proposée consiste alors à choisir quels algorithmes tester pour trouver un bon candidat en effectuant un minimum de tests. On exploite alors des modèles de dominance entre algorithmes pour ne tester que les plus susceptibles de surpasser les actuels meilleurs candidats.

Approche	Références
Adaptation de biais	P. Brazdil et al. 1994 ; Brodley 1995 ; Ganti et al. 1999 ; Gordon et Desjardins 1995 ; Kalousis et Hilario 2000
Modélisation des performances	Gama et Brazdil 1995 ; Köpf et al. 2000
Exploitation de cas	P. B. Brazdil et al. 2003 ; Soares et Brazdil 2000
Comparaisons par paires	Kalousis et Hilario 2001a ; Sun et Pfahringer 2013

TABLE 2.3 – Récapitulatif des approches de méta-apprentissage

Ces nombreuses techniques de méta-apprentissage ont ainsi permis une amélioration régulière des performances des expériences d'apprentissage effectuées, mais divers obstacles viennent retarder cette amélioration. Un des premiers obstacles auquel se sont heurtées nombre de tentatives précoces de méta-apprentissage est le coût de la méta-donnée (Gama et Brazdil 1995). En effet, la donnée d'une expérience de méta-apprentissage, ou méta-donnée, décrit le domaine de l'apprentissage. En pratique, cela consiste en un ensemble d'expériences d'apprentissage effectuées. Un **metadataset** est une collection d'instances, chacune décrivant l'application d'un algorithme d'apprentissage à un jeu de données et y qualifiant sa performance. Ainsi, la construction d'un *metadataset* requiert un grand nombre d'expériences d'apprentissage, ce qui, au vu de la prolifération des algorithmes d'apprentissage disponibles, peut rapidement s'avérer très couteux. Une majorité d'expériences de méta-apprentissage ont ainsi utilisé des *metadatasets* de quelques centaines d'instances (Kalousis 2002), et malgré les récentes opportunités fournies par des initiatives telles qu'*OpenML* (Vanschoren et al. 2013), cet ordre de grandeur tarde à augmenter, limitant la significativité des résultats obtenus.

D'autre part, les premières études approfondies du domaine (Kalousis 2002 ; Kalousis et Hilario 2001a) ont rapidement mis en avant un autre écueil : le biais du méta-apprentissage.

En effet, les techniques employées au méta-niveau ne sont pas exemptes d'un biais propre, ce qui signifie que leur performance dépendra en grande partie de leur adéquation au problème de méta-apprentissage soumis. Ceci appelle alors à l'emploi d'une couche de méta-méta-apprentissage pour améliorer la performance du niveau méta, mais c'est alors du biais de cette dernière dont on dépend... Cette récursion apparemment infinie de méta^n-niveaux, conceptualisée par Vilalta et Drissi (2002), serait une adaptation complète au problème, mais en pratique aucune expérience sur ne serait-ce que le second niveau de méta n'a pu être menée. Ceci se comprend aisément si l'on garde à l'esprit qu'une expérience du n-ième méta-niveau nécessite de rassembler un nombre conséquent d'expériences du $n-1$-ième niveau, ce qui limite toute discussion en la matière à des considérations théoriques.

Enfin, l'un des derniers verrous majeurs du méta-apprentissage repose sur la caractérisation des jeux de données. En effet, pour caractériser l'adéquation entre un algorithme et des jeux de données, il paraît nécessaire de définir un ensemble de propriétés de jeux de données sur lesquelles baser nos décisions. La définition de ces propriétés s'est révélée un obstacle important dans nombre d'expériences de méta-apprentissage (P. B. Brazdil et al. 2003), et fut rapidement identifiée comme un problème récurrent du domaine (Giraud-Carrier et al. 2004). De plus, des expériences (Kalousis 2002 ; Kalousis et Hilario 2001a) ont révélé une grande dépendance de la performance du méta-niveau vis-à-vis de cette caractérisation des jeux de données. C'est donc là un élément critique à l'efficacité du méta-apprentissage, et pour lequel les perspectives sont nombreuses.

2.4 Caractérisation de jeux de données

Les premières tentatives de caractérisation de jeux de données coïncident avec les premiers comparatifs d'algorithmes d'apprentissage. Dans les années 90, le projet *STATLOG* (King et al. 1995 ; Michie et al. 1994) comparait les performances d'une vingtaine d'algorithmes d'apprentissage sur 12 jeux de données. Le résultat immédiat étant la grande sensibilité de ces performances au jeu de données, des descripteurs de jeux de données ont été proposés pour caractériser les mécaniques impliquées dans cette dépendance. Ces premiers descripteurs étaient simples : on y trouve par exemple les nombres d'instances ou d'attributs (Figure 2.1), ou des descripteurs statistiques des distributions des attributs, comme la variance moyenne (Figure 2.2) ou le kurtosis moyen des attributs numériques. Cette simple caractérisation offrait pour la première fois la possibilité d'expliciter l'expertise nécessaire à la sélection d'un algorithme approprié. La méthodologie pouvant être étendue au-delà des seuls algorithmes d'apprentissage, on retrouve par la suite de telles

problématiques de caractérisation dans d'autres domaines, telle la sélection d'instances (Leyva et al. 2015).

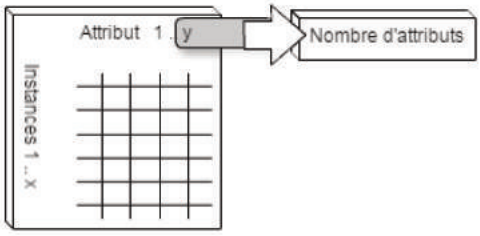

FIGURE 2.1 – Méta-attribut simple : le nombre d'attributs

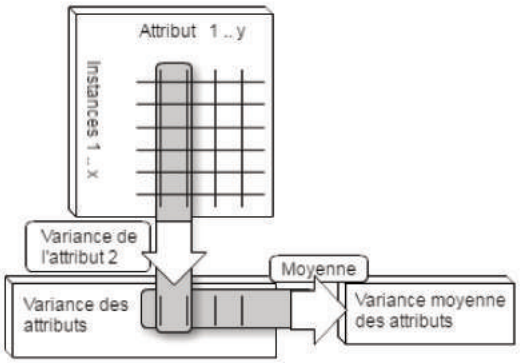

FIGURE 2.2 – Méta-attribut statistique : la variance moyenne des attributs

Le méta-apprentissage se basant sur cette caractérisation de jeux de données, nombres d'approches ont alors été proposées pour mieux caractériser les jeux de données et donc potentiellement améliorer les performances du méta-apprentissage. C'est dans ce contexte que l'on peut désigner les descripteurs formant la caractérisation comme **méta-attributs de jeux de données**. Certaines approches préconisaient le développement de méta-attributs décrivant des propriétés topologiques et géométriques "clés" des jeux de données (Ho et Basu 2002). Malgré des fondations mathématiques solides, de tels méta-attributs se révélaient souvent rédhibitoirement coûteux à calculer pour un gain de performance non justifié. D'autres propriétés de nature information-théorétique se sont en revanche révélées des méta-attributs intéressants. Castiello et al. (2005) proposent un bon aperçu de tels méta-attributs définis grâce à l'entropie de Shannon (Shannon 2001), donc décrivant par exemple l'informativité des attributs vis-à-vis de la cible. L'*information mutuelle*, ainsi présentée en Figure 2.3, mesure la réduction d'incertitude quant à la cible apportée par la connaissance d'un attribut. La moyenne de cette *information mutuelle* sur les différents attributs donne alors une mesure de l'informativité du jeu de données vis-à-vis de la cible.

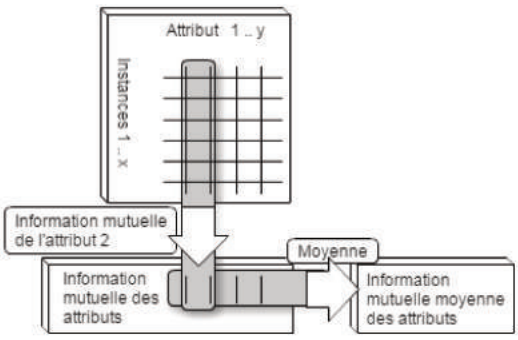

FIGURE 2.3 – Méta-attribut information-théorétique : l'information mutuelle moyenne des attributs

Le *landmarking* (Pfahringer et al. 2000) propose une approche radicalement différente au problème de caractérisation. En effet, cette approche propose de considérer la performance sur le jeu de données d'un ensemble d'algorithmes d'apprentissage très simples comme caractérisation. Pfahringer et al. (ibid.) proposent ainsi un ensemble de quelques algorithmes aux biais bien diversifiés dont les performances seront autant de méta-attributs permettant de caractériser des jeux de données (voir Figure 2.4). Le raisonnement s'inscrit dans la suite directe du méta-apprentissage, supposant que le biais d'algorithmes complexes peut être décrit par celui d'algorithmes plus simples. Fürnkranz et Petrak (2002) et Fürnkranz et Petrak (2001) notamment explorent des variantes de cette technique, proposant par exemple de limiter l'évaluation des *landmarkers* (les algorithmes simples dont on prend les performances comme méta-attributs) à des sous-parties des jeux de données. Y. Peng et al. (2002) vont plus loin et proposent l'emploi de propriétés plus complexes des modèles construits, comme la longueur des branches d'un arbre de décision construit sur le jeu de données.

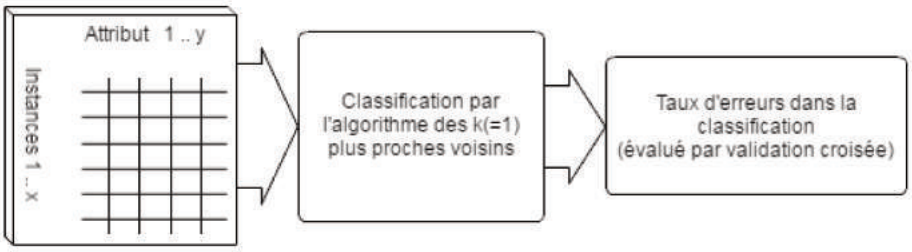

FIGURE 2.4 – Landmarker : le taux d'erreur des *1-plus-proches-voisins*

Cette prolifération de méta-attributs fit apparaitre le besoin de pouvoir les comprendre et les choisir opportunément. Pinto et al. (2014) proposent ainsi une méthodologie de manipulation de méta-attributs, permettant une définition plus stricte des concepts manipulés, mais aucun résultat significatif n'a pu établir de dominance d'une approche de caractérisation par rapport aux autres.

> Le problème du *méta* se répète : c'est l'adéquation entre l'algorithme de méta-apprentissage et les méta-attributs qui va conditionner la performance du méta-apprentissage.

La solution naturelle à ce problème passerait alors par l'emploi de techniques classiques de sélection d'attributs. On retrouve ainsi plusieurs tentatives de sélection de méta-attributs (Kalousis et Hilario 2001b ; Todorovski et al. 2000), en particulier dans le cadre du projet *METAL*, permettant d'améliorer la performance du méta-apprentissage. Un résultat demeure en revanche : aucun ensemble de méta-attributs caractérisant dominant

systématiquement les autres n'a pu être établi. La performance reste conditionnée par l'adéquation de l'algorithme de méta-apprentissage avec les méta-attributs, qui est à faire au cas par cas.

2.5 Sélection d'attributs

La sélection d'attributs est donc une problématique importante au méta-niveau (Kalousis 2002 ; Kalousis et Hilario 2001b ; Todorovski et al. 2000), mais son objectif initial était avant tout d'aider à la compréhension et à l'utilisation des données au niveau de base. Les premières approches de sélection d'attributs ont en effet accompagné le passage à l'échelle de la donnée et les débuts de l'extraction de connaissance (Liu et Motoda 2012), avant de venir compléter les techniques d'apprentissage en préparant la donnée. Pour discriminer entre attributs, de nombreux critères et algorithmes ont été proposés. *Relief* (Kira et Rendell 1992) est l'un des plus connus, se reposant sur des propriétés statistiques des attributs pour en sélectionner un sous-ensemble intéressant. Sa force réside en sa capacité à prendre en compte les interdépendances entre attributs, et des comparaisons ont établi son intérêt en situation réelle (Robnik-Šikonja et Kononenko 2003). D'autres approches, telles celle de Koller et Sahami (1996), ont proposé des critères basés sur des concepts information-théorétique, tentant d'approximer le sous-ensemble théorique optimal d'attributs.

> La pierre angulaire de la sélection d'attributs repose dans la définition des critères de **pertinence** et de **redondance**. En effet, on cherche à constituer un ensemble d'attributs apportant un maximum d'information, donc non-redondants entre eux, mais aussi informatifs que possible vis-à-vis de notre objectif.

De nombreuses approches proposent ainsi leur définition de ces critères, comme le critère *"minimal-redundancy-maximal-relevance"* de H. Peng et al. (2005), mais des travaux plus récents (Brown et al. 2012) ont subsumé ces critères comme dérivant de la probabilité conditionnelle de la classe. Le risque potentiel présenté réside dans les possibilités de généralisation. En effet, apprendre d'un petit ensemble d'attributs sélectionnés selon un critère précis peut facilement induire un phénomène de sur-apprentissage, où les résultats produits perdent toute généralité. Des combinaisons de critères (Somol et al. 2009) sont mises en avant pour répondre à ce problème, mais on a surtout pu mettre en évidence une autre propriété capitale de la sélection d'attributs : la **stabilité** (Gulgezen et al. 2009).

> Une méthode de sélection est stable si elle fournit des résultats cohérents face à des variations simples d'un jeu de données. Si une modification non significative de la donnée entraîne la sélection d'un ensemble d'attributs totalement différents, c'est que l'ensemble sélectionné n'était pas si informatif...

Des comparaisons prenant en compte à la fois l'efficacité et la stabilité des méthodes de sélection d'attributs ont pu établir la dominance de certains critères, constituant un front de Pareto de critères dominants (Brown et al. 2012). Mais là encore, le choix de l'algorithme le plus adapté dépend largement de la donnée étudiée et des algorithmes subséquemment utilisés. On retrouve alors les bases d'un problème au méta-niveau : comment choisir la sélection d'attributs adaptée ?

Approche	Références
Prise en compte des interdépendances entre attributs	Kira et Rendell 1992 ; Robnik-Šikonja et Kononenko 2003
Approximation de l'ensemble théorique optimal	Koller et Sahami 1996
Critères de pertinence et redondance	Brown et al. 2012 ; H. Peng et al. 2005 ; Somol et al. 2009

TABLE 2.4 – Récapitulatif des approches de sélection d'attributs

La sélection d'attributs s'est rapidement répandue dans de nombreux domaines (Guyon et Elisseeff 2003), apportant un avantage indéniable aux domaines présentant couramment de très nombreux attributs pour trop peu d'instances, comme la biologie ou la médecine (Huang et al. 2007). Ces secteurs d'application n'ayant pas nécessairement accès à une expertise en science des données, le choix et l'application de méthodes de sélection d'attributs se révèlent des problèmes tout aussi complexes que l'utilisation d'algorithmes d'apprentissage. Une problématique émerge alors : est il possible d'unifier au méta-niveau la sélection d'attributs et l'apprentissage ? Peut-on assister l'analyse de données jusqu'à la recommandation de processus d'analyse complets ?

2.6 Méta-analyse

On entre alors dans le domaine de la méta-analyse, ou **comment adapter au mieux le processus d'analyse de données au problème étudié**. Divers domaines se sont attaqués à l'automatisation de ce problème sous des angles très différents, donnant lieu à un panel de techniques variées. On peut ainsi trouver des applications hors contexte de techniques existantes, comme avec Misir et Sebag (2013) qui proposent une analogie

entre la sélection d'algorithmes et le filtrage collaboratif, et appliquent efficacement une technique de filtrage à ce problème fondamental de méta-analyse. Venant d'horizons bien différents, on peut trouver le système multi-agents *Pikater* (Kazik et al. 2011), capable de définir un processus complet de fouille de données adapté à la tâche en cours. Bacardit et Llorà (2013) proposent quant à eux l'emploi d'algorithmes génétiques pour l'adaptation des processus d'apprentissage, permettant à la méta-analyse *d'évoluer* avec son domaine d'application. On peut trouver chez Blockeel (2015) une intéressante proposition de langage déclaratif pour la fouille de données. Les approches déclaratives présentent en effet divers avantages qui bénéficieraient grandement à l'analyse de données, et répondraient en particulier à un besoin de clarté et d'intelligibilité de plus en plus pressant.

Ces dernières années, plusieurs domaines ont vu leur paradigme évoluer grâce aux apports de techniques de méta-analyse. On peut penser notamment au problème de satisfiabilité (SAT), pour lequel l'arrivée de portfolios d'algorithmes (Xu et al. 2012) avec de simples techniques de sélection a considérablement fait évoluer le paysage : les recherches basculent de la construction d'un algorithme optimal vers l'application de l'algorithme le mieux adapté à un problème particulier. L'optimisation voit également aparaître des techniques d'adaptation de l'algorithme au problème, telle celle proposée par Caraffini et al. (2014), qui analysent la topologie interne du problème pour construire l'algorithme le plus adapté. Des travaux plus récents (Bischl et al. 2016) proposent une standardisation du problème de sélection d'algorithmes, permettant le développement de méthodes générales et non plus limitées à un domaine d'application restreint. On peut ainsi comparer diverses approches de sélection d'algorithmes de manière unifiée sur de la donnée décrivant l'application d'algorithmes variés sur des problèmes divers.

> Un obstacle prépondérant en matière de méta-analyse réside en la *recommandation de processus d'analyse complets*. Un processus d'analyse peut se représenter sous la forme d'un graphe d'opérateurs interconnectés ayant chacun une fonction propre, allant du pré-traitement de la donnée (remplacement de valeurs manquantes, suppression d'instances aberrantes, etc...) à la construction de modèle, en passant par diverses opérations très spécialisées, comme la pondération des attributs. Le choix des opérateurs, leur paramétrisation et leur connexion sont autant de problèmes complexes, reconnus comme verrous prépondérants de l'analyse de données (Bernstein et al. 2005).

On recense ainsi divers efforts dans ce sens, commençant dans les années 90 avec les systèmes experts *Consultant* (Sleeman et al. 1995) capables de conseiller un expert domaine dans sa construction d'un processus d'analyse de données. On trouve plus récemment des

initiatives de partage communautaire d'expériences d'analyse de données, comme *OpenML* (Vanschoren et al. 2013) permettant une intégration des environnements d'analyse de données (par exemple *Rapidminer* chez Rijn et Vanschoren (2015)). Ceci permet de tirer parti des usages communs dans la définition de nouveaux processus mais requiert toujours une certaine expertise en analyse de données. Chaque "étape" du processus d'analyse de données est ainsi un problème non trivial pour lequel l'utilisateur final aura besoin d'assistance (Serban et al. 2013). Reprenons donc les principales étapes d'un processus d'analyse :

L'étape de **pré-traitement** de la donnée a souvent été reconnue comme décisive (Kotsiantis et al. 2006). En effet, l'efficacité des traitements et analyses potentiels dépend en grande partie de la *qualité* de la donnée. Se débarrasser d'information redondante, inutile ou incorrecte, normaliser les échelles de valeurs ou altérer le format intrinsèque de la donnée sont autant d'opérations critiques au succès de l'analyse, mais dont le bon emploi est souvent complexe. On peut trouver des techniques simples d'assistance au pré-traitement comme le système *Clementine* (Engels et Theusinger 1998), qui se base sur un petit ensemble de propriétés du jeu de données pour recommander les pré-traitements appropriés, mais ne tenant pas réellement compte de l'analyse à effectuer. Escalante et al. (2009), suivi par Sun (2014) et Sun et al. (2012) et Olson et al. (2016), proposent de lier le pré-traitement à l'apprentissage, pouvant par exemple recommander une chaîne de traitements plus complexe faisant précéder la classification par un nettoyage de la donnée et une sélection d'attributs. Ces approches identifient le problème de méta-analyse à une tâche d'optimisation : dans l'espace des traitements possibles, trouver celui qui conviendra le mieux à la donnée présentée. Elles se restreignent alors à un petit "espace des traitements" pour permettre aux heuristiques d'optimisation (de type *Particle Swarm* ou génétique) de le parcourir efficacement, et donc de trouver de bonnes solutions en un temps raisonnable. L'ensemble des traitements existants a en revanche tendance à croître rapidement, ce qui limite l'intérêt de méta-analyses ainsi restreintes.

La **paramétrisation des opérateurs** est une autre étape cruciale du processus d'analyse. En effet, les différents opérateurs d'un processus d'analyse, qu'il s'agisse de pré-traitements, d'opérateurs de modélisation ou d'opérations plus particulières, présentent en général un ensemble de paramètres permettant de faire varier leur comportement. Ces variations peuvent être très importantes, et le bon réglage de ces paramètres est souvent primordial au succès de l'analyse. Une réponse naturelle est alors d'employer des techniques classiques d'optimisation pour trouver une combinaison de paramètres adaptée à la situation courante. Ce type d'approche se caractérise par leur stratégie d'exploration

de l'espace des paramètres (recherche en grille, aléatoire...), avec comme problématique principale le coût parfois élevé d'évaluation d'une combinaison (Bergstra et Bengio 2012). Certaines approches, proposées par Feurer et al. (2015) ou Wistuba et al. (2015), tentent de prendre avantage des techniques existantes de méta-apprentissage pour permettre une meilleure optimisation des paramètres. Il est ainsi possible d'utiliser comme point de départ de l'optimisation une combinaison de paramètres que l'on sait s'être bien comportée par le passé sur un jeu de données similaire. Si le coût d'évaluation d'une combinaison est prohibitif, on peut se restreindre à recommander directement une combinaison de valeurs *par défaut* adaptées à la situation (Mantovani et al. 2015). Une proposition très intéressante de Thornton et al. (2013) est de combiner la sélection d'algorithmes et l'optimisation des paramètres. La technique d'optimisation bayésienne employée produit des résultats surprenants, surclassant souvent l'emploi successif de techniques de sélection d'algorithmes et d'optimisation des paramètres de l'état de l'art.

Le **post-traitement** est une étape souvent négligée du processus d'analyse. C'est pourtant l'étape la plus proche de l'utilisateur final, et celle qui sera souvent responsable de sa compréhension et bonne utilisation des résultats produits. L'emploi en situation réelle de résultats d'analyse de données peut parfois rencontrer les risques importants du domaine, par exemple pour le diagnostic médical ou la prévision de défaillance de systèmes vitaux. Dans de telles situations, l'utilisateur souhaitera pouvoir estimer la *fiabilité* d'une prédiction sur laquelle il va baser sa décision. Diverses méthodes permettent ainsi d'estimer la fiabilité des prédictions d'un algorithme d'apprentissage (Bosnić et Kononenko 2009), mais elles-mêmes présentent des performances variables dans des situations différentes. On pourra ici aussi employer des principes de méta-apprentissage pour assurer l'adéquation de la méthode d'estimation de fiabilité des prédictions à l'apprentissage effectué (Bosnić et Kononenko 2010). Cette étape n'a cependant à ce jour pas été reliée au reste du processus. Afin de faciliter la compréhension et l'acceptation des prédictions, une autre approche serait d'en expliquer le raisonnement. C'est ce que l'on peut par exemple trouver chez Strumbelj et al. (2010) qui permet de mesurer la contribution de chaque attribut à la prédiction, et ce indépendamment de l'algorithme d'apprentissage utilisé. Ceci permet de justifier la prédiction auprès de l'utilisateur, qui pourra souvent rattacher ces contributions à sa connaissance du domaine.

Le défi majeur de la méta-analyse réside aujourd'hui en la fusion des processus de méta-analyse associés à ces différentes étapes. Bernstein et al. (2005) ont proposé le terme d'*Intelligent Discovery Assistant (IDA)* pour qualifier les environnements ainsi capables d'accompagner l'utilisateur tout au long du processus d'analyse. Serban et al. (2013) reprennent ce terme pour proposer une excellente revue des *IDA* existants, étudier leurs limites, et investiguer les prochains objectifs du domaine. Plusieurs approches proposent déjà la construction complète de processus d'analyse. On peut penser par exemple au projet *e-LICO* dont Nguyen et al. (2014) présentent la méthode de planification basée sur une ontologie de la fouille de données (Hilario et al. 2011). Cette planification considère les motifs fréquents dans les analyses réalisées par le passé pour construire dynamiquement le processus d'analyse. Également basée sur une ontologie, l'approche proposée par Diamantini et al. (2009a,b) se distingue par son orientation très *IOPE (Input Output Precondition Effect)* du problème, construisant des processus en ajoutant récursivement des opérateurs compatibles avec les flux disponibles. Zakova et al. (2011) tentent de se rapprocher des domaines d'application en proposant une ontologie capable de faire le lien avec la connaissance domaine, sous forme de bases de données ou d'ontologies spécialisées. Il est alors possible de construire des processus complexes combinant et analysant les données de multiples sources. L'emploi d'ontologies s'avérant un élément récurrent, J. Kietz et al. (2009), Kietz, Serban et Bernstein (2010) et Kietz, Serban, Bernstein et Fischer (2010) proposent un environnement dédié au développement d'ontologies d'analyse de données, assurant la vérification et la cohérence lors de l'extension d'ontologies. Repris par J.-U. Kietz et al. (2014) et Serban (2013) pour proposer l'ontologie *Exposé* permettant la planification hiérarchique de tâches d'analyse, on note une volonté d'unification de ces ontologies divergentes. L'initiative *ML-shema*[1] est aujourd'hui en tête de cet important travail d'unification, qui permettrait aux diverses approches citées ici de dépasser le stade de la preuve de concept. L'unification des ontologies existantes et leur peuplement avec les très nombreux opérateurs disponibles sont cependant des tâches de grande ampleur, dont les résultats ne seront probablement pas exploitables avant plusieurs années. Il est alors légitime de se demander en parallèle si d'autres approches du problème seraient possibles ?

1. `https://github.com/ML-Schema/core`

Sous-domaine	**Approche**	**Références**
Optimisation, satisfiabilité, etc...	Portfolios d'algorithmes	Caraffini et al. 2014 ; Xu et al. 2012
Pré-traitement	Système expert	Engels et Theusinger 1998
	Sélection simultanée de l'algorithme	Escalante et al. 2009 ; Olson et al. 2016 ; Sun 2014 ; Sun et al. 2012
Sélection d'algorithme	Filtrage collaboratif	Misir et Sebag 2013
	Transfert entre applications	Bischl et al. 2016
Paramétrisation	Techniques d'optimisation	Bergstra et Bengio 2012
	Valeurs par défaut	Mantovani et al. 2015
	Méta-apprentissage	Feurer et al. 2015 ; Wistuba et al. 2015
	Sélection simultanée de l'algorithme	Thornton et al. 2013
Post-traitement	Méta-apprentissage	Bosnić et Kononenko 2010
Processus d'analyse complet	Système expert	Sleeman et al. 1995
	Système multi-agents	Kazik et al. 2011
	Planification aidée par ontologie	Hilario et al. 2011 ; J.-U. Kietz et al. 2014 ; Nguyen et al. 2014 ; Serban 2013 ; Zakova et al. 2011
	Ajout récursif d'opérateur compatible	Diamantini et al. 2009a,b

TABLE 2.5 – Récapitulatif des approches de méta-analyse

2.7 Conclusion

L'analyse de données est bien souvent conduite par des experts de terrain de domaines bien différents, ayant peu d'expérience en science des données, comme illustré par Parthiban et Subramanian (2007), Lan et al. (2012), ou Zeevi et al. (2015). Ceci implique souvent de leur part un investissement important, pour des analyses parfois triviales. Les chercheurs en biologie ou en médecine par exemple, se trouvent très dépendants des outils d'analyse qui leur sont fournis, ce qui a attiré suffisamment d'attention pour que ce problème soit étudié par d'autres communautés. L'accessibilité des techniques d'apprentissage, par exemple, a ainsi reçu une certaine attention (Kononenko 2001), et on peut trouver divers aperçus vulgarisant des approches de fouille de données (Obenshain 2004). Plus récemment, des approches collaboratives du problème ont vu le jour (Goble et al. 2010), mais se heurtent souvent à des problématiques de propriété des données et des procédés. On s'intéressera donc à ce besoin d'*assistance* à l'analyse de données, qui, toujours insatisfait, a donné naissance au domaine de la méta-analyse (Serban et al. 2013). Les premières approches du sujet étudiées en 2.6 n'ayant jamais atteint de réel déploie-

ment, nous tenterons en particulier de proposer de nouvelles approches de méta-analyse pour adresser ce problème d'assistance à l'analyse de données. Notre objectif peut alors se résumer comme suit :

> Étudier de nouvelles approches performantes de méta-analyse pour proposer une assistance à l'analyse de données adaptée à des utilisateurs non-experts.

Pour ce faire, une première étape cruciale est de déterminer ce qu'est une méta-analyse *performante*. Pour évaluer tout élément d'une analyse au méta-niveau, nous aurons besoin d'une procédure d'évaluation adaptée, aucun standard n'ayant encore été établi dans ce domaine relativement neuf. L'objet du **chapitre 3** sera donc d'établir un **cadre d'évaluation** permettant la comparaison d'approches de méta-analyse potentiellement très différentes.

Ensuite, afin d'ouvrir de nouvelles voies, nous nous intéresserons à un verrou majeur de la méta-analyse : la **caractérisation de jeux de données**. Il s'agit en effet d'un facteur de performance majeur de la méta-analyse, dont les techniques les plus couramment employées conduisent à la perte d'une importante quantité d'information (Kalousis et al. 2004). On étudie ainsi au **chapitre 4** les propriétés désirables d'une *méthode de caractérisation* pour proposer une dissimilarité entre jeux de données inspirée des approches "non-essentialistes" de Duin (2015). Nous rejoindrons alors d'autres techniques faisant usage de toute l'information disponible (Smid 2016 ; Smid et Neruda 2014) pour aboutir à de nouvelles approches de méta-analyse.

L'utilisation de cette caractérisation par dissimilarité permet de recommander facilement des processus d'analyse de données complets. Nous décrirons donc dans le **chapitre 5** la forme des nouvelles approches de méta-analyses ainsi rendues possibles, ainsi que les **processus afférents d'assistance à l'analyse de données**.

Enfin, le **chapitre 6** présentera un **bilan des contributions** et travaux réalisés, suivi des nombreuses **perspectives de recherche** ouvertes. Le domaine étudié étant large et toujours en grande partie inexploré, on pourra noter que chaque verrou levé apporte de nouvelles perspectives, chaque contribution soulève davantage de questions que de réponses. Ce sera l'un des obstacles majeurs à notre progression, mais également ce qui fait l'intérêt du domaine. Aujourd'hui, toute la connaissance du domaine ne suffit pas à mesurer l'ampleur de la tâche, mais on ouvre les portes d'un monde de perspectives et d'innovations dont l'impact sociétal pourrait se révéler déterminant.

Bibliographie du chapitre

Aha, David (1992). « Generalizing from case studies : A case study ». In : *9th International Conference on Machine Learning*, p. 1–10 (cf. p. 28).

— (2013). *Lazy learning*. Springer Science & Business Media (cf. p. 23, 26).

Bacardit, Jaume et Xavier Llorà (2013). « Large-scale data mining using genetics-based machine learning ». In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 3.1, p. 37–61 (cf. p. 24, 26, 37).

Bailey, Timothy et Charles Elkan (1993). « Estimating the accuracy of learned concepts." » In : *International Joint Conference on Artificial Intelligence*. Citeseer (cf. p. 28).

Benavoli, Alessio, Giorgio Corani, Francesca Mangili et Marco Zaffalon (2015). « A Bayesian nonparametric procedure for comparing algorithms ». In : *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, p. 1264–1272 (cf. p. 28).

Bergstra, James et Yoshua Bengio (2012). « Random search for hyper-parameter optimization ». In : *The Journal of Machine Learning Research* 13.1, p. 281–305 (cf. p. 39, 42).

Bernstein, Abraham, Foster Provost et Shawndra Hill (2005). « Toward intelligent assistance for a data mining process : An ontology-based approach for cost-sensitive classification ». In : *Knowledge and Data Engineering, IEEE Transactions on* 17.4, p. 503–518 (cf. p. 38, 41).

Bischl, Bernd et al. (2016). « Aslib : A benchmark library for algorithm selection ». In : *Artificial Intelligence* 237, p. 41–58 (cf. p. 37, 42).

Blockeel, Hendrik (2015). « Data Mining : From Procedural to Declarative Approaches ». In : *New Generation Computing* 33.2, p. 115–135 (cf. p. 37).

Bosnić, Zoran et Igor Kononenko (2009). « An Overview of Advances in Reliability Estimation of Individual Predictions in Machine Learning ». In : *Intelligent Data Analysis* 13.2, p. 385–401. issn : 1088-467X (cf. p. 39).

— (2010). « Automatic selection of reliability estimates for individual regression predictions ». In : *The Knowledge Engineering Review* 25.01, p. 27–47 (cf. p. 40, 42).

Brazdil, Pavel B, Carlos Soares et Joaquim Pinto Da Costa (2003). « Ranking learning algorithms : Using IBL and meta-learning on accuracy and time results ». In : *Machine Learning* 50.3, p. 251–277 (cf. p. 30–32).

Brazdil, Pavel, Christophe Giraud Carrier, Carlos Soares et Ricardo Vilalta (2008). *Metalearning : Applications to data mining*. Springer Science & Business Media (cf. p. 29).

Brazdil, Pavel, João Gama et Bob Henery (1994). « Characterizing the applicability of classification algorithms using meta-level learning ». In : *European conference on machine learning*. Springer, p. 83–102 (cf. p. 30, 31, 65, 96, 97).

Breiman, Leo (1996). « Bagging predictors ». In : *Machine learning* 24.2, p. 123–140 (cf. p. 25, 26).

— (2001). « Random forests ». In : *Machine learning* 45.1, p. 5–32 (cf. p. 25, 26, 112, 139).

Brin, Sergey, Rajeev Motwani, Jeffrey D Ullman et Shalom Tsur (1997). « Dynamic itemset counting and implication rules for market basket data ». In : *ACM SIGMOD Record*. T. 26. 2. ACM, p. 255–264 (cf. p. 24, 26).

Brodley, Carla E (1995). « Recursive automatic bias selection for classifier construction ». In : *Machine Learning* 20.1-2, p. 63–94 (cf. p. 30, 31).

Brown, Gavin, Adam Pocock, Ming-Jie Zhao et Mikel Luján (2012). « Conditional likelihood maximisation : a unifying framework for information theoretic feature selection ». In : *The Journal of Machine Learning Research* 13.1, p. 27–66 (cf. p. 35, 36, 126, 144, 146).

Burges, Christopher JC (1998). « A tutorial on support vector machines for pattern recognition ». In : *Data mining and knowledge discovery* 2.2, p. 121–167 (cf. p. 24, 26).

Caraffini, Fabio, Ferrante Neri et Lorenzo Picinali (2014). « An analysis on separability for memetic computing automatic design ». In : *Information Sciences* 265, p. 1–22 (cf. p. 37, 42).

Castiello, Ciro, Giovanna Castellano et Anna Maria Fanelli (2005). « Meta-data : Characterization of input features for meta-learning ». In : *Modeling decisions for artificial intelligence*. Springer, p. 457–468 (cf. p. 33).

Chapelle, Olivier, Bernhard Scholkopf et Alexander Zien (2009). « Semi-Supervised Learning ». In : *IEEE Transactions on Neural Networks* 20.3, p. 542–542 (cf. p. 24, 26).

Chen, Tianqi et Carlos Guestrin (2016). « Xgboost : A scalable tree boosting system ». In : *22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, p. 785–794 (cf. p. 25, 26).

Cleary, John G, Leonard E Trigg et al. (1995). « K* : An instance-based learner using an entropic distance measure ». In : *Proceedings of the 12th International Conference on Machine learning*. T. 5, p. 108–114 (cf. p. 23, 26, 112, 137).

Cohen, Jacob (1968). « Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. » In : *Psychological bulletin* 70.4, p. 213 (cf. p. 27, 28, 62, 100, 170).

Cover, Thomas M et Peter E Hart (1967). « Nearest neighbor pattern classification ». In : *Information Theory, IEEE Transactions on* 13.1, p. 21–27 (cf. p. 23).

Demšar, Janez (2006). « Statistical comparisons of classifiers over multiple data sets ». In : *The Journal of Machine Learning Research* 7, p. 1–30 (cf. p. 28, 68).

Diamantini, Claudia, Domenico Potena et Emanuele Storti (2009a). « Kddonto : An ontology for discovery and composition of kdd algorithms ». In : *Third Generation*

Data Mining : Towards Service-Oriented Knowledge Discovery (SoKD), p. 13–24 (cf. p. 41, 42, 159, 163).

— (2009b). « Ontology-driven KDD process composition ». In : *Advances in Intelligent Data Analysis VIII*. Springer, p. 285–296 (cf. p. 41, 42, 159, 163).

Dong, Lin, Eibe Frank et Stefan Kramer (2005). « Ensembles of balanced nested dichotomies for multi-class problems ». In : *Knowledge Discovery in Databases : PKDD 2005*. Springer, p. 84–95 (cf. p. 25, 26, 69, 95).

Duin, Robert PW (2015). « The Dissimilarity Representation for finding Universals from Particulars by an anti-essentialist Approach ». In : *Pattern Recognition Letters* 64, p. 37–43. issn : 0167-8655. doi : http://dx.doi.org/10.1016/j.patrec.2015.04.015. (Visité le 01/04/2017) (cf. p. 43).

Engels, Robert et Christiane Theusinger (1998). « Using a Data Metric for Preprocessing Advice for Data Mining Applications. » In : *ECAI*. Citeseer, p. 430–434 (cf. p. 38, 42).

Escalante, Hugo Jair, Manuel Montes et Luis Enrique Sucar (2009). « Particle swarm model selection ». In : *The Journal of Machine Learning Research* 10, p. 405–440 (cf. p. 38, 42).

Feurer, Matthias, T Springenberg et Frank Hutter (2015). « Initializing bayesian hyperparameter optimization via meta-learning ». In : *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (cf. p. 39, 42).

Frank, Eibe (2014). *Fully supervised training of Gaussian radial basis function networks in WEKA*. Rapp. tech. Department of Computer Science, The University of Waikato (cf. p. 26, 112).

Frank, Eibe, Yong Wang, Stuart Inglis, Geoffrey Holmes et Ian H Witten (1998). « Using model trees for classification ». In : *Machine Learning* 32.1, p. 63–76 (cf. p. 23, 26, 95).

Freund, Yoav, Robert Schapire et N Abe (1999). « A short introduction to boosting ». In : *Journal-Japanese Society For Artificial Intelligence* 14.771-780, p. 1612 (cf. p. 25, 26).

Fürnkranz, J et J Petrak (2002). *Extended data characteristics*. Rapp. tech. Accessed 12/11/15 at citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.302. METAL consortium (cf. p. 34, 62).

Fürnkranz, Johannes (1999). « Separate-and-conquer rule learning ». In : *Artificial Intelligence Review* 13.1, p. 3–54 (cf. p. 22, 26).

Fürnkranz, Johannes et Johann Petrak (2001). « An evaluation of landmarking variants ». In : *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, p. 57–68 (cf. p. 34).

Gama, Joao et Pavel Brazdil (1995). « Characterization of classification algorithms ». In : *Progress in Artificial Intelligence*. Springer, p. 189–200 (cf. p. 30, 31, 153).

Ganti, Venkatesh, Johannes Gehrke et Raghu Ramakrishnan (1999). « A framework for measuring changes in data characteristics ». In : *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, p. 126–137 (cf. p. 30, 31).

Giraud-Carrier, Christophe, Ricardo Vilalta et Pavel Brazdil (2004). « Introduction to the special issue on meta-learning ». In : *Machine learning* 54.3, p. 187–193 (cf. p. 29, 32, 77, 96).

Goble, Carole A et al. (2010). « myExperiment : a repository and social network for the sharing of bioinformatics workflows ». In : *Nucleic acids research* 38.suppl 2, p. 677–682 (cf. p. 42).

Gordon, Diana F et Marie Desjardins (1995). « Evaluation and selection of biases in machine learning ». In : *Machine Learning* 20.1-2, p. 5–22 (cf. p. 29, 31).

Gulgezen, Gokhan, Zehra Cataltepe et Lei Yu (2009). « Stable and accurate feature selection ». In : *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, p. 455–468 (cf. p. 36).

Guyon, Isabelle et André Elisseeff (2003). « An introduction to variable and feature selection ». In : *The Journal of Machine Learning Research* 3, p. 1157–1182 (cf. p. 36).

Hanley, James A et Barbara J McNeil (1982). « The meaning and use of the area under a receiver operating characteristic (ROC) curve. » In : *Radiology* 143.1, p. 29–36 (cf. p. 27, 28).

Hilario, Melanie, Phong Nguyen, Huyen Do, Adam Woznica et Alexandros Kalousis (2011). « Ontology-based meta-mining of knowledge discovery workflows ». In : *Meta-Learning in Computational Intelligence*. Springer, p. 273–315 (cf. p. 41, 42).

Hinton, Geoffrey E, Simon Osindero et Yee-Whye Teh (2006). « A fast learning algorithm for deep belief nets ». In : *Neural computation* 18.7, p. 1527–1554 (cf. p. 23, 26).

Ho, Tin Kam, Jonathan J Hull et Sargur N Srihari (1994). « Decision combination in multiple classifier systems ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 16.1, p. 66–75 (cf. p. 25, 26).

Ho, Tin Kamo et Mitra Basu (2002). « Complexity measures of supervised classification problems ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.3, p. 289–300 (cf. p. 33, 146).

Huang, Yue, Paul McCullagh, Norman Black et Roy Harper (2007). « Feature Selection and Classification Model Construction on Type 2 Diabetic Patients Data ». In : *Artificial Intelligence in Medicine* 41.3, p. 251–262. issn : 0933-3657. doi : 10.1016/j.artmed.2007.07.002. url : http://dx.doi.org/10.1016/j.artmed.2007.07.002 (visité le 01/04/2017) (cf. p. 36).

Jain, Anil K, M Narasimha Murty et Patrick J Flynn (1999). « Data clustering : a review ». In : *ACM computing surveys (CSUR)* 31.3, p. 264–323 (cf. p. 24, 26).

Jensen, Finn V (1996). *An introduction to Bayesian networks*. T. 210. UCL press London (cf. p. 24, 26).

Kalousis, Alexandros (2002). « Algorithm selection via meta-learning ». Thèse de doct. Universite de Geneve (cf. p. 31, 32, 35, 78, 79).

Kalousis, Alexandros, João Gama et Melanie Hilario (2004). « On data and algorithms : Understanding inductive performance ». In : *Machine Learning* 54.3, p. 275–312 (cf. p. 43, 61).

Kalousis, Alexandros et Maelanie Hilario (2000). « Building algorithm profiles for prior model selection in knowledge discovery systems ». In : *International journal of engineering intelligent systems for electrical engineering and communications* 8.2, p. 77–88 (cf. p. 30, 31).

— (2001a). « Model selection via meta-learning : a comparative study ». In : *International Journal on Artificial Intelligence Tools* 10.04, p. 525–554 (cf. p. 30–32, 55, 78, 96).

Kalousis, Alexandros et Melanie Hilario (2001b). « Feature Selection for Meta-learning ». In : *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. PAKDD. London, UK, UK : Springer-Verlag, p. 222–233. isbn : 3-540-41910-1. url : http://dl.acm.org/citation.cfm?id=646419.693650 (visité le 01/04/2017) (cf. p. 35, 61, 62).

Kazik, O., K. Peskova, M. Pilt et R. Neruda (2011). « Meta Learning in Multi-agent Systems for Data Mining ». In : *2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. T. 2, p. 433–434. doi : 10.1109/WI-IAT.2011.233. (Visité le 01/04/2017) (cf. p. 37, 42).

Kietz, J, Floarea Serban, Abraham Bernstein et Simon Fischer (2009). « Towards cooperative planning of data mining workflows ». In : *Proceedings of the Third Generation Data Mining Workshop at the 2009 European Conference on Machine Learning (ECML 2009)*, p. 1–12 (cf. p. 41).

Kietz, Jörg-Uwe, Floarea Serban et Abraham Bernstein (2010). « eProPlan : A tool to model automatic generation of data mining workflows ». In : *Proceedings of the 3rd Planning to Learn Workshop (WS9) at ECAI*. T. 2010 (cf. p. 41).

Kietz, Jörg-Uwe, Floarea Serban, Abraham Bernstein et Simon Fischer (2010). « Data mining workflow templates for intelligent discovery assistance and auto-experimentation ». In : *Third-Generation Data Mining : Towards Service-oriented Knowledge Discovery (SoKD-10)*, p. 1–12 (cf. p. 41).

Kietz, Jörg-Uwe, Floarea Serban, Simon Fischer et Abraham Bernstein (2014). « "Semantics Inside !" But lets not tell the Data Miners : Intelligent Support for Data Mining ». In : *European Semantic Web Conference ESWC 2014*. Sous la dir. de Fabien Gandon et Claudia Amato. Lecture Notes in Computer Science 8465. Springer, p. 706–720. isbn : 978-3-319-07443-6. doi : 10.1007/978-3-319-07443-6_47. (Visité le 01/04/2017) (cf. p. 41, 42).

King, Ross D., Cao Feng et Alistair Sutherland (1995). « Statlog : comparison of classification algorithms on large real-world problems ». In : *Applied Artificial Intelligence an International Journal* 9.3, p. 289–333 (cf. p. 32).

Kira, Kenji et Larry A Rendell (1992). « The feature selection problem : Traditional methods and a new algorithm ». In : *AAAI*. T. 2, p. 129–134 (cf. p. 35, 36).

Kohavi, Ron et al. (1995). « A study of cross-validation and bootstrap for accuracy estimation and model selection ». In : *Ijcai*. T. 14. 2. Stanford, CA, p. 1137–1145 (cf. p. 28).

Koller, Daphne et Mehran Sahami (1996). *Toward optimal feature selection*. Rapp. tech. Stanford InfoLab (cf. p. 35, 36).

Kononenko, Igor (2001). « Machine learning for medical diagnosis : history, state of the art and perspective ». In : *Artificial Intelligence in Medicine* 23, p. 89–109 (cf. p. 27, 28, 42).

Kononenko, Igor et Ivan Bratko (1991). « Information-Based Evaluation Criterion for Classifier's Performance ». In : *Machine Learning* 6.1, p. 67–80. issn : 0885-6125 (cf. p. 27, 28, 63, 169).

Köpf, Christian, Charles Taylor et Jörg Keller (2000). « Meta-analysis : from data characterisation for meta-learning to meta-regression ». In : *Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP*. Citeseer (cf. p. 30, 31).

Kotsiantis, S, D Kanellopoulos et PE Pintelas (2006). « Data preprocessing for supervised leaning ». In : *International Journal of Computer Science* 1.2, p. 111–117 (cf. p. 38).

Kotsiantis, S, I Zaharakis et P Pintelas (2007). *Supervised machine learning : A review of classification techniques* (cf. p. 21).

Kuncheva, Ludmila I et Christopher J Whitaker (2003). « Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy ». In : *Machine learning* 51.2, p. 181–207 (cf. p. 25, 26).

Lan, Guo-Cheng et al. (2012). « Disease Risk Prediction by Mining Personalized Health Trend Patterns : A Case Study on Diabetes ». In : *2012 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, p. 27–32. doi : 10.1109/TAAI.2012.53. (Visité le 01/04/2017) (cf. p. 42).

Leite, Rui, Pavel Brazdil et Joaquin Vanschoren (2012). « Selecting classification algorithms with active testing ». In : *Machine Learning and Data Mining in Pattern Recognition*. Springer, p. 117–131 (cf. p. 31, 55, 61, 96).

Leyva, Enrique, Adriana Gonzalez et Roxana Perez (2015). « A Set of Complexity Measures Designed for Applying Meta-Learning to Instance Selection ». In : *Knowledge and Data Engineering, IEEE Transactions on* 27.2, p. 354–367 (cf. p. 33, 61, 62).

Littlestone, Nick et Manfred K Warmuth (1994). « The weighted majority algorithm ». In : *Information and computation* 108.2, p. 212–261 (cf. p. 23, 26).

Liu, Huan et Hiroshi Motoda (2012). *Feature selection for knowledge discovery and data mining*. T. 454. Springer Science & Business Media (cf. p. 35).

Mantovani, Rafael Gomes, André LD Rossi, Joaquin Vanschoren, André Carlos Ponce de Leon Carvalho et al. (2015). « Meta-learning Recommendation of Default Hyperparameter Values for SVMs in Classification Tasks ». In : *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ; International Workshop on Meta-Learning and Algorithm Selection*. University of Porto, p. 80–92. url : http://ceur-ws.org/Vol-1455/#paper-09 (visité le 01/04/2017) (cf. p. 39, 42).

Michie, Donald, David J Spiegelhalter et Charles C Taylor (1994). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA : Ellis Horwood. isbn : 0-13-106360-X (cf. p. 29, 32, 61).

Misir, Mustafa et Michele Sebag (2013). *Algorithm selection as a collaborative filtering problem*. Rapp. tech. CNRS - Centre National de la Recherche Scientifique : UMR8623, p. 43 (cf. p. 37, 42).

Mitchell, Tom M (1997). « Machine learning ». In : *Burr Ridge, IL : McGraw Hill* 45 (cf. p. 21).

Murthy, Sreerama K (1998). « Automatic construction of decision trees from data : A multi-disciplinary survey ». In : *Data mining and knowledge discovery* 2.4, p. 345–389 (cf. p. 22, 26).

Nakhaeizadeh, Gholamreza et Alexander Schnabl (1997). « Development of Multi-Criteria Metrics for Evaluation of Data Mining Algorithms. » In : *KDD*, p. 37–42 (cf. p. 27).

Nguyen, Phong, Melanie Hilario et Alexandros Kalousis (2014). « Using meta-mining to support data mining workflow planning and optimization ». In : *Journal of Artificial Intelligence Research*, p. 605–644 (cf. p. 41, 42, 151, 159, 163).

Nilsson, Nils J (1965). *Learning machines.* (Cf. p. 24, 26).

Obenshain, Mary K (2004). « Application of data mining techniques to healthcare data ». In : *Infection Control* 25.08, p. 690–695 (cf. p. 25, 42, 153, 173).

Olson, Randal S., Nathan Bartley, Ryan J. Urbanowicz et Jason H. Moore (2016). « Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science ». In : *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. GECCO '16. Denver, Colorado, USA : ACM, p. 485–492. isbn : 978-1-4503-4206-3. doi : 10.1145/2908812.2908918. url : http://doi.acm.org/10.1145/2908812.2908918 (visité le 01/04/2017) (cf. p. 38, 42).

Palaniappan, Sellappan et Rafiah Awang (2008). « Intelligent Heart Disease Prediction System Using Data Mining Techniques ». In : *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*. AICCSA. Washington, DC, USA : IEEE Computer Society, p. 108–115. isbn : 978-1-4244-1967-8. doi : 10.1109/AICCSA.2008.4493524. url : http://dx.doi.org/10.1109/AICCSA.2008.4493524 (visité le 01/04/2017) (cf. p. 25, 173).

Parthiban, Latha et R Subramanian (2007). « Intelligent heart disease prediction system using CANFIS and genetic algorithm ». In : *International Journal of Biological and Life Sciences* 3.3, p. 157–160 (cf. p. 26, 42).

Peng, Hanchuan, Fuhui Long et Chris Ding (2005). « Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.8, p. 1226–1238 (cf. p. 35, 36, 146).

Peng, Yonghong, Peter A Flach, Pavel Brazdil et Carlos Soares (2002). « Decision tree-based data characterization for meta-learning ». In : *IDDM-2002*, p. 111 (cf. p. 34, 61, 146).

Pfahringer, Bernhard, Hilan Bensusan et Christophe Giraud-Carrier (2000). « Tell me who can learn you and i can tell you who you are : Landmarking various learning algorithms ». In : *Proceedings of the 17th international conference on machine learning*, p. 743–750 (cf. p. 34, 61, 100).

Pinto, Fábio, Carlos Soares et Joao Mendes-Moreira (2014). « A framework to decompose and develop metafeatures ». In : *Meta-Learning and Algorithm Selection Workshop at ECAI 2014* (cf. p. 34).

Pizarro, Joaquin, Elisa Guerrero et Pedro L Galindo (2002). « Multiple comparison procedures applied to model selection ». In : *Neurocomputing* 48.1, p. 155–173 (cf. p. 28).

Platt, John et al. (1998). *Sequential minimal optimization : A fast algorithm for training support vector machines*. Rapp. tech. ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING (cf. p. 25, 26).

Provost, Foster J, Tom Fawcett et Ron Kohavi (1998). « The case against accuracy estimation for comparing induction algorithms. » In : *ICML*. T. 98, p. 445–453 (cf. p. 27, 28, 56).

Quinlan, J Ross (1987). « Generating Production Rules from Decision Trees. » In : *IJCAI*. T. 87. Citeseer, p. 304–307 (cf. p. 22, 26).

— (2014). *C4. 5 : programs for machine learning*. Elsevier (cf. p. 22, 26).

Rijn, Jan N. van et Joaquin Vanschoren (2015). « Sharing RapidMiner Workflows and Experiments with OpenML ». In : *MetaSel@ PKDD/ECML*, p. 93–103. url : http://ceur-ws.org/Vol-1455/#paper-10 (visité le 01/04/2017) (cf. p. 38).

Robnik-Šikonja, Marko et Igor Kononenko (2003). « Theoretical and empirical analysis of ReliefF and RReliefF ». In : *Machine learning* 53.1-2, p. 23–69 (cf. p. 35, 36, 62).

Rosenblatt, Frank (1962). *Principles of neurodynamics*. Spartan Book (cf. p. 23, 26).

Rumelhart, David E, Geoffrey E Hinton et Ronald J Williams (1988). « Learning representations by back-propagating errors ». In : *Cognitive modeling* 5, p. 3 (cf. p. 23, 26).

Safavian, S R et D Landgrebe (1991). « A survey of decision tree classifier methodology ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 21.3, p. 660–674. issn : 0018-9472. doi : 10.1109/21.97458. (Visité le 01/04/2017) (cf. p. 22, 26).

Schaffer, Cullen (1994). « A conservation law for generalization performance ». In : *Proceedings of the 11th international conference on machine learning*, p. 259–265 (cf. p. 29).

Schmidhuber, Jürgen (2015). « Deep learning in neural networks : An overview ». In : *Neural Networks* 61, p. 85–117 (cf. p. 23, 26).

Serban, F (2013). « Toward effective support for data mining using intelligent discovery assistance ». Thèse de doct. University of Zurich (cf. p. 41, 42, 77, 159).

Serban, Floarea, Joaquin Vanschoren, Jörg-Uwe Kietz et Abraham Bernstein (2013). « A survey of intelligent assistants for data analysis ». In : *ACM Computing Surveys (CSUR)* 45.3, p. 31 (cf. p. 17, 38, 41, 42, 55, 151).

Shannon, Claude Elwood (2001). « A mathematical theory of communication ». In : *Mobile Computing and Communications Review* 5.1, p. 3–55 (cf. p. 33).

Shavlik, Jude W, Raymond J Mooney et Geoffrey G Towell (1991). « Symbolic and neural learning algorithms : An experimental comparison ». In : *Machine learning* 6.2, p. 111–143 (cf. p. 28).

Sleeman, D, Michalis Rissakis, Susan Craw, Nicolas Graner et Sunil Sharma (1995). « Consultant-2 : Pre-and post-processing of machine learning applications ». In : *International Journal of Human-Computer Studies* 43.1. doi : 10.1006/ijhc.1995.1035. url : http://dx.doi.org/10.1006/ijhc.1995.1035 (visité le 01/04/2017) (cf. p. 38, 42).

Smid, Jakub (2016). « Computational Intelligence Methods in Metalearning ». Thèse de doct. Charles University in Prague (cf. p. 43, 78, 86, 88, 99, 104, 185).

Smid, Jakub et Roman Neruda (2014). « Comparing datasets by attribute alignment ». In : *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*. IEEE, p. 56–62 (cf. p. 43, 78).

Smola, Alex J et Bernhard Schölkopf (2004). « A tutorial on support vector regression ». In : *Statistics and computing* 14.3, p. 199–222 (cf. p. 25, 26, 112).

Soares, Carlos et Pavel B Brazdil (2000). « Zoomed ranking : Selection of classification algorithms based on relevant performance information ». In : *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, p. 126–135 (cf. p. 30, 31).

Somol, Petr, Jiří Grim et Pavel Pudil (2009). « Criteria ensembles in feature selection ». In : *International Workshop on Multiple Classifier Systems*. Springer, p. 304–313 (cf. p. 36).

Strumbelj, Erik, Zoran Bosnic, Igor Kononenko, Branko Zakotnik et Cvetka Grasic ; Kuhar (2010). « Explanation and Reliability of Prediction Models : The Case of Breast Cancer Recurrence ». In : *Knowledge and Information Systems* 24.2, p. 305–324. issn : 0219-1377 (cf. p. 27, 28, 40).

Strumbelj, Erik et Igor Kononenko (2008). « Towards a Model Independent Method for Explaining Classification for Individual Instances ». In : *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*. DaWaK. Berlin, Heidelberg : Springer-Verlag, p. 273–282. isbn : 978-3-540-85835-5. doi : 10.1007/978-3-540-85836-2_26. url : http://dx.doi.org/10.1007/978-3-540-85836-2_26 (visité le 01/04/2017) (cf. p. 27, 28).

Sun, Quan (2014). « Meta-Learning and the Full Model Selection Problem ». In : *Unpublished PhD thesis, University of Waikato* (cf. p. 38, 42).

Sun, Quan et Bernhard Pfahringer (2013). « Pairwise meta-rules for better meta-learning-based algorithm ranking ». In : *Machine learning* 93.1, p. 141–161 (cf. p. 30, 31, 61, 65, 96, 146).

Sun, Quan, Bernhard Pfahringer et Michael Mayo (2012). « Full model selection in the space of data mining operators ». In : *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, p. 1503–1504 (cf. p. 38, 42, 96).

Sutton, Richard S et Andrew G Barto (1998). *Reinforcement learning : An introduction*. T. 1. 1. MIT press Cambridge (cf. p. 24, 26).

Thornton, Chris, Frank Hutter, Holger H Hoos et Kevin Leyton-Brown (2013). « Auto-WEKA : Combined selection and hyperparameter optimization of classification

algorithms ». In : *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, p. 847–855 (cf. p. 39, 42).

Todorovski, Ljupco, Pavel Brazdil et Carlos Soares (2000). « Report on the experiments with feature selection in meta-level learning ». In : *Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP : forum for practical problem presentation and prospective solutions*. Citeseer, p. 27–39 (cf. p. 35, 61, 62).

Vanschoren, Joaquin, Hendrik Blockeel, Bernhard Pfahringer et Geoffrey Holmes (2012). « Experiment databases ». In : *Machine Learning* 87.2, p. 127–158 (cf. p. 28, 90).

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl et Luis Torgo (2013). « OpenML : Networked Science in Machine Learning ». In : *SIGKDD Explorations* 15.2, p. 49–60. doi : 10.1145/2641190.2641198. url : http://doi.acm.org/10.1145/2641190.2641198 (visité le 01/04/2017) (cf. p. 28, 31, 38, 57, 60, 168).

Vapnik, Vladimir (2013). *The nature of statistical learning theory*. Springer Science & Business Media (cf. p. 24, 26).

Vilalta, Ricardo et Youssef Drissi (2002). « A Perspective View and Survey of Meta-learning ». In : *Artificial Intelligence Review* 18.2, p. 77–95. issn : 0269-2821. doi : 10.1023/A:1019956318069. url : http://dx.doi.org/10.1023/A:1019956318069 (visité le 01/04/2017) (cf. p. 29, 32, 61).

Wistuba, Martin, Nicolas Schilling et Lars Schmidt-Thieme (2015). « Learning Data Set Similarities for Hyperparameter Optimization Initializations ». In : *MetaSel@ PKDD/ECML*, p. 15–26. url : http://ceur-ws.org/Vol-1455/#paper-04 (visité le 01/04/2017) (cf. p. 39, 42, 79).

Wolpert, D (1992). « Stacked Generalization ». In : *Neural Networks*, p. 241–259 (cf. p. 25, 26).

Wolpert, David H (1996). « The lack of a priori distinctions between learning algorithms ». In : *Neural computation* 8.7, p. 1341–1390 (cf. p. 29, 95, 173, 175).

Wolpert, David H et William G Macready (1997). « No free lunch theorems for optimization ». In : *Evolutionary Computation, IEEE Transactions on* 1.1, p. 67–82 (cf. p. 29, 95).

Xu, Lin, Frank Hutter, Jonathan Shen, Holger H Hoos et Kevin Leyton-Brown (2012). « SATzilla2012 : improved algorithm selection based on cost-sensitive classification models ». In : *SAT Challenge 2012 : Solver and Benchmark Descriptions*, p. 57–58 (cf. p. 37, 42, 55).

Zakova, Monika, Petr Kremen, Filip Zelezny et Nada Lavrac (2011). « Automating knowledge discovery workflow composition through ontology-based planning ». In : *Automation Science and Engineering, IEEE Transactions on* 8.2, p. 253–264 (cf. p. 41, 42, 55, 77, 151, 159).

Zeevi, David et al. (2015). « Personalized Nutrition by Prediction of Glycemic Responses ». In : *Cell* 163.5, p. 1079–1094 (cf. p. 42, 152).

Zhang, Guoqiang Peter (2000). « Neural networks for classification : a survey ». In : *Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on* 30.4, p. 451–462 (cf. p. 23, 26).

Zheng, Zijian, Ron Kohavi et Llew Mason (2001). « Real world performance of association rule algorithms ». In : *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, p. 401–406 (cf. p. 28).

Zhu, Xiaojin (2010). « Semi-Supervised Learning ». In : *Encyclopedia of Machine Learning*. Sous la dir. de Claude Sammut et Geoffrey I. Webb. Boston, MA : Springer US, p. 892–897. isbn : 978-0-387-30164-8. doi : 10.1007/978-0-387-30164-8_749. url : http://dx.doi.org/10.1007/978-0-387-30164-8_749 (visité le 01/04/2017) (cf. p. 24, 26).

La science voudrait tout expliquer, et quand il est impossible de l'expliquer, elle déclare qu'il n'y a rien à expliquer !

Abraham Stoker

Je ne crois aux statistiques que lorsque je les ai moi-même falsifiées.

---

sir Winston Leonard Spencer Churchill

Chapitre 3

Cadre d'évaluation de méta-analyses

Ce chapitre présente un cadre d'évaluation et de comparaison adapté aux spécificités de la méta-analyse de données. Ce dernier est illustré dans une expérience de comparaison de méta-apprentissages. L'exploration des résultats par des tests d'hypothèses statistiques permet de caractériser finement la performance de plusieurs méta-apprentissages, illustrant l'intérêt de cette approche pour l'extraction de connaissances décrivant le méta-niveau.

3.1 Introduction

La méta-analyse de données désigne la recherche d'une méthode efficace ou optimale permettant d'adresser un problème d'analyse de données. Cela recouvre une grande variété de tâches, évoquées dans le chapitre 2, dont certaines ont d'ores et déjà été abondamment étudiées. Par exemple, pour le problème de satisfiabilité booléenne (SAT), différentes approches de type *portfolio* ont été développées (Xu et al. 2012), reposant sur la sélection d'un algorithme approprié à la résolution d'une instance particulière du problème. La sélection d'algorithmes a également été employée pour des problèmes d'apprentissage, type classification (Leite et al. 2012) ou régression (Kalousis et Hilario 2001a). Ces problèmes particuliers ont été étudiés isolément, mais rarement considérés comme de simples instances du problème d'analyse de données. En particulier, la recommandation de chaînes de traitements d'analyse de données a reçu un intérêt croissant ces dernière années (Serban et al. 2013 ; Zakova et al. 2011). Ce problème consiste en la construction de chaînes de traitements permettant de résoudre différents problèmes d'analyse de données.

L'émergence de ces nouvelles approches amène la question de leur évaluation et comparaison. En effet, les critères employés dans l'évaluation des méthodes dédiées à des sous-problèmes spécifiques diffèrent souvent. Par exemple, comparer la performance d'une recherche de motif et d'une régression n'est pas trivial. Afin de pouvoir évaluer et comparer les méthodes existantes et futures de méta-analyse de données, nous nous attachons à construire un cadre général basé sur un critère unifié.

D'autre part, l'analyse de données reposant toujours principalement sur l'expertise humaine, la connaissance du méta-domaine est partielle et souvent implicite. Un moyen de construire explicitement cette connaissance pourrait consolider notre compréhension du domaine et aider à orienter les recherches à venir. Nous accorderons donc une grande importance à la compréhensibilité des résultats et à la qualification de leur validité.

Nous proposons donc un cadre d'évaluation et de comparaison adapté aux spécificités de la méta-analyse de données, basé sur un critère de performance unifié. Une expérience à grande échelle a été réalisée pour en démontrer la praticabilité, et les tests statistiques employés pour l'exploration des résultats qualifient la validité des connaissances produites. Nous présenterons tout d'abord un exemple de notre cas d'étude de méta-apprentissage, puis détaillerons l'expérience de comparaison réalisée. Nous en explorerons ensuite les résultats à l'aide de tests d'hypothèse statistique, afin de comparer précisément des approches particulières, puis d'extraire des connaissances qualifiant le problème de méta-apprentissage.

3.2 Proposition d'un critère de performance

Afin de pouvoir proposer et illustrer un critère de performance de méta-analyse sur un exemple, on introduit un cas d'étude de méta-apprentissage : la sélection d'algorithmes de classification. La classification étant l'un des champs les plus actifs de l'apprentissage, il est en effet notablement plus facile d'en collecter des expériences documentées. On disposera donc d'un ensemble d'algorithmes de classification, ou *classifieurs*, parmi lesquels choisir, et d'un ensemble de jeux de données sur lesquels réaliser les expériences de classification. Pour un jeu de données particulier, l'objectif d'une expérience de sélection d'algorithmes est de choisir un classifieur maximisant un critère donné. Dans cet exemple, nous utiliserons le critère traditionnel de précision (proportion d'instances bien classées), qui, bien que simpliste (Provost et al. 1998), reste très intuitif.

Pour réaliser cette sélection, nous envisageons d'employer une technique de méta-apprentissage, ce qui nécessite de collecter un ensemble de données décrivant les per-

formances de nos algorithmes. Afin de construire ce jeu de données du méta-niveau, ou *metadataset*, il faut alors collecter deux éléments fondamentaux. La précision de nos algorithmes de classification doit tout d'abord être mesurée sur un ensemble de jeux de données (voir Table 3.1). Ces jeux de données doivent ensuite être caractérisés par un ensemble fixe de descripteurs qui seront les attributs du *metadataset*, donc *méta-attributs* (voir Table 3.2, et section 2.4 de l'état de l'art). Ces éléments sont ici extraits de la base d'OpenML (Vanschoren et al. 2013), dont la base d'expériences d'apprentissage répertorie les résultats d'exécutions de nombreux classifieurs sur des milliers de jeux de données de divers domaines.

	$classifieur_1$	$classifieur_2$	...	$classifieur_{93}$
Jeu de données $_1$	0.8	0.9	...	...
Jeu de données $_2$	0.9	0.7	...	...
...	...	...	...	...
Jeu de données $_{434}$	...	...	...	...

TABLE 3.1 – Précision des algorithmes de classification sur divers jeux de données

	$Nombre\ d'Instances$	$Nombre\ d'Attributs$	...	$Méta-Attribut_{105}$
Jeu de données $_1$	100	62	...	...
Jeu de données $_2$	5000	13	...	...
...	...	...	...	...
Jeu de données $_{434}$	360	20	...	...

TABLE 3.2 – Jeux de données caractérisés par un ensemble de méta-attributs

On peut alors combiner ces éléments pour construire le *metadataset* représenté en Table 3.3. Nous utiliserons alors un méta-apprentissage de type classification au méta-niveau, c'est-à-dire que nous emploierons des méthodes de classification pour prédire quel classifieur sera le plus performant sur un nouveau jeu de données. La classe d'une instance du *metadataset* doit donc identifier quel algorithme a présenté la meilleure performance (*i.e.* la meilleure précision) sur le jeu de données qu'elle décrit.

	$Nombre\ d'Instances$	...	$Méta-Attribut_{105}$	$Meilleur\ Classifieur$
Jeu de données $_1$	100	...	4	$classifieur_{18}$
Jeu de données $_2$	5000	...	92	$classifieur_7$
...	...	...	...	...
Jeu de données $_{434}$	360	...	13	$classifieur_{63}$

TABLE 3.3 – *Metadataset* pour classification au méta-niveau

Il faut ensuite résoudre ce problème de méta-classification. Dans cet exemple ceci s'effectue par une forme de validation croisée *"leave one out"* selon le pseudocode de l'Algorithme 1. On remarquera que les algorithmes de sélection d'attributs et de classification mentionnés sont purement illustratifs.

Algorithme 1 : Exemple d'exécution.

foreach *Instance de jeu de données* $dataset_i$ **do**
- Exclure $dataset_i$ du jeu de données de méta-classification
- Appliquer l'algorithme de sélection d'attributs ReliefF au jeu de données de méta-classification
- Apprendre un modèle en appliquant l'algorithme C4.5 au jeu de données de méta-classification réduit
- Utiliser cet arbre de classification pour prédire la classe de l'instance $dataset_i$

Pour chaque jeu de données, on dispose alors d'un label de classe prédit, identifiant quel classifieur devrait y obtenir les meilleures performances, selon le modèle construit sur les autres instances. L'objectif est ensuite de mesurer la performance de cette expérience à partir de cet ensemble de prédictions. Pour ce faire, il convient d'utiliser un critère caractérisant la performance de l'apprentissage au méta-niveau. On peut considérer que l'expérience de méta-classification (sélection d'algorithmes de classification) a un *bon* résultat si la performance de l'algorithme choisi est *élevée*. C'est cette intuition que tente de capturer le critère de performance suivant. On définit au préalable le classifieur par défaut c_{def} :

Définition 1 *On définit le classifieur par défaut* c_{def} *tel que pour tout jeu de données* $dataset_i$*, le modèle construit par* c_{def} *est la fonction constante égale à la médiane de la classe de* $dataset_i$*.*

Cela signifie par exemple que si $dataset_i$ contient 100 instances réparties en trois classes, A (20 instances), B (50 instances) et C (30 instances), une prédiction faite à l'aide de c_{def} renverra toujours B (la classe majoritaire).

Définition 2 *Soit* $\boldsymbol{p}$ *la performance du classifieur* $\boldsymbol{classifier}_j$ *sur le jeu de données* $\boldsymbol{dataset}_i$ *selon le critère choisi (ici, la précision). Soient alors* **best** *la performance du meilleur classifieur de* $\boldsymbol{classifier}_{1..m}$ *sur le jeu de données* $\boldsymbol{dataset}_i$*, et* **def** *la performance du classifieur par défaut* c_{def} *sur ce même jeu de données. On définit la performance* **perf** *d'une expérience au méta-niveau sur le jeu de données* $\boldsymbol{dataset}_i$ *:*

$$perf = 1 - \frac{|best - p|}{|best - def|}$$

Ce critère de performance atteint son maximum de 1 quand le classifieur choisi obtient la meilleure précision, et atteint 0 quand le classifieur choisi présente la même précision que le classifieur par défaut. Le cas négatif traduit une performance inférieure à celle du classifieur par défaut. Bien que simple, ce critère permet de comparer la performance d'expériences construites sur différents méta-problèmes, mais il reste nécessaire de lui fournir la valeur neutre du critère d'évaluation considéré.

En mesurant cette valeur de performance au méta-niveau sur tous les jeux de données considérés, on peut estimer la performance réelle du méta-apprentissage réalisé par l'algorithme 1. Ces valeurs de performances sont commensurables et peuvent donc être directement comparées, mais l'intérêt principal de l'approche réside dans la structure multi-dimensionnelle des expériences. Elle permet en effet d'isoler les différentes dimensions des expériences (comme le classifieur utilisé au méta-niveau ou le critère de performance de base choisi...) et d'étudier leur impact sur les performances au méta-niveau.

3.3 Dimensions des expériences au méta-niveau

Nous recensons ici les différentes dimensions pouvant varier d'une expérience à une autre, dont celles considérées dans l'exemple (Figure 3.1) ainsi que d'autre dimensions envisageables. L'objectif de cette représentation est de pouvoir isoler l'impact de ces dimensions sur les performances.

Par exemple, si l'on restreint nos résultats à la dimension de la *Méthode de résolution au méta-niveau* (dans l'exemple précédent, il s'agit de l'algorithme de méta-apprentissage utilisé) en en faisant la moyenne selon toutes les autres dimensions, on obtient une estimation de la performance des diverses *Méthodes de résolution au méta-niveau* utilisées.

Ce procédé permet de mitiger l'impact des autres dimensions sur notre résultat. De plus, comme nous le verrons en section 3.5, ce format permet l'emploi de tests d'hypothèse statistique pour qualifier la significativité des résultats produits.

3.3.1 Metadataset

Toute expérience de méta-analyse nécessite une connaissance du méta-niveau. Cette dernière peut prendre diverses formes, et devra être adaptée à la méthode de résolution employée. Dans l'exemple, on utilise une technique de méta-apprentissage, ce qui contraint à exprimer la connaissance du méta-niveau sous une forme compréhensible par un algorithme d'apprentissage : un jeu de données. Ce dernier décrivant la connaissance du méta-niveau, on le qualifie de *Metadataset*.

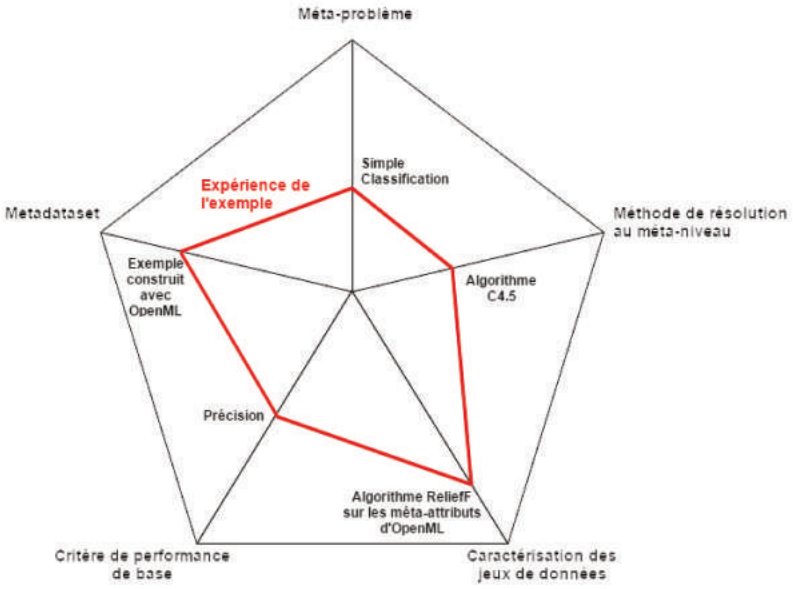

FIGURE 3.1 – Dimensions considérées dans l'exemple

Comme vu dans l'exemple et dans la section 2.3 de l'état de l'art, sa construction peut s'avérer complexe et coûteuse, car la connaissance du méta-niveau s'obtient classiquement par la collecte de nombreuses expériences du niveau de base décrivant efficacement le domaine. Ce dernier critère est souvent difficile à définir, mais dans notre cas, on peut l'interpréter par le "réalisme" des jeux de données choisis. En effet, le choix des jeux de données sur lesquels sont produites les expériences du niveau de base est un biais. On souhaiterait donc les choisir comme étant un bon échantillon représentatif des jeux de données *de la réalité*. Cet ensemble diffère de l'ensemble abstrait de tous les jeux de données *possibles*, et sa meilleure estimation intuitive serait l'ensemble de tous les jeux de données *produits à ce jour*. Les considérations sur l'essence des jeux de données *de la réalité* relèvent davantage de philosophie, mais un fait demeure : les propriétés intrinsèques caractérisant ces jeux de données *de la réalité* sont loin d'être claires. La "solution" communément acceptée à ce problème est d'utiliser un ensemble *aussi grand que possible*, ce qui devrait en effet le rapprocher de l'ensemble de tous les jeux de données *produits à ce jour*. Cependant, là encore, *"aussi grand que possible"* est vague, et peut généralement se traduire par *"au moins aussi grand que dans les études précédentes"*. Dans ce contexte, les initiatives de bases d'expériences comme OpenML (Vanschoren et al. 2013), recensant des milliers de jeux de données de diverses provenances, peuvent être considérées comme les meilleures représentations disponibles à ce jour de l'espace des jeux de données *de la réalité*.

Au delà de son coût de production, le *metadataset* est également un facteur de complexité important des expériences au méta-niveau : traiter davantage d'information se fera rarement en temps linéaire. Afin de limiter l'explosion computationnelle, on se restreint communément à un unique *metadataset* de taille raisonnable par rapport à l'état de l'art, mais il est raisonnable de supposer que deux *metadatasets*, quand bien même tous deux de bons représentants des jeux de données *de la réalité*, induiraient des résultats différents. Étudier l'impact du *metadataset* sur la performance au méta-niveau pourrait donc être justifié, malgré l'augmentation en complexité engendrée.

3.3.2 Caractérisation de jeux de données

Comme vu en section 2.4 de l'état de l'art, le problème de caractérisation de jeux de données a été adressé selon deux axes principaux :

- Le premier consiste en l'emploi de mesures statistiques et information-théorétiques pour décrire le jeu de données. Cette approche, notamment mise en avant par le projet STATLOG (Michie et al. 1994), et employée dans une majorité d'études postérieures (Kalousis et al. 2004 ; Leite et al. 2012 ; Leyva et al. 2015 ; Sun et Pfahringer 2013 ; Vilalta et Drissi 2002), présente nombre de mesures très expressives, mais sa performance repose intégralement sur l'adéquation entre le biais de l'apprentissage effectué au méta-niveau et l'ensemble de mesures choisies. On note parfois l'emploi de techniques de sélection d'attributs à ce méta-niveau (Kalousis et Hilario 2001b), mais les résultats expérimentaux ne permettent pas de conclure à la supériorité de quelconque mesure indépendamment du méta-apprentissage employé (Todorovski et al. 2000).

- Le second axe d'approche considère quant à lui non pas des propriétés intrinsèques du jeu de données étudié, mais plutôt la performance d'algorithmes d'apprentissage simples exécutés dessus. Introduit comme "*landmarking*" par (Pfahringer et al. 2000), cette approche emploie initialement le taux d'erreur d'un ensemble d'algorithmes basiques comme méta-données. Comme précédemment, les résultats suggèrent une forte dépendance de l'efficacité de cette approche avec le choix des algorithmes de base et du méta-niveau, ne révélant aucune combinaison uniformément supérieure. Des développements postérieurs ont introduit des mesures plus complexes, tel (Y. Peng et al. 2002) proposant comme méta-attributs des propriété-

tés structurelles d'un arbre de décision construit sur la donnée. Les expériences conduites par (Fürnkranz et Petrak 2002) sur ces différentes approches tendent à conclure que toutes peuvent réaliser de bonnes performances dans diverses parties de l'ensemble des jeux de données, sans qu'aucune ne domine globalement.

La base d'OpenML fournit plus d'une centaine de tels descripteurs, couvrant les différentes approches au problème de caractérisation de jeux de données. Plusieurs expériences comparatives (Kalousis et Hilario 2001b ; Todorovski et al. 2000) ont cependant montré que l'efficacité d'une caractérisation particulière dépend principalement de son adéquation avec le méta-problème et la solution employée au méta-niveau. Prenant exemple sur le succès d'approches consistant à adapter cette caractérisation à des ensembles de problèmes plus restreints (Leyva et al. 2015), une expérience intéressante serait d'adapter directement notre caractérisation au *metadataset* employé. Dans l'exemple, ceci est réalisé par l'application de l'algorithme de sélection d'attributs *ReliefF* (Robnik-Šikonja et Kononenko 2003), qui sélectionne l'ensemble des méta-attributs à utiliser pour construire le modèle prédictif au méta-niveau. *ReliefF* utilise alors des critères qui lui sont propres (voir section 2.5 de l'état de l'art) pour juger quels méta-attributs sont les plus *informatifs* dans le *metadataset* utilisé. On a ainsi adapté la caractérisation à notre *metadataset* particulier. Cette adaptation pourrait cependant être réalisée par d'autres méthodes. L'emploi d'un algorithme de sélection d'attributs différent aurait probablement choisi un autre ensemble de méta-attributs. De même, utiliser un autre ensemble initial de méta-attributs engendrerait une caractérisation différente.

Explorer cette dimension de la caractérisation des instances au méta-niveau (donc ici de jeux de données) implique donc de varier les ensembles de méta-attributs retenus, que ce soit par l'emploi d'ensembles initiaux différents ou par l'utilisation de diverses techniques de sélection d'attributs au méta-niveau.

3.3.3 Critère de performance au niveau de base

Le choix du critère de performance au niveau de base est déterminant, il spécifie l'*objectif* de l'expérience au méta-niveau. Cette dernière a en effet pour but de permettre de *meilleures* expériences au niveau de base, ce qui nécessite un critère capable de qualifier la qualité de ces expériences. Dans l'exemple, on a mesuré la qualité des expériences de classification par leur *précision* (proportion d'instances bien classées), mais d'autres critères existent. On pourrait par exemple utiliser le *Kappa* de Cohen (1968) qui me-

sure l'agrément entre prédictions et valeurs réelles en prenant en compte la possibilité de concordance contingente (prédiction tombant juste par hasard). Une classification triviale attribuant systématiquement la classe majoritaire peut parfois présenter une excellente *précision* alors qu'elle n'apporte rien au problème, certains critères comme l'*Information score* (Kononenko et Bratko 1991) mesurent donc l'*information* apportée par le modèle produit pour juger de la qualité de la classification.

Il existe ainsi de nombreux critères de performance au niveau de base permettant de juger de la qualité des expériences. Des expériences au méta-niveau utilisant différents critères se comporteront probablement de manière différente : les méthodes du niveau de base aboutissant à la meilleure *précision* ne seront probablement pas les mêmes que celles ayant le meilleur *Information score*. Décliner l'expérience au méta-niveau sur différents critères de performance au niveau de base permettrait alors d'en étudier les interactions avec les autres dimensions et de comparer leur impact sur les performances au méta-niveau (par exemple, certains critères sont peut-être notablement plus faciles à améliorer).

3.3.4 Méthode de résolution au méta-niveau et Hyperparamètres

Dans l'exemple, afin de prédire quel classifieur associer à chaque jeu de données, on a utilisé le modèle construit par une implémentation particulière de l'algorithme *C4.5*. Utiliser un autre algorithme, voire une implémentation différente de *C4.5*, produit en général un modèle différent, et donc des prédictions et une performance au méta-niveau différentes. De plus, les méthodes de résolution au méta-niveau présentent en général un ensemble d'hyperparamètres permettant d'en ajuster le fonctionnement. Dans l'exemple, ce sont les valeurs par défaut de ces paramètres qui ont été utilisées, mais pour de nombreuses méthodes, le juste réglage des hyperparamètres permet une amélioration significative des performances.

On peut ainsi considérer dans cette dimension l'ensemble des méthodes de résolution envisageables, chacune augmentée de toutes ses paramétrisations possibles. L'exploration en profondeur d'une dimension aussi vaste se révélant rapidement beaucoup trop coûteuse, des restrictions seront nécessaires. Une première possibilité serait de n'utiliser que les hyperparamètres par défaut, ce qui limite la dimension aux seules méthodes de résolution. Il s'agit en revanche d'un biais potentiellement important, car les diverses méthodes n'ont pas la même sensibilité aux hyperparamètres : certaines auront toujours besoin d'un réglage pour se montrer compétitives, tandis que d'autres peuvent apparaitre très convenables avec leur paramétrisation par défaut et bénéficieront peu d'un réglage plus attentif.

Une autre possibilité serait d'employer systématiquement une technique d'optimisation d'hyperparamètres sur les méthodes de résolution au méta-niveau. Il s'agit en général d'une heuristique explorant l'espace des hyperparamètres de la méthode pour en trouver une combinaison en maximisant les performances dans le cas présenté. On s'assurerait ainsi de fournir à chacune des méthodes des hyperparamètres lui autorisant de bonnes performances sur le *metadataset* utilisé. Ceci permettrait de mitiger le biais dû aux hyperparamètres, mais avec un coût computationnel potentiellement important. En effet, les techniques d'optimisation requièrent souvent de nombreuses exécutions de la méthode à paramétrer, ce qui démultiplierait la complexité déjà importante des expériences.

Enfin, il est possible de considérer l'optimisation des hyperparamètres comme une dimension de plus. Il existe en effet de nombreuses techniques d'optimisation aux comportements divers, qui trouveront souvent des optima différents. On peut alors les séparer de la dimension des méthodes de résolution au méta-niveau pour considérer l'optimisation des hyperparamètres comme une dimension propre. Cela implique alors d'exécuter chaque méthode de résolution au méta-niveau avec chacune des techniques d'optimisation, rajoutant ainsi un nouveau facteur de complexité à l'expérience. En revanche, on mitige ici efficacement le biais lié au choix de la méthode d'optimisation.

L'intérêt d'étudier la dimension des méthodes de résolution au méta-niveau est immédiat : on peut les comparer et rechercher les plus performantes sur le méta-problème considéré. Étudier la dimension des techniques d'optimisation d'hyperparamètres permet de même de comparer leur efficacité sur le méta-problème, ainsi que d'investiguer leurs interdépendances avec les autres dimensions. On peut en particulier supposer que certaines techniques d'optimisation seront mieux adaptées à certaines méthodes de résolution, et leur identification permettrait de gagner des perspectives intéressantes dans un domaine relativement nouveau.

3.3.5 Méta-problème

Enfin, l'élément fondamental d'une expérience au méta-niveau est le méta-problème considéré. Dans l'exemple, nous nous sommes limités au méta-problème très simple de sélection d'algorithmes de classification. Cependant, comme on a pu le voir section 2.6 de l'état de l'art, la résolution de méta-problèmes plus complexes répond à un besoin réel. On pourrait donc par exemple envisager d'étendre notre méta-problème de sélection de classifieur à d'autres approches d'apprentissage supervisé, comme la régression. De même, on peut ne pas se limiter à la sélection d'algorithmes, et prendre comme méta-problème

la recommandation de chaînes de traitements complètes d'analyse de données. On peut ainsi imaginer une hiérarchie de méta-problèmes de plus en plus généraux, nécessitant aux différents niveaux des approches différentes pour être résolus efficacement. Étudier les résultats d'expériences au méta-niveau selon cette dimension permettrait donc de mieux comprendre les relations entre différents méta-problèmes et de pouvoir estimer le *coût* d'une augmentation de généralité du méta-problème.

D'autre part, la *forme* du méta-problème reste un élément à considérer. Dans l'exemple, on a choisi de donner à notre méta-problème de sélection d'algorithmes de classification la forme d'un problème de classification. En effet, on a construit un *metadataset* adapté à de la classification et utilisé des classifieurs comme méthodes de résolution au méta-niveau. Sans altérer le méta-problème en lui-même, on aurait pu changer la *forme* de sa résolution, par exemple en utilisant des techniques de régression pour prédire la performance attendue de chaque classifieur sur les jeux de données du *metadataset* pour sélectionner le plus performant (stratégie employée par P. Brazdil et al. (1994)). La section 2.3 de l'état de l'art présente des exemples similaires de méta-apprentissages qui auraient pu donner forme à ce méta-problème, comme (Sun et Pfahringer 2013) qui apprend un modèle de dominance pour chaque paire d'algorithmes de l'ensemble de sélection. Ici encore, cette *forme* interne du méta-problème pourrait être considérée soit comme partie de la dimension du méta-problème, soit comme une nouvelle dimension indépendante, réitérant les problématiques de complexité computationelle vues dans la section précédente. On voit en revanche facilement l'intérêt d'étudier cette dimension : la comparaison directe de ces différentes *formes* de résolution et leurs relations aux méthodes de résolution employées ont très probablement un rôle prépondérant vis-à-vis de la performance des expériences au méta-niveau.

3.4 Complexité et exécution des expériences au méta-niveau

On reprend ici le méta-problème de l'exemple (sélection de classifieur par classification au méta-niveau) pour illustrer l'application de l'analyse dimensionnelle proposée.

Comme stipulé précédemment, le jeu de données de méta-classification est construit en utilisant des données d'OpenML, qui répertorie nombre de jeux de données et algorithmes de classification. La construction du jeu de données de méta-classification requérant de nombreux algorithmes évalués sur un ensemble de jeux de données, nous avons employé une technique de recherche de bi-clique maximale (Uno et al. 2004) pour trouver les plus grands ensembles de jeux de données et de classifieurs tels que chaque élément des deux

ensembles a été évalué en conjonction avec tous les éléments de l'autre ensemble. Ceci nous a permis de nous restreindre aux 93 algorithmes de classification et 434 jeux de données vu précédemment (listes en annexe, Tables A.1 et A.2). Nous avons ensuite extrait les valeurs d'évaluation de 11 critères de performance (voir Table 3.4) sur chacune des 40.000 exécutions que cela représente. Enfin nous avons extrait les valeurs des 105 méta-attributs OpenML (Table A.3) pour chaque jeu de données, et construit le *metadataset* correspondant.

area_under_roc_curve	The area under the ROC curve (AUROC), calculated using the Mann-Whitney U-test.
predictive_accuracy	The Predictive Accuracy is the percentage of instances that are classified correctly.
precision	Precision is defined as the number of true positive predictions, divided by the sum of the number of true positives and false positives.
recall	Recall is defined as the number of true positive predictions, divided by the sum of the number of true positives and false negatives.
kappa	Cohen's kappa coefficient is a statistical measure of agreement for qualitative (categorical) items : it measures the agreement of prediction with the true class.
f_measure	The F-Measure is the harmonic mean of precision and recall, also known as the the traditional F-measure, balanced F-score, or F1-score.
root_mean_squared_error	The Root Mean Squared Error (RMSE) measures how close the model's predictions are to the actual target values. It is the square root of the Mean Squared Error (MSE), the sum of the squared differences between the predicted value and the actual value.
root_relative_squared_error	The Root Relative Squared Error (RRSE) is the Root Mean Squared Error (RMSE) divided by the Root Mean Prior Squared Error (RMPSE).
mean_absolute_error	The mean absolute error (MAE) measures how close the model's predictions are to the actual target values. It is the sum of the absolute value of the difference of each instance prediction and the actual value.
relative_absolute_error	The Relative Absolute Error (RAE) is the mean absolute error (MAE) divided by the mean prior absolute error (MPAE).
kb_relative_information_score	The Kononenko and Bratko Information score, divided by the prior entropy of the class distribution.

TABLE 3.4 – Critères de performance de base utilisés

Pour itérer l'expérience, il est ensuite nécessaire de définir des ensembles d'algorithmes de classification et de sélection d'attributs à utiliser au méta-niveau. Nous avons choisi d'utiliser les algorithmes de l'API de *Weka* (Hall et al. 2009), qui de par son statut de référence, bénéficie d'implémentations de nombreux algorithmes de l'état de l'art. Nous

évaluons ainsi plus de 2600 classifieurs et 60 méthodes de sélection d'attributs de Weka, comprenant en particulier des méthodes d'augmentation et combinaison, telles le *boosting* ou *bagging*, ou des variations significatives d'un même algorithme (listes en annexe, Tables A.4 et A.5). On peut voir en Figure 3.2 la taille des dimensions ainsi générées, le nombre d'expériences au méta-niveau à effectuer étant alors le produit des tailles de ces dimensions.

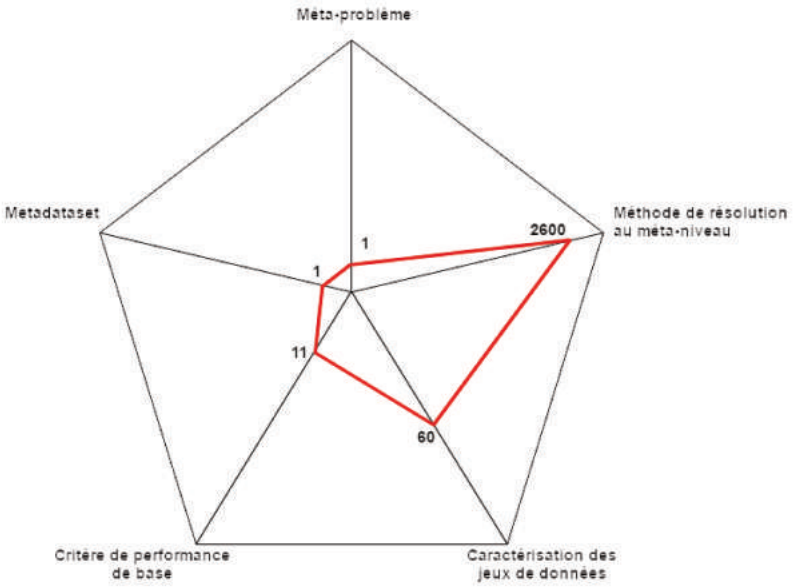

FIGURE 3.2 – Taille des dimensions d'expérimentation

Les exécutions individuelles sont instanciées automatiquement, et déléguées à un répartiteur de tâches SLURM (Yoo et al. 2003) gérant les 640 nœuds du cluster *OSIRIM*[1]. Le million et demi d'expériences résultantes au méta-niveau totalisent plus de 700 millions de cycles apprentissage-prédiction-évaluations, ce qui, même avec des ressources importantes, reste très coûteux (plus d'un mois de temps réel en exécution parallèle, pour des dizaines d'années de temps machine).

3.5 Interprétation des résultats

Résulte de ces exécutions une matrice de résultats en 4 dimensions : pour chaque combinaison *(Jeu de données, Sélection de méta-attributs, Méta-classifieur, Critère de performance)* on dispose d'une évaluation de la performance du méta-apprentissage résultant. Extraire de la connaissance de ce résultat revient alors à comparer des ensembles de ces combinaisons. Par exemple, pour comparer la performance d'une nouvelle méthode X

1. Voir : `http://osirim.irit.fr/`

de sélection d'algorithmes à l'état de l'art existant, il faudra comparer les évaluations des combinaisons impliquant la méthode X à celles impliquant diverses méthodes de l'état de l'art.

Une simple comparaison de performance moyenne permet la découverte de tendances. Cependant, afin de contrôler le risque que ces tendances ne reflètent pas de réelles différences de distribution, il conviendra d'effectuer des tests d'hypothèse statistique appropriés. Ces tests ayant la réputation d'être souvent mal employés (Demšar 2006), nous nous sommes limités à des tests non paramétriques dont les hypothèses sont vérifiables.

Nous illustrerons notre méthodologie sur le test de Friedman (Friedman 1940). En effet, ce dernier permet la comparaison de n échantillons appariés, ce qui coïncide parfaitement avec nos résultats multidimensionnels. Choisissant une dimension à étudier (par exemple celle du critère de performance de base), on peut très naturellement rassembler en échantillons les expériences utilisant le même critère, en les appariant selon toutes les autres dimensions (voir Table 3.5).

	Echantillons			
Triplets	Critère C_1	Critère C_2	Critère C_3	...
(Méta-classifieur, Sélection de méta-attributs, Jeu de données)	Performance	.	.	.
...	.	.	.	.

TABLE 3.5 – Échantillonnage de la performance des expériences au méta-niveau selon la dimension du critère de performance de base

Le test de Friedman est ainsi conçu pour permettre la comparaison de ces échantillons. Plus précisément, son objet est de discréditer l'hypothèse nulle :

$$H_0 = \{ \text{ Les } n \text{ échantillons proviennent d'une même distribution } \}$$

Le test mesure ainsi la probabilité p d'observer les résultats mesurés sous l'hypothèse nulle. Vu que l'on a effectivement observé ces résultats, si cette probabilité p de les avoir observés sous H_0 est faible (communément $p < 0.05$), on peut raisonnablement rejeter H_0 au profit de :

$$H_1 = \{ \text{ Parmi les } n \text{ échantillons, au moins un provient d'une distribution différente } \}$$

Le test de Friedman permet donc de s'assurer de l'existence de différences réelles entre nos échantillons, mais trouver précisément lesquels diffèrent nécessite l'emploi de tests *post-hoc* afin de maîtriser le risque de produire au moins une erreur type 1 sur un ensemble de

tests. Ce risque global d'erreur de type 1, représente la probabilité qu'au moins une des hypothèses rejetées soit en réalité valide, ce qui pose problème lorsque l'on veut comparer deux à deux une multitude d'échantillons. Pour chaque paire d'échantillons on veut pouvoir vérifier si l'un présente des performances significativement supérieures à l'autre, ce qui revient à rejeter :

$$H_0 = \{ \text{ Les 2 échantillons proviennent d'une même distribution } \}$$

On utilisera donc le test *post-hoc* de Nemenyi (Nemenyi 1963) qui permet de maîtriser ce risque global d'erreur en effectuant les $n!$ comparaisons nécessaires. Pour chaque paire d'échantillons, on peut ainsi savoir si l'un domine significativement l'autre, ce qui donne un bon aperçu des effets de la dimension échantillonnée sur les performances au méta-niveau.

Aucune supposition ne pouvant être faite quant aux distributions de performances dans nos échantillons, nous nous sommes limités aux tests non-paramétriques de Friedman et Nemenyi. Ces tests ont cependant quelques pré-requis simples qu'il convient de vérifier. Les triplets doivent tout d'abord être représentatifs de la population, ce qui est ici simplement obtenu en en considérant l'entièreté. Les échantillons doivent ensuite être indépendants les uns des autres, ce qui est ici naturel : la réalisation d'une expérience n'influera en rien sur les suivantes. Enfin, les valeurs mesurées doivent être ordinales, ce qui est bien le cas pour notre mesure de performance au méta-niveau. L'emploi de ces tests non-paramétriques est donc bien valide, et, bien qu'ils soient en général moins *puissants* que leurs équivalents paramétriques, nous verrons que l'échelle de l'expérience permet néanmoins de facilement dégager des résultats significatifs.

Illustrons donc l'intérêt du test de Friedman sur un exemple. Parmi nos différents méta-classifieurs, une méthode d'ensembles, *END* (*Ensemble of Nested Dichotomies* de Dong et al. (2005)), a souvent présenté d'excellentes performances. Cette méthode nécessite qu'on lui fournisse un classifieur grâce auquel appliquer son procédé de construction d'ensemble, et sa performance variera donc sensiblement selon le classifieur fourni. On souhaite comparer les performances d'instanciations d'*END* utilisant différents classifieurs de type "k plus proches voisins (kNN)" (décrits en Table 3.6) [2]. Ces approches ont une performance moyenne très proche, différant de moins d'un centième de l'écart type des performances. Pourtant, le test de Friedman conclut à une différence significative ($p < 10^{-90}$) entre ces approches.

2. Manhattan-10 a été tronquée car présentant de trop nombreux échecs d'exécution.

k	Distance	Performance
5	Manhattan	0,8828
5	Euclidean	0,8853
10	Euclidean	0,8899
20	Euclidean	0,8872
20	Manhattan	0,8832

TABLE 3.6 – Performance moyenne d'END employant différents $k - NN$.

Trouver lesquelles diffèrent nécessite l'emploi de tests *post-hoc*, afin de maîtriser le risque de produire au moins une erreur type 1 sur un ensemble de tests. On présente en Figure 3.3 les résultats du test de Nemenyi, dont la valeur critique prend en compte le fait que 10 tests sont réalisés pour garantir un risque global d'erreur de type 1 inférieur à un seuil de tolérance (communément 0.05). On peut y voir que les variantes 10-Euclidienne et 20-Manhattan ne peuvent être considérées différentes de la 20-Euclidienne pour ce risque d'erreur. En revanche toutes les autres sont jugées significativement différentes malgré des performances très similaires !

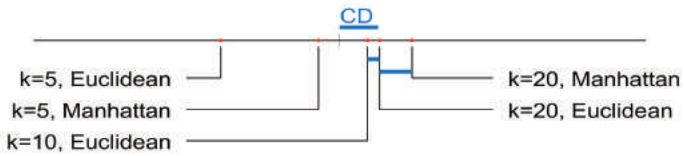

FIGURE 3.3 – Résultats du test de Nemenyi. (les groupes connectés ne sont pas significativement différents)

Exécuter ce test sur un plus grand nombre d'ensembles permet également d'extraire des résultats intéressants. Comparons par exemple les diverses méthodes de sélections d'attributs employées au méta-niveau. Le test de Friedman retourne une *p-value* de 0 (inférieure à la précision machine), nous assurant que la sélection d'attributs au méta-niveau *a un effet* sur les performances du méta-apprentissage. La Figure 3.4 (page 72) présente en détail le résultat du test de Nemenyi correspondant. On peut y remarquer certains groupes de méthodes virtuellement équivalentes, ainsi qu'y visualiser l'impact des paramètres de certaines méthodes, mais le résultat le plus frappant est que la majorité des méthodes sont significativement *moins* performantes que l'absence de sélection d'attributs au méta-niveau, suggérant l'importance de grands ensembles de méta-attributs pour la performance du méta-apprentissage. Ce résultat surprenant reste à tempérer par son exploration dans les dimensions non considérées dans cet exemple. Il se peut ainsi que sur un

méta-problème différent, la sélection d'attribut au méta-niveau se révèle très utile. Cependant, sur l'espace multi-dimensionnel exploré dans cette expérience, le test établit bien la dégradation significative des performances au méta-niveau par une majorité de méthodes de sélection d'attribut au méta-niveau.

3.6 Conclusion

Plus d'un million d'expériences de méta-apprentissage ont été réalisées pour démontrer la praticabilité du cadre d'évaluation, et les tests statistiques employés pour l'exploration des résultats valident les connaissances produites. Ils permettent par ailleurs d'étudier par une visualisation intuitive diverses questions que l'on peut se poser sur l'analyse de données, comme illustré ci-avant : *Quelle sélection d'attributs employer au méta-niveau ?* Au delà de son intérêt naturel dans l'évaluation de nouvelles méthodes de méta-analyse de données, cette approche permet d'étudier divers aspects du méta-niveau encore mal connus, tels la caractérisation de jeux de données (*Représentation statistique ou information théorétique ?*), ou la dépendance au critère d'optimisation de base (*Est il plus facile d'améliorer le kappa que l'AUC ?*).

Nous utiliserons donc la méthodologie présentée ici pour évaluer différentes propositions et améliorations possible au méta-niveau. Il conviendra alors de bien instancier les dimensions du cadre d'évaluation pour répondre au besoin de l'analyse. Ce dernier est en effet extrêmement polyvalent mais d'une complexité computationnelle potentiellement importante, facteur du produit des tailles des dimensions étudiées.

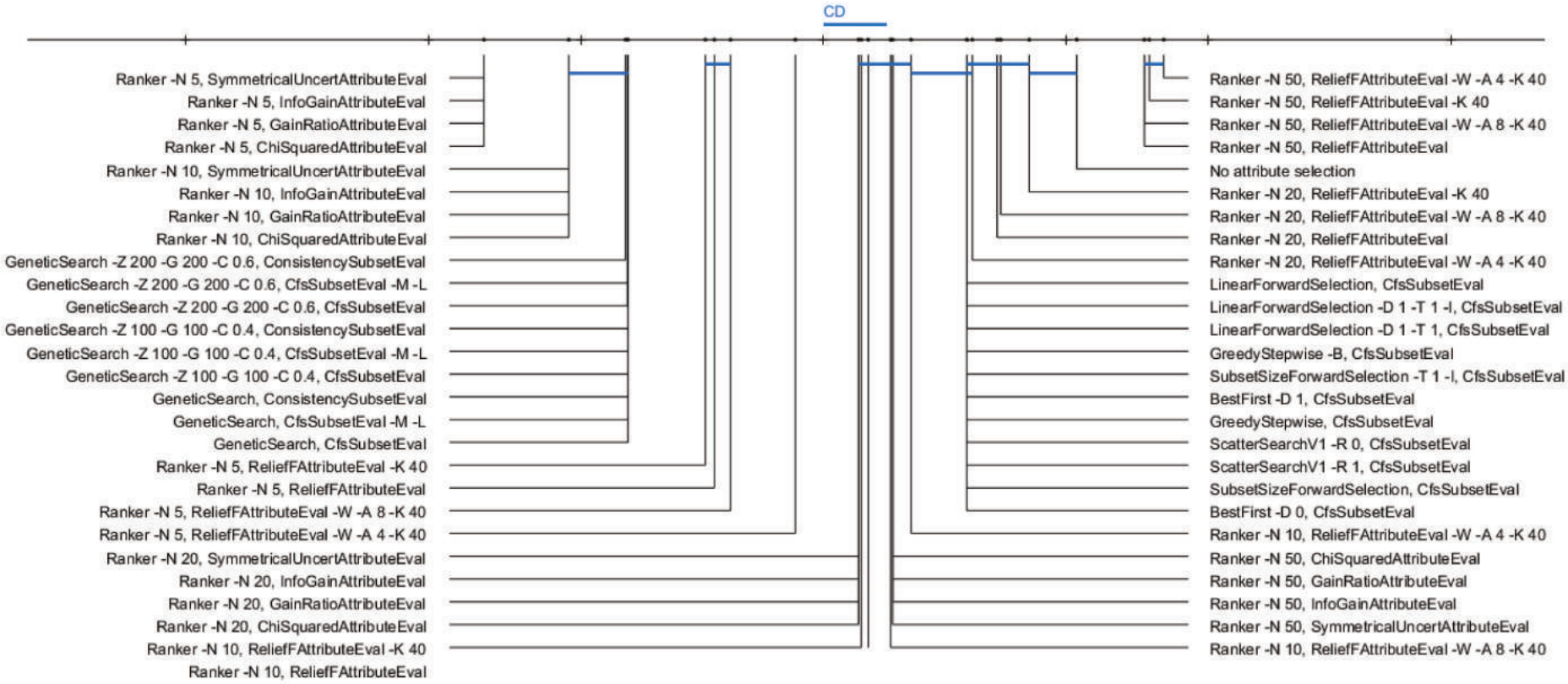

FIGURE 3.4 – Resultats du test de Nemenyi sur les méthodes de sélections d'attributs employées au méta-niveau. (les groupes connectés ne sont pas significativement différents)

Bibliographie du chapitre

Brazdil, Pavel, João Gama et Bob Henery (1994). « Characterizing the applicability of classification algorithms using meta-level learning ». In : *European conference on machine learning*. Springer, p. 83–102 (cf. p. 30, 31, 65, 96, 97).

Cohen, Jacob (1968). « Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. » In : *Psychological bulletin* 70.4, p. 213 (cf. p. 27, 28, 62, 100, 170).

Demšar, Janez (2006). « Statistical comparisons of classifiers over multiple data sets ». In : *The Journal of Machine Learning Research* 7, p. 1–30 (cf. p. 28, 68).

Dong, Lin, Eibe Frank et Stefan Kramer (2005). « Ensembles of balanced nested dichotomies for multi-class problems ». In : *Knowledge Discovery in Databases : PKDD 2005*. Springer, p. 84–95 (cf. p. 25, 26, 69, 95).

Friedman, Milton (1940). « A comparison of alternative tests of significance for the problem of m rankings ». In : *The Annals of Mathematical Statistics* 11.1, p. 86–92 (cf. p. 68).

Fürnkranz, J et J Petrak (2002). *Extended data characteristics*. Rapp. tech. Accessed 12/11/15 at `citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.302`. METAL consortium (cf. p. 34, 62).

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann et Ian H Witten (2009). « The WEKA data mining software : an update ». In : *ACM SIGKDD explorations newsletter* 11.1, p. 10–18 (cf. p. 66, 100, 168).

Kalousis, Alexandros, João Gama et Melanie Hilario (2004). « On data and algorithms : Understanding inductive performance ». In : *Machine Learning* 54.3, p. 275–312 (cf. p. 43, 61).

Kalousis, Alexandros et Maelanie Hilario (2001a). « Model selection via meta-learning : a comparative study ». In : *International Journal on Artificial Intelligence Tools* 10.04, p. 525–554 (cf. p. 30–32, 55, 78, 96).

Kalousis, Alexandros et Melanie Hilario (2001b). « Feature Selection for Meta-learning ». In : *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. PAKDD. London, UK, UK : Springer-Verlag, p. 222–233. isbn : 3-540-41910-1. url : `http://dl.acm.org/citation.cfm?id=646419.693650` (visité le 01/04/2017) (cf. p. 35, 61, 62).

Kononenko, Igor et Ivan Bratko (1991). « Information-Based Evaluation Criterion for Classifier's Performance ». In : *Machine Learning* 6.1, p. 67–80. issn : 0885-6125 (cf. p. 27, 28, 63, 169).

Leite, Rui, Pavel Brazdil et Joaquin Vanschoren (2012). « Selecting classification algorithms with active testing ». In : *Machine Learning and Data Mining in Pattern Recognition*. Springer, p. 117–131 (cf. p. 31, 55, 61, 96).

Leyva, Enrique, Adriana Gonzalez et Roxana Perez (2015). « A Set of Complexity Measures Designed for Applying Meta-Learning to Instance Selection ». In : *Knowledge and Data Engineering, IEEE Transactions on* 27.2, p. 354–367 (cf. p. 33, 61, 62).

Michie, Donald, David J Spiegelhalter et Charles C Taylor (1994). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA : Ellis Horwood. isbn : 0-13-106360-X (cf. p. 29, 32, 61).

Nemenyi, PB (1963). « Distribution-free multiple comparisons ». Thèse de doct. Princeton University (cf. p. 69).

Peng, Yonghong, Peter A Flach, Pavel Brazdil et Carlos Soares (2002). « Decision tree-based data characterization for meta-learning ». In : *IDDM-2002*, p. 111 (cf. p. 34, 61, 146).

Pfahringer, Bernhard, Hilan Bensusan et Christophe Giraud-Carrier (2000). « Tell me who can learn you and i can tell you who you are : Landmarking various learning algorithms ». In : *Proceedings of the 17th international conference on machine learning*, p. 743–750 (cf. p. 34, 61, 100).

Provost, Foster J, Tom Fawcett et Ron Kohavi (1998). « The case against accuracy estimation for comparing induction algorithms. » In : *ICML*. T. 98, p. 445–453 (cf. p. 27, 28, 56).

Robnik-Šikonja, Marko et Igor Kononenko (2003). « Theoretical and empirical analysis of ReliefF and RReliefF ». In : *Machine learning* 53.1-2, p. 23–69 (cf. p. 35, 36, 62).

Serban, Floarea, Joaquin Vanschoren, Jörg-Uwe Kietz et Abraham Bernstein (2013). « A survey of intelligent assistants for data analysis ». In : *ACM Computing Surveys (CSUR)* 45.3, p. 31 (cf. p. 17, 38, 41, 42, 55, 151).

Sun, Quan et Bernhard Pfahringer (2013). « Pairwise meta-rules for better meta-learning-based algorithm ranking ». In : *Machine learning* 93.1, p. 141–161 (cf. p. 30, 31, 61, 65, 96, 146).

Todorovski, Ljupco, Pavel Brazdil et Carlos Soares (2000). « Report on the experiments with feature selection in meta-level learning ». In : *Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP : forum for practical problem presentation and prospective solutions*. Citeseer, p. 27–39 (cf. p. 35, 61, 62).

Uno, Takeaki, Tatsuya Asai, Yuzo Uchida et Hiroki Arimura (2004). « An efficient algorithm for enumerating closed patterns in transaction databases ». In : *Discovery science*, p. 16–31 (cf. p. 65).

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl et Luis Torgo (2013). « OpenML : Networked Science in Machine Learning ». In : *SIGKDD Explorations* 15.2, p. 49–60. doi : 10.1145/2641190.2641198. url : http://doi.acm.org/10.1145/2641190.2641198 (visité le 01/04/2017) (cf. p. 28, 31, 38, 57, 60, 168).

Vilalta, Ricardo et Youssef Drissi (2002). « A Perspective View and Survey of Meta-learning ». In : *Artificial Intelligence Review* 18.2, p. 77–95. issn : 0269-2821. doi : 10.1023/A:1019956318069. url : http://dx.doi.org/10.1023/A:1019956318069 (visité le 01/04/2017) (cf. p. 29, 32, 61).

Xu, Lin, Frank Hutter, Jonathan Shen, Holger H Hoos et Kevin Leyton-Brown (2012). « SATzilla2012 : improved algorithm selection based on cost-sensitive classification models ». In : *SAT Challenge 2012 : Solver and Benchmark Descriptions*, p. 57–58 (cf. p. 37, 42, 55).

Yoo, Andy B, Morris A Jette et Mark Grondona (2003). « Slurm : Simple linux utility for resource management ». In : *Job Scheduling Strategies for Parallel Processing*. Springer, p. 44–60 (cf. p. 67, 113).

Zakova, Monika, Petr Kremen, Filip Zelezny et Nada Lavrac (2011). « Automating knowledge discovery workflow composition through ontology-based planning ». In : *Automation Science and Engineering, IEEE Transactions on* 8.2, p. 253–264 (cf. p. 41, 42, 55, 77, 151, 159).

La mathématique est l'art de donner le même nom à des choses différentes.

Henri Poincaré

De toute façon, l'abstraction mathématique chez l'humain [laisse trop à désirer].

Gabriel Ferrettini

Chapitre 4

Dissimilarité entre jeux de données

Comme présenté en section 2.4 de l'état de l'art, la caractérisation de jeux de données reste un verrou majeur de l'analyse de données intelligente. Une majorité d'approches à ce problème agrègent les informations décrivant les attributs individuels des jeux de données, ce qui représente une perte d'information. Nous proposons une approche par dissimilarité permettant d'éviter cette agrégation, et étudions son intérêt dans la caractérisation des performances d'algorithmes de classification et dans la résolution de problèmes de méta-apprentissage.

4.1 Motivation

4.1.1 Introduction

Ces dernières années ont vu naître un besoin grandissant en analyse de données, qui est souvent conduite par des experts de différents domaines ayant peu d'expérience en science des données. Afin de leur permettre de tout de même exploiter efficacement leurs données, divers travaux ont proposé des méthodes d'assistance intelligente à l'analyse de données (Serban 2013 ; Zakova et al. 2011). La caractérisation de jeu de données, problème apparu avec les premières ébauches de méta-apprentissage (Giraud-Carrier et al. 2004), constitue encore l'un des verrous majeurs de l'assistance intelligente à l'analyse de données.

Dans le cadre général du méta-apprentissage, le problème de caractérisation de jeu de données consiste en la définition d'un ensemble de propriétés de jeu de données (ou méta-attributs) permettant leur caractérisation précise qui doit de plus être utilisable par des algorithmes de méta-apprentissage. Afin de se conformer aux prérequis de la plupart

des algorithmes de méta-apprentissage, ces propriétés sont généralement agrégées en vecteurs d'attributs de taille fixe, ce qui peut représenter une importante perte d'information (Kalousis et Hilario 2001a). Nous étudions la possibilité que les limitations des techniques courantes de caractérisation de jeu de données soient l'un des obstacles majeurs à la bonne performance de la sélection d'algorithmes. Nous nous concentrons en particulier sur la définition d'une représentation des jeux de données permettant d'utiliser toute l'information disponible pour leur caractérisation.

4.1.2 Exemple

Le problème de caractérisation de jeux de données a donc déjà reçu une certaine attention dans le domaine du méta-apprentissage (voir la section 2.4 de l'état de l'art), mais l'agrégation des méta-attributs en vecteur de taille fixe y reste une constante. Cette agrégation représente cependant une importante perte d'information, que certaines approches ont déjà tenté de limiter, notamment par l'utilisation d'histogrammes (Kalousis 2002), ou par la définition de distances entre jeux de données (Smid 2016 ; Smid et Neruda 2014). On peut illustrer ce problème d'agrégation sur l'exemple suivant.

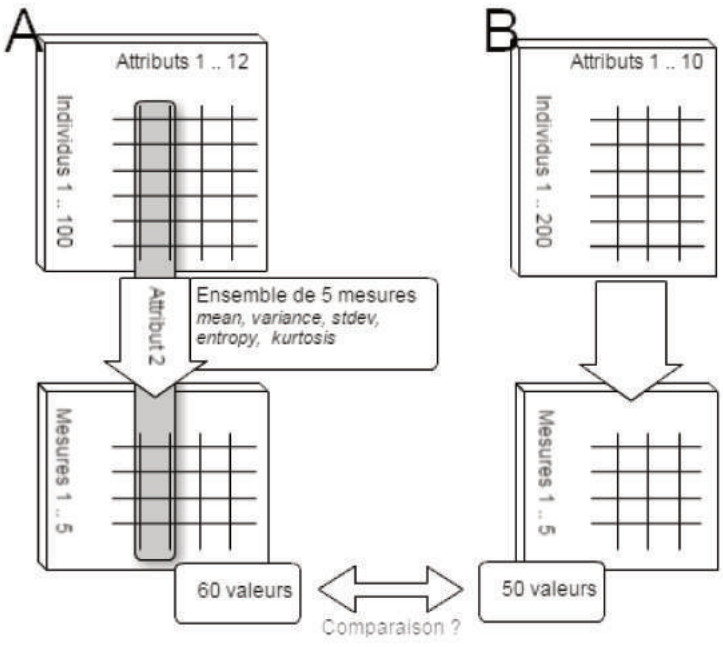

FIGURE 4.1 – Propriétés d'attributs individuels

Considérons deux jeux de données, **A** et **B** illustrés en figure 4.1. **A** décrit 12 attributs de 100 individus, et **B** 10 attributs de 200 individus. On souhaite comparer les résultats

de 5 mesures statistiques et informationnelles relevées sur les attributs individuels de ces jeux de données (comme illustré sur le second attribut de **A**). Ces mesures décrivant des attributs sont ainsi qualifiées de *méta-attributs*.

L'information complète que l'on souhaite comparer est donc un vecteur de 60 valeurs pour **A** et de 50 pour **B**. Une approche classique (Kalousis 2002 ; Wistuba et al. 2015) serait de faire une moyenne de chaque méta-attribut selon les différents attributs des jeux de données, perdant ainsi l'information caractérisant individuellement chaque attribut (Figure 4.2)).

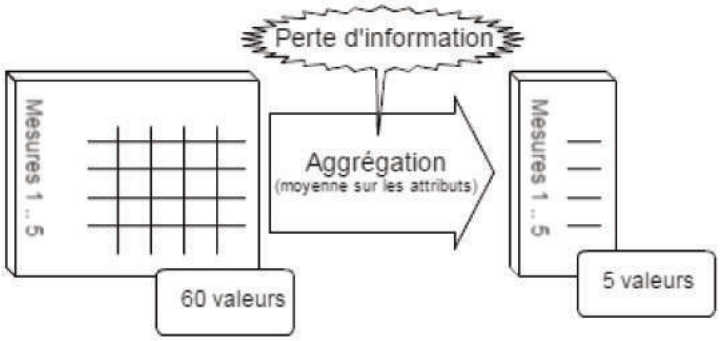

FIGURE 4.2 – Moyenne sur les attributs

Notre approche consiste à comparer les attributs de **A** et **B** par paires les plus similaires, comparant les attributs en surnombre à d'hypothétiques attributs vides. L'hypothèse émise ici est qu'un attribut absent équivaut à un attribut dont aucune valeur n'est connue. Pour en revenir à l'exemple, la comparaison des 5 mesures s'effectuera donc entre l'attribut de **A** et l'attribut de **B** les plus similaires *selon ces mêmes mesures*, puis sur les seconds plus similaires et ainsi de suite, pour finir par comparer les mesures relevées sur les deux attributs surnuméraires de **A** avec leur valeur sur un hypothétique attribut vide de **B**. Cette comparaison par paires permet de s'affranchir de l'ordre de présentation des attributs, qui ne recèle aucune information, se concentrant sur la topologie réelle du jeu de données (Figure 4.3)).

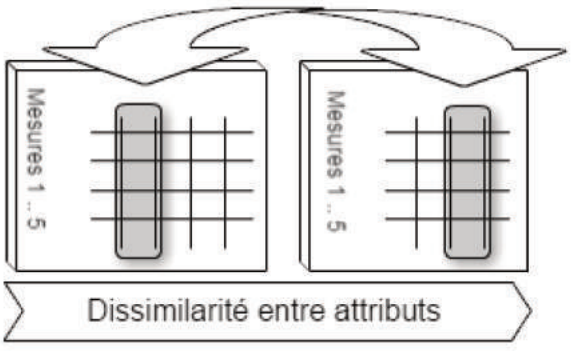

FIGURE 4.3 – Comparaison directe d'attributs

Cette comparaison peut s'effectuer par le biais d'une dissimilarité prenant en compte les propriétés des attributs individuels des jeux de données.

4.2 Fonction de dissimilarité

4.2.1 Propriétés désirables

Avant de proposer une fonction candidate, il convient d'étudier les propriétés qu'elle devrait présenter. Soit Ω l'ensemble des jeux de données, et $x, x' \in \Omega$ des instances de jeu de données. Traditionnellement, les propriétés usuelles des distances ne sont pas jugées nécessaires (Wang et al. 2009), ne conservant que la positivité ($d(x, x') \geq 0$). Parmi l'ensemble des propriétés d'une distance traditionnelle, nous préférerions cependant conserver uniquement celles présentant une interprétation naturelle dans le contexte de la caractérisation de jeu de données.

- $\surd$ Positivité ($d(x, x') \geq 0$) : une dissimilarité doit quantifier la différence absolue entre éléments, donc naturellement positive.
- $\surd$ Indiscernabilité des identiques ($x = x' \rightarrow d(x, x') = 0$) : des jeux de données rigoureusement identiques doivent être considérés aussi similaires que possible.
- $\times$ Identité des indiscernables ($d(x, x') = 0 \rightarrow x = x'$) : des jeux de données physiquement différents doivent pouvoir être considérés parfaitement similaires (considérer par exemple l'ordre de présentation des attributs).
- $\surd$ Symétrie ($d(x, x') = d(x', x)$) : l'ordre de présentation des jeux de données est indifférent, il parait donc naturel de l'ignorer.
- $\times$ Inégalité triangulaire ($d(x, x'') \leq d(x, x') + d(x', x'')$) : perd tout sens en l'absence d'identité des indiscernables. On peut avoir $d(x, x') = d(x', x'') = 0$ et néanmoins $x \neq x''$ et $d(x, x'') \geq 0$...

Définition 3 *Soit A un ensemble et d une fonction de $A^2 \rightarrow \mathbb{R}$. d est une* ***fonction de dissimilarité*** *sur A si et seulement si, $\forall x, x' \in A^2$:*

- $d(x, x') \geq 0$ *(Positivité)*
- $x = x' \rightarrow d(x, x') = 0$ *(Indiscernabilité des identiques)*
- $d(x, x') = d(x', x)$ *(Symétrie)*

Afin de construire une dissimilarité entre jeux de données, il faudra pouvoir comparer les valeurs de différentes propriétés de ces jeux de données. Ces propriétés étant possiblement très différentes les unes des autres de par leur sémantique et représentation, une normalisation sera nécessaire pour garantir qu'aucune composante ne domine les autres

ou ne soit ignorée. Ces propriétés sont formalisées dans la définition ci-dessous, où un ensemble de valeurs sera dit *atomique* s'il ne peut être divisé en sous-ensemble dont l'observation indépendante produirait autant d'information que l'observation de l'ensemble complet.

Définition 4 *Soit A un ensemble fini et d une fonction de dissimilarité sur A. d est une* ***fonction de dissimilarité normalisée*** *sur A si et seulement si au moins l'une des propriétés suivantes est vérifiée :*

1. *A est atomique et d est bornée sur A*
2. *Il existe une suite d'ensembles $E_1...E_n$ tels que $A = \prod_{i=1}^{n} E_i$, et une suite de fonctions de dissimilarité $d_1...d_n$ respectivement normalisées sur $E_1...E_n$, telles que :*

$$\forall \delta \in \mathbb{R},\ \exists \Delta \in \mathbb{R} \text{ tel que } \forall i \in [1..n] \text{ et } \forall a, b, c \in A,$$

$$Si\ d_i(a_i, b_i) = d_i(a_i, c_i) + \delta * \max_{(x,y)\in A^2}(d_i(x_i, y_i))$$
$$et\ \forall j \in [1..n],\ j \neq i,\ \ d_j(a_j, b_j) = d_j(a_j, c_j)$$

$$Alors\ d(a, b) = d(a, c) + \Delta \text{ et } \Delta = 0 \leftrightarrow \delta = 0$$

En d'autres termes, une variation d'amplitude δ relativement à sa borne supérieure, de toute composante d_i entre deux éléments de A induit une même variation Δ de d.

4.2.2 Fonction candidate

Nous proposerons ici une fonction de dissimilarité particulière présentant les propriétés énoncées précédemment. Pour construire ces fonctions, nous considérerons un ensemble fini de jeux de données ω, et deux ensembles de mesures. Le premier, G, consistera en des propriétés générales de jeu de données, telles que présentées en section 2. Le second, F, consistera en des propriétés capables de caractériser les attributs individuels de jeux de données.

Définition 5 *Soit $E_1...E_n$ une suite d'ensembles finis et A leur produit cartésien $\prod_{i=1}^{n} E_i$. Soit $d_1...d_n$ une suite de fonctions de dissimilarité respectivement sur $E_1...E_n$. On définit la* ***dissimilarité normalisée par la borne supérieure*** *sur A selon $d_1...d_n$, $d_A^{ubr} : A^2 \mapsto \mathbb{R}^+$ telle que*[1] *:*

$$\forall a, b \in A,\ d_A^{ubr}(a, b) = \sum_{i=1}^{n} \frac{d_i(a_i, b_i)}{\max\limits_{(x,y)\in A^2}(d_i(x_i, y_i))} \tag{4.1}$$

1. *ubr* pour *upper bound relative*

Proposition 1 *Soit $E_1...E_n$ une suite d'ensembles finis et A leur produit cartésien $\prod_{i=1}^{n} E_i$. Soit $d_1...d_n$ une suite de fonctions de dissimilarité respectivement normalisées sur $E_1...E_n$. Alors, la* ***dissimilarité normalisée par la borne supérieure*** *sur A selon $d_1...d_n$ est une* ***fonction de dissimilarité normalisée*** *sur A.*

Preuve 1 *Soit $\delta \in \mathbb{R}^*$. Les E_i étant des ensembles finis, et les d_i étant des dissimilarités normalisées, $\max_{(x,y)\in A^2} d_i(x_i, y_i)$ existe $\forall i$. Soit alors $(X, Y, Z) \in A^3$ tels que*

$$\exists i \in [1..n], d_i(X_i, Y_i) = d_i(X_i, Z_i) + \delta * \max_{(x,y)\in A^2}(d_i(x_i, y_i))$$
$$\forall j \in [1..n],\ j \neq i,\ d_j(X_j, Y_j) = d_j(X_j, Z_j)$$

Ce qui implique $d_A^{ubr}(X, Y) = d_A^{ubr}(X, Z) + \delta$. Ainsi, $\exists \Delta = \delta$, et d_A^{ubr} est bien une ***fonction de dissimilarité normalisée*** *sur A.*

Supposant que l'on puisse construire des fonctions de dissimilarité normalisées sur $G(\omega)$ et $F(\omega)$, on pourrait donc proposer d_ω^{ubr} comme fonction de dissimilarité normalisée sur ω. Afin d'alléger les notations, dans les paragraphes suivants, pour toute fonction H définie sur ω, on notera abusivement $d_H(H(x), H(y)) = d_H(x, y)$.

4.2.2.1 Méta-attributs des jeux de données

Soit un ensemble G de méta-attributs de jeux de données. Les valeurs $g(\omega)$ de l'un de ces méta-attributs g sur nos jeux de données constitueront le cas typique d'ensembles *atomiques* à partir desquels calculer la dissimilarité. On doit donc définir pour chaque méta-attribut g une dissimilarité bornée $d_g : g(\omega)^2 \mapsto \mathbb{R}^+$ (par exemple la différence absolue), qui selon la définition 2.1 sera donc normalisée. Ceci permet d'introduire la dissimilarité normalisée par la borne supérieure (cf. Eq. 4.1) sur $G(\omega)$ selon $\{d_g | g \in G\}$:

$$\forall x, y \in \omega,\ d_{G(\omega)}^{ubr}(x, y) = \sum_{g\in G} \frac{d_g(x, y)}{\max_{(x',y')\in\omega^2}(d_g(x', y'))} \tag{4.2}$$

En pratique, cela coïncidera généralement avec une distance de Manhattan normalisée. Cela pose en revanche les fondations nécessaires au prochain type de mesures : les méta-attributs caractérisant les attributs individuels des jeux de données.

4.2.2.2 Méta-attributs des attributs

Soit un ensemble F de *méta-attributs des attributs* permettant de caractériser les attributs individuels de jeux de données. Certains pourront caractériser tout type d'attribut (le *nombre de valeurs manquantes*, par exemple), tandis que d'autres seront restreints à

des types particuliers. Dans la définition de ces méta-attributs, nous considérons les deux types d'attributs les plus représentés : attributs nominaux (prenant un nombre fini de valeurs discrètes) et numériques (prenant valeur dans un espace non fini, souvent $\mathbb{R}$). Les vecteurs de méta-attributs caractérisant les attributs individuels présenteront donc nécessairement des valeurs manquantes (notées $\emptyset$) pour les mesures inadaptées à leur type, ce qui est un obstacle majeur à leur comparaison. En effet, la signification d'une différence de valeur entre deux jeux de données est intrinsèquement dépendante du méta-attribut considéré, et varie grandement d'un méta-attribut à l'autre. Afin de pouvoir comparer la valeur $f(x_i)$ d'un méta-attribut $f \in F$ sur x_i le i^{th} attribut du jeu de données $x \in \omega$, et x'_j le j^{th} attribut du jeu de données $x' \in \omega$, on introduit les fonctions $\delta_f : f(\omega)^2 \mapsto \mathbb{R}^+$ et $\delta_f^\emptyset : f(\omega) \mapsto \mathbb{R}^+$ telles que :

1. δ_f est une dissimilarité bornée sur l'ensemble atomique $f(\omega)$ (donc normalisée)
2. $\delta_f(x_i, \emptyset) = \delta_f(\emptyset, x_i) = \delta_f^\emptyset(x_i)$
3. $\delta_f^\emptyset$ est la *dissimilarité à l'absence de valeur* du méta-attribut f. Elle doit être définie en considérant le *sens* d'une valeur manquante de f, et sera détaillée plus avant par la suite.

On peut alors définir $\delta_F : F(\omega)^2 \mapsto \mathbb{R}^+$:

$$\delta_F(x_i, x'_j) = \sum_{f \in F} \frac{\delta_f(x_i, x'_j)}{\max\limits_{\substack{(y,y') \in \omega^2 \\ (p,q) \in \mathbb{N}^2}} \delta_f(y_p, y'_q)))} \tag{4.3}$$

δ_F permet de comparer des attributs de différents jeux de données selon les *Méta-attributs des attributs* de F. Cependant, comme illustré précédemment en Figure 4.3, l'objectif est de comparer ces attributs par paires. Ceci requiert un mapping entre les attributs de deux jeux de données :

Définition 6 *On définit une fonction de mapping σ comme une application associant à une paire de datasets $(x, x') \in \omega^2$, possédant respectivement n et n' attributs, une paire d'applications (σ_1, σ_2) injectives respectivement de $[\![1, n]\!]$ dans $[\![1, n']\!] \cup \{\emptyset\}$ et de $[\![1, n']\!]$ dans $[\![1, n]\!] \cup \{\emptyset\}$, telles que $\forall a \in [\![1, n]\!]$, et $b \in [\![1, n']\!]$, $(\sigma_1(a) = b) \Leftrightarrow (\sigma_2(b) = a)$. (Voir figure 4.4)*

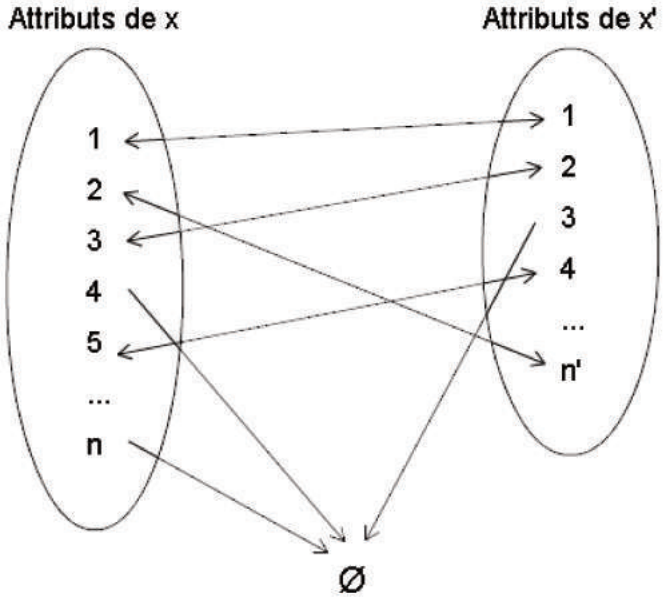

FIGURE 4.4 – Exemple de fonction de Mapping

Étant donnée une fonction de mapping σ, on peut définir $d^{\sigma}_{F(\omega)} : F(\omega)^2 \mapsto \mathbb{R}^+$ telle que :

$$d^{\sigma}_{F(\omega)}(x,x') = \frac{1}{max(n,n')}\Big(\sum_{\sigma_1(i)=j}^{i,j} \delta_F(x_i,x'_j) + \sum_{\sigma_1(i)=\emptyset}^{i} \delta_F(x_i,\emptyset) + \sum_{\sigma_2(j)=\emptyset}^{j} \delta_F(\emptyset,x'_j)\Big) \quad (4.4)$$

Proposition 2 *Une fonction de mapping σ est dite optimale si et seulement si*

$$\forall x,x' \in \omega d^{\sigma}_{F(\omega)}(x,x') = \min_{\sigma' \in Mappings(x,x')} d^{\sigma'}_{F(\omega)}(x,x')$$

Proposition 3 *Avec σ une fonction de mapping optimale, $d^{\sigma}_{F(\omega)}$ est une fonction de dissimilarité normalisée sur $F(\omega)$.*

Preuve 2 *On démontre successivement quatre propriétés : Positivité, Indiscernabilité des identiques, Symétrie et Normalisation (au sens de la Définition 4).*

1. *Soient $x,x' \in \omega$ et σ une fonction de mapping optimale.*

   Montrons que $d^{\sigma}_{F(\omega)}(x,x') \geq 0$.

   $\forall f \in F$, δ_f est une fonction de dissimilarité, donc

   $$\forall i,j \in [1,n] * [1,n'],\ \delta_f(x_i,x'_j) \geq 0$$

   et par somme, $d^{\sigma}_{F(\omega)}(x,x') \geq 0$

■

2. *Soit $x \in \omega$ ayant n attributs et σ une fonction de mapping optimale.*

   Montrons $d^{\sigma}_{F(\omega)}(x,x) = 0$.

$$d^{\sigma}_{F(\omega)}(x,x) = \frac{1}{n}\Big(\sum_{\sigma_1(i)=j}^{i,j} \delta_F(x_i,x_j) + \sum_{\sigma_1(i)=\emptyset}^{i} \delta_F(x_i,\emptyset) + \sum_{\sigma_2(j)=\emptyset}^{j} \delta_F(\emptyset,x_j)\Big)$$

   $\forall f \in F$, δ_f est une fonction de dissimilarité, donc

$$\forall i,j \in [1,n],\ \delta_f(x_i,x_j) \geq 0$$

$$\textit{et par somme, } d^{\sigma}_{F(\omega)}(x,x') \geq 0$$

   De plus, comme δ_f est une fonction de dissimilarité,

$$\forall i \in [1,n],\ \delta_f(x_i,x_i) = 0$$

   Donc, pour le mapping (Id, Id), avec Id la fonction Identité, on a :

$$d^{(Id,Id)}_{F(\omega)}(x,x) = \frac{1}{n}\Big(\sum_{i\in[1,n]} \delta_F(x_i,x_i)\Big) = 0$$

   Or, σ est optimale, donc :

$$d^{\sigma}_{F(\omega)}(x,x) = \min_{\sigma' \in Mappings(x,x)} d^{\sigma'}_{F(\omega)}(x,x)$$

$$0 \leq d^{\sigma}_{F(\omega)}(x,x) \leq d^{(Id,Id)}_{F(\omega)}(x,x)$$

$$0 \leq d^{\sigma}_{F(\omega)}(x,x) \leq 0$$

$$d^{\sigma}_{F(\omega)}(x,x) = 0$$

■

3. *Soient $x, x' \in \omega$ possédant respectivement n et n' attributs et σ une fonction de mapping optimale. Montrons que $d^{\sigma}_{F(\omega)}(x,x') = d^{\sigma}_{F(\omega)}(x',x)$.*

   Notons $\sigma(x,x') = (\sigma_1,\sigma_2)$ et $\sigma(x',x) = (\sigma'_1,\sigma'_2)$

   Moyennant substitution, supposons que $d^{\sigma}_{F(\omega)}(x,x') \leq d^{\sigma}_{F(\omega)}(x',x)$

   $\forall f \in F$, δ_f est une fonction de dissimilarité, donc

$$\forall i,j \in [1,n] * [1,n'],\quad \begin{cases} \delta_f(x_i,x'_j) = \delta_f(x'_j,x_i) \\ \delta_f(x_i,\emptyset) = \delta_f(\emptyset,x_i) \\ \delta_f(\emptyset,x'_j) = \delta_f(x'_j,\emptyset) \end{cases}$$

Il existe donc un mapping $(\sigma'_1, \sigma'_2) = (\sigma_2, \sigma_1)$ *tel que*

$$\frac{1}{max(n,n')}\Big(\sum_{\sigma_1(i)=j}^{i,j}\delta_F(x_i,x'_j)+\sum_{\sigma_1(i)=\emptyset}^{i}\delta_F(x_i,\emptyset)+\sum_{\sigma_2(j)=\emptyset}^{j}\delta_F(\emptyset,x'_j)\Big) \quad (A)$$

$$=$$

$$\frac{1}{max(n,n')}\Big(\sum_{\sigma_2(j)=i}^{i,j}\delta_F(x'_j,x_i)+\sum_{\sigma_2(i)=\emptyset}^{i}\delta_F(\emptyset,x_i)+\sum_{\sigma_1(j)=\emptyset}^{j}\delta_F(x'_j,\emptyset)\Big) \quad (B)$$

Or, σ *est une fonction de mapping optimale, donc*

$$d^{\sigma}_{F(\omega)}(x',x) \leq A$$

Et comme $A = B = d^{\sigma}_{F(\omega)}(x,x')$

$$d^{\sigma}_{F(\omega)}(x,x') \leq d^{\sigma}_{F(\omega)}(x',x) \leq d^{\sigma}_{F(\omega)}(x,x')$$

$$d^{\sigma}_{F(\omega)}(x,x') = d^{\sigma}_{F(\omega)}(x',x)$$

■

4. *Soit* $x \in \omega$*. L'ensemble* $F(x_i)$ *des valeurs de méta-attributs décrivant le feature* i *de* x *est alors une description atomique de* x_i*. De plus,* ω *est fini, ce qui entraine* $F(\omega)$ *fini et donc* $d^{\sigma}_{F(\omega)}$ *bornée sur* $F(\omega)$ *atomique. Selon la Définition 2.1,* $d^{\sigma}_{F(\omega)}$ *est donc bien normalisée sur* $F(\omega)$*.* ■

$d^{\sigma}_{F(\omega)}$ *est donc bien une fonction de dissimilarité normalisée sur* $F(\omega)$*.* □

4.2.2.3 Mappings

La fonction de mapping σ détermine donc comment les attributs seront comparés entre eux. On voudra alors apparier les attributs les plus similaires. Pour ce faire plusieurs options sont possibles :

Séparation des attributs par type Une part non négligeable des méta-attributs des attributs n'est calculable que sur un type d'attribut particulier, entrainant de nombreuses valeurs manquantes dans la comparaison d'attributs de types différents. Ceci justifie selon (Smid 2016) de considérer séparément les attributs de type différent dans le problème d'appariement. Ne considérant que les types numérique et nominal, cette séparation donne lieu à des mappings du type décrit en figure 4.5, qui seront qualifiés de "Split".

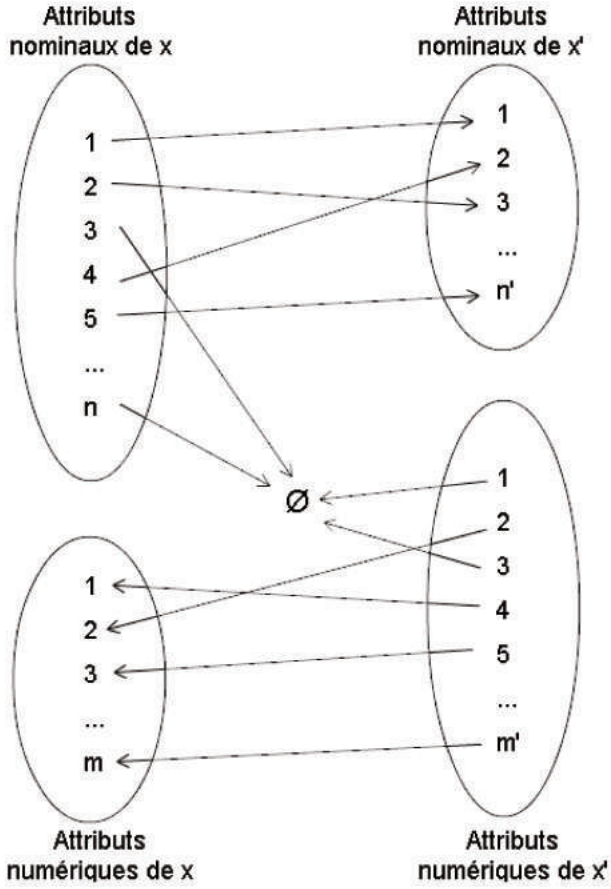

FIGURE 4.5 – Mapping "Split" avec séparation des attributs par type

Si au contraire on apparie simultanément les attributs numériques et nominaux, on obtient des mappings injectifs du plus grand ensemble d'attributs vers le plus petit (voir figure 4.6).

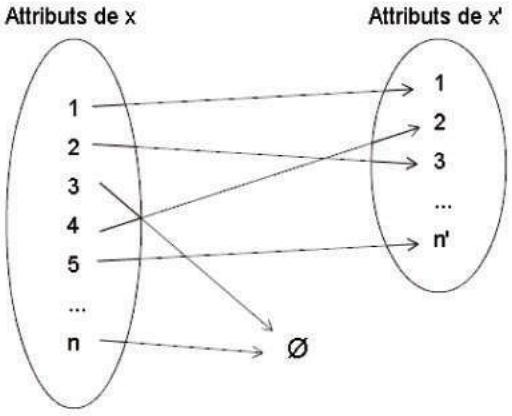

FIGURE 4.6 – Mapping "Mix" sans séparation des attributs par type

Méthode de minimisation Une fonction de mapping optimale apparie les attributs de manière à minimiser la dissimilarité résultante $d^{\sigma}_{F(\omega)}(x, x')$. On peut donc considérer la méthode de recherche de ce minimum comme part intégrante de la fonction de mapping. Une méthode de recherche exacte est proposée dans (Smid 2016), selon l'algorithme de Kuhn-Munkres (ou *algorithme hongrois*) (Kuhn 1955). Ce dernier adresse le problème d'affectation, usuellement représenté de la façon suivante :

> Problème d'affectation : Soit x équipes et y tâches, $x \geq y$, et une matrice $x \times y$ de réels positifs, contenant le temps nécessaire à chaque équipe pour réaliser chaque tâche. On souhaite affecter chaque tâche à une équipe afin de minimiser le temps total de réalisation, c'est-à-dire la somme des temps pris pour chaque tâche.

Prenons donc $x, x' \in \omega$ possédant respectivement n et n' attributs, avec $n \geq n'$. On identifie alors aux "équipes" les attributs de x et aux "tâches" ceux de x'. On doit ainsi calculer la matrice 4.1 des dissimilarités entre les attributs de x et x', ce qui représente une complexité en $\mathcal{O}(n^2\ c_F)$ avec c_F la complexité de δ_F.

	1	...	n'
1	$\delta_F(x_1, x'_1)$	.	$\delta_F(x_1, x'_{n'})$
...	.	.	.
...	.	.	.
...	.	.	.
n	$\delta_F(x_n, x'_1)$	.	$\delta_F(x_n, x'_{n'})$

TABLE 4.1 – Matrice des dissimilarités entre les attributs de x et x'

L'algorithme de Kuhn-Munkres, plus tard révisé par Edmonds et Karp, et indépendament Tomizawa (A. Frank 2005), fournit alors en temps polynomial ($\mathcal{O}(n^3)$) une solution optimale à ce problème d'affectation. Cette solution prend la forme d'une affectation des n' attributs de x' à des attributs distincts de x (voir Table 4.2). En associant alors les potentiels attributs surnuméraires (non affectés) de x à $\emptyset$, on obtient le mapping recherché.

	1	...	n'
1	$\delta_F(x_1, x'_1)$	.	$\delta_F(x_1, x'_{n'})$
...	.	.	.
...	.	.	.
...	.	.	.
n	$\delta_F(x_n, x'_1)$	.	$\delta_F(x_n, x'_{n'})$

TABLE 4.2 – Forme d'une solution au problème d'affectation

L'application de cette méthode exacte de minimisation pouvant se révéler très couteuse, on considérera à des fins de comparaison l'utilisation d'une méthode de minimisation naïve

de complexité moindre, détaillée par l'Algorithme 2.

Algorithme 2 : Méthode de minimisation gloutonne : "Greedy"

$Att_x \leftarrow \{1...n\}$
pour tous les *Attributs j de* x' **faire**
  pour tous les *Attributs i de x dans* Att_x **faire**
    Calculer $\delta_F(x_i, x'_j)$
  $k \leftarrow i$ tel que $\delta_F(x_i, x'_j) = \min_{a \in Att_x} \delta_F(x_a, x'_j)$
  Associer x_k à x'_j
  Retirer k de Att_x
pour tous les *Attributs i de x restant dans* Att_x **faire**
  Associer x_i à $\emptyset$

Cette méthode en $\mathcal{O}(\frac{n^2}{2} c_F)$ sera en général non optimale. Or, si l'on construit $d^{\sigma}_{F(\omega)}$ sur une fonction de mapping non optimale, on perd la symétrie et l'indiscernabilité des identiques, et donc la garantie d'avoir une fonction de dissimilarité normalisée.

Ces méthodes de minimisation fonctionnent indépendamment de la séparation des attributs par type. En effet, si l'on sépare les attributs numériques et nominaux, on résout simplement deux problèmes d'affectation plus simples.

Récapitulatif La complexité des méthodes de minimisation exposées ci-avant dépend de celle de la fonction δ_F. Cette fonction comprend une normalisation par la borne supérieure, donc potentiellement coûteuse sur de grands ensembles si cette borne doit être calculée. On s'affranchira de ce problème à l'implémentation, en maintenant un registre de maxima des ensembles à normaliser. Ceci permet de calculer la dissimilarité entre deux attributs δ_F en $\mathcal{O}(card(F))$, complexité de l'ordre du nombre de méta-attributs des attributs.

	Complexité	Description
GreedyMix	$\mathcal{O}(\frac{card(F)}{2} n^2)$	Méthode de minimisation naïve appliquée simultanément aux attributs numériques et nominaux.
GreedySplit	$\mathcal{O}(\frac{card(F)}{2} n^2)$	Méthode de minimisation naïve appliquée séparément aux attributs numériques et nominaux.
ExactMix	$\mathcal{O}(n^3)$	Algorithme Hongrois appliqué simultanément aux attributs numériques et nominaux.
ExactSplit	$\mathcal{O}(n^3)$	Algorithme Hongrois appliqué séparément aux attributs numériques et nominaux.

TABLE 4.3 – Fonctions de mapping considérées

On peut alors récapituler les différentes fonctions de mapping considérées en Table 4.3. A noter que bien que les ordres de complexité soient équivalents, les méthodes "Split" seront en général de complexité inférieure, et dans le pire cas où tous les attributs sont de même type, équivalentes aux méthodes "Mix".

4.2.2.4 Composition

Par le biais des équations 4.2 et 4.4, on peut donc proposer $\forall x, y \in \omega$ la **dissimilarité normalisée par la borne supérieure** sur ω selon $d^{ubr}_{G(\omega)}$ et $d^{\sigma}_{F(\omega)}$ telle que :

$$d^{ubr}_{\omega}(x,y) = \frac{d^{ubr}_{G(\omega)}(x,y)}{\max\limits_{(x',y')\in\omega^2}(d^{ubr}_{G(\omega)}(x',y'))} + \frac{d^{\sigma}_{F(\omega)}(x,y)}{\max\limits_{(x',y')\in\omega^2}(d^{\sigma}_{F(\omega)}(x',y'))} \quad (4.5)$$

D'après la Proposition 1, d^{ubr}_{ω} est bien une fonction de dissimilarité normalisée sur ω.

4.3 Preuve de concept

Afin d'évaluer l'intérêt de la dissimilarité proposée, des expériences ont été réalisées dans plusieurs contextes pour en mesurer le potentiel. Une implémentation simple de la dissimilarité utilisant une fonction de mapping de type "GreedyMix" (méthode de minimisation naïve appliquée simultanément aux attributs numériques et nominaux) a été réalisée pour preuve de concept, puis soumise à une procédure d'évaluation en deux étapes. Le choix du mapping *GreedyMix* vise à limiter la complexité déjà importante de l'expérience et à fournir un estimateur bas de l'intérêt des dissimilarités : le *GreedyMix* étant l'approche la plus naïve, les autres mappings pourront probablement mieux se comporter.

Dans un premier temps, nous étudierons son potentiel pour l'apprentissage, en utilisant les méthodes d'estimation présentées dans (Wang et al. 2009). Dans un second temps, nous décrirons une expérience de méta-apprentissage reprenant le cadre d'évaluation du chapitre 3 pour examiner l'intérêt de la dissimilarité dans un large panel de scénarios.

Ces deux approches requerront la définition d'un problème précis de méta-apprentissage comme cas d'étude, et la collecte de données caractérisant ce problème. Nous réutiliserons le problème de méta-apprentissage du chapitre 3 afin de bénéficier des résultats d'expérience déjà produits. On dispose donc toujours des informations de la base d'OpenML (Vanschoren et al. 2012) décrites précédemment.

4.3.1 Évaluation Théorique

Dans (Wang et al. 2009), Wang & al. proposent une définition intuitive de la qualité d'une fonction de dissimilarité dans le contexte de l'apprentissage. Selon leur définition, une fonction de dissimilarité est *strongly* (ϵ, γ)*-good* pour un problème donné de classification binaire, si une proportion au moins $1 - \epsilon$ des exemples $z = (x, y)$ satisfait :

$$P(d(x, x') < d(x, x'') \mid y' = y,\ y'' = -y) \geq \tfrac{1}{2} + \tfrac{\gamma}{2}$$

En d'autres termes, plus la probabilité que la dissimilarité juge deux exemples aléatoires de même classe plus proches que deux exemples aléatoires de classes différentes est grande, mieux elle permettra de séparer les classes. Cette interprétation nous amène à définir un problème de classification binaire entrant dans les attributions de la dissimilarité proposée.

Considérons un ensemble X de jeux de données de classification, et un ensemble A de classifieurs. On exécute chaque classifieur de A sur chaque jeu de données de X et mesure un critère de performance c du modèle résultant. Ensuite, pour chaque jeu de données $x \in X$, on définit l'ensemble A_x des algorithmes *appropriés* sur ce jeu de données selon le critère de performance c, comme ceux étant à au plus un écart type en dessous du meilleur :

$$A_x = \{a \in A \text{ tels que } |\max_{a' \in A}(c(a', x)) - c(a, x)| \leq \sigma_x\}$$

On peut donc considérer, pour tout algorithme $a \in A$, le problème de classification binaire où les instances sont les jeux de données $x \in X$, et dont la classe spécifie si a est approprié sur x. Ces problèmes caractérisent donc l'adéquation entre les différents classifieurs et jeux de données, ce qui est un objectif intuitif de la dissimilarité proposée.

Pour évaluer la dissimilarité, on peut alors calculer pour chaque classifieur $a \in A$ et chaque jeu de données $x \in X$, la probabilité que la dissimilarité juge deux exemples aléatoires de même classe plus proches que deux exemples aléatoires de classes différentes, ce qui nous donnera la (ϵ, γ)*-goodness* de d_ω^{ubr} :

$$P(d_\omega^{ubr}(x, x') < d_\omega^{ubr}(x, x'') \mid a \in A_x,\ a \in A_{x'},\ a \notin A_{x''})$$

Le résultat suivant de (ibid.), stipule que si d est une fonction de dissimilarité *strongly* (ϵ, γ)*-good*, alors il existe un classifieur simple construit sur d qui, sur le choix de $n = \frac{4}{\gamma^2} ln \frac{1}{\delta}$ paires d'exemples aléatoires de classes différentes, aura, avec une probabilité

au moins $1 - \delta$, un taux d'erreur d'au plus $\epsilon + \delta$. Ce résultat permet d'estimer de manière naturelle l'adéquation de la dissimilarité au problème étudié.

On peut alors évaluer l'intérêt théorique de la dissimilarité pour l'apprentissage en calculant son (ϵ, γ)-*goodness* dans diverses situations et en la comparant avec les aggrégations classiques des méta-attributs des jeux de données : distance euclidienne et de Manhattan. On construit pour ce faire les problèmes d'adéquation de 93 classifieurs sur 434 jeux de données d'OpenML (listés en annexe, Table A.1 et A.2), sur lesquels évaluer la (ϵ, γ)-*goodness* de nos trois fonctions. Le paramètre γ, réduisant quadratiquement le nombre d'instances nécessaires, a été amené aussi haut que possible en conservant $\epsilon \leq 0.05$. Les résultats sont présentés en Table 4.4, moyennés selon les classifieurs et jeux de données. La Table 4.5 présente le risque δ et la borne de taux d'erreur obtenus pour différents nombres d'exemples.

Dissimilarité	ϵ	γ
d_ω^{ubr}	0,05	0,024
Distance Euclidienne	0,05	0,015
Distance de Manhattan	0,05	0,014

TABLE 4.4 – (ϵ, γ)-*goodness* moyenne avec différentes dissimilarités

Comme on peut le voir en Table 4.5 (page 93), la dissimilarité proposée offre des bornes intéressantes au taux d'erreur avec sensiblement moins d'exemples (au pire 11.3% d'erreurs pour 20.000 exemples, ce qui n'est atteint par la distance euclidienne qu'avec 50.000 exemples...). Elle semble donc être plus adaptée à la caractérisation d'adéquation entre classifieur et jeu de données que les autres distances étudiées. Ce résultat est en revanche nécessairement dépendant du choix des algorithmes et jeux de données sur lesquels sont construits ces problèmes d'adéquation, et on ne peut donc pas le supposer généralisable. Ce qui est montré ici, est que pour *certains* algorithmes, l'utilisation de la dissimilarité proposée permet de caractériser leur adéquation aux jeux de données plus efficacement que les distances traditionnelles. Parmi les algorithmes où la dissimilarité présente d'excellentes performances, on peut noter une majorité de classifieurs de type arbre de décision. On pourrait donc postuler que la dissimilarité caractérise bien l'adéquation des approches par arbres de décision aux jeux de données, et donc que cette adéquation dépend largement des méta-attributs d'attributs individuels utilisés par d_ω^{ubr}.

	1000 exemples		5000 exemples	
	δ	erreur max	δ	erreur max
d_ω^{ubr}	0,871	0,921	0,501	0,551
Distance Euclidienne	0,945	0,995	0,755	0,805
Distance de Manhattan	0,952	1,002	0,783	0,833

	20000 exemples		50000 exemples	
	δ	erreur max	δ	erreur max
d_ω^{ubr}	0,063	0,113	0,001	0,051
Distance Euclidienne	0,325	0,375	0,060	0,110
Distance de Manhattan	0,375	0,425	0,086	0,136

TABLE 4.5 – Borne du taux d'erreur obtenu avec une probabilité $1 - \delta$ pour différents nombres d'exemples et par des classifieurs construits sur différentes dissimilarités.

4.3.2 Évaluation Expérimentale

Afin d'évaluer l'intérêt de la dissimilarité dans le cadre de la sélection d'algorithmes par méta-apprentissage, nous avons mis en place une série d'expériences comparatives autour du problème de classification au méta-niveau reprenant le cadre d'évaluation du chapitre 3.

À l'expérience décrite dans le chapitre chapitre 3, nous ajoutons donc un classifieur des "k plus proches voisins" (*kNN*, ou *IBk* dans Weka) employant la dissimilarité proposée comme distance entre instances, et permettons à ce classifieur les mêmes déclinaisons d'hyperparamètres et méthodes d'augmentation que les autres classifieurs. On réalise donc une nouvelle étude au méta-niveau, comparant de nombreux classifieurs dont une partie emploie la fonction de dissimilarité proposée.

Ce procédé fournit alors une valeur de performance pour chacune des plus de 200.000 nouvelles exécutions. Notre objectif étant de discriminer entre classifieurs au méta-niveau, et les valeurs de performance étant commensurables, il est possible d'en faire la moyenne selon les autres dimensions. On obtient donc, pour chaque classifieur au méta-niveau, sa performance moyenne pour différentes approches de sélection d'attributs, critères d'évaluation à maximiser, et ensembles d'apprentissage. On minimise ainsi la variance introduite par ces différents facteurs de bruit. Notre jeu de données de méta-classification étant néanmoins de taille relativement petite (434 instances), on ne peut complètement négliger la variance de ces résultats, et il conviendra donc de les étudier dans leur globalité, car plus

on "zoome" sur les résultats, plus on a de chances d'observer un résultat contingent, fonction uniquement du contexte de l'expérience. Nous avons donc concentré nos observations sur la répartition des classifieurs de méta-niveau utilisant la dissimilarité proposée dans la population. Dans la Figure 4.7, on compare, pour différents seuils de performance, la proportion de classifieurs de méta-niveau utilisant ou non la dissimilarité proposée. On peut constater que parmi les approches conseillant en moyenne des classifieurs jusqu'à 5% moins performants que le meilleur, 20% sont basées sur la dissimilarité proposée, alors que cette proportion est de moins de 10% dans la population complète.

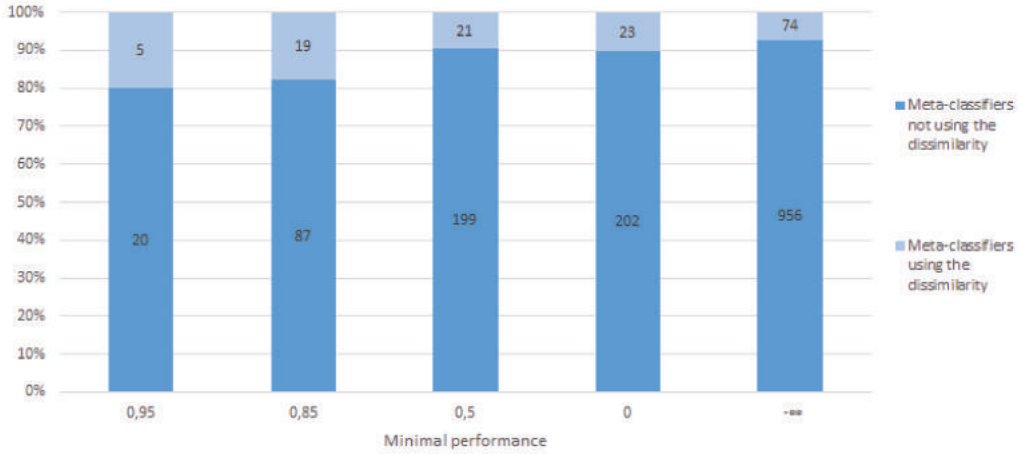

FIGURE 4.7 – Populations de méta-classifieurs atteignant divers seuils de performance

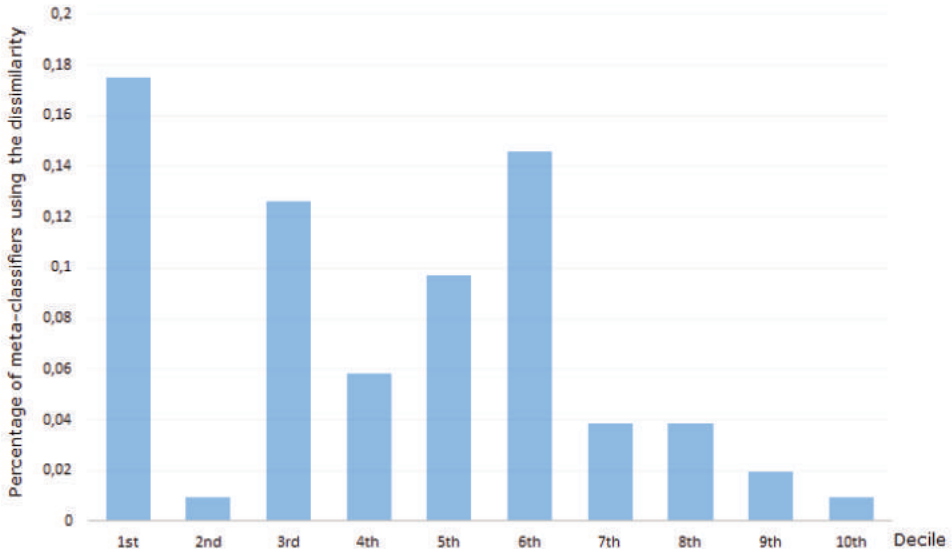

FIGURE 4.8 – Proportion dans chaque décile de performance de méta-classifieurs utilisant la dissimilarité

La Figure 4.8 présente la proportion de classifieurs de méta-niveau utilisant la dissi-

milarité étudiée dans chaque décile de performance. Là encore, on peut noter que parmi les 10% de meilleurs classifieurs de méta-niveau, plus de 17% utilisent la dissimilarité. De plus, malgré des divergences aux deuxième, quatrième et sixième déciles, on peut observer que cette proportion semble décroître linéairement avec la baisse de performance. Cette tendance est plus visible en Figure 4.9, présentant la proportion de classifieurs de méta-niveau utilisant la dissimilarité par quintiles cumulés.

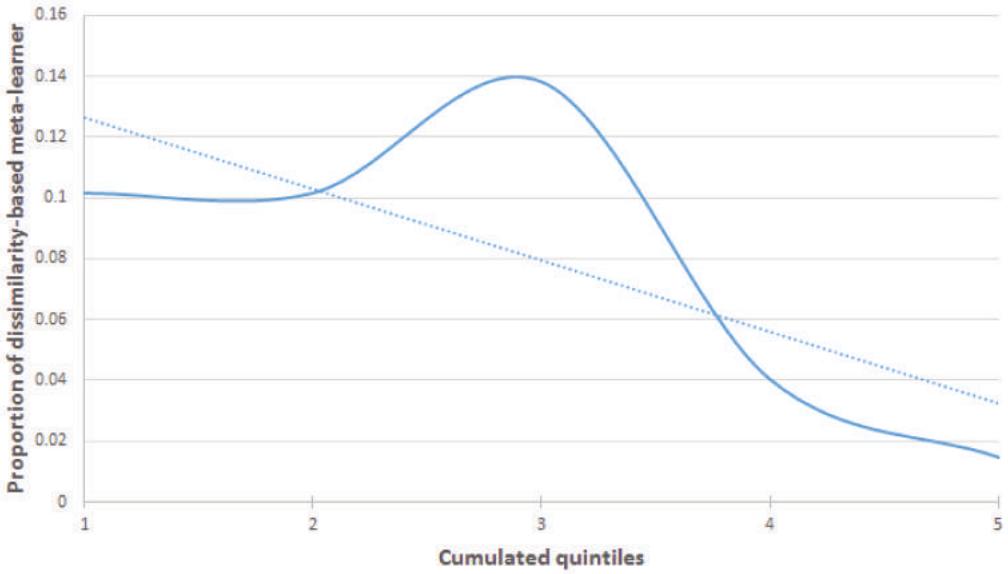

FIGURE 4.9 – Proportion de méta-classifieurs utilisant la dissimilarité par quintiles de performance cumulés

Ces résultats ne permettent pas d'établir la supériorité générale des approches employant la dissimilarité basique étudiée, mais montrent bien que certaines approches en bénéficient grandement. Ce constat n'est pas particulièrement surprenant dans la mesure où le méta-apprentissage est toujours concerné par les limitations des *"no free lunch theorems"* (D. H. Wolpert 1996 ; Wolpert et Macready 1997). En effet, toute nouvelle méthode peut, au mieux, faire montre de meilleures performances sur une partie spécifique du problème. Parmi les méthodes bénéficiant le plus de l'utilisation de la dissimilarité, on peut relever une majorité d'approches divisant le problème en plusieurs sous-problèmes de classification binaire, telles *"ensemblesOfNestedDichotomies"* (Dong et al. 2005) ou *"ClassificationViaRegression"* (Frank et al. 1998). Ceci pourrait suggérer que la dissimilarité proposée est particulièrement efficace sur de tels sous-problèmes, qui reviennent à caractériser les domaines de bonne performance d'algorithmes particuliers. Il s'agit là de l'une des problématiques récurrentes de l'apprentissage, et donc d'un développement potentiel intéressant pour la dissimilarité.

4.4 Expériences comparatives

Au vu des résultats encourageants de la preuve de concept, on peut étudier plus en détail l'impact en terme de performances des différents composants de la dissimilarité. Nous avons donc implémenté des variantes de ces différents composants, et développé un protocole approprié à leur évaluation, suivant le cadre d'évaluation du chapitre 3. Ce chapitre présentera tout d'abord le méta-problème sur lequel seront évaluées les dissimilarités, puis présentera les variations de dissimilarité étudiées. Nous détaillerons ensuite les expériences menées et leurs résultats.

4.4.1 Forme du méta-problème

Les résultats de la preuve de concept semblent suggérer que les dissimilarités prenant en compte les méta-attributs des attributs seraient particulièrement appropriées pour caractériser la performance d'algorithmes d'apprentissage particuliers sur des jeux de données. On s'attachera donc ici à construire un méta-problème exploitant cette caractéristique.

Dans la preuve de concept, nous avons évalué la dissimilarité sur un problème de méta-classification, où la "classe" d'un jeu de données était l'algorithme qui y obtenait les meilleures performances. Il s'agit de l'une des premières approches de méta-apprentissage, ayant l'avantage d'être très simple, mais de nombreuses méthodes plus performantes ont depuis été développées (Giraud-Carrier et al. 2004). Par exemple, dans (P. Brazdil et al. 1994), le méta-problème est divisé en autant de sous-problèmes que de classifieurs, et l'on apprend alors pour chacun un modèle d'applicabilité. Dans (Kalousis et Hilario 2001a), on apprend plutôt un modèle pour chaque paire de classifieurs, prédisant sur quels jeux de données chacun dominera. Cette décomposition du méta-problème par paire de classifieurs est reprise dans (Sun et Pfahringer 2013), où des ensembles de règles sont appris pour comparer les performances des classifieurs dans chaque paire. D'autres travaux brisent le cadre traditionnel du problème de méta-apprentissage, tel (Leite et al. 2012), qui au lieu d'utiliser des ensembles de méta-données décrivant la performance de divers algorithmes sur des jeux de données, propose une stratégie de recherche d'algorithmes performants minimisant le nombre de tests à réaliser. De même, (Sun et al. 2012) considère l'optimisation des hyperparamètres en plus de la sélection d'algorithmes, et utilise des techniques d'optimisation pour chercher des solutions performantes.

Dans le cas présent, on peut définir un méta-problème s'appuyant sur l'abondance

des méta-données d'OpenML, qui devrait idéalement refléter les points forts supposés de la dissimilarité. Afin de caractériser la performance individuelle des algorithmes, on peut reprendre une division du méta-problème par algorithme (P. Brazdil et al. 1994). En revanche, plutôt que de se limiter à prédire si les algorithmes sont applicables ou non, la dissimilarité permet de manière très intuitive de modéliser directement leur performance. En effet, un simple algorithme de type *"k plus proches voisins"* permet d'agréger la performance d'un algorithme sur les k jeux de données les plus proches (selon la dissimilarité). Le pseudocode de l'Algorithme 3 décrit la structure du méta-problème retenue.

Algorithme 3 : Forme d'une solution au méta-problème

Data :
- Un metadataset $\mathcal{M}$ décrivant la performance de x classifieurs sur y jeux de données
- Un nouveau jeu de données D

Result : La recommandation d'un ensemble de n algorithmes $c_1...c_n$ de $\mathcal{M}$ supposés capables de bonnes performances sur D, et leur intérêt relatif attendu $\alpha_1...\alpha_n$

$Voisins \leftarrow k$ jeux de données du metadataset les plus proches de D selon la dissimilarité considérée
foreach *Classifieur* c *de* $\mathcal{M}$ **do**
    Estimer la performance inconnue de c sur D selon la performance connue de c sur les $Voisins$ de D
Ordonner les classifieurs du metadataset selon l'estimation de leur performance sur D
Recommander les n meilleurs pondérés selon leur performance estimée

En plus de la dissimilarité elle-même, certains éléments de ce procédé peuvent varier, et impacteront potentiellement les résultats d'expériences. Il conviendra donc de considérer différentes valeurs pour ces paramètres, pour autoriser un maximum de généralité aux résultats d'expérience. Le nombre k de voisins considérés prendra des valeurs communes pour des ensembles de l'ordre de la centaine :

$$k \in \{3, 5, 10\}$$

L'estimation de la performance d'un classifieur c sur D selon sa performance sur les voisins de D pourra soit être une simple moyenne des performances de c sur les voisins de D, soit être pondérée par la dissimilarité :

$$\text{perf}_{mean}(c, D) = \frac{1}{k} \sum_{V \in Voisins} \text{perf}(c, V)$$

$$\text{perf}_{weighted}(c, D) = \frac{\sum_{V \in Voisins} d_{\omega}^{ubr}(D, V) * perf(c, V)}{\sum_{V \in Voisins} d_{\omega}^{ubr}(D, V)}$$

De plus, si l'on accepte ainsi en sortie un *ensemble* d'algorithmes recommandés, on doit définir un critère de performance capable d'évaluer des solutions au méta-problème produisant de tels ensembles. On généralise ainsi le critère introduit dans la preuve de concept :

Définition 7 *Soit une solution $\mathcal{S}$ au méta-problème $(\mathcal{M}, D)$ recommandant les classifieurs $c_1...c_n$ avec un poids relatif $\alpha_1...\alpha_n$. Soient $p_1...p_n$ les performances réelles respectives de $c_1...c_n$ sur D. Soient alors* ***best*** *la meilleure performance obtenue par un classifieur de $\mathcal{M}$ sur le jeu de données D, et* ***def*** *la performance du classifieur par défaut (prédisant la classe majoritaire) sur D. On définit tout d'abord la performance d'une recommandation c_i :*

$$perf(c_i) = \max(-1, 1 - \frac{|best - p_i|}{|best - def|})$$

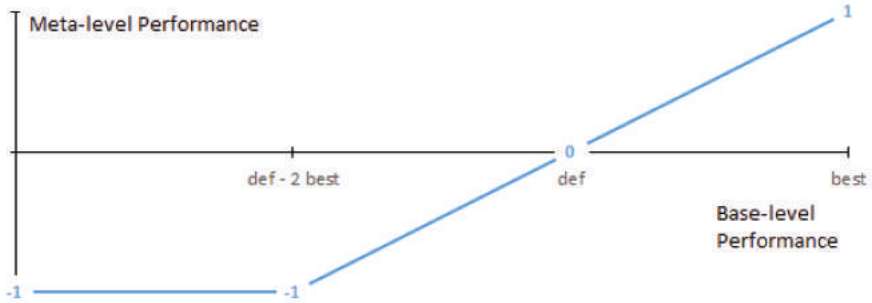

FIGURE 4.10 – Performance d'une recommandation c_i

Ce critère reprend celui de la preuve de concept, atteignant son maximum de 1 *quand le classifieur recommandé présente une valeur de précison maximale, et* 0 *quand il présente la même précision que le classifieur par défaut. On limite cependant ce critère en* -1 *car il fait peu de sens de discriminer entre des recommandations inutiles. On définit alors la performance de notre solution $\mathcal{S}$ au méta-problème $(\mathcal{M}, D)$:*

$$perf(\mathcal{S}) = \frac{\sum\limits_{i \in \{1...n\}} \alpha_i * perf(c_i)}{\sum\limits_{i \in \{1...n\}} \alpha_i}$$

Ce nouveau critère ne peut prendre de valeurs extrêmes que plus difficilement, requérant pour ce faire des recommandations unanimement extrêmes. Ceci devrait limiter la variance des résultats, mais pour étudier plus précisément l'impact en terme de performances

du nombre de recommandations, on fera varier le nombre d'algorithmes recommandés en sortie :

$$n \in \{1, 3, 5\}$$

4.4.2 Ensembles de méta-attributs

Les méta-attributs retenus dans cette expérience sont en partie repris des travaux de J. Smid (Smid 2016) et reprennent les mesures courantes additionnées de diverses variations plus particulières. On les divisera en différents ensembles selon leur provenance afin d'évaluer l'impact de l'utilisation de différents méta-attributs sans trop augmenter la taille de l'expérience. On définit pour chaque méta-attribut une dissimilarité bornée sur son ensemble de valeurs adjoint de $\emptyset$, comme une distance de Manhattan sur son ensemble de valeur et une "distance à l'absence de valeur" particulière sur $\emptyset$.

Soit donc pour un méta-attribut a sa dissimilarité associée $d_a : (a(\omega) \cup \emptyset)^2 \mapsto \mathbb{R}^+$ telle que :

$$d_a(x, y) = \begin{cases} |x - y| & si \quad (x, y) \in a(\omega)^2 \\ \delta_a^{\emptyset}(x) & si \quad x \in a(\omega) \quad et \quad y = \emptyset \\ \delta_a^{\emptyset}(y) & si \quad y \in a(\omega) \quad et \quad x = \emptyset \\ \delta_a^{\emptyset}(\emptyset) & si \quad x = y = \emptyset \end{cases}$$

On détaillera donc ici les différents méta-attributs retenus et leur "distance à l'absence de valeur" $\delta^{\emptyset}$ associée. Cette "distance à l'absence de valeur" sera en général une distance de Manhattan à la valeur du méta-attribut considéré sur un attribut hypothétique vide ou uniforme (n'ayant qu'une seule valeur). Un tel attribut n'apporte en effet aucune information et est en cela identifiable à l'absence d'attribut. Pour certains méta-attributs, ce raisonnement n'est pas possible ou ne fait aucun sens (par exemple, comparer la moyenne d'un attribut avec celle d'un attribut vide est absurde), et l'on y considérera $\delta^{\emptyset}(x) = 0$. On associe une "distance à l'absence de valeur" à la fois aux méta-attributs généraux des jeux de données et à ceux des attributs, car elle est nécessaire à la définition de la dissimilarité sur les attributs, et assure une certaine robustesse aux valeurs manquantes de méta-attributs généraux.

4.4.2.1 Méta-attributs généraux des jeux de données

On présente ici les différents méta-attributs généraux des jeux de données. Ces derniers consistent en des propriétés simples des jeux de données (Table 4.6), un descriptif global des attributs numériques (Table 4.7), un descriptif global des attributs nominaux (Table 4.8), et en la performance de *landmarkers* évalués selon plusieurs critères (Tables 4.9, 4.10 et 4.11). Les *landmarkers* sont des algorithmes d'apprentissage simples, dont l'usage a été introduit dans (Pfahringer et al. 2000), que l'on applique au jeu de données considéré, afin d'y évaluer leur performance. Ceux utilisés ici proviennent de l'API Weka (Hall et al. 2009) et rassemblent différentes techniques classiques d'apprentissage. Les critères de performance retenus sont l'aire sous la courbe ROC (Table 4.9), le taux d'erreur (Table 4.10) et le coefficient Kappa de Cohen (Cohen 1968) (Table 4.11), qui capturent des aspects de la performance conceptuellement assez différents pour être considérés séparément.

Méta-attribut	Description	$\delta^{\emptyset}(x)$
DefaultAccuracy	The predictive accuracy obtained by predicting the majority class.	0
Dimensionality	Number of attributes divided by the number of instances.	0
MajorityClassPercentage	Percentage of rows with the class with the most assigned index.	$\|x-1\|$
MajorityClassSize	The number of instances that have the majority class.	0
MinorityClassPercentage	Percentage of rows with the class with the least assigned index.	$\|x-1\|$
MinorityClassSize	The number of instances that have the minority class.	0
NumberOfBinaryFeatures	Count of binary attributes.	0
NumberOfClasses	The number of classes in the class attribute.	0
NumberOfFeatures	Number of attributes (columns) of the dataset.	0
NumberOfInstances	Number of instances (rows) of the dataset.	0
NumberOfInstancesWithMissingValues	Number of instances with at least one value missing.	0
NumberOfMissingValues	Number of missing values in the dataset.	0
NumberOfNumericFeatures	Count of categorical attributes.	0
NumberOfSymbolicFeatures	Count of nominal attributes.	0
PercentageOfBinaryFeatures	Percentage of binary attributes.	0
PercentageOfInstancesWithMissingValues	Percentage of instances with missing values.	$\|x-1\|$
PercentageOfMissingValues	Percentage of missing values.	$\|x-1\|$
PercentageOfNumericFeatures	Percentage of numerical attributes.	0
PercentageOfSymbolicFeatures	Percentage of nominal attributes.	0

TABLE 4.6 – Méta-attributs simples des jeux de données

Méta-attribut	Description	$\delta^{\emptyset}(x)$
MeanMeansOfNumericAtts	Mean of means among numeric attributes.	0
MeanStdDevOfNumericAtts	Mean standard deviation of numeric attributes.	$\|x\|$
MeanKurtosisOfNumericAtts	Mean kurtosis among numeric attributes.	$\|x+1,2\|$
MeanSkewnessOfNumericAtts	Mean skewness among numeric attributes.	$\|x\|$
MinMeansOfNumericAtts	Min of means among numeric attributes.	0
MinStdDevOfNumericAtts	Min standard deviation of numeric attributes.	$\|x\|$
MinKurtosisOfNumericAtts	Min kurtosis among numeric attributes.	$\|x+1,2\|$
MinSkewnessOfNumericAtts	Min skewness among numeric attributes.	$\|x\|$
MaxMeansOfNumericAtts	Max of means among numeric attributes.	0
MaxStdDevOfNumericAtts	Max standard deviation of numeric attributes.	$\|x\|$
MaxKurtosisOfNumericAtts	Max kurtosis among numeric attributes.	$\|x+1,2\|$
MaxSkewnessOfNumericAtts	Max skewness among numeric attributes.	$\|x\|$
Quartile1MeansOfNumericAtts	First quartile of means among numeric attributes.	0
Quartile1StdDevOfNumericAtts	First quartile of standard deviation of numeric attributes.	$\|x\|$
Quartile1KurtosisOfNumericAtts	First quartile of kurtosis among numeric attributes.	$\|x+1,2\|$
Quartile1SkewnessOfNumericAtts	First quartile of skewness among numeric attributes.	$\|x\|$
Quartile2MeansOfNumericAtts	Second quartile of means among numeric attributes.	0
Quartile2StdDevOfNumericAtts	Second quartile of standard deviation of numeric attributes.	$\|x\|$
Quartile2KurtosisOfNumericAtts	Second quartile of kurtosis among numeric attributes.	$\|x+1,2\|$
Quartile2SkewnessOfNumericAtts	Second quartile of skewness among numeric attributes.	$\|x\|$
Quartile3MeansOfNumericAtts	Third quartile of means among numeric attributes.	0
Quartile3StdDevOfNumericAtts	Third quartile of standard deviation of numeric attributes.	$\|x\|$
Quartile3KurtosisOfNumericAtts	Third quartile of kurtosis among numeric attributes.	$\|x+1,2\|$
Quartile3SkewnessOfNumericAtts	Third quartile of skewness among numeric attributes.	$\|x\|$

TABLE 4.7 – Méta-attributs généraux décrivant les attributs numériques

Méta-attribut	Description	$\delta^{\emptyset}(x)$
ClassEntropy	Entropy of the target attribute.	0
EquivalentNumberOfAtts	An estimate of the amount of useful attributes.	0
NoiseToSignalRatio	An estimate of the amount of non-useful information in the attributes regarding the class.	0
MeanAttributeEntropy	Mean of entropy among attributes.	$\|x\|$
MeanMutualInformation	Mean of mutual information between the nominal attributes and the target attribute.	$\|x\|$
MinAttributeEntropy	Min of entropy among attributes.	$\|x\|$
MinMutualInformation	Min of mutual information between the nominal attributes and the target attribute.	$\|x\|$
MaxAttributeEntropy	Max of entropy among attributes.	$\|x\|$
MaxMutualInformation	Max of mutual information between the nominal attributes and the target attribute.	$\|x\|$
Quartile1AttributeEntropy	First quartile of entropy among attributes.	$\|x\|$
Quartile1MutualInformation	First quartile of mutual information between the nominal attributes and the target attribute.	$\|x\|$
Quartile2AttributeEntropy	Second quartile of entropy among attributes.	$\|x\|$
Quartile2MutualInformation	Second quartile of mutual information between the nominal attributes and the target attribute.	$\|x\|$
Quartile3AttributeEntropy	Third quartile of entropy among attributes.	$\|x\|$
Quartile3MutualInformation	Third quartile of mutual information between the nominal attributes and the target attribute.	$\|x\|$
MaxNominalAttDistinctValues	The maximum number of distinct values among attributes of the nominal type.	$\|x\|$
MinNominalAttDistinctValues	The minimal number of distinct values among attributes of the nominal type.	$\|x\|$
MeanNominalAttDistinctValues	Mean of number of distinct values among the attributes of the nominal type.	$\|x\|$
StdvNominalAttDistinctValues	Standard deviation of the number of distinct values among nominal attributes.	$\|x\|$

TABLE 4.8 – Méta-attributs généraux décrivant les attributs nominaux

Méta-attribut	Description	$\delta^{\emptyset}(x)$
DecisionStumpAUC	Area Under ROC achieved by the landmarker weka.classifiers.trees.DecisionStump	$\|x - 0,5\|$
J48AUC	Area Under ROC achieved by the landmarker weka.classifiers.trees.J48	$\|x - 0,5\|$
JRipAUC	Area Under ROC achieved by the landmarker weka.classifiers.rules.Jrip	$\|x - 0,5\|$
kNN_3NAUC	Area Under ROC achieved by the landmarker weka.classifiers.lazy.IBk -K 3	$\|x - 0,5\|$
NaiveBayesAUC	Area Under ROC achieved by the landmarker weka.classifiers.bayes.NaiveBayes	$\|x - 0,5\|$
NBTreeAUC	Area Under ROC achieved by the landmarker weka.classifiers.trees.NBTree	$\|x - 0,5\|$
RandomTreeDepth3AUC	Area Under ROC achieved by the landmarker weka.classifiers.trees.RandomTree -depth 3	$\|x - 0,5\|$
REPTreeDepth3AUC	Area Under ROC achieved by the landmarker weka.classifiers.trees.REPTree -L 3	$\|x - 0,5\|$
SimpleLogisticAUC	Area Under ROC achieved by the landmarker weka.classifiers.functions.SimpleLogistic	$\|x - 0,5\|$

TABLE 4.9 – "Aire sous la courbe" des landmarkers

Méta-attribut	Description	$\delta^{\emptyset}(x)$
DecisionStumpErrRate	Error rate achieved by the landmarker weka.classifiers.trees.DecisionStump	$\|x - 1\|$
J48ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.J48	$\|x - 1\|$
JRipErrRate	Error rate achieved by the landmarker weka.classifiers.rules.Jrip	$\|x - 1\|$
kNN_3NErrRate	Error rate achieved by the landmarker weka.classifiers.lazy.IBk -K 3	$\|x - 1\|$
NaiveBayesErrRate	Error rate achieved by the landmarker weka.classifiers.bayes.NaiveBayes	$\|x - 1\|$
NBTreeErrRate	Error rate achieved by the landmarker weka.classifiers.trees.NBTree	$\|x - 1\|$
RandomTreeDepth3ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.RandomTree -depth 3	$\|x - 1\|$
REPTreeDepth3ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.REPTree -L 3	$\|x - 1\|$
SimpleLogisticErrRate	Error rate achieved by the landmarker weka.classifiers.functions.SimpleLogistic	$\|x - 1\|$

TABLE 4.10 – Taux d'erreur des landmarkers

Méta-attribut	Description	$\delta^{\emptyset}(x)$
DecisionStumpKappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.DecisionStump	$\|x\|$
J48Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.J48	$\|x\|$
JRipKappa	Kappa coefficient achieved by the landmarker weka.classifiers.rules.Jrip	$\|x\|$
kNN_3NKappa	Kappa coefficient achieved by the landmarker weka.classifiers.lazy.IBk -K 3	$\|x\|$
NaiveBayesKappa	Kappa coefficient achieved by the landmarker weka.classifiers.bayes.NaiveBayes	$\|x\|$
NBTreeKappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.NBTree	$\|x\|$
RandomTreeDepth3Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.RandomTree -depth 3	$\|x\|$
REPTreeDepth3Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.REPTree -L 3	$\|x\|$
SimpleLogisticKappa	Kappa coefficient achieved by the landmarker weka.classifiers.functions.SimpleLogistic	$\|x\|$

TABLE 4.11 – Kappa de Cohen des landmarkers

On forme alors trois ensembles de méta-attributs généraux :

DMFg_min	: Tables 4.6 - 4.7 - 4.8	Aucun *landmarker*.
DMFg_red	: Tables 4.6 - 4.7 - 4.8 - 4.9	+ AUC des *landmarkers*.
DMFg_full	: Tables 4.6 - 4.7 - 4.8 - 4.9 - 4.10 - 4.11	+ Tous les *landmarkers*.

4.4.2.2 Méta-attributs des attributs

On présente ici les différents méta-attributs retenus pour les attributs individuels de jeux de données. Ces derniers consistent en des propriétés simples communes à tous types d'attributs (Table 4.12), des propriétés exclusives aux attributs numériques (Table 4.14), des propriétés exclusives aux attributs nominaux (Table 4.13), et des versions normalisées de certaines des propriétés précédentes (Tables 4.15 et 4.16). Cette normalisation est proposée dans (Smid 2016), suite au constat que les distributions de certains méta-attributs sur un ensemble de jeux de données courants peuvent se révéler peu informatives. En effet, certains méta-attributs sont fortement corrélés à la taille (nombre d'instances) du jeu de données, comme par exemple le *nombre* de valeur manquantes. La solution proposée et d'ajouter une nouvelle version de ces méta-attributs, normalisée par le nombre d'instances. Ces méta-attributs normalisés sont présentés en Table 4.15 pour ceux indépendants du type d'attribut, et en Table 4.16 pour ceux exclusifs aux attributs numériques.

Méta-attribut	Description	$\delta^{\emptyset}(x)$
ValuesCount	Number of values.	$\|x-1\|$
NonMissingValuesCount	Number of non missing values.	$\|x\|$
MissingValuesCount	Number of missing values.	0
Distinct	Number of distinct values.	$\|x-1\|$
AverageClassCount	Average count of occurrences among different classes.	0
Entropy	Entropy of the values.	$\|x-1\|$
MostFequentClassCount	Count of the most probable class.	0
LeastFequentClassCount	Count of the least probable class.	0
ModeClassCount	Mode of the number of distinct values.	0
MedianClassCount	Median of the number of distinct values.	0
PearsonCorrellationCoefficient	Pearson Correlation Coeeficient of the values with the target attribute.	$\|x\|$
SpearmanCorrelationCoefficient	Spearman Correlation Coeeficient of the values with the target attribute.	$\|x\|$
CovarianceWithTarget	Covariance of the values with the target attribute.	$\|x\|$

TABLE 4.12 – Méta-attributs simples des attributs

Méta-attribut	Description	$\delta^{\emptyset}(x)$
UniformDiscrete	Result of Pearson's chi-squared test for discrete uniform distribution.	$\|x-1\|$
ChiSquare-UniformDistribution	Statistic value for the Pearson's chi-squared test.	$\|x\|$
RationOfDistinguishing-CategoriesByKolmogorov-SmirnoffSlashChiSquare	Ratio of attribute values that after sub-setting the dataset to that attribute value leads to different distribution of the target as indicated by the Kolmogorov-Smirnoff test.	0
RationOfDistinguishing-CategoriesByUtest	Ratio of attribute values that after sub-setting the dataset to that attribute value leads to different distribution of the target as indicated by the Mann-Whitney U-test.	0

TABLE 4.13 – Méta-attributs spécifiques aux attributs nominaux

Méta-attribut	Description	$\delta^{\emptyset}(x)$
IsUniform	Whether statistical test did or not reject that the attribute values corresponds to a uniform distribution.	$\|x-1\|$
IntegersOnly	Whether attribute values are only integers.	0
Min	Minimal value of the attribute values.	0
Max	Maximal value of the attribute values.	0
Kurtosis	Kurtosis of the values.	$\|x+1,2\|$
Mean	Mean of the values.	0
Skewness	Skewness of the values.	$\|x\|$
StandardDeviation	Standard deviation of the values.	$\|x\|$
Variance	Variance of the values.	$\|x\|$
Mode	Mode of the values.	0
Median	Median of the values.	0
ValueRange	Difference between maximum and minimum of the values.	$\|x\|$
LowerOuterFence	Lower outer fence of the values.	0
HigherOuterFence	Higher outer fence of the values.	0
LowerQuartile	Lower quartile of the values.	0
HigherQuartile	Higher quartile of the values.	0
HigherConfidence	Higher confidence interval of the values.	0
LowerConfidence	Lower confidence interval of the values.	0
PositiveCount	Number of positive values.	$\|x\|$
NegativeCount	Number of negative values.	$\|x\|$

TABLE 4.14 – Méta-attributs spécifiques aux attributs numériques

Méta-attribut	Description	$\delta^{\emptyset}(x)$
MissingValues	1 if count of missing values is greater than 0, 0 otherwise.	$\|x-1\|$
AveragePercentageOfClass	Percentage of the occurrences among classes.	$\|x-1\|$
PercentageOfMissing	Percentage of missing values in the attribute.	$\|x-1\|$
PercentageOfNonMissing	Percentage of non missing values in the attribute.	$\|x\|$
PercentageOfMostFrequentClass	Percentage of the most frequent class.	$\|x-1\|$
PercentageOfLeastFrequentClass	Percentage of the least frequent class.	$\|x-1\|$
ModeClassPercentage	Percentage of mode of class count calculated as ModeFrequentClassCount / ValuesCount.	$\|x-1\|$
MedianClassPercentage	Percentage of median of class count calculated as MedianFrequentClassCount / ValuesCount.	$\|x-1\|$

TABLE 4.15 – Méta-attributs normalisés selon le nombre d'instances

Méta-attribut	Description	$\delta^{\emptyset}(x)$
PositivePercentage	Percentage of positive values.	$\|x\|$
NegativePercentage	Percentage of negative values.	$\|x\|$
HasPositiveValues	1 if attribute has at least one positive value, 0 otherwise.	$\|x\|$
HasNegativeValues	1 if attribute has at least one negative value, 0 otherwise.	$\|x\|$

TABLE 4.16 – Méta-attributs normalisés spécifiques aux attributs numériques

On forme alors deux ensembles de méta-attributs des attributs :

DMFf_base : Tables 4.12 - 4.13 - 4.14 Pas de méta-attributs normalisés.

DMFf_full : Tables 4.12 - 4.13 - 4.14 - 4.15 - 4.16 Ajout des méta-attributs normalisés.

4.4.3 Fonctions de dissimilarité

On définira ici les différentes dissimilarités à comparer. Les éléments nécessaires à la définition d'une dissimilarité particulière sont, d'une part des ensembles de méta-attributs généraux et méta-attributs des attributs, tels que présentés dans la section précédente, et d'autre part des fonctions de dissimilarité sur ces méta-attributs généraux et méta-attributs des attributs. On présentera donc les fonctions qui seront considérées, avant de lister les dissimilarités ainsi formées pour comparaison.

4.4.3.1 Dissimilarités sur les méta-attributs généraux

Pour fournir une dissimilarité sur les méta-attributs généraux, on comparera le candidat présenté en définition 5, la dissimilarité normalisée par la borne supérieure d_G^{ubr}, à des distances classiques. On considérera un simple panel constitué des distances Euclidienne (norme 2), de Manhattan (norme 1), et de Tchebychev (norme infinie). On notera :

dissimG Dissimilarité normalisée par la borne supérieure

distEucl Distance Euclidienne

distMan Distance de Manhattan

distTcheb Distance de Tchebychev

4.4.3.2 Dissimilarités sur les attributs

Une dissimilarité sur les méta-attributs des attributs a été définie dans l'équation 4.3, se basant sur les différentes méthodes d'appariement des attributs décrites en section 4.2.2.3. Les dissimilarités construites selon ces différentes méthodes d'appariement peuvent alors être comparées entre elles.

D'autre part, des techniques existent dans le domaine du test statistique pour comparer directement des distributions. En identifiant un attribut de jeu de données à une distribution, on pourrait utiliser de telles techniques pour construire une dissimilarité entre attributs. On considère donc le test de Kolmogorov-Smirnov (Smirnov 1948), permettant de tester la significativité des différences entre deux échantillons de données. Ce dernier à l'avantage d'être non-paramétrique (aucun pré-requis sur les distributions comparées), ce qui nous permet de l'appliquer directement à tout attribut numérique. Afin de pouvoir l'appliquer également à des attributs nominaux, on considérera une simple association d'index entiers uniques aux catégories. On peut alors définir une nouvelle fonction de dissimilarité δ_{KS} sur les attributs de jeux de données comme la statistique résultante d'un test de Kolmogorov-Smirnov pour l'hypothèse nulle selon laquelle deux attributs proviennent d'une même distribution :

Définition 8 *Soient x et y deux jeux de données de ω. On note x_i le i^{th} attribut de x, et ω_a l'ensemble des attributs des jeux de données de ω. On note $KS(H_0)$ la statistique résultante d'un test de Kolmogorov-Smirnov pour l'hypothèse nulle H_0. On définit alors $\delta_{KS} : \omega_a^2 \mapsto \mathbb{R}^+$ telle que :*

$$\delta_{KS}(x_i, y_j) = KS(\text{"}x_i \text{ et } y_j \text{ sont issus de la même distribution"})$$

Proposition 4 *δ_{KS} est une fonction de dissimilarité normalisée.*

Preuve 3 *On doit montrer quatre propriétés : Positivité, Indiscernabilité des identiques, Symétrie et Normalisation (au sens de la Définition 4). La positivité, l'indiscernabilité des identiques, et la symétrie sont assurées par les propriétés de la statistique de Kolmogorov-Smirnov, qui, pour les échantillons x_i et y_j, est la borne supérieure de la différence absolue entre les fonctions de répartitions empirique de x_i et y_j.*

- *Une différence absolue est positive, donc $\delta_{KS}(x_i, y_j) \geq 0$.*
- *La différence absolue entre deux fonctions de répartitions identiques est toujours nulle, donc $\delta_{KS}(x_i, x_i) = 0$.*
- *Une différence absolue est symétrique, donc $\delta_{KS}(x_i, y_j) = \delta_{KS}(y_j, x_i)$.*

La distribution d'un attribut est un bon exemple d'ensemble atomique (on ne peut le diviser sans perte d'information). De plus, ω étant fini, δ_{KS} est nécessairement bornée sur ω. δ_{KS} est donc bien normalisée au sens de la Définition 4.

□

Afin de construire une fonction de dissimilarité sur les attributs à partir de δ_{KS}, on reprend l'équation 4.4. Soient donc $x, x' \in \omega$ ayant respectivement n et n' attributs. Étant donnée une fonction de mapping σ, on peut définir $d^{\sigma}_{KS(\omega)} : \omega^2 \mapsto \mathbb{R}^+$ telle que :

$$d^{\sigma}_{KS(\omega)}(x, x') = \frac{1}{max(n, n')}\Big(\sum_{\sigma_1(i)=j}^{i,j} \delta_{KS}(x_i, x'_j) + \sum_{\sigma_1(i)=\emptyset}^{i} \delta_{KS}(x_i, \emptyset) + \sum_{\sigma_2(j)=\emptyset}^{j} \delta_{KS}(\emptyset, x'_j)\Big) \quad (4.6)$$

δ_{KS} étant bien une fonction de dissimilarité normalisée, la preuve 2 tient toujours et assure que $d^{\sigma}_{KS(\omega)}$ est une fonction de dissimilarité normalisée sur les attributs des jeux de données de ω. δ_{KS} ne nécessitant pas à proprement parler de méta-attributs des attributs, on notera **DMFf_dist** l'ensemble des distributions des attributs.

On comparera donc deux dissimilarités sur les méta-attributs des attributs, d^{σ}_{F} et d^{σ}_{KS}, utilisant les différents *mappings* σ décrits en Table 4.3 : **greedyMix**, **exactMix**, **greedySplit** et **exactSplit**. On utilisera bien sûr d^{σ}_{KS} sur **DMFf_dist** et d^{σ}_{F} sur **DMFf_base** et **DMFf_full**.

4.4.3.3 Fonctions complètes comparées

On dispose donc de diverses variantes pour les différents éléments composant nos dissimilarités entre jeux de données. On présente en Table 4.17 (page 110) les dissimilarités ainsi formées qui seront comparées dans les expériences suivantes.

Les neuf premières, simples distances sur méta-attributs généraux, auront un rôle de baseline, tandis que les trente-six autres nous permettront d'étudier les interactions potentiellement complexes entre leurs différents éléments.

4.4.4 Cadre d'expérimentation

4.4.4.1 MetaDataset

Afin d'instancier le méta-problème décrit en section 4.4.1, on doit construire un *metadataset* décrivant la performance d'un ensemble de classifieurs sur un ensemble de jeux de données. On raffine pour cela la procédure employée dans la preuve de concept, pour construire ce *metadataset* depuis les données d'OpenML.

On se limite tout d'abord à quatre critères de performances bien distincts. En effet, les résultats de la preuve de concept ont montré que plusieurs critères de performance

Méta-attributs généraux	Dissimilarité sur les méta-attributs généraux	Méta-attributs des attributs	Dissimilarité sur les attributs
DMFg_full	distEucl	none	none
DMFg_full	distMan	none	none
DMFg_full	distTcheb	none	none
DMFg_red	distEucl	none	none
DMFg_red	distMan	none	none
DMFg_red	distTcheb	none	none
DMFg_min	distEucl	none	none
DMFg_min	distMan	none	none
DMFg_min	distTcheb	none	none
DMFg_full	dissimG	DMFf_full	greedyMix
DMFg_full	dissimG	DMFf_full	exactMix
DMFg_full	dissimG	DMFf_full	greedySplit
DMFg_full	dissimG	DMFf_full	exactSplit
DMFg_full	dissimG	DMFf_base	greedyMix
DMFg_full	dissimG	DMFf_base	exactMix
DMFg_full	dissimG	DMFf_base	greedySplit
DMFg_full	dissimG	DMFf_base	exactSplit
DMFg_full	dissimG	DMFf_dist	greedyMix
DMFg_full	dissimG	DMFf_dist	exactMix
DMFg_full	dissimG	DMFf_dist	greedySplit
DMFg_full	dissimG	DMFf_dist	exactSplit
DMFg_red	dissimG	DMFf_full	greedyMix
DMFg_red	dissimG	DMFf_full	exactMix
DMFg_red	dissimG	DMFf_full	greedySplit
DMFg_red	dissimG	DMFf_full	exactSplit
DMFg_red	dissimG	DMFf_base	greedyMix
DMFg_red	dissimG	DMFf_base	exactMix
DMFg_red	dissimG	DMFf_base	greedySplit
DMFg_red	dissimG	DMFf_base	exactSplit
DMFg_red	dissimG	DMFf_dist	greedyMix
DMFg_red	dissimG	DMFf_dist	exactMix
DMFg_red	dissimG	DMFf_dist	greedySplit
DMFg_red	dissimG	DMFf_dist	exactSplit
DMFg_min	dissimG	DMFf_full	greedyMix
DMFg_min	dissimG	DMFf_full	exactMix
DMFg_min	dissimG	DMFf_full	greedySplit
DMFg_min	dissimG	DMFf_full	exactSplit
DMFg_min	dissimG	DMFf_base	greedyMix
DMFg_min	dissimG	DMFf_base	exactMix
DMFg_min	dissimG	DMFf_base	greedySplit
DMFg_min	dissimG	DMFf_base	exactSplit
DMFg_min	dissimG	DMFf_dist	greedyMix
DMFg_min	dissimG	DMFf_dist	exactMix
DMFg_min	dissimG	DMFf_dist	greedySplit
DMFg_min	dissimG	DMFf_dist	exactSplit

TABLE 4.17 – Fonctions de dissimilarité comparées

menaient à des résultats très corrélés. Réduire le nombre de critères permet donc de limiter la complexité de l'expérience sans trop impacter la valeur des résultats. Les critères retenus sont présentés en Table 4.18.

Critère	Description
area_under_roc_curve	The area under the ROC curve (AUROC), calculated using the Mann-Whitney U-test.
predictive_accuracy	The Predictive Accuracy is the percentage of instances that are classified correctly.
kappa	Cohen's kappa coefficient is a statistical measure of agreement for qualitative (categorical) items : it measures the agreement of prediction with the true class.
kb_relative_information_score	The Kononenko and Bratko Information score, divided by the prior entropy of the class distribution, measures the information produced by the model.

TABLE 4.18 – Critères de performance retenus

On répète ensuite la procédure de recherche de cliques décrite dans le chapitre 3 pour trouver des ensembles de classifieurs et de jeux de données tels que chaque classifieur ait été évalué sur chaque jeu de données selon nos quatre critères choisis. Dans cette itération, on se limitera à des jeux de données d'au plus cent attributs, afin là-encore de limiter la complexité de l'expérience (dont un facteur de complexité déterminant est celui des méthodes d'appariement des attributs). Les ensembles ainsi produits, de respectivement 48 classifieurs et 395 jeux de données, sont présentés en annexe, Tables A.6 et A.7. Le *metadataset* complet est disponible au format *ARFF* en ligne [2].

4.4.4.2 Baseline

Les dissimilarités à comparer au méta-niveau ont été présentées en Table 4.17 et seront évaluées selon le protocole décrit par l'algorithme 3. Une première partie de notre *baseline* est constituée des distances classiques qui y sont présentées, mais une comparaison à des méthodes d'apprentissage traditionnelles reste souhaitable. On introduit donc une légère variante du méta-problème permettant d'évaluer des algorithmes d'apprentissage traditionnels dans des circonstances semblables, venant ainsi les ajouter à notre *baseline*. Ce protocole, présenté dans l'algorithme 4, permettra de comparer nos approches utilisant des dissimilarités à des algorithmes d'apprentissage de l'état de l'art. Les algorithmes sélectionnés sont décrits en Table 4.19.

2. `https://github.com/WilliamRaynaut/Dataset_dissimilarities`

Algorithme 4 : Méta-problème résolu par un algorithme d'apprentissage traditionnel $\mathcal{A}$

Data :
- Un metadataset $\mathcal{M}$ décrivant la performance de x classifieurs sur y jeux de données
- Un nouveau jeu de données D

Result : La recommandation d'un ensemble de n algorithmes $c_1...c_n$ de $\mathcal{M}$ supposés capables de bonnes performances sur D, et leur intérêt relatif attendu $\alpha_1...\alpha_n$

foreach *Classifieur c de* $\mathcal{M}$ **do**
  Construire avec $\mathcal{A}$ un modèle de la performance de c à partir de $\mathcal{M}$
  Prédire la performance de c sur D selon ce modèle.

Ordonner les classifieurs du metadataset selon l'estimation de leur performance sur D

Recommander les n meilleurs classifieurs, pondérés selon leur performance estimée

Implémentation Weka	Description
GaussianProcesses	Gaussian Processes for regression without hyperparameter-tuning. See (MacKay 1998).
LinearRegression	Linear regression for prediction. Uses the Akaike criterion for model selection, and is able to deal with weighted instances. See (Akaike 1974).
RBFRegressor	Radial basis function networks, trained in a fully supervised manner by minimizing squared error with the BFGS method. See (E. Frank 2014).
SMOreg	Sequential minimal optimization algorithm for training a support vector regression model. See (Smola et Schölkopf 2004).
RandomForest	Ensemble of decision trees outputting the mean prediction of the individual trees. See (Breiman 2001).
KStar	Instance-based classifier where the class of a test instance is based upon the class of those training instances similar to it, as determined by an entropy-based distance function. See (Cleary, Trigg et al. 1995).
M5Rules	Generates a decision list for regression problems using separate-and-conquer. Builds a model tree using M5 In each iteration and makes the best leaf into a rule. See (Holmes et al. 1999).

TABLE 4.19 – Algorithmes d'apprentissage de la *baseline*

4.4.4.3 Exécutions

On évalue donc nos algorithmes d'apprentissage (Table 4.19) et dissimilarités (Table 4.17) sur le metadataset, respectivement selon les algorithmes 4 et 3. On explore ainsi l'espace formé sur les dimensions décrites en Table 4.20

Dimension	Détails	Taille
Algorithme au méta-niveau	Traditionnels (Table 4.19)	7
	Dissimilarités (Table 4.17)	270
Critère de performance de base	Table 4.18	4
Jeu de données (instance de base)	Table A.7	395
Nombre n d'algorithmes de base recommandés	$\{1, 3, 5\}$	3

TABLE 4.20 – Dimensions de l'expérience

Les dissimilarités apparaissent plus nombreuses qu'en Table 4.17 car leur nombre est multiplié par les dimensions présentées dans la définition du méta-problème. En effet, le nombre k de voisins considérés dans l'algorithme 3 et la méthode *nnDist* d'estimation de performance d'un classifieur à partir de celle de ses voisins, sont des dimensions internes des dissimilarités. Les dimensions de l'espace des dissimilarités sont présentées en Table 4.21.

Dimension	Détails	Taille
Méta-attributs généraux	Table 4.17	3
Dissimilarité sur les méta-attributs généraux	Table 4.17	4
Méta-attributs des attributs	Table 4.17	3
Dissimilarité sur les attributs	Table 4.17	4
Nombre k de voisins considérés	$\{3, 5, 10\}$	3
Méthode *nnDist* d'estimation de performance d'un classifieur à partir de celle de ses voisins	{ *mean, weighted* }	2

TABLE 4.21 – Dimensions de l'espace des dissimilarités

L'expérience complète nécessite donc 33.180 exécutions de l'algorithme 4 et 1.279.800 exécutions de l'algorithme 3 pour évaluer la performance au méta-niveau en chaque point de l'espace. Ceci nécessite un important degré de parallélisme pour être calculé en un temps raisonnable, encore une fois obtenu en générant dynamiquement les exécutions et en les déléguant à un répartiteur de tâches SLURM (Yoo et al. 2003) gérant les 640 nœuds du cluster *OSIRIM*[3]. De plus, un pré-calcul des dissimilarités a été effectué afin de minimiser la redondance entre les exécutions. Ce pré-calcul a de plus permis de mesurer le temps de calcul des différentes dissimilarités[4], présenté en Figure 4.11.

3. voir `osirim.irit.fr`

4. Des optimisations postérieures ont permis d'accélérer le calcul de dissimilarité d'un facteur 500 à 1000. Les temps présentés ici ont valeur comparative entre les différentes dissimilarités.

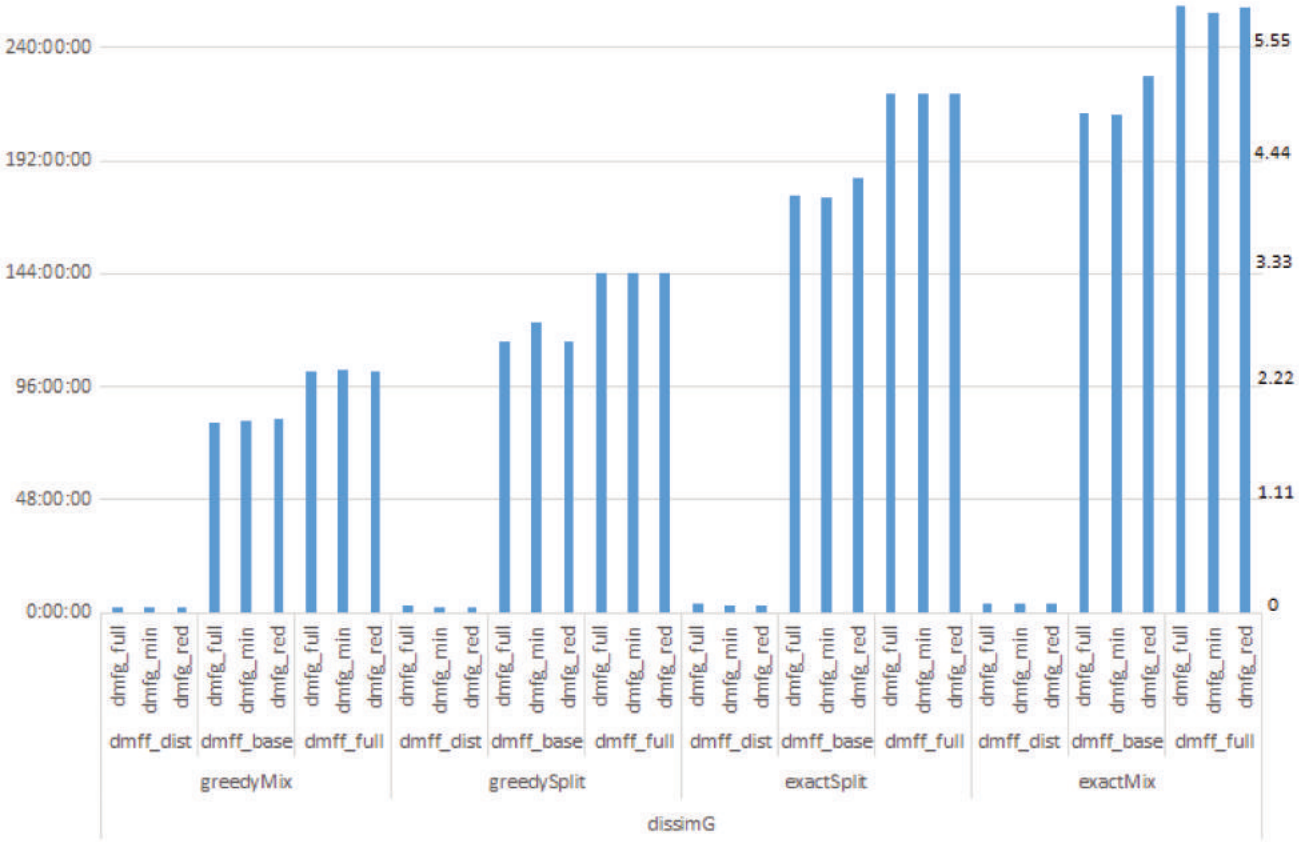

FIGURE 4.11 – Temps de calcul des dissimilarités. Total en heures à gauche, et moyenne en secondes à droite.

Ces temps d'exécution sont globalement compatibles avec les complexités attendues [5], montrant une forte dépendance envers la méthode d'appariement des attributs, et le nombre de méta-attributs des attributs. La dissimilarité sur les attributs basée sur le test de Kolmogorov présente des temps d'exécution bien moindres (environ un à trois pour-cents) que celle basée sur les méta-attributs, pour les ensembles étudiés ici (37 à 49 méta-attributs).

4.4.5 Analyse dimensionnelle des résultats

Les 1.312.980 exécutions détaillées dans la section précédente résultent en autant de valeurs de performance au méta-niveau. Ces résultats sont stockés dans une base de données SQL, dont les mécanismes sont propices à l'analyse dimensionnelle. On étudiera ainsi l'influence individuelle des différentes dimensions sur les performances au méta-niveau pour tenter d'y déceler des tendances, qui devront ensuite être soumises à des tests d'hy-

5. Des analyses ultérieures ont montré que la majeure partie des temps d'exécution était due à un goulot d'étranglement lors des accès base de données. Les temps réels seront ainsi bien moindre, mais les comparaisons relatives établies ici restent valides.

pothèse statistique pour validation. On utilisera en particulier le test de Friedman sous l'hypothèse nulle d'identité des distributions pour valider l'existence d'une tendance, et le test de Nemenyi pour en établir le sens. Cette procédure, décrite au chapitre 3, ne sera complètement développée que dans la première analyse de la sous-section suivante, où l'on étudie l'impact du nombre k de voisins considérés par l'algorithme 3. Les analyses suivantes présenteront toujours l'intégralité du raisonnement suivi, mais le protocole des tests statistiques n'y sera développé que s'il diffère du standard présenté.

4.4.5.1 Facteurs secondaires

On qualifie de secondaires les facteurs pouvant influer sur la performance au méta-niveau mais ne faisant pas partie intégrante des dissimilarités. On étudie ici leur impact sur la performance au méta-niveau et leurs relations avec les différentes dissimilarités.

Nombre de voisins considérés L'algorithme 3 estime la performance des classifieurs sur un nouveau jeu de données D selon leur performance sur les k plus proches voisins de D, au sens de la dissimilarité évaluée. Ce nombre k prend ici des valeurs communes pour des ensembles de l'ordre de la centaine (Batista et Silva 2009) : $k \in \{3, 5, 10\}$. Pour étudier l'impact de ce facteur k, on observe le comportement des différentes dissimilarités pour chaque valeur de k, en moyennant selon les autres dimensions. Cette performance moyenne au méta-niveau est présentée en Figure 4.12.

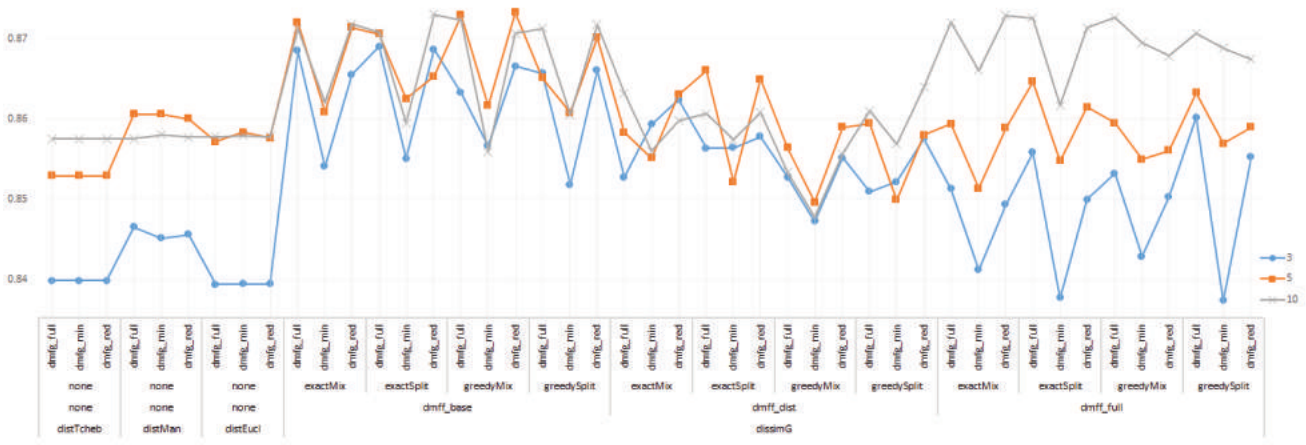

FIGURE 4.12 – Moyenne des performances au méta-niveau selon le nombre k de voisins considérés.

On rappelle que ces valeurs de performance moyenne sont un pourcentage du maximum connu : par exemple, la dissimilarité *dissimG - DMFg_full - greedySplit - DMFf_full* affiche une performance moyenne de 0.87 pour $k = 10$, ce qui signifie que les exécutions

de l'algorithme 3 sur cette dissimilarité avec $k = 10$ ont permis de trouver des classifieurs en moyenne 87% aussi performant que le meilleur. En observant la Figure 4.12, on peut conjecturer plusieurs tendances :

1. La performance au méta-niveau pour $k = 3$ semble *souvent* inférieure à celle obtenue pour $k = 5$ et $k = 10$.
2. Pour les dissimilarités utilisant *DMFf_full*, les performances au méta-niveau apparaissent toujours croissantes de $k = 3$ à $k = 5$ puis $k = 10$.

Afin de contrôler le risque que ces tendances ne reflètent pas de réelles différences entre nos distributions, on fait appel aux tests d'hypothèse statistique de Friedman et Nemenyi. Le test de Friedman est un test non paramétrique, ne posant aucune condition sur la forme des distributions sous-jacentes, ce qui est nécessaire dans ce contexte multidimensionnel où aucune distribution n'est connue. Il permet de comparer des échantillons de valeurs dans le but d'assurer l'improbabilité de l'hypothèse nulle $H_0 = \{$ Les différents échantillons sont tirés de la même distribution $\}$. La *p-value* retournée par le test de Friedman (apparaissant dans les figures type 4.13) mesure ainsi la probabilité de l'observation faîte sous H_0. Ici, cela signifie que si H_0 est vrai ($\leftrightarrow$ si le facteur k n'a pas de réelle influence sur la performance), alors la probabilité d'observer les distributions en Figure 4.12 était de $9.2496 * 10^{-13}$ (*p-value* en Figure 4.13). Ceci nous assure que H_0 est hautement improbable : le facteur k a bien une influence sur la performance, mais cela ne suffit pas à valider l'existence des tendances relevées plus haut.

Pour ce faire, on fait appel au test *post-hoc* de Nemenyi. Ce dernier permet de comparer les échantillons deux à deux, déterminant lesquels sont significativement différents, tout en contrôlant le risque global d'erreur type 1. En effet, comparer de nombreux échantillons rend exponentiel le risque de commettre au moins une erreur de type 1 (de valider une différence infondée). Le test de Nemenyi permet de maitriser ce risque quel que soit le nombre d'échantillons. Dans nos expériences, on contraindra ce risque à la valeur courante de 0.05, ce qui signifie que chaque test de Nemenyi effectué a un risque d'au plus 5% de discriminer deux échantillons qui n'étaient pas réellement discernables. Le test résulte ainsi en une *différence critique* CD, représentant la différence nécessaire entre le *rang moyen* de deux échantillons (statistique produite par le test de Friedman) pour pouvoir les considérer significativement différents. Ceci est représenté en Figure 4.13, où l'on classe les différents échantillons par rang moyen. Ceux trop proches pour pouvoir être considérés significativement différents sont liés entre eux. Ici, cela signifie que l'échantillon $k = 3$ est

significativement moins bon que les échantillons $k = 10$ et $k = 5$, mais que ces derniers ne sont pas suffisamment éloignés pour être jugés significativement différents (sans courir un risque d'erreur de plus de 5%). La valeur indiquée à côté de l'identifiant de l'échantillon est sa performance moyenne, qui permet de constater les écarts de moyenne parfois très faibles entre échantillons jugés différents.

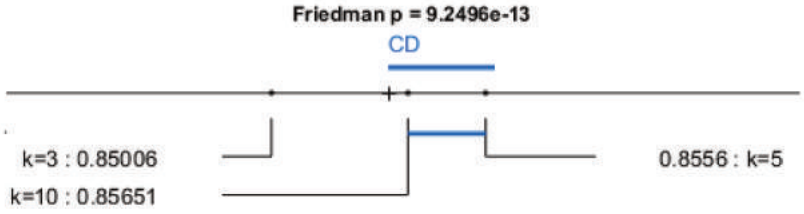

FIGURE 4.13 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes valeurs de k.

Les résultats du test de Nemenyi en Figure 4.13 valident donc notre première tendance : choisir $k = 3$ mène à des performances significativement inférieures. Pour valider la seconde, on répète les tests de Friedman et Nemenyi en se limitant cette fois aux dissimilarités utilisant *DMFf_full*. Les résultats, présentés en Figure 4.14, montrent bien que $k = 10$ y est significativement meilleur que $k = 5$, lui-même significativement meilleur que $k = 3$.

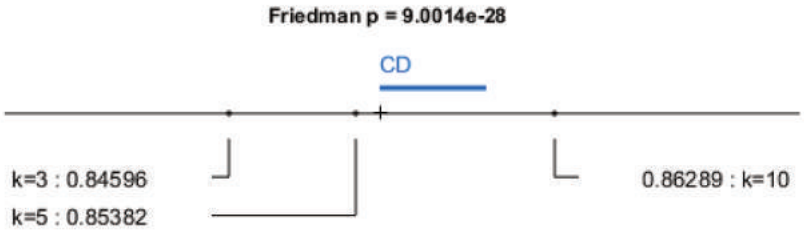

FIGURE 4.14 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes valeurs de k, pour les dissimilarités utilisant *DMFf_full*.

Ces différents résultats nous permettent d'écarter $k = 3$, et pointe vers l'utilisation de $k = 10$, en particulier en conjonction avec *DMFf_full*. Ceci pourrait indiquer la nécessité de considérer davantage de voisins si l'on souhaite exploiter de grands ensembles de méta-attributs des attributs.

Cette méthodologie de test, par étude des distributions puis validation par Friedman et Nemenyi, guidera ainsi notre étude des résultats, et sera reproduite selon nos différentes dimensions pour évaluer l'impact de chaque facteur identifié plus tôt.

Méthode d'estimation de performance d'un classifieur selon celle de ses voisins Pour estimer la performance d'un classifieur sur un nouveau jeu de données D, l'algorithme 3 peut utiliser les deux fonctions proposées : soit **"mean"**, la performance moyenne du classifieur sur les k plus proches voisins de D, soit **"weighted"**, moyenne cette fois pondérée par la dissimilarité entre D et ses voisins. On présente en Figure 4.15 (page 119) la moyenne de performance au méta-niveau des dissimilarités selon la méthode d'estimation utilisée.

On remarque une importante similitude entre ces courbes, avec une différence quasi-constante à l'avantage de la méthode **"weighted"**. Comme précédemment on valide cette tendance par un test de Friedman avec post-hoc Nemenyi en Figure 4.16.

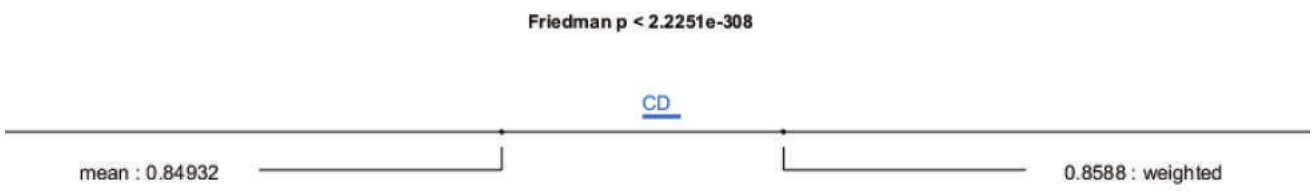

FIGURE 4.16 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes méthodes d'estimation.

Le test valide bien la supériorité globale de la méthode **"weighted"**. Ce résultat conforte l'intérêt perçu des dissimilarités, et leur utilité pour l'estimation de performance au niveau de base.

Critères de performance de base Notre critère de performance au méta-niveau se définit par rapport à un critère de performance de base que l'on cherche à optimiser par le travail au méta-niveau. Dans cette expérience, le problème de base est celui de la classification supervisée, pour lequel on a retenu quatre critères de performance à la sémantique distincte, décrits précédemment en Table 4.18. On présente en Figure 4.17 (page 120) la moyenne de performance au méta-niveau des dissimilarités et de la *baseline* selon le critère de performance de base utilisé (la *baseline* apparait à gauche, suivie des différentes dissimilarités).

La performance au méta-niveau semble donc meilleure pour optimiser l'aire sous la courbe de *ROC* que pour le *kappa* de Cohen, lui-même meilleur que la précision et l'*information score* de Kononenko et Bratko. On ne peut pas réellement discriminer entre ces deux derniers, mais l'*information score* semble devancer la précision pour les dissimi-

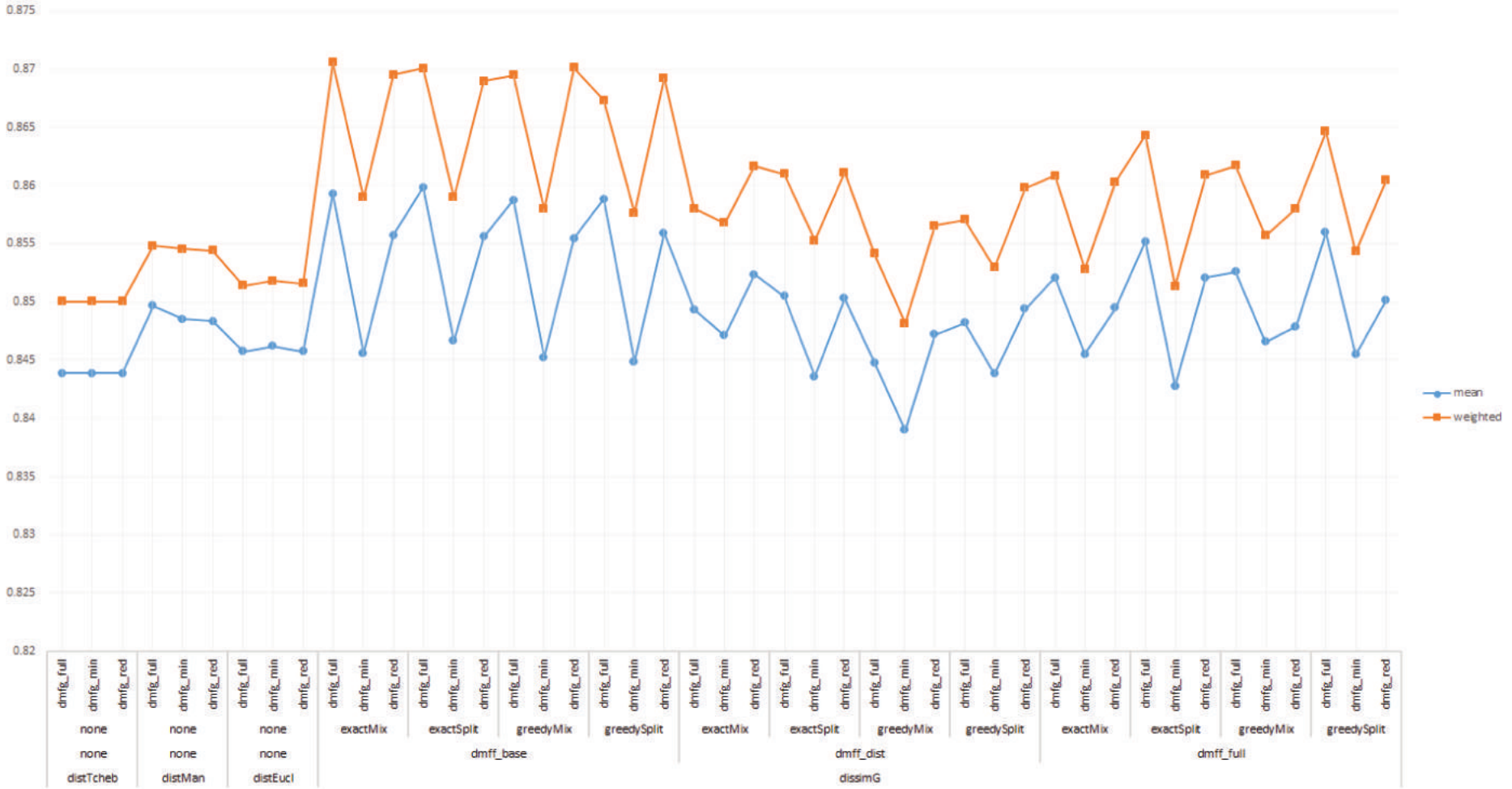

FIGURE 4.15 – Moyenne des performances au méta-niveau selon la méthode d'estimation utilisée.

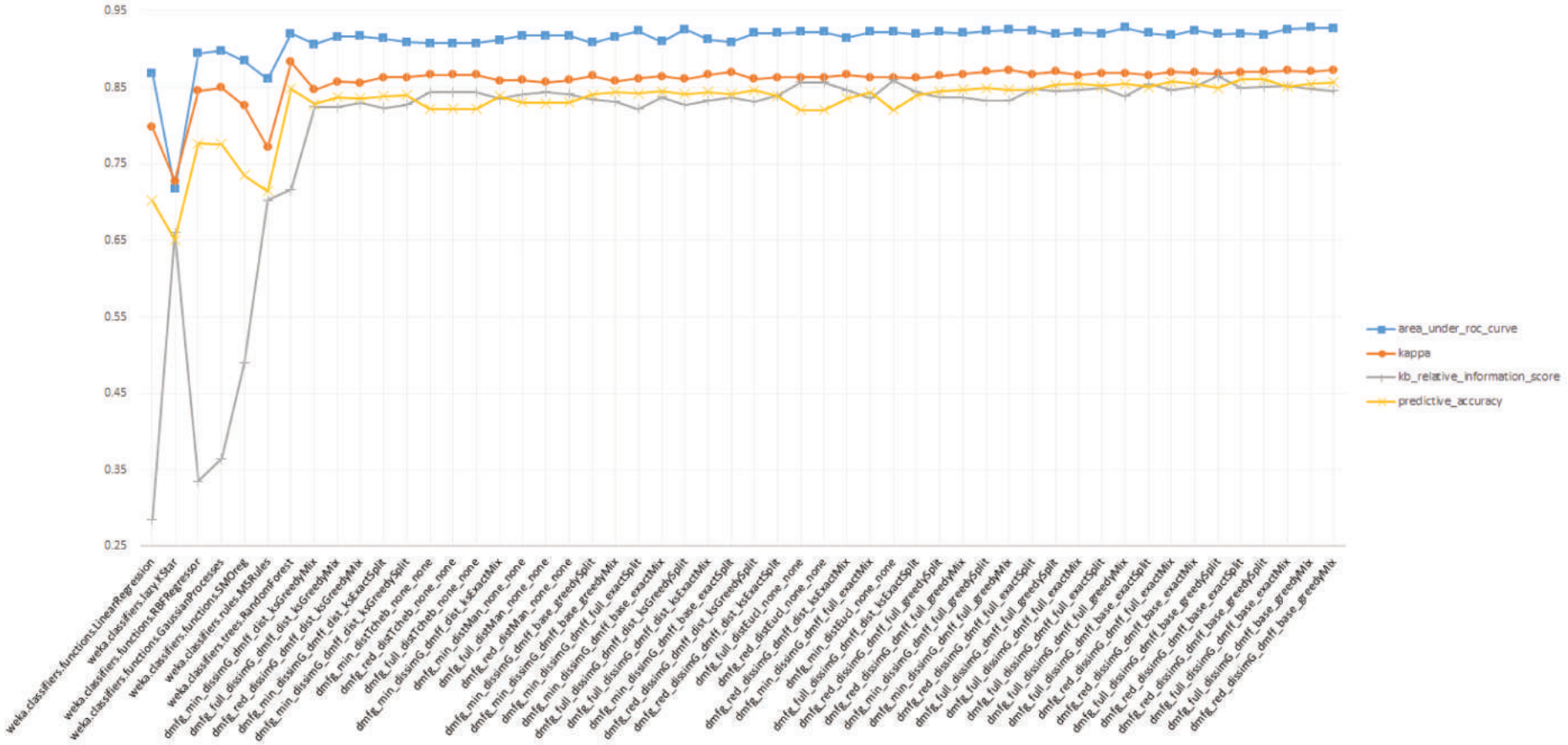

FIGURE 4.17 – Moyenne des performances au méta-niveau selon le critère de performance de base utilisé.

larités utilisant de simples distances. On vérifie ces tendances respectivement en Figures 4.18 et 4.19.

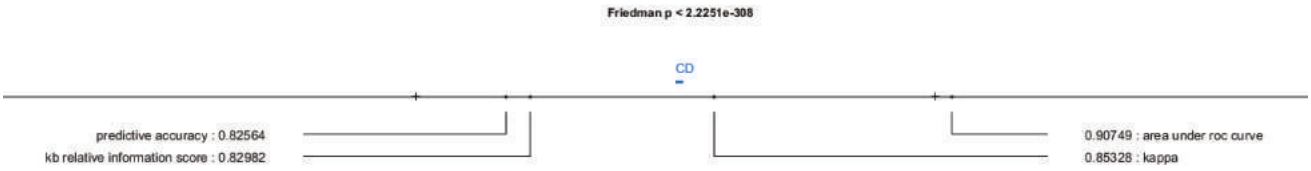

FIGURE 4.18 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents critères de performance de base.

FIGURE 4.19 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents critères de performance de base, pour les dissimilarités utilisant des distances simples.

Les Figures 4.18 et 4.19 nous assurent donc de l'ordonnancement global des critères par les performances au méta-niveau qu'ils induisent, en effet plus marqué pour les dissimilarités utilisant des distances simples.

Une étude plus approfondie de la relation critère - performance se concentrant sur les dissimilarités semble faire apparaître une inversion de tendance entre l'*information score* et la précision lorsque l'on utilise *DMFg_full* (voir Figure 4.20 en page 122). On tente de valider cette tendance en Figure 4.21.

FIGURE 4.21 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents critères de performance de base, pour les dissimilarités utilisant *DMFg_full*.

Le test de Nemenyi en Figure 4.21 ne parvient pas à déceler d'inversion de tendance. L'ordonnancement en performance induite des différents critères semble donc valide dans une portion significative des cas de test, et aucun contre-exemple significatif n'a pu être trouvé. Tous nos éléments pointent donc vers l'aire sous la courbe de ROC, dont l'utilisa-

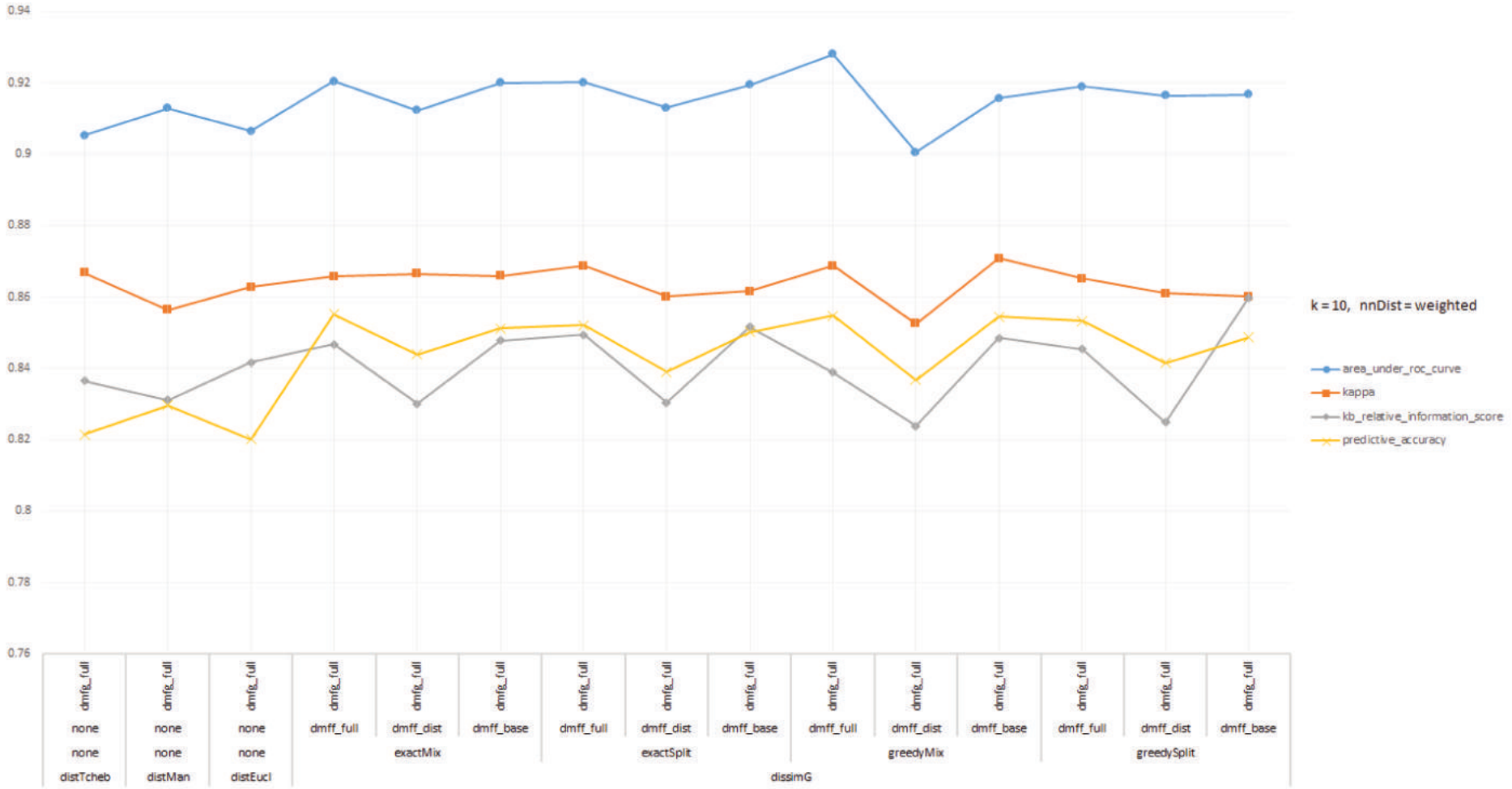

FIGURE 4.20 – Moyenne des performances au méta-niveau selon le critère de performance de base utilisé, pour les dissimilarités utilisant *DMFg_full*.

tion comme critère de performance de base semble mener aux meilleures performances au méta-niveau, et qui est généralement reconnu comme un bon critère de performance en classification.

Nombre de recommandations Comme indiqué précédemment, le nombre n de recommandations effectuées par les algorithmes 3 et 4 prend ses valeurs dans l'ensemble $\{1, 3, 5\}$. L'introduction de ce facteur avait pour objectif de "lisser" les résultats (réduire leur variance). On présente tout d'abord en Figure 4.22 (page 124) la moyenne de performance au méta-niveau des dissimilarités et de la *baseline* selon le nombre de recommandations effectuées.

Aucune tendance ne se distingue sur les algorithmes de la *baseline*, mais sur les dissimilarités, il semble qu'augmenter le nombre de recommandations impacte négativement les performances. On vérifie cela en Figure 4.23.

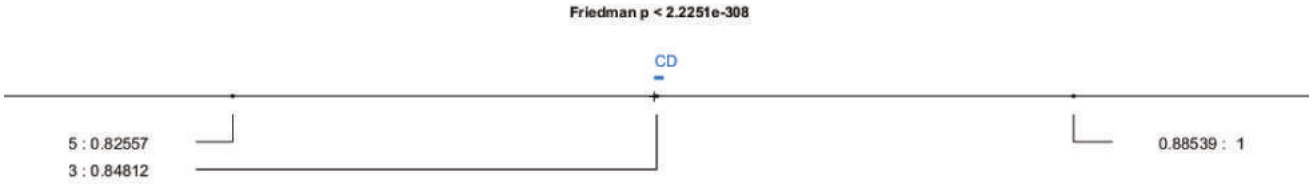

FIGURE 4.23 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes valeurs du nombre de recommandations.

Augmenter le nombre de recommandations impacte donc bien négativement les performances au méta-niveau. Ce compromis était attendu, le but étant de contrôler la variance des résultats. On étudie donc en Figure 4.24 la variance des performances au méta-niveau selon le nombre de recommandations.

On observe donc les variances les plus basses sur le critère d'AUC, sans que le nombre de recommandations ne semble l'impacter. Le seul critère où la multiplication des recommandations a l'effet escompté et limite la variance, est celui de précision, mais les valeurs y restent peu intéressantes. Cette étude de variance conforte donc notre choix du critère d'aire sous la courbe de *ROC*, mais rend apparent le peu d'intérêt de la multiplication des recommandations, du point de vue des performances. D'autres contraintes du domaine de la recommandation peuvent justifier la multiplication des recommandations, mais la qualité des solutions trouvées semble suffisamment élevée pour que cela entraine une perte significative de performance. Ce *tradeoff* sera bien sûr à considérer, mais dépasse le cadre de la présente étude.

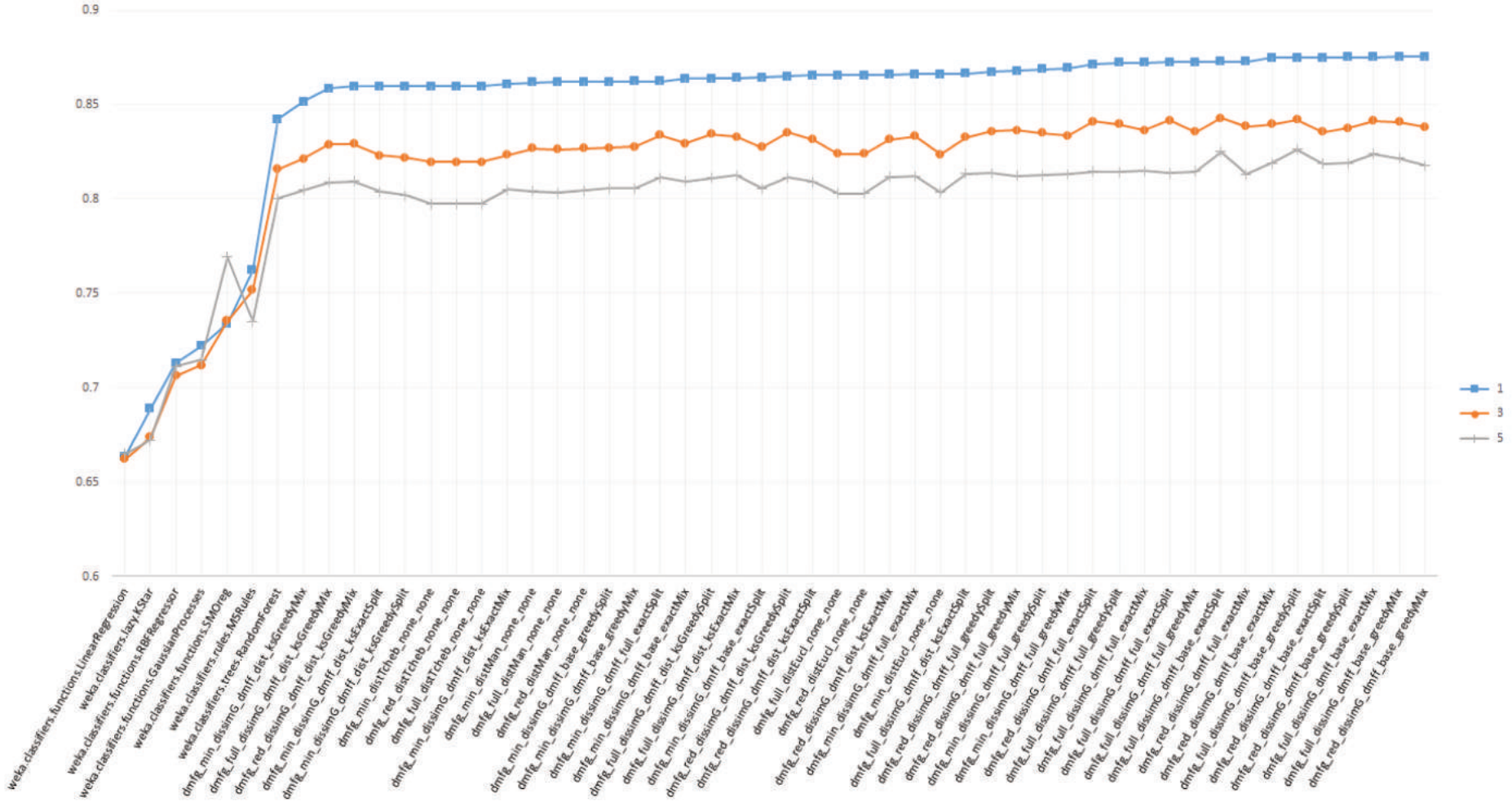

FIGURE 4.22 – Moyenne des performances au méta-niveau selon le nombre de recommandations.

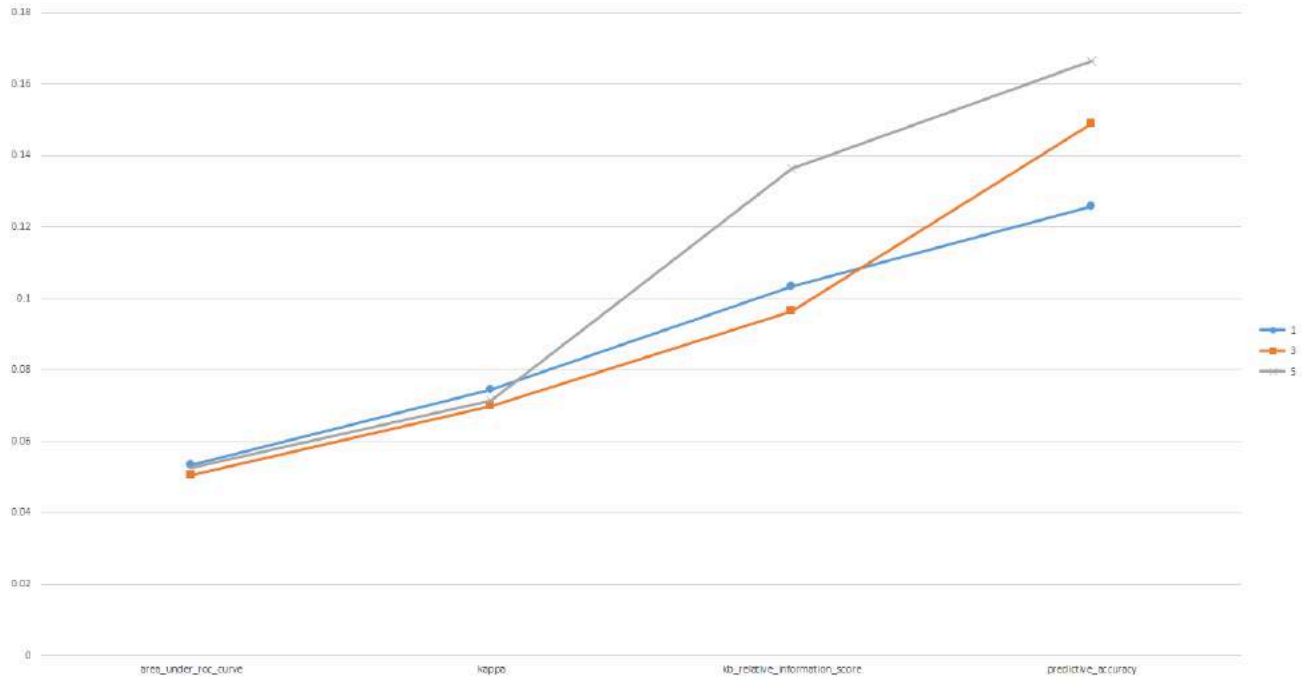

FIGURE 4.24 – Variance des performances au méta-niveau par critère selon le nombre de recommandations.

Espace optimal Ces études préliminaires nous ont permis de gagner une meilleure compréhension de l'impact des facteurs secondaires, et en particulier de déterminer un "espace optimal", ensemble de valeurs des facteurs secondaires maximisant la performance au méta-niveau. Dans certaines des observations suivantes, on se placera dans cet espace optimal pour analyser le comportement des dissimilarités dans ce qui serait le plus proche d'un futur cas d'application réel. Comme espace optimal, on retient donc les valeurs de $k = 10$, nnDist=*weighted*, criterion=AUC, et $n = 1$. D'autre part, on prendra toujours la performance au méta-niveau pour $n = 1$ (sauf mention contraire explicite).

4.4.5.2 Facteurs primaires

On qualifie de facteurs primaires les éléments fonctionnels variables des dissimilarités. On étudiera ici leur impact sur les performances au méta-niveau et les potentielles interactions entre eux.

Méta-attributs généraux Nos ensembles de méta-attributs généraux diffèrent par le nombre de méta-attributs de *landmarking* employés. *DMFg_full* en contient 27 (pour 62 autres méta-attributs), *DMFg_red* seulement 9, et *DMFg_min* aucun. On présente en Figure 4.25 (page 127) les performances moyennes au méta-niveau pour les dissimilarités

utilisant ces ensembles, selon leurs autres facteurs primaires. L'intérêt des *landmarkers* y semble avéré, et on le vérifie en Figure 4.26.

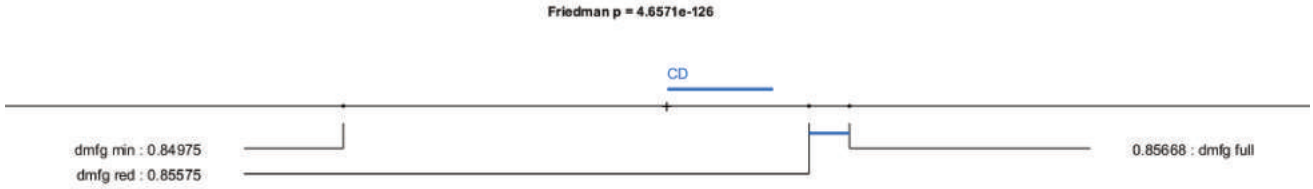

FIGURE 4.26 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensembles de méta-attributs généraux utilisés.

DMFg_full et *DMFg_red* distancent significativement *DMFg_min*, validant l'intérêt des méta-attributs de *landmarking*, mais aucune différence significative n'est établie entre eux. On peut rejoindre ici des travaux de sélection d'attributs insistant sur l'importance de la diversité au sein des attributs (Brown et al. 2012). Il pourrait ainsi être intéressant d'étudier l'impact de la *diversité* des *landmarkers* et autres méta-attributs généraux choisis sur la performance au méta-niveau.

Si l'on restreint l'étude à l'espace optimal (Figure 4.27 en page 128), on ne décèle plus de tendance particulière. De même, le test statistique (Figure 4.28) révèle uniquement la domination de *DMFg_red* sur *DMFg_min*, insistant sur l'intérêt de la présence de *landmarkers*, nonobstant leur nombre.

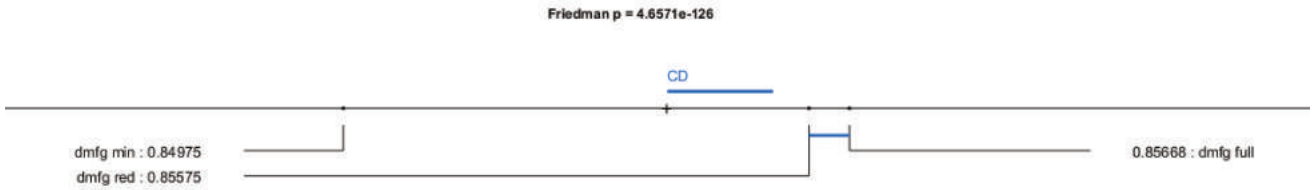

FIGURE 4.28 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensembles de méta-attributs généraux utilisés, sur l'espace optimal.

Dissimilarité sur les méta-attributs généraux Les dissimilarités sur les méta-attributs généraux considérées sont un panel de distances adjoint de la dissimilarité normalisée par la borne supérieure. On observe donc en Figure 4.29 (page 129) comment se comporte la dissimilarité normalisée par la borne supérieure par rapport aux distances classiques dans notre espace optimal.

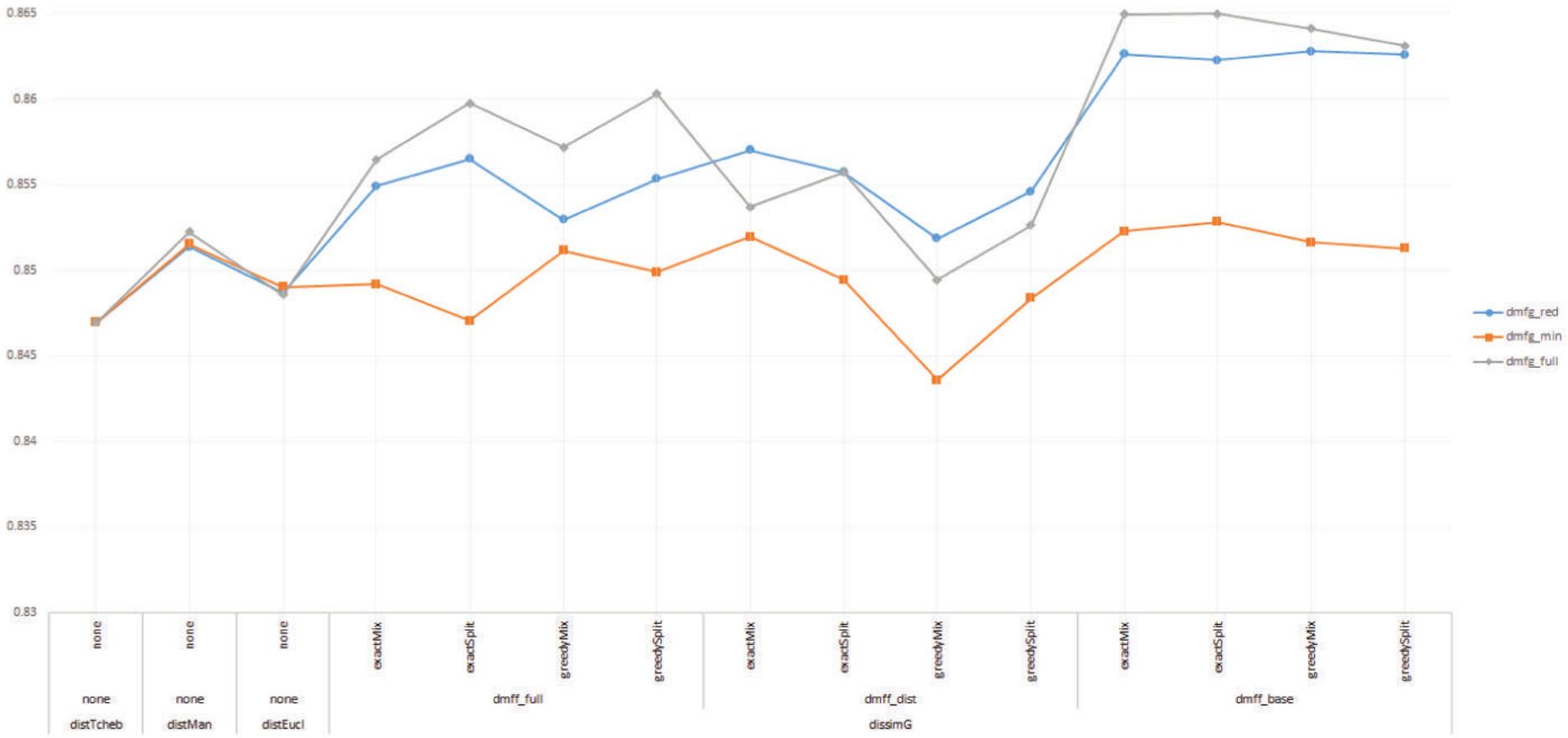

FIGURE 4.25 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs généraux utilisé.

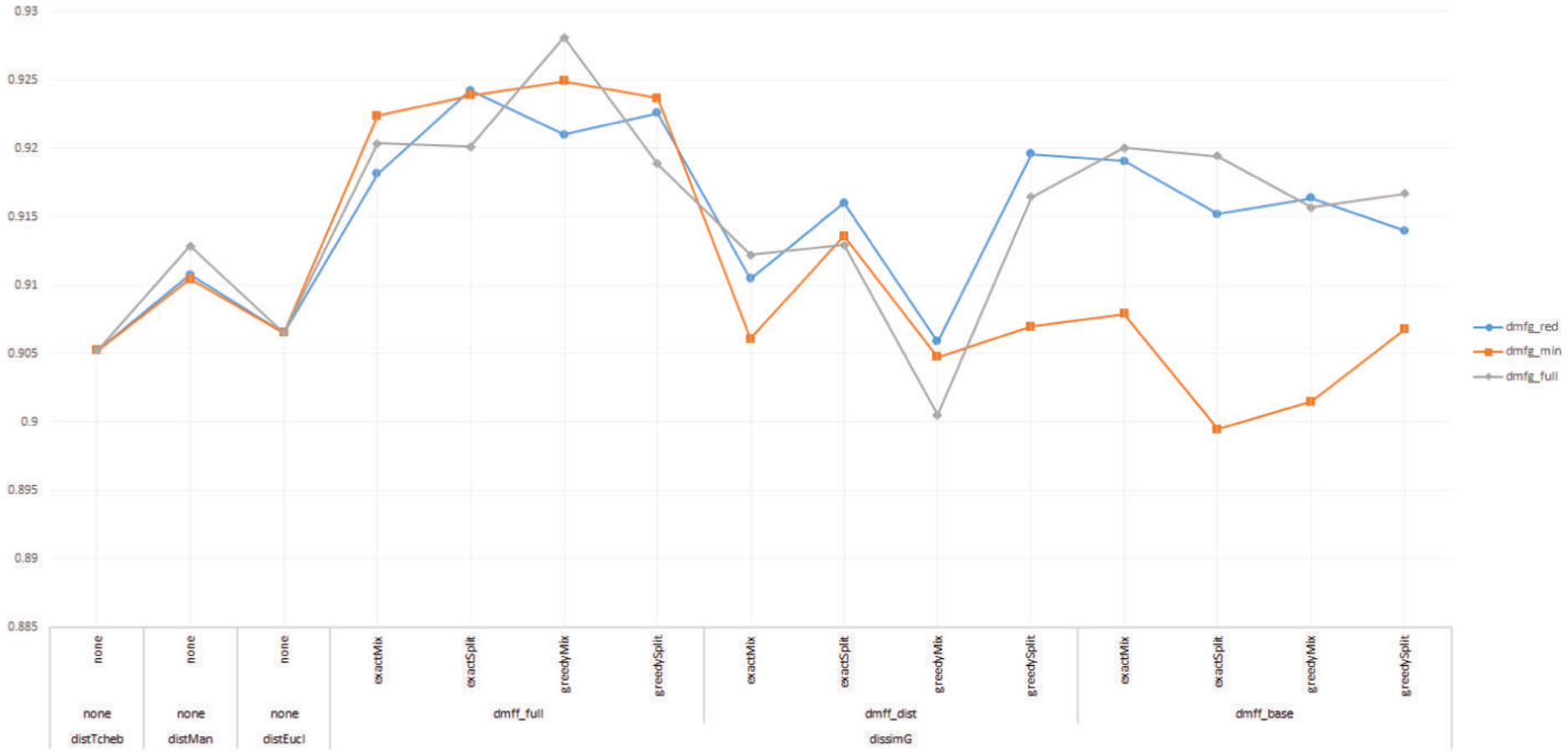

FIGURE 4.27 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs généraux utilisé, sur l'espace optimal.

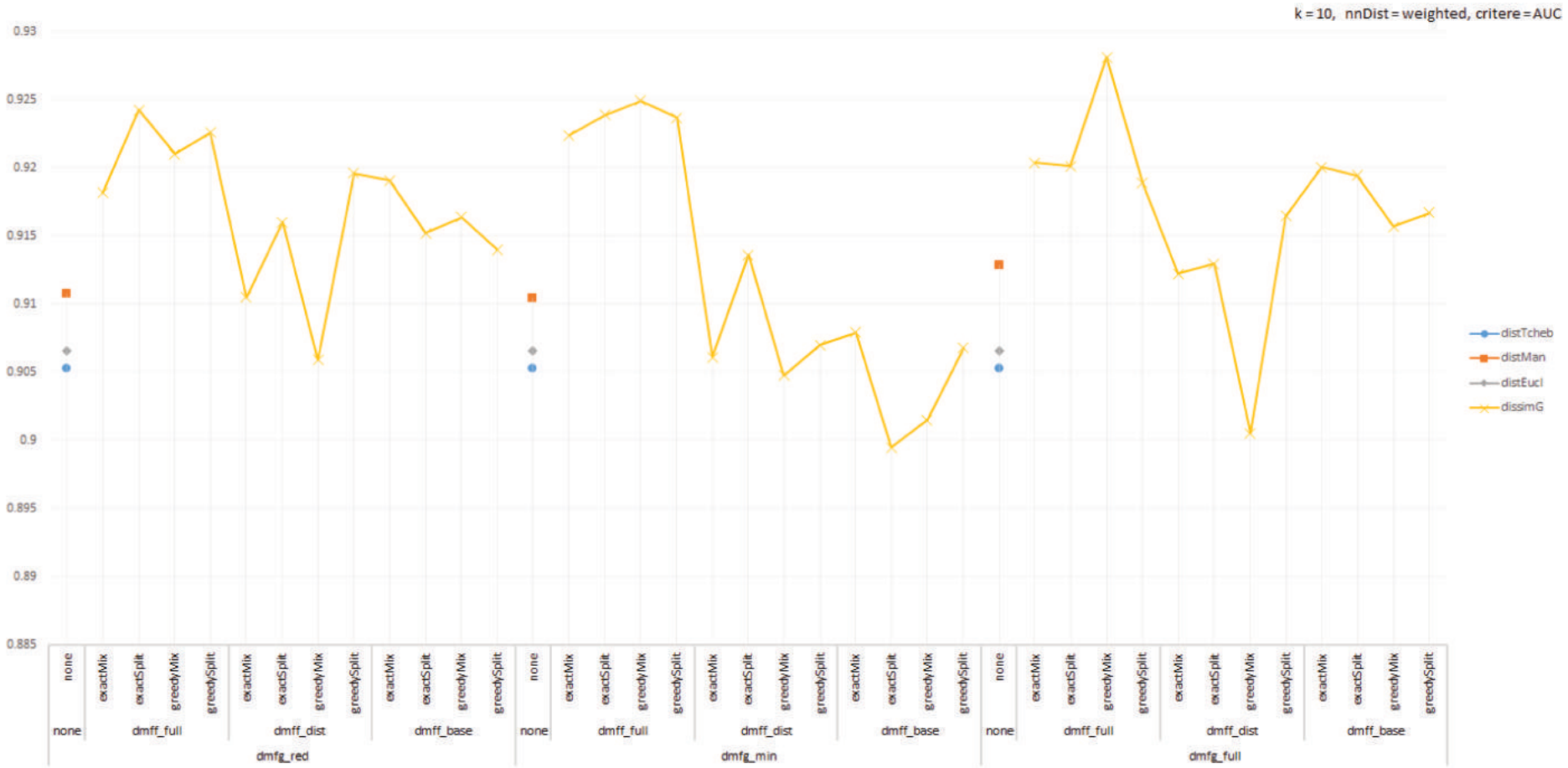

FIGURE 4.29 – Moyenne des performances au méta-niveau selon la dissimilarité sur les méta-attributs généraux.

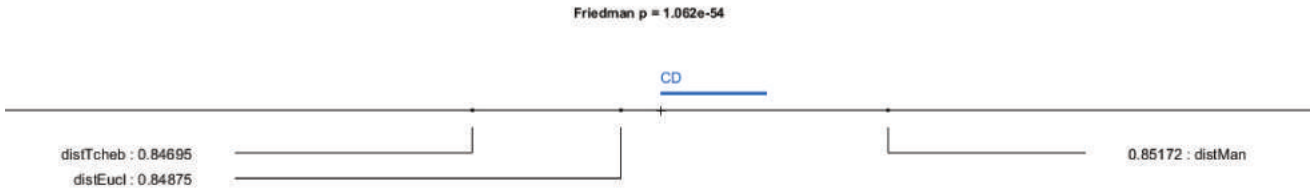

FIGURE 4.30 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarité sur les méta-attributs généraux.

L'asymétrie du problème rend impossible la comparaison statistique de $dissimG$ et des distances (on a besoin de mesures appariées, différant selon une seule dimension), mais on peut observer en Figure 4.29 (page 129) la dominance de $dissimG$ dans une majorité de cas. De plus, une comparaison des distances confirme la dominance de la distance de la distance de Manhattan sur les deux autres (voir Figure 4.30), ce qui conforte son choix comme brique de base de la dissimilarité normalisée par la borne supérieure.

Le test de Mann-Whitney, conçu pour permettre la comparaison d'échantillons de tailles différentes, est ici une alternative valide au test de Friedman pour comparer les distances à la dissimilarité. On présente en Table 4.22 les résultats de tests de Mann-Whitney pour l'hypothèse H_0 : la probabilité qu'une observation de l'échantillon "G=$dissimG$" soit supérieure à une observation de l'échantillon "G≠$dissimG$" est égale à la probabilité qu'une observation de l'échantillon "G≠$dissimG$" soit supérieure à une observation de l'échantillon "G=$dissimG$" (et de même avec G=$distMan$).

Échantillon	Taille	Moyenne	*p-value*
G≠$dissimG$	85320	0.8491	2.12E-34
G=$dissimG$	341280	0.8553	
G=$distMan$	28440	0.8517	4.97E-04
G=$dissimG$	341280	0.8553	

TABLE 4.22 – Résultats des tests de Mann-Whitney comparant les distances à la dissimilarité.

On peut donc dans les deux cas valider la supériorité de $dissimG$ sur les distances, bien qu'elle soit en grande partie une distance de Manhattan. En particulier, la dominance de $dissimG$ sur la distance de Manhattan assure de l'intérêt de la normalisation par la borne supérieure et de la prise en compte des valeurs de méta-attributs manquantes par dissimilarité à l'absence de valeur.

Méta-attributs des attributs Nos ensembles de méta-attributs des attributs diffèrent par l'adjonction de mesures normalisées. *DMFf_base* comprend un ensemble de méta-attributs adaptés aux attributs numériques et nominaux, auquel *DMFf_full* ajoute quelques méta-attributs normalisés par le nombre d'instances. *DMFf_dist*, l'ensemble des distributions des attributs, ne représente pas à proprement parler un ensemble de méta-attributs des attributs, mais peut être considéré comme tel par la dissimilarité δ^{σ}_{KS} utilisant le test de Kolmogorov-Smirnov, comme décrit précédemment.

On présente en Figure 4.31 (page 132) les performances moyennes au méta-niveau pour les dissimilarités utilisant ces ensembles, selon leurs autres facteurs primaires. Comme précédemment, les différents ensembles surclassent la baseline (*none*), mais l'asymétrie ne permet pas de le valider par le test de Friedman. En revanche, on peut observer une certaine tendance de *DMFf_base* à mieux se comporter que les autres ensembles, ce que l'on vérifie en Figure 4.32.

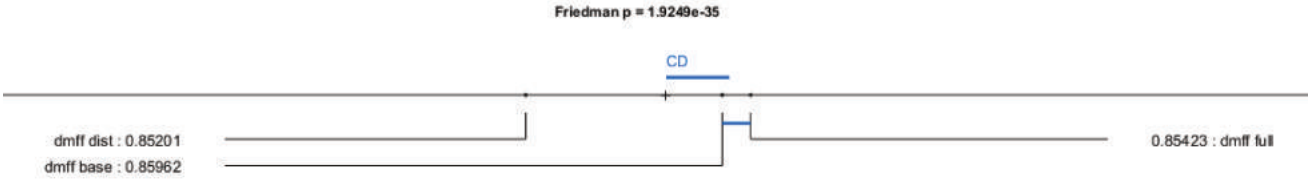

FIGURE 4.32 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensembles de méta-attributs des attributs.

Le résultat observé ne coïncide pas complètement avec la tendance attendue : le test ne permet pas de conclure à une différence significative entre *DMFf_base* et *DMFf_full*. En revanche il établit que tous deux sont significativement supérieurs à *DMFf_dist*. On se place ensuite dans le cas particulier des valeurs optimales de facteurs secondaires isolées dans la section précédente ($k = 10$, nnDist=*weighted*, criterion=AUC, $n = 1$), pour répéter l'observation (Figure 4.34 en page 133). *DMFf_full* semble alors clairement dominer, ce que l'on va chercher à confirmer avec le test en Figure 4.33.

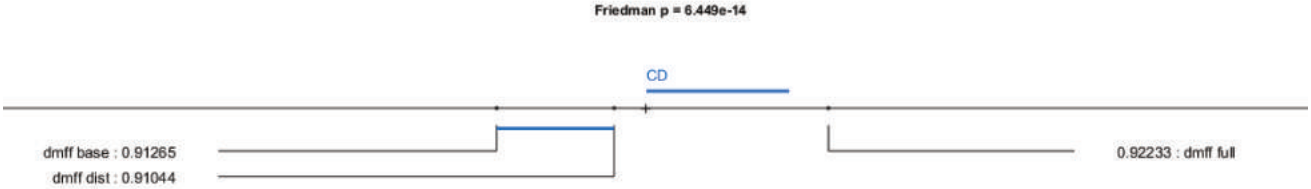

FIGURE 4.33 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensembles de méta-attributs des attributs, sur l'espace optimal.

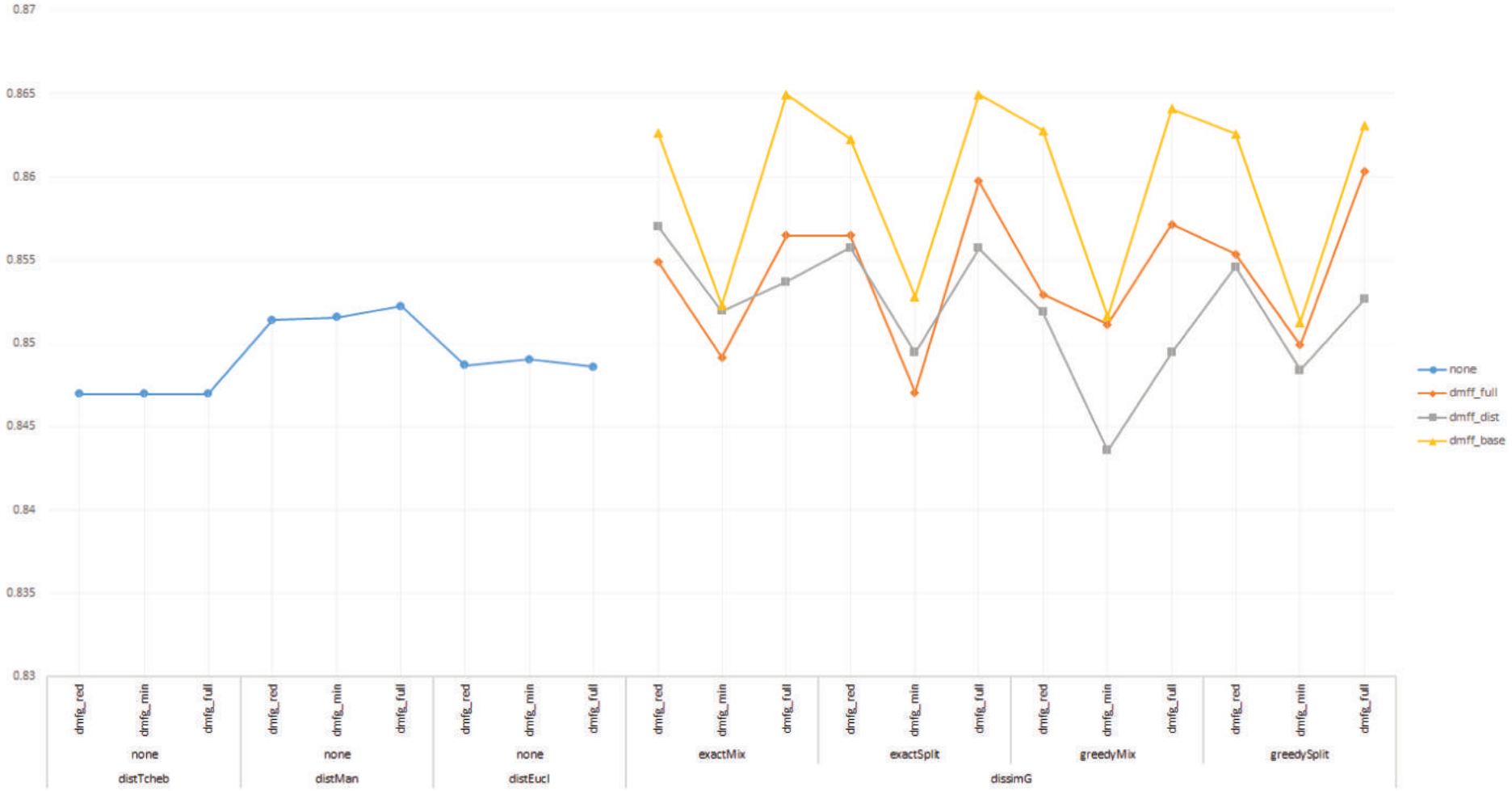

FIGURE 4.31 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs des attributs utilisé.

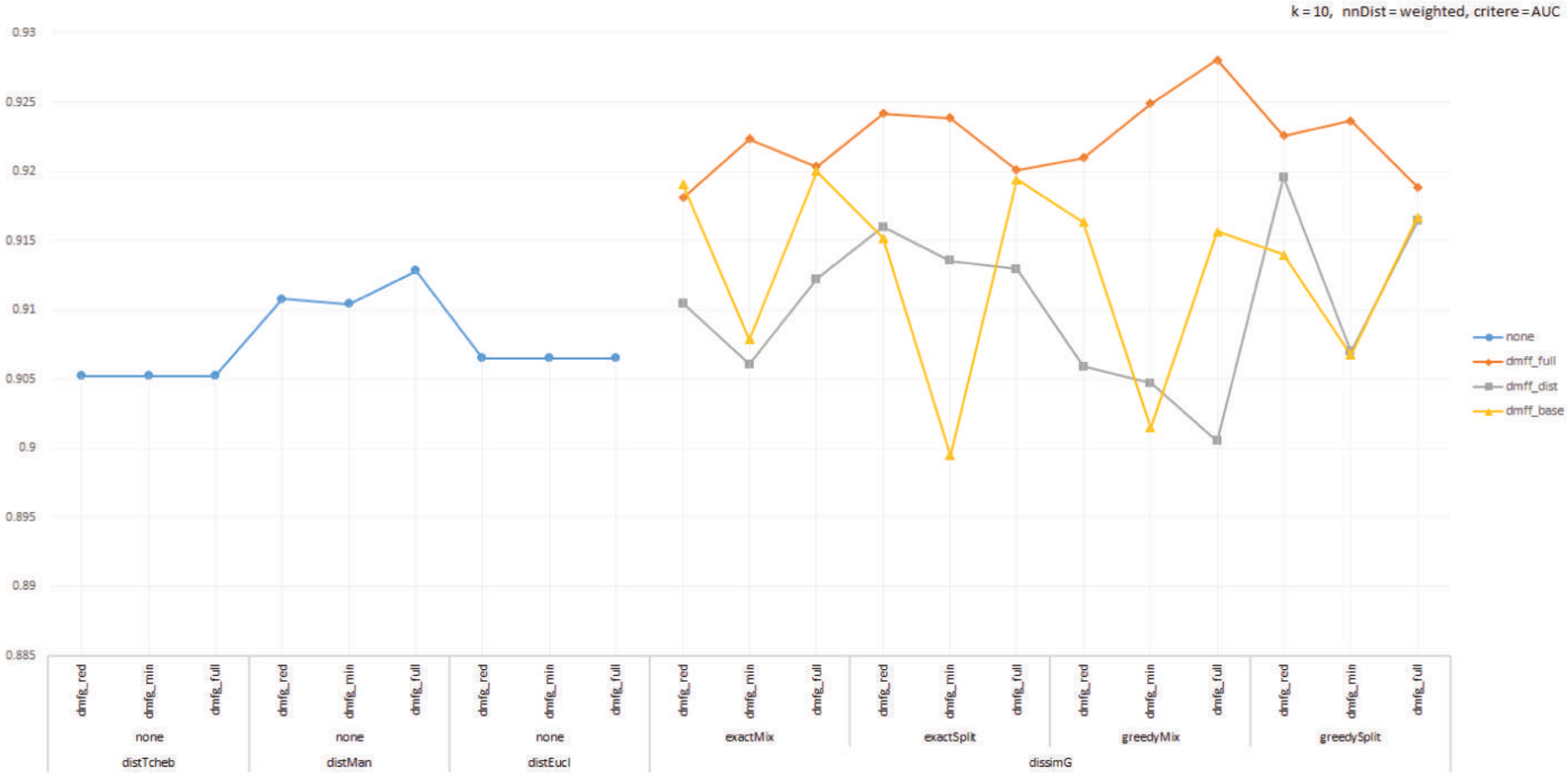

FIGURE 4.34 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs des attributs utilisé, sur l'espace optimal.

Le test valide cette fois la tendance perçue : *DMFf_full* domine les autres ensembles dans notre espace optimal. Ce résultat confirme l'intérêt des méta-attributs normalisés, d'autant plus qu'il émerge lorsqu'on se place dans la situation de performance optimale des dissimilarités.

Le test de Friedman n'étant pas adapté à ce cas particulier, on utilisera à nouveau le test de Mann-Whitney afin de valider l'intérêt global d'utiliser les méta-attributs des attributs individuels. On comparera donc ici les dissimilarités de la *baseline*, n'utilisant pas de méta-attributs des attributs (DMFf=*none*), avec l'ensemble des autres dissimilarités (DMFf≠*none*). On procède à un test de Mann-Whitney pour l'hypothèse H_0 : la probabilité qu'une observation de l'échantillon "DMFf=*none*" soit supérieure à une observation de l'échantillon "DMFf≠*none*" est égale à la probabilité qu'une observation de l'échantillon "DMFf≠*none*" soit supérieure à une observation de l'échantillon "DMFf=*none*".

On présente en Table 4.23 les résultats de ce test sur l'espace complet, puis en se limitant au critère d'aire sous la courbe, et enfin en se limitant à l'espace optimal.

Échantillon	Taille	Moyenne	*p-value*
DMFf=*none*	85320	0.8491	2.12E-34
DMFf≠*none*	341280	0.8553	
DMFf=*none*, criterion=*AUC*	21330	0.9058	1.04E-51
DMFf≠*none*, criterion=*AUC*	85320	0.9079	
DMFf=*none*, criterion=*AUC*, $k = 10$, nnDist=*weighted*	3555	0.9077	9.53E-07
DMFf≠*none*, criterion=*AUC*, $k = 10$, nnDist=*weighted*	14220	0.9152	

TABLE 4.23 – Résultats des tests de Mann-Whitney comparant les dissimilarités proposées à celles de la baseline.

On constate bien dans tous les cas une *p-value* suffisante ($ie < 0.05$) pour écarter H_0 : les échantillons diffèrent donc significativement. La performance moyenne des dissimilarités utilisant des méta-attributs des attributs étant toujours supérieure à celles de la *baseline*, on peut bien conclure à leur dominance. L'augmentation de la *p-value* sur l'espace optimal était attendue, dans la mesure où la réduction de la taille des échantillons réduits d'autant la puissance du test. Mais on peut remarquer que se limiter au critère d'aire sous la courbe permet au contraire de la réduire davantage. Ceci met à nouveau en valeur l'adéquation des dissimilarités proposées avec ce critère, leur avantage y étant plus affirmé malgré la perte de puissance du test.

Dissimilarité sur les attributs Les dissimilarités sur les méta-attributs des attributs diffèrent par le *mapping* σ qu'elles emploient. Les distinctions principales sont entre les *mappings "greedy"* et *"exact"* utilisant respectivement un algorithme glouton et exact pour l'appariement, et entre les *"split"* et *"mix"* considérant séparément ou non les attributs numériques et nominaux.

On présente en Figure 4.35 (page 136) les performances moyennes au méta-niveau pour les dissimilarités construites sur ces différents *mappings*. Une simple observation ne révèle pas de tendance forte, et seul un test statistique (Figure 4.36) révèle la dominance de l'*exactSplit* sur le *greedyMix*. Ajoutant à cela les temps d'exécution présentés en Figure 4.11, on pourra favoriser l'utilisation de *mappings "split"*, environ 1.5 fois plus couteux que le *greedyMix* pour le *greedySplit*, et 2 fois plus couteux pour l'*exactSplit*.

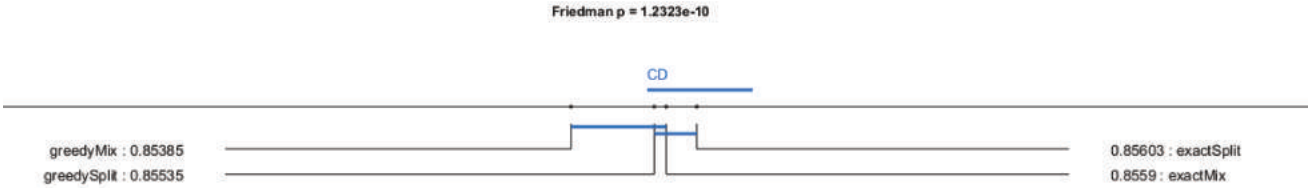

FIGURE 4.36 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarités sur les méta-attributs des attributs.

On vérifie en Figure 4.37 si ce résultat varie lorsque l'on considère uniquement les dissimilarités δ^{σ}_{KS} utilisant le test de Kolmogorov-Smirnov au lieu de méta-attributs des attributs. On arrive cette fois à établir la supériorité des deux méthodes *"exact"* sur le *greedyMix*, ce qui est très raisonnable dans la mesure où δ^{σ}_{KS} n'utilise qu'une seule valeur (le résultat du *KS-test*) pour discriminer, et serait donc plus sensible aux variations "aléatoires" induites par le *mapping "greedy"*.

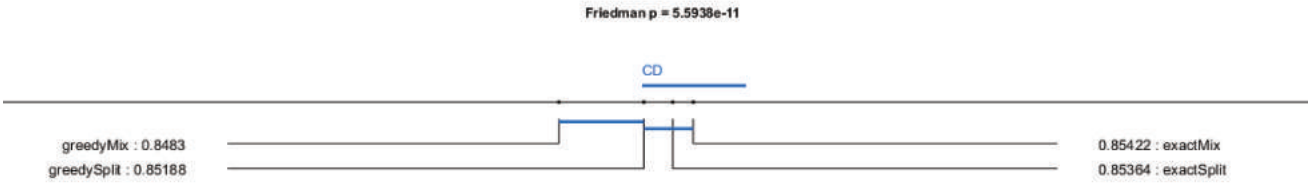

FIGURE 4.37 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarités sur les méta-attributs des attributs, et utilisant δ^{σ}_{KS}.

Le résultat précédent peut laisser penser que le choix du *mapping* a plus d'influence sur

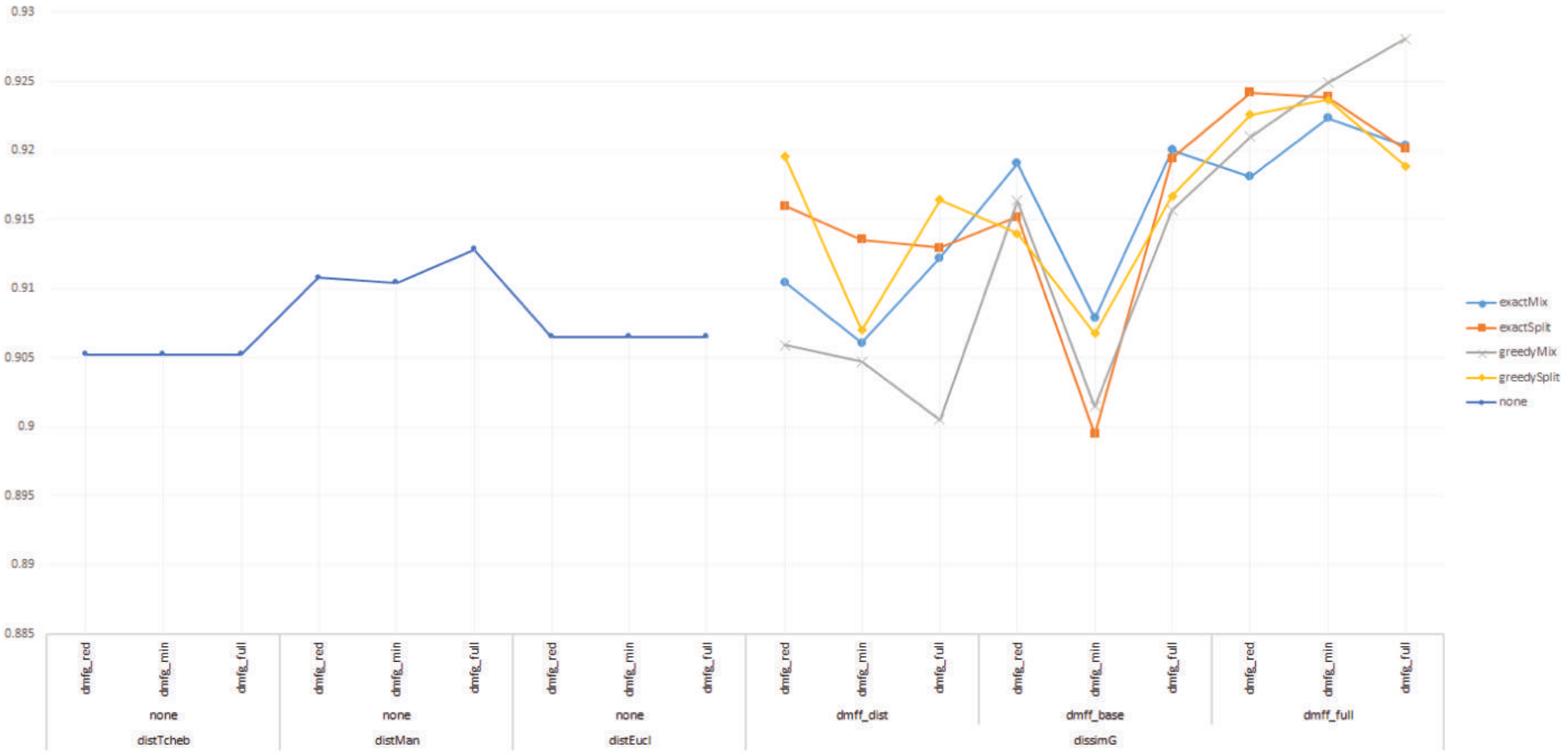

FIGURE 4.35 – Moyenne des performances au méta-niveau selon la dissimilarité sur les méta-attributs des attributs.

les dissimilarités δ^{σ}_{KS}, ce que l'on peut tenter de confirmer par un test sur les dissimilarités δ^{σ}_{F} (c'est à dire n'utilisant pas le *KS-test*). Leur nombre supérieur confère au test en Figure 4.38 une puissance supérieure, ce qui devrait permettre de trouver plus facilement les différences significatives. Or, le test ne parvient à mettre en évidence aucune différence significative, ce qui nous confirme bien une sensitivité supérieure de δ^{σ}_{KS} au choix du *mapping*.

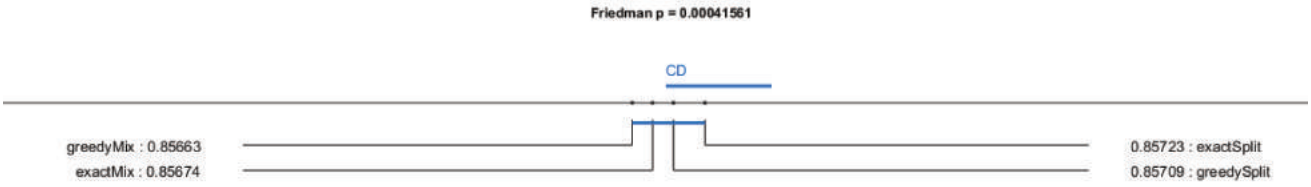

FIGURE 4.38 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarités sur les méta-attributs des attributs, et n'utilisant pas δ^{σ}_{KS}.

Ces résultats caractérisent donc un léger avantage à la fois en performance et en stabilité de δ^{σ}_{F} par rapport à δ^{σ}_{KS}, et pointent vers l'utilisation de *mappings greedySplit* ou *exactSplit*.

4.4.5.3 Comparatif global

La comparaison à la *baseline* forme le dernier spectre d'observations à mener. On étudie ainsi en Figure 4.39 (page 138) les performances des dissimilarités et des algorithmes de la *baseline*, par critère de performance de base. Les dissimilarités semblent bien dominer la *baseline*, ce qui sera à confirmer par des tests d'hypothèse, et présentent d'autre part des performances beaucoup plus stables d'un critère à l'autre (pas de gouffre pour l'*information score*, en particulier).

L'algorithme *KStar* (second de la liste, en orange) de la *baseline* semble présenter un profil de performance similaire aux dissimilarités (écart quasi-constant d'environ 20%), ce qui est raisonnable dans la mesure où il s'agit d'un algorithme proche de *kNN*, mais utilisant une distance basée sur l'entropie (Cleary, Trigg et al. 1995). Cette stabilité peut caractériser une meilleure capacité à utiliser des critères de base non-normalisés (l'*information score* peut par exemple prendre de très grandes valeurs), qui est un avantage de la structure *"instance based"* des dissimilarités.

On valide ensuite l'amélioration par rapport à la baseline par des tests d'hypothèse. Se limiter à l'espace optimal réduit trop le nombre de résultats à comparer, ce qui affecte

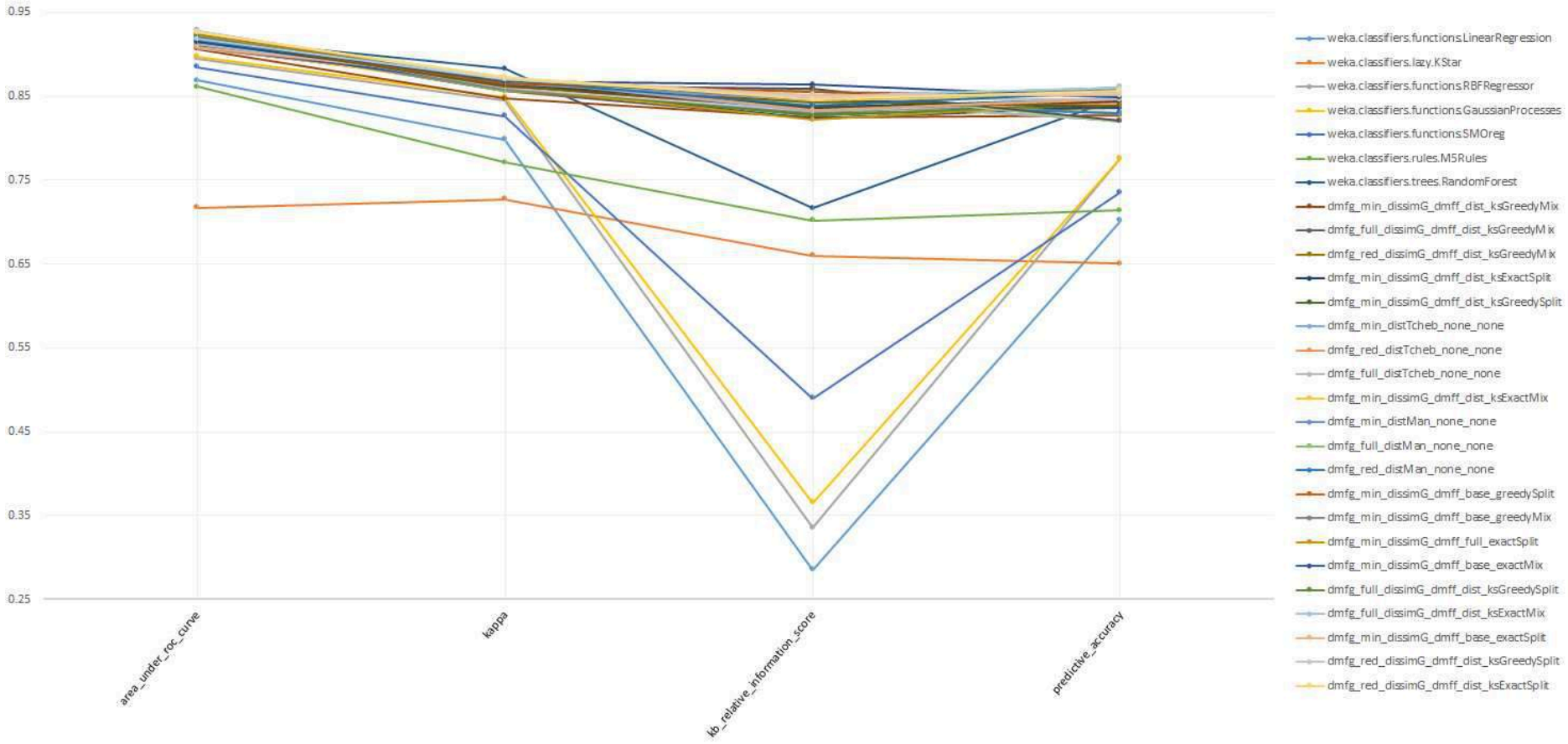

FIGURE 4.39 – Performances moyennes des différents algorithmes au méta-niveau, par critère de performance de base.

la puissance des tests et empêche de tirer des conclusions. On se limite donc en Figure 4.40 (page 140) à comparer les différents algorithmes sur le critère d'aire sous la courbe, qui était notre meilleur candidat.

On valide donc bien une différence significative entre nos dissimilarités et la *baseline* d'algorithmes traditionnels. Seules quelques dissimilarités basées sur des distances simples (donc de la baseline) ne peuvent être considérées significativement supérieures au meilleur algorithme de la baseline (RandomForest (Breiman 2001)). Le test ne suffit malheureusement pas à établir de différence significative entre ces dissimilarités de la *baseline* et les autres, mais cette différence a déjà pu être constatée dans la section précédente.

Un test plus puissant utilisant tous les critères de performance de base sur l'espace des algorithmes au méta-niveau (Figure 4.41 en page 141) confirme la supériorité des dissimilarités par rapport à tous les algorithmes traditionnels de la *baseline*, mais là encore ne met pas en évidence de différence significative entre les dissimilarités.

Ces deux figures (4.40 et 4.41, respectivement en pages 140 et 141) nous permettent par ailleurs d'investiguer les méthodes ayant obtenu les meilleures performances moyennes au méta-niveau. Sur l'ensemble des critères (4.41), on trouve une dominance (dont on ne peut établir la significance à ce niveau) de l'ensemble de méta-attributs généraux comportant un nombre réduit de *landmarkers*, et l'ensemble de méta-attributs des attributs complet. Si l'on se restreint à l'aire sous la courbe, on voit figurer en tête les dissimilarités comparant les distributions des attributs par le test de Kolmogorov-Smirnov, avec une performance au méta-niveau dépassant les 0.95. Cela signifie que ces dissimilarité ont permis de recommander des algorithmes en moyenne 95% aussi bons que le meilleur disponible !

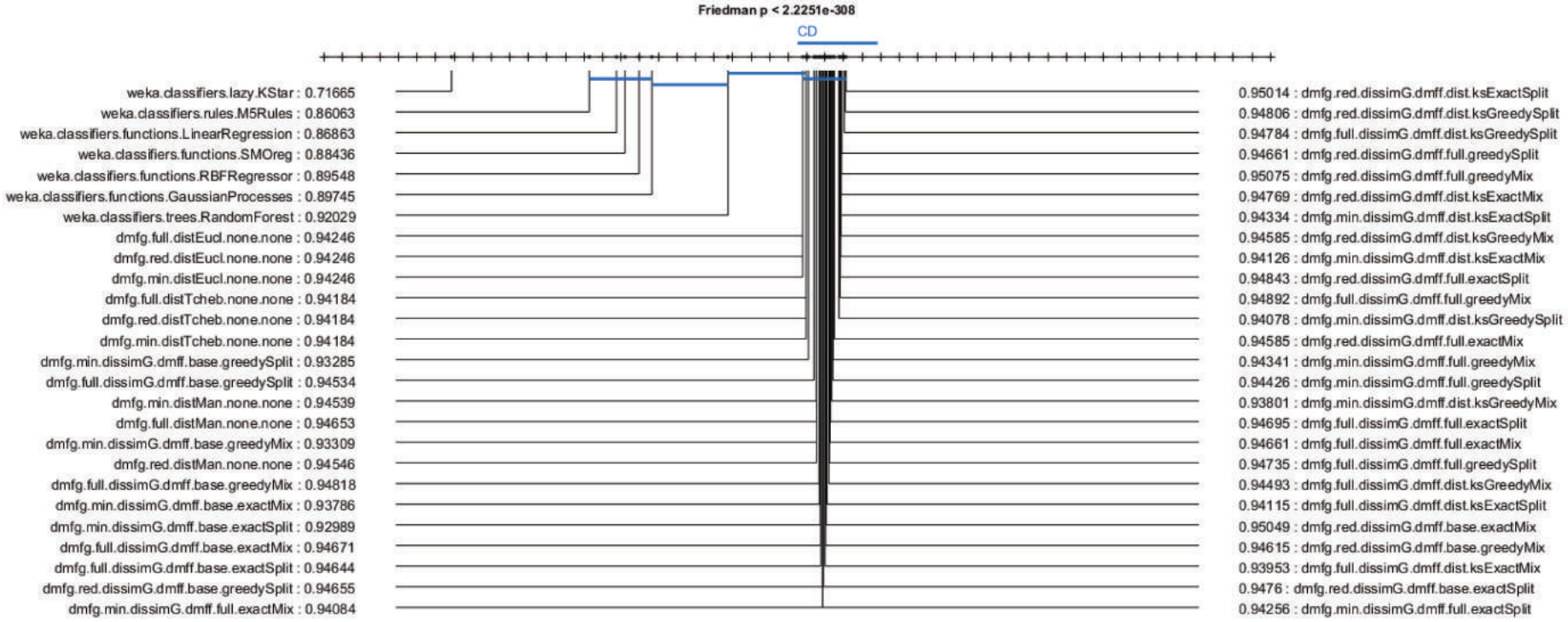

FIGURE 4.40 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents algorithmes au méta-niveau, pour le critère d'aire sous la courbe.

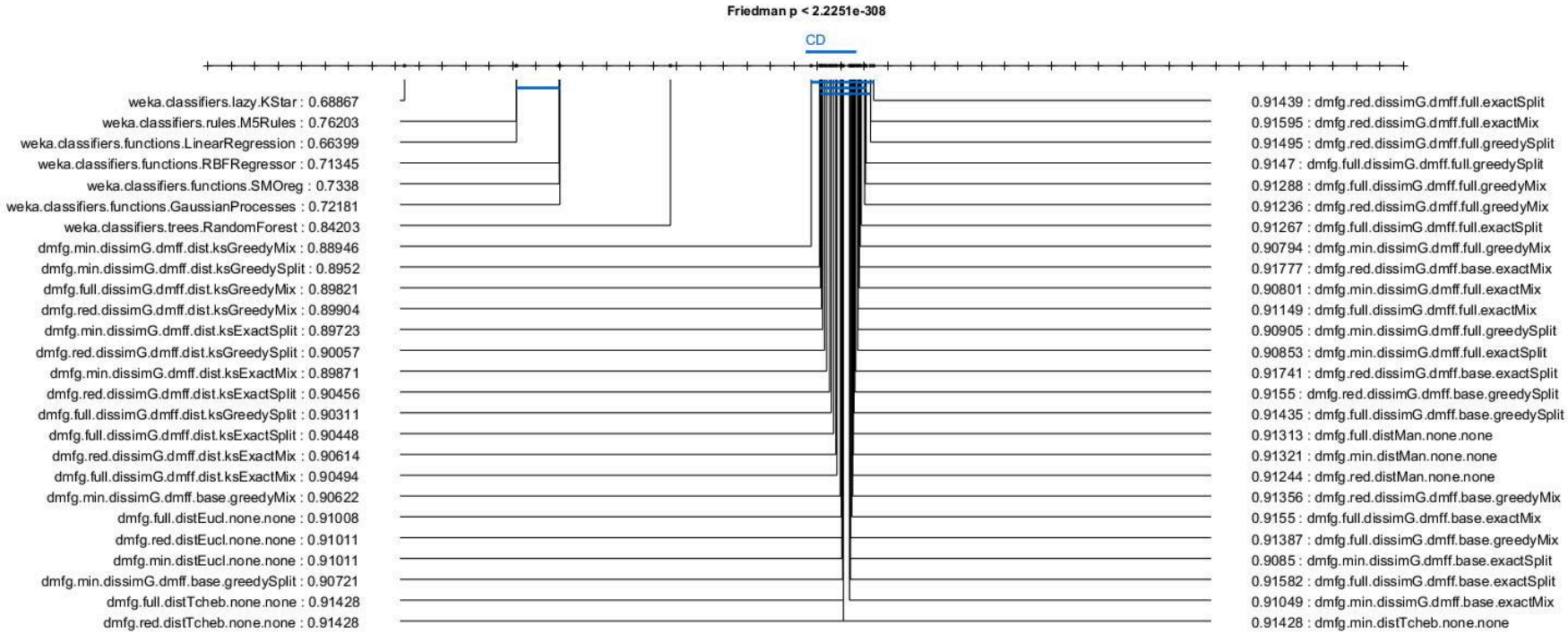

FIGURE 4.41 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents algorithmes au méta-niveau.

4.5 Discussion

4.5.1 Récapitulatif des résultats

On résume ici les différents résultats significatifs obtenus et leur interprétation. Afin de présenter les résultats de tests d'hypothèse de manière synthétique, on utilisera les notations $\prec$ et $\preceq$ pour *"est dominé significativement"* et *"est dominé sans différence significative"* respectivement.

4.5.1.1 Facteurs secondaires

On qualifie de secondaires les facteurs pouvant influer sur la performance au méta-niveau mais ne faisant pas partie intégrante des dissimilarités. On étudie ici leur impact sur la performance au méta-niveau et leurs relations avec les différentes dissimilarités.

Nombre de voisins considérés par l'algorithme 3 :

$$3 \prec 10 \preceq 5 \text{ en général}$$
$$3 \prec 5 \prec 10 \text{ sur } \textit{DMFf\_full}$$

Ceci semble indiquer la nécessité de considérer davantage de voisins pour exploiter de grands ensembles de méta-attributs des attributs.

Méthode d'estimation de performance d'un classifieur selon celle de ses voisins :

$$\textit{Mean} \prec \textit{Weighted}$$

Le test valide la supériorité globale de la méthode **"weighted"** (utilisant à nouveau la dissimilarité pour pondérer la performance des plus proches voisins). Ce résultat conforte l'intérêt perçu des dissimilarités, et leur utilité pour l'estimation de performance au niveau de base.

Critère de performance de base :

$$\textit{Predictive accuracy} \prec \textit{Information score} \prec \textit{Kappa} \prec \textit{AUC}$$

L'ordonnancement des différents critères semble valide dans une portion significative des cas de test, et aucun contre-exemple significatif n'a pu être trouvé. Tous nos éléments pointent donc vers l'aire sous la courbe de ROC, dont l'utilisation comme critère de performance de base semble mener aux meilleures performances au méta-niveau, et qui est généralement reconnu comme un bon critère de performance en classification.

Nombre de recommandations :

$$5 \prec 3 \prec 1$$

Augmenter le nombre de recommandations impacte donc bien négativement les performances au méta-niveau. Ce compromis était attendu, le but étant de contrôler la variance des résultats. On observe les variances les plus basses sur le critère d'*AUC*, sans que le nombre de recommandations ne semble l'impacter. Le seul critère où la multiplication des recommandations a l'effet escompté et limite la variance, est celui de précision, mais les valeurs y restent peu intéressantes. Cette étude conforte donc le choix du critère d'aire sous la courbe de *ROC*, mais rend apparent le peu d'intérêt de la multiplication des recommandations, du point de vue des performances. D'autres contraintes du domaine de la recommandation peuvent justifier la multiplication des recommandations, mais la qualité des solutions trouvées semble suffisamment élevée pour que cela entraine une perte significative de performance. Ce *tradeoff* sera bien sûr à considérer, mais dépasse le cadre de la présente étude.

Ces études préliminaires nous ont permis de gagner une meilleure compréhension de l'impact des facteurs secondaires, et en particulier de déterminer un "espace optimal", ensemble de valeurs des facteurs secondaires maximisant la performance au méta-niveau. Dans certaines des observations suivantes, on se placera dans cet espace optimal pour analyser le comportement des dissimilarités dans ce qui serait le plus proche d'un futur cas d'application réel.

> Comme espace optimal, on retient donc les valeurs de $k = 10$, nnDist=*weighted*, criterion=AUC, et $n = 1$. D'autre part, on prendra toujours la performance au méta-niveau pour $n = 1$ (sauf mention contraire explicite).

4.5.1.2 Facteurs primaires

On qualifie alors de primaires les éléments fonctionnels variables des dissimilarités.

Dissimilarité sur les méta-attributs généraux :

$$distTcheb \prec distEucl \prec distMan \prec dissimG$$

La dominance de la distance de Manhattan sur les autres distances en conforte le choix comme brique de base des dissimilarités. De plus, on valide la supériorité de la dissimilarité normalisée par la borne supérieure, ce qui confirme l'intéret de cette normalisation et de la prise en compte des valeurs de méta-attributs manquantes par dissimilarité à l'absence de valeur.

Méta-attributs généraux :

$$DMFg\_min \prec DMFg\_red \preceq DMFg\_full$$

L'utilisation de méta-attributs de type *landmarker* améliore significativement la performance, mais la multiplication des évaluations des *landmarkers* ne présente pas d'effet significatif. On peut rejoindre ici des travaux de sélection d'attributs insistant sur l'importance de la diversité au sein des attributs (Brown et al. 2012). Il pourrait ainsi être intéressant d'étudier l'impact de la *diversité* des *landmarkers* et autres méta-attributs généraux choisis sur la performance au méta-niveau.

Dissimilarité sur les attributs :

$$greedyMix \prec exactSplit$$

Le coût d'exécution du *mapping exactSplit* dépassant celui du *greedyMix* d'un simple facteur 2, ces résultats pointent vers l'utilisation du *mapping* exact avec séparation des attributs par type.

Méta-attributs des attributs :

$$DMFf\_none \prec DMFf\_dist \prec DMFf\_base \preceq DMFf\_full \text{ en général}$$
$$DMFf\_none \prec DMFf\_base \preceq DMFf\_dist \prec DMFf\_full \text{ sur l'espace optimal}$$

La dominance de *DMFf_full* sur l'espace optimal confirme l'intérêt des méta-attributs normalisés, d'autant plus qu'il émerge lorsqu'on se place dans la situation de performance optimale des dissimilarités. *DMFf_none* est de plus toujours dominé, ce qui confirme ici l'intérêt d'utiliser les méta-attributs des attributs.

4.5.1.3 Comparatif global

La comparaison à la *baseline* forme le dernier spectre d'observations à mener. On compare ainsi directement les performances des dissimilarités et des algorithmes de la *baseline*, confirmant la supériorité des dissimilarités par rapport aux algorithmes traditionnels étudiés. Le test ne suffit malheureusement pas à établir de différence significative entre les différentes dissimilarités, mais de telles différences ont déjà pu être constatées dans les paragraphes précédents.

Ces résultats, dans l'ensemble très positifs, démontrent l'intérêt des approches proposées dans le cas de la sélection d'algorithmes de classification, problème standard de méta-analyse. La proximité des biais entre différents problèmes de méta-analyse permet de supposer que ces approches par dissimilarité pourront y avoir de bons résultats, et ce moyennant très peu de modifications pour de nouveaux problèmes de sélection d'algorithme. Les performances obtenues au méta-niveau, pouvant dépasser les 0.95 sur notre ensemble de 395 jeux de données, signifient que certaines approches par dissimilarité ont permis d'identifier des algorithmes de classification en moyenne 95% aussi performants que le meilleur sur chaque jeu de données. Le processus est coûteux si pratiqué *offline* : calculer la matrice complète des dissimilarités entre jeux de données peut prendre des heures pour de grands ensembles. En revanche, dans une perspective *online*, l'ajout d'un nouveau jeu de données ne nécessite que de calculer sa dissimilarité à ceux déjà présents, au lieu de reconstruire le modèle complet. Ces approches seront donc particulièrement adaptées à des processus de sélection d'algorithmes actifs maintenant une base de cas sur lesquels construire leurs recommandations.

4.5.2 Conclusion

Nous avons donc ici proposé des fonctions de dissimilarité entre jeux de données présentant un ensemble de propriétés désirables, et capables d'employer des méta-attributs caractérisant des attributs particuliers de ces jeux de données. Nous avons montré qu'elles permettent de caractériser l'adéquation d'algorithmes de classification avec des jeux de données plus efficacement que des distances traditionnelles, et qu'elles peuvent être employées avec de bonnes performances dans le contexte de régression au méta-niveau pour la sélection d'algorithmes.

De nombreuses pistes d'amélioration restent cependant à explorer. Tout d'abord, notre dissimilarité permet d'utiliser des méta-attributs caractérisant des attributs particuliers des jeux de données, mais diverses expériences (Brown et al. 2012 ; H. Peng et al. 2005) ont montré que les propriétés d'un attribut *dans le contexte des autres attributs* sont au moins aussi importantes. Il serait alors intéressant de permettre l'utilisation de tels méta-attributs relationnels, comme la covariance, ou l'information mutuelle, par la dissimilarité. D'autre part, bien que divers et provenant d'approches très différentes, les méta-attributs employés dans nos expériences ne couvrent pas complètement l'état de l'art en la matière. Ces dernières années ont été riches en contributions introduisant de nouveaux méta-attributs (Ho et Basu 2002 ; Ntoutsi et al. 2008 ; Y. Peng et al. 2002 ; Sun et Pfahringer 2013), dont l'utilisation pourrait révéler l'intérêt d'une approche par dissimilarité dans de nouveaux contextes. Enfin, comme l'efficacité de l'approche par dissimilarité apparaît très dépendante du contexte (comme c'est souvent le cas en apprentissage et méta-apprentissage), il pourrait être intéressant de concevoir une méthode d'évaluation de méta-attributs considérant leurs diverses natures (globaux, liés à un attribut, relationnels...). Il serait alors possible de caractériser l'utilité des divers méta-attributs dans une variété de situations et donc d'approfondir notre connaissance du problème de méta-apprentissage.

Dans le cadre de l'assistance intelligente à l'analyse de données, un atout particulier de notre approche est qu'elle permet une caractérisation unifiée des expériences d'analyse de données. En effet, disposant d'une quelconque représentation du processus d'analyse de données et de ses résultats, il est possible de l'intégrer dans la dissimilarité, permettant ainsi la comparaison directe d'expériences complètes. Il s'agit là d'un premier pas vers de nouvelles approches d'assistance intelligente à l'analyse de données, permettant notamment l'utilisation directe d'heuristiques pour la découverte et recommandation de processus d'analyse adaptés.

Bibliographie du chapitre

Akaike, H (1974). « A new look at the statistical model identification ». In : *IEEE Transactions on Automatic Control* 19.6, p. 716–723 (cf. p. 112).

Batista, GEAPA et Diego Furtado Silva (2009). « How k-nearest neighbor parameters affect its performance ». In : *Argentine Symposium on Artificial Intelligence*, p. 1–12 (cf. p. 115).

Brazdil, Pavel, João Gama et Bob Henery (1994). « Characterizing the applicability of classification algorithms using meta-level learning ». In : *European conference on machine learning*. Springer, p. 83–102 (cf. p. 30, 31, 65, 96, 97).

Breiman, Leo (2001). « Random forests ». In : *Machine learning* 45.1, p. 5–32 (cf. p. 25, 26, 112, 139).

Brown, Gavin, Adam Pocock, Ming-Jie Zhao et Mikel Luján (2012). « Conditional likelihood maximisation : a unifying framework for information theoretic feature selection ». In : *The Journal of Machine Learning Research* 13.1, p. 27–66 (cf. p. 35, 36, 126, 144, 146).

Cleary, John G, Leonard E Trigg et al. (1995). « K* : An instance-based learner using an entropic distance measure ». In : *Proceedings of the 12th International Conference on Machine learning*. T. 5, p. 108–114 (cf. p. 23, 26, 112, 137).

Cohen, Jacob (1968). « Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. » In : *Psychological bulletin* 70.4, p. 213 (cf. p. 27, 28, 62, 100, 170).

Dong, Lin, Eibe Frank et Stefan Kramer (2005). « Ensembles of balanced nested dichotomies for multi-class problems ». In : *Knowledge Discovery in Databases : PKDD 2005*. Springer, p. 84–95 (cf. p. 25, 26, 69, 95).

Frank, András (2005). « On Kuhn's Hungarian method : a tribute from Hungary ». In : *Naval Research Logistics (NRL)* 52.1, p. 2–5 (cf. p. 88).

Frank, Eibe (2014). *Fully supervised training of Gaussian radial basis function networks in WEKA*. Rapp. tech. Department of Computer Science, The University of Waikato (cf. p. 26, 112).

Frank, Eibe, Yong Wang, Stuart Inglis, Geoffrey Holmes et Ian H Witten (1998). « Using model trees for classification ». In : *Machine Learning* 32.1, p. 63–76 (cf. p. 23, 26, 95).

Giraud-Carrier, Christophe, Ricardo Vilalta et Pavel Brazdil (2004). « Introduction to the special issue on meta-learning ». In : *Machine learning* 54.3, p. 187–193 (cf. p. 29, 32, 77, 96).

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann et Ian H Witten (2009). « The WEKA data mining software : an update ». In : *ACM SIGKDD explorations newsletter* 11.1, p. 10–18 (cf. p. 66, 100, 168).

Ho, Tin Kamo et Mitra Basu (2002). « Complexity measures of supervised classification problems ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.3, p. 289–300 (cf. p. 33, 146).

Holmes, Geoffrey, Mark Hall et Eibe Prank (1999). « Generating rule sets from model trees ». In : *Australasian Joint Conference on Artificial Intelligence*. Springer, p. 1–12 (cf. p. 112).

Kalousis, Alexandros (2002). « Algorithm selection via meta-learning ». Thèse de doct. Universite de Geneve (cf. p. 31, 32, 35, 78, 79).

Kalousis, Alexandros et Maelanie Hilario (2001a). « Model selection via meta-learning : a comparative study ». In : *International Journal on Artificial Intelligence Tools* 10.04, p. 525–554 (cf. p. 30–32, 55, 78, 96).

Kuhn, Harold W (1955). « The Hungarian method for the assignment problem ». In : *Naval research logistics quarterly* 2.1-2, p. 83–97 (cf. p. 88).

Leite, Rui, Pavel Brazdil et Joaquin Vanschoren (2012). « Selecting classification algorithms with active testing ». In : *Machine Learning and Data Mining in Pattern Recognition*. Springer, p. 117–131 (cf. p. 31, 55, 61, 96).

MacKay, David JC (1998). « Introduction to Gaussian processes ». In : *NATO ASI Series F Computer and Systems Sciences* 168, p. 133–166 (cf. p. 112).

Ntoutsi, Irene, Alexandros Kalousis et Yannis Theodoridis (2008). « A general framework for estimating similarity of datasets and decision trees : exploring semantic similarity of decision trees. » In : *SDM*. SIAM, p. 810–821 (cf. p. 146).

Peng, Hanchuan, Fuhui Long et Chris Ding (2005). « Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.8, p. 1226–1238 (cf. p. 35, 36, 146).

Peng, Yonghong, Peter A Flach, Pavel Brazdil et Carlos Soares (2002). « Decision tree-based data characterization for meta-learning ». In : *IDDM-2002*, p. 111 (cf. p. 34, 61, 146).

Pfahringer, Bernhard, Hilan Bensusan et Christophe Giraud-Carrier (2000). « Tell me who can learn you and i can tell you who you are : Landmarking various learning algorithms ». In : *Proceedings of the 17th international conference on machine learning*, p. 743–750 (cf. p. 34, 61, 100).

Serban, F (2013). « Toward effective support for data mining using intelligent discovery assistance ». Thèse de doct. University of Zurich (cf. p. 41, 42, 77, 159).

Smid, Jakub (2016). « Computational Intelligence Methods in Metalearning ». Thèse de doct. Charles University in Prague (cf. p. 43, 78, 86, 88, 99, 104, 185).

Smid, Jakub et Roman Neruda (2014). « Comparing datasets by attribute alignment ». In : *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*. IEEE, p. 56–62 (cf. p. 43, 78).

Smirnov, N (1948). « Table for Estimating the Goodness of Fit of Empirical Distributions ». In : *The Annals of Mathematical Statistics* 19.2, p. 279–281 (cf. p. 108).

Smola, Alex J et Bernhard Schölkopf (2004). « A tutorial on support vector regression ». In : *Statistics and computing* 14.3, p. 199–222 (cf. p. 25, 26, 112).

Sun, Quan et Bernhard Pfahringer (2013). « Pairwise meta-rules for better meta-learning-based algorithm ranking ». In : *Machine learning* 93.1, p. 141–161 (cf. p. 30, 31, 61, 65, 96, 146).

Sun, Quan, Bernhard Pfahringer et Michael Mayo (2012). « Full model selection in the space of data mining operators ». In : *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, p. 1503–1504 (cf. p. 38, 42, 96).

Vanschoren, Joaquin, Hendrik Blockeel, Bernhard Pfahringer et Geoffrey Holmes (2012). « Experiment databases ». In : *Machine Learning* 87.2, p. 127–158 (cf. p. 28, 90).

Wang, Liwei, Masashi Sugiyama, Cheng Yang, Kohei Hatano et Jufu Feng (2009). « Theory and algorithm for learning with dissimilarity functions ». In : *Neural computation* 21.5, p. 1459–1484 (cf. p. 80, 90, 91).

Wistuba, Martin, Nicolas Schilling et Lars Schmidt-Thieme (2015). « Learning Data Set Similarities for Hyperparameter Optimization Initializations ». In : *MetaSel@ PKDD/ECML*, p. 15–26. url : `http://ceur-ws.org/Vol-1455/#paper-04` (visité le 01/04/2017) (cf. p. 39, 42, 79).

Wolpert, David H (1996). « The lack of a priori distinctions between learning algorithms ». In : *Neural computation* 8.7, p. 1341–1390 (cf. p. 29, 95, 173, 175).

Wolpert, David H et William G Macready (1997). « No free lunch theorems for optimization ». In : *Evolutionary Computation, IEEE Transactions on* 1.1, p. 67–82 (cf. p. 29, 95).

Yoo, Andy B, Morris A Jette et Mark Grondona (2003). « Slurm : Simple linux utility for resource management ». In : *Job Scheduling Strategies for Parallel Processing*. Springer, p. 44–60 (cf. p. 67, 113).

Zakova, Monika, Petr Kremen, Filip Zelezny et Nada Lavrac (2011). « Automating knowledge discovery workflow composition through ontology-based planning ». In : *Automation Science and Engineering, IEEE Transactions on* 8.2, p. 253–264 (cf. p. 41, 42, 55, 77, 151, 159).

L'intuition est l'art, spécifique à l'esprit humain, d'élaborer une réponse correcte à partir de données qui, par elles-mêmes, sont incomplètes, voire trompeuses.

Isaac Asimov

Un programme informatique fait ce que vous lui avez dit de faire, pas ce que vous voulez qu'il fasse.

Troisième loi de Greer

Chapitre 5

Assistance à l'analyse de données

Ce chapitre présente les nouvelles approches de méta-analyse rendues possibles par les dissimilarités proposées au chapitre 4, ainsi qu'une preuve de concept d'assistance à l'analyse de données en faisant usage.

5.1 Introduction

La méta-analyse de données désigne la recherche d'une méthode efficace ou optimale permettant d'adresser un problème d'analyse de données. Cela recouvre une grande variété de tâches, dont certaines ont d'ores et déjà été abondamment étudiées (voir section 2.6 de l'état de l'art). En particulier, la recommandation de chaînes de traitements d'analyse complète de données a reçu un intérêt croissant ces dernière années (Nguyen et al. 2014 ; Serban et al. 2013 ; Zakova et al. 2011). Ce problème consiste en la construction de chaînes de traitements permettant de résoudre différents problèmes d'analyse de données. Les différentes approches évoquées s'adressent à des utilisateurs ayant une bonne expérience en analyse de données, soit pour être capable d'en exploiter des résultats souvent obscurs, soit directement pour guider l'algorithme dans sa construction d'une chaîne de traitements adaptée. C'est là l'un des problèmes que l'on souhaite adresser : on veut recommander des chaînes de traitements complètes à des utilisateurs non experts. On doit donc être capable de construire une recommandation adaptée au problème de l'utilisateur, prenant en compte ses données et ses préférences. Le processus d'analyse doit de plus aboutir dans un environnement facilitant les interactions avec l'utilisateur. Ce dernier doit permettre à l'utilisateur de modifier la chaîne de traitements recommandée de manière simple et guidée, et l'assister dans son appropriation des résultats d'analyse.

D'autre part, afin d'explorer de nouvelles formes de méta-analyse, on s'appuiera sur

les dissimilarités entre jeux de données présentées au chapitre 4. Ces dernières permettent l'exploitation directe de bases d'expériences passées dans la construction de nouvelles analyses. On utilisera ainsi la connaissance des performances des nombreuses analyses passées pour choisir lesquelles employer dans une nouvelle situation.

On décrira donc tout d'abord de façon générale les nouvelles approches de méta-analyse rendues possibles par les dissimilarités proposées au chapitre 4. On présentera ensuite une preuve de concept d'assistance à l'analyse de données faisant usage d'une telle approche, et l'évaluation qui en est faite.

5.2 Méthode de Méta-analyse

5.2.1 Principe

On se trouve donc dans une situation où un utilisateur, expert dans son domaine métier et non en analyse de données, a exprimé un besoin d'analyse et collecté de la donnée pouvant être utilisée pour y répondre (voir partie supérieure de la Figure 5.1). La construction d'un processus d'analyse approprié étant un problème complexe et coûteux (Zeevi et al. 2015), on veut dans un premier temps recommander à l'utilisateur un processus d'analyse répondant raisonnablement bien à son besoin.

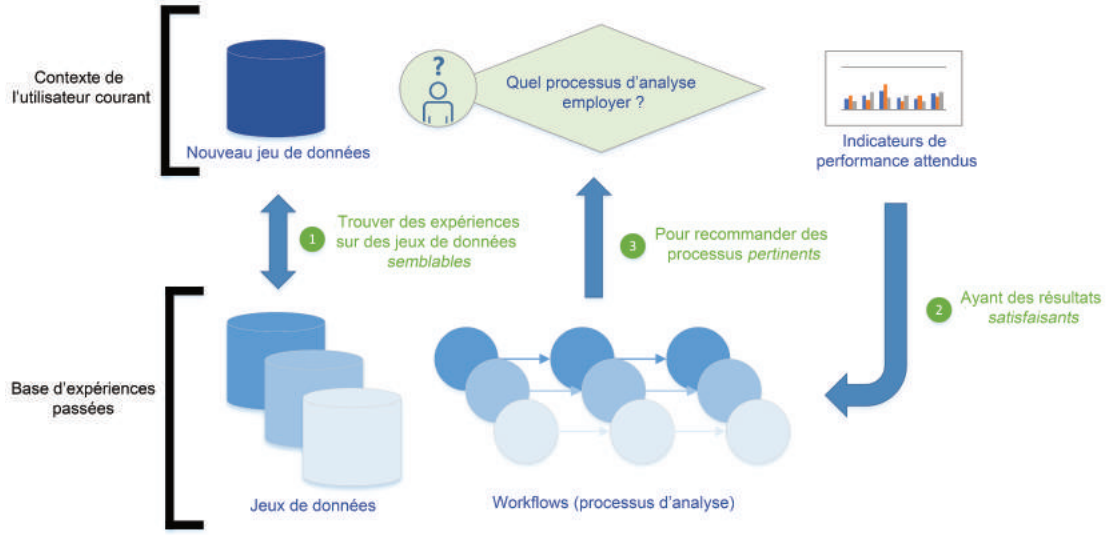

FIGURE 5.1 – Principe de méta-analyse

Pour ce faire, on dispose d'une connaissance du domaine de l'analyse de données sous la forme d'une base d'expériences passées (voir partie inférieure de la Figure 5.1), chacune représentant l'application d'un processus d'analyse complet (ou *workflow*) à un jeu de données et en qualifiant les résultats. Afin de trouver dans cette base les expériences

passées *potentiellement pertinentes* pour le problème de l'utilisateur, on peut discriminer selon deux critères :

1. L'une des hypothèses fondatrices du méta-apprentissage stipule qu'un même algorithme exhibera souvent des performances similaires sur des jeux de données *semblables* (Gama et Brazdil 1995). On peut étendre ici cette hypothèse à l'ensemble du domaine de l'analyse pour supposer qu'un processus d'analyse qui s'est bien comporté sur un jeu de données *semblable* à celui de notre utilisateur a de bonnes chances d'être également efficace sur ce dernier. On peut alors simplement employer les dissimilarités proposées au chapitre 4 pour trouver ces jeux de données *semblables*, et donc les processus y ayant été employés (voir ① en Figure 5.1).
2. Ensuite, on peut exprimer le besoin utilisateur par un ensemble d'indicateurs de performances attendus, et chercher parmi les expériences passées celles maximisant ces indicateurs (voir ② en Figure 5.1).

Les expériences satisfaisant ces deux critères seront donc très probablement pertinentes dans le contexte de l'utilisateur actuel, et pourront être utilisées pour recommander un processus d'analyse adapté (voir ③ en Figure 5.1).

5.2.2 Cas d'utilisation

On présente en Figure 5.2 (page 154) la méthode de méta-analyse envisagée sur un cas d'utilisation. Plaçons-nous tout d'abord dans la perspective de l'utilisateur. Ce dernier dispose d'un jeu de données décrivant différents attributs d'un certain nombre d'individus (instances), et a un besoin de modélisation (problème d'analyse de données) particulier au regard de ce jeu de données (Obenshain 2004). Par exemple, un médecin essayant de prédire l'occurrence d'une pathologie donnera priorité à la sensibilité du modèle (détecter tous les positifs, au risque d'avoir des faux-positifs), tandis qu'un chercheur modélisant un phénomène encore mal compris priorisera plutôt l'informativité et l'explicabilité du modèle pour en extraire un maximum de connaissances. C'est donc là un problème métier précis, que l'on veut exprimer sous la forme d'un problème d'analyse de données, en modélisant au mieux le besoin utilisateur par des préférences et contraintes sur les résultats d'analyse. Cette spécification de la tâche d'analyse sers de base à une méta-analyse telle que représentée dans en Figure 5.2.

Ensuite, dans une perspective d'assistance à l'analyse, on relève cette fois deux objectifs : apporter une bonne solution au problème soumis par l'utilisateur, et générer ce faisant de la connaissance afin de trouver de meilleures solutions à l'avenir.

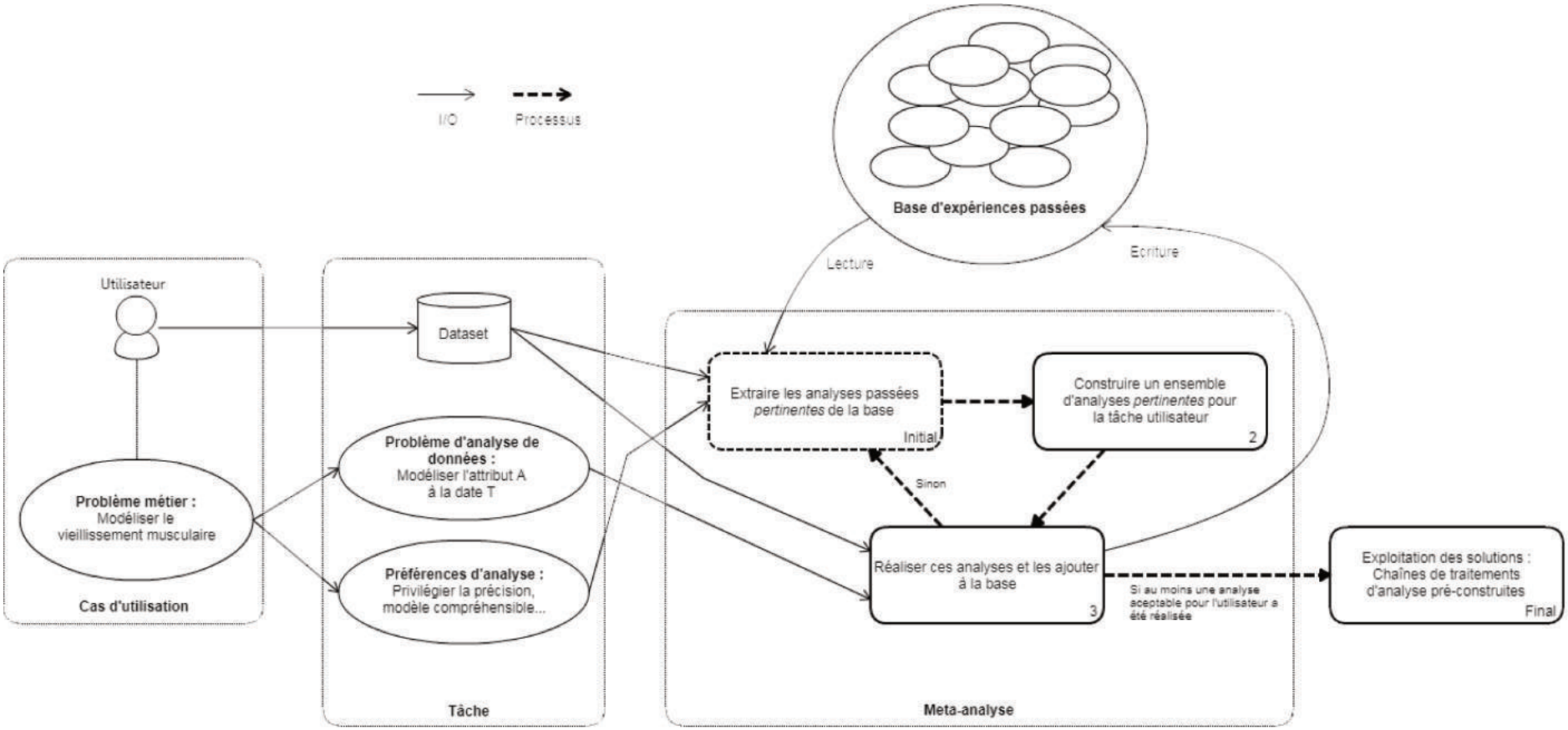

FIGURE 5.2 – Méta-analyse sur la tâche d'analyse spécifié à partir d'un problème métier.

Notre connaissance du méta-domaine prend la forme d'une *base d'expériences passées* (Fig. 5.2). Chacune de ces expériences représente l'application d'un traitement d'analyse à un jeu de données, dont le résultat est évalué selon un ensemble de critères. Par exemple, une expérience de cette base pourrait représenter l'information que le jeu de données fourni par notre médecin a été modélisé à l'aide d'un algorithme de classification Bayésien naïf, et que le modèle résultant présentait tel score de précision, sensibilité et spécificité.

A partir de cette *base d'expériences* et de la *tâche* définie selon le besoin utilisateur, notre processus de méta-analyse peut se décomposer en trois étapes représentées en Figure 5.2 :

Initiale. La première étape consiste à trouver dans la base d'expérience passées celles pouvant se révéler *pertinentes* dans le contexte de l'utilisateur actuel. Comme présenté en 5.2.1, on peut réaliser cette recherche en se basant sur deux informations : le jeu de données de l'utilisateur et son besoin en matière d'analyse. On peut donc trouver un ensemble d'expériences passées *pertinentes*, effectuées sur des jeux de données similaires et/ou présentant des profils de performance conformes aux attentes de l'utilisateur.

2. La seconde étape consiste à produire, à partir de ces expériences passées, des analyses du problème de l'utilisateur qui pourront se révéler pertinentes. Il s'agit donc de construire des processus d'analyse complets qui seront applicables au problème de l'utilisateur. Pour ce faire, le plus simple serait de directement réutiliser les processus employés dans les expériences passées pertinentes mises en évidence lors de l'étape *Initiale*, mais des approches plus élaborées sont possibles. On pourrait par exemple mélanger les processus d'analyse de ces expériences passées pertinentes pour tirer avantage de leur forces potentiellement différentes, ou chercher parmi leurs points communs ce qui leur a permis d'être considérés pertinents pour le réemployer. Quelle que soit l'approche envisagée, l'adaptation automatique des processus employés dans les analyses passées au problème de l'utilisateur sera une opération nécessaire et non triviale : les entrées, sorties et préconditions des processus d'analyses sont loin d'être standardisés et foisonnent d'exceptions...

3. La dernière étape de la boucle consiste en la réalisation des analyses du problème utilisateur ainsi produites. Il s'agit donc d'exécuter de manière complètement autonome les analyses du problème utilisateur construites lors de l'étape 2, et de mesurer leur performance réelles selon un ensemble d'indicateurs. Ces nouvelles expériences peuvent alors être ajoutées à la base d'expériences passées, remplissant

notre objectif *d'apprentissage* : on dispose de nouvelles connaissances qui aideront lors d'exécutions futures. De plus, si parmi les expériences ainsi réalisées, au moins une répond convenablement au besoin utilisateur, on a rempli notre premier objectif, qui était d'apporter une bonne solution au problème soumis par l'utilisateur. On peut alors recommander le processus d'analyse associé et faire avancer l'assistance utilisateur vers l'appropriation des résultats. Si malheureusement aucune des expériences réalisées ne répond aux besoins de l'utilisateur, il faudra en chercher de nouvelles, et l'on reboucle sur l'étape *Initiale* en relâchant peu à peu nos contraintes, jusqu'à l'obtention d'une solution acceptable.

On peut voir que la réelle complexité computationelle de cette méthode de méta-analyse réside à l'étape 3, où l'on exécute des analyses complètes du problème utilisateur pouvant individuellement se révéler très longues. Elles sont en revanche parfaitement indépendantes les unes des autres, et une implémentation parallèle satisfaisante permettrait de mitiger largement cette complexité.

5.2.3 Approches de construction d'expériences

L'étape 2 se révèle au final le point décisif de cette méthode de méta-analyse. L'approche choisie pour produire, à partir des expériences passées pertinentes, des analyses prometteuses du problème de l'utilisateur, sera déterminante pour l'efficacité de la méta-analyse. On présentera donc ici quelques approches envisageables. La notion de *workflow* sera souvent utilisée ici et requiert une définition plus complète :

Définition 9 *Un processus d'analyse, ou workflow, est un graphe orienté acyclique d'opérateurs, présentant exactement un nœud terminal (voir figure 5.3 en page 157, le graphe étant orienté de bas en haut). Les opérateurs présentent un nombre arbitraire d'entrées et de sorties, pouvant être connectées par les arrêtes du graphe. Pour simplifier, on ne considérera dans les exemples présentés ici que des opérateurs à une seule sortie, que l'on représente alors au-dessus du nœud. La sortie de l'opérateur constituant le nœud terminal ("Expected Output" en figure 5.3) est une solution de la tâche utilisateur.*

5.2.3.1 Réutilisation directe

Le moyen le plus simple de construire de nouvelles analyses du problème utilisateur serait de directement réutiliser les processus employés dans les expériences passées pertinentes mises en évidence lors de l'étape *Initiale*.

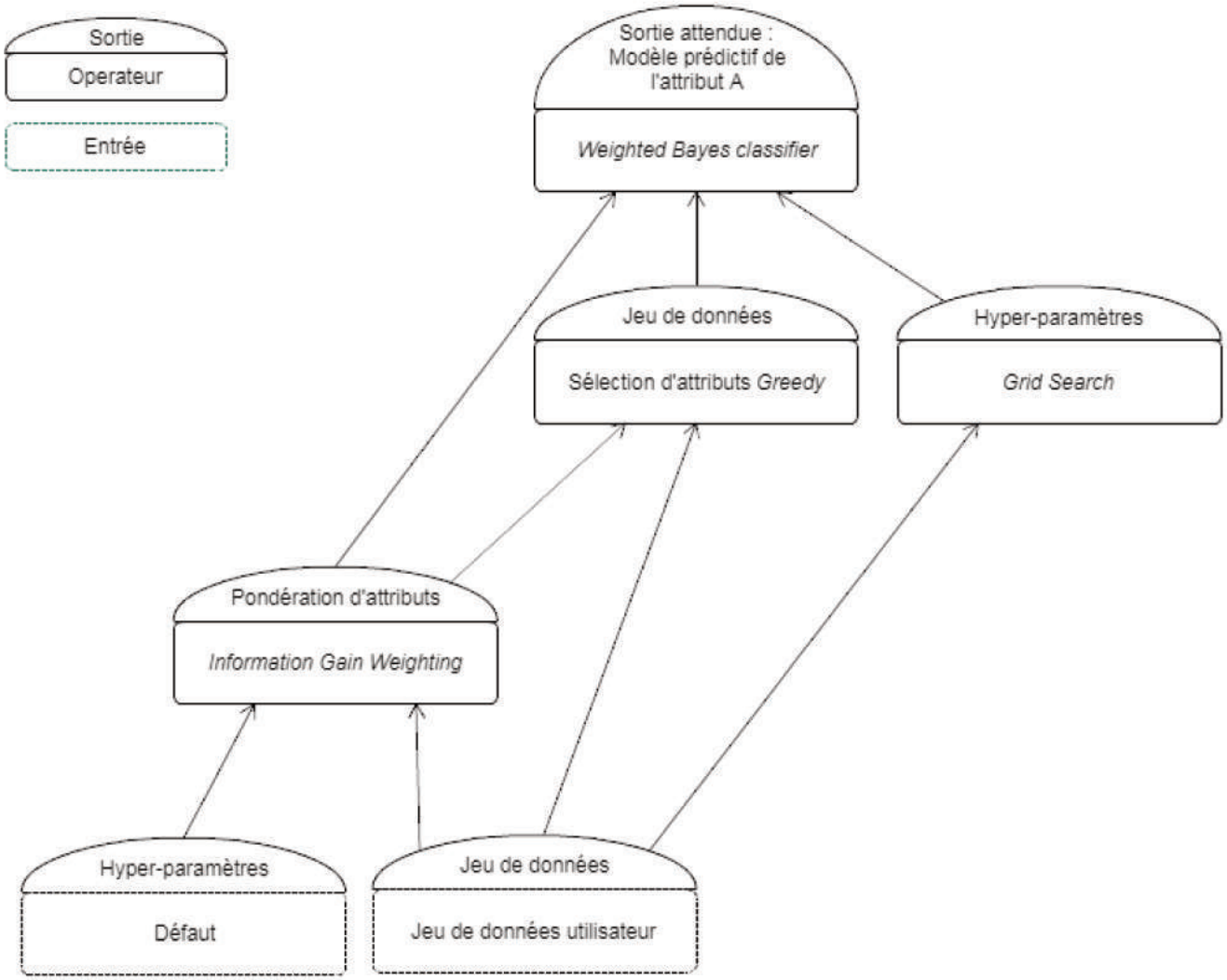

FIGURE 5.3 – Workflow de classification

Considérons par exemple le *Workflow* de la figure 5.3 (page 157), construit par un analyste pour prédire l'apparition de diabètes dans une population d'indiens Pima. Il dispose d'un jeu de données [1] décrivant différents paramètres métaboliques (les *attributs*) des individus de la population (les *instances*), et un attribut cible (la *classe*) stipulant si un test médical de l'individu concerné a pu mettre en évidence un diabète. Le *Workflow* de la figure 5.3 consiste alors en l'application d'un algorithme de classification ("*Weighted Bayes classifier*") à ce jeu de données après l'application de pré-traitements. En effet, les paramètres métaboliques impliqués dans le développement d'un diabète sont mal connus, et notre analyste a donc décidé d'effectuer un traitement de sélection d'attributs ("*Greedy*") avant la classification pour essayer d'exclure les attributs sans intérêt. L'opérateur de calcul de poids ("*Information Gain Weighting*") calcule le potentiel de discernement des différents attributs, ce qui permet d'estimer leur importance à la fois pour la sélection d'attributs et lors de la classification. Enfin, les hyper-paramètres de ces différentes opérations sont soit configurés par défaut, soit optimisés par des heuristiques dédiées ("*Grid Search*").

La construction de ce *workflow de classification* a demandé du temps et un certain niveau d'expertise à notre analyste, mais lui a permis d'apporter une bonne solution à son problème de prédiction pour le diagnostic. Son expérience a ensuite été intégrée dans notre *base d'expériences passées* car pouvant potentiellement être à nouveau utile dans le futur.

Entre maintenant un nouvel utilisateur ayant peu d'expérience en analyse de données, mais ayant un besoin similaire (prédire la récurrence d'un cancer du sein [2]). L'exploration de la *base d'expériences passées*, telle que décrite en 5.2.1, nous retourne l'expérience du diabètes des indiens Pima comme pertinente dans ce nouveau contexte. On peut alors simplement appliquer le *Workflow* de la figure 5.3 au jeu de données de notre nouvel utilisateur pour lui apporter rapidement une solution qui sera probablement intéressante. La difficulté de l'approche réside dans cette adaptation du *Workflow* au nouveau jeu de données, qui bien que généralement simple, peut se révéler complexe si des opérateurs se révèlent incompatibles avec certaines propriétés structurelles du nouveau jeu de données.

5.2.3.2 Planification heuristique

Plutôt que de réutiliser les processus d'analyses passées essentiellement inchangés, il pourrait être intéressant d'essayer de construire un nouveau processus en réponse au besoin utilisateur. C'est une approche que l'on retrouve dans une majorité de techniques

1. `https://www.openml.org/d/37`
2. `https://www.openml.org/d/13`

récentes d'assistance à l'analyse de données (Nguyen et al. 2014 ; Serban 2013 ; Zakova et al. 2011). On peut en revanche se démarquer de ces dernières par le fait que l'on a extrait de notre *base d'expériences passées* un petit ensemble d'expériences (et donc de *workflows*) *pertinents* dans le contexte de l'utilisateur courant. Les heuristiques de planification que l'on peut concevoir pour construire de nouveaux processus répondant au besoin utilisateur peuvent alors faire usage de ces *workflows* pertinents. Toute heuristique capable de parcourir l'espace des *workflows* en faisant usage de nos *workflows* pertinents (que ce soit en tant que point d'origine, objectif, ou constante d'une opération...) peut alors apporter une réponse au problème. On illustrera ici le fonctionnement d'une telle heuristique, inspirée de l'*Ajout récursif d'opérateur compatible* introduit par Diamantini et al. (2009a,b).

On veut donc construire un *workflow* d'analyse répondant au problème utilisateur. Pour cela on dispose :

- Des spécifications de la tâche de l'utilisateur. En reprenant l'exemple précédent de prédiction de la récurrence d'un cancer du sein, la tâche serait de construire un modèle prédictif de l'attribut décrivant cette récurrence dans le jeu de données de l'utilisateur [3]. On connait donc la forme de la solution attendue par l'utilisateur, qui sera la sortie de l'opérateur terminal de notre *workflow*.
- D'un ensemble d'expériences passées jugées pertinentes dans ce contexte, donc ayant obtenu des résultats qui conviendraient à notre utilisateur sur des jeux de données semblables.

On définit ensuite deux opérations sur des *workflows* :

Expand. On se réfère ici à la Figure 5.4 (page 160). L'opération **Expand** considère une sortie déjà spécifiée (soit comme solution attendue par l'utilisateur (*Sortie attendue* en ①), soit comme *Entrée* nécessaire à un autre opérateur (le *Jeu de données* en ②)), et ajoute un nouvel opérateur capable de la produire. Sur la première opération **Expand** de la Figure 5.4 (passage du ① au ②), on a ainsi considéré la sortie attendue par l'utilisateur pour lui associer un opérateur capable de la produire (le *Weighted Bayes classifier*). Cet opérateur demandant trois *Entrées*, on les développe en ② : on retrouve le jeu de données sur lequel construire le modèle de classification, une pondération des attributs de ce jeu de données, et un ensemble d'hyper-paramètres

3. `https://www.openml.org/d/13`

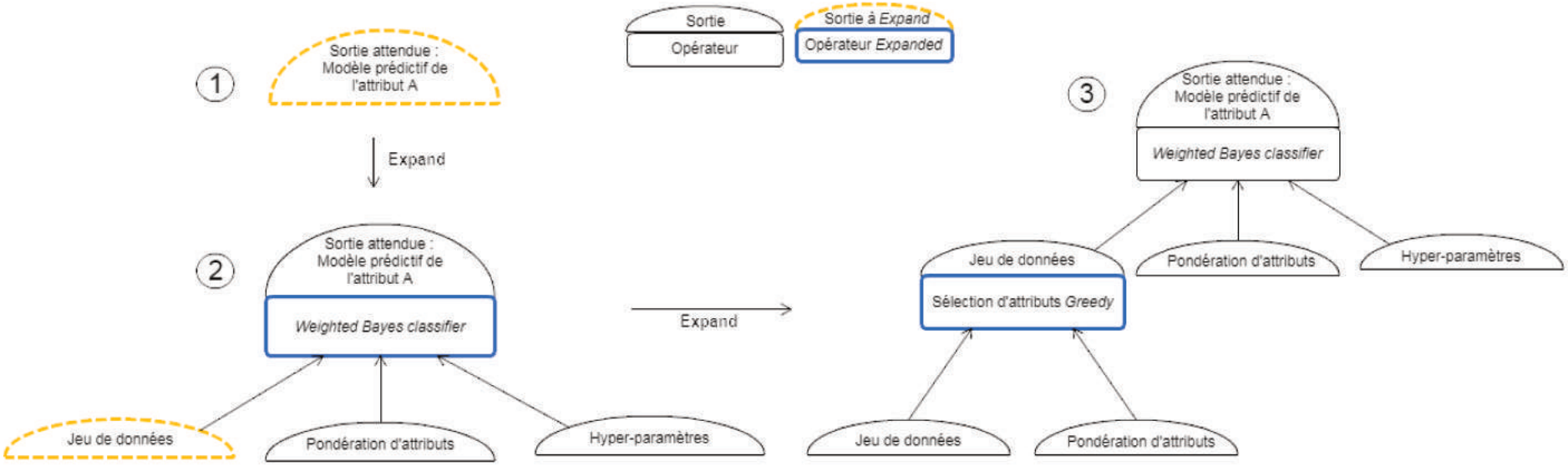

FIGURE 5.4 – Opération *Expand*

FIGURE 5.5 – Fin des itérations d'*Expand*

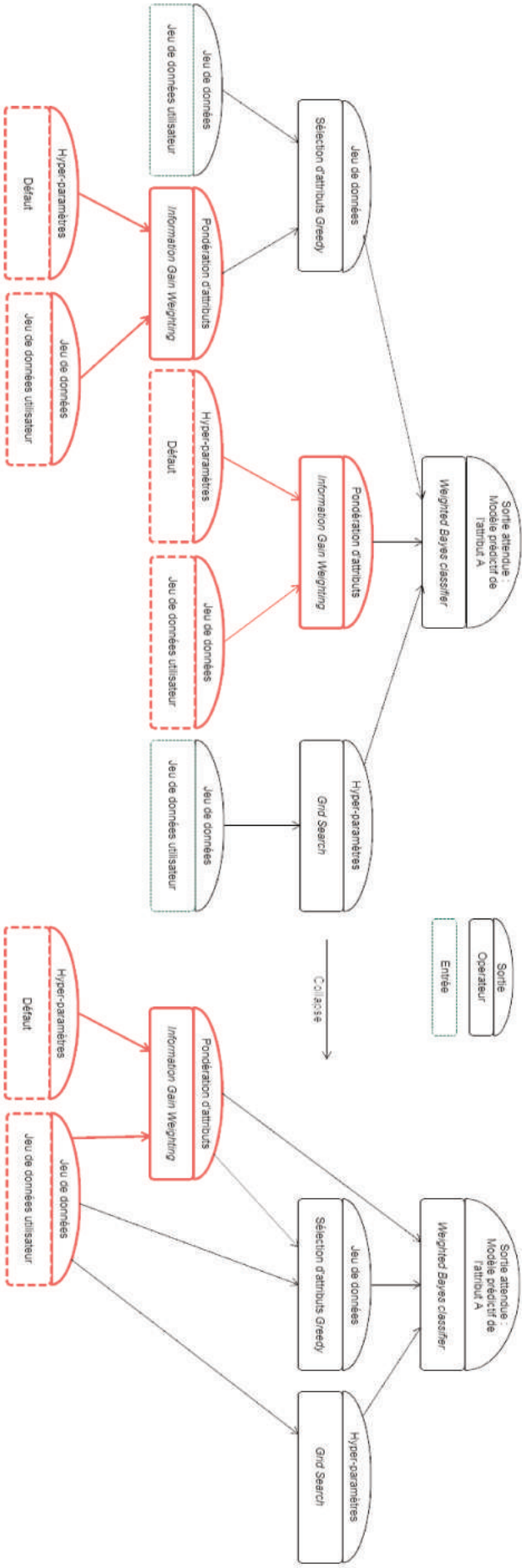

FIGURE 5.6 – Opération *Collapse*

permettant d'altérer le fonctionnement du classifieur. On illustre par le passage de ② à ③ une seconde application de l'opération **Expand**, cette fois considérant le jeu de données demandé par le classifieur.

Collapse. On se réfère ici à la Figure 5.6 (page 162). L'opération **Collapse** considère un *workflow* complet ne présentant plus de *sorties* "libres" (non associées à un opérateur pouvant les produire), et fusionne à partir de la base les chaînes d'opérateurs identiques. On voit par exemple en Figure 5.6 que le sous-graphe en rouge, présent en double dans le *workflow* avant le **Collapse**, ne s'y trouve plus qu'en un seul exemplaire, mais fournit toujours bien les *inputs* demandés par les opérateurs suivants.

Le fonctionnement de cette heuristique d'ajout récursif d'opérateur compatible peut alors se résumer comme présenté dans l'algorithme 5.

Algorithme 5 : Ajout récursif d'opérateur compatible

```
Partir d'un workflow vide
Ajouter la sortie attendue par l'utilisateur
tant que le workflow présente au moins une sortie "libre" faire
    Expand un sortie "libre"
Collapse
```

Une application pas-à-pas de cet algorithme commencerait en Figure 5.4 (page 160). En ① on considère la sortie attendue par l'utilisateur, que l'on **Expand** pour aller en ②. Pour passer de ② à ③ on choisi et **Expand** ensuite le *jeu de données*. On boucle ainsi sur les *sorties* "libres" jusqu'à se trouver dans la situation de la Figure 5.5 (page 161), et on **Collapse** le *workflow* ainsi obtenu (Figure 5.6 en page 162).

Le comportement de cette heuristique sera donc fortement impacté par la définition de l'opération **Expand**. Les solutions discutées précédemment se révèlent des variantes proches de cette heuristique basant leur **Expand** sur différents critères. Nguyen et al. (2014), par exemple, utilisent des techniques de fouille de motif pour constituer un répertoire de sous-chaîne d'opérateurs courantes, et **Expand** directement des sous-chaînes au lieu d'opérateurs uniques. Diamantini et al. (2009a,b) utilisent plutôt une ontologie détaillée des formats d'entrée-sortie pour **Expand** avec un opérateur adéquat. Dans notre cas, on veut utiliser la connaissance disponible dans les expériences passées jugées pertinentes que l'on a à disposition. On pourrait ainsi simplement imaginer un **Expand** qui irait chercher ses opérateurs compatibles dans les *workflows* des expériences pertinentes. On construit ainsi de nouveaux *workflows* utilisant des structures présentes dans des expériences passées jugées pertinentes.

5.2.3.3 Heuristique évolutionnaire

Une autre approche possible repose sur le parallèle que l'on peut faire entre notre base d'expériences passées et la population de solutions candidates d'une heuristique évolutionnaire (Hao et Solnon 2014). En effet, si l'on considère notre base d'expériences passées comme une population de solutions candidates, une itération du processus de méta-analyse décrit en Figure 5.2 (page 154) revient à faire *évoluer* la population pour la rapprocher d'un point désirable (ici une expérience performante sur le problème utilisateur). On peut alors pousser le parallèle pour prendre avantage des techniques développées dans le domaine des heuristiques évolutionnaires. Par exemple, en identifiant le processus de méta-analyse à un algorithme génétique, on pourrait définir notre opération de construction de nouvelles analyses basées sur les expériences passées *pertinentes* à l'aide d'opérateurs de **Crossing-over** et **Mutation** :

Crossing-over. Les opérations de *crossing-over* imitent typiquement l'échange de portions de chromosomes pouvant se produire lors de la reproduction. Dans le cadre des heuristiques évolutionnaires, cela se traduit par la combinaison de deux solutions, remplaçant une partie de la première par une partie de la seconde. On peut reproduire un tel comportement sur les *workflows* manipulés ici. On présente ainsi en Figure 5.7 (page 165) deux *workflows* extraits d'expériences passées pertinentes, que l'on souhaite combiner par cette opération de *crossing-over*. On identifie alors un noeud de chacun de ces *workflows*, tels que tous deux présentent un output semblable. On retire alors ce noeud (et potentiellement son sous-arbre propre) du premier *workflow* pour le remplacer par celui du second (en dupliquant le sous-arbre complet). Le *workflow* obtenu (à droite en Figure 5.7) sera souvent redondant. Une opération **Collapse** telle que décrite précédemment permet alors d'obtenir un résultat compact (voir Figure 5.8 en page 166). Cette opération permet donc la construction de nouveaux *workflows* pouvant présenter de bons résultats car construits sur ceux d'expériences jugées pertinentes.

Mutation. L'opération de mutation complète le parallèle évolutionnaire en imitant les modifications aléatoires de l'information génétique pouvant être transmises à la génération suivante. En termes heuristiques, cela consiste à apporter une modification aléatoire à quelques individus lors de chaque génération. Là encore, on peut définir une opération analogue sur les *workflows*. On présente en Figure 5.9 (page 167) un premier *workflow* (à gauche) extrait d'une expérience passée pertinente. On supprime alors un nœud de ce *workflow* (et potentiellement son sous-arbre propre) pour le

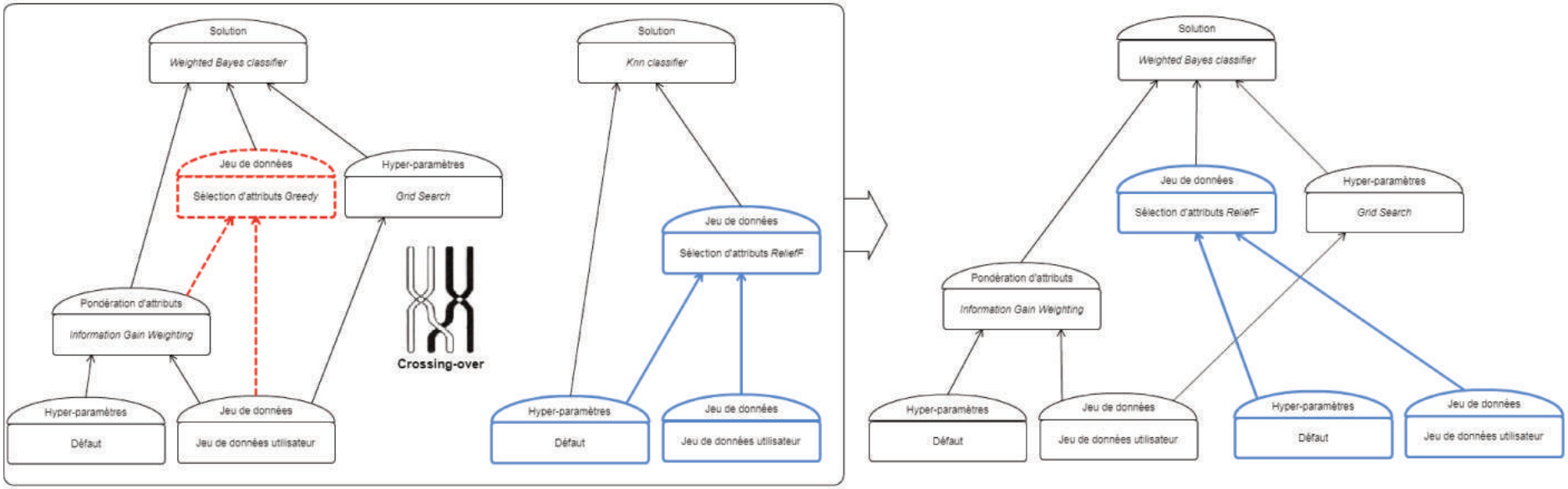

FIGURE 5.7 – Opération de Crossing-over

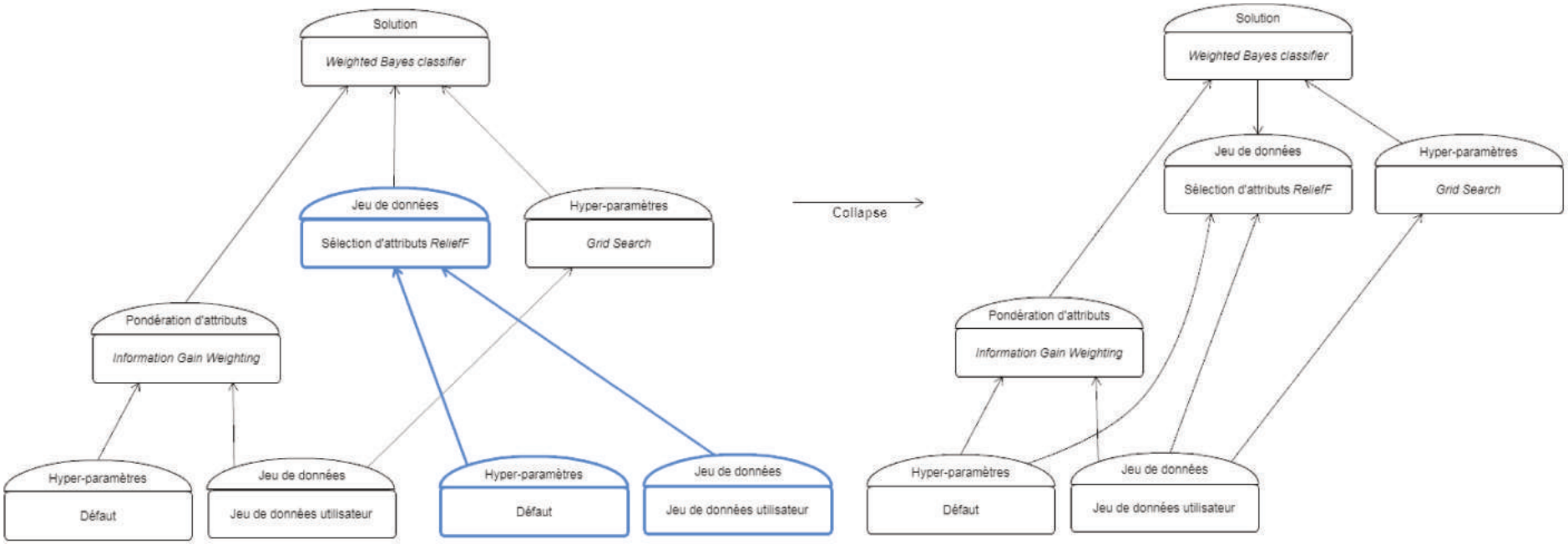

FIGURE 5.8 – Fin de Crossing-over

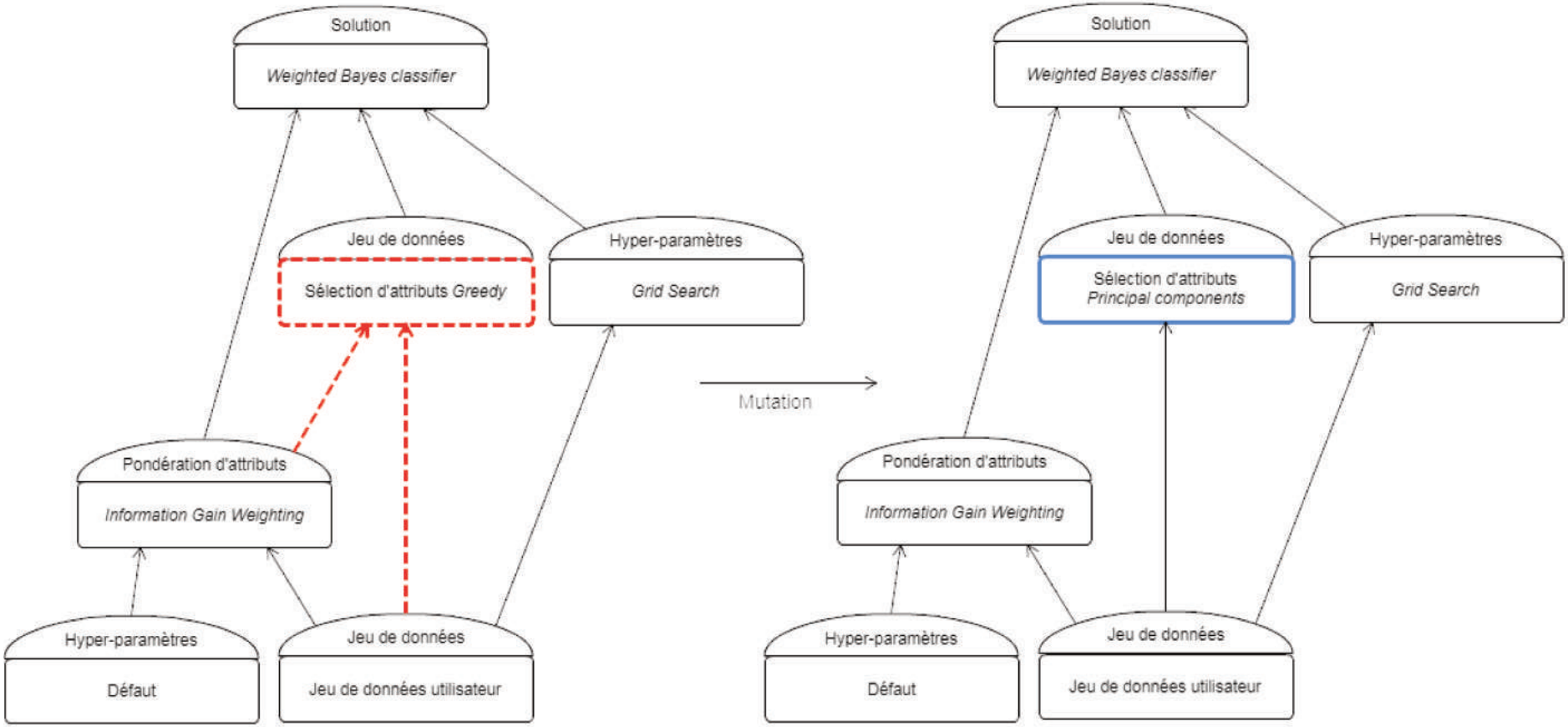

FIGURE 5.9 – Opération de Mutation

remplacer par un opérateur aléatoire capable de produire l'*output* attendu. Cette opération assure en général l'exploration de l'espace au fil des générations, ce qui permet souvent de s'extraire de minima locaux. Ici cela se traduit par l'expérimentation de nouveaux opérateurs dans les *workflows* d'expériences pertinentes, permettant de tester des combinaison potentiellement innovantes.

L'opération de construction de nouvelles analyses basées sur les expériences passées *pertinentes* consisterait alors, comme dans un algorithme génétique, à générer des *workflows* par **Crossing-over** entre certaines des analyses pertinentes, et à en faire muter une partie. La définition précise des proportions d'expériences à affecter par l'une ou l'autre opération, ainsi que les appariements à faire pour le **Crossing-over** restent des tâches potentiellement complexes. Le domaine des heuristiques évolutionnaires a en revanche été abondamment étudié, et peut fournir de bonnes solutions aux problèmes communs. En particulier, on peut s'intéresser aux particularités des algorithmes mémétiques (Smith 2007), qui améliorent la qualité des solutions individuelles par des techniques de recherches locales. On pourrait ainsi imaginer une étape d'*optimisation* des *workflows* générés, potentiellement testant et remplaçant des opérateurs insatisfaisants. Dans un autre registre, il serait possible de s'inspirer des heuristiques évolutionnaires multi-objectifs (Zhou et al. 2011), pour modifier la prise en compte des préférences utilisateur. Au lieu de faire évoluer l'ensemble de la population vers un objectif unique résumant ce besoin, on peut l'exprimer selon l'ensemble de ses composants et évoluer simultanément dans toutes ces directions. On dessine ainsi en quelque sorte un *front de Pareto* de solutions maximisant différents critères, permettant à l'utilisateur plus de flexibilité dans l'expression de son besoin et dans le choix de la réponse la plus adaptée.

5.3 Preuve de concept

Afin d'étudier la validité des approches proposées, on conçoit et implémente une preuve de concept très simple de l'assistance par méta-analyse proposée, utilisant l'approche la plus simpliste pour la construction d'expériences : la réutilisation directe (*cf.* 5.2.3.1). On restreint dans un premier temps le domaine d'analyse sur lequel assister l'utilisateur aux problèmes de classification supervisée. De même, on se restreint aux opérateurs et algorithmes de l'environnement Weka (Hall et al. 2009) pour construire les processus d'analyse. On utilisera la plateforme collaborative d'expériences d'apprentissage OpenML (Vanschoren et al. 2013) pour construire la base d'expérience passées, profitant des travaux déjà effectués dans la définition et le stockage d'expériences d'apprentissage.

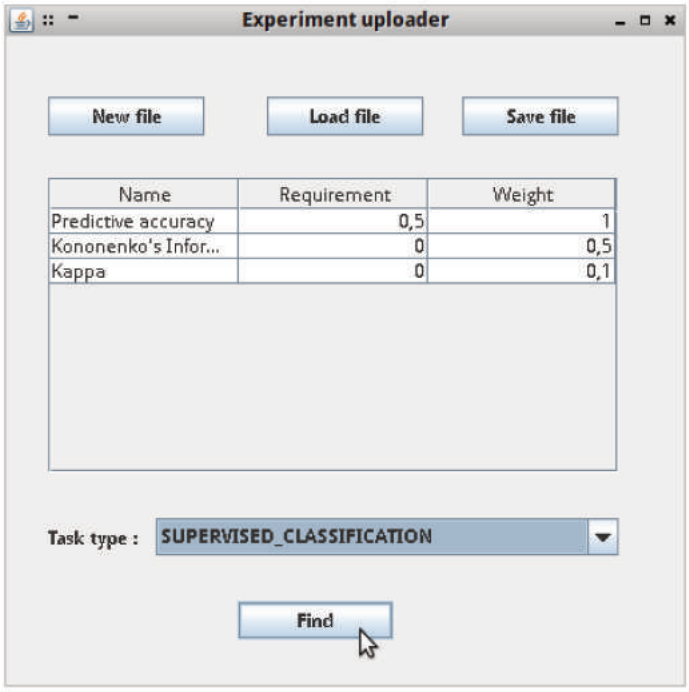

FIGURE 5.10 – Interface de construction de tâche

5.3.1 Implémentation

Reprenant le déroulement du cas d'utilisation présenté en Figure 5.2 (page 154), un utilisateur de l'assistant arrivant avec un jeu de données et un problème métier doit tout d'abord assurer l'expression de son problème sous la forme d'une tâche d'analyse (donc ici d'un problème de classification supervisée). On le supposera dans un premier temps capable d'assurer cette correspondance via l'interface de la Figure 5.10, mais l'assistance de l'utilisateur dans la spécification de cette tâche constituera une fonctionnalité cruciale d'un assistant complet. Dans cette preuve de concept, la spécification sous forme de tâche du besoin utilisateur prendra la forme d'une liste de critères d'évaluations de classification supervisée, chacun associé à une exigence (*requirement*), minimum demandé par l'utilisateur, et une pondération (*weight*), caractérisant l'importance relative de ce critère pour l'utilisateur. Par exemple, dans la spécification du besoin effectuée en Figure 5.10, on retrouve :

— Le critère de *Predictive accuracy*, mesurant le pourcentage de classification correcte, avec une exigence de $0,5$ et un poids de 1. Cela signifie que l'utilisateur exige du modèle produit au moins 50% de classifications correctes, et que c'est un critère important pour lui (poids maximal de 1).

— L'*Information Score* de Kononenko et Bratko (1991) mesure l'information produite par le modèle, différenciant classifications triviales et plus complexes. L'utilisateur a

spécifié une exigence de 0, signifiant que le modèle doit apporter plus d'information que ce qui est évident à première vue, et un poids de $0,5$, signifiant que ce critère est pour lui moins critique que le premier.

— Le Kappa de Cohen (1968) mesure l'accord entre prédictions et valeurs réelles en prenant en compte la possibilité de concordance contingente. On a ici un exigence de 0, demandant un accord plus fréquent que ce que le hasard donnerait, et un poids de $0,1$, reléguant ce critère à un point de décision subsidiaire.

L'assistant fait ainsi spécifier à l'utilisateur son jeu de données et son besoin en terme de performances pour construire la tâche de méta-analyse courante (Figure 5.10). On applique ensuite le principe de méta-analyse décrit en 5.2.1 pour trouver des expériences passées similaires en terme de données et de résultats. En particulier, on calcule la dissimilarité entre jeux de données selon l'approche décrite au chapitre 4, utilisant un appariement *ExactSplit* des méta-attributs des attributs (appariement exact considérant séparément attributs numériques et nominaux). L'adéquation des résultats d'expériences passées aux besoins utilisateur est évaluée de la manière suivante :

Définition 10 Dans le contexte d'une expérience passée connue $e \in E$ et d'un besoin utilisateur spécifié $b \in B$. Soit $c \in C$ l'ensemble des critères de performance applicables à la tâche en cours. Soient alors :

— $a_c \in \mathbb{R}$ la valeur du critère c mesurée lors de l'expérience e
— $\delta_c \in \{0, 1\}$ le booléen "*higher is better*" spécifiant le sens d'évolution positif de c
— $M_c \in \overline{\mathbb{R}}$ l'optimum théorique de c
— $\lambda_c \in \mathbb{R}$ le poids de c spécifié dans b
— $r_c \in \mathbb{R}$ la valeur minimale (ou maximale si $\delta_c = 0$) de c requise par b
— $C' \subseteq C$ l'ensemble des critères de performance employés pour spécifier b

On définit alors la différence entre un besoin utilisateur spécifié $b \in B$ et le résultat d'une expérience passée $e \in E$:

$$\Delta(e, b) = \frac{1}{\sum\limits_{c \in C'} \lambda_c} \sum_{c \in C'} \lambda_c * p(a_c)$$

Où $p(a_c) \in [0,1]$ est la pénalité sur le critère c pour la valeur a :

$$p(a_c) = \begin{cases} 1 & \text{si } \delta_c = 1 \text{ et } a_c < r_c \\ 1 & \text{si } \delta_c = 0 \text{ et } a_c > r_c \\ \dfrac{|M_c - a_c|}{\max\limits_{e \in E} |M_c - a_c|} & \text{si } M_c \in \mathbb{R} \\ \dfrac{(1 + |a_c|)^{-1}}{\max\limits_{e \in E} (1 + |a_c|)^{-1}} & \text{si } M_c \notin \mathbb{R} \end{cases}$$

Cette différence $\Delta(e, b)$ est ainsi une somme de pénalités sur les différents critères considérés, pondérée par les poids associés par l'utilisateur à chacun de ces critères. Pour un critère c, cette pénalité est maximisée (et vaut donc 1) si l'exigence r_c n'est pas atteinte. Si c admet un optimum réel, la pénalité est l'écart absolu à cet optimum, normalisé sur l'ensemble des expériences de E. Sinon, on prend un inverse décalé de a_c, toujours normalisé sur l'ensemble des expériences de E.

À l'aide de la dissimilarité entre jeux de données et de ce calcul d'adéquation des résultats d'expériences passées aux besoins utilisateur, on peut extraire de la base les expériences passées les plus similaires en termes de données et de résultats. Selon l'approche de réutilisation directe décrite en 5.2.3.1, on souhaite simplement recommander à l'utilisateur l'emploi du *workflow* de l'expérience passée ainsi jugée la plus *pertinente*. On extrait ainsi ce *workflow* pour le présenter à l'utilisateur dans l'interface *KnowledgeFlow* de *Weka* (voir Figure 5.11 en page 172), en l'adaptant bien sûr au jeu de données utilisateur. Cette interface donne un bon aperçu du processus d'analyse recommandé, et permet de l'exécuter en un clic. De plus, l'utilisateur peut directement modifier le processus s'il le souhaite, en modifiant les paramètres de certains opérateurs, voire en en remplaçant. L'utilisateur peut ainsi par exemple modifier de manière intuitive la visualisation des résultats d'analyse sans avoir à recommencer le procédé complet, ou bien construire un processus d'analyse plus complexe en utilisant la recommandation comme base...

5.3.2 Validation

Afin de valider cette preuve de concept, on conçoit une expérience simple de recommandation d'algorithme de classification. On construit une base d'expériences passées en appliquant 10 algorithmes de classification classiques de l'état de l'art sur 10 tâches de classification d'OpenML (voir en Annexe A Tables A.19 et A.18). Les tâches ont été choi-

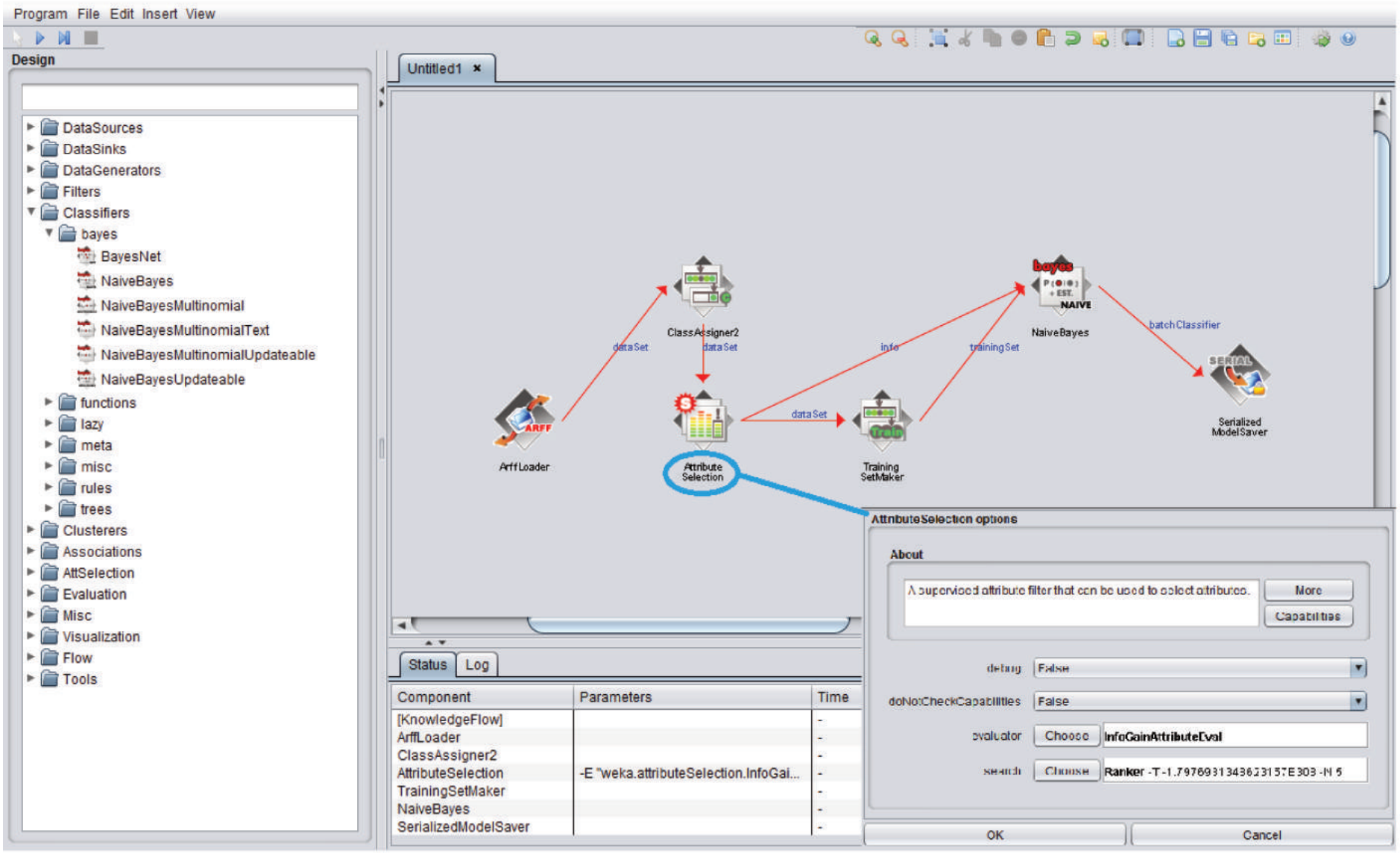

FIGURE 5.11 – Processus recommandé dans le *KnowledgeFlow Weka*

sies de sorte à représenter divers problèmes de classification (binomiale et multinomiale) sur un ensemble de jeux de données de divers domaines. Les algorithmes de classification ont également été sélectionnés de sorte à présenter une réelle diversité de zones d'expertise, et représentent ainsi un bon échantillon des techniques majeures de la littérature.

L'objectif est ensuite d'utiliser cette base d'expériences passées pour choisir, sur 5 nouvelles tâches de classification d'OpenML, lequel de nos 10 algorithmes de classification recommander. On comparera la performance de cette preuve de concept à 3 autres méthodes de sélection. Ces dernières ont été définies de sorte à imiter des stratégies souvent employées par des analystes de différents domaines pour choisir un méthode de classification à appliquer sur un problème donné (Obenshain 2004 ; Palaniappan et Awang 2008) :

1. **Connaissance domaine** : Cette stratégie consiste à sélectionner un classifieur sur la base de connaissances du domaine trouvée en ligne ou dans la littérature. Un utilisateur non-expert en analyse de données se référera souvent au premier ensemble de *"guidelines"* qu'il pourra trouver par une rapide recherche. Par exemple, un analyste financier pourra facilement choisir d'employer un classifieur Bayésien naïf après avoir lu que ce dernier se comporte mieux dans une situation d'indépendance entre attributs, s'il pense cette hypothèse valide sur ses données. En pratique, on simulera cette stratégie en demandant à un groupe d'experts en analyse de données de sélectionner un classifieur adapté à chaque tâche en utilisant leur connaissance et expérience du domaine. Ce protocole biaise naturellement les performances de cette stratégie, mais il s'agit d'un biais positif : nos utilisateurs ont une meilleure expérience que ceux dont on simule le comportement, ce qui est donc acceptable pour une *baseline* à surpasser.

2. **Sélection du meilleur** : Cette stratégie consiste à mesurer sur la base d'expériences passées quel algorithme a en moyenne présenté les meilleures performances. Ce procédé couramment employé par des utilisateurs ayant une connaissance superficielle de l'apprentissage automatique, et se base sur l'hypothèse que certains algorithmes sont meilleurs que d'autres dans l'absolu. Les *No free lunch theorems* de D. H. Wolpert (1996) ont prouvé cette hypothèse invalide, mais la stratégie de **Sélection du meilleur** peut néanmoins amener à des choix parfaitement raisonnables dans certains cas. En effet, si les jeux de données constituant la base d'expériences passées sont raisonnablement semblables à celui considéré, ils auront une compatibilité similaire aux algorithmes disponibles, menant à des choix acceptables.

3. **"*Subsampling*"** : Cette stratégie, souvent employée par des analystes se retrouvant face à des volumes de données plus importants que ce qu'ils sont habitués à traiter, consiste à mesurer la performance réelle des algorithmes disponibles sur une fraction (100 instances) du jeu de données courant. On sélectionne ainsi l'algorithme le plus performant sur ce "*subsample*" du jeu de données, émettant l'hypothèse qu'il sera toujours performant sur le jeu de données complet. Cela n'est bien sûr pas nécessairement le cas, mais la performance mesurée par *subsampling* peut dans de nombreux cas se révéler un estimateur acceptable de la performance réelle, et permettre des choix satisfaisants. On peut remarquer que dans ce cadre expérimental, cette approche "triche", en faisant usage d'informations *a posteriori* (elle entraîne des classifieurs sur une portion du jeu de données étudié), alors que toutes les autres n'utilisent que des informations disponibles *a priori*. Il s'agit là encore d'un biais positif acceptable dans la mesure où ce *subsampling* constitue une *baseline* à surpasser.

On évalue la performance de la recommandation selon le critère présenté au chapitre 3, dont on rappelle simplement l'interprétation : une performance au méta-niveau de 0.87 signifie que le classifieur recommandé a présenté une performance réelle équivalant 87% de celle du classifieur s'étant le mieux comporté sur le jeu de données courant. Ceci permet une comparaison simple et intuitive des stratégies de recommandation. On évalue ainsi la performance de la preuve de concept d'assistant décrit dans la section précédente, ainsi que celles des trois stratégies de sélection présentée ci-avant, sur 5 nouvelles tâches de classification (listées en Annexe A, Table A.18) pour présenter leurs performances respectives en Table 5.1.

Tâche de test	**Connaissance domaine**	**Sélection du meilleur**	***Subsampling***	**Méta-analyse (*POC*)**
Electricity	0.460	0.652	0.657	**0.998**
Spambase	**0.966**	0.880	0.861	0.880
Iris	0.959	0.969	**1.000**	0.969
Anneal	0.803	0.852	**1.000**	0.852
Car	**0.876**	0.810	0.835	0.810
Mean	**0.813**	**0.833**	**0.871**	**0.902**

TABLE 5.1 – Performance des différentes stratégies de recommandation

On peut remarquer que la stratégie de **Connaissance domaine** mène à des résultats peu consistants et facilement extrêmes. Ceci est raisonnable dans la mesure où l'information employée pour faire cette sélection est par nature incomplète, et souvent difficile à

interpréter pour des non-experts. C'est en effet l'un des obstacles majeurs à l'accessibilité de l'analyse pour des non-experts : la connaissance du méta-domaine (le *savoir-faire* permettant de construire, tester et comparer des processus d'analyse adaptés) est implicite et très variable, car faisant partie de l'expérience des analystes. De plus, l'expérience acquise dans l'analyse des données d'un domaine précis peut facilement se révéler inapplicable à un autre, pour peu que la topologie des données employées diffère trop...

La stratégie de **Sélection du meilleur** fait montre de performances légèrement supérieures en exploitant de manière répétée l'algorithme s'étant montré le plus compétitif sur la base d'expériences passées. La faille de cette stratégie reste bien sûr sa grande sensibilité à la composition de cette base : l'information qu'elle apporte sur la performance des algorithmes n'est valide que si la donnée courante est semblable à celle de la base. Aucun algorithme ne pouvant surclasser les autres dans l'absolu (D. H. Wolpert 1996), cette stratégie ne peut avoir de sens que sur de petits ensembles de jeux de données semblables, ce qui la rapprocherait alors considérablement de notre approche de méta-analyse...

Le ***Subsampling*** donne de bons résultats, mais, comme stipulé précédemment, fait usage d'information *a posteriori* pour effectuer ses recommandations. On peut également remarquer que cette stratégie a permis dans deux cas d'identifier l'algorithme optimal (performance au méta-niveau de 1), mais dans un autre s'est montré relativement faible (0.657). Ceci révèle une faiblesse de cette stratégie : la sensibilité au nombre d'instances. En effet, les jeux de données *Iris* et *Anneal* sont relativement petits (moins de 1000 *instances*) ce qui rend l'estimation par *Subsampling* très fidèle à la réalité, vu qu'elle est alors faite sur une portion significative du jeu de données. Pour des jeux de données plus importants (comme *Electricity* dépassant les 45.000 instances) l'estimation est faite sur une fraction négligeable du jeu de données et peut donc être très éloignée de la réalité.

Enfin, bien que les recommandations obtenues par **méta-analyse** soient rarement les meilleures (dans un seul des 5 cas de test), on peut remarquer que leur performance moyenne au méta-niveau est supérieure et à celle des autres stratégies, passant la barre des 90%. Les résultats semblent plus consistant que ceux des autres stratégies, suggérant une plus grande stabilité, qui sera néanmoins à investiguer plus avant. Il faut de plus souligner que les stratégies de la *baseline* font au plus un usage limité de la base d'expériences passées, alors que la méta-analyse repose intégralement dessus. Cela implique que la croissance de cette base au fil des expériences permettra une régulière amélioration des performance de la méta-analyse, dont les autres stratégies ne sauraient bénéficier. Cette relation entre le contenu de la base et la qualité de la méta-analyse reste à étudier plus précisément, mais

l'intérêt de l'approche est apparent, même avec les choix naïfs de cette preuve de concept.

5.4 Conclusion

On a donc proposé une méthode de méta-analyse reposant sur les dissimilarités entre jeux de données présentées au chapitre 4, permettant de faire usage des *workflows* les plus *pertinents* utilisés par le passé pour les recommander ou en construire de nouveaux. Une preuve de concept recommandant l'emploi du *workflow* passé le plus pertinent a été développée, et a permis de valider l'intérêt de l'approche de méta-analyse envisagée. De nombreuses pistes restent cependant à explorer :

— L'implémentation des méthodes de constructions plus avancées décrites en 5.2.3 devrait permettre d'améliorer encore davantage les performances au méta-niveau. De plus, ces méthodes offrent la possibilité d'innover, c'est à dire de proposer des *workflows* encore jamais utilisés mais susceptibles de bonnes performances. Ceci laisse envisager un important potentiel : si les *workflows* construits sont *nouveaux*, une itération du processus de méta-analyse effectue une réelle exploration de l'espace d'analyse, améliorant d'autant plus vite la performance des futures itérations.

— L'implémentation actuelle se limite à l'environnement de construction de *workflows* de Weka, mais cela n'est en aucun cas contraint par le modèle sous-jacent. Le développement de l'assistant pour inclure d'autres environnements serait très souhaitable pour maximiser l'accessibilité par tout profil d'utilisateur. Cette extension aurait de plus l'intéret de permettre la traduction de *workflows* entre les différents environnements, qui pose aujourd'hui problème aux diverses communautés tentant de poursuivre des analyses collaboratives dans des environnements différents.

Bibliographie du chapitre

Cohen, Jacob (1968). « Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. » In : *Psychological bulletin* 70.4, p. 213 (cf. p. 27, 28, 62, 100, 170).

Diamantini, Claudia, Domenico Potena et Emanuele Storti (2009a). « Kddonto : An ontology for discovery and composition of kdd algorithms ». In : *Third Generation Data Mining : Towards Service-Oriented Knowledge Discovery (SoKD)*, p. 13–24 (cf. p. 41, 42, 159, 163).

— (2009b). « Ontology-driven KDD process composition ». In : *Advances in Intelligent Data Analysis VIII*. Springer, p. 285–296 (cf. p. 41, 42, 159, 163).

Gama, Joao et Pavel Brazdil (1995). « Characterization of classification algorithms ». In : *Progress in Artificial Intelligence*. Springer, p. 189–200 (cf. p. 30, 31, 153).

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann et Ian H Witten (2009). « The WEKA data mining software : an update ». In : *ACM SIGKDD explorations newsletter* 11.1, p. 10–18 (cf. p. 66, 100, 168).

Hao, Jin-Kao et Christine Solnon (2014). *Méta-heuristiques et intelligence artificielle*. Rapp. tech. LIRIS - Laboratoire d'InfoRmatique en Image et Systemes d'information, p. 1–19. url : `https://hal.archives-ouvertes.fr/hal-01313162` (visité le 01/04/2017) (cf. p. 164, 188).

Kononenko, Igor et Ivan Bratko (1991). « Information-Based Evaluation Criterion for Classifier's Performance ». In : *Machine Learning* 6.1, p. 67–80. issn : 0885-6125 (cf. p. 27, 28, 63, 169).

Nguyen, Phong, Melanie Hilario et Alexandros Kalousis (2014). « Using meta-mining to support data mining workflow planning and optimization ». In : *Journal of Artificial Intelligence Research*, p. 605–644 (cf. p. 41, 42, 151, 159, 163).

Obenshain, Mary K (2004). « Application of data mining techniques to healthcare data ». In : *Infection Control* 25.08, p. 690–695 (cf. p. 25, 42, 153, 173).

Palaniappan, Sellappan et Rafiah Awang (2008). « Intelligent Heart Disease Prediction System Using Data Mining Techniques ». In : *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*. AICCSA. Washington, DC, USA : IEEE Computer Society, p. 108–115. isbn : 978-1-4244-1967-8. doi : `10.1109/AICCSA.2008.4493524`. url : `http://dx.doi.org/10.1109/AICCSA.2008.4493524` (visité le 01/04/2017) (cf. p. 25, 173).

Serban, F (2013). « Toward effective support for data mining using intelligent discovery assistance ». Thèse de doct. University of Zurich (cf. p. 41, 42, 77, 159).

Serban, Floarea, Joaquin Vanschoren, Jörg-Uwe Kietz et Abraham Bernstein (2013). « A survey of intelligent assistants for data analysis ». In : *ACM Computing Surveys (CSUR)* 45.3, p. 31 (cf. p. 17, 38, 41, 42, 55, 151).

Smith, Jim E (2007). « Coevolving memetic algorithms : a review and progress report ». In : *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on* 37.1, p. 6–17 (cf. p. 168).

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl et Luis Torgo (2013). « OpenML : Networked Science in Machine Learning ». In : *SIGKDD Explorations* 15.2, p. 49–60. doi : 10.1145/2641190.2641198. url : http://doi.acm.org/10.1145/2641190.2641198 (visité le 01/04/2017) (cf. p. 28, 31, 38, 57, 60, 168).

Wolpert, David H (1996). « The lack of a priori distinctions between learning algorithms ». In : *Neural computation* 8.7, p. 1341–1390 (cf. p. 29, 95, 173, 175).

Zakova, Monika, Petr Kremen, Filip Zelezny et Nada Lavrac (2011). « Automating knowledge discovery workflow composition through ontology-based planning ». In : *Automation Science and Engineering, IEEE Transactions on* 8.2, p. 253–264 (cf. p. 41, 42, 55, 77, 151, 159).

Zeevi, David et al. (2015). « Personalized Nutrition by Prediction of Glycemic Responses ». In : *Cell* 163.5, p. 1079–1094 (cf. p. 42, 152).

Zhou, Aimin, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan et Qingfu Zhang (2011). « Multiobjective evolutionary algorithms : A survey of the state of the art ». In : *Swarm and Evolutionary Computation* 1.1, p. 32–49 (cf. p. 168).

L'éternité, c'est long.
Surtout vers la fin.

Woody Allen

where doing it man
where MAKING THIS HAPPEN

Sweet Bro

Chapitre 6

Bilan et Perspectives

6.1 Bilan

Afin d'atteindre nos objectifs et de proposer de nouvelles approches performantes de méta-analyse pour l'assistance à l'analyse de données, différents verrous ont dû être levés. On rappellera ici ce cheminement et les contributions qui en ont résulté.

Évaluation de méta-analyses - Trouver des approches performantes de méta-analyse nécessite de déterminer ce qu'est la *performance* d'une méta-analyse. Le premier verrou à adresser consistait donc en la construction d'une procédure d'évaluation adaptée aux spécificités de la méta-analyse. L'intuition suivie pour ce faire fut de lier la performance de la méta-analyse à la performance relative de l'analyse finalement fournie à l'utilisateur. Nous introduisons ainsi un cadre d'évaluation et de comparaison adapté aux spécificités de la méta-analyse de données, basé sur un critère de performance unifié. Une importante série d'expériences de méta-apprentissage a été réalisée pour démontrer la praticabilité de ce cadre d'évaluation. Nous proposons alors une méthodologie d'exploration dimensionnelle des résultats, permettant de répondre à des questions précises et de qualifier la validité des connaissances produites à l'aide de tests d'hypothèses statistiques. Ces derniers permettent par ailleurs d'étudier, par une visualisation intuitive, diverses questions que l'on peut se poser sur l'analyse de données, comme illustré au chapitre 3 par l'analyse des méthodes de sélection d'attributs à employer au méta-niveau.

Ces contributions ont donné lieu à publication en conférences nationales (Raynaut et al. 2017a) et internationales (Raynaut et al. 2016c). Le cadre d'évaluation produit a de plus été ré-employé systématiquement pour évaluer les différentes propositions et améliorations possibles au méta-niveau, lors de la levée des verrous suivants.

Dissimilarité entre jeux de données - Afin de permettre de nouvelles approches de méta-analyse, nous nous sommes ensuite intéressés à un verrou majeur du domaine : la caractérisation de jeux de données. Suite au constat d'une importante perte d'information dans les méthodes de caractérisation communément employées, nous avons proposé l'emploi de fonctions de dissimilarité entre jeux de données. Nous avons défini un ensemble de propriétés désirables pour proposer des fonctions capables de prendre en compte l'entièreté de l'information disponible, ce qui passe par la caractérisation des attributs particuliers de ces jeux de données. Nous avons ensuite montré que ces dissimilarités permettent de caractériser l'adéquation d'algorithmes de classification avec des jeux de données plus efficacement que des distances traditionnelles, et qu'elles peuvent être employées avec de bonnes performances dans un contexte de régression au méta-niveau pour sélection d'algorithme. Au-delà du simple intérêt de l'approche, nos évaluations par analyses dimensionnelles ont permis d'étudier en détail l'impact des différents facteurs et composants de ces fonctions de dissimilarité sur la performance au méta-niveau. On peut ainsi proposer un candidat qualifié comme brique de base pour de nouvelles approches de méta-analyse.

A divers stades de maturité, ces contributions ont donné lieu à publication en conférences nationales (Raynaut et al. 2016a,b,d) et internationales (Raynaut et al. 2015). L'intégralité des expériences réalisées a également donné lieu à l'émission d'un rapport technique (Raynaut et al. 2017b) et sollicitation pour publication en revue nationale (numéro spécial BDA de la revue ISI). Ces fonctions de dissimilarité ont enfin permis la réalisation d'un prototype d'assistant à l'analyse de données basé sur de nouvelles approches de méta-analyse.

Méta-attributs de jeux de données - En parallèle avec le travail sur les dissimilarités, nous nous sommes intéressés à un autre aspect du verrou de caractérisation des jeu de données : les méta-attributs de jeux de données. On peut en effet recenser de très nombreuses mesures caractérisant diverses *propriétés* de jeux de données, mais leur intérêt pour la méta-analyse, bien qu'avéré, reste souvent imparfaitement compris. Nos recherches sur le sujet ont mené à une collaboration avec le groupe de recherche international *OpenML* ayant pour projet de fournir une plateforme collaborative d'expériences d'apprentissage. Un des objectifs principaux de l'initiative *OpenML* étant de permettre diverses approches naturelles de méta-analyse, la définition et l'implémentation d'ensembles complets de méta-attributs fut une contribution importante à l'avancée du projet [1]. La liste complète des méta-attributs définis et implémentés pour le projet est disponible en an-

1. Voir `https://www.openml.org/index.php/search?q=%2520measure_type%3Adata_quality`

nexe, Tables A.8, A.9, A.10, A.11, A.12, A.13, A.14 et A.15. Ces contributions ont de plus permis la réalisation de quelques expériences préliminaires d'analyse et de comparaison de ces méta-attributs (Voir 6.2.1).

Assistance à l'analyse de données - Nous avons ensuite proposé une méthode de méta-analyse reposant sur les dissimilarités entre jeux de données présentées au chapitre 4, permettant de faire usage des *workflows* les plus *pertinents* utilisés par le passé pour en recommander ou en construire de nouveaux. Une preuve de concept recommandant simplement l'emploi du *workflow* passé le plus pertinent a été développée, et a permis de valider l'intérêt de l'approche de méta-analyse envisagée.

Nombre des solutions proposées n'ont pas encore été implémentées, mais devant les résultats encourageants des expériences préliminaires effectuées, leur développement est appelé à se poursuivre. Les contributions et propositions détaillées au cours des différents chapitres ont donné lieu à de nombreuses perspectives d'amélioration pour l'assistance à l'analyse de données, mais nombres d'entre elles présentent le potentiel de sujets de recherches indépendants, requérant un travail important. Nous détaillerons ainsi dans les paragraphes suivants les principales opportunités de recherche se présentant maintenant.

6.2 Perspectives

Notre première perspective de recherche concerne les méta-attributs de jeux de données. S'agissant d'un point fondamental des approches proposées, quelques expériences préliminaires ont été réalisées pour une meilleure compréhension des verrous du sujet. On présente ainsi ces expériences et leur résultats en 6.2.1. Une autre perspective intéressante a été évoqué au chapitre 5, dans la capacité d'innovation des méthodes d'assistances, qui sera ainsi discutée ensuite. Enfin, bien que nos propositions constituent un noyau fonctionnel à un assistant intelligent à l'analyse de données, d'importants travaux seront nécessaires pour améliorer le rapport à l'utilisateur en amont (définition du besoin et du problème) comme en aval (compréhension et acceptation des résultats). Ces verrous pourront être approchés sous plusieurs angles dont nous essaierons alors d'offrir un bon aperçu.

6.2.1 Méta-attributs de jeux de données

Comme nous avons pu le constater dans les résultats obtenus en 4.4 et dans la litérature (section 2.4 de l'état de l'art), le choix des méta-attributs joue un rôle important pour la performance au méta-niveau. Afin d'investiguer la nature de cette relation, nous avons

effectué quelques expériences préliminaires sur les méta-attributs définis et implémentés pour OpenML (Voir 6.1).

Nous avons construit l'expérience autour de deux objectifs principaux :

1. Déterminer si certains méta-attributs sont plus *utiles* que d'autres au méta-niveau.
2. Étudier l'influence du nombre de méta-attributs utilisés sur la performance au méta-niveau.

Pour ce faire, nous utilisons le cadre d'évaluation du chapitre 3, sur un simple méta-problème de sélection d'algorithmes de classification. Nous construisons un *metadataset* restreint de 25 classifieurs par 100 jeux de données (listes disponibles en annexe, Tables A.17 et A.16), utilisant le critère d'aire sous la courbe de ROC, identifié comme supérieur dans les expériences du chapitre 4. Nous résolvons ce méta-problème par une régression au méta-niveau (selon l'algorithme 3) basée sur une dissimilarité de type *ExactSplit* utilisant les facteurs secondaires identifiés comme optimaux en 4.4. Nous faisons ensuite varier l'ensemble de méta-attributs considérés entre les expériences. L'exploration complète de l'espace combinatoire des méta-attributs étant prohibitivement coûteuse (2^{166} combinaisons de méta-attributs possibles), nous sélectionnons les ensembles de méta-attributs à tester par tirage aléatoire. Sur les 2 millions d'expériences ainsi réalisées, le nombre d'évaluations de chaque méta-attribut est ainsi sensiblement égal (au plus 3% d'écart entre les plus et moins représentés). Afin de déterminer si certains méta-attributs sont plus *utiles* que d'autres au méta-niveau, on compare, pour chaque méta-attribut, la performance moyenne des expériences ayant utilisé ce méta-attribut. Les résultats ainsi obtenus sont présentés en Figure 6.1 pour les méta-attributs généraux et en Figure 6.2 pour les méta-attributs des attributs.

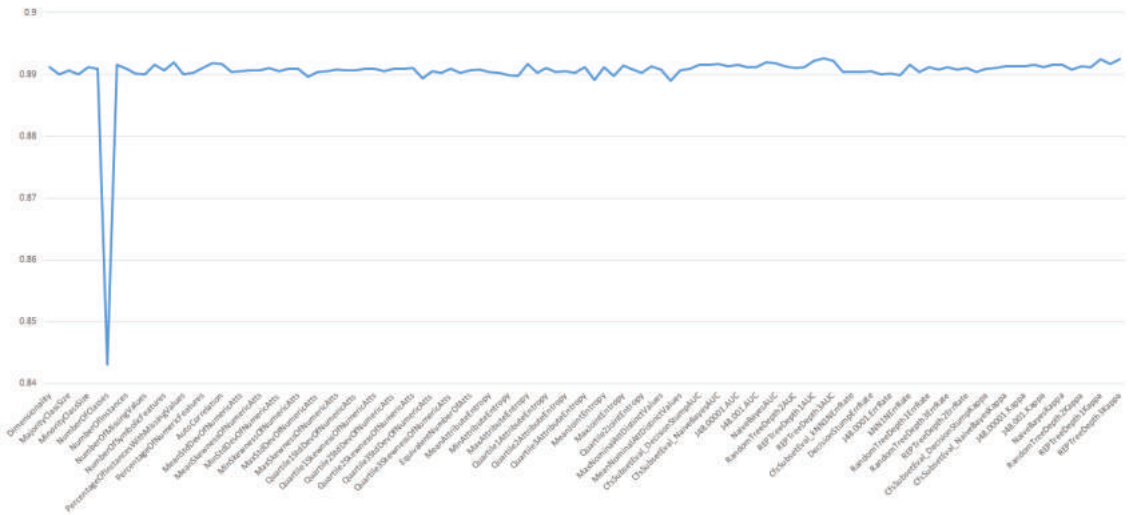

FIGURE 6.1 – Performance moyenne des expériences ayant utilisé les différents méta-attributs généraux

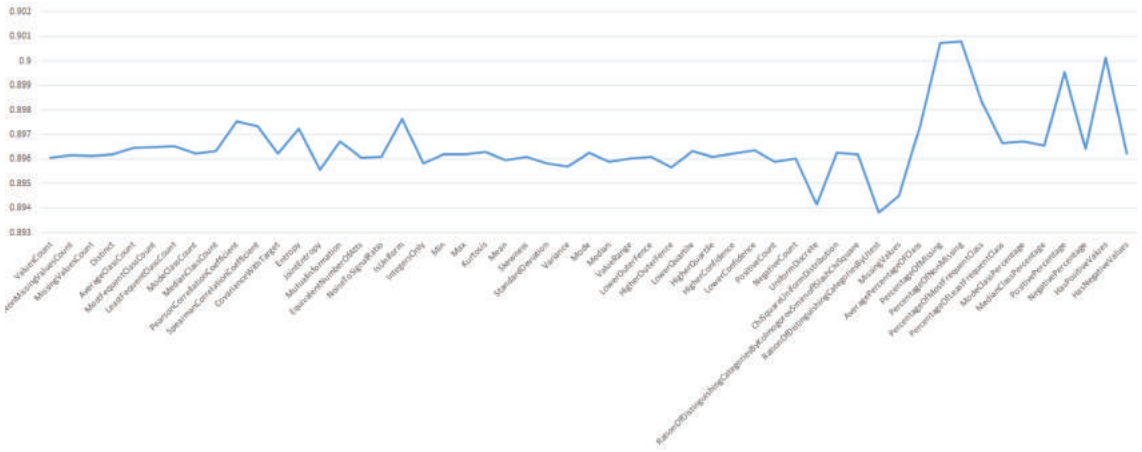

FIGURE 6.2 – Performance moyenne des expériences ayant utilisé les différents méta-attributs des attributs

Parmi les méta-attributs généraux présentés en Figure 6.1, seul le *nombre de classes* paraît mener à des performances sensiblement inférieures. Ceci paraît naturel dans la mesure ou les jeux de données du *metadataset* sont dédiés à de la classification binaire (le *nombre de classes* est toujours de 2). Ce méta-attribut non pertinent est donc bien identifié comme tel par l'expérience, ce qui rend d'autant plus étonnante l'apparente équivalence de tous les autres méta-attributs généraux. Ceci pourrait être attribué aux groupes de méta-attributs fortement dépendants. On trouve en effet parmi nos 113 méta-attributs généraux de nombreux groupes de méta-attributs fortement dépendants, comme par exemple ceux décrivant la distribution d'une mesure sur les attributs (*MeanJointEntropy*, *MinJointEntropy*, *MaxJointEntropy*, *Quartile1JointEntropy*, *Quartile2JointEntropy*, et *Quartile3JointEntropy*), ou les différentes évaluations d'un même *landmarker* (*NaiveBayesAUC*, *NaiveBayesErrRate*, et *NaiveBayesKappa*). Les tirages aléatoires dans l'espace des 113 méta-attributs généraux ont ainsi de fortes chances de piocher des méta-attributs de plusieurs de ces groupes, et donc de présenter une information très similaire, lissant les performances au méta-niveau.

Parmi les méta-attributs des attributs présentés en Figure 6.2, on peut remarquer une apparente supériorité des méta-attributs normalisés figurant à l'extrême droite. Il s'agit principalement de méta-attributs booléens (comme *HasPositiveValues*) ou de pourcentages (comme *PercentageOfMissing* ou *PercentageOfMostFrequentClass*), introduits initialement par Smid (2016) et déjà constatés intéressants dans le chapitre 4. Malheureusement, l'exploration complète de l'espace combinatoire des méta-attributs est, comme nous l'avons vu plus haut, prohibitivement coûteuse, ce qui interdit l'emploi de tests statistiques comparant des échantillons appariés, et on ne peut donc pas valider la significance de ce résultat. Au vu du nombre d'expériences effectuées et du lissage constaté des résultats, on peut

cependant conjecturer que les plus importantes différences observées ne sont pas fortuites, mais qu'au contraire une grande partie des différences réelles s'est vue *gommée* par ce lissage.

La comparaison directe de méta-attributs produit donc des résultats mitigés, mais notre second objectif était d'étudier l'influence du *nombre* de méta-attributs employés. Nous présentons ainsi en Figure 6.3 la performance moyenne au méta-niveau selon le nombre de méta-attributs généraux utilisés, et de même en Figure 6.4 selon le nombre de méta-attributs des attributs.

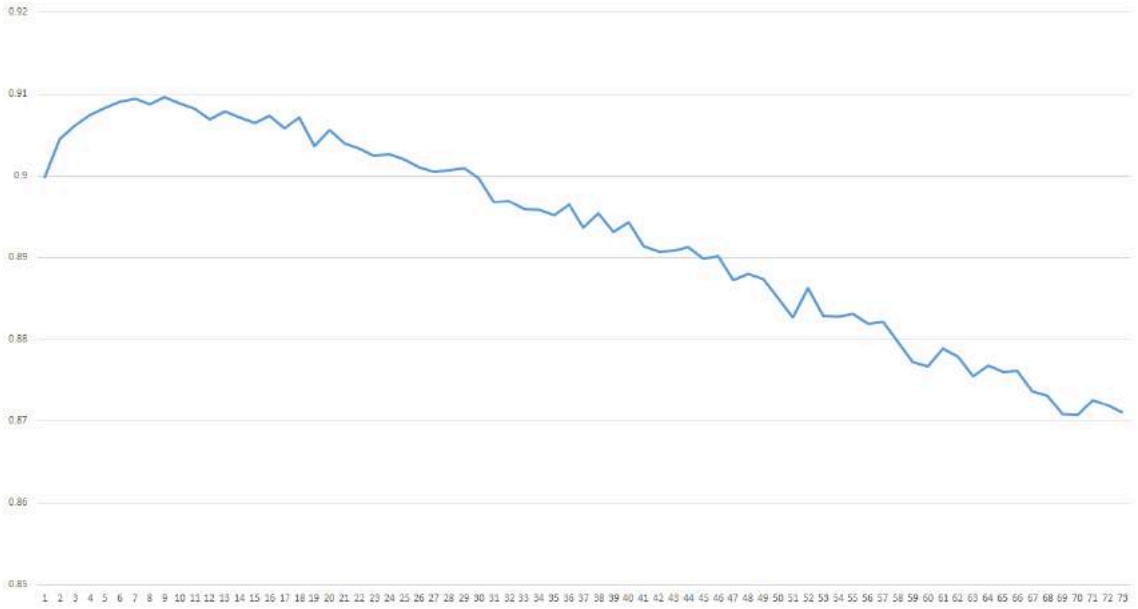

FIGURE 6.3 – Performance moyenne selon le nombre de méta-attributs généraux utilisés

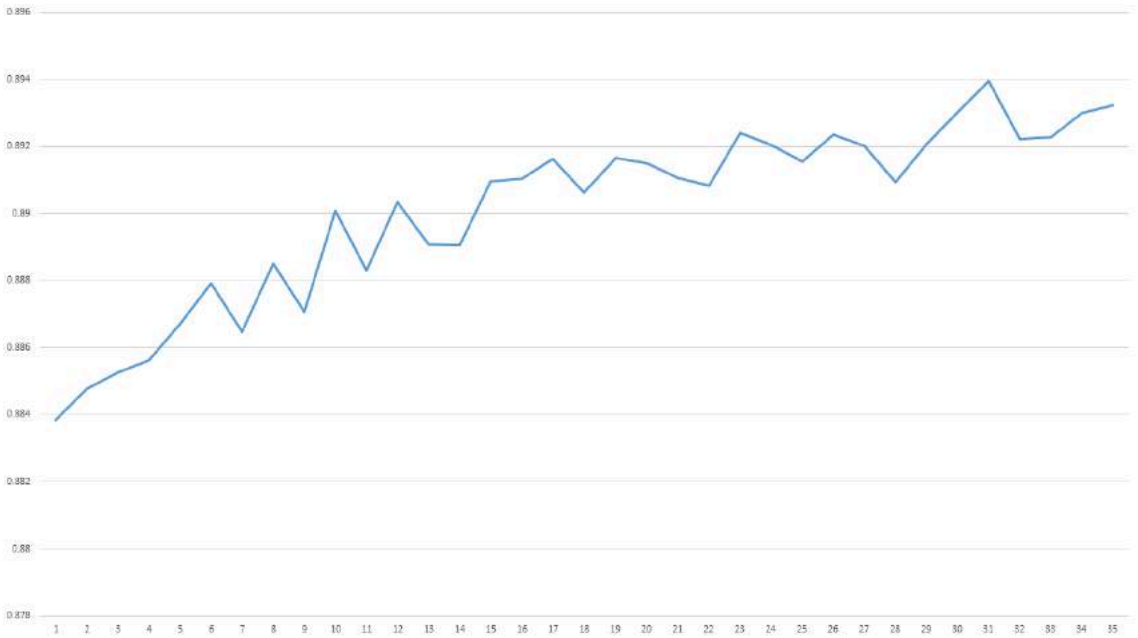

FIGURE 6.4 – Performance moyenne selon le nombre de méta-attributs des attributs utilisés

En Figure 6.3, on observe un plateau de performance optimale lors de l'utilisation de 8 à 12 méta-attributs généraux, ce qui vient corroborer les résultats antérieurs de dégradation des performances par de trop grands ensembles de méta-attributs (voir en 4.4 et dans la section 2.4 de l'état de l'art). Pour les méta-attributs des attributs en Figure 6.3, on constate à l'inverse une apparente amélioration des performances avec le nombre de méta-attributs. Ceci pourrait s'expliquer par le moins grand nombre de méta-attributs fortement dépendants parmi les méta-attributs des attributs. En effet, si ces différents méta-attributs apportent une information réellement différente et pertinente, en utiliser davantage devrait permettre d'améliorer la performance, si tant est que l'on soit capable d'utiliser l'entièreté de l'information apportée. Ce constat vient alors directement corroborer notre hypothèse initiale ayant mené à l'introduction de la dissimilarité : la perte d'information est fortement préjudiciable à la caractérisation de jeux de données.

La Figure 6.3 semble présenter un autre motif surprenant vis-à-vis du nombre de méta-attributs des attributs utilisés. Un nombre *pair* de méta-attributs des attributs semble très souvent mener à des performance *inférieures* au nombre *impair* suivant, malgré la tendance globale à la hausse. Cela donne cet aspect en dent de scie observable en plusieurs points de la courbe. Ce résultat peu naturel défie l'interprétation, et vient renforcer le besoin d'expériences plus approfondies en la matière.

La compréhension et l'optimisation des ensembles de méta-attributs à utiliser en est ainsi encore à ses débuts, en particulier pour les méta-attributs des attributs, virtuellement inutilisés par le passé. Des expériences plus approfondies permettraient certainement d'y voir plus clair, et bénéficieraient grandement d'un protocole dédié, permettant l'exploration de l'espace combinatoire des méta-attributs (nécessaire pour le calcul de réelles mesures de *contribution* des différents méta-attributs). De même, pour les méta-attributs généraux, des expériences prenant en compte le nombre optimal de méta-attributs à utiliser et les groupes de méta-attributs fortement dépendants pourraient permettre d'identifier des combinaisons mieux adaptées à diverses situations.

6.2.2 Assistance intelligente et innovation

Comme nous l'avons vu au chapitre 5, on peut efficacement tirer parti des expériences passées pour en recommander de nouvelles, mais l'étape suivante sera de construire dynamiquement ces nouvelles expériences pour l'utilisateur courant. Il s'agit là d'une problématique d'*Innovation* cruciale en analyse de données. En effet, s'agissant d'un domaine relativement nouveau et extrêmement prolifique, de nombreuses nouvelles techniques d'analyse

voient le jour chaque année, ouvrant de nouveaux horizons mais rendant aussi potentiellement obsolètes des approches plus anciennes. Afin de rester performant dans ce contexte de rapide évolution, un assistant à l'analyse de données se doit de pouvoir *Innover*, c'est à dire recommander des chaînes de traitements inédites, ou l'emploi d'algorithmes encore méconnus.

On rencontre alors cet équilibre entre amélioration et exploration commun à de nombreuses méta-heuristiques (Hao et Solnon 2014). En effet, tester des solutions innovantes est un risque vis-à-vis du problème de l'utilisateur courant : tester une solution inconnue, c'est repousser le test d'une solution que l'on avait déjà jugée pertinente. Si la solution innovante ne se révèle pas intéressante, on a fait perdre du temps à l'utilisateur sans bénéfice pour lui. En revanche, on a réalisé une *exploration* de l'espace : l'expérience faite avec la chaîne de traitements innovante n'est pas perdue, elle rejoint notre base d'expériences passées. Quand un autre utilisateur se présentera avec un nouveau problème, peut-être s'apercevra t'on alors grâce à cette expérience que notre chaîne de traitements innovante est remarquablement pertinente dans ce nouveau contexte.

C'est ce potentiel que l'on espère des méthodes de constructions plus avancés décrites en 5.2.3. Au-delà d'améliorer encore davantage les performances au méta-niveau, ces méthodes offrent la possibilité d'innover, c'est à dire de proposer des *workflows* encore jamais utilisés mais susceptibles de bonnes performances. Ceci laisse envisager un cycle vertueux : si les *workflows* construits sont *nouveaux*, une itération du processus de méta-analyse effectue une réelle exploration de l'espace d'analyse, améliorant d'autant plus la performance des futures itérations. L'équilibre entre performance et exploration nécessitera certainement un important travail d'ajustement, mais offre des perspectives encourageantes.

6.2.3 Assistance en amont

La définition du besoin utilisateur constitue un autre verrou majeur pour l'assistance à l'analyse. En effet, même dans le cas où l'utilisateur a une bonne idée du problème métier qu'il souhaite adresser, il ne le formulera typiquement pas selon un modèle d'analyse de données. Cette formalisation du problème requiert souvent un travail important, impliquant déjà l'intervention d'un analyste qui devra acquérir une compréhension suffisante du problème métier pour le traduire convenablement. Afin d'identifier les facteurs clés de ce processus de formulation du problème, un premier scénario d'interaction a été réalisé. Il consiste à étudier l'interaction de deux chercheurs, respectivement en biologie et analyse de données, dans l'approche d'un problème métier de biologie : "*Qualifier l'impact des*

paramètres métaboliques et fonctionnels initiaux sur l'évolution du potentiel musculaire, dans le cadre d'un traitement à l'apeline[2]".

En contexte, l'expérience de biologie étudiée fait intervenir deux populations de souris, l'une subissant un traitement à l'apeline, et mesure l'évolution chez chaque souris d'un ensemble de paramètres (poids, endurance, force des pattes...). Un processus itératif de discussion et démonstration s'est alors mis en place entre les deux chercheurs pour aboutir à la reformulation du problème métier énoncé ci-avant en deux problèmes d'analyse :

1. Définir un indicateur de potentiel musculaire maximisant l'écart entre souris traitées et non traitées. Ce problème a déjà nécessité la construction d'un processus d'analyse complexe. On a tout d'abord contraint la forme mathématique de l'indicateur selon la connaissance du domaine, puis utilisé des méthodes d'optimisation pour isoler les fonctions assurant la plus grande disparité statistique entre les populations.
2. Qualifier l'impact des paramètres métaboliques et fonctionnels de départ sur l'évolution de cet indicateur chez la population traitée. Ce problème a pu être adressé par des techniques de pondération et de sélection d'attributs précédent une régression, afin d'isoler les attributs les plus *informatifs*.

Un des premiers constats au sortir de ce scénario fut la construction de deux problèmes d'analyse pour résoudre un unique problème métier. Cette *reformulation* peut ainsi présenter une grande complexité, laissant envisager des problèmes métiers ne présentant pas de formulation naturelle évidente en analyse. La procédure de reformulation elle-même a nécessité de nombreux échanges, révélant de nombreux écarts (prévisibles) de terminologie, et insistant sur la complexité du processus. La construction d'un dictionnaire vulgarisé des tâches d'analyse réalisables pourrait constituer un premier pas vers l'indépendance de l'utilisateur, en lui permettant d'identifier son problème à des scénarios plus simples.

Un second aspect abordé fut celui de la définition des préférences utilisateur en terme d'analyse. En effet, l'interprétation des différents critères d'évaluation des résultats d'analyse est facilement déroutante, et pourrait également bénéficier d'une description vulgarisée. On pourrait imaginer constituer un ensemble de *profils* de préférences, décrivant les attentes d'utilisateurs faces à différents scénarios. L'utilisateur pourrait alors choisir un tel profil par analogie entre son problème et les situations présentées.

Le dernier verrou mis en avant par ce scénario consiste en l'utilisation de la connaissance domaine de l'utilisateur au sein même des processus d'analyse. En effet, l'une des

2. peptide jouant un rôle lors du processus d'insulino-résistance

étapes d'analyse du premier problème consistait en la définition d'une classe de fonctions candidates sur lesquelles réaliser une optimisation. Cette définition faisant intervenir les connaissances domaines de l'utilisateur, il s'agit d'une opération naturellement interactive, dont l'automatisation s'avérerait coûteuse.

L'assistance à la définition du besoin en analyse présente ainsi encore de nombreuses perspectives, et sera indispensable au déploiement d'un environnement d'assistance complet.

6.2.4 Assistance en aval

Enfin, même en supposant qu'un utilisateur ait une bonne connaissance des données à analyser, il lui est généralement très difficile de s'approprier toutes les techniques possibles pour obtenir le résultat le plus pertinent. Cette appropriation des traitements et de leurs résultats reste un verrou important dans les outils actuels d'assistance à l'analyse. Cette étape est pourtant responsable, dans un contexte réel, de la compréhension et de la fiabilité des résultats produits. Pour permettre l'évolution de l'assistance à l'analyse vers une réelle exploitation, il sera donc nécessaire de proposer des techniques permettant à l'utilisateur de s'approprier les démarches d'analyse recommandées. Cette appropriation peut être vue de deux manières complémentaires :

— En adaptant les recommandations au contexte de l'utilisateur. Ceci implique la réalisation préalable d'un important travail de modélisation et de construction du besoin, tel que décrit ci-avant (6.2.3), en y ajoutant *l'expertise* de l'utilisateur. On peut alors tenter de proposer des processus d'analyses *familiers* pour l'utilisateur. On peut imaginer de pousser la méta-analyse plus avant en proposant de nouveaux processus, inconnus de l'utilisateur, en les lui *expliquant*, par exemple par analogie avec des techniques qui lui sont familières.

— En permettant l'exploration de processus d'analyse réalisés par d'autres utilisateurs, par exemple à l'aide d'un langage de navigation. Cela supposera une modélisation précise des processus et des opérateurs, tant en syntaxe qu'en sémantique. Il serait alors beaucoup plus simple pour des utilisateurs d'ouvrir la "boite noire" que constitue souvent les processus d'analyse. Il serait alors possible de fournir mécaniquement une information structurée sur les opérateurs d'analyse et leur agencement, facilitant l'appropriation des processus complets.

6.3 Conclusion

Ces nombreuses perspectives attestent ainsi des voies de recherche que l'on a pu ouvrir. L'élan acquis ne sera pas perdu, une nouvelle thèse prenant d'ores et déjà la relève avec l'objectif d'adresser les diverses perspectives centrées utilisateur. L'amélioration des processus de recommandation de chaîne de traitements est de même appelée à se poursuivre rapidement, notamment par l'implémentation des approches décrites au chapitre 5. La conclusion de ce travail n'est ainsi qu'un commencement, un premier pas vers une réponse adaptée à un besoin grandissant aussi vite que nos masses de données.

Bibliographie du chapitre

Hao, Jin-Kao et Christine Solnon (2014). *Méta-heuristiques et intelligence artificielle*. Rapp. tech. LIRIS - Laboratoire d'InfoRmatique en Image et Systemes d'information, p. 1–19. URL : https://hal.archives-ouvertes.fr/hal-01313162 (visité le 01/04/2017) (cf. p. 164, 188).

Raynaut, William, Chantal Soule-Dupuy et Nathalie Valles-Parlangeau (2016a). « Caractérisation d'instances d'apprentissage pour méta-mining évolutionnaire ». In : *Extraction et Gestion des Connaissances*. (Classée C par core.edu.au). Poster. URL : http://oatao.univ-toulouse.fr/17181/ (cf. p. 182).

— (2016b). « Caractérisation d'instances d'apprentissage pour méta-mining évolutionnaire ». In : *Fouille de Données Complexes*. Atelier EGC, p. 25–30. URL : http://egc2016.univ-reims.fr/data/%5C_uploaded/file/2016-actes-fdc.pdf (cf. p. 182).

— (2016c). « Meta-Mining Evaluation Framework : A large scale proof of concept on Meta-Learning ». In : *29th Australasian Joint Conference on Artificial Intelligence*. (Classée B par core.edu.au). Springer, p. 215–228. URL : https://link.springer.com/book/10.1007/978-3-319-50127-7 (cf. p. 181).

— (2016d). « Une approche par dissimilarité pour la caractérisation de jeux de données ». In : *32ème Conférence sur la Gestion de Données - Principes, Technologies et Applications (BDA)*. URL : https://hal.archives-ouvertes.fr/hal-01470866 (cf. p. 182).

— (2017a). « Cadre d'Evaluation pour la Méta-Analyse de Données ». In : *Extraction et Gestion des Connaissances*. (Classée C par core.edu.au). Poster. URL : https://hal.archives-ouvertes.fr/hal-01470864 (cf. p. 181).

— (2017b). *Dissimilarités entre jeux de données*. Rapp. scient. RR–2017–07–FR. IRIT. URL : https://hal.archives-ouvertes.fr/hal-01501485 (visité le 11/04/2017) (cf. p. 182).

Raynaut, William, Chantal Soulé-Dupuy, Nathalie Vallés-Parlangeau, Cédric Dray et Philippe Valet (2015). « Characterization of Learning Instances for Evolutionary Meta-Learning ». In : *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD Doctoral Consortium)*. (Classée A* par core.edu.au). Springer, p. 198–206. URL : http://urn.fi/URN:ISBN:978-952-60-6443-7 (cf. p. 182).

Smid, Jakub (2016). « Computational Intelligence Methods in Metalearning ». Thèse de doct. Charles University in Prague (cf. p. 43, 78, 86, 88, 99, 104, 185).

Bibliographie Complète

Livres

Aha, David (2013). *Lazy learning*. Springer Science & Business Media (cf. p. 23, 26).

Brazdil, Pavel, Christophe Giraud Carrier, Carlos Soares et Ricardo Vilalta (2008). *Metalearning : Applications to data mining*. Springer Science & Business Media (cf. p. 29).

Jensen, Finn V (1996). *An introduction to Bayesian networks*. T. 210. UCL press London (cf. p. 24, 26).

Liu, Huan et Hiroshi Motoda (2012). *Feature selection for knowledge discovery and data mining*. T. 454. Springer Science & Business Media (cf. p. 35).

Michie, Donald, David J Spiegelhalter et Charles C Taylor (1994). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA : Ellis Horwood. isbn : 0-13-106360-X (cf. p. 29, 32, 61).

Quinlan, J Ross (2014). *C4. 5 : programs for machine learning*. Elsevier (cf. p. 22, 26).

Rosenblatt, Frank (1962). *Principles of neurodynamics*. Spartan Book (cf. p. 23, 26).

Sutton, Richard S et Andrew G Barto (1998). *Reinforcement learning : An introduction*. T. 1. 1. MIT press Cambridge (cf. p. 24, 26).

Vapnik, Vladimir (2013). *The nature of statistical learning theory*. Springer Science & Business Media (cf. p. 24, 26).

Articles

Aha, David (1992). « Generalizing from case studies : A case study ». In : *9th International Conference on Machine Learning*, p. 1–10 (cf. p. 28).

Akaike, H (1974). « A new look at the statistical model identification ». In : *IEEE Transactions on Automatic Control* 19.6, p. 716–723 (cf. p. 112).

Bacardit, Jaume et Xavier Llorà (2013). « Large-scale data mining using genetics-based machine learning ». In : *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* 3.1, p. 37–61 (cf. p. 24, 26, 37).

Bailey, Timothy et Charles Elkan (1993). « Estimating the accuracy of learned concepts." » In : *International Joint Conference on Artificial Intelligence*. Citeseer (cf. p. 28).

Batista, GEAPA et Diego Furtado Silva (2009). « How k-nearest neighbor parameters affect its performance ». In : *Argentine Symposium on Artificial Intelligence*, p. 1–12 (cf. p. 115).

Benavoli, Alessio, Giorgio Corani, Francesca Mangili et Marco Zaffalon (2015). « A Bayesian nonparametric procedure for comparing algorithms ». In : *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, p. 1264–1272 (cf. p. 28).

Bergstra, James et Yoshua Bengio (2012). « Random search for hyper-parameter optimization ». In : *The Journal of Machine Learning Research* 13.1, p. 281–305 (cf. p. 39, 42).

Bernstein, Abraham, Foster Provost et Shawndra Hill (2005). « Toward intelligent assistance for a data mining process : An ontology-based approach for cost-sensitive classification ». In : *Knowledge and Data Engineering, IEEE Transactions on* 17.4, p. 503–518 (cf. p. 38, 41).

Bischl, Bernd et al. (2016). « Aslib : A benchmark library for algorithm selection ». In : *Artificial Intelligence* 237, p. 41–58 (cf. p. 37, 42).

Blockeel, Hendrik (2015). « Data Mining : From Procedural to Declarative Approaches ». In : *New Generation Computing* 33.2, p. 115–135 (cf. p. 37).

Bosnić, Zoran et Igor Kononenko (2009). « An Overview of Advances in Reliability Estimation of Individual Predictions in Machine Learning ». In : *Intelligent Data Analysis* 13.2, p. 385–401. issn : 1088-467X (cf. p. 39).

— (2010). « Automatic selection of reliability estimates for individual regression predictions ». In : *The Knowledge Engineering Review* 25.01, p. 27–47 (cf. p. 40, 42).

Brazdil, Pavel B, Carlos Soares et Joaquim Pinto Da Costa (2003). « Ranking learning algorithms : Using IBL and meta-learning on accuracy and time results ». In : *Machine Learning* 50.3, p. 251–277 (cf. p. 30–32).

Brazdil, Pavel, João Gama et Bob Henery (1994). « Characterizing the applicability of classification algorithms using meta-level learning ». In : *European conference on machine learning*. Springer, p. 83–102 (cf. p. 30, 31, 65, 96, 97).

Breiman, Leo (1996). « Bagging predictors ». In : *Machine learning* 24.2, p. 123–140 (cf. p. 25, 26).

— (2001). « Random forests ». In : *Machine learning* 45.1, p. 5–32 (cf. p. 25, 26, 112, 139).

Brin, Sergey, Rajeev Motwani, Jeffrey D Ullman et Shalom Tsur (1997). « Dynamic itemset counting and implication rules for market basket data ». In : *ACM SIGMOD Record*. T. 26. 2. ACM, p. 255–264 (cf. p. 24, 26).

Brodley, Carla E (1995). « Recursive automatic bias selection for classifier construction ». In : *Machine Learning* 20.1-2, p. 63–94 (cf. p. 30, 31).

Brown, Gavin, Adam Pocock, Ming-Jie Zhao et Mikel Luján (2012). « Conditional likelihood maximisation : a unifying framework for information theoretic feature selection ». In : *The Journal of Machine Learning Research* 13.1, p. 27–66 (cf. p. 35, 36, 126, 144, 146).

Burges, Christopher JC (1998). « A tutorial on support vector machines for pattern recognition ». In : *Data mining and knowledge discovery* 2.2, p. 121–167 (cf. p. 24, 26).

Caraffini, Fabio, Ferrante Neri et Lorenzo Picinali (2014). « An analysis on separability for memetic computing automatic design ». In : *Information Sciences* 265, p. 1–22 (cf. p. 37, 42).

Castiello, Ciro, Giovanna Castellano et Anna Maria Fanelli (2005). « Meta-data : Characterization of input features for meta-learning ». In : *Modeling decisions for artificial intelligence*. Springer, p. 457–468 (cf. p. 33).

Chapelle, Olivier, Bernhard Scholkopf et Alexander Zien (2009). « Semi-Supervised Learning ». In : *IEEE Transactions on Neural Networks* 20.3, p. 542–542 (cf. p. 24, 26).

Chen, Tianqi et Carlos Guestrin (2016). « Xgboost : A scalable tree boosting system ». In : *22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, p. 785–794 (cf. p. 25, 26).

Cleary, John G, Leonard E Trigg et al. (1995). « K* : An instance-based learner using an entropic distance measure ». In : *Proceedings of the 12th International Conference on Machine learning*. T. 5, p. 108–114 (cf. p. 23, 26, 112, 137).

Cohen, Jacob (1968). « Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. » In : *Psychological bulletin* 70.4, p. 213 (cf. p. 27, 28, 62, 100, 170).

Cover, Thomas M et Peter E Hart (1967). « Nearest neighbor pattern classification ». In : *Information Theory, IEEE Transactions on* 13.1, p. 21–27 (cf. p. 23).

Demšar, Janez (2006). « Statistical comparisons of classifiers over multiple data sets ». In : *The Journal of Machine Learning Research* 7, p. 1–30 (cf. p. 28, 68).

Diamantini, Claudia, Domenico Potena et Emanuele Storti (2009a). « Kddonto : An ontology for discovery and composition of kdd algorithms ». In : *Third Generation Data Mining : Towards Service-Oriented Knowledge Discovery (SoKD)*, p. 13–24 (cf. p. 41, 42, 159, 163).

— (2009b). « Ontology-driven KDD process composition ». In : *Advances in Intelligent Data Analysis VIII*. Springer, p. 285–296 (cf. p. 41, 42, 159, 163).

Dong, Lin, Eibe Frank et Stefan Kramer (2005). « Ensembles of balanced nested dichotomies for multi-class problems ». In : *Knowledge Discovery in Databases : PKDD 2005*. Springer, p. 84–95 (cf. p. 25, 26, 69, 95).

Duin, Robert PW (2015). « The Dissimilarity Representation for finding Universals from Particulars by an anti-essentialist Approach ». In : *Pattern Recognition Letters* 64, p. 37–43. issn : 0167-8655. doi : http://dx.doi.org/10.1016/j.patrec.2015.04.015. (Visité le 01/04/2017) (cf. p. 43).

Engels, Robert et Christiane Theusinger (1998). « Using a Data Metric for Preprocessing Advice for Data Mining Applications. » In : *ECAI*. Citeseer, p. 430–434 (cf. p. 38, 42).

Escalante, Hugo Jair, Manuel Montes et Luis Enrique Sucar (2009). « Particle swarm model selection ». In : *The Journal of Machine Learning Research* 10, p. 405–440 (cf. p. 38, 42).

Feurer, Matthias, T Springenberg et Frank Hutter (2015). « Initializing bayesian hyperparameter optimization via meta-learning ». In : *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (cf. p. 39, 42).

Frank, András (2005). « On Kuhn's Hungarian method : a tribute from Hungary ». In : *Naval Research Logistics (NRL)* 52.1, p. 2–5 (cf. p. 88).

Frank, Eibe, Yong Wang, Stuart Inglis, Geoffrey Holmes et Ian H Witten (1998). « Using model trees for classification ». In : *Machine Learning* 32.1, p. 63–76 (cf. p. 23, 26, 95).

Freund, Yoav, Robert Schapire et N Abe (1999). « A short introduction to boosting ». In : *Journal-Japanese Society For Artificial Intelligence* 14.771-780, p. 1612 (cf. p. 25, 26).

Friedman, Milton (1940). « A comparison of alternative tests of significance for the problem of m rankings ». In : *The Annals of Mathematical Statistics* 11.1, p. 86–92 (cf. p. 68).

Fürnkranz, Johannes (1999). « Separate-and-conquer rule learning ». In : *Artificial Intelligence Review* 13.1, p. 3–54 (cf. p. 22, 26).

Fürnkranz, Johannes et Johann Petrak (2001). « An evaluation of landmarking variants ». In : *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, p. 57–68 (cf. p. 34).

Gama, Joao et Pavel Brazdil (1995). « Characterization of classification algorithms ». In : *Progress in Artificial Intelligence*. Springer, p. 189–200 (cf. p. 30, 31, 153).

Ganti, Venkatesh, Johannes Gehrke et Raghu Ramakrishnan (1999). « A framework for measuring changes in data characteristics ». In : *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, p. 126–137 (cf. p. 30, 31).

Giraud-Carrier, Christophe, Ricardo Vilalta et Pavel Brazdil (2004). « Introduction to the special issue on meta-learning ». In : *Machine learning* 54.3, p. 187–193 (cf. p. 29, 32, 77, 96).

Goble, Carole A et al. (2010). « myExperiment : a repository and social network for the sharing of bioinformatics workflows ». In : *Nucleic acids research* 38.suppl 2, p. 677–682 (cf. p. 42).

Gordon, Diana F et Marie Desjardins (1995). « Evaluation and selection of biases in machine learning ». In : *Machine Learning* 20.1-2, p. 5–22 (cf. p. 29, 31).

Gulgezen, Gokhan, Zehra Cataltepe et Lei Yu (2009). « Stable and accurate feature selection ». In : *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, p. 455–468 (cf. p. 36).

Guyon, Isabelle et André Elisseeff (2003). « An introduction to variable and feature selection ». In : *The Journal of Machine Learning Research* 3, p. 1157–1182 (cf. p. 36).

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann et Ian H Witten (2009). « The WEKA data mining software : an update ». In : *ACM SIGKDD explorations newsletter* 11.1, p. 10–18 (cf. p. 66, 100, 168).

Hanley, James A et Barbara J McNeil (1982). « The meaning and use of the area under a receiver operating characteristic (ROC) curve. » In : *Radiology* 143.1, p. 29–36 (cf. p. 27, 28).

Hilario, Melanie, Phong Nguyen, Huyen Do, Adam Woznica et Alexandros Kalousis (2011). « Ontology-based meta-mining of knowledge discovery workflows ». In : *Meta-Learning in Computational Intelligence*. Springer, p. 273–315 (cf. p. 41, 42).

Hinton, Geoffrey E, Simon Osindero et Yee-Whye Teh (2006). « A fast learning algorithm for deep belief nets ». In : *Neural computation* 18.7, p. 1527–1554 (cf. p. 23, 26).

Ho, Tin Kam, Jonathan J Hull et Sargur N Srihari (1994). « Decision combination in multiple classifier systems ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 16.1, p. 66–75 (cf. p. 25, 26).

Ho, Tin Kamo et Mitra Basu (2002). « Complexity measures of supervised classification problems ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.3, p. 289–300 (cf. p. 33, 146).

Holmes, Geoffrey, Mark Hall et Eibe Prank (1999). « Generating rule sets from model trees ». In : *Australasian Joint Conference on Artificial Intelligence*. Springer, p. 1–12 (cf. p. 112).

Huang, Yue, Paul McCullagh, Norman Black et Roy Harper (2007). « Feature Selection and Classification Model Construction on Type 2 Diabetic Patients Data ». In : *Artificial Intelligence in Medicine* 41.3, p. 251–262. issn : 0933-3657. doi : 10.1016/j.artmed.2007.07.002. url : http://dx.doi.org/10.1016/j.artmed.2007.07.002 (visité le 01/04/2017) (cf. p. 36).

Jain, Anil K, M Narasimha Murty et Patrick J Flynn (1999). « Data clustering : a review ». In : *ACM computing surveys (CSUR)* 31.3, p. 264–323 (cf. p. 24, 26).

Kalousis, Alexandros, João Gama et Melanie Hilario (2004). « On data and algorithms : Understanding inductive performance ». In : *Machine Learning* 54.3, p. 275–312 (cf. p. 43, 61).

Kalousis, Alexandros et Maelanie Hilario (2000). « Building algorithm profiles for prior model selection in knowledge discovery systems ». In : *International journal of engineering intelligent systems for electrical engineering and communications* 8.2, p. 77–88 (cf. p. 30, 31).

— (2001a). « Model selection via meta-learning : a comparative study ». In : *International Journal on Artificial Intelligence Tools* 10.04, p. 525–554 (cf. p. 30–32, 55, 78, 96).

Kalousis, Alexandros et Melanie Hilario (2001b). « Feature Selection for Meta-learning ». In : *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. PAKDD. London, UK, UK : Springer-Verlag, p. 222–233. isbn : 3-540-41910-1. url : http://dl.acm.org/citation.cfm?id=646419.693650 (visité le 01/04/2017) (cf. p. 35, 61, 62).

Kazik, O., K. Peskova, M. Pilt et R. Neruda (2011). « Meta Learning in Multi-agent Systems for Data Mining ». In : *2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. T. 2, p. 433–434. doi : 10.1109/WI-IAT.2011.233. (Visité le 01/04/2017) (cf. p. 37, 42).

Kietz, J, Floarea Serban, Abraham Bernstein et Simon Fischer (2009). « Towards cooperative planning of data mining workflows ». In : *Proceedings of the Third Generation Data Mining Workshop at the 2009 European Conference on Machine Learning (ECML 2009)*, p. 1–12 (cf. p. 41).

Kietz, Jörg-Uwe, Floarea Serban et Abraham Bernstein (2010). « eProPlan : A tool to model automatic generation of data mining workflows ». In : *Proceedings of the 3rd Planning to Learn Workshop (WS9) at ECAI*. T. 2010 (cf. p. 41).

Kietz, Jörg-Uwe, Floarea Serban, Abraham Bernstein et Simon Fischer (2010). « Data mining workflow templates for intelligent discovery assistance and auto-experimentation ». In : *Third-Generation Data Mining : Towards Service-oriented Knowledge Discovery (SoKD-10)*, p. 1–12 (cf. p. 41).

Kietz, Jörg-Uwe, Floarea Serban, Simon Fischer et Abraham Bernstein (2014). « "Semantics Inside !" But lets not tell the Data Miners : Intelligent Support for Data Mining ». In : *European Semantic Web Conference ESWC 2014*. Sous la dir. de Fabien Gandon et Claudia Amato. Lecture Notes in Computer Science 8465. Springer, p. 706–720. isbn : 978-3-319-07443-6. doi : 10.1007/978-3-319-07443-6_47. (Visité le 01/04/2017) (cf. p. 41, 42).

King, Ross D., Cao Feng et Alistair Sutherland (1995). « Statlog : comparison of classification algorithms on large real-world problems ». In : *Applied Artificial Intelligence an International Journal* 9.3, p. 289–333 (cf. p. 32).

Kira, Kenji et Larry A Rendell (1992). « The feature selection problem : Traditional methods and a new algorithm ». In : *AAAI*. T. 2, p. 129–134 (cf. p. 35, 36).

Kohavi, Ron et al. (1995). « A study of cross-validation and bootstrap for accuracy estimation and model selection ». In : *Ijcai*. T. 14. 2. Stanford, CA, p. 1137–1145 (cf. p. 28).

Kononenko, Igor (2001). « Machine learning for medical diagnosis : history, state of the art and perspective ». In : *Artificial Intelligence in Medicine* 23, p. 89–109 (cf. p. 27, 28, 42).

Kononenko, Igor et Ivan Bratko (1991). « Information-Based Evaluation Criterion for Classifier's Performance ». In : *Machine Learning* 6.1, p. 67–80. issn : 0885-6125 (cf. p. 27, 28, 63, 169).

Köpf, Christian, Charles Taylor et Jörg Keller (2000). « Meta-analysis : from data characterisation for meta-learning to meta-regression ». In : *Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP*. Citeseer (cf. p. 30, 31).

Kotsiantis, S, D Kanellopoulos et PE Pintelas (2006). « Data preprocessing for supervised leaning ». In : *International Journal of Computer Science* 1.2, p. 111–117 (cf. p. 38).

Kuhn, Harold W (1955). « The Hungarian method for the assignment problem ». In : *Naval research logistics quarterly* 2.1-2, p. 83–97 (cf. p. 88).

Kuncheva, Ludmila I et Christopher J Whitaker (2003). « Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy ». In : *Machine learning* 51.2, p. 181–207 (cf. p. 25, 26).

Lan, Guo-Cheng et al. (2012). « Disease Risk Prediction by Mining Personalized Health Trend Patterns : A Case Study on Diabetes ». In : *2012 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, p. 27–32. doi : 10.1109/TAAI.2012.53. (Visité le 01/04/2017) (cf. p. 42).

Leite, Rui, Pavel Brazdil et Joaquin Vanschoren (2012). « Selecting classification algorithms with active testing ». In : *Machine Learning and Data Mining in Pattern Recognition*. Springer, p. 117–131 (cf. p. 31, 55, 61, 96).

Leyva, Enrique, Adriana Gonzalez et Roxana Perez (2015). « A Set of Complexity Measures Designed for Applying Meta-Learning to Instance Selection ». In : *Knowledge and Data Engineering, IEEE Transactions on* 27.2, p. 354–367 (cf. p. 33, 61, 62).

Littlestone, Nick et Manfred K Warmuth (1994). « The weighted majority algorithm ». In : *Information and computation* 108.2, p. 212–261 (cf. p. 23, 26).

MacKay, David JC (1998). « Introduction to Gaussian processes ». In : *NATO ASI Series F Computer and Systems Sciences* 168, p. 133–166 (cf. p. 112).

Mantovani, Rafael Gomes, André LD Rossi, Joaquin Vanschoren, André Carlos Ponce de Leon Carvalho et al. (2015). « Meta-learning Recommendation of Default Hyperparameter Values for SVMs in Classification Tasks ». In : *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ; International Workshop on Meta-Learning and Algorithm Selection*. University of Porto, p. 80–92. url : http://ceur-ws.org/Vol-1455/#paper-09 (visité le 01/04/2017) (cf. p. 39, 42).

Mitchell, Tom M (1997). « Machine learning ». In : *Burr Ridge, IL : McGraw Hill* 45 (cf. p. 21).

Murthy, Sreerama K (1998). « Automatic construction of decision trees from data : A multi-disciplinary survey ». In : *Data mining and knowledge discovery* 2.4, p. 345–389 (cf. p. 22, 26).

Nakhaeizadeh, Gholamreza et Alexander Schnabl (1997). « Development of Multi-Criteria Metrics for Evaluation of Data Mining Algorithms. » In : *KDD*, p. 37–42 (cf. p. 27).

Nguyen, Phong, Melanie Hilario et Alexandros Kalousis (2014). « Using meta-mining to support data mining workflow planning and optimization ». In : *Journal of Artificial Intelligence Research*, p. 605–644 (cf. p. 41, 42, 151, 159, 163).

Ntoutsi, Irene, Alexandros Kalousis et Yannis Theodoridis (2008). « A general framework for estimating similarity of datasets and decision trees : exploring semantic similarity of decision trees. » In : *SDM*. SIAM, p. 810–821 (cf. p. 146).

Obenshain, Mary K (2004). « Application of data mining techniques to healthcare data ». In : *Infection Control* 25.08, p. 690–695 (cf. p. 25, 42, 153, 173).

Olson, Randal S., Nathan Bartley, Ryan J. Urbanowicz et Jason H. Moore (2016). « Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science ». In : *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. GECCO '16. Denver, Colorado, USA : ACM, p. 485–492. isbn : 978-1-4503-4206-3. doi : 10.1145/2908812.2908918. url : http://doi.acm.org/10.1145/2908812.2908918 (visité le 01/04/2017) (cf. p. 38, 42).

Palaniappan, Sellappan et Rafiah Awang (2008). « Intelligent Heart Disease Prediction System Using Data Mining Techniques ». In : *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*. AICCSA. Washington, DC, USA : IEEE Computer Society, p. 108–115. isbn : 978-1-4244-1967-8. doi : 10.1109/AICCSA.2008.4493524. url : http://dx.doi.org/10.1109/AICCSA.2008.4493524 (visité le 01/04/2017) (cf. p. 25, 173).

Parthiban, Latha et R Subramanian (2007). « Intelligent heart disease prediction system using CANFIS and genetic algorithm ». In : *International Journal of Biological and Life Sciences* 3.3, p. 157–160 (cf. p. 26, 42).

Peng, Hanchuan, Fuhui Long et Chris Ding (2005). « Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy ». In : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.8, p. 1226–1238 (cf. p. 35, 36, 146).

Peng, Yonghong, Peter A Flach, Pavel Brazdil et Carlos Soares (2002). « Decision tree-based data characterization for meta-learning ». In : *IDDM-2002*, p. 111 (cf. p. 34, 61, 146).

Pfahringer, Bernhard, Hilan Bensusan et Christophe Giraud-Carrier (2000). « Tell me who can learn you and i can tell you who you are : Landmarking various learning algorithms ». In : *Proceedings of the 17th international conference on machine learning*, p. 743–750 (cf. p. 34, 61, 100).

Pinto, Fábio, Carlos Soares et Joao Mendes-Moreira (2014). « A framework to decompose and develop metafeatures ». In : *Meta-Learning and Algorithm Selection Workshop at ECAI 2014* (cf. p. 34).

Pizarro, Joaqun, Elisa Guerrero et Pedro L Galindo (2002). « Multiple comparison procedures applied to model selection ». In : *Neurocomputing* 48.1, p. 155–173 (cf. p. 28).

Provost, Foster J, Tom Fawcett et Ron Kohavi (1998). « The case against accuracy estimation for comparing induction algorithms. » In : *ICML*. T. 98, p. 445–453 (cf. p. 27, 28, 56).

Quinlan, J Ross (1987). « Generating Production Rules from Decision Trees. » In : *IJCAI*. T. 87. Citeseer, p. 304–307 (cf. p. 22, 26).

Raynaut, William, Chantal Soule-Dupuy et Nathalie Valles-Parlangeau (2016a). « Caractérisation d'instances d'apprentissage pour méta-mining évolutionnaire ». In : *Extraction et Gestion des Connaissances*. (Classée C par core.edu.au). Poster. url : http://oatao.univ-toulouse.fr/17181/ (cf. p. 182).

Raynaut, William, Chantal Soule-Dupuy et Nathalie Valles-Parlangeau (2016b). « Caractérisation d'instances d'apprentissage pour méta-mining évolutionnaire ». In : *Fouille de Données Complexes*. Atelier EGC, p. 25–30. url : http://egc2016.univ-reims.fr/data/%5C_uploaded/file/2016-actes-fdc.pdf (cf. p. 182).

— (2016c). « Meta-Mining Evaluation Framework : A large scale proof of concept on Meta-Learning ». In : *29th Australasian Joint Conference on Artificial Intelligence*. (Classée B par core.edu.au). Springer, p. 215–228. url : https://link.springer.com/book/10.1007/978-3-319-50127-7 (cf. p. 181).

— (2016d). « Une approche par dissimilarité pour la caractérisation de jeux de données ». In : *32ème Conférence sur la Gestion de Données - Principes, Technologies et Applications (BDA)*. URL : https://hal.archives-ouvertes.fr/hal-01470866 (cf. p. 182).
— (2017a). « Cadre d'Evaluation pour la Méta-Analyse de Données ». In : *Extraction et Gestion des Connaissances*. (Classée C par core.edu.au). Poster. URL : https://hal.archives-ouvertes.fr/hal-01470864 (cf. p. 181).
Raynaut, William, Chantal Soulé-Dupuy, Nathalie Vallés-Parlangeau, Cédric Dray et Philippe Valet (2015). « Characterization of Learning Instances for Evolutionary Meta-Learning ». In : *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD Doctoral Consortium)*. (Classée A* par core.edu.au). Springer, p. 198–206. URL : http://urn.fi/URN:ISBN:978-952-60-6443-7 (cf. p. 182).
Rijn, Jan N. van et Joaquin Vanschoren (2015). « Sharing RapidMiner Workflows and Experiments with OpenML ». In : *MetaSel@ PKDD/ECML*, p. 93–103. URL : http://ceur-ws.org/Vol-1455/#paper-10 (visité le 01/04/2017) (cf. p. 38).
Robnik-Šikonja, Marko et Igor Kononenko (2003). « Theoretical and empirical analysis of ReliefF and RReliefF ». In : *Machine learning* 53.1-2, p. 23–69 (cf. p. 35, 36, 62).
Rumelhart, David E, Geoffrey E Hinton et Ronald J Williams (1988). « Learning representations by back-propagating errors ». In : *Cognitive modeling* 5, p. 3 (cf. p. 23, 26).
Safavian, S R et D Landgrebe (1991). « A survey of decision tree classifier methodology ». In : *IEEE Transactions on Systems, Man, and Cybernetics* 21.3, p. 660–674. ISSN : 0018-9472. DOI : 10.1109/21.97458. (Visité le 01/04/2017) (cf. p. 22, 26).
Schaffer, Cullen (1994). « A conservation law for generalization performance ». In : *Proceedings of the 11th international conference on machine learning*, p. 259–265 (cf. p. 29).
Schmidhuber, Jürgen (2015). « Deep learning in neural networks : An overview ». In : *Neural Networks* 61, p. 85–117 (cf. p. 23, 26).
Serban, Floarea, Joaquin Vanschoren, Jörg-Uwe Kietz et Abraham Bernstein (2013). « A survey of intelligent assistants for data analysis ». In : *ACM Computing Surveys (CSUR)* 45.3, p. 31 (cf. p. 17, 38, 41, 42, 55, 151).
Shannon, Claude Elwood (2001). « A mathematical theory of communication ». In : *Mobile Computing and Communications Review* 5.1, p. 3–55 (cf. p. 33).
Shavlik, Jude W, Raymond J Mooney et Geoffrey G Towell (1991). « Symbolic and neural learning algorithms : An experimental comparison ». In : *Machine learning* 6.2, p. 111–143 (cf. p. 28).
Sleeman, D, Michalis Rissakis, Susan Craw, Nicolas Graner et Sunil Sharma (1995). « Consultant-2 : Pre-and post-processing of machine learning applications ». In : *International Journal of Human-Computer Studies* 43.1. DOI : 10.1006/ijhc.1995.1035. URL : http://dx.doi.org/10.1006/ijhc.1995.1035 (visité le 01/04/2017) (cf. p. 38, 42).

Smid, Jakub et Roman Neruda (2014). « Comparing datasets by attribute alignment ». In : *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*. IEEE, p. 56–62 (cf. p. 43, 78).

Smirnov, N (1948). « Table for Estimating the Goodness of Fit of Empirical Distributions ». In : *The Annals of Mathematical Statistics* 19.2, p. 279–281 (cf. p. 108).

Smith, Jim E (2007). « Coevolving memetic algorithms : a review and progress report ». In : *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on* 37.1, p. 6–17 (cf. p. 168).

Smola, Alex J et Bernhard Schölkopf (2004). « A tutorial on support vector regression ». In : *Statistics and computing* 14.3, p. 199–222 (cf. p. 25, 26, 112).

Soares, Carlos et Pavel B Brazdil (2000). « Zoomed ranking : Selection of classification algorithms based on relevant performance information ». In : *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, p. 126–135 (cf. p. 30, 31).

Somol, Petr, Jiří Grim et Pavel Pudil (2009). « Criteria ensembles in feature selection ». In : *International Workshop on Multiple Classifier Systems*. Springer, p. 304–313 (cf. p. 36).

Strumbelj, Erik, Zoran Bosnic, Igor Kononenko, Branko Zakotnik et Cvetka Grasic ; Kuhar (2010). « Explanation and Reliability of Prediction Models : The Case of Breast Cancer Recurrence ». In : *Knowledge and Information Systems* 24.2, p. 305–324. issn : 0219-1377 (cf. p. 27, 28, 40).

Strumbelj, Erik et Igor Kononenko (2008). « Towards a Model Independent Method for Explaining Classification for Individual Instances ». In : *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*. DaWaK. Berlin, Heidelberg : Springer-Verlag, p. 273–282. isbn : 978-3-540-85835-5. doi : 10.1007/978-3-540-85836-2_26. url : http://dx.doi.org/10.1007/978-3-540-85836-2_26 (visité le 01/04/2017) (cf. p. 27, 28).

Sun, Quan (2014). « Meta-Learning and the Full Model Selection Problem ». In : *Unpublished PhD thesis, University of Waikato* (cf. p. 38, 42).

Sun, Quan et Bernhard Pfahringer (2013). « Pairwise meta-rules for better meta-learning-based algorithm ranking ». In : *Machine learning* 93.1, p. 141–161 (cf. p. 30, 31, 61, 65, 96, 146).

Sun, Quan, Bernhard Pfahringer et Michael Mayo (2012). « Full model selection in the space of data mining operators ». In : *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, p. 1503–1504 (cf. p. 38, 42, 96).

Thornton, Chris, Frank Hutter, Holger H Hoos et Kevin Leyton-Brown (2013). « Auto-WEKA : Combined selection and hyperparameter optimization of classification algorithms ». In : *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, p. 847–855 (cf. p. 39, 42).

Todorovski, Ljupco, Pavel Brazdil et Carlos Soares (2000). « Report on the experiments with feature selection in meta-level learning ». In : *Proceedings of the PKDD-00 workshop on data mining, decision support, meta-learning and ILP : forum for prac-*

tical problem presentation and prospective solutions. Citeseer, p. 27–39 (cf. p. 35, 61, 62).

Uno, Takeaki, Tatsuya Asai, Yuzo Uchida et Hiroki Arimura (2004). « An efficient algorithm for enumerating closed patterns in transaction databases ». In : *Discovery science*, p. 16–31 (cf. p. 65).

Vanschoren, Joaquin, Hendrik Blockeel, Bernhard Pfahringer et Geoffrey Holmes (2012). « Experiment databases ». In : *Machine Learning* 87.2, p. 127–158 (cf. p. 28, 90).

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl et Luis Torgo (2013). « OpenML : Networked Science in Machine Learning ». In : *SIGKDD Explorations* 15.2, p. 49–60. doi : 10.1145/2641190.2641198. url : http://doi.acm.org/10.1145/2641190.2641198 (visité le 01/04/2017) (cf. p. 28, 31, 38, 57, 60, 168).

Vilalta, Ricardo et Youssef Drissi (2002). « A Perspective View and Survey of Meta-learning ». In : *Artificial Intelligence Review* 18.2, p. 77–95. issn : 0269-2821. doi : 10.1023/A:1019956318069. url : http://dx.doi.org/10.1023/A:1019956318069 (visité le 01/04/2017) (cf. p. 29, 32, 61).

Wang, Liwei, Masashi Sugiyama, Cheng Yang, Kohei Hatano et Jufu Feng (2009). « Theory and algorithm for learning with dissimilarity functions ». In : *Neural computation* 21.5, p. 1459–1484 (cf. p. 80, 90, 91).

Wistuba, Martin, Nicolas Schilling et Lars Schmidt-Thieme (2015). « Learning Data Set Similarities for Hyperparameter Optimization Initializations ». In : *MetaSel@ PKDD/ECML*, p. 15–26. url : http://ceur-ws.org/Vol-1455/#paper-04 (visité le 01/04/2017) (cf. p. 39, 42, 79).

Wolpert, D (1992). « Stacked Generalization ». In : *Neural Networks*, p. 241–259 (cf. p. 25, 26).

Wolpert, David H (1996). « The lack of a priori distinctions between learning algorithms ». In : *Neural computation* 8.7, p. 1341–1390 (cf. p. 29, 95, 173, 175).

Wolpert, David H et William G Macready (1997). « No free lunch theorems for optimization ». In : *Evolutionary Computation, IEEE Transactions on* 1.1, p. 67–82 (cf. p. 29, 95).

Xu, Lin, Frank Hutter, Jonathan Shen, Holger H Hoos et Kevin Leyton-Brown (2012). « SATzilla2012 : improved algorithm selection based on cost-sensitive classification models ». In : *SAT Challenge 2012 : Solver and Benchmark Descriptions*, p. 57–58 (cf. p. 37, 42, 55).

Yoo, Andy B, Morris A Jette et Mark Grondona (2003). « Slurm : Simple linux utility for resource management ». In : *Job Scheduling Strategies for Parallel Processing*. Springer, p. 44–60 (cf. p. 67, 113).

Zakova, Monika, Petr Kremen, Filip Zelezny et Nada Lavrac (2011). « Automating knowledge discovery workflow composition through ontology-based planning ». In : *Automation Science and Engineering, IEEE Transactions on* 8.2, p. 253–264 (cf. p. 41, 42, 55, 77, 151, 159).

Zeevi, David et al. (2015). « Personalized Nutrition by Prediction of Glycemic Responses ». In : *Cell* 163.5, p. 1079–1094 (cf. p. 42, 152).

Zhang, Guoqiang Peter (2000). « Neural networks for classification : a survey ». In : *Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on* 30.4, p. 451–462 (cf. p. 23, 26).

Zheng, Zijian, Ron Kohavi et Llew Mason (2001). « Real world performance of association rule algorithms ». In : *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, p. 401–406 (cf. p. 28).

Zhou, Aimin, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan et Qingfu Zhang (2011). « Multiobjective evolutionary algorithms : A survey of the state of the art ». In : *Swarm and Evolutionary Computation* 1.1, p. 32–49 (cf. p. 168).

Autres

Frank, Eibe (2014). *Fully supervised training of Gaussian radial basis function networks in WEKA*. Rapp. tech. Department of Computer Science, The University of Waikato (cf. p. 26, 112).

Fürnkranz, J et J Petrak (2002). *Extended data characteristics*. Rapp. tech. Accessed 12/11/15 at `citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.302`. METAL consortium (cf. p. 34, 62).

Hao, Jin-Kao et Christine Solnon (2014). *Méta-heuristiques et intelligence artificielle*. Rapp. tech. LIRIS - Laboratoire d'InfoRmatique en Image et Systemes d'information, p. 1–19. URL : `https://hal.archives-ouvertes.fr/hal-01313162` (visité le 01/04/2017) (cf. p. 164, 188).

Kalousis, Alexandros (2002). « Algorithm selection via meta-learning ». Thèse de doct. Universite de Geneve (cf. p. 31, 32, 35, 78, 79).

Koller, Daphne et Mehran Sahami (1996). *Toward optimal feature selection*. Rapp. tech. Stanford InfoLab (cf. p. 35, 36).

Kotsiantis, S, I Zaharakis et P Pintelas (2007). *Supervised machine learning : A review of classification techniques* (cf. p. 21).

Misir, Mustafa et Michele Sebag (2013). *Algorithm selection as a collaborative filtering problem*. Rapp. tech. CNRS - Centre National de la Recherche Scientifique : UMR8623, p. 43 (cf. p. 37, 42).

Nemenyi, PB (1963). « Distribution-free multiple comparisons ». Thèse de doct. Princeton University (cf. p. 69).

Nilsson, Nils J (1965). *Learning machines.* (Cf. p. 24, 26).

Platt, John et al. (1998). *Sequential minimal optimization : A fast algorithm for training support vector machines*. Rapp. tech. ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING (cf. p. 25, 26).

Raynaut, William, Chantal Soule-Dupuy et Nathalie Valles-Parlangeau (2017b). *Dissimilarités entre jeux de données*. Rapp. scient. RR–2017–07–FR. IRIT. URL : `https://hal.archives-ouvertes.fr/hal-01501485` (visité le 11/04/2017) (cf. p. 182).

Serban, F (2013). « Toward effective support for data mining using intelligent discovery assistance ». Thèse de doct. University of Zurich (cf. p. 41, 42, 77, 159).

Smid, Jakub (2016). « Computational Intelligence Methods in Metalearning ». Thèse de doct. Charles University in Prague (cf. p. 43, 78, 86, 88, 99, 104, 185).

Zhu, Xiaojin (2010). « Semi-Supervised Learning ». In : *Encyclopedia of Machine Learning*. Sous la dir. de Claude Sammut et Geoffrey I. Webb. Boston, MA : Springer US, p. 892–897. isbn : 978-0-387-30164-8. doi : 10.1007/978-0-387-30164-8_749. url : http://dx.doi.org/10.1007/978-0-387-30164-8_749 (visité le 01/04/2017) (cf. p. 24, 26).

Annexe A

Listings

TABLE A.1 – Chaines de traitement weka employés comme classifieurs de base dans le chapitre 3 et dans la preuve de concept du chapitre 4

Bagging REPTree
SMO PolyKernel
SMO RBFKernel
AdaBoostM1 IBk
AdaBoostM1 J48
AdaBoostM1 JRip
AdaBoostM1 LADTree
AdaBoostM1 LMT
AdaBoostM1 LWL DecisionStump
AdaBoostM1 NaiveBayes
AdaBoostM1 OneR
AdaBoostM1 RandomForest
AdaBoostM1 RandomTree
AdaBoostM1 REPTree
AdaBoostM1 SMO PolyKernel
LogitBoost DecisionStump
MultiBoostAB DecisionStump
Bagging J48
Bagging JRip
Bagging LinearRegression
Bagging LMT
Bagging LWL DecisionStump
Bagging OneR
Bagging RandomForest
Bagging SMO PolyKernel
END ND J48
Grading ZeroR
GridSearch PLSFilter LinearRegression
IterativeClassifierOptimizer LogitBoost DecisionStump
RacedIncrementalLogitBoost DecisionStump
RandomCommittee RandomTree
Vote ZeroR
MultiBoostAB NaiveBayes
MultiBoostAB GaussianProcesses PolyKernel
Logistic
J48
ZeroR
Ridor
OneR
PART
OLM
SimpleCart
REPTree
RandomTree
RandomForest
NBTree
LMT
J48graft
HoeffdingTree
FT
ExtraTree
DecisionStump
BFTree
JRip
DecisionTable BestFirst
ConjunctiveRule
HyperPipes
InputMappedClassifier ZeroR
VFI
IB1
IBk
KStar
LBR
LWL DecisionStump
GaussianProcesses PolyKernel
LibLINEAR
Logistic
RBFClassifier
NaiveBayes
FilteredClassifier Standardize NaiveBayes
PrincipalComponents Ranker J48
FilteredClassifier PrincipalComponents J48
A1DE
BayesNet K2
LibSVM
SimpleLogistic
SMO PolyKernel
SMO RBFKernel
Winnow
AdaBoostM1 DecisionStump
CfsSubsetEval BestFirst J48
Bagging REPTree
Dagging DecisionStump
END ND J48
Grading ZeroR
LogitBoost DecisionStump
MultiBoostAB DecisionStump
RacedIncrementalLogitBoost DecisionStump
RandomSubSpace REPTree
RotationForest PrincipalComponents J48
FURIA
LADTree
FilteredClassifier Discretize J48

TABLE A.2 – Jeux de données OpenML utilisés dans le chapitre 3 et dans la preuve de concept du chapitre 4

anneal	analcatdata_germangss	fri_c3_250_25	analcatdata_birthday
anneal	analcatdata_bankruptcy	bank32nh	iris
kr-vs-kp	fl2000	fri_c4_250_100	analcatdata_authorship
labor	prnn_viruses	analcatdata_vehicle	mfeat-fourier
arrhythmia	biomed	sleuth_case1102	squash-stored
audiology	colleges_aaup	fri_c4_500_25	wine
liver-disorders	rmftsa_sleepdata	kdd_el_nino-small	hayes-roth
autos	sleuth_ex2016	autoHorse	autos
lymph	sleuth_ex2015	stock	kdd_JapaneseVowels
balance-scale	visualizing_livestock	analcatdata_runshoes	mfeat-factors
mfeat-factors	diggle_table_a2	breastTumor	waveform-5000
breast-cancer	fruitfly	wind	optdigits
mfeat-fourier	fri_c3_100_50	schlvote	heart-c
mfeat-karhunen	rmftsa_ladata	fri_c0_100_50	cmc
mfeat-morphological	veteran	tecator	squash-unstored
mfeat-pixel	abalone	analcatdata_gsssexsurvey	analcatdata_marketing
car	pwLinear	housing	collins
mfeat-zernike	analcatdata_vineyard	fishcatch	fl2000
cmc	bank8FM	fri_c4_500_10	anneal
mushroom	fri_c2_100_5	bolts	eucalyptus
colic	analcatdata_supreme	hungarian	car
colic	visualizing_slope	analcatdata_gviolence	analcatdata_broadway
optdigits	fri_c1_250_5	vinnie	kdd_ipums_la_97-small
credit-a	baskball	auto93	vehicle
page-blocks	fri_c0_250_50	sleuth_ex2016	mfeat-zernike
cylinder-bands	machine_cpu	fri_c4_250_10	prnn_fglass
postoperative-patient-data	cpu_small	sleuth_ex2015	balance-scale
dermatology	visualizing_environmental	analcatdata_neavote	analcatdata_bondrate
segment	space_ga	visualizing_livestock	audiology
diabetes	pharynx	fri_c4_100_25	hypothyroid
sick	rmftsa_sleepdata	fri_c2_500_10	sponge
ecoli	fri_c4_500_100	fri_c1_500_5	kdd_ipums_la_98-small
sonar	fri_c3_250_5	pollen	primary-tumor
glass	auto_price	boston	kdd_synthetic_control
soybean	fri_c1_250_25	fri_c3_250_50	glass
haberman	servo	rabe_131	lymph
spambase	analcatdata_wildcat	analcatdata_chlamydia	bridges
tae	fri_c3_500_5	fri_c1_100_50	analcatdata_reviewer
heart-c	pm10	fri_c2_250_50	white-clover
tic-tac-toe	puma32H	fri_c4_100_10	dermatology
heart-h	wisconsin	fri_c2_500_25	ecoli
heart-statlog	fri_c0_100_5	mu284	flags
vehicle	sleuth_ex1605	pollution	analcatdata_challenger
hepatitis	autoPrice	fri_c0_500_5	analcatdata_dmft
vote	meta	transplant	confidence
hypothyroid	analcatdata_election2000	no2	vowel
ionosphere	analcatdata_olympic2000	mbagrade	arrhythmia
waveform-5000	cpu_act	fri_c0_500_50	kdd_ipums_la_99-small
iris	fri_c2_100_10	fri_c0_100_25	mfeat-karhunen
zoo	fri_c0_250_10	cloud	page-blocks
lung-cancer	analcatdata_apnea3	sleuth_case1202	mfeat-pixel
molecular-biology_promoters	analcatdata_apnea2	sleuth_case1201	soybean
primary-tumor	fri_c1_500_50	visualizing_hamster	analcatdata_germangss
shuttle-landing-control	analcatdata_apnea1	rabe_148	grub-damage
solar-flare	fri_c3_100_25	chscase_geyser1	ada_prior
solar-flare	fri_c1_250_50	fri_c3_500_25	gina_agnostic
yeast	strikes	colleges_aaup	gina_prior2
satimage	quake	analcatdata_negotiation	gina_prior
abalone	fri_c0_250_25	chscase_census6	ada_agnostic
baseball	disclosure_x_bias	sleuth_case2002	kc1-top5
braziltourism	fri_c2_100_25	chscase_adopt	usp05
wine	fri_c0_250_5	chscase_census5	pc4
eucalyptus	sleuth_ex1714	chscase_census4	pc3
meta_all.arff	bodyfat	chscase_census3	mc2
meta_batchincremental.arff	fri_c1_500_25	chscase_census2	cm1_req
meta_ensembles.arff	rabe_265	fri_c2_250_5	mc1
meta_instanceincremental.arff	rabe_266	plasma_retinol	ar1
flags	fri_c3_100_10	fri_c3_100_5	ar3
Australian	newton_hema	fri_c4_250_50	ar4
vowel	wind_correlations	fri_c2_500_50	ar5
oil_spill	cleveland	analcatdata_seropositive	kc2
scene	witmer_census_1980	fri_c2_100_50	ar6
thyroid_sick	triazines	visualizing_soil	kc3
yeast_ml8	fri_c1_100_10	visualizing_galaxy	kc1-binary
hayes-roth	elusage	fri_c0_500_25	kc1
monks-problems-1	diabetes_numeric	disclosure_z	pc1
monks-problems-2	fri_c2_500_5	fri_c4_100_50	pc2
monks-problems-3	fri_c3_250_10	fri_c4_250_25	mw1
SPECT	fri_c2_250_25	socmob	jEdit_4.0_4.2
SPECTF	disclosure_x_tampered	fri_c1_250_10	desharnais
grub-damage	cpu	fri_c3_500_10	datatrieve
pasture	cholesterol	fri_c3_500_50	PopularKids
squash-stored	pyrim	sleuth_ex1221	teachingAssistant
squash-unstored	chscase_funds	water-treatment	AP_Omentum_Prostate
white-clover	delta_ailerons	lowbwt	AP_Breast_Omentum
sponge	hutsof99_logis	chscase_health	AP_Endometrium_Colon
aids	fri_c4_500_50	fri_c0_500_10	AP_Colon_Kidney
JapaneseVowels	kin8nm	echoMonths	AP_Endometrium_Prostate
synthetic_control	fri_c0_100_10	kidney	AP_Breast_Colon
analcatdata_boxing2	mushroom	visualizing_ethanol	AP_Omentum_Kidney
prnn_crabs	pbc	arsenic-male-bladder	AP_Ovary_Kidney
analcatdata_boxing1	rmftsa_ctoarrivals	quake	AP_Endometrium_Omentum
analcatdata_lawsuit	fri_c1_100_25	arsenic-female-bladder	AP_Prostate_Lung
irish	chscase_vine2	arsenic-female-lung	AP_Endometrium_Ovary
analcatdata_broadwaymult	chscase_vine1	arsenic-male-lung	OVA_Colon
analcatdata_bondrate	puma8NH	prnn_fglass	AP_Ovary_Uterus
analcatdata_halloffame	diggle_table_a1	spectrometer	AP_Lung_Kidney
analcatdata_asbestos	diggle_table_a2	tae	AP_Endometrium_Uterus
analcatdata_creditscore	delta_elevators	molecular-biology_promoters	AP_Breast_Ovary
analcatdata_challenger	chatfield_4	braziltourism	OVA_Ovary
prnn_synth	fri_c1_500_10	segment	pc1_req
analcatdata_cyyoung8092	boston_corrected	postoperative-patient-data	Internet-Advertisements
schizo	sensory	analcatdata_broadwaymult	Click_prediction_small
analcatdata_japansolvent	disclosure_x_noise	mfeat-morphological	Click_prediction_small
confidence	fri_c4_100_100	heart-h	Click_prediction_small
analcatdata_dmft	fri_c1_100_5	pasture	Click_prediction_small
profb	fri_c2_250_10	zoo	eating
lupus	autoMpg	analcatdata_halloffame	physionet_asystole
physionet_bradycardia	physionet_tachycardia		

TABLE A.3 – Méta-attributs OpenML utilisés dans le chapitre 3 et dans la preuve de concept du chapitre 4

NumberOfFeatures	J48.001.ErrRate
NumberOfInstances	REPTreeDepth1AUC
NumberOfInstancesWithMissingValues	REPTreeDepth1ErrRate
NumberOfMissingValues	REPTreeDepth1Kappa
NumberOfNumericFeatures	REPTreeDepth2AUC
DefaultAccuracy	REPTreeDepth2ErrRate
MajorityClassSize	REPTreeDepth2Kappa
MinorityClassSize	REPTreeDepth3AUC
NumberOfClasses	REPTreeDepth3ErrRate
Dimensionality	REPTreeDepth3Kappa
IncompleteInstanceCount	J48.00001.Kappa
InstanceCount	J48.0001.Kappa
MaxNominalAttDistinctValues	J48.001.Kappa
MeanKurtosisOfNumericAtts	JRipAUC
MeanMeansOfNumericAtts	JRipErrRate
MeanNominalAttDistinctValues	JRipKappa
MeanSkewnessOfNumericAtts	RandomTreeDepth1AUC
MeanStdDevOfNumericAtts	RandomTreeDepth1ErrRate
MinNominalAttDistinctValues	RandomTreeDepth1Kappa
NumAttributes	RandomTreeDepth2AUC
NumBinaryAtts	RandomTreeDepth2ErrRate
NumMissingValues	RandomTreeDepth2Kappa
NumNominalAtts	RandomTreeDepth3AUC
NumNumericAtts	RandomTreeDepth3ErrRate
PercentageOfBinaryAtts	RandomTreeDepth3Kappa
PercentageOfMissingValues	SimpleLogisticAUC
PercentageOfNominalAtts	SimpleLogisticErrRate
PercentageOfNumericAtts	SimpleLogisticKappa
StdvNominalAttDistinctValues	kNN_1NAUC
HoeffdingAdwin.changes	kNN_1NErrRate
HoeffdingAdwin.warnings	kNN_1NKappa
HoeffdingDDM.changes	kNN_2NAUC
HoeffdingDDM.warnings	kNN_2NErrRate
NaiveBayesAdwin.changes	kNN_2NKappa
NaiveBayesAdwin.warnings	kNN_3NAUC
NaiveBayesDdm.changes	kNN_3NErrRate
NaiveBayesDdm.warnings	kNN_3NKappa
ClassCount	NBTreeAUC
ClassEntropy	NBTreeErrRate
EquivalentNumberOfAtts	NBTreeKappa
MeanAttributeEntropy	NaiveBayesAUC
MeanMutualInformation	NaiveBayesErrRate
NegativePercentage	NaiveBayesKappa
NoiseToSignalRatio	SVMe1AUC
PositivePercentage	SVMe1ErrRate
DecisionStumpAUC	SVMe1Kappa
DecisionStumpErrRate	SVMe2AUC
DecisionStumpKappa	SVMe2ErrRate
J48.00001.AUC	SVMe2Kappa
J48.00001.ErrRate	SVMe3AUC
J48.0001.AUC	SVMe3ErrRate
J48.0001.ErrRate	SVMe3Kappa
J48.001.AUC	

TABLE A.4 – Classifieurs Weka utilisés au méta-niveau dans le chapitre 3 et dans la preuve de concept du chapitre 4

bayes.AODE	lazy.KStar
bayes.BayesNet	lazy.LBR
bayes.NaiveBayes	rules.ConjunctiveRule
functions.GaussianProcesses	rules.DecisionTable
functions.LinearRegression	rules.DTNB
functions.Logistic	rules.JRip
functions.MLPClassifier	rules.M5Rules
functions.MLPRegressor	rules.NNge
functions.MultilayerPerceptron	rules.OneR
functions.RBFClassifier	rules.PART
functions.RBFRegressor	rules.Ridor
functions.SimpleLogistic	rules.ZeroR
functions.SMO	misc.HyperPipes
functions.SMOreg	misc.VFI
functions.SPegasos	meta.AdaBoostM1
functions.VotedPerceptron	meta.AdditiveRegression
functions.Winnow	meta.Bagging
trees.ADTree	meta.ClassificationViaRegression
trees.BFTree	meta.Dagging
trees.DecisionStump	meta.Decorate
trees.FT	meta.nestedDichotomies.ND
trees.J48	meta.nestedDichotomies.ClassBalancedND
trees.J48graft	meta.nestedDichotomies.DataNearBalancedND
trees.LADTree	meta.END
trees.LMT	meta.LogitBoost
trees.M5P	meta.MultiBoostAB
trees.NBTree	meta.MultiClassClassifier
trees.RandomForest	meta.RandomCommittee
trees.REPTree	meta.RandomSubSpace
trees.SimpleCart	meta.RegressionByDiscretization
lazy.IBk	meta.RotationForest

TABLE A.5 – Méthodes de sélection d'attributs Weka utilisés au méta-niveau dans le chapitre 3 et dans la preuve de concept du chapitre 4

Méthodes de recherche	Méthodes d'évaluation
BestFirst	CfsSubsetEval
GeneticSearch	ConsistencySubsetEval
GreedyStepwise	ChiSquaredAttributeEval
LinearForwardSelection	GainRatioAttributeEval
ScatterSearchV1	InfoGainAttributeEval
SubsetSizeForwardSelection	ReliefFAttributeEval
TabuSearch	SVMAttributeEval
Ranker	SymmetricalUncertAttributeEval

TABLE A.6 – Chaines de traitement weka employés comme classifieurs de base dans les expériences comparatives du chapitre 4

A1DE	LMT
AdaBoostM1_DecisionStump	Logistic
Bagging_REPTree	LogitBoost_DecisionStump
BayesNet_K2	LWL_DecisionStump
BFTree	MultiBoostAB_DecisionStump
ConjunctiveRule	NaiveBayes
Dagging_DecisionStump	NBTree
DecisionStump	OLM
DecisionTable_BestFirst	OneR
END_ND_J48	PART
FT	RacedIncrementalLogitBoost_DecisionStump
FURIA	RandomForest
Grading_ZeroR	RandomSubSpace_REPTree
HoeffdingTree	RandomTree
HyperPipes	RBFClassifier
IB1	REPTree
IBk	Ridor
J48	RotationForest_PrincipalComponents_J48
J48graft	SimpleCart
JRip	SimpleLogistic
KStar	SMO_PolyKernel
LADTree	SMO_RBFKernel
LibLINEAR	VFI
LibSVM	ZeroR

TABLE A.7 – Jeux de données OpenML utilisés dans les expériences comparatives du chapitre 4

anneal
anneal
kr-vs-kp
labor
audiology
autos
lymph
balance-scale
breast-cancer
mfeat-fourier
breast-w
mfeat-karhunen
mfeat-morphological
car
mfeat-zernike
cmc
mushroom
nursery
colic
optdigits
credit-a
page-blocks
credit-g
pendigits
postoperative-patient-data
dermatology
segment
diabetes
ecoli
sonar
glass
soybean
haberman
spambase
tae
heart-c
tic-tac-toe
heart-h
heart-statlog
vehicle
hepatitis
vote
ionosphere
waveform-5000
iris
BNG(cmc,nominal,55296)
electricity
primary-tumor
solar-flare
solar-flare
adult
yeast
satimage
abalone
braziltourism
eucalyptus
BNG(breast-w)
meta_all
meta_batchincremental
meta_ensembles
meta_instanceincremental
lung-cancer
wine
hypothyroid
shuttle-landing-control
Australian
sick
monks-problems-1
monks-problems-2
monks-problems-3
SPECT
SPECTF
grub-damage
pasture
squash-stored
squash-unstored
white-clover
aids
JapaneseVowels
ipums_la_97-small
analcatdata_boxing2
prnn_crabs
analcatdata_boxing1
analcatdata_lawsuit
irish
analcatdata_broadwaymult
analcatdata_authorship
analcatdata_asbestos
analcatdata_creditscore
prnn_synth
analcatdata_cyyoung8092
schizo
confidence
analcatdata_dmft
profb
lupus
analcatdata_germangss
prnn_viruses
biomed
rmftsa_sleepdata
sleuth_ex2016
sleuth_ex2015
visualizing_livestock
diggle_table_a2
fruitfly
fri_c3_1000_25
fri_c3_100_50
rmftsa_ladata
veteran
abalone
pwLinear
pol
fri_c4_1000_25
analcatdata_vineyard
bank8FM
fri_c2_100_5
2dplanes
analcatdata_supreme
visualizing_slope
fri_c1_250_5
baskball
fri_c0_250_50
machine_cpu
ailerons
cpu_small
visualizing_environmental
space_ga
sleep
fri_c3_1000_10
rmftsa_sleepdata
fri_c1_1000_5
fri_c3_250_5
auto_price
fri_c1_250_25
servo
analcatdata_wildcat
fri_c3_500_5
pm10
fri_c4_1000_10
puma32H
wisconsin
fri_c0_100_5
sleuth_ex1605
autoPrice
meta
cpu_act
fri_c2_100_10
fri_c0_250_10
analcatdata_apnea3
analcatdata_apnea2
fri_c1_500_50
analcatdata_apnea1
fri_c3_100_25
fri_c1_250_50
strikes
quake
fri_c0_250_25
disclosure_x_bias
fri_c2_100_25
fri_c0_250_5
sleuth_ex1714
bodyfat
fri_c1_500_25
rabe_265
rabe_266
fri_c3_100_10
newton_hema
wind_correlations
cleveland
triazines
fri_c1_100_10
elusage
diabetes_numeric
fri_c2_500_5
fri_c3_250_10
fri_c2_250_25
disclosure_x_tampered
cpu
fri_c4_1000_50
cholesterol
fri_c0_1000_5
pyrim
pbcseq
delta_ailerons
hutsof99_logis
fri_c4_500_50
fri_c3_1000_50
kin8nm
fri_c0_100_10
mushroom
pbc
rmftsa_ctoarrivals
fri_c1_100_25
fri_c3_1000_5
chscase_vine2
chscase_vine1
puma8NH
diggle_table_a1
diggle_table_a2
delta_elevators
chatfield_4
house_16H
cal_housing
houses
fri_c1_500_10
boston_corrected
sensory
disclosure_x_noise
fri_c1_100_5
fri_c2_250_10
autoMpg
fri_c3_250_25
bank32nh
analcatdata_vehicle
sleuth_case1102
fri_c1_1000_50
fri_c4_500_25
kdd_el_nino-small
autoHorse
stock
analcatdata_runshoes
house_8L
breastTumor
fri_c0_1000_10
elevators
wind
schlvote
fri_c0_1000_25
fri_c0_100_50
analcatdata_gsssexsurvey
housing
fishcatch
fri_c4_500_10
bolts
hungarian
vinnie
auto93
sleuth_ex2016
fri_c4_250_10
sleuth_ex2015
fri_c2_1000_50
visualizing_livestock
fri_c4_100_25
fri_c2_500_10
fri_c1_500_5
pollen
boston
fri_c3_250_50
rabe_131
analcatdata_chlamydia
fri_c1_100_50
fri_c2_250_50
fri_c4_100_10
fri_c2_500_25
mu284
mv
pollution
fri_c0_500_5
transplant
no2
mbagrade
fri_c0_500_50
fri_c0_100_25
cloud
sleuth_case1202
visualizing_hamster
rabe_148
chscase_geyser1
fri_c3_500_25
hip
analcatdata_negotiation
chscase_census6
fried
sleuth_case2002
fri_c2_1000_25
fri_c0_1000_50
chscase_census5
chscase_census4
chscase_census3
chscase_census2
fri_c1_1000_10
fri_c2_250_5
fri_c2_1000_5
fri_c2_1000_10
plasma_retinol
fri_c3_100_5
fri_c1_1000_25
fri_c4_250_50
fri_c2_500_50
analcatdata_seropositive
fri_c2_100_50
visualizing_soil
visualizing_galaxy
fri_c0_500_25
disclosure_z
fri_c4_100_50
fri_c4_250_25
socmob
fri_c1_250_10
fri_c3_500_10
fri_c3_500_50
lowbwt
fri_c0_500_10
echoMonths
kidney
visualizing_ethanol
arsenic-male-bladder
quake
arsenic-female-bladder
arsenic-female-lung
arsenic-male-lung
tae
braziltourism
segment
nursery
postoperative-patient-data
analcatdata_broadwaymult
mfeat-morphological
heart-h
pasture
cars
analcatdata_birthday
iris
analcatdata_authorship
mfeat-fourier
squash-stored
wine
hayes-roth
autos
kdd_JapaneseVowels
letter
waveform-5000
optdigits
kdd_internet_usage
heart-c
cmc
squash-unstored
analcatdata_marketing
fl2000
anneal
eucalyptus
car
kdd_ipums_la_97-small
vehicle
mfeat-zernike
prnn_fglass
balance-scale
audiology
hypothyroid
kdd_ipums_la_98-small
primary-tumor
glass
lymph
white-clover
dermatology
ecoli
analcatdata_challenger
analcatdata_dmft
confidence
kdd_ipums_la_99-small
pendigits
mfeat-karhunen
page-blocks
soybean
analcatdata_germangss
grub-damage
ada_prior
ada_agnostic
eye_movements
kc1-top5
mozilla4
jEdit_4.2_4.3
pc4
pc3
jm1
mc2
cm1_req
mc1
ar1
ar3
ar4
ar5
kc2
ar6
kc3
kc1-binary
kc1
pc1
pc2
mw1
jEdit_4.0_4.2
desharnais
datatrieve
teachingAssistant
pc1_req

TABLE A.8 – Méta-attributs généraux standards définis et implémentés pour OpenML (voir 6.1).

Name	Description
Dimensionality	Number of attributes divided by the number of instances.
MajorityClassPercentage	Percentage of instances belonging to the most frequent class. This will be the accuracy of the default classifier, which always predict the most frequent class.
MajorityClassSize	Number of instances belonging to the most frequent class.
MinorityClassPercentage	Percentage of instances belonging to the least frequent class.
MinorityClassSize	Number of instances belonging to the least frequent class.
NumberOfBinaryFeatures	Number of binary attributes.
NumberOfClasses	Number of distinct values of the target attribute (if it is nominal).
NumberOfFeatures	Number of attributes (columns) of the dataset.
NumberOfInstances	Number of instances (rows) of the dataset.
NumberOfInstancesWithMissingValues	Number of instances with at least one value missing.
NumberOfMissingValues	Number of missing values in the dataset.
NumberOfNumericFeatures	Number of numeric attributes.
NumberOfSymbolicFeatures	Number of nominal attributes.
PercentageOfBinaryFeatures	Percentage of binary attributes.
PercentageOfInstancesWithMissingValues	Percentage of instances having missing values.
PercentageOfMissingValues	Percentage of missing values.
PercentageOfNumericFeatures	Percentage of numeric attributes.
PercentageOfSymbolicFeatures	Percentage of nominal attributes.
AutoCorrelation	Average class difference between consecutive instances.

TABLE A.9 – Méta-attributs généraux décrivant des types d'attributs définis et implémentés pour OpenML (voir 6.1).

Name	Description
MeanMeansOfNumericAtts	Mean of means among attributes of the numeric type.
MeanStdDevOfNumericAtts	Mean standard deviation of attributes of the numeric type.
MeanKurtosisOfNumericAtts	Mean kurtosis among attributes of the numeric type.
MeanSkewnessOfNumericAtts	Mean skewness among attributes of the numeric type.
MinMeansOfNumericAtts	Minimum of means among attributes of the numeric type.
MinStdDevOfNumericAtts	Minimum standard deviation of attributes of the numeric type.
MinKurtosisOfNumericAtts	Minimum kurtosis among attributes of the numeric type.
MinSkewnessOfNumericAtts	Minimum skewness among attributes of the numeric type.
MaxMeansOfNumericAtts	Maximum of means among attributes of the numeric type.
MaxStdDevOfNumericAtts	Maximum standard deviation of attributes of the numeric type.
MaxKurtosisOfNumericAtts	Maximum kurtosis among attributes of the numeric type.
MaxSkewnessOfNumericAtts	Maximum skewness among attributes of the numeric type.
Quartile1MeansOfNumericAtts	First quartile of means among attributes of the numeric type.
Quartile1StdDevOfNumericAtts	First quartile of standard deviation of attributes of the numeric type.
Quartile1KurtosisOfNumericAtts	First quartile of kurtosis among attributes of the numeric type.
Quartile1SkewnessOfNumericAtts	First quartile of skewness among attributes of the numeric type.
Quartile2MeansOfNumericAtts	Second quartile (Median) of means among attributes of the numeric type.
Quartile2StdDevOfNumericAtts	Second quartile (Median) of standard deviation of attributes of the numeric type.
Quartile2KurtosisOfNumericAtts	Second quartile (Median) of kurtosis among attributes of the numeric type.
Quartile2SkewnessOfNumericAtts	Second quartile (Median) of skewness among attributes of the numeric type.
Quartile3MeansOfNumericAtts	Third quartile of means among attributes of the numeric type.
Quartile3StdDevOfNumericAtts	Third quartile of standard deviation of attributes of the numeric type.
Quartile3KurtosisOfNumericAtts	Third quartile of kurtosis among attributes of the numeric type.
Quartile3SkewnessOfNumericAtts	Third quartile of skewness among attributes of the numeric type.
MaxNominalAttDistinctValues	The maximum number of distinct values among attributes of the nominal type.
MinNominalAttDistinctValues	The minimal number of distinct values among attributes of the nominal type.
MeanNominalAttDistinctValues	Average number of distinct values among the attributes of the nominal type.
StdvNominalAttDistinctValues	Standard deviation of the number of distinct values among attributes of the nominal type.

TABLE A.10 – Méta-attributs généraux information théorétiques définis et implémentés pour OpenML (voir 6.1).

Name	Description
ClassEntropy	Entropy of the target attribute values.
EquivalentNumberOfAtts	Number of attributes needed to optimally describe the class (under the assumption of independence among attributes). Equals ClassEntropy divided by MeanMutualInformation.
NoiseToSignalRatio	An estimate of the amount of irrelevant information in the attributes regarding the class. Equals (MeanAttributeEntropy - MeanMutualInformation) divided by MeanMutualInformation.
MeanAttributeEntropy	Average entropy of the attributes.
MeanMutualInformation	Average mutual information between the attributes and the target attribute.
MinAttributeEntropy	Minimal entropy among attributes.
MinMutualInformation	Minimal mutual information between the attributes and the target attribute.
MaxAttributeEntropy	Maximum entropy among attributes.
MaxMutualInformation	Maximum mutual information between the attributes and the target attribute.
Quartile1AttributeEntropy	First quartile of entropy among attributes.
Quartile1MutualInformation	First quartile of mutual information between the attributes and the target attribute.
Quartile2AttributeEntropy	Second quartile (Median) of entropy among attributes.
Quartile2MutualInformation	Second quartile (Median) of mutual information between the attributes and the target attribute.
Quartile3AttributeEntropy	Third quartile of entropy among attributes.
Quartile3MutualInformation	Third quartile of mutual information between the attributes and the target attribute.
MeanJointEntropy	Average entropy of the combined system of variables from the pairs (attribute value, class value).
MinJointEntropy	Minimal among attributes of entropy of the combined system of variables from the pairs (attribute value, class value).
MaxJointEntropy	Maximum among attributes of entropy of the combined system of variables from the pairs (attribute value, class value).
Quartile1JointEntropy	First quartile among attributes of entropy of the combined system of variables from the pairs (attribute value, class value).
Quartile2JointEntropy	Second quartile (Median) among attributes of entropy of the combined system of variables from the pairs (attribute value, class value).
Quartile3JointEntropy	Third quartile among attributes of entropy of the combined system of variables from the pairs (attribute value, class value).

TABLE A.11 – Méta-attributs généraux de landmarking (AUC) définis et implémentés pour OpenML (voir 6.1).

Name	Description
CfsSubsetEval_DecisionStumpAUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.DecisionStump -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
CfsSubsetEval_kNN1NAUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.lazy.IBk -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
CfsSubsetEval_NaiveBayesAUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.bayes.NaiveBayes -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
DecisionStumpAUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.DecisionStump
J48.00001.AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.J48 -C .00001
J48.0001.AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.J48 -C .0001
J48.001.AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.J48 -C .001
kNN1NAUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.lazy.IBk
NaiveBayesAUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.bayes.NaiveBayes
RandomTreeDepth1AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.RandomTree -depth 1
RandomTreeDepth2AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.RandomTree -depth 2
RandomTreeDepth3AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.RandomTree -depth 3
REPTreeDepth1AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.REPTree -L 1
REPTreeDepth2AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.REPTree -L 2
REPTreeDepth3AUC	Area Under the ROC Curve achieved by the landmarker weka.classifiers.trees.REPTree -L 3

TABLE A.12 – Méta-attributs généraux de landmarking (Error rate) définis et implémentés pour OpenML (voir 6.1).

Name	Description
CfsSubsetEval_DecisionStumpErrRate	Error rate achieved by the landmarker weka.classifiers.trees.DecisionStump -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
CfsSubsetEval_kNN1NErrRate	Error rate achieved by the landmarker weka.classifiers.lazy.IBk -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
CfsSubsetEval_NaiveBayesErrRate	Error rate achieved by the landmarker weka.classifiers.bayes.NaiveBayes -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
DecisionStumpErrRate	Error rate achieved by the landmarker weka.classifiers.trees.DecisionStump
J48.00001.ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.J48 -C .00001
J48.0001.ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.J48 -C .0001
J48.001.ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.J48 -C .001
kNN1NErrRate	Error rate achieved by the landmarker weka.classifiers.lazy.IBk
NaiveBayesErrRate	Error rate achieved by the landmarker weka.classifiers.bayes.NaiveBayes
RandomTreeDepth1ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.RandomTree -depth 1
RandomTreeDepth2ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.RandomTree -depth 2
RandomTreeDepth3ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.RandomTree -depth 3
REPTreeDepth1ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.REPTree -L 1
REPTreeDepth2ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.REPTree -L 2
REPTreeDepth3ErrRate	Error rate achieved by the landmarker weka.classifiers.trees.REPTree -L 3

TABLE A.13 – Méta-attributs généraux de landmarking (Kappa) définis et implémentés pour OpenML (voir 6.1).

Name	Description
CfsSubsetEval_DecisionStumpKappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.DecisionStump -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
CfsSubsetEval_kNN1NKappa	Kappa coefficient achieved by the landmarker weka.classifiers.lazy.IBk -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
CfsSubsetEval_NaiveBayesKappa	Kappa coefficient achieved by the landmarker weka.classifiers.bayes.NaiveBayes -E weka.attributeSelection.CfsSubsetEval -P 1 -E 1 -S weka.attributeSelection.BestFirst -D 1 -N 5 -W
DecisionStumpKappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.DecisionStump
J48.00001.Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.J48 -C .00001
J48.0001.Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.J48 -C .0001
J48.001.Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.J48 -C .001
kNN1NKappa	Kappa coefficient achieved by the landmarker weka.classifiers.lazy.IBk
NaiveBayesKappa	Kappa coefficient achieved by the landmarker weka.classifiers.bayes.NaiveBayes
RandomTreeDepth1Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.RandomTree -depth 1
RandomTreeDepth2Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.RandomTree -depth 2
RandomTreeDepth3Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.RandomTree -depth 3
REPTreeDepth1Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.REPTree -L 1
REPTreeDepth2Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.REPTree -L 2
REPTreeDepth3Kappa	Kappa coefficient achieved by the landmarker weka.classifiers.trees.REPTree -L 3

TABLE A.14 – Méta-attributs des attributs sans restriction de type définis et implémentés pour OpenML (voir 6.1).

Name	Description
ValuesCount	Total number of values. Equivalent to the number of instances in the dataset.
NonMissingValuesCount	Number of non missing values.
MissingValuesCount	Number of missing values.
Distinct	Number of distinct attribute values.
AverageClassCount	Average count of occurrences of the distinct attribute values.
MostFequentClassCount	Count of the most occuring attribute value.
LeastFequentClassCount	Count of the least occuring attribute value.
ModeClassCount	Most frequent count of occurrences of the distinct attribute values.
MedianClassCount	Median count of occurrences of the distinct attribute values.
PearsonCorrellationCoefficient	Pearson Correlation Coeficient between the attribute values and the target attribute.
SpearmanCorrelationCoefficient	Spearman Correlation Coeficient between the attribute values and the target attribute.
CovarianceWithTarget	Covariance between the attribute values and the target attribute.
Entropy	Entropy of the attribute values.
JointEntropy	Total entropy of the combined system of variables from the pairs (attribute value, class value).
MutualInformation	Measures the information conveyed by the attribute about the class.
EquivalentNumberOfAtts	Number of attributes as informative as this one needed to optimally describe the class (under the assumption of independence among attributes). Equals ClassEntropy divided by MutualInformation.
NoiseToSignalRatio	An estimate of the amount of irrelevant information in the attribute regarding the class. Equals (AttributeEntropy - MutualInformation) divided by MutualInformation.
MissingValues	1 if the attribute has at least one missing value, 0 otherwise.
AveragePercentageOfClass	Average occurrence rate of the distinct attribute values.
PercentageOfMissing	Percentage of missing attribute values.
PercentageOfNonMissing	Percentage of non-missing attribute values.
PercentageOfMostFrequentClass	Occurrence rate of the most frequent attribute value.
PercentageOfLeastFrequentClass	Occurrence rate of the least frequent attribute value.
ModeClassPercentage	Most frequent occurrence rate of the distinct attribute values.
MedianClassPercentage	Median occurrence rate of the distinct attribute values.

TABLE A.15 – Méta-attributs des attributs avec restriction de type définis et implémentés pour OpenML (voir 6.1).

Name	Description
IsUniform	0 if the Kolmogorov-Smirnov test rejects H0 (the attribute distribution is a uniform distribution) at the 0.05 significance level, 1 otherwise.
IntegersOnly	1 if attribute values consist exclusively of integers, 0 otherwise.
Min	Minimum of the attribute values.
Max	Maximum of the attribute values.
Kurtosis	Kurtosis of the attribute values.
Mean	Mean of the attribute values.
Skewness	Skewness of the attribute values.
StandardDeviation	Standard deviation of the attribute values.
Variance	Variance of the attribute values.
Mode	Mode of the attribute values.
Median	Median of the attribute values.
ValueRange	Difference between maximum and minimum of the attribute values.
LowerOuterFence	Lower outer fence of the attribute values. Values beyond are strong outliers.
HigherOuterFence	Higher outer fence of the attribute values. Values beyond are strong outliers.
LowerQuartile	Lower quartile (25th percentile) of the attribute values.
HigherQuartile	Higher quartile (75th percentile) of the attribute values.
HigherConfidence	Higher confidence interval (for the 95% confidence level) of the attribute values.
LowerConfidence	Lower confidence interval (for the 95% confidence level) of the attribute values.
PositiveCount	Number of positive attribute values.
NegativeCount	Number of stricly negative attribute values.
UniformDiscrete	0 if the chi-squared test rejects H0 (the attribute distribution is a uniform discrete distribution) at the 0.05 significance level, 1 otherwise.
ChiSquareUniformDistribution	Statistic value of the chi-squared test for H0 (the attribute distribution is a uniform discrete distribution) .
RationOfDistinguishingCategories ByKolmogorovSmirnoffSlashChiSquare	Ratio of attribute values that after sub-setting the dataset to that attribute value lead to different distribution of the target as indicated by the Kolmogorov-Smirnoff statistical test.
RationOfDistinguishingCategories ByUtest	Ratio of attribute values that after sub-setting the dataset to that attribute value lead to different distribution of the target as indicated by the Mann-Whitney U-test.
PositivePercentage	Percentage of positive attribute values.
NegativePercentage	Percentage of strictly negative attribute values.
HasPositiveValues	1 if the attribute has at least one positive value, 0 otherwise.
HasNegativeValues	1 if the attribute has at least one strictly negative value, 0 otherwise.

TABLE A.16 – Jeux de données OpenML utilisés dans les expériences de comparaison de méta-attributs en 6.2.1

kr-vs-kp	pol
labor	2dplanes
breast-cancer	analcatdata_supreme
breast-w	visualizing_slope
colic	machine_cpu
credit-a	ailerons
credit-g	cpu_small
cylinder-bands	visualizing_environmental
diabetes	space_ga
sonar	pharynx
haberman	sleep
spambase	autoPrice
tic-tac-toe	meta
heart-statlog	analcatdata_election2000
hepatitis	analcatdata_uktrainacc
vote	cpu_act
ionosphere	analcatdata_michiganacc
electricity	quake
solar-flare	newton_hema
adult	wind_correlations
shuttle-landing-control	cleveland
Australian	witmer_census_1980
sick	triazines
mammography	disclosure_x_tampered
oil_spill	pyrim
thyroid_sick	chscase_funds
monks-problems-1	pbcseq
SPECT	delta_ailerons
SPECTF	hutsof99_logis
analcatdata_boxing1	chscase_vine2
analcatdata_lawsuit	chscase_vine1
analcatdata_asbestos	puma8NH
analcatdata_creditscore	diggle_table_a1
backache	delta_elevators
prnn_synth	chatfield_4
analcatdata_cyyoung8092	house_16H
schizo	cal_housing
analcatdata_japansolvent	kdd_el_nino-small
profb	autoHorse
lupus	stock
analcatdata_bankruptcy	analcatdata_runshoes
biomed	house_8L
sleuth_ex2015	breastTumor
vineyard	fri_c0_1000_10
fruitfly	elevators
fri_c3_100_50	wind
rmftsa_ladata	schlvote
veteran	bolts
abalone	hungarian
pwLinear	analcatdata_gviolence

TABLE A.17 – Chaines de traitement weka employés comme classifieurs de base dans les expériences de comparaison de méta-attributs en 6.2.1

weka.J48
weka.IBk
weka.REPTree
weka.RandomForest
weka.HoeffdingTree
weka.JRip
weka.NaiveBayes
weka.Logistic
weka.MultilayerPerceptron
weka.SGD
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_J48
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_REPTree
weka.SMO_PolyKernel
weka.AdaBoostM1_DecisionStump
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_IBk
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_NaiveBayes
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_RandomForest
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_HoeffdingTree
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_SGD
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_SMO_PolyKernel
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_Logistic
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_MultilayerPerceptron
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_JRip
weka.AttributeSelectedClassifier_CfsSubsetEval_BestFirst_AdaBoostM1_DecisionStump
weka.LibSVM

TABLE A.18 – Tâches OpenML utilisés dans la preuve de concept du chapitre 5

Dataset name	Description
tic-tac-toe	Binomial classification 10 attributes * 1000 instances.
phoneme	Binomial classification 6 attributes * 5400 instances.
diabetes	Binomial classification 9 attributes * 700 instances.
wilt	Binomial classification 6 attributes * 4800 instances. Very Imbalanced.
Australian	Binomial classification 15 attributes * 700 instances.
optdigits	Multinomial classification 65 attributes * 5600 instances.
letter	Multinomial classification 17 attributes * 20000 instances.
spectrometer	Multinomial classification 100 attributes * 500 instances. Very Imbalanced.
satimage	Multinomial classification 37 attributes * 6400 instances.
yeast	Multinomial classification 9 attributes * 1500 instances. Very Imbalanced.
electricity	Binomial classification 9 attributes * 45000 instances.
spambase	Binomial classification 58 attributes * 4600 instances.
iris	Multinomial classification 5 attributes * 150 instances.
anneal	Multinomial classification 39 attributes * 900 instances. Very Imbalanced.
car	Multinomial classification 7 attributes * 1700 instances. Imbalanced.

TABLE A.19 – Algorithmes utilisés dans la preuve de concept du chapitre 5

Algorithm	Description
Bagging(J48)	Machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method.
A1DE	AODE achieves highly accurate classification by averaging over all of a small space of alternative naive-Bayes-like models that have weaker (and hence less detrimental) independence assumptions than naive Bayes. The resulting algorithm is computationally efficient while delivering highly accurate classification on many learning tasks.
Hyperpipes	For each category a HyperPipe is constructed that contains all points of that category (essentially records the attribute bounds observed for each category). Test instances are classified according to the category that "most contains the instance".
J48	Implementation of C4.5 that builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy.
Multilayer Perceptron	A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs.
Naive Bayes	It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
Random Forest	Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees.
SMO	Sequential minimal optimization (SMO) is an algorithm for solving the quadratic programming (QP) problem that arises during the training of support vector machines.
Adaboost(DecisionStump)	AdaBoost, short for "Adaptive Boosting" can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier.
IBk	K-nearest neighbours classifier. Can select appropriate value of K based on cross-validation. Can also do distance weighting.