Dissimilarités entre jeux de données

William Raynaut, Chantal Soulé-Dupuy, Nathalie Vallès-Parlangeau

Dissimilarités entre jeux de données

William Raynaut
Chantal Soule-Dupuy
Nathalie Valles-Parlangeau

Rapport IRIT/RR--2017--07--FR
3 avril 2017

Résumé

La caractérisation de jeu 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.

Table des matières

1 Motivation **5**
1.1 Introduction 5
1.2 Approches précédentes 6
1.3 Exemple 7

2 Fonction de dissimilarité **9**
2.1 Propriétés désirables 9
2.2 Fonction candidate 10

3 Preuve de concept **19**
3.1 Évaluation Théorique 19
3.2 Évaluation Expérimentale 22

4 Expériences comparatives **27**
4.1 Forme du méta-problème 27
4.2 Ensembles de méta-attributs 30
4.3 Fonctions de dissimilarité 36
4.4 Cadre d'expérimentation 39
4.5 Analyse dimensionnelle des résultats 42

5 Discussion **63**
5.1 Récapitulatif des résultats 63
5.2 Conclusion 65

A Listings **70**

Chapitre 1

Motivation

1.1 Introduction

L'émergence du phénomène de données massives crée un besoin grandissant en analyse de données, et bien souvent, cette analyse est 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 [46, 31]. La caractérisation de jeu de données, problème apparu avec les premières ébauches de méta-apprentissage [13], 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 [19]. 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.

1.2 Approches précédentes

On peut distinguer deux catégories d'approches au problème de caractérisation de jeux de données :

- 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 [26], et employée dans une majorité d'études postérieures [40, 18, 23, 35, 24], 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 [20], mais les résultats expérimentaux ne permettent pas de conclure à la supériorité de quelconque mesure indépendamment du méta-apprentissage employé [37].

- 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 [30], cette approche emploie initialement le taux d'erreur d'un ensemble d'algorithmes basiques comme métadonnées. 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 [29] proposant comme méta-attributs des propriétés structurelles d'un arbre de décision construit sur la donnée. Les expériences conduites par [12] 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.

Le problème de caractérisation de jeux de données a donc déjà reçu une certaine attention dans le domaine du méta-apprentissage, 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 [17]. On peut illustrer ce problème sur l'exemple suivant.

1.3 Exemple

Considérons deux jeux de données, **A** et **B** illustrés en figure 1.1. **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**).

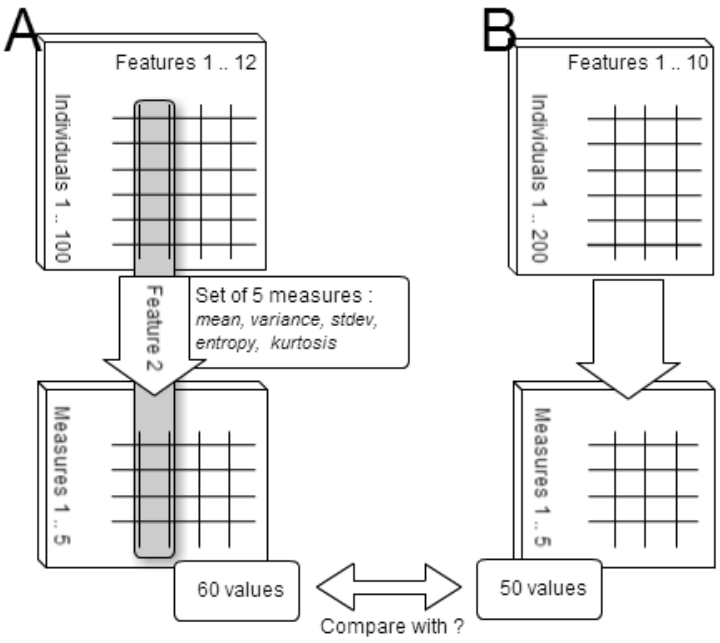

FIGURE 1.1 – Propriétés d'attributs individuels

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 [17, 42] 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 1.2)).

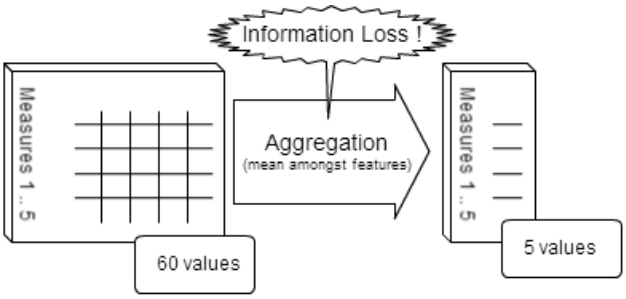

FIGURE 1.2 – Moyenne sur les attributs

Notre approche est de 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 1.3)).

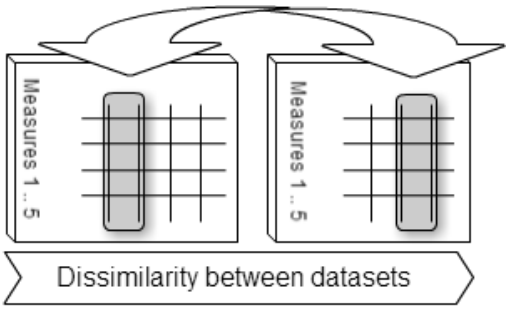

FIGURE 1.3 – Comparaison directe d'attributs

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

Chapitre 2

Fonction de dissimilarité

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 [41], ne conservant que la positivité ($d(x, x') \geq 0$). Nous préférerions cependant conserver uniquement des propriétés 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 1 *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 2 *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.

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 3 *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] *:*

$$\forall a, b \in A,\ d_A^{ubr}(a, b) = \sum_{i=1}^{n} \frac{d_i(a_i, b_i)}{\max_{(x,y)\in A^2}(d_i(x_i, y_i))} \tag{2.1}$$

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*

1. *ubr* pour *upper bound relative*

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

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 $d_H(H(x), H(y)) = d_H(x, y)$.

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 lesquels 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. 2.1) sur $G(\omega)$ selon $\{d_g | g \in G\}$:

$$\forall x, y \in \omega,\ d_{G(\omega)}^{ubr}(x, y) = \sum_{g\in G} \frac{d_g(x, y)}{\max_{(x',y')\in\omega^2}(d_g(x', y'))} \tag{2.2}$$

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.

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

$$\delta_F(x_i, x'_j) = \sum_{f \in F} \frac{\delta_f(x_i, x'_j)}{\max\limits_{\substack{(y,y') \in \omega^2 \\ (p,q) \in \mathbb{N}^2}} \delta_f(y_p, y'_q)))} \tag{2.3}$$

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 1.3, l'objectif est de comparer ces attributs par paires. Ceci requiert un mapping entre les attributs de deux jeux de données :

Définition 4 *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 2.1)*

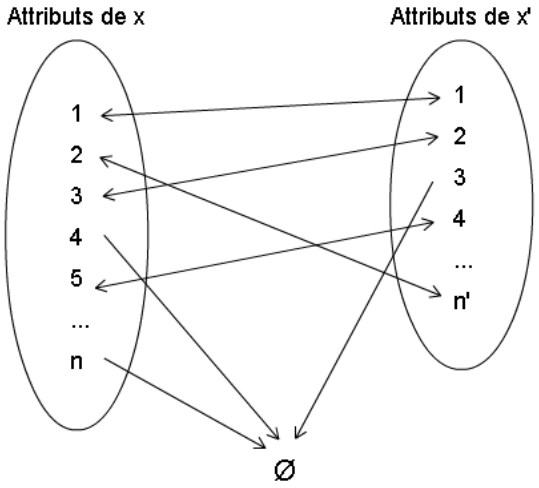

FIGURE 2.1 – Exemple de fonction de Mapping

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

$$d^{\sigma}_{F(\omega)}(x, x') = \frac{1}{max(n, n')}\Big(\sum_{\sigma_1(i)=j}^{i,j} \delta_F(x_i, x'_j) + \sum_{\sigma_1(i)=\emptyset}^{i} \delta_F(x_i, \emptyset) + \sum_{\sigma_2(j)=\emptyset}^{j} \delta_F(\emptyset, x'_j)\Big) \tag{2.4}$$

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 2).*

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

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 [32] 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 2.2, qui seront qualifiés de "Split".

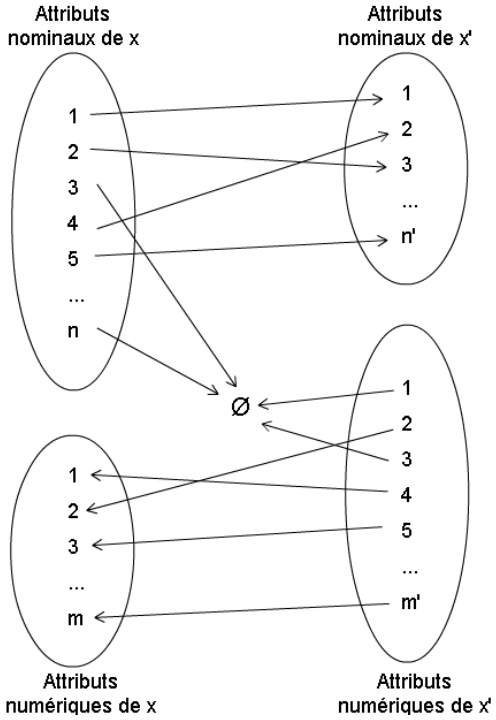

FIGURE 2.2 – Mapping "Split" avec séparation des attributs par type

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 2.3).

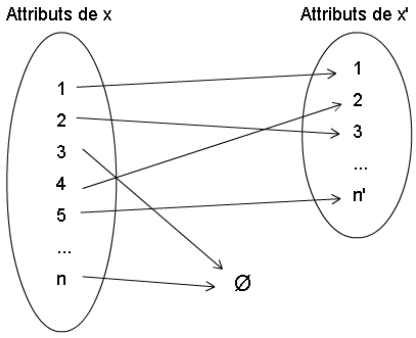

FIGURE 2.3 – Mapping "Mix" sans séparation des attributs par type

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 [32], selon l'algorithme de Kuhn-Munkres (ou *algorithme hongrois*) [22]. 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 2.1 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 2.1 – Matrice des dissimilarités entre les attributs de x et x'

L'algorithme de Kuhn-Munkres plus tard révisé par Edmonds et Karp, et indépendamment Tomizawa [9], fourni 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 2.2). 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 2.2 – 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 en Algorithme 1.

Algorithme 1 : Méthode de minimisation naïve : "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 un normalisation par la borne supérieure, donc potentiellement couteuse 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.

On peut alors récapituler les différentes fonction de mapping considérées en Table 2.3 :

	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 2.3 – Fonctions de mapping considérées

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 ou tous les attributs sont de même type, équivalentes aux méthodes "Mix".

2.2.4 Composition

Par le biais des équations 2.2 et 2.4, 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 :

$$d^{ubr}_{\omega}(x,y) = \frac{d^{ubr}_{G(\omega)}(x,y)}{\max\limits_{(x',y')\in\omega^2}(d^{ubr}_{G(\omega)}(x',y'))} + \frac{d^{\sigma}_{F(\omega)}(x,y)}{\max\limits_{(x',y')\in\omega^2}(d^{\sigma}_{F(\omega)}(x',y'))} \quad (2.5)$$

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

Chapitre 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 "Greedy-Mix" (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.

Dans un premier temps, nous étudierons son potentiel pour l'apprentissage, en utilisant les méthodes d'estimation présentées dans [41]. Dans un second temps, nous décrirons une importante expérience de méta-apprentissage évaluant 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 avons concentré nos efforts sur les problèmes de classification. En effet, s'agissant du cas le plus typique de l'apprentissage, il est notablement plus facile de collecter des expériences documentées en classification. La description de ces expériences proviendra d'OpenML [39], une importante base collaborative d'expérience d'apprentissage.

3.1 Évaluation Théorique

Dans [41], 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 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 [41], 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 situation 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 a été amené aussi haut que possible en conservant $\epsilon \leq 0.05$. Les résultats sont présentés en Table 3.1, moyennés selon les classifieurs et jeux de données. La Table 3.2 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 3.1 – (ϵ, γ)-*goodness* moyenne avec différentes dissimilarités

Comme on peut le voir, la dissimilarité proposée offre des bornes intéressantes au taux d'erreur avec sensiblement moins d'exemples (au pire 11.3% d'erreur pour 20k exemples, ce qui n'est atteint par la distance euclidienne qu'autour de 50k...). 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 3.2 – Borne du taux d'erreur obtenu avec une probabilité $1 - \delta$ pour différents nombres d'exemples et par des classifieurs construits sur différentes dissimilarités.

3.2 Évaluation Expérimentale

Afin d'évaluer l'intérêt de la dissimilarité dans le cadre de la sélection d'algorithme par méta-apprentissage, nous avons mis en place une série d'expériences comparatives autour du problème de méta-classification. Nous illustrerons tout d'abord ce problème par un exemple d'exécution, puis détaillerons les facteurs qui varieront au fil des exécutions.

3.2.1 Exemple d'exécution

Pour un jeu de données de classification particulier, l'objectif d'une expérience de sélection d'algorithme est de choisir un classifieur maximisant un critère donné. Dans cet exemple, on utilisera le critère traditionnel de précision (bien que biaisé [21], il est l'un des plus intuitifs). Pour construire le jeu de données de méta-classification, on extrait de la base d'OpenML deux ensembles de données. Le premier décrit la précision de différents algorithmes de classification sur un ensemble de jeux de données (Table 3.3), tandis que le second caractérise ces jeux de données selon un ensemble de méta-attributs (Table 3.4). Une liste des méta-attributs OpenML est présentée en annexe, Table A.3.

	$classifier_1$	$classifier_2$	...	$classifier_{93}$
$dataset_1$	0.8	0.9	...	...
$dataset_2$	0.9	0.7	...	...
...	...	...	...	...
$dataset_{434}$	...	...	...	...

TABLE 3.3 – Précision de 93 algorithmes de classification sur 434 jeux de données

	$NumberOfInstances$	$NumberOfFeatures$	...	$MetaAttribute_{105}$
$dataset_1$	100	62	...	...
$dataset_2$	5000	13	...	...
...	...	...	...	...
$dataset_{434}$	360	20	...	...

TABLE 3.4 – Caractérisation des jeux de données selon 105 méta-attributs

On peut alors construire le jeu de données de méta-classification représenté en Table 3.5. La classe d'une instance de ce jeu de données de méta-classification identifie quel algorithme présente la meilleure performance (*i.e.* la meilleure précision) sur le jeu de données qu'elle décrit (selon les performances recensées en Table 3.3).

	$NumberOfInstances$	...	$MetaAttribute_{105}$	$Class$
$dataset_1$	100	...	4	$classifier_{18}$
$dataset_2$	5000	...	92	$classifier_7$
...	...	...	...	...
$dataset_{434}$	360	...	13	$classifier_{63}$

TABLE 3.5 – Jeu de données de méta-classification

Il faut ensuite résoudre ce problème de méta-classification. Dans cet exemple ceci s'effectue par une forme de *"leave one out"* selon le pseudocode de l'Algorithme 2.

Algorithme 2 : Exemple d'exécution

foreach *Instance de jeu de données* $dataset_i$ **do**
- Exclure $dataset_i$ du jeu de données de méta-classification
- Appliquer une méthode de sélection d'attributs au jeu de données de méta-classification
- Appliquer un algorithme de classification au jeu de données de méta-classification réduit pour apprendre un modèle
- Utiliser ce modèle 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 caractériser 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'algorithme 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 :

Définition 5 *Soit* $\boldsymbol{p}$ *la performance du classifieur* $\boldsymbol{classifier}_j$ *sur le jeu de données* $\boldsymbol{dataset}_i$ *selon le critère choisi (ici, la précison). Soient alors* $\boldsymbol{best}$ *la performance du meilleur classifieur de* $\boldsymbol{classifier}_{1..m}$ *sur le jeu de données* $\boldsymbol{dataset}_i$*, et* $\boldsymbol{def}$ *la performance du classifieur par défaut (prédisant la classe majoritaire) sur ce même jeu de données. On définit la performance* $\boldsymbol{perf}$ *de notre expérience 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 prédit présente une valeur de précison maximale, et atteint 0 quand le classifieur prédit présente la même précision que le classifieur par défaut. Le cas négatif traduit une performance inférieure à celle du classifieur par défaut. La performance d'une expérience de méta-classification dépend donc de la qualité du classifieur sélectionné.

3.2.2 Cadre d'expérimentation

Comme stipulé précédemment, le jeu de données de méta-classification est construit en utilisant des données d'OpenML, qui répertorie plus de 2500 jeux de données et 2000 algorithmes. 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 [38] 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 au 93 algorithmes de classification et 434 jeux de données vu précédemment. Nous avons ensuite extrait les valeurs d'évaluation de 11 critères de performance sur chacune des 40k exécutions que cela représente. Enfin nous avons extrait les valeurs des 105 méta-attributs OpenML pour chaque jeu de données (listes en annexe A).

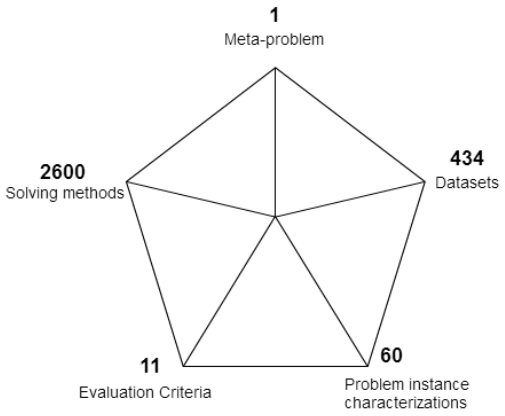

FIGURE 3.1 – Dimensions des exécutions

Pour itérer l'expérience présentée dans l'exemple précédent, il est ensuite nécessaire de définir des ensembles d'algorithmes de classification et de sélection d'attributs pour le méta-niveau. Nous avons choisi d'utiliser les algorithmes de l'API *Weka* [14], 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 l'API Weka (listes en annexe A, Tables A.4 et A.5), comprenant en particulier des méthodes d'augmentation telles le "boosting" ou "bagging". À cela, nous ajoutons un classifieur kNN (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.

Les exécutions individuelles sont instanciées automatiquement, et déléguées à un répartiteur de tâches SLURM [45] gérant les 640 nœuds du cluster *OSIRIM* (voir `osirim.irit.fr`). Les 200k exécutions résultantes totalisent plus de 700 millions de cycles apprentissage-prédiction-évaluations, ce qui, même avec des ressources importantes, reste très couteux (plus d'un mois de temps réel en exécution parallèle, pour des dizaines d'années de temps machine).

3.2.3 Résultats

Ce procédé fournit donc une valeur de performance pour chacune des 200k 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 de 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 3.2, on compare, pour différents seuils de performance, le nombre 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 performant que le meilleur, 20% sont basées sur la dissimilarité proposée, alors que cette proportion est de moins de 10% dans la population complète.

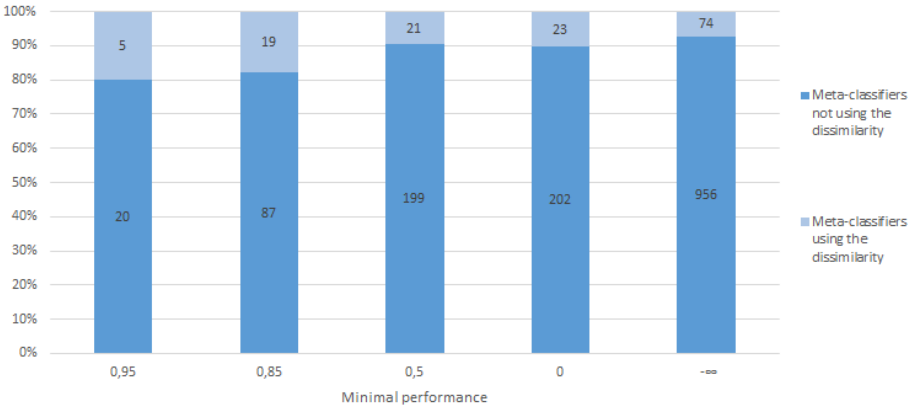

FIGURE 3.2 – Populations de méta-classifieurs atteignant divers seuils de performance

La Figure 3.3 présente la proportion de classifieurs de méta-niveau utilisant la dissimilarité 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 3.4, présentant la proportion de classifieurs de méta-niveau utilisant la dissimilarité par quintiles cumulés.

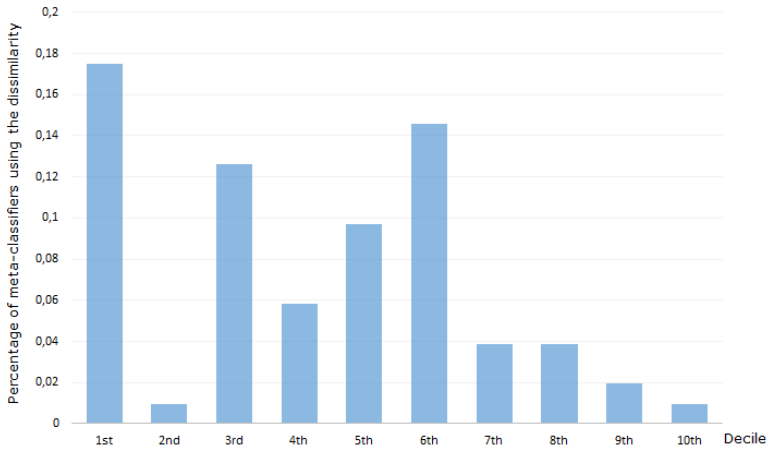

FIGURE 3.3 – Proportion dans chaque décile de performance de méta-classifieurs utilisant la dissimilarité

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"* [43, 44]. 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"* [8] ou *"ClassificationViaRegression"* [11]. 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.

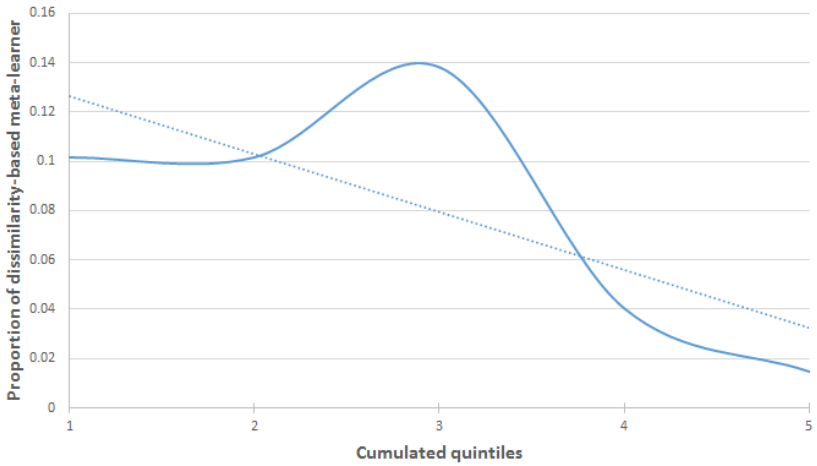

FIGURE 3.4 – Proportion de méta-classifieurs utilisant la dissimilarité par quintiles de performance cumulés

Chapitre 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 cadre expérimental approprié à leur évaluation. 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.

4.1 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 [13]. Par exemple, dans [3], 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 [19], 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 [35], 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 [23], 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, [36] 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'abondante méta-donnée 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 [3]. 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 meta-dataset $\mathcal{M}$ 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 meta-dataset 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 meta-dataset 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 6 *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 :*

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

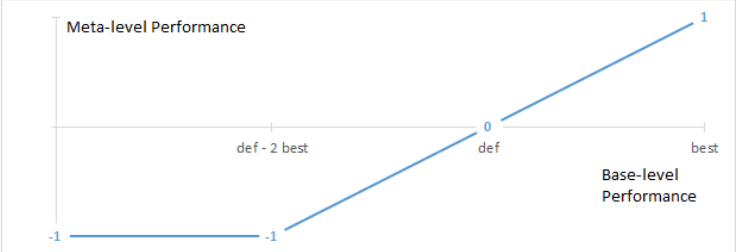

FIGURE 4.1 – Performance d'une recommandation c_i

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 $(\mathcal{M}, D)$:

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

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 :

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

4.2 Ensembles de méta-attributs

Les méta-attributs retenus dans cette expérience sont en partie repris des travaux de J. Smid [32] 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 différent 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 :

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

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.

4.2.1 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 4.1), un descriptif global des attributs numériques (Table 4.2), un descriptif global des attributs nominaux (Table 4.3), et en la performance de *landmarkers* évalués selon plusieurs critères (Tables 4.4, 4.5 et 4.6). Les *landmarkers* sont des algorithmes d'apprentissage simples, dont l'usage a été introduit dans [30], que l'on applique au jeu de données considéré, afin d'y évaluer leur performance. Ceux utilisés ici proviennent de l'API Weka [14] et rassemblent différentes techniques classiques d'apprentissage. Les critères de performance retenus sont l'aire sous la courbe ROC (Table 4.4), le taux d'erreur (Table 4.5) et le coefficient Kappa de Cohen [7] (Table 4.6), qui capturent des aspects de la performance conceptuellement assez différents pour être considérés séparément.

TABLE 4.1 – Méta-attributs simples des jeux de données

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 4.2 – Méta-attributs généraux décrivant les attributs numériques

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 4.3 – Méta-attributs généraux décrivant les attributs nominaux

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 4.4 – "Aire sous la courbe" des landmarkers

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 4.5 – Taux d'erreur 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 4.6 – Kappa de Cohen 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	$\|x\|$

On forme alors trois ensembles de méta-attributs généraux :

DMFg_min	: Tables 4.1 - 4.2 - 4.3	Aucun *landmarker*.
DMFg_red	: Tables 4.1 - 4.2 - 4.3 - 4.4	AUC des *landmarkers*.
DMFg_full	: Tables 4.1 - 4.2 - 4.3 - 4.4 - 4.5 - 4.6	Tous les *landmarkers*.

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 4.7), des propriétés exclusives aux attributs numériques (Table 4.9), des propriétés exclusives aux attributs nominaux (Table 4.8), et des versions normalisées de certaines des propriétés précédentes (Tables 4.10 et 4.11). Cette normalisation est proposée dans [32], 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 4.10 pour ceux indépendant du type d'attribut, et en Table 4.11 pour ceux exclusif aux attributs numériques.

Table 4.7 – Méta-attributs simples des attributs

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 4.8 – Méta-attributs spécifiques aux attributs nominaux

Méta-attribut	Description	$\delta^{\emptyset}(x)$
UniformDiscrete	Result of Pearson's chi-squared test for discrete uniform distribution.	$\|x-1\|$
ChiSquareUniformDistribution	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 4.9 – Méta-attributs spécifiques aux attributs numériques

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 4.10 – Méta-attributs normalisés selon le nombre d'instances

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 4.11 – Méta-attributs normalisés spécifiques aux attributs numériques

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.	$\|x\|$

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

DMFf_base : Tables 4.7 - 4.8 - 4.9 Pas de méta-attributs normalisés.
DMFf_full : Tables 4.7 - 4.8 - 4.9 - 4.10 - 4.11 Ajout des méta-attributs normalisés.

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.

4.3.1 Dissimilarités sur les méta-attributs généraux

Pour fournir une dissimilarité sur les méta-attributs généraux, on comparera le candidat présenté en définition 3, 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

4.3.2 Dissimilarités sur les attributs

Une dissimilarité sur les méta-attributs des attributs a été définie dans l'équation 2.3, se basant sur les différentes méthodes d'appariement des attributs décrites en section 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 [33], 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 7 *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 :*

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

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 2). 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 2. □

Afin de construire une fonctions de dissimilarité sur les attributs à partir de δ_{KS}, on reprend l'équation 2.4. 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 :

$$d^{\sigma}_{KS(\omega)}(x, x') = \frac{1}{max(n, n')}\left(\sum_{\sigma_1(i)=j}^{i,j} \delta_{KS}(x_i, x'_j) + \sum_{\sigma_1(i)=\emptyset}^{i} \delta_{KS}(x_i, \emptyset) + \sum_{\sigma_2(j)=\emptyset}^{j} \delta_{KS}(\emptyset, x'_j)\right) \tag{4.1}$$

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 2.3 : **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**.

4.3.3 Fonctions complètes comparées

On dispose donc de diverses variantes pour les différents éléments composant nos dissimilarités entre jeux de données. On présente en Table 4.12 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.

TABLE 4.12 – Fonctions de dissimilarité comparées

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

4.4 Cadre d'expérimentation

4.4.1 Meta-Dataset

Afin d'instancier le méta-problème décrit en section 4.1, on doit construire un *méta-dataset* 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 *méta-dataset* 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 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 4.13.

TABLE 4.13 – Critères de performance retenus

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.

On répète ensuite la procédure de recherche de cliques décrite dans la preuve de concept 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.7 et A.8. Le *méta-dataset* complet est disponible au format *ARFF* en ligne [1].

4.4.2 Baseline

Les dissimilarités à comparer au méta-niveau ont été présentées en Table 4.12 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 désirable. 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'algo-

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

rithme 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 4.14.

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

Data :
- Un meta-dataset $\mathcal{M}$ 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 meta-dataset selon l'estimation de leur performance sur D
Recommander les n meilleurs pondérés selon leur performance estimée

TABLE 4.14 – Algorithmes d'apprentissage de la *baseline*

Implémentation Weka	Description
GaussianProcesses	Gaussian Processes for regression without hyperparameter-tuning. See [25].
LinearRegression	Linear regression for prediction. Uses the Akaike criterion for model selection, and is able to deal with weighted instances. See [1].
RBFRegressor	Radial basis function networks, trained in a fully supervised manner by minimizing squared error with the BFGS method. See [10].
SMOreg	Sequential minimal optimization algorithm for training a support vector regression model. See [34].
RandomForest	Ensemble of decision trees outputting the mean prediction of the individual trees. See [4].
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 [6].
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 [16].

4.4.3 Exécutions

On évalue donc nos algorithmes d'apprentissage (Table 4.14) et dissimilarités (Table 4.12) sur le meta-dataset, respectivement selon les algorithmes 4 et 3. On explore ainsi l'espace formé sur les dimensions décrites en Table 4.15

TABLE 4.15 – Dimensions de l'expérience

Dimension	Détails	Taille
Algorithme au méta-niveau	Traditionnels (Table 4.14)	7
	Dissimilarités (Table 4.12)	270
Critère de performance de base	Table 4.13	4
Jeu de données (instance de base)	Table A.8	395
Nombre n d'algorithmes de base recommandés	$\{1, 3, 5\}$	3

Les dissimilarités apparaissent plus nombreuses qu'en Table 4.12 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 4.16.

TABLE 4.16 – Dimensions de l'espace des dissimilarités

Dimension	Détails	Taille
Méta-attributs généraux	Table 4.12	3
Dissimilarité sur les méta-attributs généraux	Table 4.12	4
Méta-attributs des attributs	Table 4.12	3
Dissimilarité sur les attributs	Table 4.12	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

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 [45] gérant les 640 nœuds du cluster *OSIRIM* (voir `osirim.irit.fr`). 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 exact des différentes dissimilarités, présenté en Figure 4.2.

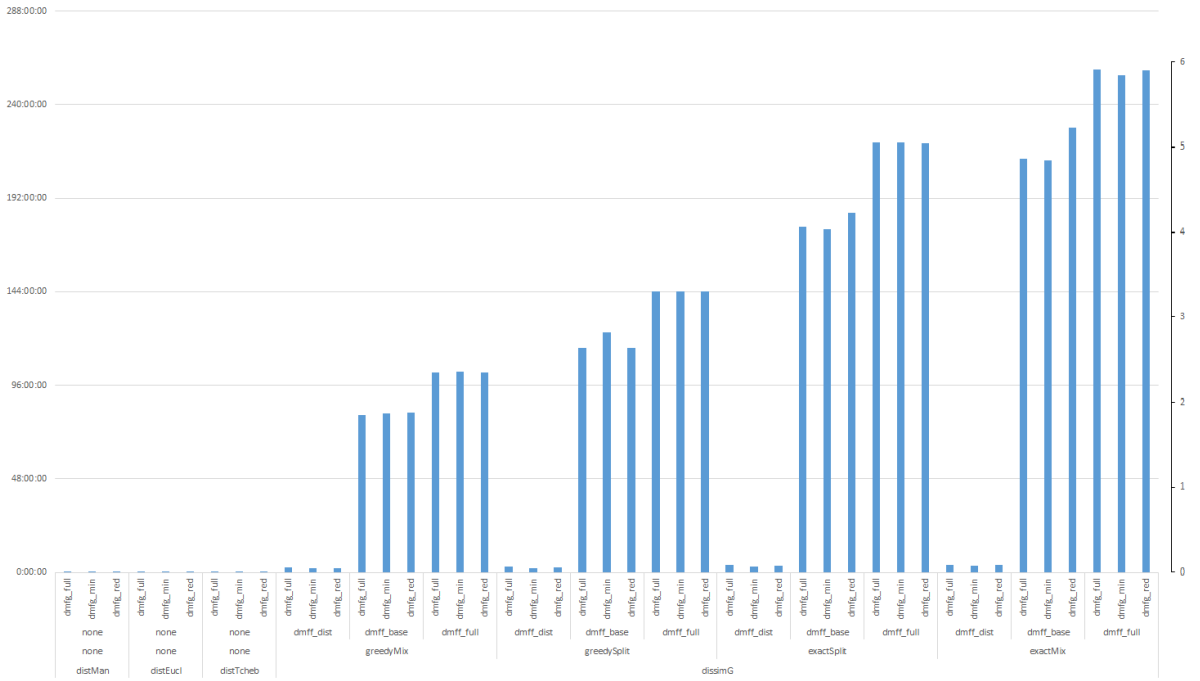

FIGURE 4.2 – Temps de calcul des dissimilarités. Total en heures à gauche, et moyenne en secondes à droite.

Ces temps d'exécution sont globalement compatibles avec les complexités attendues, 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).

4.5 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'hypothèse 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 est détaillée 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.

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 proche 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 [2] : $k \in \{3, 5, 10\}$. 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 4.3.

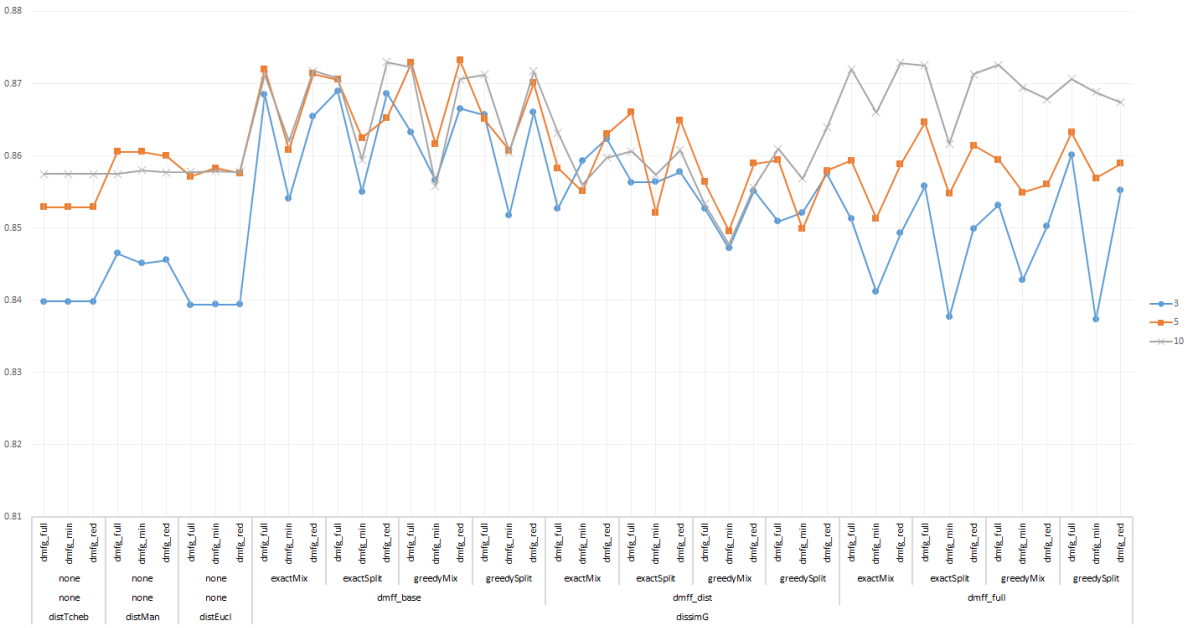

FIGURE 4.3 – Moyenne des performances au méta-niveau selon le nombre k de voisins considérés.

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 4.3, 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 4.4) 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 4.3 était de $9.2496 * 10^{-13}$ (*p-value* en Figure 4.4). 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 controllant 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 4.4, 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.

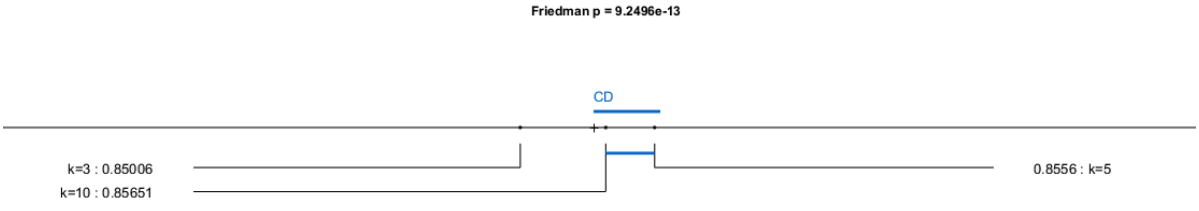

FIGURE 4.4 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes valeurs de k.

Les résultats du test de Nemenyi en Figure 4.4 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 4.5, montrent bien que $k = 10$ y est significativement meilleur que $k = 5$, lui-même significativement meilleur que $k = 3$.

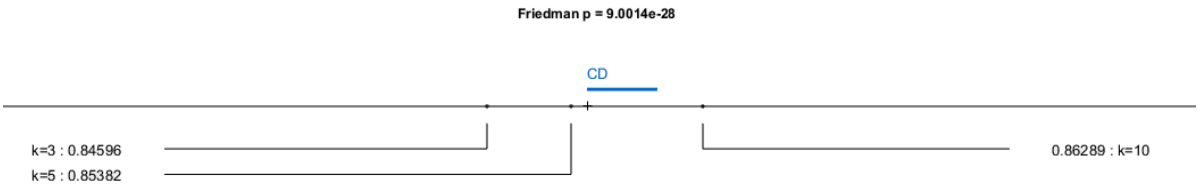

FIGURE 4.5 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes valeurs de k, pour les dissimilarités utilisant *DMFf_full*.

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 4.6 la moyenne de performance au méta-niveau des dissimilarités selon la méthode d'estimation utilisée.

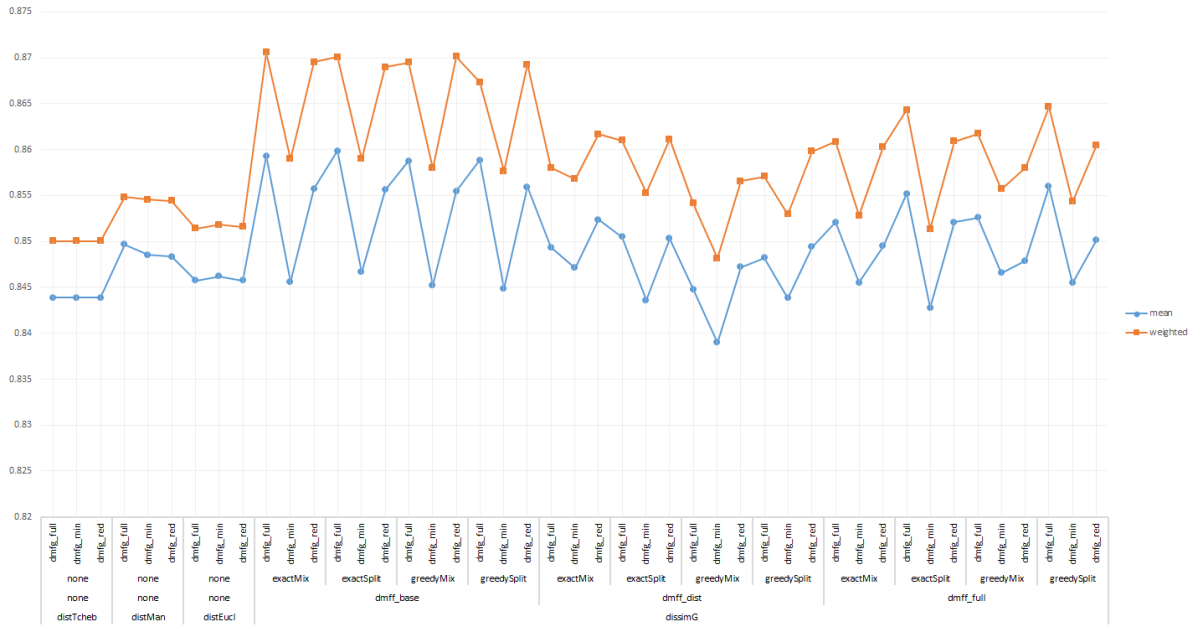

FIGURE 4.6 – Moyenne des performances au méta-niveau selon la méthode d'estimation utilisée.

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 4.7.

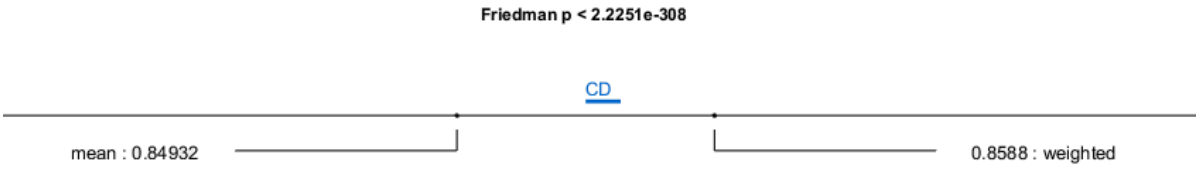

FIGURE 4.7 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes méthodes d'estimation.

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 défini 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 4.13. On présente en Figure 4.8 la moyenne de performance au méta-niveau des dissimilarités et de la *baseline* selon le critères de performance de base utilisé (la *baseline* apparait à gauche, suivie des différentes dissimilarités).

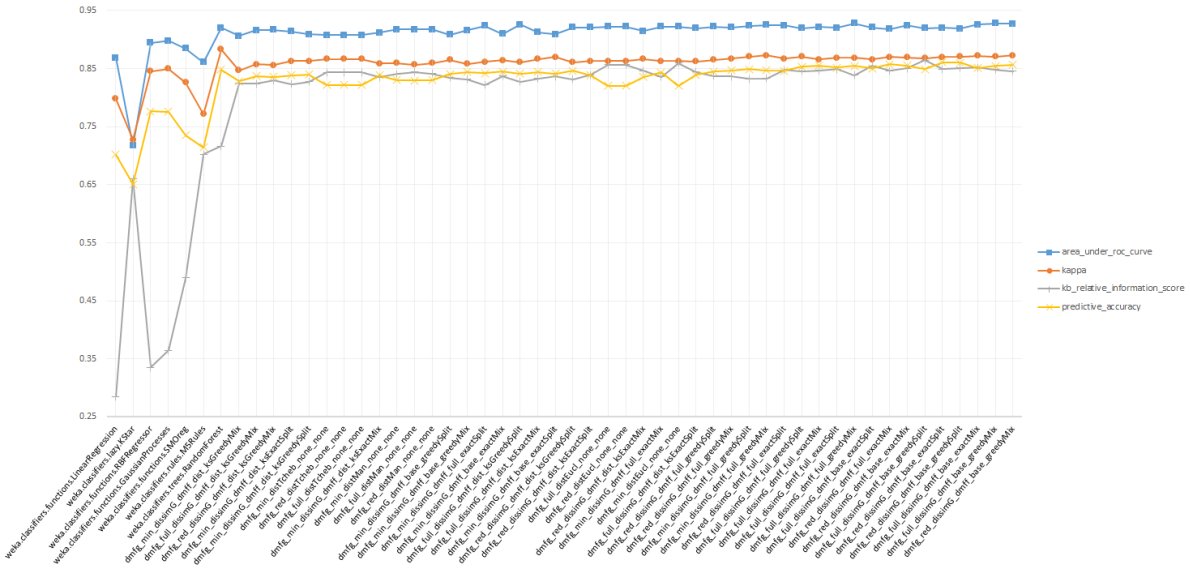

FIGURE 4.8 – Moyenne des performances au méta-niveau selon le critères de performance de base utilisé.

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 dissimilarités utilisant de simples distances. On vérifie ces tendances respectivement en Figures 4.9 et 4.10.

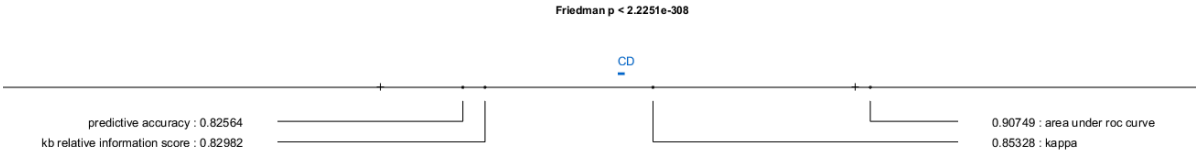

FIGURE 4.9 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents critères de performance de base.

FIGURE 4.10 – 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 4.9 et 4.10 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 4.11). On tente de valider cette tendance en Figure 4.12.

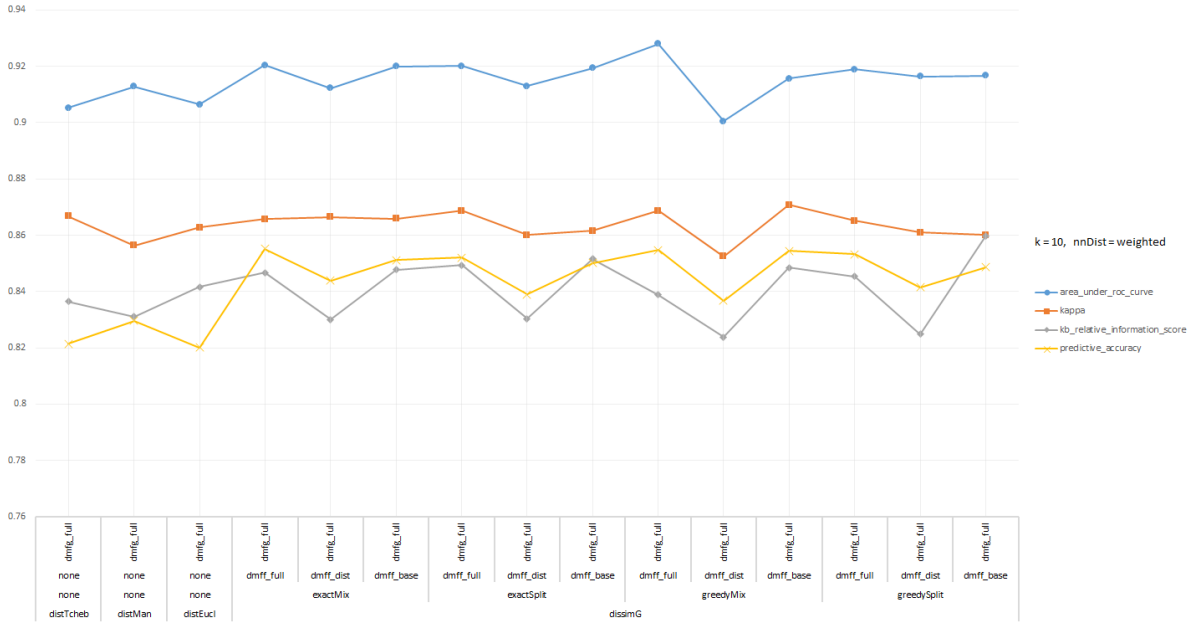

FIGURE 4.11 – Moyenne des performances au méta-niveau selon le critères de performance de base utilisé, pour les dissimilarités utilisant *DMFg_full*.

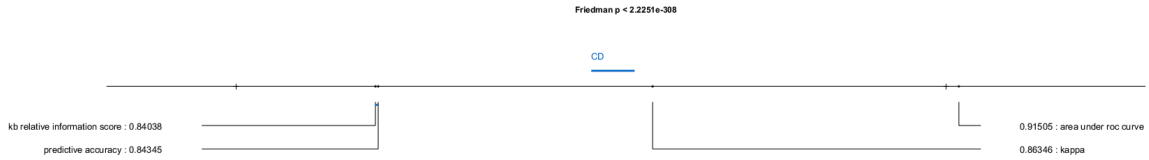

FIGURE 4.12 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents critères de performance de base, pour les dissimilarités utilisant *DMFg_full*.

Le test de Nemenyi en Figure 4.12 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'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

Comme indiqué précédemment, le nombre n de recommandations effectués 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 4.13 la moyenne de performance au méta-niveau des dissimilarités et de la *baseline* selon le nombre de recommandations effectuées.

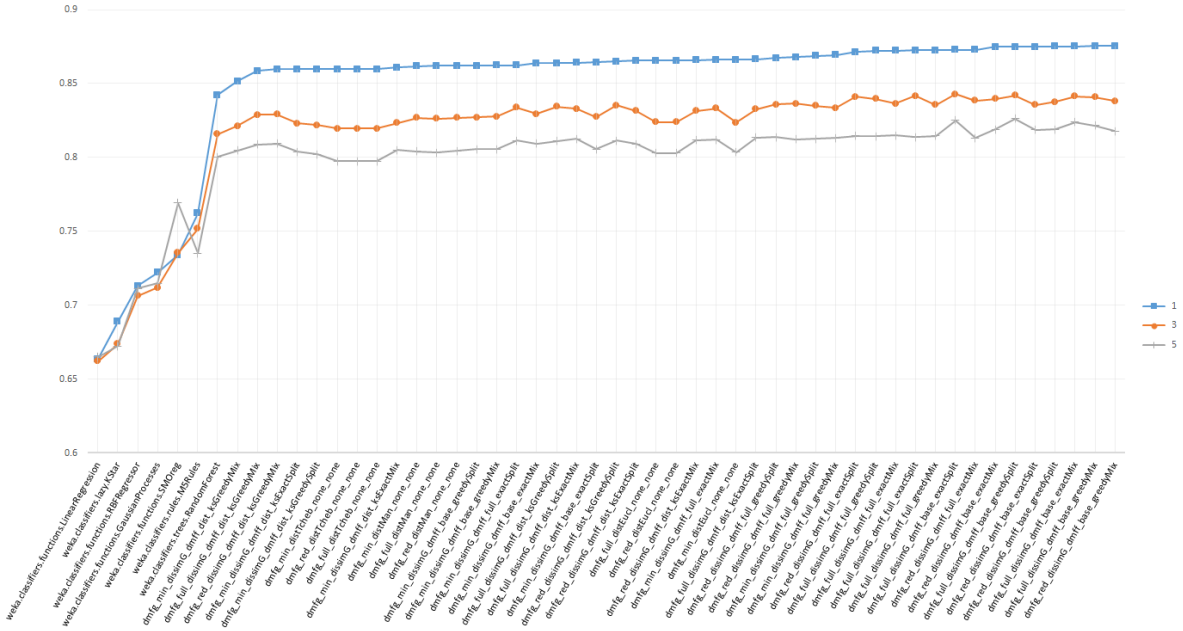

FIGURE 4.13 – Moyenne des performances au méta-niveau selon le nombre de recommandations.

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 4.14.

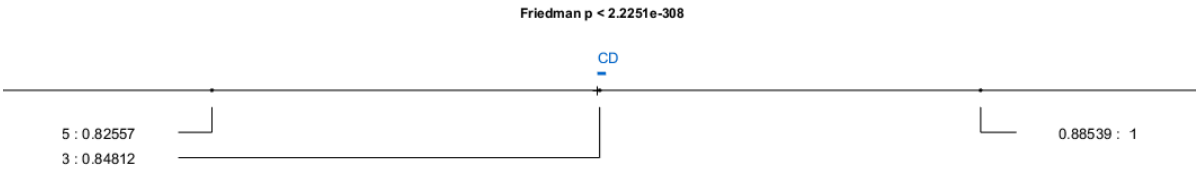

FIGURE 4.14 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes valeurs du nombre de recommandations.

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 4.15 la variance des performances au méta-niveau selon le nombre de recommandations.

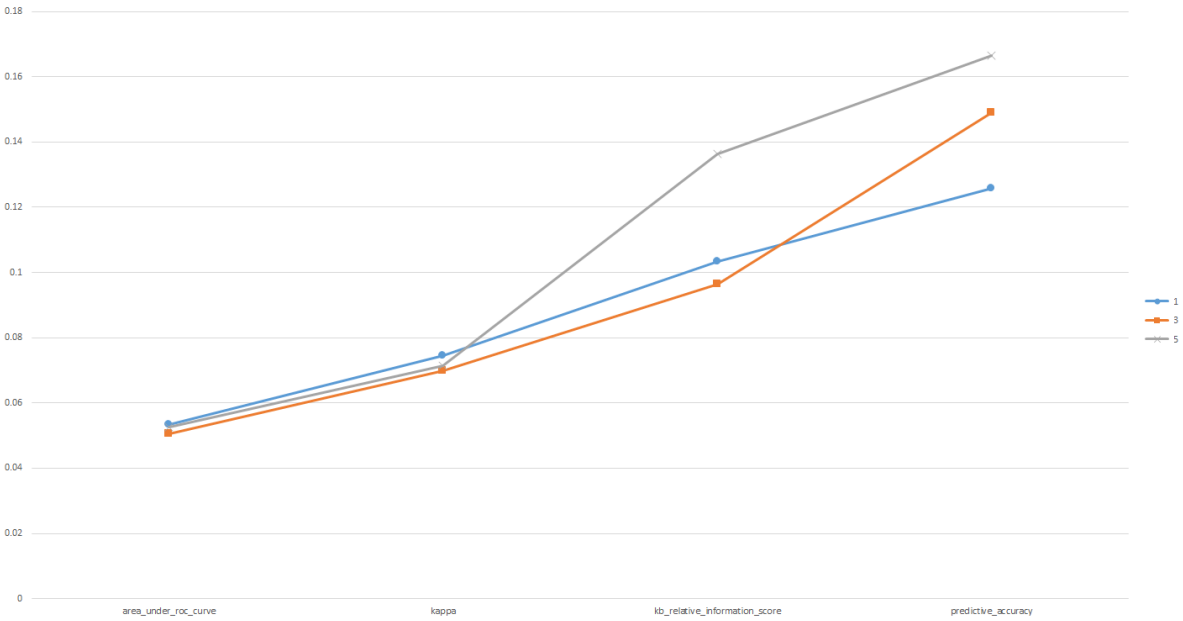

FIGURE 4.15 – Variance des performances au méta-niveau par critère selon le nombre de recommandations.

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.

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).

4.5.2 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 4.16 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 4.17.

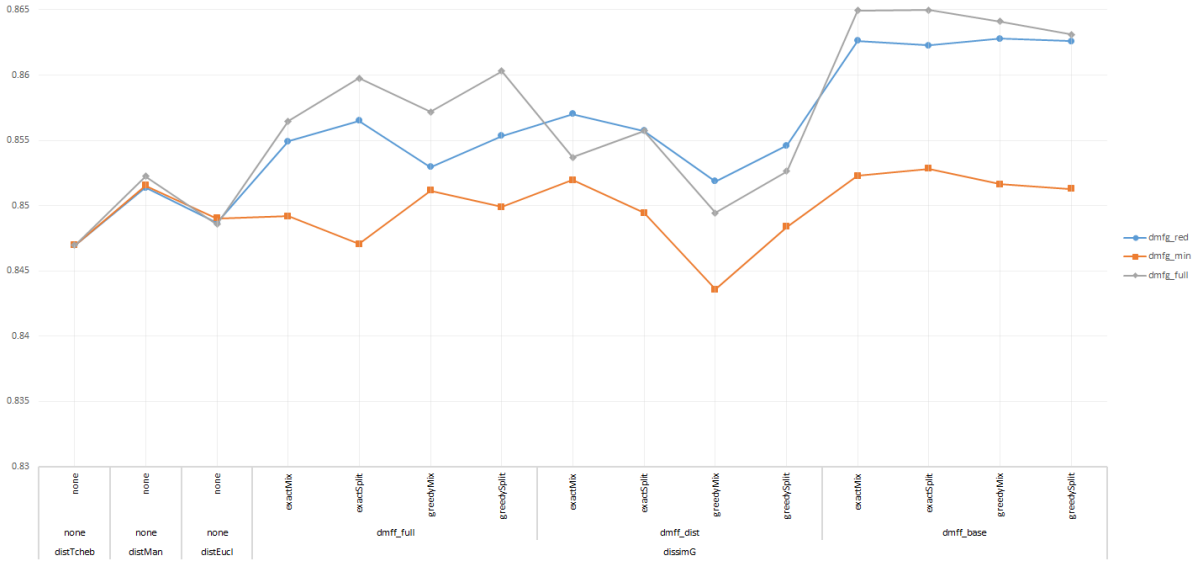

FIGURE 4.16 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs généraux utilisé.

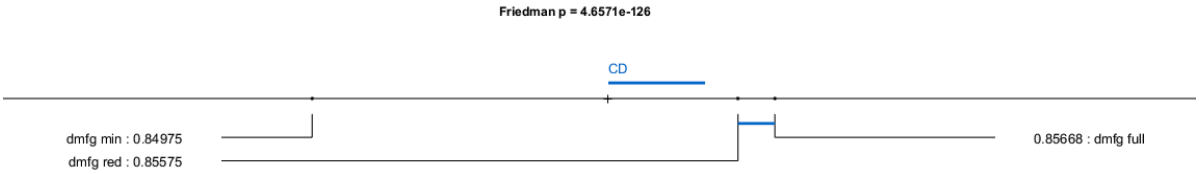

FIGURE 4.17 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensembles de méta-attributs généraux utilisés.

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 [5]. 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.

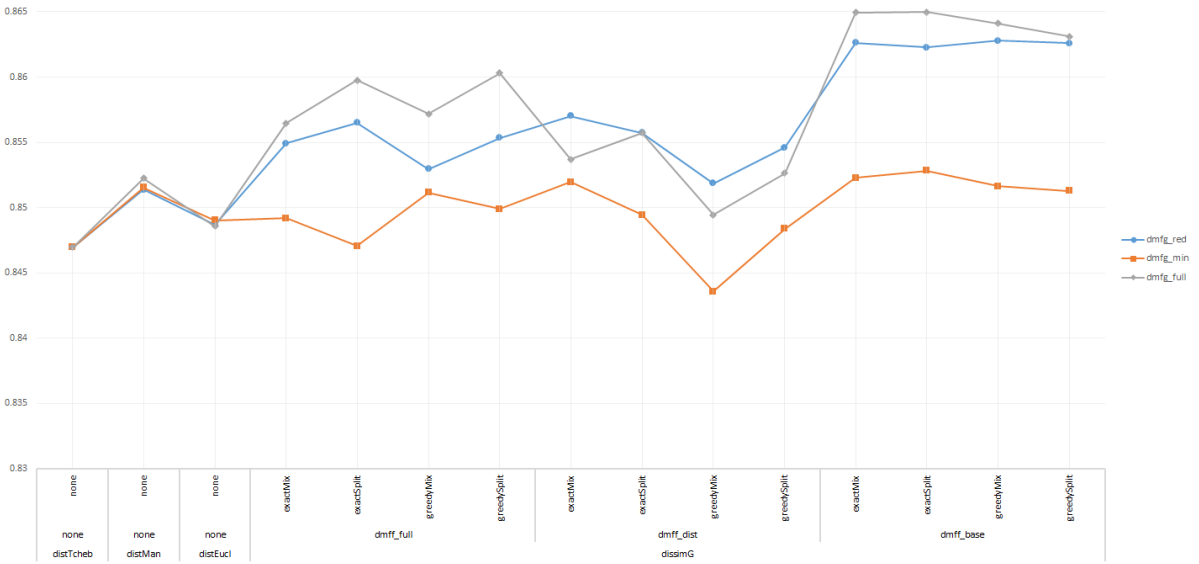

FIGURE 4.18 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs généraux utilisé, sur l'espace optimal.

Si l'on restreint l'étude à l'espace optimal (Figure 4.18), on ne décèle plus de tendance particulière. De même, le test statistique (Figure 4.19) révèle uniquement la domination de *DMFg_red* sur *DMFg_min*, insistant sur l'intérêt de la présence de *landmarkers*, nonobstant leur nombre.

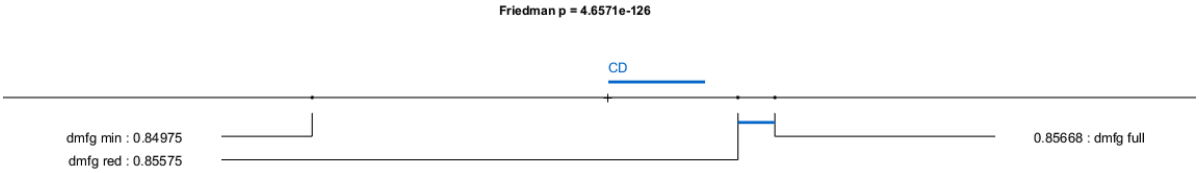

FIGURE 4.19 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensembles de méta-attributs généraux utilisés, sur l'espace optimal.

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 4.20 comment se comporte la dissimilarité normalisée par la borne supérieure par rapport aux distances classiques dans notre espace optimal.

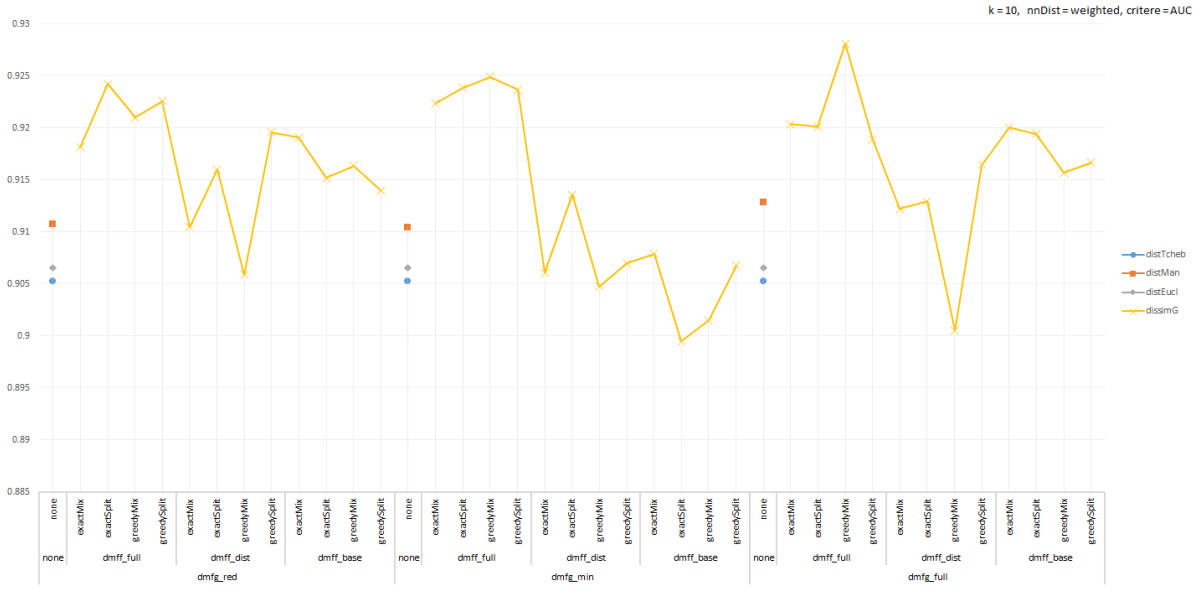

FIGURE 4.20 – Moyenne des performances au méta-niveau selon la dissimilarité sur les méta-attributs généraux.

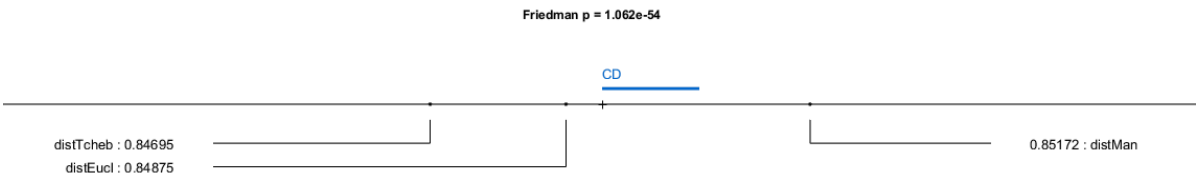

FIGURE 4.21 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarité sur les méta-attributs généraux.

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 4.20 la dominance de *dissimG* dans une majorité de cas. De plus, une comparaison entre elles des distances confirme la dominance de la distance de la distance de Manhattan sur les deux autres (voir Figure 4.21), 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 taille différente, est ici une alternative valide au test de Friedman pour comparer les distances à la dissimilarité. On présente en Table 4.17 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*).

TABLE 4.17 – Résultats des tests de Mann-Whitney comparant les distances à la dissimilarité.

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	

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.

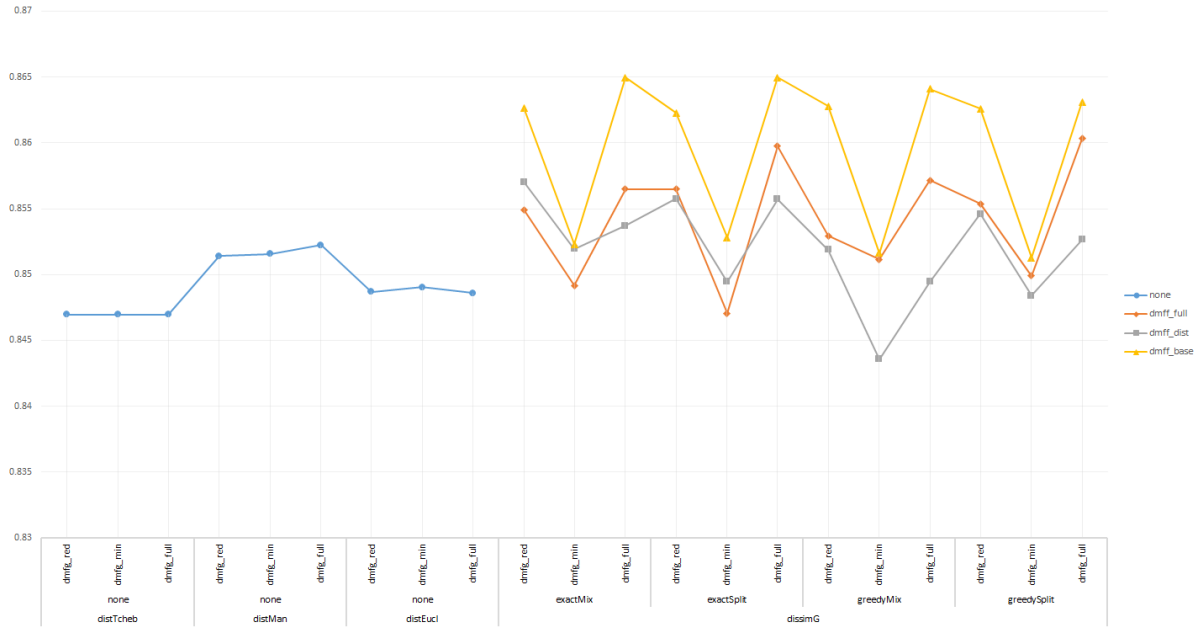

FIGURE 4.22 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs des attributs utilisé.

On présente en Figure 4.22 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 4.23.

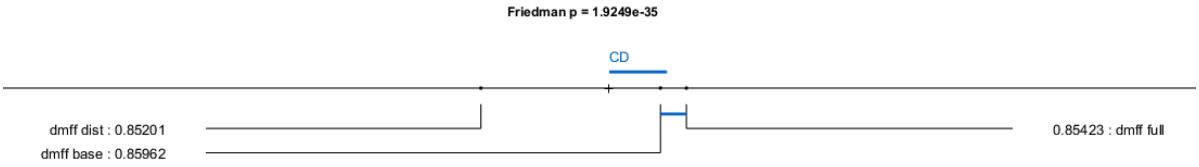

FIGURE 4.23 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensemble de méta-attributs des attributs.

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 établie 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 4.24). *DMFf_full* semble alors clairement dominer, ce que l'on va chercher à confirmer avec le test en Figure 4.25.

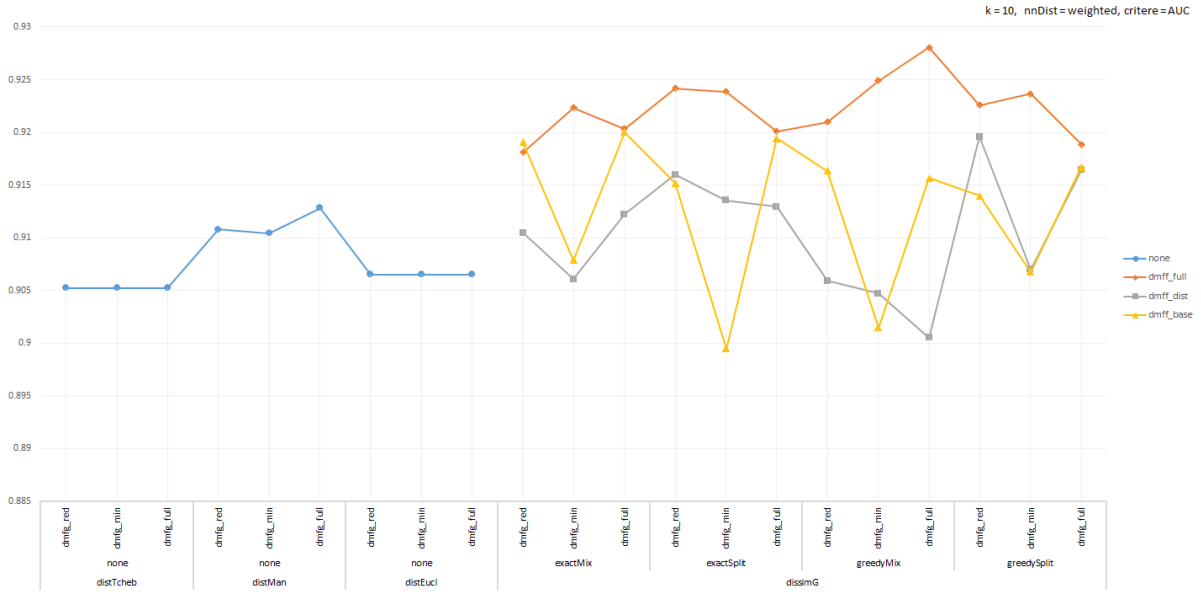

FIGURE 4.24 – Moyenne des performances au méta-niveau selon l'ensemble de méta-attributs des attributs utilisé, sur l'espace optimal.

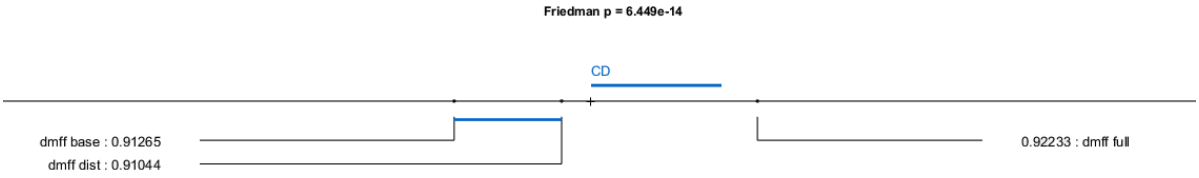

FIGURE 4.25 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents ensemble de méta-attributs des attributs, sur l'espace optimal.

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 4.18 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.

TABLE 4.18 – Résultats des tests de Mann-Whitney comparant les dissimilarités proposées à celles de la baseline.

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	

On constate bien dans tous les cas une *p-value* suffisante 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 4.26 les performances moyennes au méta-niveau pour les dissimilarités construite sur ces différents *mappings*. Une simple observation ne révèle pas de tendance forte, et seul un test statistique (Figure 4.27) révèle la dominance de l'*exactSplit* sur le *greedyMix*. Ajoutant à cela les temps d'exécution présentés en Figure 4.2, 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*.

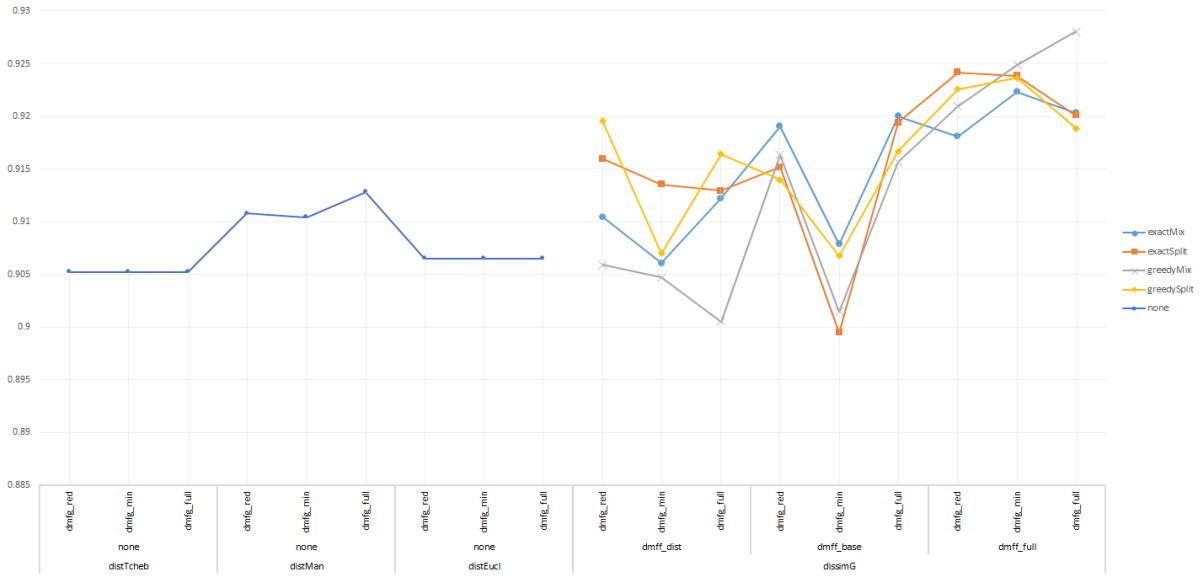

FIGURE 4.26 – Moyenne des performances au méta-niveau selon la dissimilarité sur les méta-attributs des attributs.

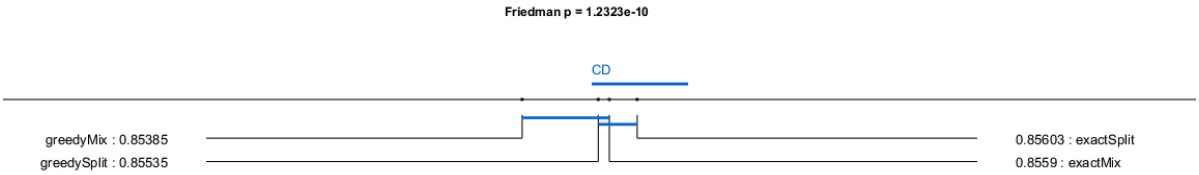

FIGURE 4.27 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarités sur les méta-attributs des attributs.

On vérifie en Figure 4.28 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"*.

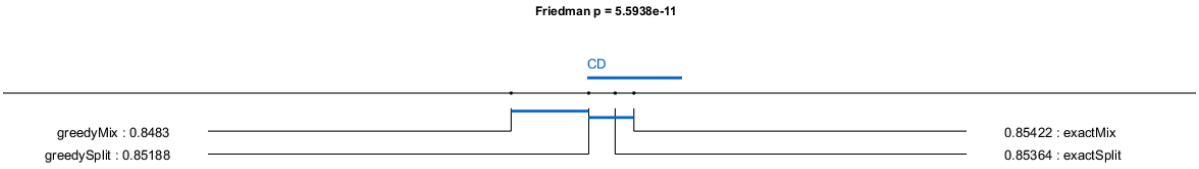

FIGURE 4.28 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarités sur les méta-attributs des attributs, et utilisant δ^{σ}_{KS}.

Le résultat précédent peut laisser penser que le choix du *mapping* a plus d'influence sur 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 4.29 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*.

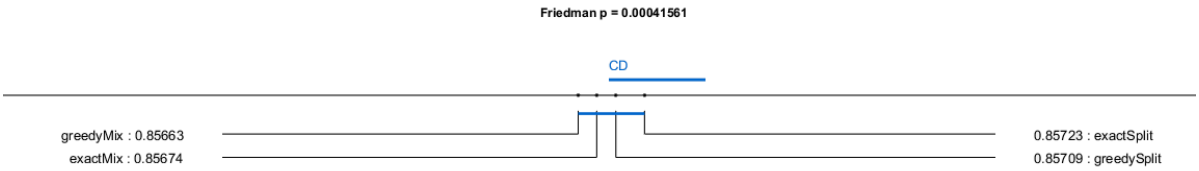

FIGURE 4.29 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différentes dissimilarités sur les méta-attributs des attributs, et n'utilisant pas δ^{σ}_{KS}.

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*.

4.5.3 Comparatif global

La comparaison à la *baseline* forme le dernier spectre d'observations à mener. On étudie ainsi en Figure 4.30 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).

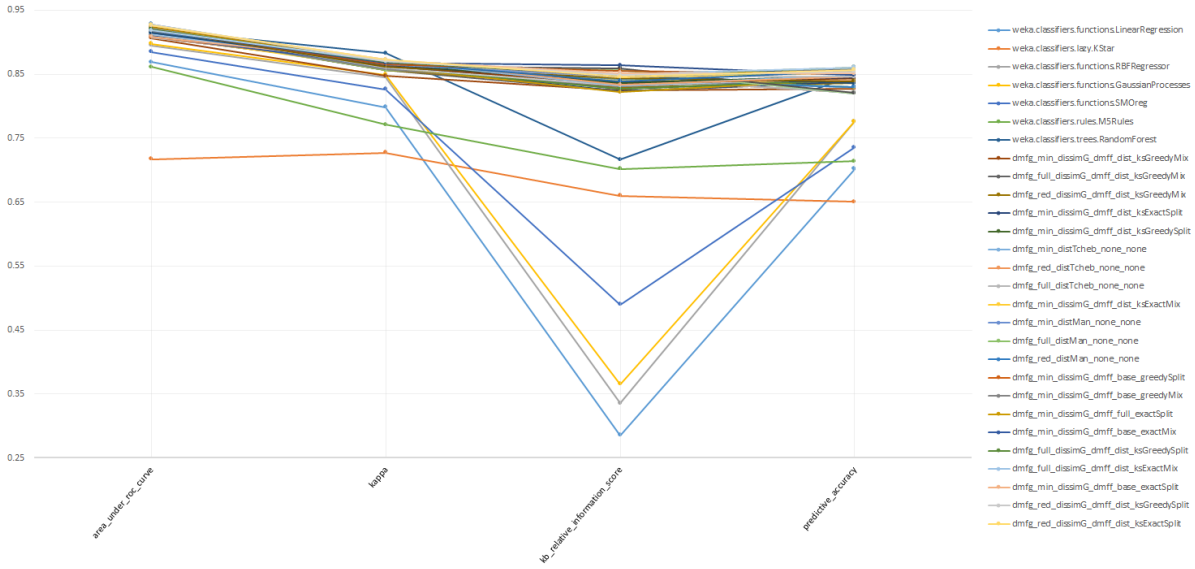

FIGURE 4.30 – Performances moyennes des différents algorithmes au méta-niveau, par critère de performance de base.

L'algorithme *KStar* (en orange, second de la liste) 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 [6]. 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 la puissance des tests et empêche de tirer des conclusions. On se limite donc en Figure 4.31 à comparer les différents algorithmes sur le critère d'aire sous la courbe, qui était notre meilleur candidat.

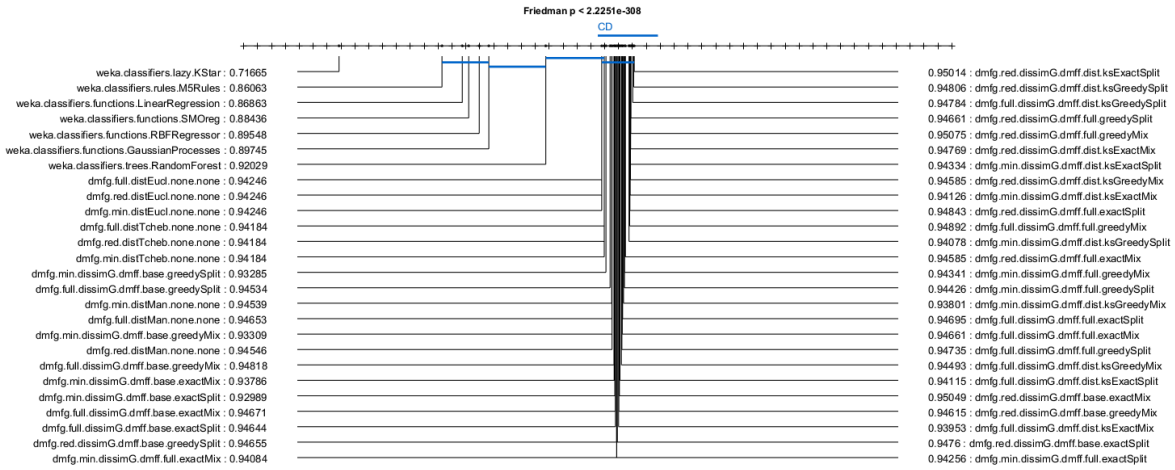

FIGURE 4.31 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents algorithmes au méta-niveau, pour le critère d'aire sous la courbe.

On valide donc bien une différence significative entre nos dissimilarités et la *baseline* d'algorithme traditionnels. Seules quelques dissimilarités basées sur des distances simples (donc de la baseline) ne peuvent être considérées significativement suppérieures au meilleur algorithme de la baseline (RandomForest [4]). 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 niveau méta (Figure 4.32) 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.

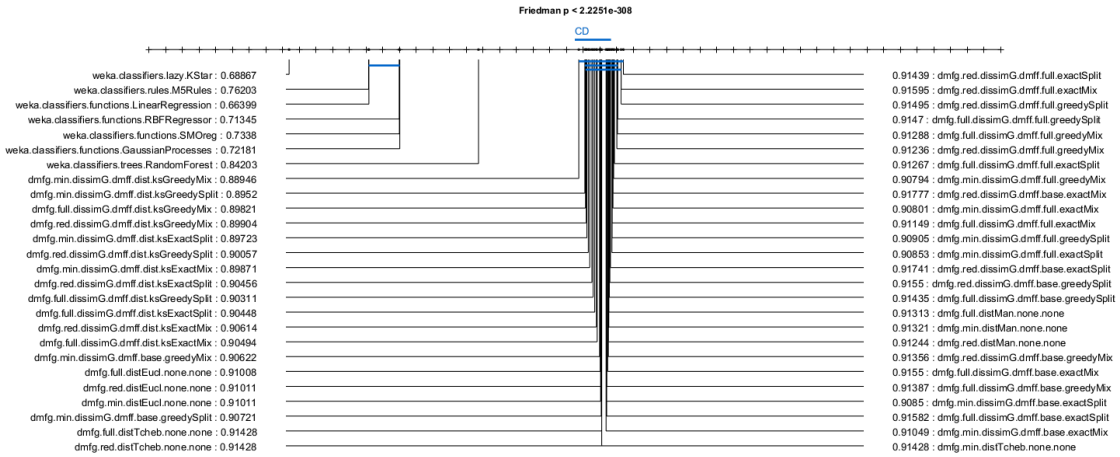

FIGURE 4.32 – Résultats du test de Friedman avec *post-hoc* de Nemenyi entre les échantillons représentant les différents algorithmes au méta-niveau.

Ces deux figures (4.31 et 4.32) nous permettent par ailleurs d'investiguer les méthodes ayant obtenu les meilleures performances moyennes au méta-niveau. Sur l'ensemble des critères (4.32), 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 !

Chapitre 5

Discussion

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 shynthétique, on utilisera les notations $\prec$ et $\preceq$ pour *"est dominé significativement"* et *"est dominé sans différence significative"* respectivement.

— 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 :

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

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'aire sous la courbe semble sensiblement plus facile à modéliser.

— Nombre de recommandations :

$$5 \prec 3 \prec 1$$

L'augmentation du nombre de recommandations ayant peu d'effet sur la variance des performances, elle présente peu d'intérêt immédiat.

On détermine ainsi un "espace optimal", ensemble de valeurs des facteurs secondaires maximisant la performance au méta-niveau : $k = 10$, nnDist=*weighted*, criterion=AUC, et $n = 1$. On étudie ensuite le propriétés des dissimilarités, soit sur cet espace optimal, soit dans le cas général.

— Méta-attributs généraux :

$$\textit{DMFg\_min} \prec \textit{DMFg\_red} \preceq \textit{DMFg\_full}$$

L'utilisation de méta-attributs de type *landmarker* améliore significativement la performance, mais la diversification des évaluations des *landmarkers* ne présente pas d'effet significatif. On retombe sur des problématiques classiques de sélection d'attributs, où l'on voudra nos méta-attributs informatifs et diversifiés.

— 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 des attributs :

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

On confirme ici l'intérêt des méta-attributs des attributs, et en particulier de leurs versions normalisées.

— Dissimilarité sur les attributs :

$$greedyMix \prec exactSplit$$

Le cout d'exécution du *mapping exactSplit* dépasse celui du *greedyMix* d'un simple facteur 2, ce qui pointe vers l'utilisation du *mapping* exact avec séparation des attributs par type.

— Comparatif global :

$$\text{Algorithmes traditionnels de la } baseline \prec \text{Dissimilarités}$$

On confirme donc l'intérêt global de l'approche par dissimilarité sur ce problème de sélection d'algorithme de classification.

Ces résultats, dans l'ensemble très positifs, démontrent l'intérêt des approches proposées pour la sélection d'algorithme 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 couteux si pratiqué *offline* : calculer la matrice complète des dissimilarités entre jeux de données peut prendre des jours 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'algorithme actifs maintenant une base de cas sur lesquels construire leurs recommandations.

5.2 Conclusion

Nous avons 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'elle permet de caractériser l'adéquation d'algorithmes de classification avec des jeux de données plus efficacement que des distances traditionnelles, et qu'elle peut être employée avec de bonnes performances dans le contexte de classification au méta-niveau.

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 [28, 5] 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 [29, 15, 27, 35], 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 nouvelle 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

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6) :716–723, December 1974.

[2] GEAPA Batista and Diego Furtado Silva. How k-nearest neighbor parameters affect its performance. In *Argentine Symposium on Artificial Intelligence*, pages 1–12, 2009.

[3] Pavel Brazdil, João Gama, and Bob Henery. Characterizing the applicability of classification algorithms using meta-level learning. In *European conference on machine learning*, pages 83–102. Springer, 1994.

[4] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[5] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation : a unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13(1) :27–66, 2012.

[6] John G Cleary, Leonard E Trigg, et al. K* : An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine learning*, volume 5, pages 108–114, 1995.

[7] Jacob Cohen. Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4) :213, 1968.

[8] Lin Dong, Eibe Frank, and Stefan Kramer. Ensembles of balanced nested dichotomies for multi-class problems. In *Knowledge Discovery in Databases : PKDD 2005*, pages 84–95. Springer, 2005.

[9] András Frank. On kuhn's hungarian method : a tribute from hungary. *Naval Research Logistics (NRL)*, 52(1) :2–5, 2005.

[10] Eibe Frank. Fully supervised training of gaussian radial basis function networks in weka. Technical report, Department of Computer Science, The University of Waikato, 2014.

[11] Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes, and Ian H Witten. Using model trees for classification. *Machine Learning*, 32(1) :63–76, 1998.

[12] J Fürnkranz and J Petrak. Extended data characteristics. Technical report, METAL consortium, 2002. Accessed 12/11/15 at `citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.302`.

[13] Christophe Giraud-Carrier, Ricardo Vilalta, and Pavel Brazdil. Introduction to the special issue on meta-learning. *Machine learning*, 54(3) :187–193, 2004.

[14] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software : an update. *ACM SIGKDD explorations newsletter*, 11(1) :10–18, 2009.

[15] Tin Kamo Ho and Mitra Basu. Complexity measures of supervised classification problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3) :289–300, 2002.

[16] Geoffrey Holmes, Mark Hall, and Eibe Prank. Generating rule sets from model trees. In *Australasian Joint Conference on Artificial Intelligence*, pages 1–12. Springer, 1999.

[17] Alexandros Kalousis. *Algorithm selection via meta-learning.* PhD thesis, Universite de Geneve, 2002.

[18] Alexandros Kalousis, João Gama, and Melanie Hilario. On data and algorithms : Understanding inductive performance. *Machine Learning*, 54(3) :275–312, 2004.

[19] Alexandros Kalousis and Maelanie Hilario. Model selection via meta-learning : a comparative study. *International Journal on Artificial Intelligence Tools*, 10(04) :525–554, 2001.

[20] Alexandros Kalousis and Melanie Hilario. Feature selection for meta-learning. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD '01, pages 222–233, London, UK, UK, 2001. Springer-Verlag.

[21] Igor Kononenko and Ivan Bratko. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6(1) :67–80, January 1991.

[22] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2) :83–97, 1955.

[23] Rui Leite, Pavel Brazdil, and Joaquin Vanschoren. Selecting classification algorithms with active testing. In *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer, 2012.

[24] Enrique Leyva, Adriana Gonzalez, and Roxana Perez. A set of complexity measures designed for applying meta-learning to instance selection. *Knowledge and Data Engineering, IEEE Transactions on*, 27(2) :354–367, 2015.

[25] David JC MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168 :133–166, 1998.

[26] Donald Michie, David J Spiegelhalter, and Charles C Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, Upper Saddle River, NJ, USA, 1994.

[27] Irene Ntoutsi, Alexandros Kalousis, and Yannis Theodoridis. A general framework for estimating similarity of datasets and decision trees : exploring semantic similarity of decision trees. In *SDM*, pages 810–821. SIAM, 2008.

[28] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8) :1226–1238, 2005.

[29] Yonghong Peng, Peter A Flach, Pavel Brazdil, and Carlos Soares. Decision tree-based data characterization for meta-learning. *IDDM-2002*, page 111, 2002.

[30] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. 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*, pages 743–750, 2000.

[31] F Serban. *Toward effective support for data mining using intelligent discovery assistance*. PhD thesis, 2013.

[32] Jakub Smid. *Computational Intelligence Methods in Metalearning*. PhD thesis, 2016.

[33] N Smirnov. Table for estimating the goodness of fit of empirical distributions. *The Annals of Mathematical Statistics*, 19(2) :279–281, 1948.

[34] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3) :199–222, 2004.

[35] Quan Sun and Bernhard Pfahringer. Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine learning*, 93(1) :141–161, 2013.

[36] Quan Sun, Bernhard Pfahringer, and Michael Mayo. Full model selection in the space of data mining operators. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1503–1504. ACM, 2012.

[37] Ljupco Todorovski, Pavel Brazdil, and Carlos Soares. 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*, pages 27–39. Citeseer, 2000.

[38] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Discovery science*, pages 16–31, 2004.

[39] Joaquin Vanschoren, Hendrik Blockeel, Bernhard Pfahringer, and Geoffrey Holmes. Experiment databases. *Machine Learning*, 87(2) :127–158, 2012.

[40] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2) :77–95, October 2002.

[41] Liwei Wang, Masashi Sugiyama, Cheng Yang, Kohei Hatano, and Jufu Feng. Theory and algorithm for learning with dissimilarity functions. *Neural computation*, 21(5) :1459–1484, 2009.

[42] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning data set similarities for hyperparameter optimization initializations. In *MetaSel@ PKDD/ECML*, pages 15–26, 2015.

[43] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7) :1341–1390, 1996.

[44] David H Wolpert and William G Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1) :67–82, 1997.

[45] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm : Simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.

[46] Monika Zakova, Petr Kremen, Filip Zelezny, and Nada Lavrac. Automating knowledge discovery workflow composition through ontology-based planning. *Automation Science and Engineering, IEEE Transactions on*, 8(2) :253–264, 2011.

Annexe A

Listings

Table A.1 – Chaines de traitement weka employés comme classifieurs dans la preuve de concept

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 la preuve de concept

anneal
anneal
kr-vs-kp
labor
arrhythmia
audiology
liver-disorders
autos
lymph
balance-scale
mfeat-factors
breast-cancer
mfeat-fourier
mfeat-karhunen
mfeat-morphological
mfeat-pixel
car
mfeat-zernike
cmc
mushroom
colic
colic
optdigits
credit-a
page-blocks
cylinder-bands
postoperative-patient-data
dermatology
segment
diabetes
sick
ecoli
sonar
glass
soybean
haberman
spambase
tae
heart-c
tic-tac-toe
heart-h
heart-statlog
vehicle
hepatitis
vote
hypothyroid
ionosphere
waveform-5000
iris
zoo
lung-cancer
molecular-biology_promoters
primary-tumor
shuttle-landing-control
solar-flare
solar-flare
yeast
satimage
abalone
baseball
braziltourism
wine
eucalyptus
meta_all.arff
meta_batchincremental.arff
meta_ensembles.arff
meta_instanceincremental.arff
flags
Australian
vowel
oil_spill
scene
thyroid_sick
yeast_ml8
hayes-roth
monks-problems-1
monks-problems-2
monks-problems-3
SPECT
SPECTF
grub-damage
pasture
squash-stored
squash-unstored
white-clover
sponge
aids
JapaneseVowels
synthetic_control
analcatdata_boxing2
prnn_crabs
analcatdata_boxing1
analcatdata_lawsuit
irish
analcatdata_broadwaymult
analcatdata_bondrate
analcatdata_halloffame
analcatdata_asbestos
analcatdata_creditscore
analcatdata_challenger
prnn_synth
analcatdata_cyyoung8092
schizo
analcatdata_japansolvent
confidence
analcatdata_dmft
profb
lupus
physionet_bradycardia
analcatdata_germangss
analcatdata_bankruptcy
fl2000
prnn_viruses
biomed
colleges_aaup
rmftsa_sleepdata
sleuth_ex2016
sleuth_ex2015
visualizing_livestock
diggle_table_a2
fruitfly
fri_c3_100_50
rmftsa_ladata
veteran
abalone
pwLinear
analcatdata_vineyard
bank8FM
fri_c2_100_5
analcatdata_supreme
visualizing_slope
fri_c1_250_5
baskball
fri_c0_250_50
machine_cpu
cpu_small
visualizing_environmental
space_ga
pharynx
rmftsa_sleepdata
fri_c4_500_100
fri_c3_250_5
auto_price
fri_c1_250_25
servo
analcatdata_wildcat
fri_c3_500_5
pm10
puma32H
wisconsin
fri_c0_100_5
sleuth_ex1605
autoPrice
meta
analcatdata_election2000
analcatdata_olympic2000
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
witmer_census_1980
triazines
fri_c1_100_10
elusage
diabetes_numeric
fri_c2_500_5
fri_c3_250_10
fri_c2_250_25
disclosure_x_tampered
cpu
cholesterol
pyrim
chscase_funds
delta_ailerons
hutsof99_logis
fri_c4_500_50
kin8nm
fri_c0_100_10
mushroom
pbc
rmftsa_ctoarrivals
fri_c1_100_25
chscase_vine2
chscase_vine1
puma8NH
diggle_table_a1
diggle_table_a2
delta_elevators
chatfield_4
fri_c1_500_10
boston_corrected
sensory
disclosure_x_noise
fri_c4_100_100
fri_c1_100_5
fri_c2_250_10
autoMpg
physionet_tachycardia
fri_c3_250_25
bank32nh
fri_c4_250_100
analcatdata_vehicle
sleuth_case1102
fri_c4_500_25
kdd_el_nino-small
autoHorse
stock
analcatdata_runshoes
breastTumor
wind
schlvote
fri_c0_100_50
tecator
analcatdata_gsssexsurvey
housing
fishcatch
fri_c4_500_10
bolts
hungarian
analcatdata_gviolence
vinnie
auto93
sleuth_ex2016
fri_c4_250_10
sleuth_ex2015
analcatdata_neavote
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
pollution
fri_c0_500_5
transplant
no2
mbagrade
fri_c0_500_50
fri_c0_100_25
cloud
sleuth_case1202
sleuth_case1201
visualizing_hamster
rabe_148
chscase_geyser1
fri_c3_500_25
colleges_aaup
analcatdata_negotiation
chscase_census6
sleuth_case2002
chscase_adopt
chscase_census5
chscase_census4
chscase_census3
chscase_census2
fri_c2_250_5
plasma_retinol
fri_c3_100_5
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
sleuth_ex1221
water-treatment
lowbwt
chscase_health
fri_c0_500_10
echoMonths
kidney
visualizing_ethanol
arsenic-male-bladder
quake
arsenic-female-bladder
arsenic-female-lung
arsenic-male-lung
prnn_fglass
spectrometer
tae
molecular-biology_promoters
braziltourism
segment
postoperative-patient-data
analcatdata_broadwaymult
mfeat-morphological
heart-h
pasture
zoo
analcatdata_halloffame
analcatdata_birthday
iris
analcatdata_authorship
mfeat-fourier
squash-stored
wine
hayes-roth
autos
kdd_JapaneseVowels
mfeat-factors
waveform-5000
optdigits
heart-c
cmc
squash-unstored
analcatdata_marketing
collins
fl2000
anneal
eucalyptus
car
analcatdata_broadway
kdd_ipums_la_97-small
vehicle
mfeat-zernike
prnn_fglass
balance-scale
analcatdata_bondrate
audiology
hypothyroid
sponge
kdd_ipums_la_98-small
primary-tumor
kdd_synthetic_control
glass
lymph
bridges
analcatdata_reviewer
white-clover
dermatology
ecoli
flags
analcatdata_challenger
analcatdata_dmft
confidence
vowel
arrhythmia
kdd_ipums_la_99-small
mfeat-karhunen
page-blocks
mfeat-pixel
soybean
analcatdata_germangss
grub-damage
ada_prior
gina_agnostic
gina_prior2
gina_prior
ada_agnostic
kc1-top5
usp05
pc4
pc3
mc2
cm1_req
mc1
ar1
ar3
ar4
ar5
kc2
ar6
kc3
kc1-binary
kc1
pc1
pc2
mw1
jEdit_4.0_4.2
desharnais
datatrieve
PopularKids
teachingAssistant
AP_Omentum_Prostate
AP_Breast_Omentum
AP_Endometrium_Colon
AP_Colon_Kidney
AP_Endometrium_Prostate
AP_Breast_Colon
AP_Omentum_Kidney
AP_Ovary_Kidney
AP_Endometrium_Omentum
AP_Prostate_Lung
AP_Endometrium_Ovary
OVA_Colon
AP_Ovary_Uterus
AP_Lung_Kidney
AP_Endometrium_Uterus
AP_Breast_Ovary
OVA_Ovary
pc1_req
Internet-Advertisements
Click_prediction_small
Click_prediction_small
Click_prediction_small
Click_prediction_small
eating
physionet_asystole

TABLE A.3 – Méta-attributs OpenML utilisés dans la preuve de concept

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 la preuve de concept

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 la preuve de concept

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 – Critères de performance de classification utilisés dans la preuve de concept

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 A.7 – Chaines de traitement weka employés comme classifieurs dans les expériences comparatives

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.8 – Jeux de données OpenML utilisés dans les expériences comparatives

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