ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N° d'ordre : 3287

THESE

En vue de l'obtention du : ***DOCTORAT***

Structure de Recherche : ***Laboratoire de Recherche en Informatique et Télécommunications***
Discipline : ***Sciences de l'ingénieur***
Spécialité : ***Informatique et Télécommunications***

Présentée et soutenue le 31/01/2020 par :

Btissam ZERHARI

Contributions au prétraitement de données : Élimination du bruit de classe et réduction de dimension

JURY

Salma MOULINE	PES, FSR- Université Mohammed V de Rabat, Maroc	Présidente et directrice de thèse
Abderrahim EL QADI	PES, EST-Université Mohammed V de Rabat, Maroc	Rapporteur/Examinateur
Dounia LOTFI	PH, FSR-Université Mohammed V de Rabat, Maroc	Rapporteur/Examinateur
Ilham OUMAIRA	PH, ENSA-Université Ibn Tofail de Kenitra, Maroc	Rapporteur/Examinateur
Said OUATIK EL ALAOUI	PES, ENSA-Université Ibn Tofail de Kenitra, Maroc	Examinateur
Said AMALI	PH, FSJES-Université Moulay Ismail de Meknès, Maroc	Examinateur
Ayoub AIT LAHCEN	PH, ENSA-Université Ibn Tofail de Kenitra, Maroc	Co-directeur de thèse

Année Universitaire : 2019-2020

Avant-Propos

Les travaux présentés dans ce mémoire sont effectués au Laboratoire de Recherche en Informatique et Télécommunications (LRIT), à la Faculté des Sciences de Rabat, sous la direction du Professeur Salma MOULINE et l'encadrement du Professeur Ayoub AIT LAHCEN.

Je commence par présenter ma plus vive gratitude à ma directrice de thèse Pr. MOULINE Salma, Professeur de l'enseignement supérieur à la Faculté des Sciences de Rabat. Pour la confiance qu'elle m'a accordée en acceptant de diriger ce travail, pour ces qualités humaines d'écoute et de compréhension. J'exprime ici ma profonde gratitude à son égard et l'estime respectueuse que je lui porte. Je la remercie aussi pour m'avoir fait l'honneur d'accepter de présider le jury.

Je tiens à exprimer mes remerciements à mon encadrant, Pr. AIT LAHCEN Ayoub, professeur habilité à l'Université Ibn Tofail de Kénitra, pour sa disponibilité, ainsi que pour ses précieux conseils scientifiques.

Je tiens également à exprimer mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse. Merci à Pr. El QADI Abderrahim, professeur d'enseignement supérieur à l'école supérieure de technologie de Salé (EST) d'avoir accepté de rapporter mon travail et qui m'a fait l'honneur d'être parmi les membres du Jury.

Je remercie aussi Pr. LOTFI Dounia, professeur habilité à la Faculté des Sciences de Rabat, d'être un rapporteur et examinateur de cette thèse. Je le remercie pour le temps qu'elle a consacré pour lire et évaluer mon travail.

Je voudrais également remercier Pr. OUMAIRA Ilham, professeur habilité à l'école nationale des sciences appliquées de Kenitra, pour avoir accepté de rapporter et d'examiner ce travail.

Je souhaite remercier profondément Pr. OUATIK EL ALAOUI Said, professeur de l'enseignement supérieur à l'école nationale des sciences appliquées de Kenitra (ENSA), d'avoir accepté d'examiner ce travail et pour le temps qu'il a consacré pour lire et évaluer mon travail.

Je remercie aussi Pr. AMALI Said, professeur habilité à la Faculté des Sciences Juridiques, Économiques et Sociales de l'Université Moulay Ismail de Meknès, pour avoir accepté d'examiner ce travail et de se déplacer pour participer au jury.

A l'issue de la rédaction de cette recherche, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur m'ont permis de progresser dans mes recherches. Je souligne le soutien amical et chaleureux de tous les doctorants du Laboratoire LRIT, qui ont croisé ma route durant ce parcours doctoral et, plus particulièrement, à tous ceux qui m'ont soutenu et encouragé. Un merci spécial à mes chers amis Khadija el Gajoui, Majda el Houzmari et Anas Abouyahya pour les moments magnifiques et inoubliables que nous avons passé ensemble, pour leur sympathie, leur bonne humeur permanente, leur énergie positive.

Du fond du cœur je remercie infiniment mon oncle AbdelKader pour ses précieux conseils il n'a jamais cessé de me soutenir et de m'apporter de l'aide. Je tiens à remercier aussi mon oncle Said pour tous ses remarques et conseils dont il m'a fait part, je lui remercie infiniment.

Finalement, mais pas pour autant moins important à mes yeux, je voudrais témoigner tout mon amour et ma reconnaissance à ma chère famille, à qui je dédie cette thèse. Je tiens à remercier chaudement ma mère, mon père et mon cher frère Yassine et ma chère sœur Sanaa, pour leurs encouragements et leur assistance aussi bien matérielle que morale qui m'ont permis de faire cette thèse dans de bonnes conditions, je ne saurais jamais être reconnaissante pour toute leur confiance en moi, leur affection, leurs conseils, mais aussi leur soutien indéfectible. Merci à vous, même si je ne pourrai jamais vous remercier assez.

Résumé

Les données dans le monde réel ont fortement augmenté et elles sont de plus en plus sensibles aux bruits, aux valeurs manquantes et aux incohérences. La qualité des données affecte les résultats de leur exploration. Afin d'aider à améliorer la qualité des données et faciliter le processus d'extraction d'informations, ces données doivent être prétraitées. Le prétraitement des données est l'une des étapes les plus critiques dans un processus d'exploration de données.

Notre attention s'est par conséquent portée sur deux importantes phases de prétraitement : le nettoyage des données en éliminant le bruit de classe et la réduction de la dimension en utilisant une méthode de sélection d'attributs.

Deux contributions ont été proposées pour la détection et l'élimination de bruit de classe dans cette thèse. La première est une architecture distribuée permettant la détection et l'élimination de bruit de classe. La deuxième est une approche basée sur un filtre nommé Multi-Iterative Partitioning Class Noise Filter (MIPCNF) permettant la détection de bruit de classe de manière itérative. En effet, ce filtre repose sur un algorithme dans lequel plusieurs tours de détection de bruit sont effectuées pour chaque sous ensemble de données en utilisant un ensemble de classifieurs.

Une troisième contribution pour la réduction de dimension a été proposée. Il s'agit d'un nouveau Framework, appelée Horizo-Vertical Distributed Feature Selection Approach (HVDFS) , permettant la sélection d'attributs de manière distribuée horizontalement et verticalement. Cela permet d'améliorer la qualité de la classification tout en réduisant le nombre d'attributs.

Mots-clés : *Data Mining ; Prétraitement des données ; Classification ; Nettoyage des données ; Élimination de bruit de classe ; Réduction de dimension ; Sélection d'attributs*

ABSTRACT

Real-world data is greatly increased and is becoming more sensitive to noise, missing values, and inconsistencies. The quality of the data affects the results of their exploration. In order to improve the quality of the data and facilitate the information extraction process, these data must be pre-processed. Data preprocessing is one of the most critical steps in the Data Mining process.

Our attention was therefore focused on two important pretreatment steps : data cleaning by eliminating class noise, as well as the dimensionality reduction using feature selection method.

Two contributions have been proposed for the class noise detection and elimination process in this thesis. The first is a distributed architecture for detecting and eliminating class noise. The second is a new filter named MIPCNF that allows detection class noise iteratively. Indeed, our approach relies on an algorithm in which several iterations of class noise detection are performed for each subset using a set of classifiers.

A third contribution for dimensionality reduction has been proposed. This is a new Framework, called HVDFS, allowing the selection of attributes in a distributed way horizontally and vertically. This improves the classification quality while reducing the number of the attributes.

Keywords : *Data Mining ; Data preprocessing ; Classification ; Data cleansing ; Class noise elimination ; Dimensionality reduction ; Feature selection*

TABLE DES MATIÈRES

Liste des notations et abréviations

CRISP-DM Cross-Industry Standard Process for Data Mining 6

ACC Average Acceleration Capacity 8

ROC Receiver Operating Characteristic 8

ARI Adjusted Rand Index 9

NB Naive Bayes 61

KNN K – Nearest Neighbor 62

SVM Support Vector Machines 60

LR Logistique Regression 60

RF RandomForest 61

Bagging Bootstrap Aggregating 14

CBCD Cluster-Based Concurrent Decomposition 19

NCAR Noisy Completely At Random 24

NAR Noisy At Random 24

NNAR Noisy Not At Random 24

CF Classification Filtre 27

EF Ensemble Filtre 28

IPF Iterative-Partitioning Filtre 28

INFFC Iterative Noise Filter based on the Fusion of Classifiers 28

ME Multiedit 28

ENN Edited Nearest Neighbor 28

NCNE Nearest Centroid Neighbor Edition 28

AllKNN All k-Nearest Neighbors 29

AG Algorithm Genetic 41

CFS Correlation-based Feature Selection 88

FCBF Fast Correlation-Based Filter 88

MRMR Minimum Redundancy Maximum Relevance 89

Liste des figures

Liste des tableaux

Liste des algorithmes

INTRODUCTION GÉNÉRALE

Contexte générale et motivation

La rapide augmentation des capacités de stockage conjuguée à la multiplication des bases de données à divers niveaux (administration, banques, sites internet ...) ont conduit à la constitution de très grands ensembles de données. Comment rendre les données exploitables efficacement ? Comment faire face au problème de la dimensionnalité des données ? Notre attention s'est par conséquent portée sur la réduction de la dimension des ensembles de données (la réduction du nombre d'attributs), ainsi que le nettoyage de ces bases de données. La motivation derrière le choix de ces deux prétraitements est présentée comme suite :

Pour le nettoyage des données : La saisie et l'acquisition de données sont par nature sujettes aux erreurs. De nombreux efforts peuvent être déployés sur ce processus initial en ce qui concerne la réduction des erreurs d'entrée. Le problème d'apprentissage dans des environnements bruyants a beaucoup retenu l'attention en apprentissage automatique et la plupart des algorithmes d'apprentissage inductif ont un mécanisme de traitement du bruit. Cependant, comme les classifieurs appris à partir de données bruitées donc ils vont y avoir une précision moindre. Ce qui peut avoir un effet très limité sur l'amélioration des performances du système, particulièrement dans les cas où le niveau de bruit est relativement élevé. En conséquence, pour les ensembles de données existants, une solution logique pour améliorer leur qualité consiste à tenter de nettoyer les données d'une manière ou d'une autre. En d'autres termes, explorer le jeu de données pour rechercher les problèmes éventuels et essayer de corriger les erreurs ou bien de les éliminer carrément. Pour un ensemble de données du monde réel, effectuer cette tâche «à la main» est totalement hors de question, étant donné le nombre d'heures-personnes impliquées. Certaines organisations dépensent des millions de dollars par an pour détecter les erreurs de données. Un processus manuel de nettoyage des données est également laborieux, prend du temps et c'est une source d'erreurs. Des outils utiles et puissants qui automatisent ou aident grandement le processus de nettoyage des données sont nécessaires et peuvent

constituer le seul moyen pratique et économique d'atteindre un niveau de qualité raisonnable dans un jeu de données existant.

Pour la réduction de dimension : La taille des données peut être mesurée selon deux dimensions, le nombre de variables et le nombre d'exemples (Zhu et Xindong, 2004). Ces deux dimensions peuvent prendre des valeurs très élevées, ce qui peut poser un problème lors de l'exploration et l'analyse de ces données. Pour cela, il est fondamental de mettre en place des outils de traitement de données permettant une meilleure compréhension de la valeur des connaissances disponibles dans ces données. La réduction des dimensions est l'une des plus vieilles approches permettant d'apporter des éléments de réponse à ce problème. Son objectif est de sélectionner ou d'extraire un sous-ensemble optimal de caractéristiques pertinentes pour un critère fixé auparavant. La sélection de ce sous-ensemble de caractéristiques permet d'éliminer les informations non-pertinentes et redondantes selon le critère utilisé. Cette sélection/extraction permet donc de réduire la dimension de l'espace des exemples et de rendre l'ensemble des données plus représentatif du problème. La réduction de la dimension est un problème complexe qui permet de réduire le volume d'informations à traiter et de faciliter le processus de l'apprentissage. Nous pouvons classer toutes les techniques mathématiques de réduction de dimension en deux grandes catégories : la sélection des variables et l'extraction des variables (Wei et Stephen, 2007). La sélection de variables (feature selection) a fait l'objet de nos travaux présentés dans cette thèse.

Organisation de la thèse

Ce document est composé de six chapitres, organisés de la façon suivante :

— Le chapitre 1 présente le cadre général de nos travaux de recherche. Il se place plus globalement dans le domaine de la fouille des données et plus précisément dans l'étape de prétraitement de données. Il définit aussi le contexte générale lié à notre domaine de travail dans cette thèse, ainsi que les définitions et un état d'art sur ces concepts. les deux chapitres qui suivent présentent un état d'art correspondant aux thématiques principales de la thèse.

— Le chapitre 2 présente un état d'art des travaux existants dans les domaines de nettoyage des données ainsi que les enjeux majeurs y existants. Une vision général concernant les approches proposées dans la littérature pour la détection et la lutte contre le bruit de classe.

— Le chapitre 3 est consacré à la phase de réduction de dimension. Cette partie met l'accent plus précisément sur la réduction de dimension en utilisant la sélection

d'attributs, en citant les différents algorithmes et approches proposées dans la littérature pour la réduction de dimension.

— Le chapitre 4 est consacré à la présentation détaillée de deux premières contributions de cette thèse. Celles-ci concernent la détection et l'élimination de bruit de classe à partir de grosse bases de données, une première architecture pour la détection de bruit de classe, puis une autre proposition d'une nouvelle architecture (amélioration de la première) permettant la détection et l'élimination de bruit de classe de manière itératif en utilisant un ensemble de classifieurs. Il décrit aussi le cadre de nos expérimentations suivi par l'analyse des résultats obtenus.

— Chapitre 5 décrit notre approche de réduction de dimension. Présente en détail notre approche distribuée horizontalement et verticalement pour la sélection d'attributs, y compris la partie expérimentale et l'analyse des résultats.

— Chapitre 6 présente une conclusion générale de tous les travaux présentés dans cette thèse. Nous décrivons aussi quelques pistes pouvant inspirer des futurs travaux dans la continuation de ceux présentés ici.

Première partie

État de l'art

CHAPITRE

1 PRÉTRAITEMENT ET EXPLOITATION DES DONNÉES

Sommaire

1.1 Introduction

Généralement, l'exploration de données peut être expliquée comme la science de l'analyse de données pour extraire des informations utiles à l'aide d'outils et de méthodes statistiques. L'immense croissance technologique a rendu la collecte de données moins chère et efficace. Parallèlement, une augmentation exponentielle de la puissance de calcul a rendu le traitement des données extrêmement rapide et économique. Ces avancées ont accéléré les motivations de la recherche en Data Mining. La classification des données, la sélection des caractéristiques et la détection des valeurs aberrantes sont les principaux domaines de recherche dans le domaine de l'exploration de données.

1.2 L'exploration des données (Data Mining)

Data Mining signifie l'extraction de connaissance à travers l'analyse d'une grande quantité de données pour utiliser ces connaissances dans le processus de décision. Le processus de data mining implique plusieurs étapes avant de trouver un modèle de dé-

cision qui peut être un ensemble de règles, des équations ou des fonctions de transfert complexes, selon leur objectif.

Il est très important de comprendre que le Data Mining n'est pas seulement le problème de découverte de modèles dans un ensemble de données. Ce n'est qu'une seule étape dans tout un processus suivi par les scientifiques, les ingénieurs ou toute autre personne qui cherche à extraire les connaissances à partir des données (Wirth et Jochen, 2000). En 1996 un groupe d'analystes définit le Data Mining comme étant un processus composé de cinq étapes sous le standard Cross-Industry Standard Process for Data Mining (CRISP-DM) (Adriaans et Zantinge, 1996). Le processus de Data Mining selon

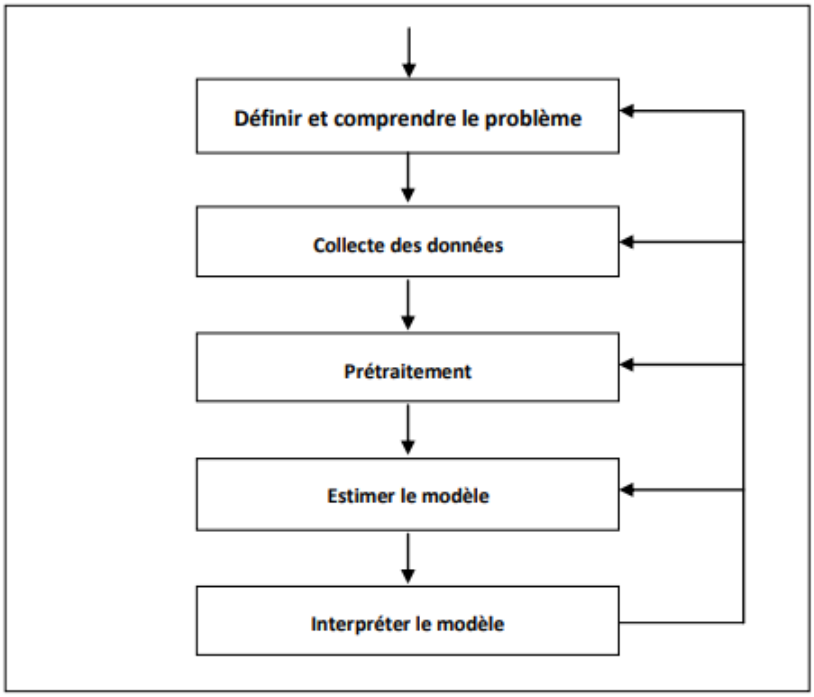

Figure 1.1 – Le standard CRISP-DM.

le standard CRISP-DM est illustré en figure (1.1) :

— *Définition et compréhension du problème* : Dans la plus part des cas, il est indispensable de comprendre la signification des données et le domaine à explorer. Sans cette compréhension, aucun algorithme ne va donner un résultat fiable (Wirth et Jochen, 2000). En effet, Avec la compréhension du problème, on peut préparer les données nécessaires à l'exploration et interpréter correctement les résultats obtenus.

— *Collecte des données* : D'après la définition du problème et des objectifs du data mining, on peut avoir une idée sur les données qui doivent être utilisées (Wirth et Jochen, 2000). Ces données n'ont pas toujours le même format et la même

structure. On peut avoir des textes, des bases de données, des pages web, ...etc. Parfois, on est amené à prendre une copie d'un système d'information en cours d'exécution, puis ramasser les données de sources éventuellement hétérogènes (fichiers, bases de données relationnelles, temporelles, ...).

— *Prétraitement* : Les données peuvent contenir plusieurs types d'anomalies : les données peuvent être omises à cause des erreurs de frappe ou à causes des erreurs dues au système lui-même, dans ce cas il faut remplacer ces données ou éliminer complètement leurs enregistrements. Les données peuvent aussi être incohérentes c-à-d qui sortent des intervalles permis, on doit dans ce cas les écarter où les normaliser. Parfois on est obligé à faire des transformations sur les données pour unifier leur poids. Le pré-traitement comporte aussi la réduction des données qui permet de réduire le nombre d'attributs pour accélérer les calculs et représenter les données sous un format optimal pour l'exploration (Wirth et Jochen, 2000). Dans la majorité des cas, le prétraitement doit préparer des informations globales sur les données pour les étapes qui suivent tel que la tendance centrale des données (moyenne, médiane, mode), le maximum et le minimum, le rang, les quartiles, la variance, ... etc. Plusieurs techniques de visualisation des données telles que les courbes, les diagrammes, les graphes,... etc., peuvent aider à la sélection et le nettoyage des données. Une fois les données collectées, nettoyées et prétraitées on les appelle entrepôt de données (datawarehouse).

— *Estimation du modèle* : Dans cette étape, on doit choisir la bonne technique pour extraire les connaissances (exploration) des données. Des techniques telles que les réseaux de neurones, les arbres de décision, les réseaux bayésiens, le clustering, ... sont utilisées. Généralement, l'implémentation se base sur plusieurs techniques, puis on choisit le bon résultat obtenu(Wirth et Jochen, 2000).

— *Interprétation du modèle et établissement des conclusions* : généralement, l'objectif du Data Mining est d'aider à la prise de décision en fournissant des modèles compréhensibles aux utilisateurs. En effet, les utilisateurs ne demandent pas des pages et des pages de chiffres, mais des interprétations des modèles obtenus. Les expériences montrent que les modèles simples sont plus compréhensibles mais moins précis, alors que ceux complexes sont plus précis mais difficiles à interpréter (Wirth et Jochen, 2000).

La classification est une technique d'exploration de données utilisée pour prédire l'appartenance à un groupe pour des instances de données. C'est l'une des techniques importantes de l'exploration de données et elle est utilisée dans diverses applications

telles que la gestion de la relation client, la reconnaissance des formes, le diagnostic des maladies et le marketing ciblé.
Les problèmes de classification des données se concentrent sur l'apprentissage d'un ensemble de données, puis utilisent les informations des données d'apprentissage pour prédire la nature de tout nouveau point de référence. La classification des données peut être une méthode d'apprentissage supervisée ou non supervisée selon la disponibilité d'information d'étiquette sur les données d'apprentissage. Les algorithmes de classification des données supervisées étudient l'ensemble étiqueté d'apprentissage (des étiquettes de classe de données d'apprentissage sont disponibles) et génèrent un modèle de classification permettant de classer tout ensemble de nouveaux points de données observées. Les algorithmes non supervisés, d'autre part, essaient de trouver des motifs dans les données d'entraînement non étiquetées.
Dans notre thèse nous sommes intéressés à traiter la classification supervisée.

1.3 La classification supervisée

La classification supervisée est une technique d'exploration de données utilisée pour prédire l'appartenance à un groupe d'instances de données en se basant sur des informations connues a priori(Kim et Ryu, 2004). C'est un processus à deux étapes : une étape d'apprentissage et une étape de classification.

L'étape d'apprentissage consiste à fournir un modèle, une fonction, permettant d'associer automatiquement des données à un élément d'un ensemble Y de classes. L'étape de classification, consiste à classer les nouvelles instances en utilisant le modèle construit dans la première étape. En général, un système n'est prêt pour une utilisation réelle qu'après une succession d'étapes d'apprentissage et de test, permettant de mettre en place une stratégie de classification efficace. Une fois le classifieur construit, il est essentiel de le valider en essayant d'estimer les erreurs de classification qu'il peut engendrer, autrement dit, la probabilité que la classe prédite pour une donnée quelconque soit incorrecte.

Le modèle construit par un algorithme d'apprentissage doit en général remplir un certain nombre de critères. Citons à titre d'exemple :

— Le taux d'erreur doit être le plus bas possible. Ce point peut être mesuré en utilisant plusieurs critères d'évaluation. A titre d'exemple, la précision Average Acceleration Capacity (ACC), l'air sous la courbe de Receiver Operating Cha-

racteristic (ROC) AUC et l'indice Adjusted Rand Index (ARI),..., etc.

— Il doit être aussi peu sensible que possible aux fluctuations aléatoires des données d'apprentissage.

— les décisions de classification doivent autant que possible être explicites et compréhensible.

Les systèmes d'apprentissage, qui permettent de prédire l'appartenance d'un nouvel exemple à une classe, peuvent être basés sur des hypothèses probabilistes (classifieur naïf de Bayesien), sur des notions de proximité (plus proches voisins) ou sur des recherches dans des espaces d'hypothèses (arbres de décision).

1.4 Classification basée sur un ensemble de classifieurs

La combinaison des classifieurs est devenue au fil du temps un domaine de recherche très riche, par rapport à l'utilisation d'un seul classifieur (Sluban *et al.*, 2014). Les méthodes de sélection et de combinaison ont prouvé leur performance dans de nombreuses applications. Les résultats des classifieurs sont combinés en utilisant des méthodes de combinaison. Deux grandes stratégies de combinaison de classifieurs sont évoquées dans la littérature : "la sélection" et "la fusion" de classifieur. L'avantage de la sélection de classifieur revient à le fait que chaque classifieur est performant sur une petite zone de l'espace des caractéristiques. Lorsqu'on souhaite émettre une décision sur un vecteur de caractéristiques $x \in R_n$, on regarde l'ensemble des réponses des classifieurs et celui donnant la plus grande proximité est sélectionné pour attribuer une classe au vecteur x. La fusion de classifieurs suppose que tous les classifieurs soient équivalents au niveau des performances sur l'ensemble de l'espace des caractéristiques. Les décisions de tous les classifieurs sont prises en compte pour la décision finale. Cette méthode a l'avantage d'être plus fiable dans certains cas, dans la mesure qu'on utilise la performance de plusieurs classifieurs pour avoir une seule décision par la fin. L'objectif d'utiliser un ensemble de classifieurs c'est d'avoir un ensemble divers de sélections. Cette diversité est la clé de cette approche.
Trois manières d'utiliser un ensemble de classifieurs (Zhang *et al.*, 2011) : parallèle, séquentiel et hybride.

1.4.1 Approche séquentielle

Dans la combinaison séquentielle on applique un ensemble de classifieurs C_i (i=1,2,...M) sur un seul sous-ensemble X_n d'une manière successive en utilisant l'équation suivante

(1.1). Chaque classifieur prend en compte la réponse du classifieur précèdent pour produire une décision. Cela peut être vu comme filtrage progressif des décisions qui permet de diminuer le taux d'erreur globale.

$$C_{AS}(X) = \frac{1}{M}\sum_{i=1}^{N} C_i(X_n) \tag{1.1}$$

Ce qui fait, les premiers classifieurs utilisés doivent être robustes, ce qui présente un inconvénient de l'approche séquentielle. Une connaissance a priori sur le comportement de chacun des classifieurs est exigée. D'autre part chaque classifieur dépend du classifieur précédent, donc une simple modification du premier classifieur peut provoquer un réapprentissage de toute la chaine.

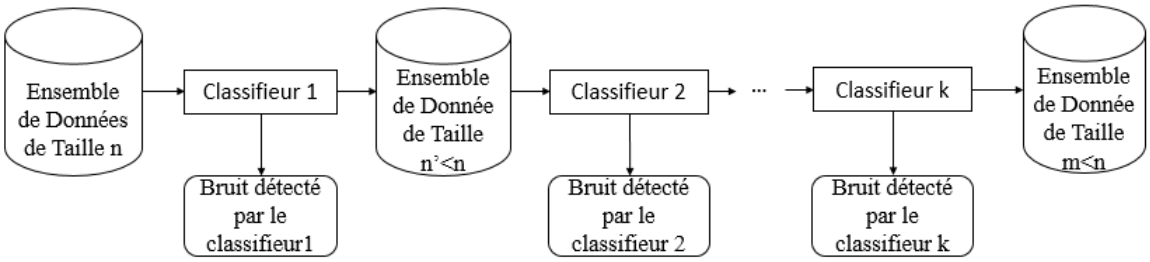

Figure 1.2 – Approche séquentielle.

1.4.2 Approche parallèle

Dans l'approche parallèle, un ensemble de classifieurs sont appliqués indépendamment sur les differentes sous ensembles de données C_i (i=1,2,...N)comme il est illustré dans la figure (1.3). On fusionne leurs réponses par la suite respectivement en ne favorisant aucun classifieur par rapport à l'autre, selon l'équation suivante (1.2).

$$C_{AP}(X) = \frac{1}{N}\sum_{i=1}^{N} C_i(X_i) \tag{1.2}$$

On pourra aussi attribuer des poids à la réponse de chaque classifieur selon sa performance. C'est une approche simple, mais l'inconvénient majeur de l'approche parallèle est qu'elle nécessite l'activation de tous les classifieurs du système qui doivent participer de manière concurrente et indépendante. Un autre inconvénient est que la décision est prise pour chaque sous-ensemble à partir d'un seul classifieur. Par conséquent, cela peut conduire à une prédiction erronée.

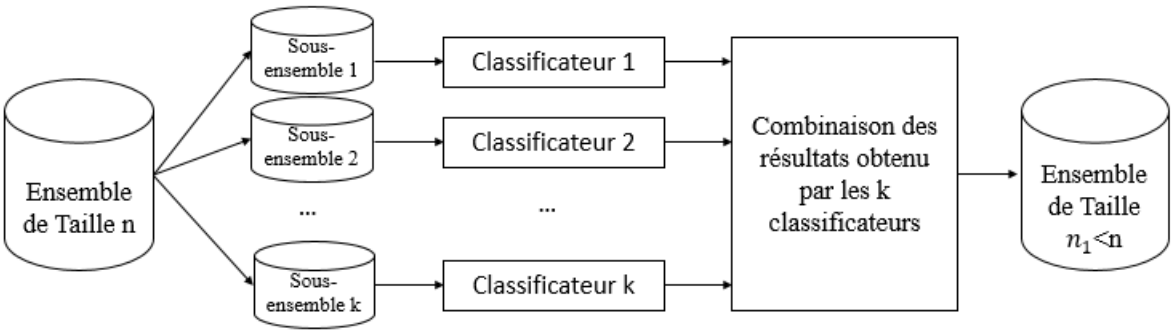

Figure 1.3 – Approche parallèle.

1.4.3 Approche hybride

L'approche hybride combine les deux approches citées précédemment afin de tirer l'avantage de chacun des deux. Cette combinaison est vérifiée avec cette formule(1.3).

$$C_{AH}(X) = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}(X) \tag{1.3}$$

Cette approche a l'avantage aussi de se bénéficier de performance des plusieurs classifieurs. Figure (1.4) présente un exemple d'une combinaison hybride. De nombreuses

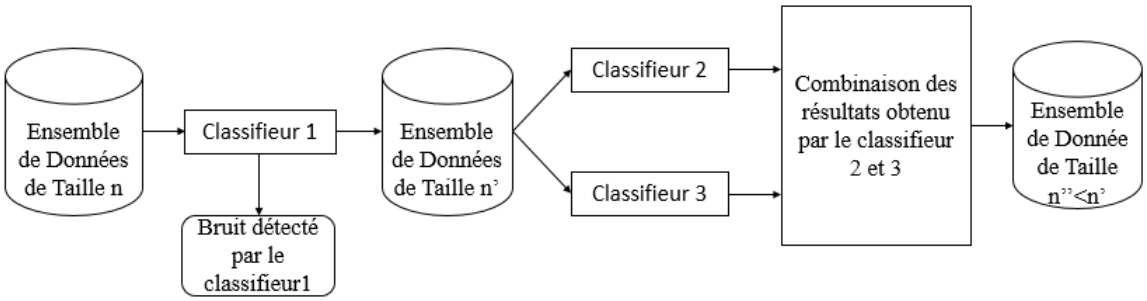

Figure 1.4 – Approche hybride (exemple de 3 classificateurs).

études montrent que la combinaison des classifieurs (séquentiel, parallèle ou hybride) améliore significativement les performances du système de reconnaissance par rapport à un seul classifieur.

Lors de la réalisation d'une classification basée sur plusieurs classifieurs, on peut combiner leurs prédictions à l'aide de l'une des techniques suivantes (Parhami, 1994) : vote à la majorité, vote par consensus, combinaison par somme pondérée ou combinaison avec un méta-classement.

1.4.4 Les méthodes de combinaison des résultats

1.4.4.1 Vote avec seuil

Les méthodes de vote peuvent pratiquement toutes être dérivées de la règle de vote avec un seuil exprimée par l'équation (1.4) :

$$E(X) = \begin{cases} C_i S_i \sum_{j=1}^{L} e_{i,j} = max_{t=1}^{N} \sum_{j=1}^{L} e_{i,j} \geqslant \lambda.L \\ Rejet\ Sinon \end{cases} \tag{1.4}$$

Où pour chaque objet x :

— L est le nombre de classifieurs ;
— C_i est la $i^{ème}$ classe retenue par le vote attribué par le $j^{ème}$ classifieur e_j à x ;
— λ correspond à la proportion des classifieurs devant répondre par la même classe pour que cette dernière soit retenue comme résultat de la combinaison. En fonction de la valeur que peut prendre λ, les votes suivants sont distingués.
Pour $\lambda = 0$, il s'agit du vote à la pluralité :

— La classe qui reçoit le plus de votes est élue comme classe finale ;
— Si toutes les classes ont le même nombre de votes, il y a un rejet.

Pour $\lambda = 0{,}5$, il s'agit du vote à la majorité :

— La classe qui reçoit plus que la moitié des votes est élue ;

Pour $\lambda = 1$, il s'agit du vote à l'unanimité :

— La classe finale est élue si tous les classifieurs proposent cette réponse. Autrement, un rejet est automatique ;
— Ce vote est plutôt fiable. Cependant, il présente l'inconvénient majeur de produire un taux de reconnaissance assez faible, d'autant plus lorsque le nombre de classifieurs augmente.

1.4.4.2 Vote majoritaire

Le vote est un moyen très simple de combiner les résultats de plusieurs classifieurs. Le résultat final sera une combinaison linéaire de tous les résultats trouvés (Parhami, 1994). Le vote majoritaire consiste à choisir la classe la plus proposée par les classifieurs, par exemple si nous avons 3 classifieurs on doit donc avoir (2 sur 3 ou 3 sur 3).

1.4.4.3 Vote par consensus

Le vote par consensus exige que tous les classifieurs classifient l'instance en tant que bruit pour qu'elle soit considérée comme un point aberrant (Parhami, 1994). Il doit donc y avoir un accord complet entre tous les classifieurs.

1.4.4.4 Combinaison par somme pondérée

Dans le cas normal, tous les poids sont constants pour tous les classifieurs sur l'ensemble des données (Parhami, 1994). Dans le cas où les classifieurs sont spécialisés dans un domaine particulier de l'espace de données, des pondérations sont attribuées aux entrées. Ainsi, l'espace de données est divisé et chaque partie est attribuée à un classifieur. Donc, pour prendre en compte l'importance d'un classifieur par rapport aux autres, une pondération a été utilisée pour représenter le concept d'importance. Le degré d'importance doit être connu a priori dans la phase d'apprentissage et la phase de test. Chaque classifieur donne une valeur qui correspond à la classe de sortie. Les résultats seront ensuite reclassés pour donner le résultat final. Dans le vote avec pondération, la réponse de chaque classifieur e_j est alors pondérée par un coefficient W_j indiquant son importance dans la combinaison comme il est montré par l'équation (1.5)

$$E(X) = \begin{cases} C_i S_i \sum_{j=1}^{L} W_j e_{i,j} = max_{t=1}^{N} \sum_{j=1}^{L} W_j e_{t,j} \\ Rejet\ Sinon \end{cases} \tag{1.5}$$

Il existe plusieurs façons de déterminer les coefficients W_j. Ils peuvent par exemple avoir été optimisés par un algorithme génétique ou selon la fiabilité de chaque classifieur en se basant sur leurs performances individuelles.

1.4.4.5 Combinaison avec un méta-classifieur

Différents résultats sont combinés par un nouveau processus d'apprentissage effectué par un "méta-classifieur". À partir de cette nouvelle prédiction, un ensemble d'apprentissage est généré pour le méta-classifieur : chaque instance X_i aura comme description la classe $C(x)$ à laquelle elle appartient, ainsi que les prédictions faites par chaque classifieur (Parhami, 1994). Le méta-classifieur utilise ce nouvel ensemble de formation pour effectuer la classification des données.

1.4.5 Les méthodes ensemblistes

Bootstrap Aggregating (Bagging) est une approche de construction d'ensemble qui utilise différents sous-ensembles de données d'apprentissage avec une méthode de classification unique (Liu *et al.*, 2009). Étant donné un ensemble d'apprentissage de taille t, Bagging prend des instances aléatoires t de l'ensemble de données avec remplacement (à l'aide d'une distribution uniforme). Ces t instances sont apprises, et ce processus est répété plusieurs fois. Étant donné que le tirage au sort est effectué avec remplacement, les instances tirées contiendront des doublons et des omissions par rapport à l'ensemble d'apprentissage initial. Chaque cycle à travers le processus aboutit à un classifieur. Après la construction de plusieurs classifieurs, les sorties de chaque classifieur sont combinées pour produire la prédiction finale (Dietterich, 2000).

Une autre approche appelée **« Boosting »** utilise également une méthode d'apprentissage unique avec différents sous-ensembles de données d'apprentissage (Dietterich, 2000). Sa structure globale est similaire à celle de la méthode Bagging, à la différence qu'elle conserve la trace de la performance de l'algorithme d'apprentissage et se concentre sur les cas qui ne sont pas correctement appris. Au lieu de choisir les instances t d'apprentissage à l'aide d'une distribution uniforme de manière aléatoire, les exemples d'apprentissage sont sélectionnés en favorisant les instances qui ne sont pas bien classées. Après plusieurs cycles, la prédiction est réalisée selon un vote pondéré des prédictions de chaque classifieur. Ainsi, les poids sont proportionnels à la précision de chaque classifieur sur son ensemble d'apprentissage. L'algorithme le plus connu de l'approche Boosting, appelé « AdaBoost ».

Les forêts aléatoires (plus connues sous Random Forest) sont une combinaison d'arbres de décision, où chaque arbre dépend des valeurs d'un vecteur aléatoire indépendamment échantillonné et avec la même distribution pour tous les arbres de la forêt (Dietterich, 2000). L'erreur de généralisation d'une forêt d'arbres dépend de la force des arbres individuels dans la forêt et de la corrélation entre eux. L'utilisation d'une sélection aléatoire de caractéristiques pour diviser chaque noeud donne des taux d'erreur qui se comparent favorablement à Adaboost.

La méthode **Bagging** exploite le principe de la diversité en utilisant les différents ensembles de données perturbées dans la base d'apprentissage des classifieurs. Autrement dit, chaque classifieur de base est entrainé sur un sous-ensemble d'échantillons pour obtenir une hypothèse de classification légèrement différente, puis il est combiné avec les

autres classifieurs pour former l'ensemble. D'un autre côté, la méthode Random Forest utilise le même principe mais avec différents ensembles d'attributs. En ce qui concerne la méthode boosting, la diversité est obtenue en augmentant les poids des échantillons mal classés de manière itérative. Généralement, ces trois méthodes (figure 1.5) utilisent des arbres de décision pour leur sensibilité aux petits changements sur l'ensemble d'apprentissage, et sont donc adaptées à la procédure de perturbation appliquée aux données d'apprentissage.

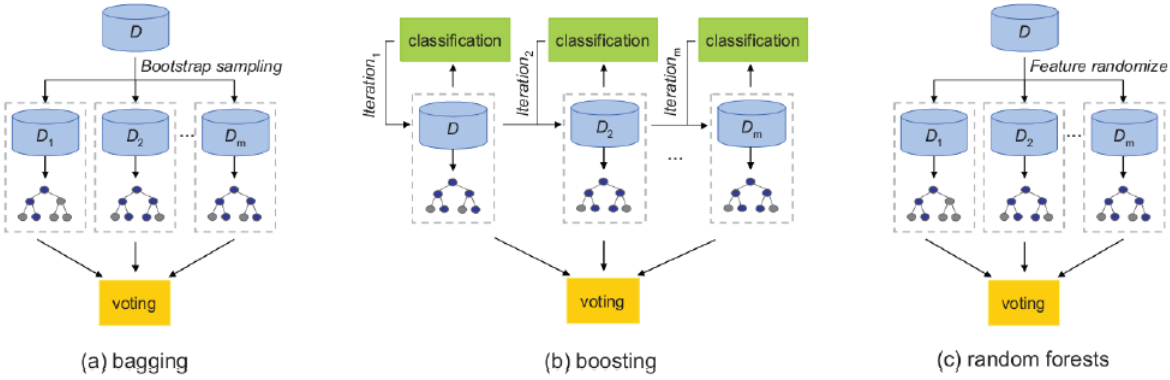

Figure 1.5 – Méthodes ensemblistes

1.5 Prétraitement pour le Data Mining

Dû à la grande taille des bases de données actuelles, les données brutes sont généralement de faible qualité. Elles sont très sensibles au bruit (valeurs erronées, aberrantes ou des erreurs de frappe), aux incohérences (divergence entre attributs) et aux valeurs manquantes, dans ce cas il faut remplacer ces données ou éliminer complètement leurs enregistrements, afin d'aider à améliorer la qualité des données. En effet l'application des algorithmes de Data Mining sur de telles données complexifie l'apprentissage et nuit à la performance ainsi qu'à la fiabilité du modèle. Le prétraitement des données est une étape cruciale et critique dans le processus d'extraction de connaissances (1.6) surtout en cas de grandes quantité de données. En effet, il permet d'améliorer la qualité des données soumises par la suite aux algorithmes de Data Mining. Les méthodes de prétraitement des données (figure 1.7) sont divisées en catégories suivantes (Alasadi et Bhaya, 2017) :

— Nettoyage des données
— Intégration de données
— Transformation de données
— Réduction des données

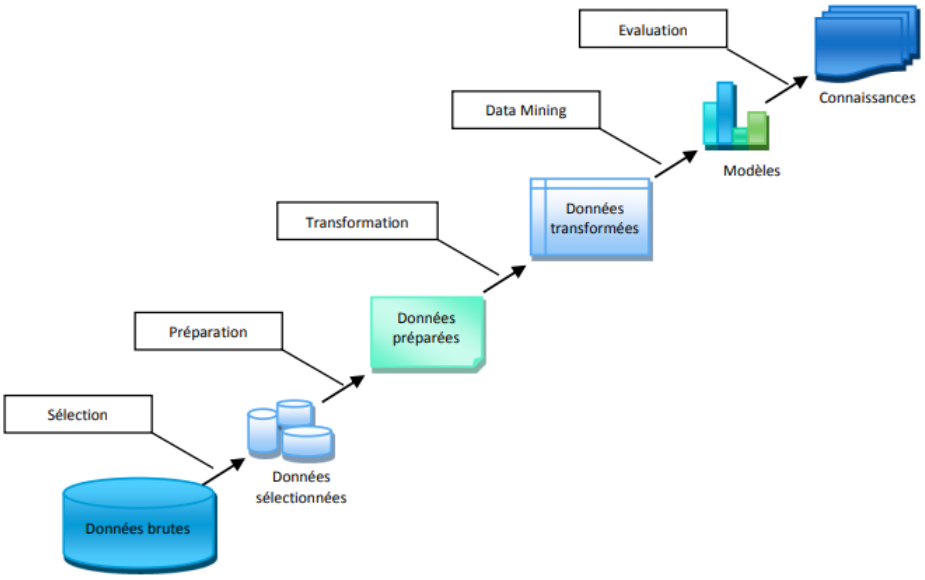

Figure 1.6 – Différentes étapes du processus d'extraction de connaissances

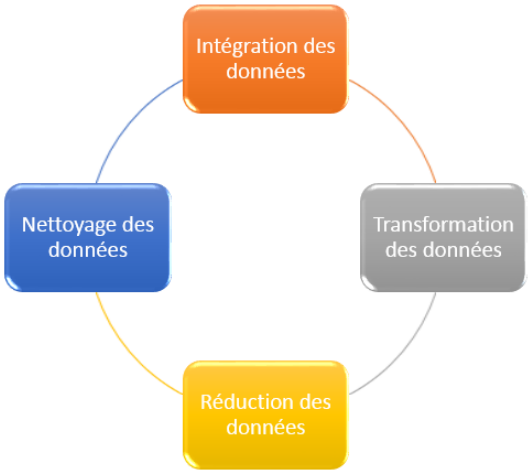

Figure 1.7 – Prétraitement des données

Nettoyage des données : Les données à analyser à l'aide des techniques d'exploration de données peuvent être incomplètes (valeurs d'attributs manquantes ou présentant un intérêt particulier, ou ne contenant que des données agrégées), bruyantes (contenant des erreurs ou des valeurs aberrantes s'écartant de celles attendues) et incohérentes (par exemple, contenant des différences dans les codes de département utilisés pour classer les articles). Les données incomplètes, bruyantes et incohérentes sont des propriétés courantes des grandes bases de données et des entrepôts de données du monde réel. Les données incomplètes peuvent survenir pour diverses raisons. Les attributs d'intérêt peuvent ne pas toujours être disponibles, tels que les informations client pour les données de transaction. D'autres données peuvent ne pas être incluses simplement parce qu'elles n'étaient pas considérées comme importantes au moment de la saisie. Les données perti-

nentes peuvent ne pas être enregistrées en raison d'un malentendu ou d'un dysfonctionnement de l'équipement. D'autre part les données incompatibles avec d'autres données enregistrées peuvent avoir été supprimées. En outre, l'enregistrement de l'historique ou les modifications des données ont peut-être été négligés. Il peut être donc nécessaire d'inférer les données manquantes. Les données peuvent être bruyantes, avec des valeurs d'attributs incorrectes, pour les raisons suivantes : Les instruments de collecte de données utilisés peuvent être défectueux. Des erreurs humaines ou informatiques peuvent survenir lors de la saisie des données. Des erreurs de transmission de données peuvent également se produire. Il peut exister des limitations technologiques, telles que la taille limitée de la mémoire tampon permettant de coordonner le transfert et la consommation synchronisés de données. Des données incorrectes peuvent également résulter des incohérences dans les conventions de nommage ou les codes de données utilisés. Les attributs en double nécessitent également un nettoyage des données. Le traitement de nettoyage des données traditionnel permettent de "nettoyer" les données en remplissant les valeurs manquantes, en remplaçant les données bruyantes, en identifiant ou en supprimant les valeurs aberrantes, et en résolvant les incohérences. Des données bruitées peuvent créer de la confusion pour la procédure d'extraction. Les données bruyantes ne sont pas toujours robustes. Par conséquent, une étape de prétraitement utile consiste à exécuter les données à l'aide de certains traitements de nettoyage des données.

Intégration de données : Il est probable qu'une tâche d'analyse de données implique une intégration de données, qui combine des données provenant de sources multiples dans un ensemble de données cohérent. Ces sources peuvent inclure plusieurs bases de données ou fichiers. Un certain nombre de problèmes doivent être pris en compte lors de l'intégration des données. L'intégration de schéma peut être délicate. Les bases de données et les entrepôts de données contiennent généralement des métadonnées, c'est-à-dire des données relatives aux données. Ces métadonnées peuvent être utilisées pour éviter des erreurs lors de l'intégration du schéma. La redondance est un autre problème important. Un attribut peut être redondant s'il peut être "dérivé" d'une autre table. Des incohérences dans la désignation des attributs ou des dimensions peuvent également entraîner des redondances dans l'ensemble de données résultant.

Transformation des données : Dans la transformation de données, les données sont transformées dans des formes appropriées à l'extraction. La transformation de données peut impliquer ce qui suit :

— Normalisation, où les données d'attribut sont mises à l'échelle de manière à tomber dans une petite plage spécifiée, telle que -1,0 à 1,0 ou 0 à 1,0.

— Agrégation, où des opérations de synthèse ou d'agrégation sont appliquées aux données. Par exemple, les données de ventes quotidiennes peuvent être agrégées de manière à calculer des montants totaux mensuels et annuels. Cette étape est généralement utilisée dans la construction d'une base de données pour l'analyse des données à plusieurs granularités.

— Généralisation des données, où les données de bas niveau ou «primitives» (brutes) sont remplacées par des concepts de niveau supérieur par le biais de l'utilisation des hiérarchies de concepts. Par exemple, les attributs catégoriels, tels que «rue», peuvent être généralisés à des concepts de niveau supérieur, tels que «ville» ou «pays». De même, les valeurs d'attributs numériques, telles que l'âge, peuvent être mappées à des concepts de niveau supérieur, tels que jeune, d'âge moyen et supérieur.

Réduction Des données : L'analyse et l'exploitation d'énormes quantités de données peuvent prendre un temps très long, rendant cette analyse irréalisable. Les techniques de réduction des données ont été utiles pour analyser la représentation réduite de l'ensemble de données, sans compromettre l'intégrité des données d'origine, tout en produisant des connaissances de qualité. Le concept de réduction des données est généralement compris comme une réduction du volume ou une réduction des dimensions (nombre d'attributs). Un certain nombre de méthodes ont facilité l'analyse d'un volume ou d'une dimension réduit de données, tout en produisant des connaissances utiles. Certaines méthodes basées sur des partitions fonctionnent sur des sous ensembles de données. En d'autres termes, l'exploration sur un ensemble de données réduit devrait être plus efficace, afin d'accélérer les calculs et représenter les données sous un format optimal pour l'exploration tout en produisant les mêmes (ou presque les mêmes) résultats d'analyse.

1.6 Classification des données à grande dimension

L'application d'un traitement de filtrage direct sur l'ensemble de données présente de nombreux inconvénients par rapport à la technique de partitionnement des données, cette dernière permet de paralléliser le traitement en différents sous-ensembles de données. Ainsi que traiter l'ensemble de donnée d'un seule cout peut entraîner un temps d'exécution exponentiel particulièrement lorsque nous parlons de grands ensembles de données, ainsi qu'un pourcentage élevé d'erreur d'étiquetage et par conséquent une grande perte d'information.

En effet, le partitionnement consiste à diviser les données d'entraînement d'origine à des ensembles d'entraînement plus petits. Un classifieur (ou ensemble des classifieurs) est formé sur chaque sous-ensemble en choisissant l'un des méthodes de combinaisons suivantes : parallèle, séquentiel ou hybride. Une fois tous les classifieurs construits, les modèles sont combinés (Maimon et Rokach, 2005). En effet, le partitionnement d'un ensemble de données en différents partitions disjointes permettra non seulement de surmonter le problème du dépassement de la taille de la mémoire, mais il conduit aussi à créer un ensemble de classifieurs divers et précis, chacun est construit à partir d'une partition disjointe. Cela peut améliorer les performances de la classification.

Deux catégories de systèmes parallèles peuvent être distinguées : partitionnement vertical et horizontal. Dans le partitionnement horizontal, l'ensemble de données est divisé en plusieurs paquets qui ont les mêmes caractéristiques que l'ensemble de données original, chacun contient un sous-ensemble des instances d'origine (voir la figure 1.8).

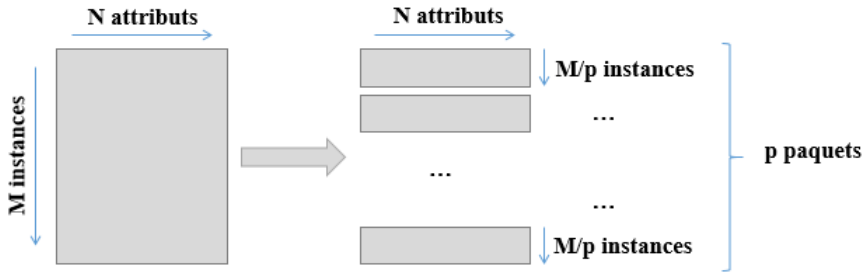

Figure 1.8 – Partitionnement horizontal

Plus récemment, un framework pour construire plusieurs classifieurs formés à partir de petits sous-ensembles de données dans un environnement distribué a été proposé (Chawla *et al.*, 2004). L'algorithme robuste d'apprentissage (RLB) proposé par (Christmann *et al.*, 2007) est également conçu pour fonctionner en cas de grands ensembles de données. L'algorithme Cluster-Based Concurrent Decomposition (CBCD) (cluster-based concurrent decomposition) (Christmann *et al.*, 2007) regroupe d'abord l'espace d'entrée en utilisant l'algorithme de classification K-means. Ensuite, il crée des sous-échantillons disjoints en utilisant les grappes (clusters), de telle manière que chaque sous-échantillon est composé d'instances de tous les clusters et représente donc l'ensemble de données. Un classifieur est appliqué à son tour à chaque sous-échantillon. Un mécanisme de vote est utilisé pour combiner les classifications des classifieurs. Une étude expérimentale indique que l'algorithme CBCD surpasse l'algorithme de bagging.

Dans le partitionnement vertical, l'ensemble de données d'origine est divisé en plusieurs paquets ayant le même nombre d'instances que l'ensemble de données d'origine,

chacun contient un sous-ensemble d'attributs de l'ensemble d'origine (voir la figure 1.9).

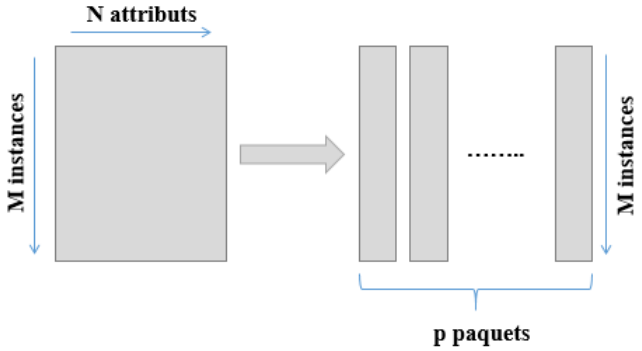

Figure 1.9 – Partitionnement vertical

(Rokach *et al.*, 2006) affirment que le partitionnement de l'ensemble d'attributs facilite potentiellement la création d'un classifieur pour les ensembles de données à haute dimension. De plus, ces méthodes peuvent être utilisées pour améliorer les performances de la classification en raison de la corrélation réduite entre les classifieurs. (Bryll *et al.*, 2003) indiquent également que la taille réduite de l'ensemble de données implique une induction plus rapide des classifieurs. Il existe trois stratégies populaires pour créer des ensembles basés sur des sous-ensembles d'attributs : une stratégie basée sur le hasard, la réduction et une autre basée sur la performance collective.

1.7 Conclusion

Dans ce chapitre, nous avons présenté le processus de prétraitement pour les grandes quantités de données. Nous avons souligné des travaux antérieurs dans un détail relativement minime et clair. Certes, le prétraitement des données à grande échelle acquiert, au fil du temps, de plus en plus d'intérêt, mais il reste toujours à répondre à certaines limitations. Plus précisément dans le domaine de la réduction de dimension et le nettoyage des données. Ce qui va être discuté davantage dans les chapitres suivants, dont on va analyser leurs points forts et leurs inconvénients dans le but de mettre l'accent sur leurs limites.

CHAPITRE

2 NETTOYAGE DES DONNÉES

Sommaire

2.1 Introduction

Force est de constater que l'on assiste aujourd'hui à une véritable explosion de la masse de données à traiter et la grande difficulté consiste à bien appréhender leur diversité. Ces données constituent un gisement de connaissances, qui bien traité et analysé, peut s'avérer un puissant atout de performance dans une entreprise ou une organisation ; mais qui, à l'inverse, mal appréhendé, peut avoir des répercussions très négatives.
Le bruit peut être divisé en deux types (Zhu et Wu, 2004) :

— Le bruit d'attributs : il est donné par les erreurs survenues lors de la saisie des valeurs des attributs. Parmi les sources de ce type de bruit on trouve : les variables avec valeurs manquantes et les données redondantes.

— Le bruit de classe : il est donné par les erreurs introduites lors de l'affectation des instances aux classes.

Ces deux catégories de bruit devraient être mieux détectées et éliminées des données. D'autres irrégularités, telles que les observations aberrantes (outliers), «une observation aberrante est une observation qui apparaît distincte des autres observations» (Zhu *et al.*, 2003), peuvent ne pas être erronées, mais seulement des cas particuliers d'exemples réguliers. Il est démontré que le bruit de classe est potentiellement plus nocif que le bruit d'attribut, ce qui souligne l'importance de la prise en compte de ce type de bruit. Cela s'explique principalement par deux raisons :

Premièrement, pour une instance donnée, il existe de nombreuses caractéristiques ou attributs, alors qu'une seule étiquette de classe est associée à cette instance.

Deuxièmement, l'importance de l'attribut est variable, alors que l'étiquette de classe est toujours importante, car elle oriente l'apprentissage vers la règle de classification souhaitée. En effet, de nombreuses recherches ont suggéré que, dans la plupart des cas, l'élimination des instances contenant le bruit d'attribut n'est pas toujours correcte ni bénéfique, car de nombreux attributs de l'instance peuvent toujours contenir des informations précieuses (Zhu *et al.*, 2003). Cependant, l'élimination du bruit de classe améliorera la précision de la classification. Dans cette thèse nous somme intéressés à la détection et l'élimination du bruit de classe.

2.2 Bruit de classe

2.2.1 Définition

La classification consiste à prédire la classe de nouveaux échantillons, en utilisant un modèle déduit des données d'entraînement. Il est supposé que chaque échantillon d'apprentissage est associé à des étiquettes observées. Cette étiquette correspond souvent à la vraie classe de l'échantillon, mais il est tout à fait possible d'être confronté à des erreurs d'étiquetage, c'est-à-dire à des exemples s'étant vus attribuer une classe erronée. L'échantillon peut être soumis à un processus de détection de bruit avant d'être présenté à l'algorithme d'apprentissage (Zhu *et al.*, 2003). C'est donc important de distinguer la vraie classe d'une instance de son étiquette observée. En effet il est important de savoir prendre en compte ce type de phénomènes, de gérer ces distributions altérées et donc de définir un cadre théorique à ce que nous appellerons apprentissage sous bruit de classification. Le traitement du bruit de classe est étroitement lié à la détection des valeurs aberrantes (Knorr *et al.*, 2000), et à la détection des anomalies (Breunig *et al.*, 2000). En effet, les instances mal étiquetées peuvent être aberrantes, si leur étiquette a

une faible probabilité de se produire dans leur voisinage. De même, les instances peuvent également sembler ”anormales”, par rapport à la classe qui correspond à leur étiquette incorrecte. Par conséquent, il est naturel que dans la littérature de nombreuses techniques de détection de bruit de classe soient très proches des techniques de détection des valeurs aberrantes et des anomalies. Cependant, il faut souligner que les instances mal étiquetées ne sont pas nécessairement des aberrations ou des anomalies. Par exemple, si des erreurs d'étiquetage surviennent dans une région limitée où toutes les classes sont équiprobables, les instances mal étiquetées sont des événements rares et ne semblent pas anormaux. De même, une valeur aberrante n'est pas nécessairement un échantillon mal étiqueté (Knorr *et al.*, 2000), car elle peut être due à un bruit d'attribut ou simplement être un événement à faible probabilité.

2.2.2 Sources

L'erreur d'étiquetage ou le le bruit de classe peut être causée par différents problèmes (Zhu et Wu, 2004) :

Premièrement, les informations disponibles peuvent être insuffisantes pour effectuer un étiquetage fiable. Par exemple si les valeurs de certaines attributs sont inconnues ou bien si le langage de description est trop limité, la quantité d'informations disponibles sera très réduites. Dans certains cas, l'information peut être également de qualité médiocre ou variable. Par exemple, les réponses d'un patient pendant l'anamnèse peuvent être imprécises ou incorrectes ou même différentes si la question est répétée.

Deuxièmement, les experts font souvent des erreurs lors de l'étiquetage. L'étiquetage est une tâche longue et coûteuse. Il existe donc un intérêt croissant pour l'utilisation d'étiquettes non couteuses, faciles à obtenir et automatique en utilisant des Frameworks comme par exemple l'Amazon Mechanical Turk1 (Buhrmester *et al.*, 2011). Les étiquettes fournies par un non-expert sont moins fiables. Or, la richesse des étiquettes disponibles peut atténuer ce problème.

Troisièmement, la classification est dans certains cas subjectifs, comme par exemple dans les applications médicales ou l'analyse de données d'images. Il peut y avoir une variabilité importante dans un étiquetage par plusieurs experts.
Éventuellement, le bruit d'étiquette peut aussi provenir de problème du codage de données ou de communication. Par exemple, dans le filtrage des spams, les sources de bruit d'étiquette comprennent une mauvaise compréhension d'un clic accidentel. On estime que les bases de données du monde réel contiennent environ cinq pour cent d'erreurs de

codage (Zhu et Wu, 2004).

2.2.3 Taxonomie

Dans (Frénay et Verleysen, 2014), une nouvelle taxonomie du bruit d'étiquette est proposée, inspirée par le travail de (Schafer et Graham, 2002). Trois types de bruit sont distingués :

— Noisy Completely At Random (NCAR) : le bruit d'étiquette se produit indépendamment de la vraie classe et des valeurs d'attributs d'une instance.

— Noisy At Random (NAR) : le bruit d'étiquette dépend du vrai classe, qui peut être utilisée pour modéliser des situations où certaines classes sont plus susceptibles d'être mal étiquetées que d'autres.

— Noisy Not At Random (NNAR) : c'est le cas le plus général, où la probabilité d'avoir un mauvais étiquetage (ou un bruit de classe) dépend également des valeurs d'attributs.

2.3 Classification en présence de bruit de classe

2.3.1 Historique

La classification consiste à apprendre un classifieur à partir d'un ensemble de données étiquetées, afin de prédire la classe des nouveaux échantillons. Cependant, les ensembles de données du monde réel peuvent contenir du bruit. De nombreux travaux ont montré que le bruit de classe peut avoir un impact négatif sur la performance des classifieurs (Zhu et Xindong, 2004). D'où l'omniprésence du bruit semble être un problème important dans l'apprentissage automatique, par exemple dans les applications médicales où la plupart des tests de diagnostic médical ne sont pas précis à 100% (Sáez *et al.*, 2016b). Il est donc nécessaire de mettre en œuvre des techniques qui éliminent le bruit ou réduisent ses conséquences. D'un autre côté, le bruit de classe modifie les étiquettes observées affectées aux instances, par exemple en définissant de manière incorrecte une étiquette négative sur une instance positive dans la classification binaire. Dans (Zhu *et al.*, 2003) et (Khoshgoftaar et Rebours, 2004a), il est montré que le bruit de classe est potentiellement plus nocif que le bruit d'attribut, ce qui met en évidence l'importance de traiter ce type de bruit. La prévalence de l'impact du bruit de classe s'explique par le fait : 1) qu'il y a beaucoup d'attributs, alors qu'il n'y a qu'une seule étiquette et 2) que l'importance de chaque attribut pour l'apprentissage est différente, alors que les classes ont toujours un

impact important sur l'apprentissage. Des résultats similaires sont obtenus dans (Abellán et Masegosa, 2010a) : le bruit d'attribut semble moins nocif que le bruit de classe pour les arbres de décision, sauf lorsqu'un grand nombre d'attributs sont polluées par le bruit d'attribut. Cette section couvre la littérature sur le bruit de classe. En particulier, les différentes définitions et conséquences du bruit de classe sont discutées, ainsi que les différentes familles d'algorithmes qui ont été proposées pour traiter le bruit de classe sont présentées.

2.3.2 L'impact de bruit de classe sur l'apprentissage

Le bruit de classe est présent dans les ensembles de données du monde réel,il a plusieurs conséquences :

Premièrement, le bruit de classe diminue les performances de la prédiction, ce qui a été théoriquement prouvé pour des modèles simples comme les classifieurs linéaires (Bi et Jeske, 2010; Michalek et Tripathi, 1980), les classifieurs quadratiques (Lachenbruch, 1979) et les classifieurs kNN (Sánchez *et al.*, 1997; Wilson et Martinez, 2000). De nombreux travaux (Quinlan, 1986a; Pechenizkiy *et al.*, 2006; Sculley et Cormack, 2008) ont empiriquement confirmé ce problème pour d'autres classifieurs tels que les arbres de décision induits par C4.5 et les SVMs, ainsi que dans le filtrage des spams. Boosting est également bien connue par le fait d'être affecté par le bruit de classe (Opitz et Maclin, 1999; Jiang, 2001). En particulier, l'algorithme Boosting adapté AdaBoost tend à donner des poids trop importants aux instances mal étiquetées (Dietterich, 2000; Jiang, 2001; Abellán et Masegosa, 2010b).

Deuxièmement, le nombre nécessaire des instances d'apprentissage (Aslam et Decatur, 1996; Angluin et Laird, 1988) peut augmenter dans le cas d'un apprentissage en présence de bruit de classe. C'est aussi le cas pour la complexité des modèles, comme par exemple le nombre de nœuds d'arbres de décision (Abellán et Masegosa, 2010b; Quinlan, 1986b; Brodley et Friedl, 1999a) et le nombre des vecteurs dans les SVMs (Brodley et Friedl, 1999a; Libralon *et al.*, 2009).

Troisièmement, les fréquences observées des classes possibles peuvent être modifiées (Dawid et Skene, 1979), ce qui est très important particulièrement dans les contextes médicaux. En effet, les études médicales sont souvent préoccupées par la mesure de l'incidence d'une maladie donnée dans une population, dont l'estimation peut être biaisée par le bruit de classe. Ceci est également important dans la validation du modèle, car les mesures de performance peuvent être mal estimées en présence de bruit de classe (Lam et Stork, 2003). On peut conclure que les conséquences du bruit de classe sont très impor-

tantes et diversifiés : diminution des performances de classification, changements dans les exigences d'apprentissage, augmentation de la complexité des modèles appris, distorsion des fréquences observées, difficultés d'identifier les caractéristiques pertinentes, etc. La nature et l'importance des conséquences dépendent, entre autres, du type et de niveau de bruit de classe, de l'algorithme d'apprentissage et des caractéristiques de l'ensemble d'apprentissage. Par conséquent, il semble important de traiter le bruit de classe et de prendre en compte ces facteurs, avant l'analyse des données bruitées.
Il existe traditionnellement deux stratégies différentes pour traiter le bruit de classe, chacune étant liée à la phase dans laquelle elles fonctionnent :

Approches au niveau des algorithmes : leur objectif est de concevoir des classifieurs robustes afin de tolérer le bruit. Ils sont moins influencés par le bruit et ne nécessitent aucun traitement de bruit préalable (Miao *et al.*, 2016; Sun *et al.*, 2016).

Approches au niveau des données : ils tentent d'éliminer ou de nettoyer le bruit présent dans les données avant d'appliquer un classifieur (Brodley et Friedl, 1999b; Gamberger *et al.*, 1999a). C'est une option populaire si l'utilisation d'un apprenant robuste est irréalisable ou inappropriée, ou même si elle vise à améliorer les résultats des classifieurs robustes.

2.4 Méthodes traditionnelles de nettoyage de données

De nombreuses méthodes ont été proposées pour nettoyer les ensembles d'apprentissage, avec différents degrés de succès. Plusieurs méthodes sont dédiées pour détecter, supprimer ou ré-étiqueter des instances mal étiquetées. Tout d'abord, des méthodes simples basées sur des seuils. Des méthodes de filtrage basées sur des prédictions de modèle, qui incluent la classification basée sur le vote et le filtrage partitionné. Méthodes basées sur des mesures de l'impact de bruit de classe. Méthodes basées sur les plus proches voisins, les graphes et les ensembles. La méthode simple pour gérer le bruit de classe consiste à supprimer les instances qui semblent être mal étiquetées.
Il existe beaucoup de méthodes de nettoyage des données dans la littérature, comme par exemple la détection des valeurs aberrantes (Outliers)(Barnett et Lewis, 1974; Hodge et Austin, 2004) et la détection d'anomalie (Schölkopf *et al.*, 2001; Hoffmann, 2007; Chandola *et al.*, 2009). Ces methodes peuvent être réutiliser dans la détection de bruit de classe, on peut par exemple utiliser simplement des méthodes basées sur des mesures d'anomalie et supprimer les instances qui dépassent un certain seuil (Sun *et al.*, 2007). On peut aussi supprimer les instances qui augmentent de manière disproportionnée la

complexité du modèle (Gamberger *et al.*, 2000a, 1999b).

Les prédictions du modèle peuvent également être utilisées pour filtrer les instances (Gamberger *et al.*, 2000b; Khoshgoftaar et Rebours, 2004b). Un simple heuristique consiste à supprimer les instances d'entraînement qui sont mal classées par un classifieur (Thongkam *et al.*, 2008; Miranda *et al.*, 2009; Jeatrakul *et al.*, 2010), mais cela peut supprimer trop d'instances (Fayyad *et al.*, 1996). Un modèle basé sur des variantes locales (Segata *et al.*, 2009, 2010) a été proposé, ainsi qu'un filtrage basé sur des votes. Avec le filtrage de vote (Brodley et Friedl, 1999a; Gamberger *et al.*, 1999b; Khoshgoftaar et Rebours, 2004b; Sluban *et al.*, 2010), une instance est enlevée lorsque tous (ou presque tous) les apprenants d'un ensemble acceptent de l'enlever, ce qui peut être adapté pour des ensembles de données volumineux et distribués (Zhu *et al.*, 2003, 2006).

De nombreuses méthodes basées sur le kNN ont également été proposées (Wilson et Martinez, 2000; Libralon *et al.*, 2009; Delany *et al.*, 2012; Gates, 1972). Par exemple, Reduced Nearest Neighbours (Gates, 1972) suppriment les instances dont la suppression ne provoque pas une erreur de classification des autres instances. De plus, comme AdaBoost tend à donner des poids importants aux instances mal étiquetées, plusieurs approches utilisent ce comportement indésirable pour détecter le bruit de classe (Verbaeten et Van Assche, 2003; Wheway, 2000; Gao *et al.*, 2010). Par exemple, Verbaeten et al. (Verbaeten et Van Assche, 2003) supprime un pourcentage donné des instances ayant les plus grands coefficients de pondération.

Une autre approche a été proposée par (Hughes *et al.*, 2004) consiste à supprimer l'étiquette des instances (et non les instances elles-mêmes) pour laquelle les experts sont moins fiables. Par la suite, l'apprentissage semi-supervisé est effectué en utilisant à la fois les instances étiquetées et les (nouveaux) instances non étiquetées. Cette méthode n'a été utilisée que dans la segmentation ECG ; une question de recherche ouverte est de savoir si elle pourrait être appliquée à d'autres paramètres.

2.5 Approches de filtrage

le filtrage de bruit est une approche très courante pour le traitement des données bruyantes : les instances détectées comme étant bruitées par le filtre sont simplement supprimées. Les stratégies de filtrage les plus connues et robustes sont décrites ci-dessous :

Classification Filtre (CF) : CF (Gamberger *et al.*, 1999c) est une approche simple de filtrage de bruit. Il divise l'ensemble d'apprentissage en plusieurs sous-ensembles. Un seul classifieur est utilisé pour chaque sous-ensemble. Les exemples qui sont mal classés par un apprenant de base sont alors éliminés à partir de l'ensemble de données d'entraî-

nement.

Ensemble Filtre (EF) : EF (Brodley et Friedl, 1999c) classe les données d'apprentissage en utilisant un ensemble de classifieurs. Ensuite, les résultats sont combinés en utilisant un système de vote consensus ou majoritaire. Par conséquent, le bruit de classe est supprimé à partir des données d'apprentissage. L'avantage principal de ce type de filtre est basé sur l'hypothèse que la collecte de prédictions de différents classifieurs pourrait fournir une meilleure détection de bruit de classe que la collecte d'informations à partir d'un seul classifieur. Comme il est montré dans (Guan *et al.*, 2013), les auteurs proposent d'utiliser le vote multiple pour la détection du bruit de classe, ce qui peut donner de meilleures performances qu'un classifieur simple.

Iterative-Partitioning Filtre (IPF) :(Khoshgoftaar et Rebours, 2007) Le filtre de partitionnement itératif ne construit qu'un seul apprenant de base en plusieurs itérations jusqu'à ce qu'un critère d'arrêt donné soit atteint. Dans chaque itération, l'ensemble de données d'apprentissage est d'abord partitionné en plusieurs sous-ensembles. Un classifieur de base est construit sur chacun de ces sous-ensembles. Ensuite, les instances mal classées sont éliminées (en utilisant un schéma de vote) et une nouvelle itération est lancée. Le fait que le bruit de classe soit éliminé itérativement est le point fort de ce type de filtres.

Iterative Noise Filter based on the Fusion of Classifiers (INFFC) : (Sáez *et al.*, 2016a) basé sur trois étapes principales. Tout d'abord, un filtrage préliminaire est appliqué en utilisant un filtre de classification, pour éliminer le bruit de classe dans l'itération courante. Ce premier filtrage permet de réduire l'influence du bruit de classe dans les étapes suivantes. Ensuite, un autre filtre (CF) est construit à partir des données partiellement propre résultat du filtrage préliminaire. Le résultat de cette étape sont deux ensembles : l'ensemble des données nettoyés et un ensemble de bruit de classe détecté. Enfin, afin de contrôler la sensibilité du filtre au bruit, un score de bruit est calculé, de sorte que le bruit de classe est finalement éliminé s'il dépasse le score de bruit.

Multiedit (ME) :(Devijver, 1986) les données d'entraînement sont partitionnées en n sous-ensembles. Le classifieur k-Nearest Neighbor est appliqué à partir de l'ensemble x en considérant l'ensemble (x + 1) mod n comme un ensemble de données d'apprentissage et les exemples mal étiquetés sont éliminés. Ce processus est répété jusqu'à ce qu'aucun exemple ne soit supprimé.

Edited Nearest Neighbor (ENN) :(Sánchez *et al.*, 2003) L'idée principale de cet algorithme est que les exemples qui ont une étiquette de classe différente de celle de la majorité de ses k plus proches voisins, sont supprimés. Nearest Centroid Neighbor Edition (NCNE) (Tomek, 1976) : Ceci est une modification de 'ENN' pour identifier et

éliminer le bruit de classe, qui consiste à enlever chaque exemple mal classé par les k plus proches voisins du centroïde (k-NCN).

All k-Nearest Neighbors (AllKNN) :(Zhang *et al.*, 2018) En variant le nombre de voisins entre 1 et k, la règle k-NN est appliquée k fois. Quand toutes les valeurs de k ont été considérées, les exemples mal classés sont écartés de l'ensemble d'entraînement.

2.6 Approches parallèles distribuées

L'apprentissage automatique distribuée peut naturellement résoudre la complexité de l'algorithme ainsi que le problème de la limitation de la mémoire pour les problèmes de l'apprentissage automatique en cas de grands ensembles des données (Peteiro-Barral et Guijarro-Berdiñas, 2013). Pour pallier l'incapacité des algorithmes d'apprentissage à utiliser toutes les données dans un délai raisonnable, la parallélisation de traitement distribue les algorithmes d'apprentissage en répartissant le processus d'apprentissage sur plusieurs ordinateurs ou processeurs (Peteiro-Barral et Guijarro-Berdiñas, 2013). L'apprentissage automatique distribué peut atteindre non seulement l'efficacité par le chargement de données en parallèle, mais aussi la tolérance aux pannes en répliquant les données sur les machines. De plus, l'utilisation de différents processus d'apprentissage pour former plusieurs classifieurs à partir des ensembles de données distribués augmente la possibilité d'atteindre une plus grande précision. Un autre avantage des algorithmes distribués ce qu'ils peuvent être intégrés en d'autres parties de la gestion des données. Cependant, concevoir et mettre en œuvre des algorithmes parallèles efficaces est extrêmement difficile (Low *et al.*, 2012).

Certains algorithmes d'apprentissage, tels que la recherche par force brute et les algorithmes génétiques, sont trivialement parallèles, et donc la parallélisation peut fournir des améliorations de performance massives. Par conséquent, les chercheurs ont développé des techniques et des outils pour paralléliser l'apprentissage automatique.

2.7 Techniques de tolérance de bruit de classe

Lorsque des informations sont disponibles sur le bruit de classe ou ses conséquences sur l'apprentissage, il devient possible de concevoir des modèles qui tiennent compte du bruit de classe. Généralement, on peut apprendre un modèle de bruit de classe simultanément avec un classifieur, ce qui déconnecte les deux composants du processus de génération de données et améliore le classifieur résultant. En un mot, le classifieur

résultant apprend à classer les instances en fonction de leur véritable classe inconnue. D'autres approches consistent à modifier l'algorithme d'apprentissage pour réduire l'influence du bruit classe. Le nettoyage des données peut également être intégré directement dans l'algorithme d'apprentissage, comme par ex. pour les SVMs. D'autres techniques tolérantes au bruit de classe, qui peuvent tolérer le bruit de classe en les modélisant. Il existe aussi les méthodes probabilistes, ainsi que les méthodes fondées sur des modèles.

2.7.1 Approches probabilistes

Dans la communauté probabiliste, certains auteurs affirment qu'il est impossible de détecter le bruit des étiquettes sans faire d'hypothèses (Swartz *et al.*, 2004a; Joseph *et al.*, 1995; Gaba et Winkler, 1992) Par exemple, (Gaba et Winkler, 1992) rapporte un modèle probabiliste prenant en compte le bruit de classe pour lequel il existe un nombre infini de solutions de maximum de vraisemblance. En fait, pour de tels problèmes d'identifiabilité (Swartz *et al.*, 2004b), des informations préalables sont nécessaires pour rompre les liens. Les probabilités bayésiennes sur les probabilités d'erreur d'étiquetage (Joseph *et al.*, 1995) peuvent être utilisées, mais elles doivent être choisies avec soin, car les résultats obtenus dépendent de la qualité de la distribution a priori (Ladouceur *et al.*, 2007). Bêta Priors (Gaba et Winkler, 1992; Joseph *et al.*, 1995; Daniel Paulino *et al.*, 2003) et Dirichlet Priors (Ruiz *et al.*, 2008; Liu *et al.*, 2009) sont des choix communs ; Des méthodes bayésiennes existent pour la régression logistique (Daniel Paulino *et al.*, 2003; Jorge Alberto *et al.*, 2004; McInturff *et al.*, 2004; Gerlach et Stamey, 2007), les modèles de Markov cachés (García-Zattera *et al.*, 2010) et les modèles graphiques (Kaster *et al.*, 2010). D'autres approches (Rekaya *et al.*, 2001; Robbins *et al.*, 2006; Hernandez-Lobato *et al.*, 2011) sont basées sur des indicateurs qui indiquent si une classe donnée a été retournée.

Des méthodes fréquentiels existent également pour traiter le bruit de classe. Une solution simple consiste à utiliser un mélange de distribution normale et une distribution «anormale» (Mansour et Parnas, 1998). Ce dernier est généralement une distribution uniforme sur le domaine d'instance, mais d'autres choix sont possibles. Lawrence et al. (Lawrence et Schólkopf, 2001) ont proposé un modèle génératif probabiliste pour traiter le bruit de classe. D'abord, les vrais labels Y sont tirés d'une distribution a priori p(Y). Ensuite, les valeurs caractéristiques sont tirées de la distribution conditionnelle p (X|Y) et les classes observées Y de la distribution conditionnelle p (Y |Y). Les valeurs des attributs et les classes observées sont connues, mais les vraies classes (cachées) doivent être éliminés des données. Par exemple, Lawrence et al. (Lawrence et Schólkopf, 2001)

dérivent l'algorithme EM pour apprendre un discriminant Fisher tout en déduisant les vraies classes. Cela a été étendu aux distributions de classes conditionnelles non gaussiennes (Li *et al.*, 2007), aux problèmes multi-classes (Bootkrajang et Kabán, 2011), aux données séquentielles (Frénay *et al.*, 2011) et à l'estimation mutuelle des informations. Des classifieurs discriminants équipés de probabilités de bruit de classe ont également été mis au point dans (Bootkrajang et Kaban, 2012, 2013). Le traitement basé sur le modèle du bruit de classe est assez intuitif, cependant une analyse théorique des algorithmes résultants est encore à ses balbutiements (Kaban et Bootkrajang, 2013). D'autre part, les garanties de minimisation des risques sous le bruit de la classe aléatoire (Natarajan *et al.*, 2013) conduisent à différentes procédures pour modifier une fonction de perte de donnée et obtenir de nouveaux algorithmes tolérants au bruit.

Le regroupement peut être utilisé pour détecter des cas mal étiquetés (Rebbapragada et Brodley, 2007; Bouveyron et Girard, 2009), sous l'hypothèse que les instances dont l'étiquette (classe) n'est pas compatible avec l'étiquette des instances voisines sont susceptibles d'être mal étiquetées. Une autre solution consiste à utiliser des fonctions de croyance (Denoeux, 2000), car elles permettent de modéliser la confiance de l'expert dans ses étiquettes. Lorsque cette information n'est pas fournie par l'expert, plusieurs approches ont été proposées pour inférer les croyances directement à partir des données (Denoeux, 2000; Younes *et al.*, 2010).

Certaines approches, pour traiter le bruit, utilisent une technique de modélisation du bruit probabiliste pour incorporer simultanément un modèle de bruit dans la construction du modèle. Ceci est conçu pour modéliser la distribution sans bruit en découplant le processus de génération de données et de bruit. Quelques exemples incluent des approches bayésiennes, en incorporant la distribution de probabilité antérieure du bruit (Du et Cai, 2015) ; approches de sous-population en utilisant un modèle de mélange de distributions normales et de bruit (Scott, 2015) ; et des fonctions de croyance pour modéliser le degré de certitude dans une étiquette de classe (Ma *et al.*, 2016).

2.7.2 Approches fondées sur des modèles

Plusieurs autres modèles non probabilistes ont été modifiés pour devenir tolérants au bruit. Par exemple, on peut empêcher les instances de prendre des poids trop importants dans les réseaux neuronaux (Swartz *et al.*, 2004b; Ladouceur *et al.*, 2007; Daniel Paulino *et al.*, 2003), les SVMs (Ruiz *et al.*, 2008; Liu *et al.*, 2009) et les ensembles obtenus avec boosting (Jorge Alberto *et al.*, 2004; McInturff *et al.*, 2004; Gerlach et Stamey, 2007; Zhang *et al.*, 2018). Des modèles robustes (García-Zattera *et al.*, 2010; Kaster *et al.*,

2010; Rekaya *et al.*, 2001; Robbins *et al.*, 2006; Hernandez-Lobato *et al.*, 2011; Mansour et Parnas, 1998) peuvent également être utilisées et sont théoriquement moins sensibles au bruit.

Contrairement à l'approche de filtrage, une approche fondée sur un modèle est plus transparente, elle est fondée sur des principes car elle inclut un modèle explicite du processus de mal-étiquetage en tant que partie intégrante de la modélisation des données. À notre connaissance, jusqu'à présent il n'y a eu que peu de tentatives pour aborder le problème en modélisant explicitement le processus de bruit.

(Steven et Hirsh, 1992) suggèrent d'apprendre à partir de données bruitées, en incorporant une connaissance préalable du processus de bruit. La probabilité a posteriori de chaque hypothèse dans l'espace d'hypothèses recherché est calculée afin d'élaguer les mauvaises hypothèses. Ils ont trouvé empiriquement que leur approche maximale a posteriori (MAP) est supérieure à l'algorithme C4.5.

(Chittineni, 1982) a incorporé les paramètres de bruit de classe dans sa fonction de vraisemblance et il a dérivé un estimateur du maximum de vraisemblance pour estimer les paramètres de classification erronées à partir des données étiquetées et non étiquetées. Il a également introduit un modèle simple pour identifier les instances mal-étiquetées en termes de seuils sur les fonctions discriminantes linéaires pour les deux cas deux-classes et plusieurs-classes. Son approche est plutôt spécialisée dans l'apprentissage semi-supervisé.

(Lawrence et Schólkopf, 2001) ont incorporé un modèle de bruit probabiliste nommé Kernel Fisher Discriminant pour la classification binaire. En supposant que les distributions des classes de données sont gaussiennes, ils ont empiriquement montré que la précision de la classification s'améliore en cas du modèle qui ne contient aucun bruit de classe. Basé sur le même modèle, (Li *et al.*, 2001) ont effectué des expériences approfondies sur des ensembles de données plus complexes. L'extension du modèle multi-classes a également motivé le développement récent d'un modèle de Markov caché tolérant au bruit pour améliorer la segmentation (Frénay *et al.*, 2011).

Tous ces travaux démontrent le grand potentiel et la flexibilité d'une approche basée sur un modèle. Les auteurs dans (A *et al.*, 1998) ont donné un fondement à un modèle statistique pour un problème de classification binaire mais n'apporte aucune solution algorithmique pour l'apprentissage des paramètres de bruit de classe. Plus récemment, (Raykar *et al.*, 2010) ont proposé un algorithme EM pour apprendre la régression logistique d'un modèle similaire à celle discutée dans (A *et al.*, 1998), pour des données avec plusieurs instances de bruit de classe. (Amini et Gallinari, 2005) utilisent la modélisation du bruit afin d'aborder l'apprentissage semi-supervisé. Ils affectent de manière aléatoire des étiquettes à un ensemble d'apprentissage non étiqueté et utilisent le modèle de bruit

pour récupérer les véritables étiquettes afin d'agrandir l'ensemble de données étiqueté. (Krithara *et al.*, 2008) ont ensuite utilisé le cadre de (Amini et Gallinari, 2005) pour étendre l'analyse sémantique probabiliste latente (Hofmann, 2001) afin d'intégrer une erreur mal-classée. Les résultats empiriques montrent que leur méthode est meilleure par rapport aux méthodes précédentes qui utilisent un modèle sans modèle d'erreur de mal-étiquetage.

Certains modifications peuvent être incorporés dans un algorithme d'apprentissage afin d'augmenter la tolérance au bruit. Par exemple, les SVM (Ghosh *et al.*, 2015; LI et Chen, 2015; Yang *et al.*, 2015), les réseaux de neurones (Khardon et Wachman, 2007) et les méthodes ensemblistes (Perez *et al.*, 2016) peuvent incorporer des mécanismes de tolérance au bruit. L'apprentissage à partir des données avec bruit de classe est un problème important dans la classification, car les classes bruitées peuvent diminuer les performances prédictives et l'induction difficile du modèle. De nombreuses techniques traitant le bruit de classe ont été proposées dans la littérature, à la fois d'un point de vue théorique et pratique. Ces techniques comprennent des techniques tolérantes au bruit ainsi que des méthodes de nettoyage des données.

2.8 Conclusion

Le bruit de classe est un phénomène complexe qui peut avoir de nombreuses conséquences négatives sur la performance de la classification. L'élimination du bruit de classe est devenue une tâche de plus en plus courante, d'où la nécessité de développer un outil ou proposer une approche permettant la détection et l'élimination de bruit de classe d'une manière fiable toute en gardant une bonne qualité de classification.

CHAPITRE

3 RÉDUCTION DE DIMENSION PAR LA SÉLECTION D'ATTRIBUTS

Sommaire

3.1 Introduction

Dans ce chapitre nous allons défini la méthode de sélection d'attributs, ensuite nous allons présenté les différents approches proposées dans la littérature. Les ensembles de données standard ont une grande dimension de colonne par rapport à dimension de la ligne. Les recherches menées au cours des dernières décennies ont ouvert la voie à des algorithmes d'exploration de données efficaces qui fonctionnent bien avec des ensembles de données standard. Mais la plupart des modèles de classification traditionnels se comportent mal lors de la manipulation de jeux de données de grande dimension. L'une des principales raisons des performances médiocres sur des jeux de données de grande dimension est le compromis entre la précision et la complexité de calcul.

Les données sont non seulement volumineuses, mais aussi complexes et variées, les techniques d'apprentissage automatique sont confrontées à un défi majeur car il est difficile de traiter un grand nombre de caractéristiques (Jain et Zongker, 1997). Le problème de mise à l'échelle apparaît dans tout algorithme d'exploration de données traditionnel lorsque la taille des données augmente au-delà de leur capacité, ce qui nuit aux performances et à l'efficacité. Ce problème peut également affecter négativement certains

autres aspects tels que les exigences de stockage excessifs, l'augmentation de la complexité temporelle et, enfin, la précision généralisée due au sur-ajustement et au bruit.

Avec cette augmentation de la dimensionnalité des données, la probabilité d'avoir des attributs non pertinents, redondants et bruyants augmente (Chang *et al.*, 2014). Une méthode courante pour réduire la dimensionnalité des données à analyser consiste à réduire le nombre de caractéristiques ou de variables à un nombre plus gérable tout en gardant la performance du modèle.

En effet, l'utilisation d'une méthode de sélection de caractéristiques adéquate permet d'éviter le sur-ajustement et d'améliorer les performances du modèle, en fournissant des modèles plus rapides et plus rentables avec une meilleure compréhension des processus de génération des données (Saeys *et al.*, 2007).

Traditionnellement, les méthodes de sélection d'attributs ont été conçues pour fonctionner dans un environnement informatique centralisé. Cependant, au cours des dernières années, de nombreuses méthodes distribuées ont été développées au lieu des approches centralisées. La première raison est que, avec l'avènement des technologies de réseau, la taille des ensembles de données a augmenté dans tous les domaines d'application, mais que les données sont souvent réparties entre les frontières institutionnelles, géographiques et organisationnelles, notamment en matière de confidentialité. Il n'est pas économique ou légal de les rassembler dans un seul endroit. Deuxièmement, lorsqu'il s'agit de grandes quantités de données, la plupart des algorithmes de sélection d'attributs existants ne sont pas bien adaptés et leur efficacité peut se détériorer considérablement au point de devenir inapplicable. Par conséquent, une solution possible pourrait être de distribuer les données, d'exécuter une méthode de sélection d'attributs sur chaque sous-ensemble de données, puis de combiner les résultats. Les données peuvent être distribuées horizontalement ou verticalement.

Les techniques de classification doivent s'adapter à des volumes de données toujours plus importants. En effet, l'apparition des grandes bases de données dans le domaine de l'apprentissage et les systèmes de fouille de données "Data Mining", ainsi que l'explosion progressive de volume de données tant en nombre d'individus qu'en nombre d'attributs a exigé une réduction de dimension, avant d'entamer la tâche de classification des données. Les principaux objectifs de la réduction de dimension sont :

— faciliter la visualisation et la compréhension des données,
— réduire l'espace de stockage nécessaire,
— réduire le temps d'apprentissage et d'utilisation,

— identifier les facteurs pertinents.

La réduction de dimension est la technique de projection d'un ensemble de points de données d'entrée sur un espace dimensionnel plus petit. En d'autres termes, les données sont représentées sur un sous-espace d'une dimensionnalité plus faible avec une perte d'informations minimale. En effet, les méthodes de réduction de la dimension des attributs permettent de rendre l'ensemble des données plus représentatif du problème. Elles permettent généralement de réduire, non seulement, l'espace de stockage nécessaire pour ces données, mais aussi le temps d'apprentissage et d'exploitation des modèles de classification. Les techniques de réduction de la dimension jouent un rôle essentiel surtout pour les bases de données à grande dimensionnalité. Il existe deux approches de réduction de la dimension, à savoir : l'extraction d'attributs et la sélection d'attributs (Wei et Stephen, 2007).

Les méthodes d'extraction d'attributs consistent à construire de nouveaux attributs à partir de l'ensemble des variables originales, généralement plus petit, tout en conservant autant que possible la structure originale des données.

Les méthodes de sélection d'attributs quant à elles permettent de ne conserver qu'un sous-ensemble de variables pertinent selon un critère de performance. Ces méthodes permettent une caractérisation plus rapide des données. La sélection d'attributs ne modifie pas la représentation originale des données. En effet, les attributs sélectionnés gardent leur sémantique de départ et peuvent alors être interprétés plus facilement par l'utilisateur.

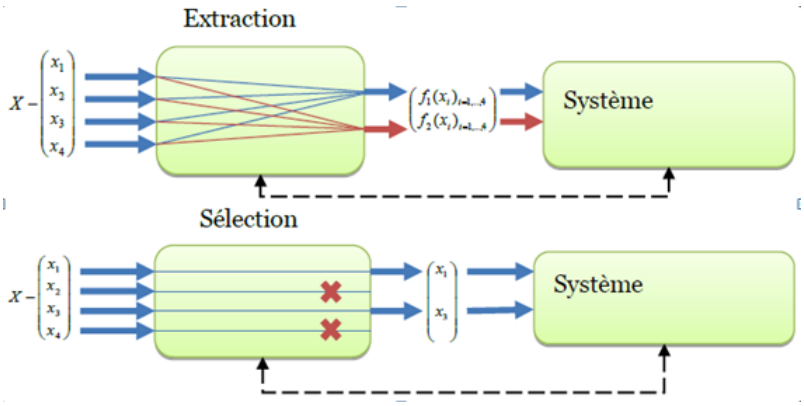

Figure 3.1 – Selection et extraction d'attributs

Dans ce contexte, nous nous intéressons à l'approche de sélection des attributs pour la classification supervisée.

3.2 Sélection d'attribut

3.2.1 Définition et concept

Les performances d'un classifieur dépendent fortement de la qualité de représentation des données à traiter, ce qui implique généralement l'obligation de représenter ces dernières au moyen d'un nombre élevé d'attributs représentatifs. Il est alors fréquent qu'une partie de celles-ci ne contienne que des informations non pertinentes, redondantes ou inutiles à la tâche de classification, rendant cette dernière plus complexe. Il est donc nécessaire, lors de la construction d'un système de classification, de limiter le nombre d'attributs pris en compte, de manière à optimiser ses performances. C'est exactement ce que fait le processus de « sélection d'attributs » qui a pour but de filtrer le vecteur des attributs de manière à en extraire l'information discriminante et pertinente améliorant la qualité du système (Bermejo *et al.*, 2012).

La sélection d'attributs est l'une des priorités dans le domaine de l'exploration de données. Pour un jeu de données, la sélection d'attributs peut être généralisée en tant que processus de sélection d'un sous-ensemble d'attributs à utiliser dans une analyse ultérieure des données. Ce sous-ensemble sélectionné d'attributs devrait capturer le maximum d'informations présentes dans le jeu de données, c'est-à-dire que le sous-ensemble d'attributs sélectionné devrait contenir les entités les plus importantes pour la construction du modèle. La sélection des d'attributs est particulièrement importante dans les jeux de données de grande dimension car elle réduit la dimensionnalité et annule ainsi les effets de la malédiction de la dimensionnalité. En outre, dans de nombreux cas réels, la sélection d'attributs est très importante pour identifier le comportement et les performances du système. En particulier dans les applications biomédicales. Dans un problème de classification de maladie dans une étude génomique, par exemple, des techniques de sélection d'attributs peuvent identifier les gènes qui différencient les cellules malades et les cellules saines. Cela aide non seulement l'analyste de données à réduire la dimension des données, mais constitue également une avancée majeure pour les biologistes afin de comprendre le système biologique et d'identifier les gènes qui déclenchent la maladie. Dans ce contexte, la sélection d'attribut constitue une étape importante dans le prétraitement des données de grande dimension, l'objectif de la sélection est de trouver un sous-ensemble optimal d'attributs pertinents, à partir d'un ensemble de variables original, selon un certain critère de performance. De plus cet ensemble doit chercher à éviter les attributs redondants.

En effet les principales motivations derrière le choix d'utiliser la sélection d'attributs sont :

— l'amélioration de la classification en éliminant les attributs ayant des informations inutiles. En effet, Dans de nombreuses situations, la sélection d'attributs peut également améliorer la précision de la prédiction. Même les algorithmes d'apprentissage les plus puissants ne peuvent pas surmonter la présence d'un grand nombre d'attributs non pertinents ou faiblement pertinents. D'autre part, une fois qu'un petit ensemble de bons attributs a été trouvé, même des algorithmes d'apprentissage très simples peuvent donner de bonnes performances. Ainsi, dans de telles situations, une première étape de sélection d'attributs peut améliorer considérablement la précision.

— l'aboutissement de l'objectif fixé, à savoir la précision et la rapidité de l'apprentissage ou bien encore l'applicabilité du classifieur proposé. Effectivement la sélection d'attributs permet d'économiser le coût de mesurer des attributs non sélectionnées. Une fois que nous avons trouvé un petit ensemble d'attributs qui permet une bonne prédiction des étiquettes (classes), nous n'avons plus besoin de mesurer le reste d'attributs. Ainsi, dans l'étape de la prédiction, nous avons seulement besoin de mesurer quelques attributs pour chaque instance. Imaginez que nous voulons prédire si un patient a une maladie spécifique en utilisant les résultats des contrôles médicaux. Il y a un grand nombre de contrôles médicaux possibles qui pourraient être prédictifs ; nous allons dire qu'il y a 1000 contrôles potentiels et que chacun d'eux coûte dix dollars à effectuer. Si nous pouvons trouver un sous-ensemble de seulement 10 attributs qui permet de bonnes performances, il économise beaucoup d'argent, et peut transformer le tout d'un infaisable en une réalisable procédure.

— l'interopérabilité des résultats en conservant la sémantique des attributs de départ ce qui présente un autre avantage de la sélection d'attributs. En effet, l'identité des attributs sélectionnés peut donner un aperçu de la nature du problème. C'est important puisque dans de nombreux cas, la capacité de souligner les caractéristiques les plus informatives est plus importante que la capacité de faire une bonne prédiction. Imaginez que nous sommes en train d'essayer de prédire si une personne a un type spécifique de cancer en utilisant des données d'expression génique. Bien que nous puissions savoir si l'individu est malade ou non par d'autres moyens, l'identité des gènes informatifs pour la prédiction peut nous donner une idée du mécanisme de la maladie impliquée, et aider à développer des médicaments.

— La réduction de la complexité des algorithmes ainsi que le temps de calcul. Ce qui facilite l'étape d'apprentissage. De nombreux algorithmes d'apprentissage populaires devenaient intraitables en termes de calcul en présence d'un grand nombre d'attributs, dans l'étape d'apprentissage ainsi que dans l'étape de prédiction. Un prétraitement de sélection d'attributs peut résoudre le problème.

3.2.2 Notion de pertinence d'un attribut

Une variable pertinente est une variable telle que sa suppression entraîne une détérioration des performances. Plus formelle, les auteurs dans (Kohavi et John, 1997) classifient les attributs en trois catégories disjoints, nommés les attributs fortement pertinents, les attributs faiblement pertinents et les attributs non pertinentes. Les attributs fortement pertinents sont indispensables et devraient figurer dans tout sous-ensemble optimal sélectionné, car leur absence conduit à une erreur de reconnaissance de la fonction cible. D'autre part, la faible pertinence indique que l'attribut en question n'est pas toujours important, mais il peut devenir nécessaire pour un sous-ensemble optimal dans certaines conditions. Alors que les attributs non pertinente, réfèrent à ceux qui sont peu significatives ou corrélées et qui doivent être éliminé.

3.2.3 Notion de redondance d'un attribut

La notion de redondance d'attributs est généralement exprimée en termes de corrélation entre attributs (Yu et Liu, 2004). Il est largement accepté que deux attributs sont redondants les uns aux autres si leurs valeurs sont complètement corrélées. Cette définition ne se généralise pas directement pour un sous-ensemble d'attributs. En effet il peut ne pas être si facile de déterminer la redondance des attributs lorsqu'un attribut est corrélé avec un ensemble d'attributs. Selon (Yu et Liu, 2004), un attribut redondant devrait donc être supprimé si il est non pertinents ou faiblement pertinent et si il comporte une couverture de Markov dans l'ensemble actuel des attributs.

Intuitivement, dans le pire des cas, l'information pertinente est noyée parmi de nombreux attributs qui expriment tous une même idée (redondante) sans intérêt pour l'utilisateur. Cette situation extrême risque de conduire à une classification sans réel intérêt pour l'utilisateur.

3.2.4 Procédure de la sélection d'attributs

La problématique de filtrage de données ou de sélection d'attributs pour la classification supervisée représente un axe de recherche très actif dans le domaine du Datamining (fouille de données) et en optimisation. Une procédure de sélection d'attributs est généralement composée de quatre étapes illustrées par la Figure 3.2

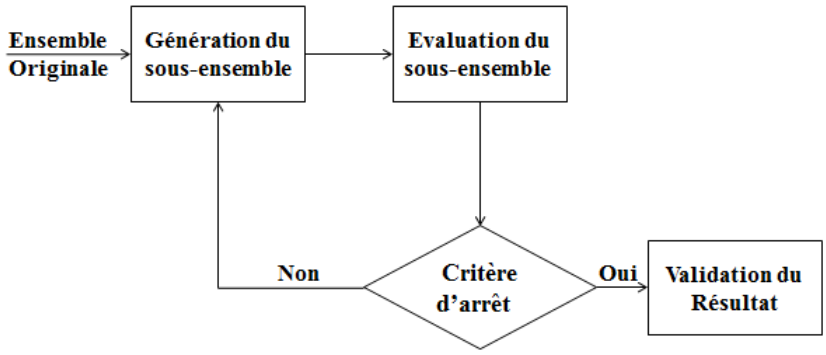

Figure 3.2 – Procédure de sélection d'attributs

Étape 1 : la procédure de génération (elle s'applique uniquement aux algorithmes de recherche de sous-ensembles).

Étape 2 : la fonction d'évaluation.

Étape 3 : le critère d'arrêt.

Étape 4 : la procédure de validation.

La procédure de génération : La procédure de génération permet, à chaque itération, de générer un sous-ensemble d'attributs qui sera évalué lors de la seconde étape de la procédure de sélection. Cette procédure peut être initiée avec un ensemble vide d'attributs ou avec l'ensemble de tous les attributs ou encore avec un sous-ensemble d'attributs choisi aléatoirement. Dans les deux premiers cas, les attributs sont itérativement ajoutés (Forward selection) ou retirés (Backward selection). Dans le troisième cas, on ajoute/retiré des attributs (tout comme dans les deux premiers cas), tandis que le nouveau sous-ensemble d'attributs est créé de manière aléatoire à chaque itération (Random generation).

En général, on ne connaît pas le nombre optimal m de variables à sélectionner. Ce nombre dépendra de la taille et de la qualité de la base d'apprentissage, la quantité et la qualité d'information disponible et de la règle de décision utilisée par le modèle. En effet, pour un ensemble initial des attributs de dimension d, le nombre total de sous-ensembles candidats qui peuvent être générés par la procédure de génération est 2d où 2 représente

deux choix : sélectionner ou ne pas sélectionner une variable. Ce nombre est d'autant plus élevé que la dimension d est grande. En conséquence, trois grandes approches de génération d'attributs ont été proposées dans la littérature : la génération complète, la génération aléatoire et la génération séquentielle (Yu et Liu, 2004).

La génération complète : Dans la procédure de génération complète, une recherche complète du sous-ensemble d'attributs optimal (selon la fonction d'évaluation utilisée) est effectuée sur tout l'espace des solutions possibles. Plusieurs procédures de recherche heuristique sont proposées afin de réduire l'espace de recherche sans pour autant compromettre les chances de trouver le sous-ensemble optimal.

La génération aléatoire (heuristique) : Contrairement aux procédures de génération complète, Cette procédure de génération aléatoire n'évalue pas toutes les solutions possibles, elle parcourt aléatoirement toutes les possibilités de sous-ensembles candidats, permettant ainsi de ne pas arrêter la recherche lorsque la fonction d'évaluation d'un sous-ensemble atteint un optimum local. Un nombre maximal d'itérations est imposé afin de limiter le temps de calcul. L'avantage de cette procédure est qu'elle ne nécessite pas l'utilisation de fonction d'évaluation monotone. D'autre part, la complexité de calcul des méthodes basées sur une génération aléatoire est quadratique contrairement aux méthodes de génération complète dont la complexité est exponentielle vis-à-vis de la dimension initiale de l'espace d'attributs. Algorithm Genetic (AG) sont les méthodes de génération aléatoire les plus utilisées.

La génération séquentielle : Le principe est d'ajouter ou de supprimer un ou plusieurs attributs au fil des itérations. C'est-à-dire à chaque itération de cette procédure, on considère à nouveau tout l'ensemble des attributs restants pour l'étape de la sélection. On distingue alors deux approches de génération séquentielles (Yu et Liu, 2004) :

Recherche vers l'avant : l'approche de type "Forward" ou Ascendante part d'un ensemble vide d'attributs auquel, à chaque itération un ou plusieurs attributs sont ajoutés.

Recherche vers l'arrière : l'approche de type "Backward" ou Descendante est l'approche inverse. Elle part de l'ensemble total des attributs et à chaque itération permet de supprimer un ou plusieurs attributs.
Comme ils n'explorent pas tous les sous-ensembles possibles d'attributs et ne permettent

pas de retour en arrière durant la recherche, Les algorithmes utilisant ces approches de génération sont donc connus par leurs simplicité de mise en œuvre et leur rapidité.

3.2.4.1 La fonction d'évaluation

La stratégie d'évaluation est la manière dont les sous-ensembles d'attributs générés à l'étape précédente sont évalués est le facteur de différenciation le plus important parmi les algorithmes de sélection d'attributs pour l'apprentissage automatique. Les techniques de sélection d'attributs sont divisibles en trois catégories, selon la manière dont elles interagissent avec le classifieur (Yu et Liu, 2004).

Les approches **"Filter"** fonctionnent indépendamment de tout algorithme d'apprentissage, les attributs non pertinents sont éliminés avant l'apprentissage. Ces algorithmes utilisent des heuristiques basées sur les caractéristiques générales des données pour évaluer les sous-ensembles d'attributs. Ces méthodes sont rapides, plus générales et moins coûteuses en temps de calcul, ce qui leur permet de gérer plus facilement des bases de données de très grandes dimensions. Cependant, leur indépendance de l'étape de classification ne permet pas de garantir que le meilleur taux de classification soit obtenu dans l'espace retenu.

Wrappers, contrairement aux approches filtre qui ignorent totalement l'influence des variables sélectionnées sur la performance de l'algorithme d'apprentissage, les approches "enveloppantes" impliquent un algorithme d'apprentissage comme une boîte noire et consiste à utiliser sa performance de prédiction pour évaluer l'utilité relative des sous-ensembles de variables. En d'autres termes, l'algorithme de sélection de caractéristiques utilise la méthode d'apprentissage comme sous-programme. Elles utilisent le taux d'erreur de classification comme critère d'évaluation. En effet, dans ces mesures, l'attribut ou les sous-ensembles d'attributs considérés sont évalués en fonction de la qualité de la classification obtenue en utilisant ces attributs. Ces méthodes permettent d'obtenir de bonnes performances.
Cependant, il existe deux inconvénients principaux qui limitent ces méthodes. Il s'agit de la complexité et le temps nécessaire pour la sélection. En effet, l'utilisation de telles méthodes nécessite pour chaque sous-espace d'attributs candidats d'effectuer la classification, ce qui peut devenir coûteux en temps de calcul, d'autant plus lorsque la dimension de l'espace d'entrée est grande. D'autre part, ces méthodes sont très dépendantes de l'algorithme de classification utilisé comme critère d'évaluation. Ce dernier, s'il est mal adapté, pourrait contribuer à une sélection de mauvais attributs.

L'approche **« embedded »** ou « intégrée » effectue la sélection des attributs dans le processus d'apprentissage. Par conséquent, la recherche d'un sous-ensemble optimal d'attributs est intégrée dans la construction du classifieur et peut être vu comme une recherche dans l'espace combiné des sous-ensembles d'attributs. En effet, le sous-ensemble d'attributs ainsi sélectionnées sera choisi de façon à optimiser le critère d'apprentissage utilisé. Cette approche est capable de capturer les dépendances à un coût de calcul inférieur à "wrappers". Dans l'approche intégrée, le classifieur sert non seulement à évaluer un sous-ensemble candidat mais aussi à guider le mécanisme de sélection. Contrairement aux approches décrites ci-dessus. En effet l'approche « filter » n'intègre pas l'apprentissage dans la sélection des attributs et l'approche « wrapper » utilise un algorithme d'apprentissage pour mesurer la qualité des sous-ensembles d'attributs.

Le tableau 6.1 fournit un résumé des caractéristiques de chaque approche, indiquant les avantages et les inconvénients les plus importants. Notant que les approches de type « filter » sont plus largement utilisées principalement pour leur coût de calcul minime ainsi que pour leur indépendance de tout algorithme de classification.

Méthodes	**Avantages**	**Inconvénients**
Filtre	— Indépendance du classificateur — Coût de calcul inférieur à «wrappers» — Rapide — Adaptable — Bonne capacité de généralisation	— Ignore la dépendance entre les attributs — Ignore l'interaction avec le classifieur
Wrappers	— Interaction avec le classifieur — Capture des dépendances entre attributs	— Calcul coûteux — Risque de sur-apprentissage — Sélection dépendante du classifieur
Embedded	— Interaction avec le classifieur — Coût de calcul inférieur à « wrappers » — Capture des dépendances entre attributs	Sélection dépendante du classifieur

Tableau 3.1 – Caractéristiques des approches de sélection d'attributs

Le nombre optimal d'attributs n'étant pas connu a priori, il est fixé grâce à un critère d'arrêt du processus de sélection. Le choix du critère d'arrêt dépend généralement de la procédure de recherche et/ou du critère d'évaluation. Les critères d'arrêt les plus fréquents sont les suivants :

3.2.4.2 Le critère d'arrêt

Le nombre optimal d'attributs n'étant pas connu a priori, l'utilisation d'une règle pour contrôler la sélection ou l'élimination d'attributs permet d'arrêter la recherche lorsqu'aucun attribut n'est plus suffisamment informatif. En effet un critère d'arrêt est fixé lors de processus de sélection (Yu et Liu, 2004). Les critères d'arrêts les plus fréquents sont :

— Un certain seuil est atteint, on arrête la recherche en atteignant un seuil sur le nombre d'attributs à sélectionner ou sur le nombre d'itérations. Cependant, le nombre d'attributs pertinents à sélectionner est très difficile de le savoir au préalable. De même, un seuil sur un nombre maximal d'itérations peut arrêter la sélection trop tôt ou trop tard.

— L'addition ou suppression de n'importe quel attribut entre deux itérations consécutives ne produit pas un meilleur sous-ensemble, en mesurant par exemple le gain d'information des taux de bonne classification obtenu par les deux sous-ensembles.

— Un sous-ensemble sélectionné est suffisamment bon, autrement dit son taux d'erreur de classification est faible.

— La recherche est accomplie.

Le choix du critère d'arrêt est ainsi un choix délicat qui reste un problème non résolu dans des nombreux algorithmes de sélection.

3.2.4.3 La procédure de validation

La validation ne fait pas partie de la procédure de sélection d'attributs mais elle permet de tester la validité du sous-ensemble d'attributs sélectionnés en effectuant plusieurs tests sur des exemples de données générées artificiellement et/ou sur des données réelles. En effet, l'ensemble des données est généralement divisé en deux sous-ensembles distincts (le sous-ensemble d'apprentissage, le sous ensemble de test). Selon la répartition des données, il existe différentes approches de validation :

La validation ne fait pas partie de la procédure de sélection d'attributs. Cependant,

elle permet de tester la validité du sous-ensemble d'attributs sélectionnés en effectuant plusieurs tests sur des exemples de données générées (Yu et Liu, 2004). Si on connait les attributs pertinents à l'avance, la façon la plus simple pour valider les attributs sélectionnés est de les comparer avec les attributs connus. Réellement, il n'existe pas de connaissances a priori dans les applications. Par conséquent, d'autres méthodes peut être utilisés par exemple on peut surveiller le changement des performances par rapport aux changements des attributs en calculant le taux d'erreur de classifieur sur l'ensemble d'attributs (si on considère le taux d'erreur de classification comme un indicateur de performance).

3.3 Approches de sélection d'attributs

Une étude comparative abordant : le nombre d'instances / nombre des attributs, le nombre de classe, les données bruyantes ainsi que le nombre des attributs non pertinents et redondants, pourrait être inapprochable et, par conséquent, la plupart des études comparatives sont centrées sur un seul problème à résoudre. Ainsi, par exemple, (Forman, 2003) a présenté une comparaison empirique de douze méthodes de sélection d'attributs évaluées sur un référentiel de 229 instances de problèmes de classification de texte ; une autre étude comparative (Sun *et al.*, 2006) est utilisée pour la détection des cancers du sein dans les mammographies. D'autres travaux sont consacrés à une approche spécifique (Zhang *et al.*, 2008) dans laquelle une étude expérimentale de huit algorithmes de sélection d'attributs basés sur des informations mutuelles typiques sur de trente-trois jeux de données est présentée ; ou une évaluation de la capacité de l'algorithme survival ReliefF (sReliefF) et d'une approche sRelief F adaptée pour sélectionner correctement la paire causative des attributs (Beretta et Santaniello, 2011).

De même, il existe des travaux qui examinent différentes méthodes de sélection d'attributs pour obtenir de bons résultats de performance en utilisant un classifieur spécifique (Bayes naïfs dans (Zhang *et al.*, 2009), C4.5 ou « review » sur SVM dans (Victo Sudha et Raj, 2011)). Avec les défis présentés par les données, plusieurs travaux tentent de faire face au problème de haute dimensionnalité, dans les deux dimensions (instances et attributs) ou l'un d'entre eux ; la plupart de ces études abordent également le problème de la multi-class (Bontempi et Meyer, 2010; Aliferis *et al.*, 2010; Hua *et al.*, 2009). De plus, la majorité des ensembles de données réels actuels présentent également des données bruitées, mais aucune étude comparative de sélection d'attributs portant sur ce problème complexe n'a été trouvée dans la littérature, bien que certains travaux inté-

ressants aient été proposés, par exemple (Thongkam *et al.*, 2008; Yang et Hu, 2008). En mettant l'accent sur les méthodes non linéaires, on peut mentionner l'étude de (Guyon *et al.*, 2005).

Enfin, d'un point de vue théorique, (Liu et Yu, 2005) ont présenté une étude des méthodes de sélection d'attributs, fournissant des lignes directrices pour la sélection des algorithmes de sélection d'attributs, ouvrant la voie à la création d'un système intégré de sélection intelligente. Une étude faite par (Dash et Liu, 2003a) présente 32 des méthodes différents groupées par le type de génération de procédures et la fonction d'évaluation utilisée par ces méthodes. Une méthode hybride a également été proposée (Bermejo *et al.*, 2012). Cette méthode utilise une méthode de filtrage lors du premier passage pour supprimer les attributs non pertinents, puis une méthode Wrapper pour un classifieur afin de réduire davantage l'ensemble d'attributs.

3.4 Approches de sélection d'attributs pour les grands ensembles de données

Bien que l'apprentissage distribué soit un domaine relativement nouveau, il attire de plus en plus d'attention depuis sa création. Il existe dans la littérature plusieurs travaux qui traitent les ensembles de données qui sont trop grands pour un apprentissage automatique en termes d'échantillons. (Chan et Stolfo, 1993) proposent plusieurs stratégies de méta-apprentissage pour intégrer des classifieurs appris indépendamment par le même apprenant dans un environnement informatique parallèle et distribué. Les expériences démontrent que l'apprentissage parallèle par un méta-apprentissage peut atteindre la même précision de prédiction avec moins de temps et d'espace mémoire que l'apprentissage purement en série. D'autres auteurs (Ananthanarayana *et al.*, 2000) ont conçu une architecture distribuée partitionnée et un algorithme efficace d'extraction de règles d'association distribuée basé sur l'arbre de modèles appelé PC-tree. Dans (Tsoumakas et Vlahavas, 2002), une nouvelle stratégie de combinaison de classifieurs est présentée. Il évolue efficacement et atteint à la fois une haute précision prédictive et une traçabilité des problèmes de haute complexité. (Das *et al.*, 2010) ont développé un algorithme local de préservation de la confidentialité pour la sélection des attributs dans un environnement peer-to-peer étendu. Bien que ce ne soit pas très populaire, il existe d'autres approches qui distribuent les données par les attributs. Dans ce travail (Skillicorn et McConnell, 2008), les auteurs ont décrit une nouvelle approche d'ensemble, dans laquelle les données sont partitionnées par les attributs. Les résultats montrent que cette technique est

simple, prédit presque aussi bien qu'une approche centralisée, distribue bien le calcul et l'accès aux données, et permet à chaque site local de garder ses données brutes privées. De plus, (Banerjee et Chakravarty, 2011) ont proposé une méthode distribuée de préservation de la confidentialité pour effectuer la sélection des attributs qui gère à la fois le partitionnement horizontal et vertical des données, dont l'efficacité a été démontrée pour les jeux de données réels, y compris les séries chronologiques. D'autres travaux (Peralta *et al.*, 2015) présentent une méthode de sélection d'attributs basée sur le calcul évolutif qui utilise le paradigme MapReduce pour obtenir des sous-ensembles d'attributs à partir de grands ensembles de données. Dans (Bolon-Canedo *et al.*, 2014), les auteurs ont présenté une méthodologie de distribution verticale des données combinant des sous-ensembles partiels basés sur des améliorations de la précision de la classification. Bien que les expériences aient montré que le temps d'exécution était considérablement raccourci alors que les performances étaient maintenues ou même améliorées par rapport aux algorithmes standards appliqués aux jeux de données non partitionnés. L'inconvénient de cette méthodologie était sa dépendance au classifieur utilisé. Afin de résoudre ce problème, un nouveau cadre de distribution du processus de sélection d'attributs (Bolon-Canedo *et al.*, 2015; Moran-Fernandez *et al.*, 2015) a été proposé, qui a procédé à une fusion pour mettre à jour le sous-ensemble final en fonction de la complexité théorique de ces attributs, en utilisant des mesures de complexité des données au lieu de l'erreur de classification. Plus récemment, les mêmes auteurs proposent une approche distribuée basée sur des mesures de complexité des données (Moran-Fernandez *et al.*, 2017), cette méthode a été réalisée à la fois pour la technique horizontale et la technique verticale. Pour combiner les sorties partielles obtenues à partir de l'algorithme de sélection de caractéristiques appliqué à chaque sous-ensemble, un processus de fusion utilisant la complexité théorique est appliqué à ces sous-ensembles d'entités. De telles mesures fournissent une base pour analyser les performances des classifieurs au-delà des estimations des taux d'erreur. (Lorena *et al.*, 2012) ont étudié la capacité des mesures de complexité des données pour expliquer la difficulté de la classification des données d'expression génique du cancer, avant et après l'application de la sélection d'attributs, démontrant que cette procédure pouvait réduire l'influence des attributs sur les taux d'erreur. (Macia *et al.*, 2013) ont proposé la caractérisation des ensembles de données en utilisant des mesures de complexité des données, utiles à la fois pour guider la conception expérimentale et pour expliquer le comportement des apprenants. (Luengo et Herrera, 2015) ont présenté une méthode d'extraction automatique pour déterminer les domaines de compétence d'un classifieur en utilisant un ensemble de mesures de complexité des données. Dans tous les travaux susmentionnés, chaque approche présente une certaine

vulnérabilité, que ça soit en termes de précision de la classification, de durée d'exécution ou d'exigences de stockage.

3.5 Extraction de caractéristiques

Les méthodes utilisées pour l'extraction d'attributs sont très variées. Nous rappellerons brièvement les méthodes linéaires principales (ACP, FDA, MDS), puis nous décrirons quelques méthodes non linéaires qui sont largement utilisés.

3.5.1 Méthodes linéaires

Nous rappelons brièvement les principes de trois méthodes classiques d'analyse de données, qui sont le fondement de plusieurs méthodes non linéaires plus récentes.

3.5.1.1 Analyse en Composantes Principales

L'analyse en composantes principales (ACP) - Principal Component Analysis (PCA) -est une ancienne approche très courante de réduction de la dimensionnalité en utilisant la méthode d'extraction de caractéristiques, qui effectue une réduction de dimension par projection des points originaux dans un sous-espace vectoriel de dimension plus réduite. En effet, il crée un ensemble d'entités dérivées ou de sous-espaces linéaires qui sont une combinaison linéaire d'entités existantes, de sorte que la variance maximale des données soit également prise en compte dans le nouveau sous-espace (Pechenizkiy *et al.*, 2006). Cela peut également être vu du point de vue de la création d'un nouveau sous-espace où chaque base de sous-espace est une combinaison linéaire d'entités existantes. L'ACP détermine des axes de projections orthogonaux, qui maximisent la variance expliquée. Dans la base formée par ces axes, les coordonnées ne sont pas corrélées. L'ACP maximise la variance de la projection dans l'espace de caractéristiques, ce qui est équivalent à minimiser l'erreur quadratique moyenne de reconstruction. L'ACP se calcule en diagonalisant la matrice de corrélations, le plus souvent en utilisant une décomposition en valeurs singulières (Singular Value Decomposition (SVD)). Elle est très utilisée car elle est simple à mettre en œuvre. Elle est limitée par son caractère linéaire : il est facile d'imaginer des situations dans lesquelles l'ACP n'apporte aucune information utilisable (par exemple, des données réparties sur un tore en dimension n).

Plusieurs variantes de l'ACP ont été proposées pour faciliter l'interprétation de la projection obtenue ; ainsi, les méthodes varimax, quartimax et equamax s'appuient sur

une rotation orthogonale des axes et les approches oblimin et promax utilisent des rotations obliques. La plus utilisée de ces variantes est sans nul doute la méthode varimax qui effectue une rotation orthogonale des axes pour obtenir des facteurs fortement corrélés à quelques variables et faiblement aux autres ; ainsi, chaque variable est identifiée à un ou à un petit nombre de facteurs et les axes sont facilement interprétables.

3.5.1.2 Analyse Discriminante

L'Analyse Factorielle Discriminante - Fisher Discriminant Analysis (FDA) - appelée aussi analyse discriminante linéaire de Fisher, s'applique lorsque les classes des individus sont connues. Elle consiste à chercher un espace vectoriel de faible dimension qui maximise la variance inter-classe. Une base de cet espace est obtenue en appliquant une Analyse en Composantes Principales sur les centroïdes des différentes classes pondérés par l'effectif de la classe correspondante.

3.5.1.3 Positionnement Multi-Dimensionnel

Dans de nombreux cas, on connaît les distances entre les points d'un ensemble d'apprentissage (on peut utiliser une mesure de similarité plus sophistiquée que la distance euclidienne, comme indiquée dans la section suivante), et on cherche à obtenir une représentation en faible dimension de ces points. La méthode de positionnement multidimensionnelle - Multi-Dimensional Scaling (MDS) - permet de construire cette représentation. L'exemple classique est d'obtenir la carte d'un pays en partant de la connaissance des distances entre chaque paire de villes. L'algorithme MDS est basé sur une recherche de valeurs propres.

3.5.2 Méthodes non linéaires

Les méthodes linéaires reposent (au moins implicitement) sur l'utilisation d'une distance euclidienne (liée au produit scalaire ordinaire). Dans de nombreuses applications, la distance euclidienne n'a pas un grand sens ; elle suppose en particulier que toutes les variables sont comparables entre elles (elles doivent donc avoir été convenablement normalisées). La théorie des espaces de Hilbert permet de définir d'autres produits scalaires, basés sur des fonctions noyaux k(x, y). k est alors une mesure de similarité entre les points de l'ensemble à traiter. Le noyau k défini implicitement une application de l'espace d'origine vers un "espace de caractéristiques" H. La dimension de l'espace H est éventuellement infinie. De nombreuses méthodes statistiques peuvent s'exprimer en ne

recourant qu'à des produits scalaires entre les points à traiter et les exemples d'apprentissage. Si l'on remplace le produit scalaire habituel par un noyau k, on rend la méthode non-linéaire ; c'est le "truc du noyau" - kernel trick -, qui a fait l'objet de nombreuses recherches depuis son introduction dans le cadre de SVM.

3.5.2.1 ACP

La première approche permettant d'appliquer l'ACP au cas de données situées sur une variété non linéaire est d'effectuer des approximations locales : on calcule une ACP pour un groupe de points proches les uns des autres. Cette approche pose le problème de la définition des voisinages et du traitement des nouveaux points rencontrés loin des exemples connus. - pour rendre l'ACP traditionnelle non linéaire. En effet, le calcul de l'ACP ne fait intervenir que des produits scalaires entre les points (pour le calcul de la matrice de covariance) et ne considère jamais les coordonnées d'un point isolé. Si l'on remplace le produit scalaire par un noyau, on calcule donc les composantes principales dans l'espace de caractéristiques H, et on peut ainsi accéder à des corrélations d'ordre supérieur entre les variables observées. Remarquons que l'on peut calculer la projection d'un point ne faisant pas partie de l'ensemble d'apprentissage, ce qui n'est pas le cas de toutes les méthodes de réduction de dimension non linéaires (Pechenizkiy *et al.*, 2006).

3.5.2.2 Isomap

Isomap est une technique de réduction de dimension qui comme la méthode de positionnement multidimensionnel (MDS) part de la connaissance d'une matrice de dissimilarités entre les paires d'individus. Cette fois le but est de trouver une variété (non linéaire) contenant les données. On exploite le fait que pour des points proches, la distance euclidienne est une bonne approximation de la distance géodésique sur la variété. On construit un graphe reliant chaque point à ses k plus proches voisins. Les longueurs des géodésiques sont alors estimées en cherchant la longueur du plus court chemin entre deux points dans le graphe. On peut alors appliquer MDS aux distances obtenues afin d'obtenir un positionnement des points dans un espace de dimension réduite.

3.5.2.3 LLE

La méthode du plongement localement linéaire- Local Linear Embedded (LLE) - a été présenté en même temps qu'Isomap et aborde le même problème par une voie différente. Chaque point est ici caractérisé par sa reconstruction à partir de ses plus

proches voisins. LLE construit une projection vers un espace linéaire de faible dimension préservant le voisinage.

3.5.2.4 Approche neuromimétique

Les réseaux de neurones auto-régressifs - Auto-encoders - sont parfois considérés comme une extension neuronale non linéaire de l'ACP. En effet, ils visent à minimiser l'erreur moyenne de reconstruction d'un individu à partir de sa projection sur un espace de dimension réduite. Ce modèle neuronal comporte trois couches cachées :

- Une couche d'encodage qui extrait une représentation non-linéaire des individus,
- Une couche de compression qui compresse l'information,
- Une couche de décodage qui permet de retrouver la représentation initiale d'un individu.

3.5.3 Conclusion

Au cours de ce chapitre, nous avons rappelé les principes de la sélection de variables et de l'extraction de caractéristiques. Avant de poursuivre, rappelons que cette thèse s'inscrit dans le cadre de l'apprentissage supervisé et que dans ce contexte, nous nous intéressons aux méthodes de réductions de dimensions pour la classification supervisée. Les techniques d'extraction de caractéristiques supervisées sont difficilement utilisables à cause de leur complexité algorithmique lorsque l'on travaille sur de grandes bases données. Il nous semble alors naturel de se focaliser sur les techniques de sélection de variables qui, à l'instar des méthodes d'extraction de caractéristiques, permettent de rester dans l'espace des observations et de ne pas imposer d'effort d'interprétation de nouvelles variables à l'utilisateur. La sélection de variables en apprentissage supervisé est un domaine encore peu exploré et les techniques existantes reposent beaucoup sur des mesures de similarité entre attributs ou sur des mesures de variances.

Dans les chapitres qui suivent, nous allons présenter nos trois contributions qui visent à, d'une part, réduire la dimensionnalité des données, d'autre part, nettoyer les données mal-étiquetées (bruit de classe), tout en améliorant la qualité de la classification.

Deuxième partie

Contributions

Chapitre

4 La détection et l'élimination de bruit de classe

Sommaire

4.1 Introduction

Il n'est pas rare, en pratique, que les données dont nous disposons pour l'apprentissage soient altérées à cause du bruit de classe. Les informations enregistrées, qui constituent l'ensemble d'apprentissage, s'avèrent donc être en partie erronées. Ce qui risque d'induire nos algorithmes en erreur, et donc de faire échouer l'apprentissage. Pourtant, à condition d'être capable de gérer ce bruit, il est probablement encore possible d'extraire des caractéristiques intéressantes de ces données. La gestion du bruit est donc un enjeu très important en apprentissage. Or dans la littérature il n'y a pas d'approches qui fournissent une architecture générale pour la détection de bruit de classe. Certaines approches proposent simplement une amélioration pour un seul niveau du processus de détection de bruit de classe. Alors que notre deux architectures sont complètes, simples et efficaces en termes de précision, couvrant toutes les étapes de la détection et de nettoyage. Ce qui va être discuté d'avantage dans ce chapitre. Le filtre proposé dans la deuxième approche est une amélioration de la première architecture proposées. Dans notre première approche

nous proposons une architecture pour la détection et l'élimination du bruit de classe en utilisant les règles d'extraction . Dans notre deuxième approche un nouveau filtre du bruit est proposé pour identifier et éliminer le bruit de classe, appelé MIPCNF. Vu qu'il n'existe pas de filtre unique qui surperforme systématiquement chacun des autres pour différents types de base de données et à différents niveaux de bruit, notre approche vise à surmonter les problèmes rencontrés dans la littérature pour la détection et l'élimination de bruit de classe. Notre filtre repose sur un algorithme dans lequel plusieurs tours de détection de bruit de classe sont effectués sur différentes partitions de données à l'aide d'un ensemble de classifieurs, en utilisant différentes stratégies de filtrage : filtre de bruit itératif, filtre de partitionnement et un filtre basé sur un ensemble de classifieurs. Les résultats expérimentaux, sur 14 bases de données du monde réel, ainsi que l'analyse statistique, démontrent que le filtre MIPCNF peut non seulement surmonter un niveau de bruit élevé, mais également surperformer les dernières stratégies de détection et d'élimination du bruit de classe à différents niveaux de bruit. Selon différents travaux faites dans littérature, nous signalons quelques problèmes trouvés dans ce contexte :

— Il n'existe pas de filtre unique surperforme les autres filtres dans différentes types de base de données et à différents niveaux de bruit. Par exemple, dans l'approche EF, le bruit est éliminé en un seul tour, ce qui peut ignorer beaucoup de bruit de classe.
— Le problème de la sélection d'un classificateur approprié en fonction de chaque problème.
— Un autre problème qui peut être soulevé est le mauvais choix de l'apprenant.
— Problème de la taille limite du jeu de données.

L'objectif de notre approche était de surmonter tous ces problèmes. Les points forts de notre filtre sont les suivants :

— Notre solution surpasse les dernières stratégies de la détection et l'élimination du bruit de classe dans différents niveaux de bruit,
— Permet la sélection des classifieurs appropriés en fonction des caractéristiques des données.
— L'algorithme est évolutif car il peut filtrer les données volumineuses et / ou distribuées.
— Les expérimentations effectuées sur différentes bases de données, montrent que notre approche surmonte un niveau de bruit très élevé.

4.2 Première approche

4.2.1 Principe et algorithme

Pour éliminer le bruit de classe dans les grandes ensembles de données, les techniques traditionnelles ne peuvent généralement pas être appliquées directement. Pour faire face à ce défi, nous proposons une architecture pour la détection et l'élimination du bruit de classe, qui peut aussi être appliquée pour les ensembles de données volumineux (figure 4.1). En s'inspirant du travail effectué dans (Zhu *et al.*, 2006). Notre architecture comprend quatre étapes principales :

— partitionnement des données en sous-ensembles de données
— extraction des règles d'association à partir de chaque sous-ensemble
— application d'un ensemble de classifieurs pour chaque sous-ensemble
— combinaison de tous les résultats afin d'obtenir une décision finale

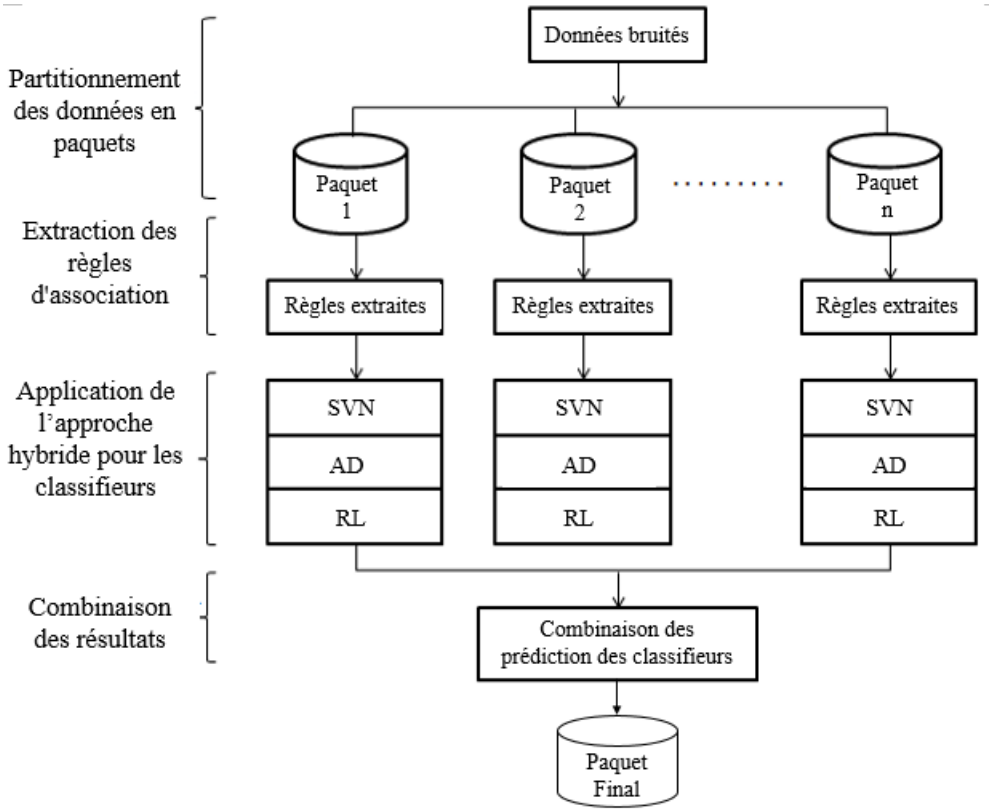

Figure 4.1 – Architecture pour la détection et l'élimination de bruit de classe

Dans un premier temps, les données sont divisées en sous-ensembles suffisamment petits pour être facilement manipulés et classés par un classifieur. L'objectif est de paralléliser le processus afin de gagner au niveau de temps d'exécution et surmonter le problème du grand volume de données.

Algorithme 4.1 Détection et élimination de bruit de classe

Entrée : l'ensemble des données E
Sortie : L'ensemble des données nettoyé CE
Partitionnement de l'ensemble E en n sous ensembles
pour $i = 1, ..., n$ **faire**
  Extraire les règles d'association R_i à partir de chaque sous ensemble
  Choisir les règles pertinentes GR_i à partir de R_i
  Appliquer m classifieurs à chaque sous ensemble
  $P_1, P_2, ...P_m$ sont les prédictions de m classifieurs
  (I,N)=vote majoritaire($P_1, P_2, ...P_m$) tel que N : est le bruit détecté et I : les bons exemples
  Ajouter N_i à TN tel que : TN=TN U N_i avec TN : est le bruit totale détecté
fin pour
$CE = E - TN$
Supprimer TN

La deuxième étape consiste à extraire les règles pertinentes de chaque sous-ensemble ; La règle d'association est un outil permettant de créer des relations intéressantes entre des attributs. Ces règles doivent être simples et précises. Parmi les algorithmes d'exploration de données capables d'extraire les règles d'association, on trouve l'algorithme Arbre de Décision, l'algorithme génétique et l'algorithme Apriori qui extrait les règles générales. Dans notre expérimentation on a utilisé l'algorithme Apriori pour l'extraction des règles d'association.

La troisième étape consiste à appliquer un ensemble de classifieurs sur chaque sous-ensemble pour détecter le bruit de classe. Les classifieurs utilisés dans notre expérimentation sont : SVM, les arbres de décision AD et la régression logistique. L'application d'un classifieur est divisée en deux étapes :

— Phase d'apprentissage, qui permet aux classifieurs de reconnaître d'autres exemples.
— Phase de test, qui teste ces classifieurs avec de nouveaux échantillons.

La méthode d'application de ces classifieurs est également importante. En fait, il pourrait être appliqué de manière séquentielle, parallèle ou hybride. Dans notre architecture, nous avons appliqué trois classifieurs pour chaque sous-ensemble en utilisant l'approche hybride. La dernière étape consiste à combiner les prédictions de différents classifieurs. Le résultat final est obtenu par un vote majoritaire. Une autre méthode de combinaison c'est le vote par consensus, l'inconvénient de ce filtrage est le risque supplémentaire lié à la conservation de données erronées. Nous adoptons donc le vote majoritaire dans nos expérimentations pour sa simplicité et son efficacité.

4.2.2 Outils et bases de données utilisées

Dans les sections qui suivent, nous présentons nos expérimentations élaborées. Pour démontrer l'efficacité de notre approche nous comparons nos résultats avec ceux obtenus dans (Zhu *et al.*, 2006). Afin de faciliter cette comparaison, nous avons utilisé alors les mêmes bases de données utilisées, "Mushroom et Adult".

4.2.2.1 Base de données

Mushroom : Cet ensemble de données comprend des descriptions d'échantillons hypothétiques correspondant à 23 espèces de champignons branchies de la famille des Agaricus et des Lépiotes (attributs nominaux). Chaque espèce est identifiée comme définitivement comestible, définitivement toxique ou d'une comestibilité inconnue et non recommandée. Cette dernière classe a été combinée avec celle empoisonnée. Le guide indique clairement qu'il n'existe pas de règles simples pour déterminer la comestibilité d'un champignon. Il contient des informations sur 8124 champignons. Sur deux classes, 4208 (51,8%) sont comestibles et 3916 (48,2%) sont toxiques.

Adult : La base de données Adult contient environ 48 842 observations avec 14 attributs dont six sont continus et huit sont catégoriques, et une étiquette (classe) binomiale indiquant un salaire <50K ou >50K USD. Les données ont été divisées en un ensemble contenant 32 561 enregistrements et un ensemble de données de test contenant 16 281 enregistrements. La classe d'emploi décrit le type d'employeur, indépendant ou fédéral, et le type d'activité détermine la profession comme agriculteur, employeur de bureau ou de direction. L'éducation contient le plus haut niveau d'éducation atteint tel que le lycée ou doctorat. L'attribut relationnel a des catégories telles que célibataire ou mariée et l'état matrimonial a des catégories telles que mariée ou divorcée. Les autres attributs nominaux sont par exemple, le pays de résidence, le sexe et l'origine. Les attributs continus sont : l'âge, les heures travaillées par semaine, le numéro d'éducation (représentation numérique de l'attribut d'éducation).
Tous les algorithmes sont implémentés avec Java et WEKA package (Witten *et al.*, 2016).

4.2.2.2 Outils d'apprentissage

Waikato Environment for Knowledge Analysis (Weka) : c'est une suite populaire de logiciels d'apprentissage automatique écrits en Java et développés à l'Université de Waikato, en New-Zélande. Weka est un logiciel libre disponible sous la licence publique générale GNU. Le workbench Weka contient un ensemble d'outils de visualisation

et d'algorithmes d'analyse de données et de modélisation prédictive, ainsi que des interfaces utilisateur graphiques facilitant l'accès à ses fonctionnalités. Weka est une collection d'algorithmes d'apprentissage automatique, entre autres, les arbres de décision et les réseaux de neurones, permettant de résoudre des problèmes d'exploration de données dans le monde réel. Il est écrit en Java et fonctionne sur presque toutes les plateformes. Les algorithmes peuvent être appliqués directement aux données ou appelés à partir d'un code Java. La version originale non-java version de Weka était une interface TCL / TK pour des algorithmes de modélisation implémentés dans d'autres langages de programmation, comme le langage C. Cette version originale était principalement conçue comme un outil d'analyse de données de domaines agricoles, mais la version la plus récente, entièrement basée sur Java (Weka 3), dont le développement a commencé en 1997, est maintenant utilisée dans de nombreux domaines d'application, en particulier dans le secteur de l'éducation et pour la recherche. Les avantages de Weka comprennent :

— Disponibilité gratuite sous la licence publique générale GNU
— Portabilité, car elle est entièrement implémentée dans le langage de programmation Java et fonctionne donc sur presque toutes les plateformes informatiques modernes.
— Une collection complète de techniques de pré-traitement et de modélisation des données
— Facilité d'utilisation grâce à ses interfaces utilisateur graphiques

Weka est une collection d'outils d'exploration de données pour :

— Pré-traitement des données
— Classification
— Régression
— Clustering
— Sélection d'attribut
— Visualisation

Toutes les techniques de Weka reposent sur l'hypothèse que les données sont disponibles sous la forme d'un seul fichier ou d'une seule relation, chaque point de données étant décrit par un nombre fixe d'attributs (normalement, des attributs numériques ou nominaux, mais certains autres types d'attributs sont également pris en charge). Weka fournit un accès aux bases de données Structured Query Language (SQL) à l'aide de Java Database Connectivity et peut traiter le résultat renvoyé par une requête de base de données. Il n'est pas capable d'exploration de données multi-relationnelle, mais un logiciel distinct permet de convertir une collection de tables de base de données liées en une seule table adaptée au traitement à l'aide de Weka. Un autre domaine important qui

n'est actuellement pas couvert par les algorithmes inclus dans la distribution de Weka est la modélisation de séquence.

4.2.2.3 Classifieurs

Les classifieurs utilisés dans notre expérimentation sont : Arbre de Décision (AD), SVM et la régression logistique (LR). Mais on peut aussi utiliser d'autre classifieurs comme : Naïve Bayes, K – Nearest Neighbor ou bien Random Forest. Tous les tests sont effectués sur un ordinateur doté d'un processeur 2,7 GHz et d'une Mémoire de 4 Go.

arbres de décision (AD) est une collection hiérarchique de règles qui décrivent comment diviser une grande collection de données en groupes en fonction des régularités des données (Omitaomu, 2006). En général un arbre de décision est un classifieur présenté sous forme d'une structure arborescente utilisée pour la classification, la régression, clustering et la fonction de prédiction. Chaque nœud de l'arbre est : i) soit un nœud "de décision" où des tests qui ont été effectués sur les valeurs d'une seule variable. ii) soit un nœud "feuille" qui détient la prédiction de la classe. Les nœuds internes d'un arbre de décision désignent les différents attributs, les branches entre les nœuds nous indiquent les valeurs possibles que ces attributs peuvent avoir dans les échantillons observés, tandis que les nœuds terminaux nous indiquent la valeur finale (classification) de la variable dépendante. Les arbres de décision sont très intéressants car ils décrivent une relation claire entre les données d'entrées et sorties cibles en fournissant des règles faciles à interpréter par l'être humain, ces règles inductives sont crées pour tous les chemins possibles de la racine à une des feuilles. Le classifieur d'arbre de décision AD utilise un algorithme simple pour classer un nouvel élément. Il faut d'abord créer un arbre de décision basé sur les valeurs d'attributs des données d'entraînement disponibles. Ainsi, chaque fois qu'il rencontre un ensemble d'éléments (ensemble d'apprentissage), il identifie l'attribut qui distingue le plus clairement les différentes instances. Cet attribut qui est capable de nous donner le plus d'informations sur les instances de données afin que nous puissions les classer le mieux, c'est à dire avoir le plus grand gain d'information. Parmi les valeurs possibles de cet attribut, s'il y a une valeur pour laquelle il n'y a pas d'ambiguïté, pour laquelle les instances de données appartenant à sa catégorie ont la même valeur pour la variable cible, alors on termine cette branche et assignons à la valeur cible que nous avons obtenue.

Support Vector Machines (SVM) sont des méthodes d'apprentissage supervisées utilisées pour la classification, ainsi que la régression. Lorsque la sortie de la fonction est une valeur continue, on dit que la méthode d'apprentissage effectue une régression ; et quand la méthode d'apprentissage peut prédire une étiquette de classe de l'objet d'entré, on l'appelle classification (Vapnik, 2013). L'idée de base de SVM est de générer un hyperplan qui pourrait séparer les points de données en deux classes. Si les deux classes sont linéairement séparables, le SVM standard tente de générer un hyperplan qui divise l'espace d'entré en deux demi-espaces disjoints où chaque classe appartient à l'un des demi-espaces. Comme il peut exister plusieurs hyperplans séparant les classes, l'algorithme SVM sélectionne l'hyperplan qui est le plus éloigné de ses points de données les plus proches. Tout nouveau point de données est classé dans une classe en fonction de son emplacement dans le demi-espace. La formule pour la sortie d'un SVM linéaire 4.1 est comme suite :

$$U = W.X - b \tag{4.1}$$

Dans cette équation w est le vecteur normal à l'hyperplan et x est le vecteur d'entré. Les points les plus proches se trouvent sur les plans u =+- 1. Une chose importante à noter à propos de SVM est que les données à séparer doivent être binaires. Même si les données ne sont pas binaires, SVM réduit le problème multi-classes à un ensemble de problèmes à deux classes et complète l'analyse par une série d'évaluations binaires sur les données. Fondamentalement, cela implique de regarder une classe particulière présente dans les données, et de prédire la valeur d'une instance pour cette classe seule d'une manière Oui / Non. Ceci est fait pour chaque classe et les résultats obtenus sont ensuite combinés. Les équations du noyau sont des fonctions qui transforment linéairement les données non séparables dans un domaine en un autre domaine où les instances deviennent linéairement séparables. Les équations du noyau peuvent être linéaires, quadratiques, gaussiennes ou toute autre chose qui atteint ce but particulier.

Logistique Regression (LR) est la méthode la plus utilisée pour modéliser la réponse des données binaire. Lorsque la réponse est binaire, elle prend typiquement la forme de 1/0, avec 1 indiquant généralement un succès et 0 un échec. Cependant, les valeurs réelles que 1 et 0 peuvent avoir, varie considérablement selon le but de l'étude. Par exemple, pour une étude des chances d'échec dans un cadre scolaire, 1 peut avoir la valeur d'échouer, et 0 de non-échouer, ou passer. Le point important est que 1 indique le premier sujet d'intérêt pour lequel une étude de réponse binaire est conçue. Modéliser

une variable de réponse binaire en utilisant la régression linéaire normale introduit un biais important dans les estimations de paramètres (Harrell, 2015). Le modèle linéaire standard suppose que les observations dans le modèle sont indépendantes. Il existe de nombreuses applications de la régression logistique. Certes ce modèle est utile pour la classification à deux classes, mais il peut être étendu à un modèle de classification multi-classe, qui est un cas particulier de champs aléatoires conditionnels. Il est également appelé le modèle d'entropie maximale dans la communauté de traitement du langage naturel.

RandomForest (RF) ou les forets aléatoires sont la généralisation du partitionnement récursif qui combine une collection d'arbres appelée un ensemble. La forêt aléatoire a d'abord été proposée par Tin Kam Ho de Bell Labs (Ho, 1995), qui a ensuite été prolongée par Leo Breiman, qui a également inventé le terme «Random Forest». Les forêts aléatoires (Lemmond *et al.*, 2010) sont une collection d'arbres distribués de façon identique dont la valeur de classe est obtenue par une variante au vote majoritaire. Le classifieur se compose d'une collection d'arbres comme des classifieurs qui utilisent un grand nombre d'arbres de décision, sont tous formés pour s'attaquer au même problème. Il y a trois facteurs principaux montrant l'individualité des arbres :

- Chaque arbre est traité en utilisant un sous-ensemble aléatoire d'échantillons.
- Lorsque l'arbre devient grand, la meilleure répartition sur chaque nœud de l'arbre est trouvée par une recherche à travers n attributs sélectionnés au hasard. Pour un ensemble de données avec N fonctionnalités, n est sélectionné et maintenu plus petit que celui de N.
- Chaque arbre est fait pour grandir au maximum afin qu'il n'y ait pas d'élagage.

Les forêts aléatoires sont des classifieurs d'arbres qui sont formés au choix aléatoire des sous-ensembles de données d'entrée lorsque la classification finale est basée sur le vote majoritaire sur les arbres.

Naive Bayes (NB) est un classifieur probabiliste simple basé sur le théorème de Bayes où chaque attribut est supposé être indépendant de la classe (Murty et Devi, 2011). L'apprentissage bayésien naïf utilise tous les attributs contenus dans les données, et les analyse individuellement comme s'ils sont tous aussi importants et indépendants les uns des autres. La classification des instances devient difficile lorsque l'ensemble de données contient un grand nombre d'attributs et de classes car il faut un nombre énorme d'observations pour estimer les probabilités(Murty et Devi, 2011). Lorsqu'un attribut est

supposé être indépendant de classe conditionnellement, cela signifie en réalité que l'effet d'une valeur de variable sur une classe donnée est indépendant des valeurs des autres variables.

K – Nearest Neighbor (KNN) est un classifieur qui ne nécessite pas d'apprendre une fonction précise d'apprentissage pour qu'il prédit la classe des nouvelles instances. Un échantillon de données dans KNN est classé en se basant sur un nombre sélectionné k des voisins les plus proches (He *et al.*, 1998).

Dans le cas des jeux de données de petites dimensions, ce modèle a la capacité de fournir à l'utilisateur un certain type d'explication concernant la classification de chaque nouvelle instance. Ces explications sont obtenues à l'aide d'une analyse simple des K plus proches voisins utilisés pour classer une instance. Les hypothèses suivies dans KNN sont :

- KNN suppose que les données sont dans un espace d'attributs, de sorte qu'ils ont le concept de distance. La distance euclidienne peut être utilisée pour calculer la distance entre les vecteurs.
- Chaque vecteur d'apprentissage est associé à un ensemble de vecteurs et à une étiquette de classe.
- K décide combien de voisins influencent la classification.

La classification des voisins les plus proches peut être décidée en calculant le nombre de valeurs de classe individuelles à partir de tous les k plus proches voisins. K est un nombre impair pour éviter les comptes en double. Il est à signaler que dans le cas des jeux de données de grandes dimensions, l'algorithme des K-plus proches voisins devient un modèle boîte noire.

4.2.3 Resultats Experimentaux

Tout d'abord, nous divisons l'ensemble de données en cinq sous-ensembles (ce nombre est choisi en se basant sur le travail effectué dans (Zhu *et al.*, 2006)). Ensuite, nous appliquons l'algorithme Apriori à chaque sous-ensemble pour extraire les règles. L'algorithme se déroule en deux étapes. La première est de rechercher des ensembles des éléments fréquents dont le support est supérieur à un seuil prédéfini. La deuxième étape repose sur l'extraction de ces ensembles des règles dont la confiance est considérée comme la plus suffisante. Le nombre de règles extraites est généralement important. Pour sélectionner les règles les plus intéressantes, il est utile de les classer par ordre décroissant selon un

indice de confiance. Une règle intéressante est une règle avec Intervalle de Confiance (IC) supérieur à la probabilité absolue du résultat ($P(R)$) telle que :

If $A \rightarrow B$ $IC = P(A,B) * P(A), IS = P(A), P(R) = IC * IS$

Où A et B sont deux expressions logiques et $P(A)$ c'est la probabilité que A soit vérifié. Ainsi, $IC > P(R)$ implique que $A \rightarrow B$ est une règle intéressante. Dans notre cas nous avons choisi de générer 10 règles associées. Un exemple des règles pertinentes extraites de la base de données Mushroom est présenté à la figure 4.2.

```
Best rules found:

 1. veil-color=w 1436 ==> veil-type=p 1436    conf:(1)
 2. gill-attachment=f 1435 ==> veil-type=p 1435    conf:(1)
 3. gill-attachment=f veil-color=w 1434 ==> veil-type=p 1434    conf:(1)
 4. gill-attachment=f 1435 ==> veil-color=w 1434    conf:(1)
 5. gill-attachment=f veil-type=p 1435 ==> veil-color=w 1434    conf:(1)
 6. gill-attachment=f 1435 ==> veil-type=p veil-color=w 1434    conf:(1)
 7. veil-color=w 1436 ==> gill-attachment=f 1434    conf:(1)
 8. veil-type=p veil-color=w 1436 ==> gill-attachment=f 1434    conf:(1)
 9. veil-color=w 1436 ==> gill-attachment=f veil-type=p 1434    conf:(1)
10. veil-type=p 1462 ==> veil-color=w 1436    conf:(0.98)
```

Figure 4.2 – Régles d'association extraites.

Les modèles de classification sont souvent représentés sous forme de règles ayant la forme suivante : $P \rightarrow C$, où P est un attribut de prédiction dans les données d'apprentissage et C est le libellé de classe ou l'attribut cible. Par exemple, si nous prenons la première règle : veil-color $= w \Rightarrow$ veil-type $= p$, nous classifierions le nouveau cas comme suit : veil-type $= p$ si nous avons veil-color $= w$. Dans nos expérimentations sur la base de données Mushroom et Adult. Nous utilisons 80% pour les données d'entraînement et 20% pour les tests. L'ensemble des données d'entraînement sera divisé en cinq sous-ensembles de données. Les caractéristiques des différents sous-ensembles sont présentés dans le tableau 4.1.

Sous-ensembles de données	**Instances**	**Pourcentage de données**
Train	7312	90
Test	812	10
1^{st}-Data Train	1462	25
2^{nd}- Data Train	1462	25
3^{rd}- Data Train	1448	25
4^{th}- Data Train	1470	25
5^{th}- Data Train	1470	25

Tableau 4.1 – Partitionnement des données

Nous ajoutons différents niveaux de bruit aux données d'entraînement. Pour chaque itération, nous utilisons 10 cross-validation. Nous évaluons les données en utilisant les facteurs suivants : $P(ER1)$ la probabilité d'avoir un faux positifs, Noise Elimination Precision (NEP) la précision d'élimination du bruit et Instances Incorrectly Classified (ICI) les instances classées incorrectement. La précision est le rapport entre le nombre de vrais positifs et la somme de vrais positifs et de faux positifs. La valeur 1 indique que tous les exemples énumérés sont positifs. ICI est le pourcentage du nombre total d'instances mal classées. Leurs définitions sont données par les équations 4.2.

$$P(ER_1) = \frac{|F \cap M|}{|G|}, NEP = \frac{|F \cap M|}{|F|} \tag{4.2}$$

$|F|$ dénote le bruit détecté, $|M|$ est le bruit ajouté, et $|G|$ est l'ensemble des instances non bruyantes. Les résultats de la détection de bruit sont donnés dans les tableaux 4.2 et 4.3. Nous calculons le pourcentage d'instances mal classées $P(ER_1)$ et la précision de la classification en fonction de différents niveaux de bruit. Nous avons comparé notre approche avec l'approche PF(Partitioning filter) proposée dans (Sun *et al.*, 2007) en terme de précision de la classification NEP, ICI et $P(ER_1)$. Les meilleurs résultats obtenus à chaque niveau de bruit sont mis en évidence (en gras). 4.2.

	Niveau de bruit	$P(ER_1)$	**Précision**	ICI
Notre Approche	10	0.048	**0.898**	**0.096**
	20	0.112	0.799	0.192
	30	0.192	0.685	0.313
	40	0.340	0.569	0.424
Approche PF	10	0.016	0.894	0.110
	20	0.029	0.938	0.204
	30	0.082	0.926	0.288
	40	0.228	0.662	0.419

Tableau 4.2 – Résultat de la détection de bruit dans la base de données Mushroom à différents niveaux de bruit

	Niveau de bruit	$P(ER_1)$	Précision	ICI
Notre Approche	5	**0.001**	**0.953**	4.36
	10	**0.005**	**0.895**	10.09
	15	**0.015**	**0.830**	15.86
	20	**0.028**	**0.764**	24.25
Approche PF	5	0.0335	0.718	6.41
	10	0.0463	0.687	10.91
	15	0.0563	0.619	15.46
	20	0.0725	0.512	20.30

Tableau 4.3 – Résultat de la détection de bruit dans la base de données Adult à différents niveaux de bruit

4.2.4 Analyse des résultats

Pour la base de données Mushroom, notre approche donne des résultats meilleurs que PF en terme de précision de la classification surtout lorsque le niveau de bruit est relativement faible. Cependant, pour la base de donnée Adult , notre approche fournit des meilleurs résultats pour l'identification de bruit de classe dans tous les niveaux de bruit. Généralement, certains bases de données (par exemple, Mushroom) peuvent contenir des redondances relativement importantes et de nombreuses exceptions, sa précision de classification est relativement faible. D'après les tableaux 4.2 et 4.3, nous pouvons conclure que notre approche est beaucoup plus efficace que l'approche PF et rarement qu'elle peut identifier des bons exemples comme du bruit (des erreurs ER_1). Pour chaque niveau de bruit, nous avons calculé le pourcentage de bruit de classe déclaré non bruyant. Les résultats obtenus sont présentés dans le tableau 4.4. Lorsque le niveau de bruit est très élevé, nous avons constaté que le pourcentage de bruit de classe dans les bons exemples est très faible par rapport à l'approche PF.

Niveau de bruit	**10**	**20**	**30**	**40**
Niveau de bruit dans les bons exemples pour notre approche	0.50	1.06	1.50	2.15
Niveau de bruit dans les bons exemples pour l'approche PF	0.173	0.654	1.233	6.173

Tableau 4.4 – Pourcentage de faux positif à différents niveaux de bruit (base de données Mushroom)

4.2.5 Conclusion

Notre première approche proposée est une architecture pour la détection et l'élimination du bruit de classe. Le processus de nettoyage des données est présenté comme suit. Étant donné un grand ensemble de données, nous divisons d'abord les données en plusieurs sous-ensembles. Ensuite, nous extrayons les meilleures règles d'association à partir de ces sous-ensembles. Ensuite, nous appliquons un ensemble de classifieurs, pour chaque sous-ensemble de données, en utilisant un modèle hybride de combinaison. Enfin, nous combinons les résultats des classifieurs de tous les sous ensembles pour obtenir une décision finale. En utilisant Multi Classifiers Filtre (MCF), nous pouvons détecter le bruit avec plus de précision qu'un filtre à classifieur unique ; MCF nous permet de se bénéficier des avantages de chaque algorithme d'apprentissage. L'utilisation d'un filtre à classifieur unique peut être risquée, car l'apprenant peut ne pas avoir la capacité d'apprendre les concepts du problème d'un domaine donné. Les résultats des expérimentations élaborés montrent l'efficacité de notre processus, en particulier lorsque l'ensemble de données est dépourvu d'un grand nombre de redondances.

4.3 Deuxième approche

4.3.1 Principe et algorithme

MIPCNF permet de détecter et d'éliminer le bruit de classe de manière itérative en utilisant un ensemble de classifieurs pour chaque sous-ensemble de données. Notre processus d'élimination du bruit de classe peut être résumé en trois étapes : Tout d'abord, on partitionne notre base de données, nous divisons les données d'apprentissage en sous-ensembles équitables. Ensuite, le filtre basé sur un ensemble de classifieurs est appliqué pour chaque sous-ensemble en plusieurs itérations. Et finalement, nous combinons la prédiction des classifieurs à l'aide d'une méthode de vote pour détecter le bruit de classe. Un schéma de notre proposition MIPCNF est présenté dans la figure 4.3.

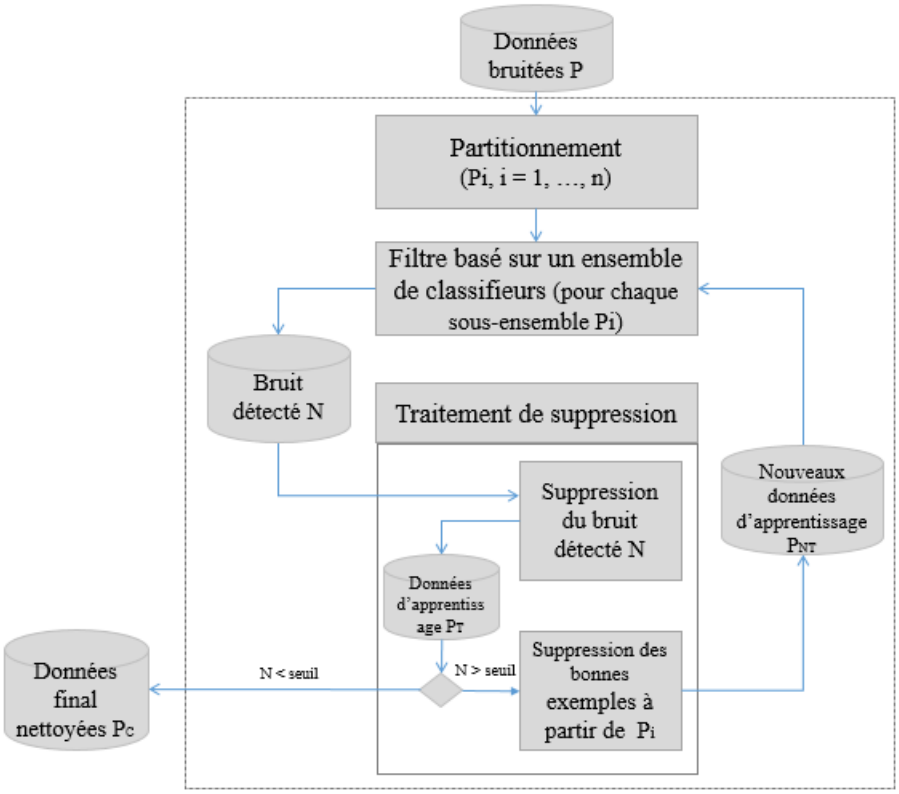

Figure 4.3 – Architecture de l'approche MIPCNF.

Les composants de notre architecture sont :

Partitionnement des données : Cette étape consiste à diviser les données en sous-ensembles équitables afin de réaliser un traitement parallèle ; cela peut être bénéfique en cas de données volumineuses.

Filtrage basé sur un ensemble de classifieurs : Un ensemble de classifieurs est appliqué à chaque sous-ensemble au lieu d'un seul classifieur. Ensuite, les sorties partielles

sont combinées en une seule prédiction en utilisant une méthode de vote. L'idée derrière le faite de collecter les prédictions de différents classifieurs c'est qu'on pourrait obtenir une meilleure détection du bruit de classe qu'à travers un seul classifieur. Cette diversité est l'une des clés de cette approche. D'autre part, l'utilisation simultanée de plusieurs classifieurs permet de combiner les avantages sans cumuler les inconvénients.

Processus de vérification : Dans l'itération en cours, nous supprimons les exemples jugés comme bruit de classe. Si le bruit détecté dépasse le seuil prédéfini, un certain nombre de bons exemples sont également supprimés du sous-ensemble. Ensuite, une nouvelle itération est lancée et un nouveau filtre basé sur un ensemble de classifieurs est appliqué en utilisant des données partiellement nettoyées. Le processus sera répété de manière itérative jusqu'à ce que le critère d'arrêt soit satisfait.

L'élimination des bons exemples présente l'avantage de permettre de réduire la taille du jeu de données. Par conséquent, le processus de classification peut être plus rapide à la prochaine itération.

Par ailleurs, le principal avantage de l'élimination du bruit en plusieurs itérations est la possibilité d'éviter le bruit de classe détecté dans la nouvelle itération.

L'algorithme détaillé de notre filtre est représenté comme suit 4.2. Les paramètres utilisés sont décrits au tableau 4.5.

Parameters	**Description**
P	L'ensemble de données de traitement
n	Nombres de sous-ensembles
N_i	Le bruit détecté dans la $i^{ème}$ itération
G_i	Portion de bons exemples dans la $i^{ème}$ itération
P_{NT}	L'ensemble de données de traitement après la suppression de N_i et G_i
P_T	L'ensemble de données de traitement avec l'ensemble N_i supprimé
P_C	Données finale nettoyées
L	L'ensemble des classifieurs
m	Nombre de classifieurs
β	Le pourcentage des bons exemples
T	Le seuil de critère d'arrêt

Tableau 4.5 – Description des paramètres utilisés dans notre filtre de bruit proposé

La différence entre notre filtre et le filtre INFFC (Sáez *et al.*, 2016a) dont on est inspiré est présentée comme suit : L'algorithme INFFC semble prendre un temps d'exécution important. En effet, l'apprentissage d'un ensemble des données en une seule fois

Algorithme 4.2 MIPCNF : Multi-Iterative Partitioning Class Noise Filter

Entrée : P, données d'apprentissage avec $\|P\|$ instances
Paramètres : n, nombre des sous-ensembles
L, ensemble des classifieurs
m, nombre des classifieurs
β, pourcentage des bons exemples à supprimer dans chaque itération
T, seuil pour le critère d'arrêt
Sortie : P_c, ensemble d'apprentissage nettoyé
$P_c \leftarrow \emptyset$
Pour chaque sous ensemble P_i tel que $\cup P_i = P$ et i allant de 1 à n
pour $i = 1, ..., n$ **faire**
  $N_i \leftarrow \emptyset$
  $G_i \leftarrow \emptyset$
  pour $j = 1, ..., m$ **faire**
    Construire le classifieur L_j et évaluer les données $L_j(P_i)$
  fin pour
  Appliquer le vote majoritaire à $L(P_i)$
  si $L(P_i, N_i) > T$ **alors**
    $G_i \leftarrow$ SelectPortionOfGoodExamples(P_i, β)
    $P_c \leftarrow P_c$ / G_i
    $P_c \leftarrow P_c$ / N_i
    Retour au début
  finsi
  $P_c = P_c \cup P_i$
fin pour

est parfois très difficile avec des ensembles de données volumineux. Par conséquent, dans certains cas, les jeux de données ne peuvent pas être traités en même temps par le même modèle. L'approche de partitionnement adoptée par notre filtre, décrit par (Zhu et Wu, 2004), essaie de résoudre le problème de la limitation de la taille du jeu de données. Dans notre approche, le jeu de données d'apprentissage est d'abord partitionné en sous-ensembles quasiment de même taille. Par ailleurs, un autre avantage de notre filtre par rapport à INFFC c'est le faite de de supprimer une partie des bons exemples à chaque itération. De ce fait, la taille du jeu de données est réduite, ce qui permet au processus d'apprentissage de fonctionner plus rapidement à la prochaine itération. Les résultats expérimentaux présentés dans la section suivante démontrent l'efficacité de notre filtre.

4.3.2 Complexité de l'algorithme

Trois aspects principaux doivent être pris en compte lors de la comparaison de la complexité de notre filtre MIPCNF à celle des autres filtres, considérés dans notre expérimentation (INFFC, EF et IPF) : l'élimination itérative, le partitionnement et l'utilisation d'un score de bruit. La différence la plus significative en termes de complexité temporelle du MIPCNF par rapport au INFFC est le calcul du score de bruit utilisé par ce dernier

(qui n'est utilisé ni par le MIPCNF, ni par EF ni par IPF), qui est principalement basé sur l'utilisation du classifieur k-NN. Pour cette raison, le filtre INFFC pourrait être plus lent que d'autres filtres, tels que MIPCNF, EF et IPF. L'augmentation possible de la complexité temporelle de la stratégie d'évaluation du bruit utilisée par le MIPCNF par rapport aux filtres EF et INFFC est due à deux causes principales : son élimination itérative des exemples bruyants (qui n'est pas utilisée par EF, mais qui est utilisée aussi par IPF et INFFC). Ainsi qu'au partitionnement des ensembles de données d'entraînement (qui n'est pas utilisé par INFFC, mais qui est utilisée aussi par IPF et EF). La complexité de calcul de notre algorithme (MIPCNF) est O (n + K n^2), où n est le nombre d'instances dans l'ensemble de données et K le nombre de classifieurs utilisés. Le temps d'exécution pris par IPF peut être estimé à O (n^2) + O (n) (pour IPF nous avons K = 1), il a approximativement la même complexité temporelle que notre filtre. La durée d'exécution d'INFFC dépend de la complexité de l'algorithme kNN utilisé dans le calcul du score, la complexité de k-NN est de O (n. m), où n et m sont respectivement le nombre d'attributs et le nombre des exemples dans l'ensemble d'apprentissage. Ainsi, la complexité temporelle globale peut être proche de O (n^2) + O (n m). La complexité temporelle de EM est K h O ([n m] / h), où K et h sont respectivement le nombre de classifieurs et le nombre de partitions, en supposant que K et h sont des constantes, la complexité temporelle globale peut être simplifiée davantage en O (n. m). Comparé à notre méthode, cet algorithme est plus rapide.

Cependant, l'importance du facteur du temps dépend d'un domaine à un autre, l'objectif principal de notre proposition est d'améliorer la précision de la classification. Dans nos travaux futurs, nous tenterons de minimiser les coûts de calcul de notre approche.

Cette section présente les détails de notre étude expérimentale en utilisant une vaste collection de base de données du monde réel. Tout d'abord, la première partie décrit les bases de données utilisées, ainsi que les paramètres et les méthodes utilisées par notre algorithme. Ensuite, les résultats expérimentaux vont être présentés, y compris une étude comparative avec d'autres filtres bien connus dans la littérature à différents niveaux de bruit. La description de l'influence de certains paramètres sur la précision de la classification est également discutée.

4.3.3 Bases de données utilisées

Nous avons évalué notre stratégie d'élimination du bruit de classe en utilisant 14 bases de données extraites à partir de KEEL et UCI (Alcala-Fdez *et al.*, 2011; Lichman,

2013). Ces bases de données fournissent une bonne représentation de différentes caractéristiques : le nombre d'instances varie de 101 à 58 000, le nombre d'attributs de 4 à 60 et le nombre de classes de 2 à 10.

— Balance : La base de données Balance contient 625 exemples qui modélisent des expériences psychologiques avec trois résultats possibles. Il permet de prédire dans quelle direction une balance est basculée ou équilibrée. Chaque exemple est classé comme ayant la balance pointée vers la droite, pointée vers la gauche ou restant exactement au milieu (équilibrée).

— Car : Cet ensemble de données représente une collection d'enregistrements d'attributs spécifiques pour des voitures données par Marco Bohanec en 1997. Il a été dérivé d'un simple modèle de décision hiérarchique. La base de données d'évaluation de la voiture contient 6 attributs : achat, principal, portes, personnes, coffre et sécurité.

— Dermatology : Cette base de données contient 366 exemples d'occurrences de cancer en dermatologie, il a été trouvé sur OpenML - dermatology. Il contient 33 attributs, dont 6 sont : le psoriasis, la dermatite séboréique, le lichen plan, le pityriasis rosé, la dermatite chronique et le pityriasis rubra pilaris.

— Ecoli : Cette base de données contient des informations sur Escherichia coli (sites de localisation des protéines). C'est une bactérie du genre Escherichia que l'on trouve couramment dans l'intestin inférieur de l'organisme à sang chaud. L'ensemble de données comprend 336 séquences de protéines étiquetées selon 8 classes.

— Ionosphere : Cette base de données contient les données radar collectées par un système à Goose Bay, au Labrador. Les cibles étaient des électrons libres dans l'ionosphère. Il contient 351 exemples, chacun utilisant 33 attributs continus, divisés en deux classes. Les retours radar «bons» sont ceux montrant des signes de structure dans l'ionosphère et les retours radar «mauvais» sont ceux qui ne le sont pas.

— Iris : Cette base de données a pour objectif de prédire le type de fleur, les espèces de plantes Iris sont : Setosa, Versicolor, Virginica. Il comprend 150 exemples de fleurs caractérisées par 4 attributs (longueur du sépale, largeur du sépale, longueur du pétale et largeur du pétale), classés en 3 classes de 50 occurrences, chaque classe faisant référence à un type de plante d'iris.

— Monk : Cette base de données vise à prédire la classe en fonction des distributions prévues. La base de données comprend 556 exemples et 6 attributs nominaux : 2 avec 2 valeurs discrètes, 3 avec 3 valeurs discrètes et 1 avec 4 valeurs.

— New-Thyroid : Cette base de données représente le diagnostic de la thyroïde,

qu'elle soit hyper ou hypo fonctionnelle. Il contient 215 exemples et 5 attributs, utilisés pour classer 3 classes de fonction thyroïdienne en tant que sur-fonction, fonction normale ou sous-fonction.

— Penbased : Cette base de données concerne la reconnaissance des chiffres manuscrits. Il est constitué de 10992 exemples et de 16 attributs, décrivant les 10 chiffres écrits 0, ..., 9. Il y a donc 10 classes. Toutes les valeurs prises sont des valeurs entières comprises entre 0 et 100 inclus.

— Pima-diabète : Cette base de données a été réalisée par l'institut national du diabète et des maladies digestives et rénales. L'ensemble de données a pour objectif de prédire de manière diagnostique si un patient est diabétique ou non. Il y a un total de 768 exemples utilisant 8 attributs (6 discrets et 2 continus). Il existe 2 classes inégalement réparties, 500 exemples pour la classe 0 et 268 instances pour la classe 1.

— NASA Space Shuttle : Cette base de données est relativement volumineuse par rapport aux autres. Il s'agit d'un ensemble de données de classification complexe traitant du positionnement des radiateurs dans la navette spatiale. La base de données de contrôle de la NASA Shuttle est relativement volumineuse et comprend près de 58 000 vecteurs. Chaque instance est décrite par 9 attributs continus (temps, flux de rad, fermeture Fpv, ouverture Fpv ouverte, haute, contournement, fermeture Bpv, ouverture Bpv et code de classe), elle est affectée à l'une des 7 classes.

— Splice : les jonctions sont des points d'une séquence d'ADN. Le problème posé dans cette base de données est de reconnaître, à partir d'une séquence d'ADN, les frontières entre exon et intron. Cette base de données ADN contient 3 390 exemples avec 60 attributs, représentant une séquence de bases ADN.

— WDBC : C'est une base de données du Wisconsin sur le cancer du sein, elle contient 569 exemples, dont 357 cas bénins et 212 cas malins. Chaque instance est décrite par un index, un diagnostic et 30 attributs à valeurs réelles.

— Zoo : Cette base de données a pour objectif de pouvoir prédire la classification des animaux. Elle se compose de 101 animaux d'un zoo. Il existe 16 attributs avec des traits différents pour décrire les animaux utilisés, pour classer les animaux en sept types : Mammifère, Oiseau, Reptile, Poisson, Amphibien, Bug et Invertébré. La majorité des classes sont réparties dans des régions bien séparées de la projection.

Le tableau 5.1 récapitule les principales caractéristiques de ces bases de données, présentées en fonction du nombre d'exemples $\#EX$, du nombre d'attributs $\#AT$ et du nombre de classes $\#CL$. Les exemples contenant des valeurs manquantes ont été éliminés

des bases de données. Afin de contrôler la quantité de bruit dans chaque base de données et de vérifier son incidence sur les méthodes de filtrage du bruit. Nous injectons différents niveaux de bruit de classe dans les ensembles de données en remplaçant les étiquettes des exemples 0 %, 5 %, 10 %, 15 %, 20 %, 25 % et 30 %. Pour chaque base de données, nous avons 7 niveaux de bruit, donc 98 ensembles de données. Toutes les expérimentations ont été effectuées en utilisant 5-fold cross-validation. Pour chaque base de données 5 exécutions ont été effectués. Cette base est divisée en un ensemble d'entraînement qui représente 80 % et un ensemble de teste de 20 %. Chaque ensemble d'entrainement est partitionnée en 3 sous-ensembles. Par conséquent, un total de 25 x 3 analyses pour 98 jeux de données sera pré-traité avec notre approche ainsi que 7 d'autres filtres de bruit. Ce qui donne 58800 exécutions à partir desquelles les résultats obtenus sont analysés dans ce chapitre. D'autres expériences sont également menées afin d'évaluer l'influence de certains paramètres sur la précision de la classification.

Bases de données	$\#Instances$	$\#Attributs$	$\#Classes$
Balance	625	4(4/0)	3
Car	1728	6(0/6)	4
Dermathology	366	33(1/32)	6
Ecoli	336	7(7/0)	8
Ionosphere	351	33(33/0)	2
Iris	150	4(4/0)	3
new-thyroid	556	6(6/0)	2
Penbased	10992	16(16/0)	10
Pima	768	8(8/0)	2
Shuttle	58000	9(9/0)	7
Splice	3190	60(0/60)	3
Wdbc	569	30(30/0)	2
Zoo	101	16(0/16)	7

Tableau 4.6 – Description des bases de données utilisées dans notre expérimentation

4.3.4 Paramètres et méthodes utilisées

Parmi la vaste gamme de classifieurs disponibles dans la littérature, trois classifieurs ont été sélectionnés pour les utiliser dans notre algorithme (AD, SVM et logistique). Cette sélection est conforme à l'étude présentée dans la dernière section, dont on a comparer la precision de la classification pour les différentes combinaisons possibles. Le tableau 4.7 présente les paramètres utilisés pour ces trois classifieurs.

Le seuil T (utilisé dans notre algorithme) est égal à 1% de la taille du jeu de données d'entraînement original E (c.-à-d. T = $\|E\|$ x 0.01). De plus, le pourcentage de bons

Tableau 4.7 – Paramètres utilisées pour les classifieurs utilisés dans notre expérimentation

Classifieurs	*Paramètres*
AD	ConfidenceFactor : 0.25, minNumObj : 2
SVM	Gamma : 0, eps : 0.001, cost : 1, degree : 3, kernel : exp(-gamma $*\|u-v\|^2$)
LOG	maxIts : -1, ridge : 1.0E-8

exemples retiré de l'ensemble d'entraînement à chaque itération est égale à 50 % (β = 50 %). Dans ce travail, tous les algorithmes ont été mis en œuvre avec Java et la suite de logiciels d'apprentissage automatique WEKA 3.6.

4.3.5 Mesure de validation et vérification

Il existe beaucoup de mesures de validation permettent d'évaluer les performances d'un classifieur, la mesure utilisée dans nos expérimentation est la précision, c'est une mesure très connues dans la littérature. Elle est généralement calculée à partir de la matrice de confusion. Le Tableau 4.8 représente une matrice de confusion pour une classification binaire.

	Classe prédite positives	**Classe prédite négatives**
Classe positives	VP	FN
Classe négatives	FP	VN

Tableau 4.8 – Matrice de confusion

Une matrice de confusion contient des informations sur les résultats réels et prévus donnés par un classifieur.

— Vrai Positif (VP) : le nombre de bonnes prédictions positives ;

— Vrai Négatif (VN) : le nombre de bonnes prédictions négatives ;

— Faux Positifs (FP) : le nombre d'erreurs sur les prévisions positives ;

— Faux Négatif (FN) : le nombre d'erreurs sur les prédictions négatives.

A partir de la matrice de confusion, on peut calculer la précision 4.3 Dans la formule suivante, VN (vrai négatif) correspond à des échantillons correctement rejetés, VP (vrai positif) à des échantillons correctement identifiés, FP (faux positif) correspond à des

échantillons mal identifiés et FN (faux négatif) signifie des échantillons mal rejetés.

$$Précision = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.3}$$

4.3.6 Résultats expérimentaux et analyse

Cette étude expérimentale vise à analyser l'efficacité de notre architecture proposée à l'aide d'une vaste collection de jeux de données du monde réel. Nous étudions sa robustesse à différents niveaux de bruit de classe par rapport à d'autres approches récentes. Nous avons également modifié certains paramètres pour connaître ceux qui conviennent le mieux avec notre algorithme et qui nous permettent d'obtenir une précision de classification meilleur.

4.3.6.1 Comparaison entre notre filtre et les méthodes de filtrage de bruit existantes

Nous comparons la robustesse de notre filtre avec 8 autres filtres proposés dans la littérature (CF, EF, IPF, INFFC, ME, NCNE, ENN and AllKNN) pour différents niveaux de bruit. La formule de la précision 4.3 utilisée, la performance du filtre proposé, est calculée à l'aide d'une matrice de confusion.

Les résultats de ces expériences est présenté dans le tableau 4.9 et la figure 4.4. Les meilleurs résultats pour chaque niveau de bruit sont mis en gras.

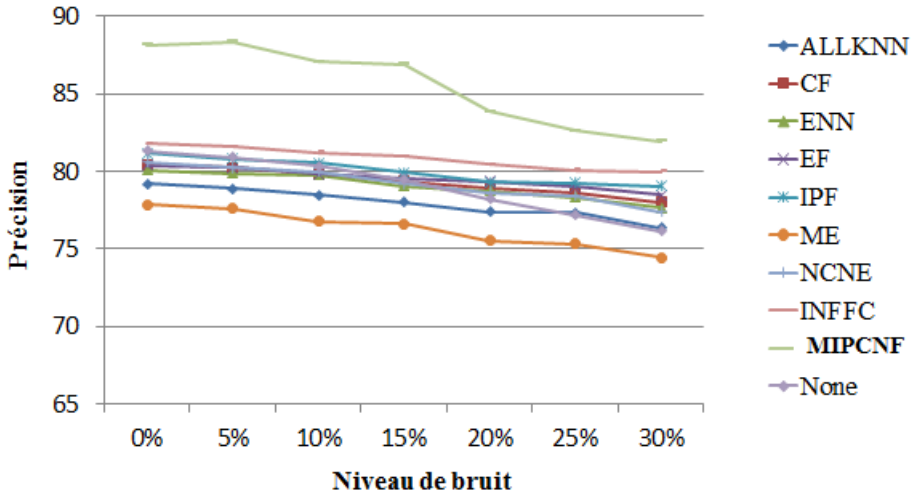

Figure 4.4 – Précision de la classification de chaque filtre à différents niveaux de bruit

Comme on peut le constater, notre filtre MIPCNF donne les meilleurs résultats à

Méthodes	Précision						
	0%	5%	10%	15%	20%	25%	30%
ALLKNN	79.20	77.65	77.78	77.98	76.86	76.32	76.00
CF	80.23	80.02	79.38	79.07	78.67	78.13	77.81
ENN	80.02	79.57	79.35	79.1	78.76	78.33	77.36
EF	80.33	80.16	79.74	79.38	79.29	78.82	78.12
IPF	81.05	80.68	80.39	79.71	79.16	79.08	78.91
ME	77.65	77.43	76.55	76.38	75.36	75.02	74.00
NCNE	80.15	79.73	79.52	78.72	78.19	78.01	76.97
INFFC	81.56	81.35	81.00	80.75	80.21	79.91	79.78
MIPCNF	**87.98**	**87.83**	**86.81**	**86.73**	**83.64**	**82.47**	**81.93**
None	81.00	80.67	80.16	79.25	78.01	76.98	76.02

Tableau 4.9 – Précision de la classification de chaque filtre à différents niveaux de bruit (les meilleurs résultats sont mis en gras)

tous les niveaux de bruit (de 5% à 30%), y compris en cas des données nettoyées (0 % de bruit). On obtient des bons résultats dans les bases de données les plus grands (penbased, shuttle,...) ainsi que pour les plus petites (iris, zoo, monk, ...). Les résultats expérimentaux montrent également que le MIPCNF présente un niveau de précision élevé par rapport aux autres filtres concurrents. En effet, les résultats du filtre INFFC suivis par IPF et EF sont également remarquables, mais restent inférieurs à ceux de notre solution MIPCNF. ME et AllKNN peuvent être considérés comme les filtres ayant la précision la plus basse.

4.3.6.2 L'impact du seuil du filtre sur la précision de la classification

Durant notre processus plusieurs itérations ont été établit. L'élimination du bruit s'arrête lorsque le critère d'arrêt est atteint : T = X.0.01, où X est la taille du jeu de données d'apprentissage.

Nous comparons la précision de la classification de notre filtre en variant la valeur du seuil et à différents niveaux de bruit. Les résultats obtenus sont illustrés au tableau 4.10 et à la figure 4.5.

Niveau de bruit	Seuil		
	0.01	0.02	0.03
5%	87.54	84.93	81.96
10%	86.90	84.75	81.79
15%	84.64	81.89	79.72
20%	83.19	77.93	74.83
25%	80.52	74.45	70.51
30%	78.15	70.32	68.85

Tableau 4.10 – la précision de la classification en utilisant différentes valeur du seuil

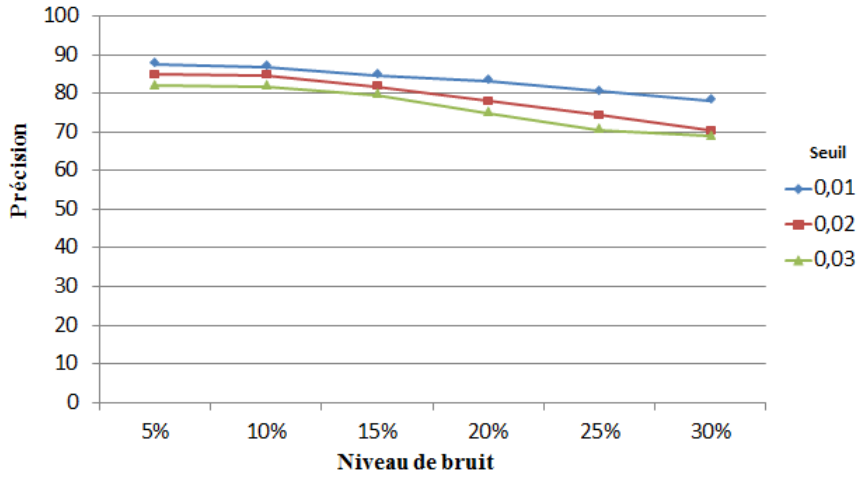

Figure 4.5 – L'impact de la valeur du seuil sur la précision de la classification

Le tableau 4.10 et la figure 4.5 montrent que lorsque le seuil est égal à 0,01, nous obtenons une précision plus élevée que les autres valeurs. En effet, pour atteindre le critère d'arrêt, nous devrions ajouter plus d'itérations, ce qui signifie plus de précision. Pour des valeurs de seuil élevées, le critère d'arrêt n'est atteint qu'en quelques itérations. Par ailleurs, nos résultats expérimentaux indiquent que lorsque le niveau de bruit est relativement faible, les résultats obtenus sont en principe suffisamment bons par rapport aux méthodes existantes. Cependant, lorsque le niveau de bruit devient élevé, il est nécessaire d'effectuer plus d'itérations jusqu'à ce que le nombre de bruits de classe identifié à chaque itération soit inférieur au seuil.

4.3.6.3 L'impact des différents combinaison des classifieurs sur la précision de la classification

Il existe plusieurs classifieurs en DataMining qui peuvent être utilisés pour la détection de bruit. Dans notre étude, nous avons sélectionné 5 classifieurs, les plus courants pour détecter le bruit de classe : AD, SVM, logistique, Naive Bayes (NB), RandomForest (RF). Dans cette section, nous comparons la précision de la classification en combinant trois classifieurs. Avec ces 5 classifieurs, nous avons 10 combinaisons possibles de trois classifieurs. Le tableau 4.11 présente les résultats expérimentaux obtenus.

Les résultats de l'expérimentation montrés dans le tableau 4.11 et Figure 4.6; nous permet de conclure que la combinaison de ces trois classifieurs (AD, SVM et logistique) est la plus recommandée pour l'utiliser dans notre algorithme.

Combinaison	AD	SVM	Logistic	NB	RF	Précision
1	x	x	x			88.88
2	x	x		x		66.66
3	x	x			x	63.63
4	x		x	x		72.72
5	x		x		x	77.77
6		x	x	x		81.81
7		x	x		x	87.50
8		x		x	x	80.00
9			x	x	x	85.71
10	x			x	x	68.75

Tableau 4.11 – La précision de la classification pour les différentes combinaisons possibles avec 30% de bruit et un seuil T = 0,01

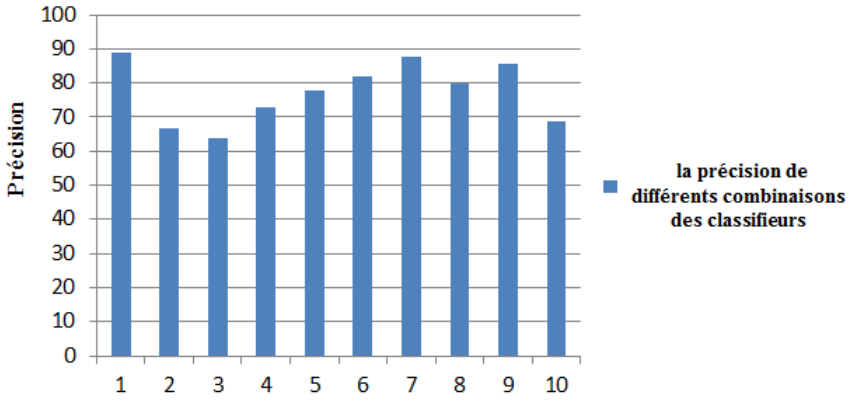

Figure 4.6 – La précision de la classification pour les différentes combinaisons possibles

4.3.6.4 *L'impact des méthodes de vote sur la précision de la classification*

En cas d'une classification en utilisant une combinaison des classifieurs, une instance est classée comme bruit en combinant les votes des différentes prédictions des classifieurs. Différentes méthodes de combinaison sont comparées dans cette partie : product, min, max, median, average rules et le vote majoritaire.

Le tableau 4.12 ainsi que la figure 4.7 montrent une comparaison de la précision de la classification des différentes règles de combinaison en utilisant la base de données Pima avec 30% de bruit et un seuil T = 0,01.

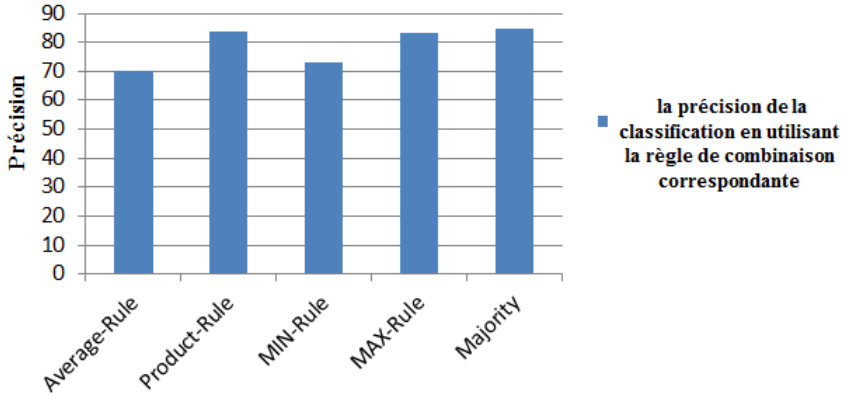

Figure 4.7 – Précision de la prévision en utilisant différentes règles de combinaison

	Average-Rule	**Product-Rule**	**MIN-Rule**	**MAX-Rule**	**Majority**
Précision de la classification	70.13	83.92	72.95	83.09	84.85

Tableau 4.12 – La précision de la classification en utilisant les différentes règles de combinaison pour la base de données Pima avec 30% de bruit et un seuil T = 0.01

Selon le résultat de l'expérimentation, on peut constater que le vote majoritaire donne les meilleurs résultats. C'est d'ailleurs la règle de combinaison choisie pour être utilisée dans notre filtre dans toutes nos expérimentations.

4.3.7 Conclusion

Le bruit de classe est un phénomène complexe qui peut avoir de nombreuses conséquences négatives sur les performances de la classification. Dans ce chapitre on a présenté notre nouvelle approche de filtrage du bruit appelée Multi-Iterative Partitioning Class Noise Filter (MIPCNF). Le but du MIPCNF est de résoudre les problèmes majeurs rencontrés dans ce domaine, principalement améliorant la précision de la classification.

Notre approche consiste à détecter et à éliminer le bruit de classe. Ce dernier est éliminé par l'application d'un processus de détection de bruit de classe en plusieurs itérations, sur différentes partitions des données et à l'aide d'un ensemble de classifieurs. Notre nouvelle approche de filtrage du bruit, MIPCNF, combine plusieurs stratégies de filtrage permettant d'augmenter la précision de la classification. Nous calculons la pré-

cision de la classification avec l'existence d'un bruit de classe à différents niveaux. Notre solution, testé sur une vaste collection de bases de données du monde réel, surpasse tous les filtres connus de la littérature (CF, EF, IPF, INFFC, ME, NCNE, ENN et AllKNN), dans différents niveaux de bruit de classe. Cela montre l'efficacité et la robustesse de notre méthode proposée. En effet, la plupart des filtres étudiés dans cette expérimentation éliminent des quantités plus élevées d'exemples non bruités par rapport à notre méthode (MIPCNF). Cependant, notre approche est capable d'éliminer les exemples bruyants même en un niveau de bruit très élevé. Nous concluons que l'approche proposée contribue à une élimination plus efficace du bruit de classe et réalise le meilleur nettoyage des données dans le contexte de données à grande échelle.

CHAPITRE 5

ARCHITECTURE POUR LA SÉLECTION D'ATTRIBUTS BASÉE SUR UNE APPROCHE DISTRIBUÉE HORIZONTALEMENT ET VERTICALEMENT

Sommaire

5.1 Introduction

La taille croissante des jeux de données soulève de grands défis pour l'apprentissage supervisé et non supervisé. Comme on peut le constater, le nombre de données ne cesse de croître à la fois dans les attributs et le nombre d'exemples. En conséquence, la gestion des fonctionnalités non pertinentes et redondantes est devenue un véritable défi. Une dimensionnalité élevée implique des besoins massifs en mémoire ainsi qu'un coût élevé de calcul pour le processus d'apprentissage. La taille des données peut être mesurée selon deux dimensions, le nombre de variables et le nombre d'exemples. Ces deux dimensions peuvent prendre des valeurs très élevées, ce qui peut poser un problème lors de l'exploration et l'analyse de ces données. Pour cela, il est fondamental de mettre en place des outils de traitement de données permettant une meilleure compréhension de la valeur des connaissances disponibles dans ces données. La réduction des dimensions est l'une des plus vieilles approches permettant d'apporter des éléments de réponse à ce problème. Son objectif est de sélectionner ou d'extraire un sous-ensemble optimal de caractéristiques pertinentes pour un critère fixé auparavant. La sélection de ce sous-ensemble de caractéristiques permet d'éliminer les informations non pertinentes et redondantes selon

le critère utilisé. Cette sélection/extraction permet donc de réduire la dimension de l'espace des exemples et rendre l'ensemble des données plus représentatif du problème. En effet, les principaux objectifs de la réduction de dimension sont :

— faciliter la visualisation et la compréhension des données,

— réduire l'espace de stockage nécessaire,

— réduire le temps d'apprentissage et d'utilisation,

— identifier les facteurs pertinents.

En conséquence, le besoin des techniques de réduction de dimensionnalité a considérablement augmenté ces dernières années afin de disposer d'un petit nombre d'exemples et / ou d'un petit nombre d'attributs. L'une de ces techniques est la sélection d'attributs. En effet, la sélection des attributs est un sujet de recherche très actif qui répond au problème de la réduction de la dimensionnalité en recherchant l'ensemble des attributs le plus compact et le plus informatif. En d'autres termes, la sélection d'attributs est définie comme le processus d'identification et de suppression d'attributs non pertinents et redondants dans le but de trouver la meilleure collection de sous-ensembles d'attributs sans perte significative d'informations utiles ni dégradation des performances. Au cours des dernières années, la sélection d'attributs a été appliquée avec succès dans différents domaines et avec différentes stratégies (figure 5.1) pour améliorer la précision de la classification. Traditionnellement, les méthodes de sélection d'attributs ont été conçues pour fonctionner dans un environnement informatique centralisé. Cependant, il n'est pas rentable de traiter l'ensemble des données à la fois. D'autre part, la plupart des algorithmes de sélection d'attributs existants ne peuvent pas gérer une grande quantité de données. En d'autres termes, leur efficacité peut être considérablement détériorée lorsqu'il s'agit de données énormes, au point de devenir inapplicable. Par conséquent, au cours des dernières années, deux grandes catégories de méthodes distribuées ont été développées au lieu des approches centralisées : les approches horizontales et verticales. Les données peuvent être distribuées soit horizontalement (par instances) (Zhu et Xindong, 2004; Wei et Stephen, 2007; Wirth et Jochen, 2000), soit verticalement (par attributs) (Adriaans et Zantinge, 1996; Kim et Ryu, 2004; Murty et Devi, 2011). Ainsi, le problème de la grande dimensionnalité peut être résolu autant que possible.

5.2 Motivations

L'approche standard de l'exploration de données aujourd'hui est l'approche centralisée. Les données qui pourraient en fait avoir été collectées à plusieurs endroits sont rassemblées à un seul endroit. Dans cet endroit central, un modèle est calculé à partir

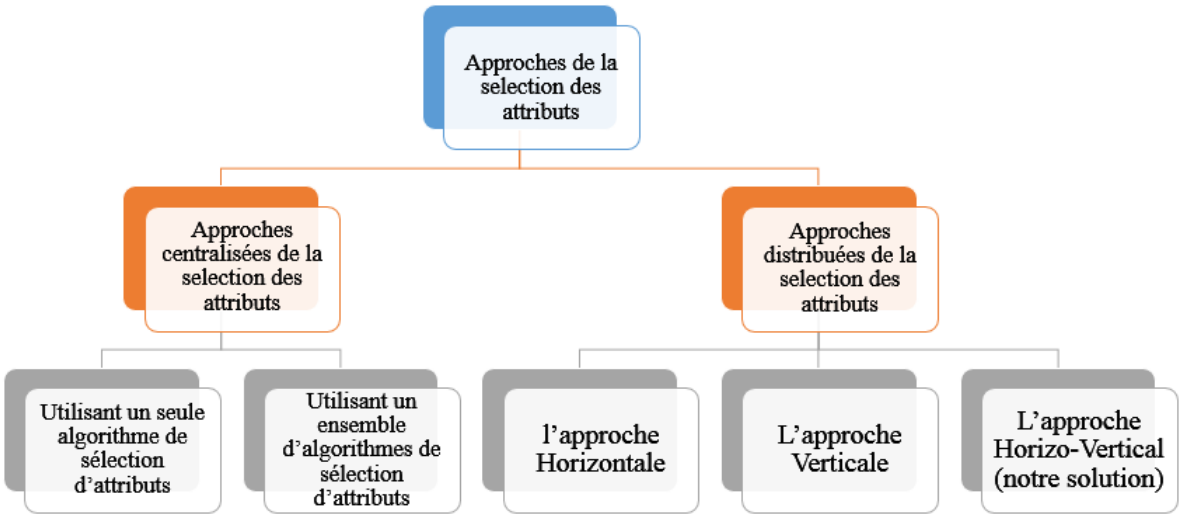

Figure 5.1 – Les Différents approches de sélection d'attributs

des données. Bien que ce processus soit facile à comprendre, et la conception du logiciel d'exploration de données est simple, l'approche centralisée présente plusieurs inconvénients :

— Le mouvement des données du point S où elles sont collectées à un emplacement central nécessite de la bande passante et du temps.

— Toutes les données sont visibles à l'emplacement central, préservant ainsi leur vie privée. Dans certaines situations, il peut ne pas être possible de centraliser les données en raison des restrictions légales découlant de la préoccupation de la vie privée. De sorte que certaines formes d'analyse ne peuvent être réalisées d'une façon centralisée.

Avec l'augmentation de la taille des bases de données, les modèles deviennent de plus en plus complexes. Il est naturel d'envisager de remplacer l'approche centralisée par des techniques distribuées, comme un moyen de réduction des coûts. Les données sont naturellement collectées à partir de différents endroits physiques géographiquement séparés, par exemple :

— Les entreprises collectent des informations sur leurs clients via des magasins physiques, sites Web et centres d'appels, qui sont généralement situés à différents endroits, peut-être même dans différents pays.

— Les organisations construites autour d'un réseau collectent toujours des données de manière distribuée, par exemple : les entreprises de téléphonie mobile obtiennent des informations sur les appels passés via la cellule dans laquelle ils ont été initiés.

— Les astrophysiciens découvrent les étoiles et les galaxies en observant les dis-

positifs situés à différentes latitudes et longitudes, et via des périphériques qui rassemblent les données sur les mêmes objets à différentes longueurs d'onde.

— Les réseaux de capteurs comportent des dispositifs pouvant être de différents types, placés dans de différents endroits. Chaque capteur voit seulement une petite fraction de la totalité des données collectées ; leurs produits doivent être intégrés pour donner une vision globale cohérente d'image.

Si nous voulons exploiter un traitement parallèle et distribué pour l'exploration de données, nous devons alors sauvegarder et examiner les données collectées, pas seulement attendre toujours qu'elles soient toutes au même endroit. En particulier, faire une analyse sur les données aux endroits où elles sont collectées introduit le traitement distribué de manière naturelle, et évite les goulots d'étranglement liés aux performances. Cette approche est particulièrement intéressante s'il y a suffisamment de traitement puissant et sophistiqué aux emplacements où les données sont rassemblées pour construire des modèles locaux, qui peuvent ensuite être déplacés à l'emplacement central et fusionnés pour produire un modèle global cohérent. Cette approche vient exactement pour compenser les faiblesses de l'exploration de données centralisées :

— Il n'est plus nécessaire de déplacer les données brutes des points de collecte sur un site central.

— Le traitement est fait sur chaque site local, de sorte que la charge du calcul est répartie sur de nombreux processeurs.

Cette approche permet une exploration de données préservant la confidentialité. Par exemple dans le monde réel, les compagnies d'assurance concurrentes pourraient encore partager leurs modèles de fraude, et ainsi réduire les coûts de fraude sans révéler les informations sensibles. Cependant, il existe plusieurs façons de collecter les données distribuées, et celles-ci nécessitent des approches différentes de construction de modèle.

La première forme de distribution est le partitionnement horizontal, différents enregistrements sont collectés sur différents sites, mais chaque enregistrement contient tous les attributs de l'objet qu'il décrit. C'est la manière la plus courante et la plus naturelle de la collecte des données. Par exemple, une entreprise multinationale travaille avec des clients dans plusieurs pays, collectant des données des différents clients dans chaque pays.

La deuxième forme de distribution est le partitionnement vertical, dont lequel des différents attributs du même ensemble d'enregistrements sont collectés sur différents sites. Chaque site recueille les valeurs d'un ou de plusieurs attributs pour chaque enregistrement, par conséquent, chaque site a une vue différente des données. Par exemple, un réseau de capteurs collecte (généralement) un attribut par chaque capteur. Une société de cartes de crédit peut collecter des données sur les transactions effectuées par le même

client dans différents pays, et peut vouloir aussi traiter les transactions en différents pays en tant que différents aspects de la consommation totale du client.
Les données partitionnées verticalement sont encore rares, mais elles sont devenues plus communes et importantes.

Dans ce travail, nous proposerons une architecture parallèle pour la sélection d'attributs en répartissant les données verticalement et horizontalement, appelée HVDFS, dans le but d'obtenir une bonne performance de classification tout en réduisant la dimensionnalité des données d'entrées. Visant à améliorer la performance de classification ainsi que réduire le nombre d'attributs et par conséquence une réduction de dimension. L'idée est de diviser les données par instances et par attributs. Après avoir réparti les données en petits sous-ensembles, un filtre est appliqué sur différentes partitions des données. Ces deux premières étapes sont répétées plusieurs fois pour obtenir un ensemble stable d'attributs. Ensuite, une procédure de fusion est effectuée qui combine les résultats partiels en un seul sous-ensemble d'attributs pertinents selon un seuil pré-calculé. Ensuite, l'ensemble d'attributs sélectionné par l'algorithme horizontal et celle par l'algorithme vertical sont combinés pour obtenir l'ensemble final des attributs.
L'efficacité de notre approche est démontrée en utilisant trois bases de données bien connues. Comparée à l'approche centralisée et aux autres approches distribuées. Quatre classifieurs ont été utilisés pour évaluer la précision de la classification de ces approches.

5.3 Approche proposée pour la sélection d'attributs

Nous pouvons résumer la procédure de notre approche proposée en trois étapes comme elle est illustrée dans la figure 5.2 :

- Étape 1 : Partitionnement horizontale de l'ensemble des données, puis l'application d'une méthodes de sélection d'attributs pour la détection des attributs redondantes.
- Étape 2 : Partitionnement verticale de l'ensemble des données, puis l'application d'une méthode de sélection d'attributs pour la détection des attributs pertinents.
- Étape 3 : Application des classificateurs pour évaluer les attributs sélectionnés.

Nous partitionnons l'ensemble des données horizontalement avec une partition disjointe de la même taille. Nous appliquons ensuite un algorithme puissant de sélection d'attributs pour la sélection des attributs redondants, tels que le MRMR et le Relief, dans chacune de ces partitions. Ensuite, nous combinons les sorties partielles en un seul sous-ensemble d'attributs. Cette procédure sera répéter en plusieurs itérations afin de capturer suffi-

samment d'informations. L'ensemble des données résultat du premier filtrage est donné en entrée pour le deuxième processus. Nous partitionnons cet ensemble des données verticalement. Cette fois, nous appliquons un algorithme de sélection d'attributs qui est plus efficace en détection des attributs non pertinents. Ensuite, les attributs sélectionnés comme attributs redondants sont supprimés. Puis, nous combinons les sorties partielles de différentes partitions pour avoir le résultat final. Afin d'évaluer les performances de notre contribution, des classifieurs de différente famille sont utilisés, ce qui est détaillé d'avantage dans la section suivante.

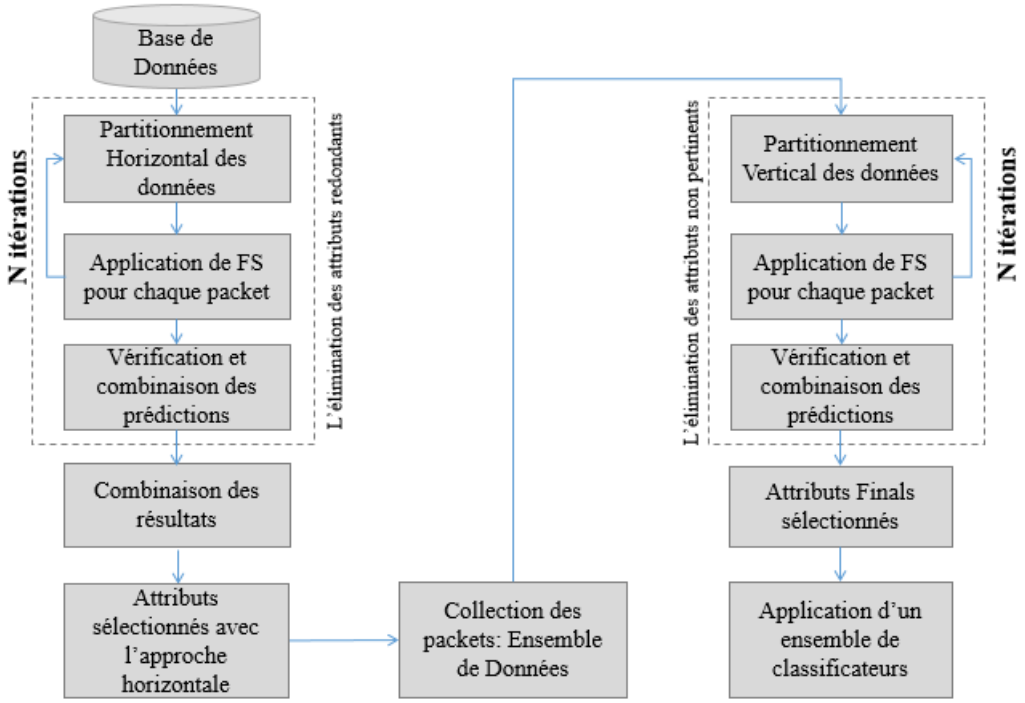

Figure 5.2 – Processus de notre approche HVDFS

Dans la section suivante nous montrerons que notre approche proposée permet de surmonter l'inconvénient commun chez toutes les approches existantes précédemment mentionnées.

5.4 Évaluation

5.4.1 Bases de données utilisées

Afin d'évaluer la performance et l'efficacité de notre approche, nous utilisons trois bases de données (Isolet, 11-Tumors et Modelon). Le tableau 5.1 présente le nombre des attributs et le nombre des instances, pour l'ensemble de données d'apprentissage et de

test, ainsi que le nombre de classes. Ces bases de données ont été divisées comme suit : 2/3 pour les données d'apprentissage et 1/3 pour les données de test. La base de donnée Isolet et 11-Tumors peuvent être téléchargées gratuitement à partir du référentiel UCI Machine Learning (Lichman, 2013), tandis que Modelon à partir du Gene Expression Model Selector (GEMS) (University, 2016).

Bases de données	#Attributs	#Instances	*#Classes*	#Références
Isolet	617	7800	26	(Lichman, 2013)
Madelon	500	2400	2	(University, 2016)
11-Tumor	12 534	172	11	(Lichman, 2013)

Tableau 5.1 – Description de bases de données utilisées dans l'expérimentation

Nous présentons brièvement les trois bases de données comme suit :

Isolet : La base de données Isolet est une base issue du domaine de la reconnaissance de la parole comporte près de 150 locuteurs parlant, 26 lettres de l'alphabet à deux reprises. Ce qui signifie que chaque locuteur a contribué à 52 exemples d'apprentissage ; Nous avons ainsi au total 7800 exemples qui sont décrits par 617 attributs. La tâche consiste à classifier une lettre prononcée en fonction de ces attributs, tels que les coefficients spectraux, les caractéristiques de contour et les caractéristiques sonorantes.

11-Tumor : Cet ensemble de données contient 172 instances et 12534 attributs de 11 différents types de tumeurs humaines, ce qui signifie un total de 11 classes.

Madelon : le Madelon est un problème de classification à deux classes, c'est une base de données comporte 2400 instances situés sur les sommets d'un hypercubes à cinq dimensions et 500 variables d'entrée continues attribuées à chaque sommet. Cela faisait partie du défi de la sélection des attributs de NIPS 2003.

5.4.2 Outils et méthodes utilisées

L'approche distribuée proposée ici peut être utilisée avec n'importe quel méthode de sélection d'attributs, ces algorithmes sont généralement moins coûteux que les wrappers ou les méthodes embarquées. Les méthodes les plus populaires sont décrits comme suit :

Chi-squared Ceci est un filtre univarié basé sur la statistique χ^2 (Liu et Setiono, 1995) qui évalue chaque caractéristique de manière indépendante par rapport aux classes. Plus la valeur du chi-carré est élevée, plus l'attribut est important par rapport à la classe.

Information Gain Le filtre Information Gain (Quinlan, 1986c) est l'une des méthodes les plus courantes d'évaluation des attributs. Ce filtre évalue les attributs en fonction de leur gain d'information et considère un seul attribut à la fois. Il fournit une classification ordonnée de tous les attributs, puis un seuil est requis pour sélectionner un certain nombre d'entre eux selon l'ordre obtenu.

Correlation-based Feature Selection (CFS) C'est un algorithme de filtrage multivarié simple qui classe les sous-ensembles selon la corrélation basée sur une fonction d'évaluation heuristique (Hall, 1999). Le biais de la fonction d'évaluation concerne les sous-ensembles qui contiennent les attributs qui sont fortement corrélés avec la classe et dé-corrélés les uns avec les autres. Les attributs non pertinents doivent être ignorés parce qu'ils auront une faible corrélation avec la classe. Les attributs redondants devraient être éliminés car ils seront fortement corrélés avec une ou plusieurs attributs restants. L'acceptation d'un attribut dépendra de la mesure dans laquelle il prédit les classes dans des zones de l'espace d'instance qui n'ont pas déjà été prédites par d'autres attributs.

Consistency-based Filter Ce filtre est basé sur la cohérence (Dash et Liu, 2003b). Il évalue la valeur d'un sous-ensemble des attributs par le niveau de cohérence dans les valeurs de la classe lorsque les données de traitement sont projetées sur le sous-ensemble d'attributs. A partir de l'espace des attributs, l'algorithme génère un sous-ensemble aléatoire S à chaque itération. Si S contient moins d'attributs que le meilleur sous-ensemble actuel, l'indice d'incohérence des données décrites par S est comparé avec l'indice d'incohérence dans le meilleur sous-ensemble. Si S est aussi cohérent ou plus cohérent de le meilleur sous-ensemble, S devient le meilleur sous-ensemble. Le critère de l'incohérence, qui est la clé du succès de cet algorithme, spécifie combien peut être la réduction de la dimension dans les données. Si le taux de cohérence des données décrites par les attributs sélectionnés est inférieur à un seuil défini, cela signifie que la réduction de dimension est acceptable. Notez que cette méthode est multivariée.

Fast Correlation-Based Filter (FCBF) La méthode de filtrage à corrélation rapide (Yu et Liu, 2003) est un algorithme multivarié qui mesure la corrélation attribut-classe et attribut-attribut. Le FCBF commence par sélectionner un ensemble d'attributs fortement corrélés avec la classe en fonction de l'incertitude symétrique (SU), définie comme le rapport entre le gain d'information et l'entropie de deux attributs. Ensuite, trois heuristiques sont appliquées qui suppriment les attributs redondants et gardent les attributs

les plus pertinents pour la classe. Le FCBF a été conçu pour les données de haute dimensionnalité et s'est avéré efficace pour éliminer les attributs non pertinents et redondants. Cependant, il ne prend pas en compte l'interaction entre les attributs.

INTERACT L'algorithme INTERACT (Zhao et Liu, 2009) utilise la même mesure de qualité en tant que filtre FCBF, c'est-à-dire SU, mais il inclut également la contribution de cohérence, qui est un indicateur sur la manière dont l'élimination d'un attribut affectera de manière significative la cohérence. L'algorithme est constitué de deux parties principales. Dans la première partie, les attributs sont classés en ordre décroissant en fonction de leurs valeurs SU. Dans la deuxième partie, les attributs sont évalués un par un à partir de la fin de la liste des attributs classées. Si la contribution de cohérence d'un attribut est inférieure à un seuil prédéfini, l'attribut est supprimé, sinon il est sélectionné. Les auteurs ont déclaré que cette méthode peut gérer l'interaction des attributs, et sélectionne efficacement les attributs pertinentes.

ReliefF Le filtre ReliefF (Kononenko, 1994) est une extension de l'algorithme Relief original. Le relief origine fonctionne en échantillonnant au hasard une instance à partir des données, puis localiser son plus proche voisin dans la même classe ainsi que dans la classe opposée. Les valeurs des attributs des voisins les plus proches sont comparées aux instances échantillonnées et utilisées pour mettre à jour les scores de pertinence pour chaque attribut. Un attribut utile doit différencier les instances de différentes classes et avoir la même valeur pour les instances de la même classe. ReliefF ajoute la capacité de traiter les problèmes multiclasse et il est également plus robuste et capable de traiter des données incomplètes et bruyantes. Cette méthode peut être appliquée dans toutes les situations à un biais faible, inclut l'interaction entre les attributs, et peut capturer les dépendances locales que d'autres méthodes manquent.

Minimum Redundancy Maximum Relevance (MRMR) La méthode MRMR (Peng *et al.*, 2005) sélectionne les attributs qui ont le plus de pertinence avec la classe cible et qui sont également les moins redondants, c'est-à-dire qui sélectionnent des attributs qui sont au maximum dissemblables. Les deux critères d'optimisation (Maximum-Relevance et Minimum-Redundancy) sont basés sur des informations mutuelles.

Dans nos expériences, nous avons choisi l'algorithme CFS pour détecter les attributs non pertinents et MRMR pour détecter les attributs redondants, conformément à la re-

commandation de (Oreski *et al.*, 2017).

Pour tester l'efficacité de notre proposition, nous avons sélectionné quatre classifieurs largement utilisés : Arbre de décision, naïfs Bayes, kNN et SVM (des détails supplémentaires sont décrits dans le chapitre de l'état de l'art). Nous comparons également notre proposition avec l'approche horizontale et verticale proposées dans (Moran-Fernandez *et al.*, 2017), ainsi qu'avec l'approche centralisée, en termes de nombre d'attributs sélectionnés et de précision de la classification.

Pour les approches de partitionnement on aura besoin de choisir une méthode de combinaison pour les différents résultats partiels, pour cette raison, nous avons exécuté notre approche avec différents méthodes de combinaison afin de trouver celle donnant la meilleure précision de classification.

Les expériences décrites ici ont été menées sur un PC Windows 8 doté d'un processeur Intel Core i5 et 1,8 GHz. Tous les algorithmes ont été mis en œuvre avec l'environnement Matlab 7.0 et WEKA DataMining Package version 3.6.

5.5 Résultats expérimentaux et analyse

Dans cette section, nous présentons les résultats expérimentaux en termes de précision de la classification et du nombre d'attributs sélectionnés. Quatre approches différentes ont été comparées : l'approche centralisée, l'approche basée sur une distribution horizontale, verticale et horizo-verticale. Pour chacune de ces stratégies, nous évaluons la précision de la classification à l'aide de quatre classifieurs AD, SVM, NB et kNN. Les méthodes de combinaisons utilisées dans notre expérimentation sont : consensus, majorité, mesure de la complexité et méthode basée sur un seuil (valeur logarithmique suggérées par (Yu et Liu, 2004)).

5.5.1 Comparaison entre l'approche horizontal, vertical et l'horizo-vertical

5.5.1.1 En terme de précision de la classification

Nous développons une nouvelle approche capable de gérer efficacement les attributs non pertinents et redondants, ce qui signifie réduire la dimensionnalité avec une meilleur sous ensemble d'attributs pertinents, par conséquent une bonne précision de classification. Le tableau 5.2 montre les résultats obtenus par les algorithmes (AD, NB, kNN et SVM) en termes de précision de la classification pour l'approche centralisée, l'approche horizontale, verticale et horizo-verticale.

	Isolet				11-Tumor				Modelen			
	H	V	HV	C	H	V	HV	C	H	V	HV	C
AD	85	86	97	82	88	89	99	88	89	89	89	88
NB	80	85	98	81	86	87	94	86	86	85	87	86
SVM	82	86	98	82	88	88	99	87	87	89	90	86
KNN	83	86	98	82	88	89	99	87	88	90	91	87

Tableau 5.2 – Précision de classification obtenue par chaque filtre sur les trois bases de données

Le tableau 5.2 montre la précision de classification obtenue par chaque filtre sur les trois bases de données en utilisant le vote majoritaire. Comme nous pouvons le constater, les meilleures performances de classification ont été obtenues avec notre approche HVDFS, pour toutes les mesures de classifieurs.

5.5.1.2 *En terme du nombre des attributs sélectionnés*

Le tableau 5.3 indique le nombre d'attributs sélectionnés par l'approche centralisée ainsi que les approches de distribution horizontale, verticale et horizo-verticale. Comme on peut le constater, le nombre d'attributs sélectionnés par l'approche verticale était supérieur à celui sélectionné par les autres approches. La raison derrière est que, avec le partitionnement verticale, les attributs ont été répartis sur des paquets. D'où la détection de la redondance des attributs sera plus difficile s'ils se trouvaient dans des partitions différentes. En revanche, il n'y a pas une différence significative entre le nombre d'attributs sélectionnés par l'approche horizontale et l'approche centralisée. D'autre part, l'ensemble d'attributs sélectionnés par notre approche était plus petit par rapport aux autres approches.

Nombre des attributs	**Isolet**	**11-Tumor**	**Modelen**
L'approche horizontale	93	236	15
L'approche verticale	200	455	23
L'approche Horizo-vertical	74	157	10
L'approche Centralized	125	312	18

Tableau 5.3 – Nombre d'attributs sélectionnés par les quatre approches

Les chercheurs affirment qu'un grand nombre d'attributs n'est pas nécessairement plus informatives, car celle-ci risquent d'être non pertinents ou redondants (Xing *et al.*, 2001). Par conséquent, il est essentiel de choisir un petit nombre d'attributs pertinents parmi un grand nombre d'attributs de manière juste et raisonnable pour une classification efficace. Nous pouvons en déduire que notre algorithme HVDFS permet de réduire

le nombre des attributs, et par conséquent la dimensionnalité des données, tout en améliorant la qualité de la classification.

5.5.1.3 *L'impact de la méthode de combinaison des résultats sur la qualité de la classification*

Nous avons comparé différentes stratégies de combinaison de résultats, en utilisant notre approche en terme de la précision de classification, les méthodes de combinaison testées dans cette expérimentation sont : le vote basé sur des mesures de complexité, le vote majoritaire, le vote par consensus et le vote basé sur une mesure logarithmique. Comme il est montré dans le tableau 5.4, la meilleure précision de classification a été obtenue en cas de l'utilisation d'un vote basé sur la mesure logarithmique. Le vote basé sur la mesure de la complexité nécessite un temps d'exécution élevé. Il est donc préférable de fixer un seuil et éviter d'effectuer un calcul spécifique.

	Majoritaire	Consensus	Logarithmic	mesure de Complexité
Horizontal	85	83	87	86
Vertical	88	85	88	87
Horizo-vertical	92	89	95	90

Tableau 5.4 – La précision de la classification pour différentes stratégies de combinaison de résultats

5.5.2 Discussion

Les expérimentation faites sur trois bases de données, avec un nombre d'attributs variant de 500 à 12534 et d'instances entre 172 et 7779, ont montré que notre approche permet d'élaborer une réduction de dimensionnalité des données tout en améliorant la qualité de classification.

L'efficacité de la majorité des algorithmes existants de sélection d'attributs s'est détériorée de manière notable en cas d'augmentation de la taille des données. Cette sélection peut être très coûteuse et parfois irréalisable. C'est pourquoi il est vivement recommandé de diviser les ensembles de données par l'une des approches distribuées, en fonction du facteur déterminant la complexité du problème (attributs, instances ou les deux).

En suivant les recommandations de (Moran-Fernandez *et al.*, 2017), l'approche verticale présente l'inconvénient de ne pas gérer les entités redondantes. En effet, avec la partition verticale, les attributs étaient ainsi réparties entre les paquets ; il sera plus difficile de détecter la redondance entre eux. Par contre, la partition horizontale n'est pas recommandée, si une meilleure performance de classification est plus importante que

des exigences de minimisation de stockage. Notre approche peut être utilisée avec n'importe quel algorithme de sélection d'attributs selon la nature des données. Pour résumer brièvement notre recommandation, nous pouvons dire qu'il est préférable d'utiliser l'approche horizontale lorsque on a un grand nombre d'instances, alors qu'il est recommandé d'utiliser l'approche verticale lorsque on a un grand nombre d'attributs. Alors que notre approche est plus bénéfique dans le cas d'un grand nombre d'instances et d'attributs au même temps.
Noté bien qu'un ensemble de données avec n attributs et m instances est classé comme petit ou grand en fonction de la nature des données et du domaine auquel ils appartiennent.

5.6 Conclusion

Un grand nombre d'attributs et d'instances peut entraîner un sur-ajustement des données (Moran-Fernandez *et al.*, 2017). Dans cette nouvelle approche, nous avons choisis d'aborder la méthode distribuée de sélection d'attributs. Il s'agit d'un problème important qui a une influence directe sur la qualité de la classification du modèle. Avec notre algorithme, le processus distribué de sélection d'attributs proposé est capable de surmonter les inconvénients des approches existantes mentionnées précédemment.
Dans ce travail, nous avons proposé une nouvelle méthode distribuée de sélection d'attributs. L'approche proposée permet de distribuer les données avec succès en utilisant à la fois les attributs et les instances, réduisant ainsi l'ensemble d'attributs sélectionnés et obtenant une meilleure performance.

Chapitre

6 Conclusion et travaux futures

Le prétraitement des données est souvent la tâche la plus coûteuse dans le processus d'extraction de connaissances à partir des données. Le prétraitement des données est un vaste domaine. La tendance actuelle, est de traiter de grands ensembles de données et d'obtenir des modèles de plus en plus intelligibles. Deux volets du prétraitement ont été traités dans ce mémoire : la réduction de dimension et le nettoyage des données. Nous allons à présent rappeler les contributions principales de cette thèse à ces deux thèmes d'étude.

1. Dans le cadre de cette thèse, nous avons pu identifier différents problèmes liés à la détection et l'élimination de bruit de classe. Nous avons commencé en proposant une architecture pour la détection de bruit de classe qui s'appuie sur l'extraction des règles d'associations pour une classification des données.
2. Une autre approche de détection et d'élimination de bruit de classe a ensuite été proposée ; cette approche est une amélioration de l'architecture précédente. Un filtre nommé MIPCNF a été proposé basé sur une élimination itérative de bruit pour des données partitionnées en petits sous-ensembles. Ce filtre a pu montrer sa capacité de détecter le bruit de classe avec différents niveaux de bruit, tout en gardant une bonne qualité de classification.
3. Après avoir contribué dans le domaine du nettoyage des données, nous avons abordé le problème de la réduction de dimension en utilisant la sélection des attributs. Nous avons tout d'abord dressé les problématiques rencontrées dans le contexte de la sélection des attributs pour les bases de données à grandes échelles. Ce prétraitement est important, c'est cette phase qui conditionne la qualité des modèles établis en fouille de données. La sélection des variables permet à la fois de supprimer les variables redondantes ainsi que le bruit engendré par certaines variables. Ainsi, de réduire la taille de l'espace de représentation des données, en effet, après un processus de sélection, seuls les attributs pertinents sont conservés. Notre contribution dans ce domaine a été de mettre en place un nouveau Framework, combinant les techniques de sélection d'attributs distribuées

horizontalement et verticalement, appelée HVDFS. Visant à réduire le nombre des attributs et par conséquent une réduction de dimension, tout en améliorant la performance de classification.

Ce mémoire de thèse n'a bien sûr pas pour but de résoudre de manière définitive le vaste problème de prétraitement et d'exploitation des données. Il reste encore de nombreux travaux à mener, nous proposons maintenant quelques pistes de recherche ouvertes qui pourraient compléter nos travaux. Plusieurs voies peuvent être envisagées dans le sillage de ce travail. Nous en esquissons ici celles qui paraissent les plus pertinentes : Vu que la plupart des données réel ne sont pas des nombres, nous envisageons appliquer nos contributions à des données textuelles, et d'autre part, adapter nos deux derniers contributions pour utiliser l'apprentissage automatique et profond (deepLearning).

Parmi nos travaux à long terme et après avoir traité deux étapes du prétraitement nous pensons par la suite à traiter une autre étape aussi intéressante que les étapes du prétraitement déjà abordées dans cette thèse. C'est la phase d'intégration des données. L'intégration de données est le problème de combiner des données résidant à différentes sources. Les scientifiques de données consacrent un temps relativement important à la phase de préparation et d'intégration des données, ce qui montre que l'intégration des données reste un défi majeur dans les applications du monde réel actuelles. L'objectif est d'utiliser les données comme si elles ne formaient qu'une seule base. Il semble pertinent d'étendre notre pondération au domaine d'intégration de données et couvrir les différentes solutions proposées, les différentes architectures distribuées et les modes d'accès aux données. Afin de pouvoir par la suite proposer une contribution dans ce domaine. Les Services Web posent des défis importants pour automatiser les applications individuelles basées sur des composants logiciels hétérogènes et inter organisationnels. Bien que les services Web, comme Representational State Transfer (REST) et Simple Object Access Protocol (SOAP)), aient été conçus pour prendre en charge une communication interopérable entre applications, l'expérience montre qu'il s'agit d'un objectif assez difficile à atteindre. Ce qui soulève des préoccupations lors de la mise en œuvre de services avec des exigences de fiabilité. Beaucoup de problèmes peuvent être rencontrés dans une communication entre des applications différentes :

1. Interopérabilité au niveau de messages SOAP.
   - Incompatibilités de domaine.
   - Incompatibilités de définition d'entité.
   - Incompatibilités de l'abstraction.
2. Interopérabilité entre REST et SOAP.

En effet, la programmation dans un environnement du système distribué est une activité très délicate pour les développeurs, qui doivent gérer tous les détails liés à la communication (par exemple l'adressage, le traitement des erreurs et la représentation des données). Cela est dû au fait que les développeurs utilisent l'abstraction de bas niveau fournie par le système d'exploitation réseau. Pour libérer les développeurs de l'activité coûteuse et fastidieuse de la résolution de tous les problèmes liés à la gestion du réseau, une plus grande abstraction doit être introduite. C'est exactement le rôle de la couche Middleware c'est la solution la mieux adaptés à l'intégration de sources web. Une comparaison des principales approches d'intégration (produits middleware) est présentée comme suit :

	Extract, Transform and Load (ETL)	**Entreprise Information Integration (EII)**	**Entreprise Application Integration (EAI)**
Flot de données	Unidirectionnel	Bidirectionnel	Bidirectionnel
Mouvements de données	Lots cédules	Au moment de la requête	Déclenché par la transaction
Latence	Journalier à mensuel	Temps réel	Quasi temps réel
Transformation et agrégation des données	Grande capacité	Moyenne capacité	Faible capacité
Volume de données	Grand	moyenne	Petit

Tableau 6.1 – Comparaison entre les approches d'intégration

Nous pouvons par exemple améliorer l'une de ces approches au niveau de sa capacité de supporter une grande quantité de données. On peut aussi proposer une adaptation de ETL pour faire un transfère bidirectionnel. Nous pensons que ce type d'approches est une alternative intéressante pour le problème d'interopérabilité que nous venons d'évoquer. En outre, elle pourrait permettre de transférer les données entre n'importe quelle source de données.

Liste des publications

Revues internationales

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « MIPCNF : Multi-Iterative Partitioning Class Noise Filter », ***Journal of Intelligent & Fuzzy Systems***,vol 37, no 5, pp 6761-6772, 2019.

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « A New Horizo-Vertical Distributed Feature Selection Approach », ***Cybernetics and Information Technologies***, vol 18, no 4, pp 15-28, 2018.

Conférences internationales

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « Detection and elimination of class noise in large datasets using partitioning filter technique », ***Proceedings of the 4th IEEE International Colloquium on Information Science and Technology (CiSt)***, IEEE, pp 194-199, Octobre 2016.

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « Class noise elimination approach for large datasets based on a combination of classifiers », ***Proceedings of the 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)***, IEEE, pp 125-130, Mai 2016.

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « Big data clustering : Algorithms and challenges », ***Proceedings of International conference on Big Data, Cloud and Applications (BDCA'15)***, May 2015.

Manifestations scientifiques nationales

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « Contribution à la classification dans le contexte du Big Data », ***la 3éme édition des journées scientifiques URAC no 29 du LRIT***, 28 Novembre 2015.

Btissam ZERHARI, Ayoub AIT LEHCEN, Salma MOULINE « Big Data Clustering », ***la 4éme édition de la journée doctoriales***, 19, 20 et 21 Février 2015.

Bibliographie

A, J., Jason Abrevaya, H. et M, F. (1998). Misclassification of the dependent variable in a discrete-response setting. *Econometrics*, 87(2):239–269.

Abellán, J. et Masegosa, A. R. (2010a). Bagging decision trees on data sets with classification noise. *In International Symposium on Foundations of Information and Knowledge Systems*, pages 248–265. Springer.

Abellán, J. et Masegosa, A. R. (2010b). Bagging decision trees on data sets with classification noise. *In International Symposium on Foundations of Information and Knowledge Systems*, pages 248–265. Springer.

Adriaans, P. et Zantinge, D. (1996). Data mining addison wesley longman limited. *In Edinbourgh Gate, Harlow, CM20 2JE, England*.

Alasadi, S. A. et Bhaya, W. S. (2017). Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107.

Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sanchez, L. et F, H. (2011). Keel data-mining software tool : data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2–3):255—287.

Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S. et Koutsoukos, X. D. (2010). Local causal and markov blanket induction for causal discovery and feature selection for classification part i : Algorithms and empirical evaluation. *Machine Learning Research*, 11:171—234.

Amini, M. R. et Gallinari, P. (2005). Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, 8(4):385—413.

Ananthanarayana, V., Subramanian, D. et Murty, M. N. (2000). Scalable, distributed and dynamic mining of association rules. *In International Conference on High-Performance Computing*, pages 559—-566. Springer.

Angluin, D. et Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Aslam, J. A. et Decatur, S. E. (1996). On the sample complexity of noise-tolerant learning. *Inf. Process. Lett.*, 57(4):189–195.

Banerjee, M. et Chakravarty, S. (2011). Privacy preserving feature selection for distributed data using virtual dimension. *In the 20th international conference on Information and knowledge management*, pages 2281—-2284. ACM.

Barnett, V. et Lewis, T. (1974). *Outliers in statistical data*. Wiley.

Beretta, L. et Santaniello, A. (2011). Implementing relieff filters to extract meaningful features from genetic lifetime datasets. *Biomedical Informatics*, 44(2):361–369.

Bermejo, P., de la Ossa, L., G'amez, J. A. et Puerta, J. M. (2012). Fast wrapper feature subset selection in high dimensional datasets by means of filter re-ranking. *Knowledge Based Systems*, 25(1):35—-44.

Bi, Y. et Jeske, D. R. (2010). The efficiency of logistic regression compared to normal discriminant analysis under class-conditional classification noise. *Journal of Multivariate Analysis*, 101(7):1622–1637.

Bolon-Canedo, V., Sanchez-Marono, N. et Alonso-Betanzos, A. (2015). A distributed feature selection approach based on a complexity measure. *In International Work Conference on Artificial Neural Networks*.

Bolon-Canedo, V., Sanchez-Marono, N. et Cervino-Rabunal, J. (2014). Toward parallel feature selection from vertically partitioned data. *In ESANN*. Citeseer.

Bontempi, G. et Meyer, P. E. (2010). Causal filter selection in microarray data. *In The 27th international conference on machine learning*, pages 95—102.

Bootkrajang, J. et Kabán, A. (2011). Multi class classification in the presence of labelling errors. *In The 19th european symposium On Artificial Neural Networks*, pages 345—-350.

Bootkrajang, J. et Kaban, A. (2012). Label-noise robust logistic regression and its applications. *In The Joint European conference on machine learning and knowledge discovery in databases*, pages 143—158. springer.

Bootkrajang, J. et Kaban, A. (2013). Classification of mislabelled microarrays using robust sparse logistic regression. *Bioinformatics*, 29(7):870—-877.

Bouveyron, C. et Girard, S. (2009). Robust supervised classification with mixture models : Learning from data with uncertain labels. *Pattern recognition*, 42(11):2649—-2658.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. et Sander, J. (2000). Lof : identifying density-based local outliers. *In ACM sigmod record*, volume 29, pages 93–104. ACM.

Brodley, C. E. et Friedl, M. A. (1999a). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

Brodley, C. E. et Friedl, M. A. (1999b). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

Brodley, C. E. et Friedl, M. A. (1999c). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

Bryll, R., Gutierrez-Osuna, R. et Quek, F. (2003). Attribute bagging : improving accuracy of classifier ensembles by using random feature subsets. *Pattern recognition*, 36(6):1291–1302.

Buhrmester, M., Kwang, T. et Gosling, S. D. (2011). Amazon's mechanical turk : A new source of inexpensive, yet high-quality, data ? *Perspectives on psychological science*, 6(1):3–5.

Chan, P. K. et Stolfo, S. J. (1993). Toward parallel and distributed learning by meta-learning. *In AAAI workshop in Knowledge Discovery in Databases*, pages 227—240.

Chandola, V., Banerjee, A. et Kumar, V. (2009). Anomaly detection : A survey. *ACM computing surveys (CSUR)*, 41(3):15.

Chang, C., Verhaegen, P. A. et Duflou, J. R. (2014). A comparison of classifiers for intelligent machine usage prediction. *In The International conference on intelligent environments*, pages 198–201. IEEE.

Chawla, N. V., Hall, L. O., Bowyer, K. W. et Kegelmeyer, W. P. (2004). Learning ensembles from bites : A scalable and accurate approach. *Journal of Machine Learning Research*, 5(Apr):421–451.

Chittineni, C. B. (1982). Maximum likelihood estimation of label imperfection probabilities and its use in the identification of mislabeled patterns. *IEEE Transactions on Geoscience and Remote Sensing*, 20:99—-111.

Christmann, A., Steinwart, I. et Hubert, M. (2007). Robust learning from bites for data mining. *Computational statistics & data analysis*, 52(1):347–361.

Daniel Paulino, C., Soares, P. et Neuhaus, J. (2003). Binomial regression with misclassification. *Biometrics*, 59(3):670–675.

Das, K., Bhaduri, K. et Kargupta, H. (2010). A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. *Knowledge and information systems*, 24(3):341—-367.

Dash, M. et Liu, H. (2003a). Consistency-based search in feature selection. *Artificial intelligence*, 151(1):155–176.

Dash, M. et Liu, H. (2003b). Consistency-based search in feature selection. *Artificial intelligence*, 151(1).

Dawid, A. P. et Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28.

Delany, S. J., Segata, N. et Mac Namee, B. (2012). Profiling instances in noise reduction. *Knowledge-Based Systems*, 31:28–40.

Denoeux, T. (2000). A neural network classifier based on dempster-shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 30(2):131—-150.

Devijver, P. A. (1986). On the editing rate of the multiedit algorithm. *Pattern Recognition Letters*, 4(1):9–12.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157.

Du, J. et Cai, Z. (2015). Modelling class noise with symmetric and asymmetric distributions. *In The Twenty-Ninth AAAI International Conference on Artificial Intelligence*, pages 2589—-2595.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. et Uthurusamy, R. (1996). Advances in knowledge discovery and data mining.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *machine learning research*, 3(Mar):1289—-1305.

Frénay, B. et Verleysen, M. (2014). Classification in the presence of label noise : a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Frénay, B., de Lannoy, G. et Verleysen, M. (2011). Label noise-tolerant hidden markov models for segmentation : application to ecgs. *In The european conference On Machine learning and Knowledge Discovery in Databases*, pages 455–470.

Gaba, A. et Winkler, R. L. (1992). Implications of errors in survey data : a bayesian model. *Management Science*, 38(7):913–925.

Gamberger, D., Lavrac, N. et Dzeroski, S. (2000a). Noise detection and elimination in data preprocessing : experiments in medical domains. *Applied Artificial Intelligence*, 14(2):205–223.

Gamberger, D., Lavrac, N. et Dzeroski, S. (2000b). Noise detection and elimination in data preprocessing : experiments in medical domains. *Applied Artificial Intelligence*, 14(2):205–223.

Gamberger, D., Lavrac, N. et Groselj, C. (1999a). Experiments with noise filtering in a medical domain. *In ICML*, pages 143–151.

Gamberger, D., Lavrac, N. et Groselj, C. (1999b). Experiments with noise filtering in a medical domain. *In ICML*, pages 143–151.

Gamberger, D., Lavrac, N. et Groselj, C. (1999c). Experiments with noise filtering in a medical domain. *In ICML*, pages 143–151.

Gao, Y., Gao, F. et Guan, X. (2010). Improved boosting algorithm with adaptive filtration. *In Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 3173–3178. IEEE.

García-Zattera, M. J., Mutsvari, T., Jara, A. et Lesaffrea, E. (2010). Correcting for misclassification for a monotone disease process with an application in dental research. *Statistical Modelling*, 29(30):3103–3117.

Gates, G. (1972). The reduced nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 18(3):431–433.

Gerlach, R. et Stamey, J. (2007). Bayesian model selection for logistic regression with misclassified outcomes. *Statistical Modelling*, 7(3):255—-273.

Ghosh, A., Manwani, N. et Sastry, P. S. (2015). Making risk minimization tolerant to label noise. *Machine Learning*, 160:93—107.

Guan, D., Yuan, W. et Shen, L. (2013). Class noise detection by multiple voting. *In Natural Computation (ICNC), 2013 Ninth International Conference on*, pages 906–911. IEEE.

Guyon, I., Bitter, H. M., Ahmed, Z., Brown, M. et Heller, J. (2005). Multivariate nonlinear feature selection with kernel methods. *In Soft computing for information processing and analysis*, pages 313—326. Springer.

Hall, M. A. (1999). Correlation-based feature selection for machine learning. *The University of Waikato*.

Harrell, F. E. (2015). Ordinal logistic regression. *In Regression modeling strategies*, pages 311–325. Springer.

He, H., Graco, W. et Yao, X. (1998). Application of genetic algorithm and k-nearest neighbour method in medical fraud detection. *In Asia-Pacific Conference on Simulated Evolution and Learning*, pages 74–81. Springer.

Hernandez-Lobato, D., Hernandez-Lobato, J. M. et Dupont, P. (2011). Robust multi-class gaussian process classification. *In International conference On Advances in Neural Information Processing Systems 24*, pages 280–288.

Ho, T. K. (1995). Random decision forests. *In Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE.

Hodge, V. et Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126.

Hoffmann, H. (2007). Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177—196.

Hua, J., Tembe, W. D. et Dougherty, E. R. (2009). Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409—-424.

Hughes, N. P., Roberts, S. J. et Tarassenko, L. (2004). Semi-supervised learning of probabilistic models for ecg segmentation. *In IEEE Engineering in Medicine and Biology Conference (EMBC)*.

Jain, A. et Zongker, D. (1997). Feature selection : Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, 19(2):153–158.

Jeatrakul, P., Wong, K. W. et Fung, C. C. (2010). Data cleaning for classification using misclassification analysis. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 14(3):297–302.

Jiang, W. (2001). Some theoretical aspects of boosting in the presence of noisy data. *In Proceedings of the Eighteenth International Conference on Machine Learning*. Citeseer.

Jorge Alberto, A., Zangiacomi Martinez, E. et Louzada-Neto, F. (2004). Binary data in the presence of misclassifications. *In The 16th International symposium on Association for Statistical Computing*, pages 581—-587.

Joseph, L., Gyorkos, T. W. et Coupal, L. (1995). Bayesian estimation of disease prevalence and the parameters of diagnostic tests in the absence of a gold standard. *American journal of epidemiology*, 141(3):263–272.

Kaban, A. et Bootkrajang, J. (2013). Learning a label-noise robust logistic regression : Analysis and experiments. *In The International Conference on Intelligent Data Engineering and Automated Learning*, pages 569—-576. springer.

Kaster, F. O., Menze, B. H., Weber, M.-A. et Hamprecht, F. A. (2010). Comparative validation of graphical models for learning tumor segmentations from noisy manual annotations. *In International MICCAI Workshop on Medical Computer Vision*, pages 74–85. Springer.

Khardon, R. et Wachman, G. (2007). Noise tolerant variants of the perceptron algorithm. *Machine Learning Research*, 8:227–248.

Khoshgoftaar, T. M. et Rebours, P. (2004a). Generating multiple noise elimination filters with the ensemble-partitioning filter. *In Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 369–375. IEEE.

Khoshgoftaar, T. M. et Rebours, P. (2004b). Generating multiple noise elimination filters with the ensemble-partitioning filter. *In Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 369–375. IEEE.

Khoshgoftaar, T. M. et Rebours, P. (2007). Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology*, 22(3):387–396.

Kim, M. W. et Ryu, J. W. (2004). Optimized fuzzy classification for data mining. *In International Conference on Database Systems for Advanced Applications*, pages 582–593. Springer.

Knorr, E. M., Ng, R. T. et Tucakov, V. (2000). Distance-based outliers : algorithms and applications. *The VLDB Journal—The International Journal on Very Large Data Bases*, 8(3-4):237–253.

Kohavi, R. et John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273—-324.

Kononenko, I. (1994). Estimating attributes : analysis and extensions of relief. *In The European conference on machine learning*, pages 171—182. Springer.

Krithara, A., Amini, M. R., Renders, J.-M. et Goutte, C. (2008). Semisupervised document classification with a mislabeling error model. *In The 30th European conference on Advances in Information Retrieval*, pages 370—381.

Lachenbruch, P. A. (1979). Note on initial misclassification effects on the quadratic discriminant function. *Technometrics*, 21(1):129–132.

Ladouceur, M., Rahme, E., Pineau, C. A. et Joseph, L. (2007). Robustness of prevalence estimates derived from misclassified data from administrative databases. *Biometrics*, 63(1):272–279.

Lam, C. P. et Stork, D. G. (2003). Evaluating classifiers by means of test data with noisy labels. *In IJCAI*, pages 513–518.

Lawrence, N. D. et Schólkopf, B. (2001). Estimating a kernel fisher discriminant in the presence of label noise. *In The 18th International conference On Machine Learning*, pages 306—-313.

Lemmond, T. D., Chen, B. Y., Hatch, A. O. et Hanley, W. G. (2010). An extended study of the discriminant random forest. *In Data Mining : A Special Issue in Annals of Information Systems*, pages 123–146. Springer.

Li, L., Darden, T. A., Weingberg, C. R., Levine, A. J. et Pedersen, L. G. (2001). Gene assessment and sample classification for gene expression data using a genetic algorithm k-nearest neighbor method. *Combinatorial chemistry and high throughput screening*, 4(8):727–739.

LI, S. T. et Chen, C. C. (2015). A regularized monotonic fuzzy support vector machine model for data mining with prior knowledge. *IEEE Transactions on Fuzzy Systems*, 23(5):1713–1727.

Li, Y., Wessels, o. F. A., de Ridder, D. et Reinders, M. J. T. (2007). Classification in the presence of class noise using a probabilistic kernel fisher method. *Pattern recognition*, 40(12):3349—3357.

Libralon, G. L., de Leon Ferreira, A. C. P., Lorena, A. C. *et al.* (2009). Pre-processing for noise detection in gene expression classification data. *Journal of the Brazilian Computer Society*, 15(1):3–11.

Lichman, M. (2013). Uci machine learning repository.

Liu, H. et Setiono, R. (1995). Chi2 : Feature selection and discretization of numeric attributes. *In The seventh international conference on Tools with artificial intelligence*, page 388–391.

Liu, H. et Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, 17(4):491–502.

Liu, J., Gustafson, P., Cherry, N. et Burstyn, I. (2009). Bayesian analysis of a matched case-control study with expert prior information on both the misclassification of exposure and the exposure-disease association. *Statistics in medicine*, 28(27):3411—-3423.

Lorena, A. C., Costa, I. G., Spolaor, N. et De Souto, M. C. (2012). Analysis of complexity indices for classification problems : cancer gene expression data. *Neurocomputing*, 75(1):33—-42.

Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A. et Hellerstein, J. M. (2012). Distributed graphlab : a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727.

Luengo, J. et Herrera, F. (2015). An automatic extraction method of the domains of competence for learning classifiers using data complexity measures. *Knowledge and Information Systems*, 42(1):147–180.

Ma, L., Destercke, S. et Wang, Y. (2016). Online active learning of decision trees with evidential data. *Pattern recognition*, 52:33–45.

Macia, N., Bernado-Mansilla, E., Orriols-Puig, A. et Ho, T. K. (2013). Learner excellence biased by data set selection : a case for data characterisation and artificial data sets. *Pattern Recognition*, 46(3):1054—-1066.

Maimon, O. et Rokach, L. (2005). Decomposition methodology for knowledge discovery and data mining. *In Data mining and knowledge discovery handbook*, pages 981–1003. Springer.

Mansour, Y. et Parnas, M. (1998). Learning conjunctions with noise under product distributions. *Information Processing Letters*, 68(4):189—196.

McInturff, P., O Johnson, W., Cowling, D. et A Gardner, I. (2004). Modelling risk when binary outcomes are subject to error. *Statistics in medicine*, 23(7):1095—1109.

Miao, Q., Cao, Y., Xia, G., Gong, M., Liu, J. et Song, J. (2016). Rboost : label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners. *IEEE transactions on neural networks and learning systems*, 27(11):2216–2228.

Michalek, J. E. et Tripathi, R. C. (1980). The effect of errors in diagnosis and measurement on the estimation of the probability of an event. *Journal of the American Statistical Association*, 75(371):713–721.

Miranda, A. L., Garcia, L. P. F., Carvalho, A. C. et Lorena, A. C. (2009). Use of classification algorithms in noise detection and elimination. *In International Conference on Hybrid Artificial Intelligence Systems*, pages 417–424. Springer.

Moran-Fernandez, L., Bolon-Canedo, V. et Alonso-Betanzos, A. (2015). A time efficient approach for distributed feature selection partitioning by features. *In Conference of the Spanish Association for Artificial Intelligence*, page 245–254. Springer.

Moran-Fernandez, L., Bolon-Canedo, V. et Alonso-Betanzos, A. (2017). Centralized vs. distributed feature selection methods based on data complexity measures. *Knowledge-Based Systems*, 117:27–45.

Murty, M. N. et Devi, V. S. (2011). *Pattern recognition : An algorithmic approach*. Springer Science, Business Media.

Natarajan, N., Dhillon, I. S., Ravikumar, P. D. et Tewari, A. (2013). Learning with noisy labels. *In Advances in Neural Information Processing Systems*, pages 1196—-1204.

Omitaomu, O. A. (2006). Chapter 4, lecture notes in data mining.

Opitz, D. et Maclin, R. (1999). Popular ensemble methods : An empirical study. *Journal of artificial intelligence research*, 11:169–198.

Oreski, D., Stjepan, O. et Bozidar, K. (2017). Effects of dataset characteristics on the performance of feature selection techniques. *Applied Soft Computing*, 52:109–119.

Parhami, B. (1994). Voting algorithms. *IEEE transactions on reliability*, 43(4): 617–629.

Pechenizkiy, M., Tsymbal, A., Puuronen, S. et Pechenizkiy, O. (2006). Class noise and supervised learning in medical domains : The effect of feature extraction. *In null*, pages 708–713. IEEE.

Peng, H., Long, F. et Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226—-1238.

Peralta, D., del Rio, S., Ramirez-Gallego, S., Triguero, I. et Benitez, J M Herrera, F. (2015). Evolutionary feature selection for big data classification : a mapreduce approach. *Mathematical Problems in Engineering*, 501.

Perez, P. S., Nozawa, S. R., Macedo, A. A. et Baranauskas, J. A. (2016). Windowing improvements towards more comprehensible models. *Knowledge Based Systems*, 92:9–22.

Peteiro-Barral, D. et Guijarro-Berdiñas, B. (2013). A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11.

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

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

Quinlan, J. R. (1986c). Induction of decision trees. *Machine Learning*, 1(1):81–––106.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L. et Moy, L. (2010). Learning from crowds. *Machine Learning Research*, 11(Apr):1297–1322.

Rebbapragada, U. et Brodley, C. E. (2007). Class noise mitigation through instance weighting. *In The 18th European Conference on Machine Learning*, pages 708—715.

Rekaya, R., Weigel, K. A. et Gianola, D. (2001). Threshold model for misclassified binary responses with applications to animal breeding. *Biometrics*, 57(4):1123—-1129.

Robbins, K., Joseph, S., Zhang, W., Rekaya, R. et Bertrand, J. (2006). Classification of incipient alzheimer patients using gene expression data : Dealing with potential misdiagnosis. *Bioinformatics*, 7(1):22—-31.

Rokach, L., Maimon, O. et Arbel, R. (2006). Selective voting—getting more for less in sensor fusion. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(03):329–350.

Ruiz, M., Girón, F. J., Pérez, C. J., Martín, J. et Rojano, C. (2008). A bayesian model for multinomial sampling with misclassified data. *Journal of Applied Statistics*, 35(4):369—-382.

Saeys, Y., Inza, I. et Larranaga, P. (2007). A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517.

Sáez, J. A., Galar, M., Luengo, J. et Herrera, F. (2016a). Inffc : an iterative class noise filter based on the fusion of classifiers with noise sensitivity control. *Information Fusion*, 27:19–32.

Sáez, J. A., Krawczyk, B. et Woźniak, M. (2016b). On the influence of class noise in medical data classification : Treatment using noise filtering methods. *Applied Artificial Intelligence*, 30(6):590–609.

Sánchez, J. S., Barandela, R., Marqués, A. I., Alejo, R. et Badenas, J. (2003). Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022.

Sánchez, J. S., Pla, F. et Ferri, F. J. (1997). Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, 18(6):507–513.

Schafer, J. L. et Graham, J. W. (2002). Missing data : our view of the state of the art. *Psychological methods*, 7(2):147.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J. et Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Scott, C. (2015). A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. *In The International Conference on Artificial Intelligence and Statistics*, pages 838–846.

Sculley, D. et Cormack, G. V. (2008). Filtering email spam in the presence of noisy user feedback. *In CEAS*. Citeseer.

Segata, N., Blanzieri, E. et Cunningham, P. (2009). A scalable noise reduction technique for large case-based systems. *In International Conference on Case-Based Reasoning*, pages 328–342. Springer.

Segata, N., Blanzieri, E., Delany, S. J. et Cunningham, P. (2010). Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, 35(2):301–331.

Skillicorn, D. B. et McConnell, S. M. (2008). Distributed prediction from vertically partitioned data. *Journal of Parallel and Distributed computing*, 68(1): 16–36.

Sluban, B., Gamberger, D. et Lavra, N. (2010). Advances in class noise detection. *In Proceedings of the 2010 conference on ECAI 2010 : 19th European Conference on Artificial Intelligence*, pages 1105–1106. IOS Press.

Sluban, B., Gamberger, D. et Lavrač, N. (2014). Ensemble-based noise detection : noise ranking and visual performance evaluation. *Data mining and knowledge discovery*, 28(2):265–303.

Steven, W. et Hirsh, H. (1992). Classifier learning from noisy data as probabilistic evidence combination. *In The 10th National Conference on Artificial Intelligence*, pages 141—-146.

Sun, B., Chen, S., Wang, J. et Chen, H. (2016). A robust multi-class adaboost algorithm for mislabeled noisy data. *Knowledge-Based Systems*, 102:87–102.

Sun, J.-w., Zhao, F.-y., Wang, C.-j. et Chen, S.-f. (2007). Identifying and correcting mislabeled training instances. *In Future generation communication and networking (FGCN 2007)*, volume 1, pages 244–250. IEEE.

Sun, Y., Babbs, C. et Delp, E. (2006). A comparison of feature selection methods for the detection of breast cancers in mammograms : adaptive sequential floating search vs genetic algorithm. *In The 27th international conference in Engineering in medicine and biology society*, pages 6532—6535. IEEE.

Swartz, T. B., Haitovsky, Y., Vexler, A. et Yang, T. Y. (2004a). Bayesian identifiability and misclassification in multinomial data. *Canadian Journal of Statistics*, 32(3):285–302.

Swartz, T. B., Haitovsky, Y., Vexler, A. et Yang, T. Y. (2004b). Bayesian identifiability and misclassification in multinomial data. *Canadian Journal of Statistics*, 32(3):285–302.

Thongkam, J., Xu, G., Zhang, Y. et Huang, F. (2008). Support vector machine for outlier detection in breast cancer survivability prediction. *In Asia-Pacific Web Conference*, pages 99–109. Springer.

Tomek, I. (1976). An experiment with the edited nearest-neighbor rule. *IEEE Transactions on systems, Man, and Cybernetics*, (6):448–452.

Tsoumakas, G. et Vlahavas, I. (2002). Distributed data mining of large classifier ensembles. *In Companion Volume of the Second Hellenic Conference on Artificial Intelligence*. Springer.

University, V. (2016). Gene expression model selector.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Verbaeten, S. et Van Assche, A. (2003). Ensemble methods for noise elimination in classification problems. *In International Workshop on Multiple Classifier Systems*, pages 317–325. Springer.

Victo Sudha, G. et Raj, V. C. (2011). Review on feature selection techniques and the impact of svm for cancer classification using gene expression profile. *International Journal of Computer Science and Engineering Survey*, 2(3):16—-27.

Wei, H.-L. et Stephen, A. B. (2007). Feature subset selection and ranking for data dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 29(1).

Wheway, V. (2000). Using boosting to detect noisy data. *In Pacific Rim International Conference on Artificial Intelligence*, pages 123–130. Springer.

Wilson, D. R. et Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286.

Wirth, R. et Jochen, H. (2000). Crisp-dm : Towards a standard process model for data mining. *In the 4th international conference on the practical applications of knowledge discovery and data mining*. Citeseer.

Witten, I. H., Frank, E., Hall, M. A. et Pal, C. J. (2016). Data mining : Practical machine learning tools and techniques. *Morgan Kaufmann*.

Xing, E., Jordan, M. et Karp, R. (2001). Feature selection for high-dimensional genomic microarray data. *In The Eighteenth International Conference on Machine Learning*, pages 601–608.

Yang, C. Y., Wang, J. J., Chou, J. J. et Lian, F. L. (2015). Confirming robustness of fuzzy support vector machine via ε - α bound. *Neurocomputing*, 162:256—266.

Yang, S. H. et Hu, B. G. (2008). Efficient feature selection in the presence of outliers and noises. *In The Asia Information Retrieval Symposium*, pages 184—-191. Springer.

Younes, Z., abdallah, F. et Denoeux, T. (2010). Evidential multi-label classification approach to learning from data with imprecise labels. *In The 13th International Conference on Information Processing and Management of Uncertainty*, pages 119—128.

Yu, L. et Liu, H. (2003). Feature selection for high-dimensional data : A fast correlationbased filter solution. *In The 20th international conference on machine learning*, volume 20, page 856.

Yu, L. et Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *Machine Learning Research*, 5:1205—-1224.

Zhang, M. L., Pena, J. M. et Robles, V. (2009). Feature selection for multi-label naive bayes classification. *Information Sciences*, 179(19):3218–3229.

Zhang, P., Zhu, X., Shi, Y., Guo, L. et Wu, X. (2011). Robust ensemble learning for mining noisy data streams. *Decision Support Systems*, 50(2):469–479.

Zhang, W.-L., Guo, B., Shen, Y., Li, D.-G. et Li, J.-K. (2018). An energy-efficient algorithm for multi-site application partitioning in mcc. *Sustainable Computing : Informatics and Systems*, 18:45–53.

Zhang, Y., Ding, C. et Li, T. (2008). Gene selection algorithm by combining relieff and mrmr. *BMC genomics*, 9(2):S27.

Zhao, Z. et Liu, H. (2009). Searching for interacting features in subset selection. *Intelligent Data Analysis*, 13(2):207–228.

Zhu, X. et Wu, X. (2004). Class noise vs. attribute noise : A quantitative study. *Artificial intelligence review*, 22(3):177–210.

Zhu, X., Wu, X. et Chen, Q. (2003). Eliminating class noise in large datasets. *In Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 920–927.

Zhu, X., Wu, X. et Chen, Q. (2006). Bridging local and global data cleansing : Identifying class noise in large, distributed data datasets. *Data mining and Knowledge discovery*, 12(2-3):275–308.

Zhu, X. et Xindong, W. (2004). Class noise vs attribute noise : A quantitative study. *Artificial intelligence review*, 22(3):177–210.