Apprentissage supervisé de données symboliques et l'adaptation aux données massives et distribuées

Raja Haddad

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Université Paris-Dauphine

Apprentissage Supervisé de Données Symboliques et l'Adaptation aux Données Massives et Distribuées.

École Doctorale de Dauphine — ED 543

Spécialité **Informatique**

COMPOSITION DU JURY :

M. Darrell LONG
University of California, Santa Cruz
Président du jury

M. Edwin DIDAY
Université Paris Dauphine
Membre du jury

M. Philippe Rigaux
CNAM
Rapporteur

Mme Rosanna VERDE
Second University of Naples
Rapporteure

Mme. Daniela GRIGORI
Université Paris Dauphine
Membre du jury

M. Filipe AFONSO
Syrokko
Membre du jury

Soutenue le 23.11.2016
par Raja HADDAD

Dirigée par **Witold LITWIN**

DAUPHINE UNIVERSITÉ PARIS | PSL RESEARCH UNIVERSITY PARIS

DÉDICACES

Je dédie ce mémoire :

À mes parents Abdallah et Fattouma
en témoignage de mon amour et de ma reconnaissance, pour l'amour qu'ils m'ont toujours donné depuis mon enfance, pour leurs sacrifices et leurs soutiens. Que Dieu les protège et leurs préserve une bonne santé et une longue vie.

À mes enfant Emna et Mohamed
la source de mon bonheur et grâce à qui j'ai persisté et je n'ai jamais baissé les bras.

À mon mari Majed
qui a su me supporter et me remonter le moral pendant ces années pas toujours faciles.

À mes sœurs Dorra et Amira et à mon frère Mohamed Ali et sa femme Rim
qui m'ont donnée le courage et la volonté d'aller au bout.

À mes beaux parents Mohamed et Ilhème
qui n'ont jamais douté de ma réussite et qui m'ont toujours soutenue.

À tous les membres de ma famille sans aucune exception. Et à tous ceux que ma réussite leur tient à cœur.

Remerciements

C'est avec un grand plaisir que je réserve ces lignes à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail. Un travail qui a été effectué grâce à un contrat CIFRE entre l'université Paris-Dauphine et la société Syrokko suivi de deux années que j'ai passées au sein du LAMSADE.

Je tiens tout d'abord à remercier mon directeur de thèse le professeur ***Witold LITWIN*** pour ses conseils avisés, ses remarques pertinentes et l'exemplarité de sa rigueur scientifique.
Je suis également extrêmement reconnaissante envers le professeur ***Edwin DIDAY*** qui a co-dirigé cette thèse. Edwin m'a transmis ses connaissances et son engagement pour faire évoluer le domaine des Analyse de Données Symboliques. Il (à travers la société Syrokko) a su créer un contexte idéal pour ce travail. En plus de son soutien, de ses conseils et du savoir qu'il m'a transmis, je tiens vivement à le remercier pour sa disponibilité et son encouragement.
Je remercie également ***Filipe Afonso*** qui a co-encadré cette thèse, pour ses conseils très précieux et la qualité de nos échanges scientifiques. C'était un vrai honneur et un vrai plaisir de collaborer avec lui durant les trois années que j'ai passées à Syrokko. Je le remercie également pour tout l'intérêt qu'il a porté à mon travail même après avoir quitter Syrokko.

J'adresse mes remerciements aux professeurs ***Rosanna VERDE*** et ***Philippe RIGAUX*** qui m'ont fait l'honneur d'évaluer mon travail de thèse et d'en être les rapporteurs. Je les remercie pour leurs remarques constructives.
Je remercie également les professeurs ***Daniela GRIGORI*** et ***Darrell LONG*** d'avoir accepté de faire partie des membres du jury.

J'adresse mes remerciements à mes collaborateurs dans la société Syrokko : ***Estelle-Sarah ELIEZER*** et ***Carole TOQUE***. C'était un vrai honneur de travailler avec vous.

Je tiens aussi à remercier tout les membres du LAMSADE. Spécialement, ***Alexis Tsoukias*** le directeur du LAMSADE pour sa disponibilité et sa bienveillance. ***Olivier Rouyer*** l'ingénieur du LAMSADE (le sauveur de tous les membres du LAMSADE contre les différents problèmes matériels et logiciels). Je ne le remercierai jamais assez pour ses qualités humaines et professionnelles.
Je tiens aussi à remercier ***Cristina BAZGAN*** pour ses encouragements et son suivi permanent de mes travaux durant et après la période où elle était responsable du programme doctorale en informatique.
Je remercie également le professeur ***Dario Colazzo*** pour ces conseils.
Je remercie également l'équipe administrative du LAMSADE spécialement Mme ***Juliette DE ROQUEFEUIL*** pour sa patience et son aide très précieuse dans le traitement des différents dossiers administratives.

Un grand remerciement à mes amies ***Asma CHARFI***, ***Nesrine BESSGAYER*** et ***Sonia***

TOUBALI pour leurs soutiens et pour leurs aides précieuses dans la relecture et la correction de ce mémoire.

Je remercie également les personnes que j'ai pu côtoyer au plus près pendant ma thèse mes collègues, docteurs et doctorants du LAMSADE pour l'ambiance très agréable qu'ils ont su créer au sein du laboratoire. En particulier : ***Raouia TAKTAK***, ***Amel BENHAMICHE***, ***Lydia TLILIANE***, ***Mohammed OULD***, ***Thomas Pontoizeau*** (notre président), ***Yassine NAGHMOUCHI***, ***Meriem MAHJOUB***, ***Lynda ATIF***, ***Mohamed Khalil LABIDI***, ***Youcef MAGNOUCHE***, ***Fabien LABERNIA***, ***Mayassa BOUORM***, ***Anaelle WILCZYNSKI***, ***Ian-christopher TERNIER***, ***Saeed HADIKHANLOO***, ***Marek CORNU***, ***Mohamed Lamine MOUHOUB***, ***Amine Louati***, ***Tom DONAT***, ***Diana JLAILATY***, ***Hiba ALILI***, etc.

Un remerciement particulier à tous mes ami(e)s, qui se reconnaissent, qui m'ont soutenu dans les moments difficiles en particulier : ***Souhila ARIB***, ***Iman BEN MANSOUR***, ***Manel BEN SALAH***, ***Wafa*** et ***Salem CHOUIKH*** et ***Kaouther GUIDARA***.

Finalement je tiens à remercier celle qui m'a permise de travailler sereinement parce qu'elle s'est occupée de mes enfants ***tatta Touria***.

Table des matières

Introduction

1 Contexte

Le volume des données circulant sur le Web, ou stockées par les entreprises est en croissance continue. Afin de pouvoir exploiter cette richesse, il est nécessaire d'extraire des connaissances à partir de très grands volumes d'informations. Le domaine ayant pour but de résoudre cette problématique est la science des données (***Data Science***) [33]. La science des données a pour but d'extraire des connaissances à partir de tous types de données (structurées ou non, de sources homogènes ou hétérogènes, etc.). Elle représente l'intersection de plusieurs disciplines comme la statistique, les mathématiques, l'intelligence artificielle et la fouille de données (***Data Mining***). Le *data mining* offre des méthodes d'analyses très utiles pour l'extraction de connaissances. Ces méthodes se divisent en deux catégories dites supervisées et non supervisées. Ces dernières ont pour but de regrouper les unités statistiques suivant leurs caractéristiques. Les méthodes supervisées proposent des modèles pour prédire une action ou une décision concernant de nouveaux individus en se basant sur leurs descriptions.

Afin de rendre accessible l'étude des données sur plusieurs niveaux d'agrégation, le domaine de l'Analyse de Données Symboliques (ADS) est apparu [40]. Depuis, ce domaine s'est développé en proposant plusieurs méthodes d'analyse spécifiques à l'ADS ([12, 37, 42, 43, 45], [44]). Ces méthodes ont été implémentées dans des outils comme Sodas [46], Syr [5] et des libraires R [85, 6, 62] permettant leurs tests et leurs applications sur de nouvelles bases de données.
L'ADS traite un nouveau type d'unités statistiques [44] : les classes d'individus et de variables appelées "symboliques". Une donnée symbolique (un intervalle, un histogramme, etc.) permet de prendre en compte la variabilité interne aux classes et ne peut donc pas être réduite à un seul nombre. Une variable est dite "symbolique" si elle associe à chaque classe d'individus une donnée symbolique.
Le principe de base de l'ADS est l'étude des données suivant différents niveaux d'agrégation. Ceci en passant de l'étude des individus (ou unité statistiques de premier

ordre) à l'étude de "classes d'individus" (considérées comme des unités statistiques de second ordre). L'utilisation de classes présente deux avantages : le premier est que les classes réduisent la taille des données ce qui est évidemment important dans un contexte de grande bases de données et le second est que souvent elles représentent une unité statistique intéressante en soi pour l'utilisateur : par exemple, pour étudier des régions plutôt que leurs habitants, des diagnostiques plutôt que des patients, ou des classes de produits achetés par chaque consommateur d'une grande surface plutôt que les produits. Ces classes d'individus sont décrites par des variables symboliques ayant pour objectif de conserver la variation interne des individus qui les composent.
Une ADS se fait en deux étapes [42] : dans la première étape, on passe des individus décrits par des données classiques vers des classes d'individus décrites par des variables symboliques alors que la deuxième étape concerne l'extension des méthodes d'apprentissage classiques aux données symboliques. Lors de la création des données symboliques à partir des données classiques, souvent les variables quantitatives sont automatiquement transformées en intervalles. Cette transformation engendre une perte d'informations sur la distribution des valeurs des variables quantitatives. Pour résoudre cette problématique, nous proposons dans cette thèse une solution pour transformer automatiquement les variables qualitatives en histogrammes.

La majorité des méthodes d'arbres de décision symboliques existantes [87, 1, 77, 97] n'acceptent pas différents types de variables explicatives et à expliquer. Afin de remédier à cette problématique, nous proposons une nouvelle méthode d'arbres de décision symbolique.

Comme la plupart des méthodes d'analyse de données, les méthodes symboliques sont incapables de traiter les grandes bases de données (***Big Data***). Généralement, pour extraire des connaissances à partir des *Big Data* un échantillonnage est appliqué afin de réduire la taille des données initiales. L'utilisation d'un échantillon pour étudier les données peut engendrer des résultats non fiables. Afin de remédier à cette problématique, plusieurs travaux ont étendu les méthodes d'analyse *Big Data* [32, 112, 78]. Cette thèse apporte une solution pour étendre l'ADS aux *Big Data*.

2 Problématiques

2.1 Extraction des Histogrammes à partir d'une Variable Continue

La première étape d'une ADS a pour objectif de construire le tableau de données symboliques à partir des données initiales classiques. Le choix de la méthode d'agrégation des données est crucial pour la qualité des résultats d'une ADS.
Généralement les variables qualitatives sont converties en diagrammes de fréquences et les variables quantitatives sont transformées en intervalles ou en histogrammes. La

transformation des variables continues en histogrammes représente une tâche délicate. Dans la littérature, il n'existait aucun outil en ADS qui fasse automatiquement cette conversion.
Cette transformation peut être faite en deux étapes en utilisant les méthodes de discrétisation classiques, selon le schéma suivant :
$var_{cont} \xrightarrow{discretisation} var_{nomi} \xrightarrow{transfomation} var_{histo}$.
L'inconvénient de cette méthode est la qualité des histogrammes obtenus : on ne peut pas garantir que ces histogrammes sont les plus discriminants pour les classes d'individus. D'où la première problématique qui est la conversion automatique des variables continues en histogrammes les plus discriminants pour les classes d'individus.

2.2 Extension de la méthode d'Arbre de Décision aux Données Symboliques

La méthode des arbres de décision est l'une des méthodes de *data mining* les plus connues et les plus utilisées. Elle doit son succès à la facilité de son interprétation et de sa compréhension par les non spécialistes en analyse de données. Dans la littérature plusieurs travaux ont traités l'extension de cette méthode aux données symboliques [87, 1, 77, 97]. A part Stree [97], ces méthodes n'acceptent pas plusieurs types de variables à expliquer et explicatives. Cependant, Stree génère des arbres de décision très courts avec des taux d'erreurs importants. D'où la deuxième problématique à résoudre qui est la création d'une nouvelle méthode d'arbres de décision symboliques permettant de traiter à la fois des variables classiques et plusieurs types de variables symboliques.

2.3 Extension de l'ADS aux Big Data

Comme on a déjà signalé, de nos jours la taille des données recueillies est de plus en plus grande. Nous passons de l'étude de quelques centaines de Ko à l'étude de données de plusieurs To. Cette évolution a poussé les *"Data Scientists"* à étendre les méthodes d'analyse de données pour pouvoir traiter ces grands volumes de données. Étant donné que les données symboliques permettent de réduire considérablement la taille des données à étudier tout en gardant le maximum d'informations [14, 12], il semble intéressant d'utiliser l'ADS pour l'étude de Big Data. Cependant, les outils d'extraction de données symboliques existants (DB2SO de SODAS [46] et TabSyr de Syr [5]) sont incapables de traiter d'aussi grands volumes de données. D'où la troisième problématique qui est l'extension de l'extraction de données symboliques à partir de Big Data.

3 Contributions

Afin de résoudre les problématiques que nous avons présentées précédemment nous avons mis en place trois méthodes.

- La première appelée ***"HistSyr"*** [45] transforme automatiquement les variables continues en histogrammes les plus discriminants pour les différentes classes d'individus. Nous avons implémenté cette méthode en utilisant le langage de programmation Java. Nous avons ainsi enrichi le logiciel Syr [5] par l'ajout d'un nouveau module permettant la transformation automatique des variables continues en histogrammes. Ensuite, nous avons comparé ses résultats à ceux d'autres méthodes de discrétisation classiques (la discrétisation en classes d'égale amplitude, la discrétisation selon les quantiles et *"Multi Interval Algorithm"* [50]. Nous avons ainsi démontré que les histogrammes résultats d'HistSyr sont plus discriminants pour les classes d'individus que ceux des autres méthodes [45].

- Notre deuxième méthode est nommée ***"SyrTree"***. Elle représente une nouvelle méthode d'arbre de décision symbolique. Elle accepte tous types de variables explicatives et traite différents type de variables à expliquer (les classes d'individus, les variables nominales et les histogrammes). Nous avons implémenté cette méthode en utilisant le langage de programmation Java. Nous avons ainsi enrichi le logiciel Syr [5]par l'ajout d'un nouveau module permettant la construction et le test d'arbres de décision symboliques. Nous avons comparé les performances prédictives de SyrTree à celles d'autres méthodes d'arbre de décision classiques (CART[20] et C4.5 [89]) et symboliques (Stree[97]). L'étude a montré que les taux d'erreurs des arbres SyrTree sur les échantillons de test sont meilleurs que ceux de Stree [97](voir chapitre 3). Les taux d'erreur des arbres SyrTree se rapprochent de ceux des méthodes classiques (CART [20] et C4.5 [89]). Nous trouvons quelques cas où les résultats de SyrTree sont meilleurs (par exemple pour la base de donnée "Breast tissu" nous avons un taux d'erreur de 25% avec SyrTree contre 34% pour CART et 36% pour C4.5) avec des arbres beaucoup plus courts et plus faciles à interpréter. Le chapitre 3 de cette thèse présente tous les résultats obtenus sur plusieurs bases de données UCI [13].

- Notre troisième méthode nommée ***"CloudHistSyr"*** représente l'extension d'HistSyr aux données scalables et distribuées. Cette méthode est composée de la succession de deux programmes Map/Reduce et d'un programme local. Tous les programmes ont été implémentés en utilisant Java. Les tests des programmes Map/Reduce ont été effectués en utilisant Amazone Web Services [1][80]. Durant ces tests nous avons utilisés différentes configurations des

1. https ://aws.amazon.com/fr/

clusters (en augmentant le nombre de machines virtuelles et en variant leurs performances). Nous avons ainsi démontré la scalabilité de nos programmes Map/Reduce. En effet, en augmentant le nombre de nœuds du cluster de test le temps d'exécution diminuait. Grâce à "CloudHistSyr" nous avons pu construire les histogrammes les plus discriminants à partir des données volumineuses (68 Go) issues d'une étude effectuée à Syrokko. Cette étude concernait les données recueillis d'un portique de test installé à Nantes. Ces données ont été recueillies dans le temps par 21 capteurs installés sur le portique d'autoroute. Vu le volume des données initiales(1.75 Go pour chaque capteur), il a été impossible d'utiliser HistSyr pour convertir les valeurs des capteurs en histogrammes. Pour cela nous avons utilisé "CloudHistsyr" qui a été capable au bout d'une vingtaine de minutes de nous retourner les histogrammes les plus discriminants pour l'un des capteurs du portique (à partir de 139 millions de valeurs). A notre connaissance, c'est la $1^{ère}$ méthode mise en place pour convertir une variable continue en histogrammes à partir de Big Data.

Toutes ces méthodes ont été utilisées dans le cadre d'études de données de clients de la société Syrokko [2].

4 Plan de la thèse

Le travail effectué dans cette thèse s'articule autour de cinq chapitres :

- Chapitre 1 : il présente présente l'Analyse de Données Symboliques (ADS). Nous exposons d'abord les notions de base : la différence entre individus et classes d'individus au niveau de la description, la définition d'un objet symbolique et d'une variable symbolique, etc. Ensuite, nous rappelons les deux étapes de toute ADS en présentant les principales méthodes existantes d'extraction et de traitement de données symboliques. Enfin, nous présentons et comparons les outils d'ADS les plus connus : les logiciels SODAS, SYR et les librairies symboliques de R.

- Chapitre 2 : il présente notre méthode "HistSyr". Nous commençons par la présentation de l'état de l'art des méthodes de discrétisation. Ensuite, nous décrivons l'algorithme d'HistSyr. Enfin, nous comparons HistSyr à d'autres méthodes de discrétisation sur des données UCI [13] et sur des données de clients de Syrokko.

2. www.syrokko.com

- Chapitre 3 : il présente notre méthode d'arbres de décision symboliques, "SyrTree". Nous présentons d'abord, l'état de l'art des méthodes d'arbres de décision classiques et symboliques. Puis, nous présentons "SyrTree". Nous spécifions pour chaque type de variables à expliquer ses critères de découpages. Ensuite, nous comparons SyrTree à d'autres méthodes d'arbre de décision. Enfin, nous présentons l'utilisation de SyrTree dans des études faites au sein de Syrokko.

- Chapitre 4 : il présente l'extension de la méthode HistSyr aux Big Data. D'abord, nous présentons l'état de l'art des méthodes de data mining scalable et distribué. Puis, nous présentons les composants de "CloudHistSyr". Ensuite, nous exposons les différentes combinaisons des composants de CloudHistSyr pour étendre HistSyr aux Big Data. Finalement, nous présentons les résultats du test de CloudHistSyr sur des données réelles issues de l'étude du portique de Nantes faite à Syrokko.

- Chapitre 5 : il conclut cette thèse. Nous rappelons d'abord les différentes contributions de cette thèse. Ensuite, nous discutons leurs limitations et nous finissons par proposer quelques perspectives.

CHAPITRE 1

Analyse des données symboliques

1 Introduction

L'analyse de Données Symbolique (ADS) est un domaine complémentaire à la fouille de données classique. L'ADS permet l'étude des unités statistiques à différents niveaux de généralité en passant des individus aux classes d'individus. Ces dernières sont décrites par des données symboliques conservant la variation interne des individus qui les composent. Ces données sont décrites par des variables symboliques à valeurs classiques (numérique ou nominale) ou symboliques (intervalles, histogrammes, loi de probabilité, fonctions, ensemble de valeurs, etc.).

Une ADS se fait en deux étapes[42]. La première concerne la création des données symboliques à partir des données classiques. La deuxième étape consiste à analyser les données symboliques en utilisant des méthodes d'analyse symboliques. Ces méthodes sont le résultat de l'extension des méthodes classiques aux données symboliques.

Les domaines d'applications de l'ADS sont très variés. En effet, en passant des individus aux classes d'individus on définit de nouvelles unités statistiques qui intéressent les utilisateurs. Par exemple en marketing (clients $\rightarrow$ comportements), transports (véhicules $\rightarrow$ trajectoires), génie civil (défauts $\rightarrow$ ouvrages), télécommunications (clients $\rightarrow$ abonnements), biologie (gènes $\rightarrow$ génomes), crédit (clients $\rightarrow$ zones géographiques), santé (patients $\rightarrow$ pathologies ou régions de patients).

Ce chapitre est organisé comme suit : nous introduisons d'abord, les notions de base de l'ADS. Ensuite, nous explicitons les étapes d'une ADS. Enfin, nous décrivons et comparons les outils d'ADS les plus connus.

2 Notions de base de l'ADS

2.1 Des individus aux classes d'individus

En statistique, un individu est un élément d'une population (l'ensemble de tous les individus étudiés), dont on mesure (ou observe) la valeur qu'il a pour la variable étudiée. C'est une unité statistique qui peut être un sujet (être libre, motivé, etc.) ou un objet (un patient, une fleur, etc.). Un individu est décrit par des variables classiques qui peuvent être qualitatives ou quantitatives. Les individus représentent des unités statistiques de premier ordre.

Une classe d'individus est un sous-ensemble d'individus. Elle est décrite par des variables symboliques qui prennent en compte la variabilité des individus qui la constituent. En ADS, les classes d'individus sont considérées comme des unités statistiques d'un niveau de généralité supérieur à celui des individus.

Comment obtient-on les classes d'individus ?

Les classes d'individus sont obtenues soit à partir des variables qualitatives des données initiales, soit en utilisant le résultat d'une classification automatique.

A partir des variables initiales : Dans ce cas l'ensemble des classes est représenté par un ensemble de catégories. Ces catégories peuvent représenter soit les différentes valeurs d'une variable qualitative issue des données initiales, soit le croisement de catégories (valeurs de plusieurs variables qualitatives). La figure 1.1 représente des exemples de chaque cas. La partie a) de cette figure illustre un exemple où la marque des voitures d'une base décrivant un parc automobile est la classe utilisée pour étudier les différentes marques de voitures. La partie b) de la figure 1.1 illustre le cas où la classe d'individus est le résultat du croisement des deux variables : la marque et le type du moteur.

En utilisant le résultat d'une classification automatique : Pour obtenir les classes d'individus, nous devons appliquer un algorithme de classification automatique sur les données initiales. Les classes résultats de cette classification représentent les classes que nous allons étudier. La figure1.2 illustre un exemple où les classes représentent le résultat d'une classification automatique faite sur une base décrivant des Iris.

2.2 Données, variables, objets et tableaux symboliques

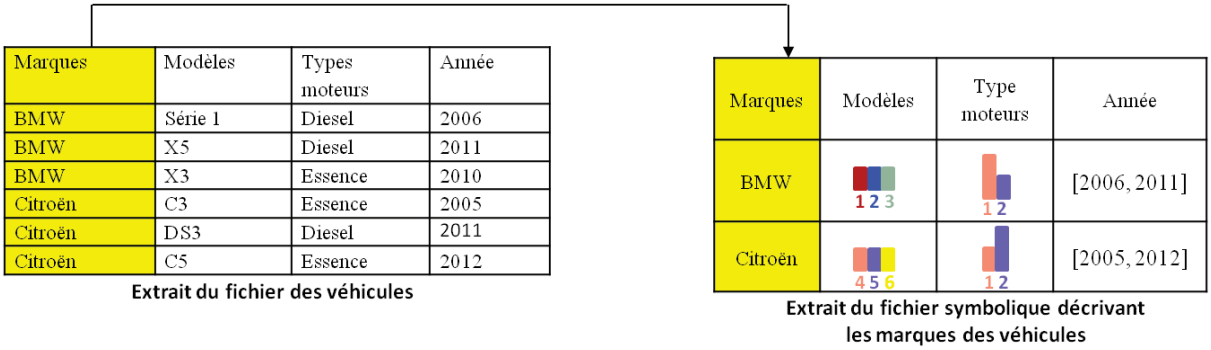
Données Symboliques

Pour prendre en compte la variabilité entre ses membres, chaque classe d'individus est décrite par des suites de catégories ou de nombres. De cette façon un nouveau type de données appelées symboliques est apparu [42]. Le terme "données symboliques" est attribué aux données pour lesquelles les opérateurs numériques standards ($+$, $-$, $\times$ et $\div$) ne peuvent être appliqués directement. Les classes d'individus sont décrites par des variables dites symboliques.

Variables symboliques

On appelle variable symbolique toute variable associant à chaque classe d'individus une valeur symbolique. Cette valeur peut être :

- qualitative unique (la variable "Boîte à vitesse" du tableau 1.1) ;
- quantitative unique (la variable "Année" du tableau 1.1) ;
- un ensemble de valeurs (la variable "Modèles" du tableau 1.1) ;



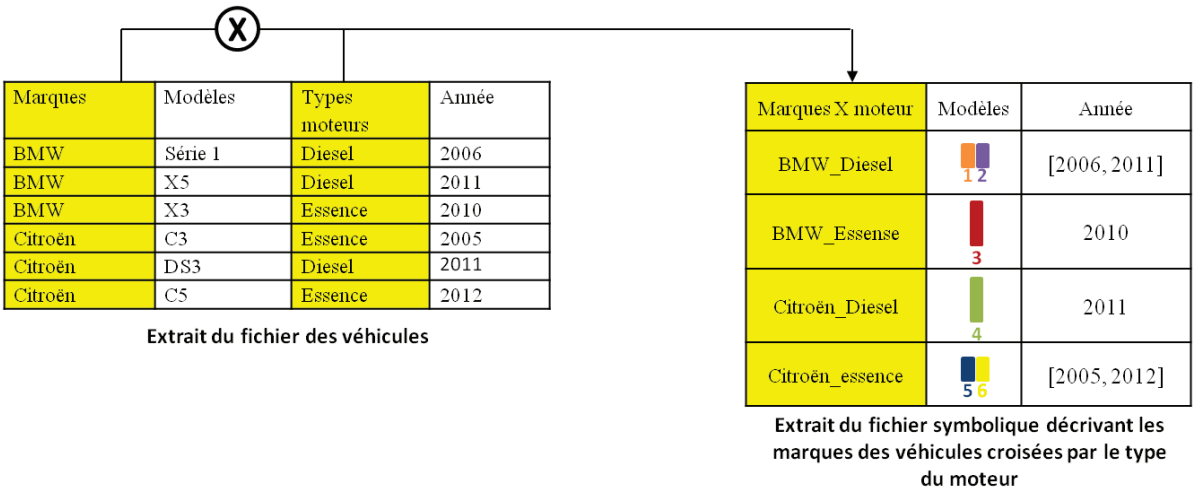
Marques	Modèles	Types moteurs	Année
BMW	Série 1	Diesel	2006
BMW	X5	Diesel	2011
BMW	X3	Essence	2010
Citroën	C3	Essence	2005
Citroën	DS3	Diesel	2011
Citroën	C5	Essence	2012

Extrait du fichier des véhicules

Marques	Modèles	Type moteurs	Année
BMW	1 2 3	1 2	[2006, 2011]
Citroën	4 5 6	1 2	[2005, 2012]

Extrait du fichier symbolique décrivant les marques des véhicules

(a) Classes d'individus issues d'une variable qualitative.



Marques	Modèles	Types moteurs	Année
BMW	Série 1	Diesel	2006
BMW	X5	Diesel	2011
BMW	X3	Essence	2010
Citroën	C3	Essence	2005
Citroën	DS3	Diesel	2011
Citroën	C5	Essence	2012

Extrait du fichier des véhicules

Marques X moteur	Modèles	Année
BMW_Diesel	1 2	[2006, 2011]
BMW_Essense	3	2010
Citroën_Diesel	4	2011
Citroën_essence	5 6	[2005, 2012]

Extrait du fichier symbolique décrivant les marques des véhicules croisées par le type du moteur

(b) Classes d'individus résultats du croisement de deux variables qualitatives.

FIGURE 1.1 – Exemples de classes d'individus issues d'une ou de plusieurs variables symboliques.

- un intervalle (la variable "Prix" du tableau 1.1) ;
- des histogrammes (la variable "Type de moteur" du tableau 1.1).

Objets symboliques

Une classe d'individus est objet du monde réel sa description par des données symbolique est une modélisation appelée "objet symbolique" [44].

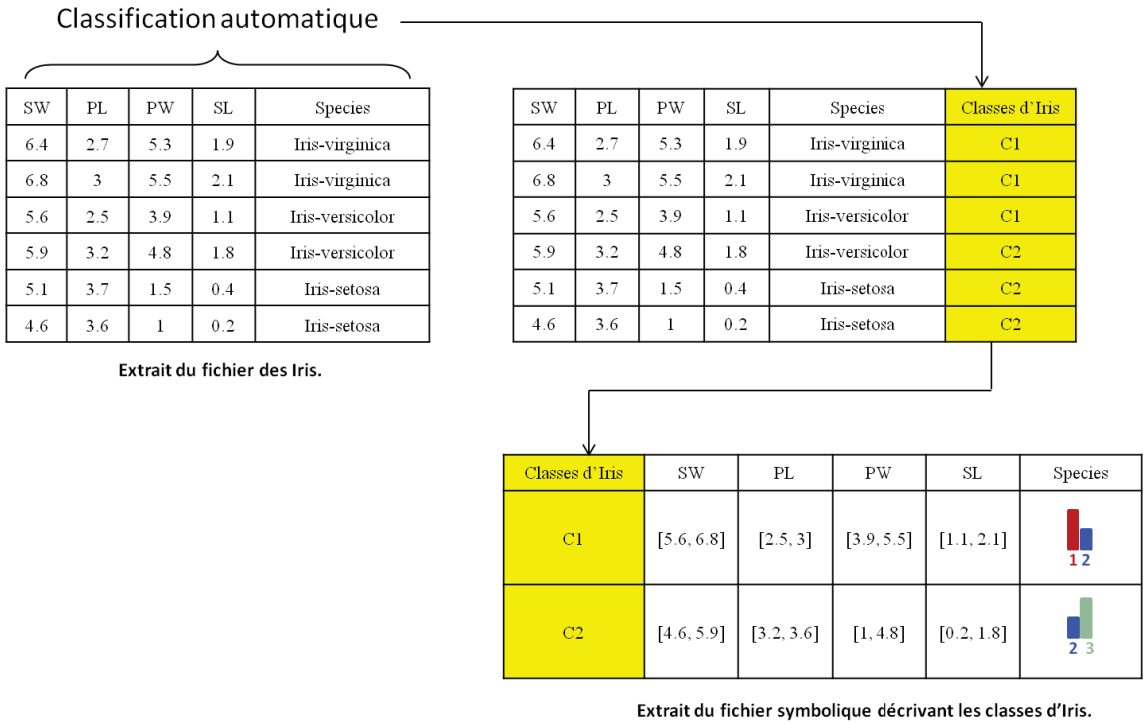


FIGURE 1.2 – Exemple de classes d'individus résultats d'une classification automatique.

Tableau de données symboliques

Un tableau de données est dit symbolique, s'il contient au moins une variable symbolique. Le tableau 1.1 représente un exemple de tableau symbolique décrivant les marques de voitures.

Marques	Modèles	Type de moteur	Année	Prix	Boîte à vitesse
BMW	{Série1, X5, X3}	(0.66)Diesel, (0.34)Essence	2014	[8500, 35000]	Automatique
Citroën	{C3, DS3, C5}	(0.34)Diesel, (0.66)Essence	2015	[6000, 25000]	Manuelle

Tableau 1.1 – Exemple d'un tableau symbolique décrivant les marques des voitures d'un parc automobile.

3 Étapes de l'ADS

Une ADS se fait en deux étapes [42][45]. La première concerne la construction du fichier symbolique à partir de données classiques. La deuxième étape représente l'étude de données symboliques obtenues en utilisant des méthodes d'analyse symbolique.

3.1 Construction des données symboliques

La création des données symboliques à partir de données classiques se fait en deux étapes. D'abord, il faut choisir ou construire les classes d'individus. Ensuite, il faut décrire ces classes par des variables symboliques résultant de la conversion des variables descriptives du classique au symbolique.

Transformation des variables du classique au symbolique

Pendant l'étape de la création des données symboliques, les variables sont traitées de façon à ce que leurs formes finales expriment la variation interne des variables initiales. Le tableau 1.2 résume la correspondance entre les différents types de variables descriptives du classique au symbolique. Ainsi nous distinguons quatre cas :

- si dans le tableau classique nous avons une variable qualitative qui prend la même valeur pour tous les individus d'une classe alors lors du passage au tableau symbolique, cette classe sera décrite par cette valeur qualitative unique.
- si les individus d'une classe sont décrits par une variable qualitative à valeurs multiples alors lors du passage aux données symboliques cette classe sera décrite par un digramme de fréquences. Chaque classe d'individus sera décrite par un digramme de fréquences.
- si dans le tableau classique nous avons une variable continue qui prend la même valeur pour tous les individus d'une classe alors lors du passage au tableau symbolique, cette classe sera décrite par cette valeur continue unique.
- si les individus d'une classe sont décrits par une variable quantitative continue, alors lors de la création des données symboliques cette classe sera décrite par un intervalle ou un histogramme. Chaque classe d'individus sera décrite par un intervalle ou un histogramme.

Exemple de création de données symboliques à partir d'un fichier classique [41]

Notre exemple traite l'ensemble de 600 oiseaux se trouvant sur une ile. Cet ensemble est composé de 200 hirondelles, 300 autruches et 100 pingouins. Chaque oiseau est

Individus décrits par une variable classique à valeur :	Classe d'individus décrite dans le tableau de données symboliques par une variable à valeur :
Qualitative unique	Qualitative unique
Qualitatives multiples	Histogrammes / diagrammes de fréquences
Quantitative unique	Quantitative unique
Quantitatives	Intervalles / histogrammes

Tableau 1.2 – Variables descriptives du classique au symbolique

décrit par son espèce, sa taille, sa couleur et sa capacité de voler ou non. A partir de ces données initiales nous voulons construire un fichier symbolique décrivant les espèces. Pour cela, nous prenons l'espèce comme classe. Chaque catégorie de la variable "Espèce" représentera une classe et sera décrite par :

- Une variable "Taille" : dans les données initiales cette variable est de type continu, donc elle sera automatiquement convertie en intervalle.
- Une variable "Vole" : dans les données initiales cette variable est de type qualitatif unique, donc elle gardera le même type.
- Une variable "Couleur" : dans les données initiales cette variable est de type qualitatif multiple donc elle sera convertie en histogramme. Une variable "Migre" : cette variable est appelée *variable conceptuelle* car elle s'applique aux classes d'individus.

La figure 1.3 résume cet exemple. Dans cette figure le premier tableau représente un extrait des données initiales qui décrivent les 600 oiseaux de différentes espèces alors que le deuxième tableau représente les données symboliques décrivant les espèces.

Généralement la conversion des données du classique au symbolique est faite automatiquement. En effet, tout logiciel d'analyse de données symbolique doit avoir un module qui permet la création des données symboliques à partir d'un ou de plusieurs fichiers de données classiques (voir section 4).

3.2 Extension des méthodes d'analyses classiques aux Données Symboliques

Dans la littérature, nous trouvons plusieurs méthodes d'analyse de données qui ont été adaptées aux données symboliques. Ces méthodes sont regroupées dans deux catégories : les méthodes non supervisées et celles supervisées.
Les méthodes non supervisées ont pour but de structurer et de simplifier les données

Tableau de données classiques

Oiseaux	Espèce	Vole	Couleur	Taille
1	Hirondelle	Oui	Noir	70
2	Pingouin	Non	Noir	80
:	:	:		:
600	Autruche	Non	Gris	125

Conversion classique → symbolique

Espèces	Vole	Couleur	Taille	Migre
Hirondelle	Oui	0.3n, 0.7g	[60, 85]	Oui
Pingouin	Non	0.1n, 0.9g	[70, 95]	Oui
Autruche	Non	0.5n, 0.5n	[85, 160]	non

Tableau de données symboliques

FIGURE 1.3 – Exemple de construction d'un tableau symbolique décrivant les différentes espèces d'oiseaux.

issues de plusieurs variables, sans en privilégier une par rapport à une autre. Elles prennent en entrée un fichier symbolique et retournent un autre fichier symbolique contenant le résultat de la méthode appliquée. Parmi ces méthodes nous citons :

- La méthode ***HIPYR*** [22] qui étend la méthode de classification hiérarchique et pyramidale aux données symboliques en se basant sur un tableau de données symboliques ou sur une matrice de dissimilarités.
- La méthode ***SCLUST*** [37] qui représente l'extension de la méthode des nuées dynamiques [39] aux données symboliques.
- La méthode ***DIV*** [28][29] qui est une méthode descendante de classification. Le résultat de cette méthode est un arbre binaire construit par partitionnement récursif.
- La méthode ***PCM*** [26, 30, 31] qui propose une extension de l'analyse en composantes principales aux données du type intervalle. Cette méthode produit la représentation graphique et les facteurs en prenant en compte la variation des intervalles et de leurs valeurs centrales.
- L'extension de l'analyse en composante principale aux données de type histogramme [43].

- L'extension des cartes de Kohonen aux données symboliques [47].

Les méthodes supervisées ont pour but d'expliquer une variable à l'aide de deux ou plusieurs variables explicatives, parmi ces méthodes nous trouvons :

- L'extension de la régression linéaire aux données symboliques ([10], [11], [36] et [3]).
- L'extension de l'algorithme Apriori et des règles d'association aux données symboliques ([2] et [4]).
- L'extension des arbres de décision aux données symboliques ([87], [1],[79],[77] et [97]). Toutes ces méthodes seront détaillées dans le chapitre 3.

4 Outils d'ADS

L'ADS est un domaine de recherche très étudié dans la littérature. Cependant, on ne trouve que quelques outils offrant la possibilité de construire et d'analyser des données symboliques. Les principaux outils d'ADS sont : SODAS, SYR et quelques *packages* dans R.

4.1 Le logiciel SODAS (*Symbolic Official Data Analysis System*)

SODAS [46] est un logiciel libre développé dans le cadre du projet européen EUROSTAT. Il permet la construction et l'analyse de données symboliques.
Les modules de cet outil peuvent être divisés en deux groupes : le premier est représenté par le module d'extraction de données symboliques à partir de données classiques, alors que le deuxième contient toutes les méthodes d'ADS implémentées dans SODAS. La figure 1.4 représente une vue d'ensemble de SODAS [1].

4.1.1 Le module d'extraction de données symboliques DB2SO

DB2SO est un module de construction de données symboliques à partir d'une base de données relationnelle ([99] et [59]). Il produit, à partir d'une base de données classiques, un fichier *XML* ou *SDS* (*Symbolic Data System*).

Afin d'extraire un fichier symbolique à partir d'une base de données relationnelle, il faut commencer par extraire la table initiale décrivant les individus à partir de la base initiale. La deuxième colonne de cette table doit impérativement contenir une variable nominale représentant les classes à étudier. Ensuite en utilisant DB2SO, il faut

1. https ://www.ceremade.dauphine.fr/SODAS/sodas-presentation.htm

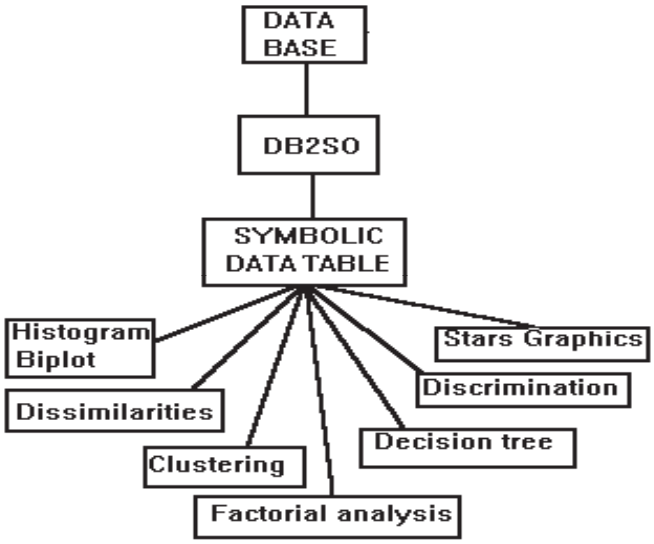


FIGURE 1.4 – Vue d'ensemble du logiciel SODAS

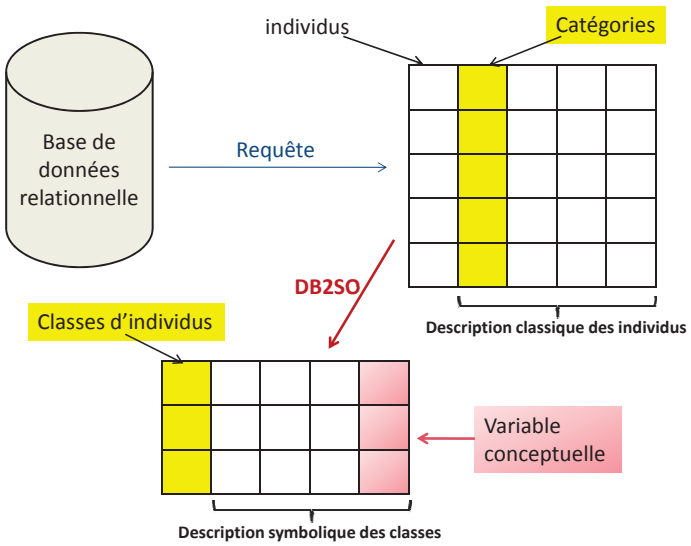


FIGURE 1.5 – Les différentes étapes du passage du classique au symbolique en utilisant DB2SO.

se connecter à la base initiale et choisir la table construite auparavant. Finalement, DB2SO construira la table des classes qui seront décrites par des variables symboliques en prenant en compte la variation interne des variables. La figure 1.5 [42] illustre cette démarche.

Exemple illustratif

Étant donné le tableau 1.3, extrait d'une base donnée de joueurs de football. Chaque joueur est décrit par cinq variables : son équipe, son âge, sa taille, sa nationalité et son poste sur le terrain. A partir de ce tableau nous construisons les données symboliques décrivant les équipes en utilisant DB2SO.

Joueur	Equipe	Age	Taille	Nationalité	Poste
Ronaldo	Real Madrid	29	186	Portugais	Attaquant
Pepe	Real Madrid	32	188	Portugais	Défenseur
Iniseta	FC Barcelone	30	170	Espagnol	Milieu
Xavi	FC Barcelone	34	170	Espagnol	Milieu
Messi	FC Barcelone	26	169	Argentin	Attaquant

Tableau 1.3 – Table de données classiques décrivant des joueurs de football.

Le tableau 1.4 représente le résultat obtenu où la première colonne représente la classe "Equipe". Chaque équipe est décrite par des variables intervalles, construites à partir des variables quantitatives "Age" et "Taille", et des variables histogrammes, construites à partir des variables qualitatives "Nationalité" et "Poste".

Equipe	Age	Taille	Nationalité	Poste
FC Barcelone	[26, 34]	[169, 170]	(1/3)Argentin, (2/3) Espagnol	(1/3)Attaquant, (2/3)Milieu
Real Madrid	[29, 32]	[186, 188]	(1)Portugais	(1/2)Attaquant, (1/2)Défenseur

Tableau 1.4 – Matrice de données symboliques décrivant les équipes de football, où chaque ligne décrit une équipe avec des variables symboliques de type intervalle ou histogramme.

4.1.2 Méthodes Symboliques implémentées dans SODAS

SODAS [2][46] implémente plusieurs modules qui permettent la manipulation et l'analyse de données symboliques. Le manuel utilisateur de SODAS [3] détaille toutes ces modules en donnant des exemples applicatifs. Les modules d'analyse symbolique de SODAS peuvent être regroupés comme suit :

2. https ://www.ceremade.dauphine.fr/SODAS/presentmetho.htm
3. https ://www.ceremade.dauphine.fr/SODAS/manutilisateur.htm

- Deux modules de calcul de dis-similarité entre objets symboliques : ***DI*** [48] et ***DIM (Distance of Modal Objects)***[82]. Le module ***DI*** permet de calculer la dis-similarité et de comparer les objets symboliques de sémantique booléenne. Ce module permet de choisir entre plusieurs méthodes de calculs de dis-similarité. Le module ***DIM*** permet le calcul de dis-similarité entre des objets symboliques de sémantique modale.

- Deux modules de classification automatique : ***DIV*** [28] et ***PYR*** [21]. Le module ***DIV*** implémente une méthode descendante de classification. Alors que le module ***PYR*** implémente l'extension de la classification hiérarchique aux données de type intervalle.

- Deux modules d'arbres de décision : ***TREE*** [87] et ***SDT***[18]. Le module ***TREE*** permet de construire un arbre de décision binaire à partir de données symboliques. Alors que le module ***SDT*** permet de construire un arbre de décision sur des données stratifiées.

- Un module d'analyse en composante principale ***PCM*** [30, 26]. Cette méthode représente l'extension de l'analyse en composante principale classique aux données symboliques décrites par des intervalles.

- Un module de visualisation des objets symboliques : ***SOE*** [83]. Il permet une visualisation (orientée utilisateur) des différents objets symboliques enregistrés dans un fichier SODAS. Il offre également la possibilité de modifier la description de ces objets symboliques (comme leurs libellés, le noms des variables descriptifs, etc.).

- Un module de calcul statistique : ***STAT*** [8]. Il permet d'effectuer des opérations statistiques sur les données symboliques.

La figure 1.6 représente l'enchainement à suivre pour configurer et exécuter toutes les méthodes d'analyse de données symboliques implémentées dans SODAS.

4.2 Le logiciel SYR

Ce logiciel a été implémenté par la société Syrokko[4]. Cette société est agréée organisme de recherche privé par le ministère de l'enseignement supérieur et de la recherche. Syrokko est un éditeur de logiciels de *data mining*.

Il s'appuie sur des méthodes statistiques très innovantes (issues de l'analyse de données symboliques "ADS") pour aller plus loin que les logiciels d'analyse de données

4. www.syrokko.com

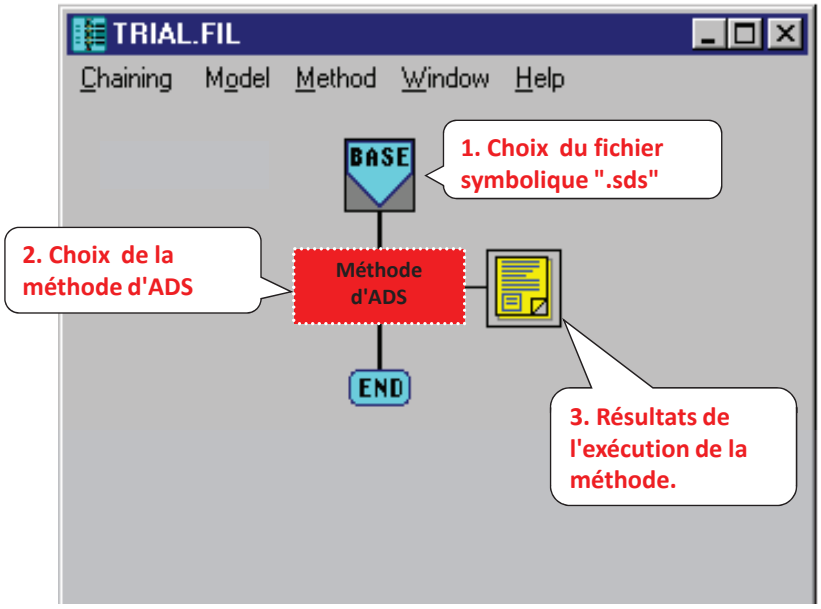


FIGURE 1.6 – Enchainement à suivre pour exécuter les méthodes d'analyse implémentées dans SODAS.

classiques. Elle propose à ses clients une plateforme logicielle, appelée SYR [5], leur permettant de mener de bout en bout une analyse de leurs données afin de trouver de nouveaux leviers d'action pour améliorer leurs compétitivités. Cette plateforme est composée de plusieurs modules. Afin de traiter toutes les étapes d'une ADS nous trouvons un module, nommé ***TabSyr***, qui permet l'extraction de données symboliques à partir de données classiques et des modules d'analyse de données symboliques (***ClustSyr***, ***NetSyr*** et ***StatSyr***) qui sont l'extension de méthodes classiques de *data mining* aux données symboliques. La figure 1.7 présente ces différents modules.

Le logiciel SYR a été utilisé pour effectuer différentes études de données au seins de Syrokko. Nous citons par exemple : l'étude de données médicales [94, 55], l'étude des données textuelles (le *textmining*) [56], l'étude des ouvrages d'art [17], l'étude de données sociales [57], etc.

4.2.1 Le module de construction et de manipulation de données symboliques TabSyr

TabSyr permet la construction de fichiers de données symboliques à partir de données classiques. Cet outil est capable de fusionner des fichiers hétérogènes (par leurs sources, formats, volumes et leurs types de données), en un tableau de données symboliques.

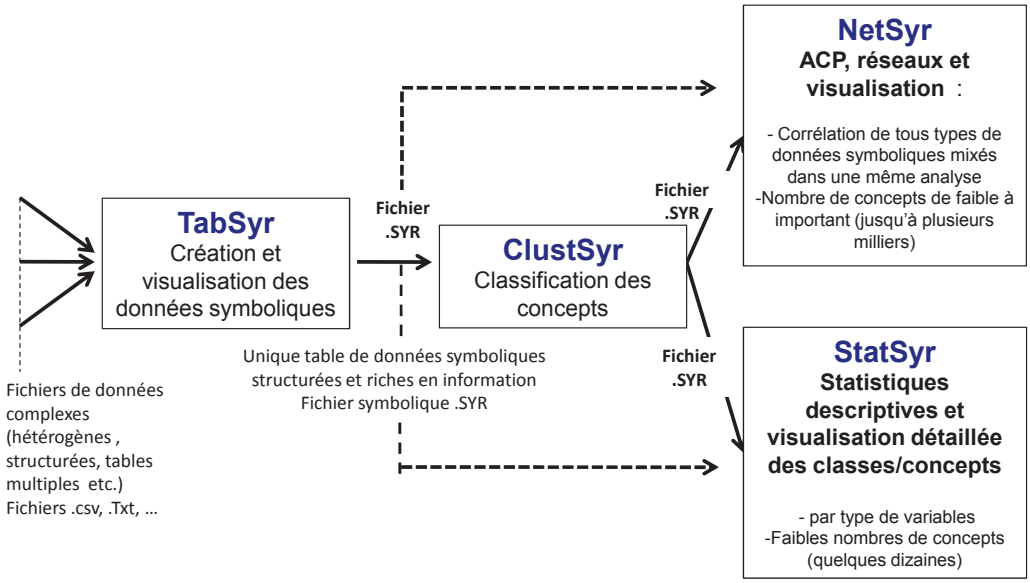


FIGURE 1.7 – Différents modules du logiciel Syr.

TabSyr accepte en entrée un ou plusieurs fichiers de données (au format .csv, .txt, etc.), où chaque ligne correspond à un individu et les colonnes sont les variables descriptives des individus. En sortie, ***TabSyr*** renvoie un fichier symbolique au format spécifique ***".syr"***. Ces fichiers peuvent être ensuite importés par l'ensemble des modules du logiciel SYR (***ClustSyr***, StatSyr et NetSyr). Ainsi, le résultat de ***TabSyr*** constitue l'entrée de l'ensemble des modules du logiciel SYR.

En plus de la création des données symboliques, ***TabSyr*** permet la visualisation graphique et la manipulation de la matrice de données symboliques. La figure 1.8 représente la visualisation d'un fichier symbolique décrivant les espèces d'oiseaux. Elle montre la structure d'un fichier ***.syr*** où chaque ligne décrit une des classes et chaque colonne représente la description d'une variable symbolique.

Ce module intègre plusieurs méthodes de tri pour les variables intervalles et histogrammes. Il offre la possibilité de masquer ou de supprimer des variables ou des classes d'individus, l'ordonnancement des lignes et des colonnes, une méthode de scoring symbolique permettant de trier les variables de la plus discriminante à la moins discriminante des classes, etc. Toutes ces méthodes sont de nouvelles méthodes

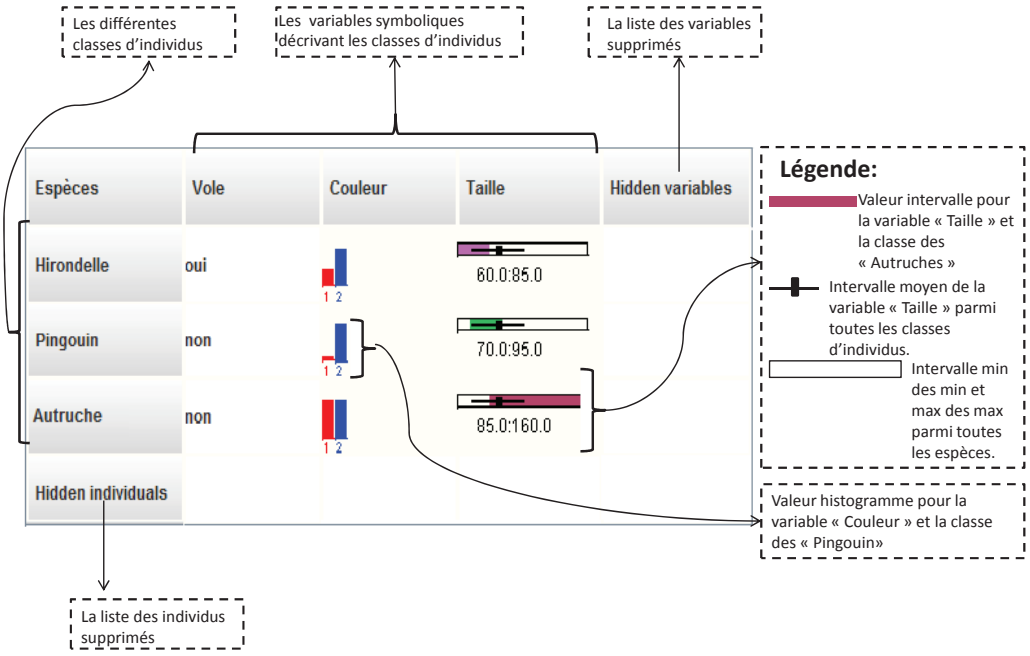


FIGURE 1.8 – Exemple de tableau de données symboliques visualisé dans TABSYR. Une matrice de données symboliques peut contenir dans chaque case : un histogramme, un intervalle, une valeur continue ou nominale.

statistiques, sur des données agrégées et fusionnées, développées par Syrokko [5].

Création et visualisation d'un fichier Symbolique à partir d'un fichier classique en utilisant *TabSyr*

Afin de construire un fichier symbolique à partir d'un fichier classique, en utilisant ***TabSyr***, il faut disposer du fichier des données classiques initiales et d'un fichier ***"pattern"*** contenant la description du fichier symbolique (voir figure 1.9).

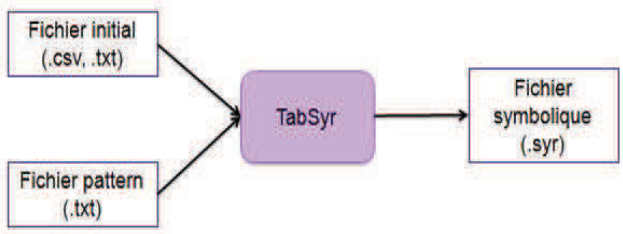


FIGURE 1.9 – Étapes de construction d'un fichier symbolique en utilisant ***TabSyr***.

Chaque ligne du fichier pattern représente la description d'une variable du fichier symbolique. Elle doit avoir la forme suivante :

Nom variable : numéro colonne : type

avec :

- Nom variable : représente le nom de la variable symbolique
- Numéro colonne : représente le numéro de la colonne de la variable classique dans le ficher initial.
- Type : le type de la variable dans le fichier symbolique. Qui peut être :
  - **c** : pour désigner la classe des individus.
  - **h** : pour les variables histogrammes
  - **i** : pour les variables intervalles
  - **q** : pour les variables quantitatives (continues)
  - **n** : pour les variables qualitatives (nominales)

Exemple illustratif

Soient les données décrivant des joueurs de football représentées par le tableau 1.3. Afin d'obtenir le fichier "***.syr***" décrivant les équipes, nous commençons par la construction du fichier pattern. La partie (a) de la figure 1.10 présente ce fichier. Ensuite, nous introduisons le pattern et le fichier de données initiales à ***TabSyr***. Enfin, nous obtenant le fichier "***.syr***" décrivant les équipes de football. La partie (b) de la figure 1.10 présente le tableau symbolique qui décrit les équipes de football. La visualisation graphique de ce fichier est représentée par la partie la figure 1.11.

Manipulation d'un fichier symbolique avec *TabSyr*

En plus de la création de fichier symbolique à partir d'un fichier classique, ***TabSyr*** permet de traiter les fichiers symboliques. Ce traitement peut se faire à travers le tri et la suppression des variables descriptives existantes ou en ajoutant d'autres variables descriptives issues d'autres fichiers.
La partie (a) de la figure 1.12 représente le résultat du tri des variables descriptives des équipes de football de la moins discriminante à la plus discriminante. Nous remarquons que "Taille" est la variable qui permet de différencier le plus les deux équipes.
La partie (a) de la figure 1.12 illustre l'exemple de suppression de la variable explicative "Nationalité" à partir du fichier décrivant les équipes de football.

Equipe:2:c
Age:3:i
Taille:4:i
Nationalité:5:h
Poste:6:h

(a) Fichier pattern.

A	B	C	D	E
Equipe	Age	Taille	Nationalité	Poste
Concept	Interval	Interval	Histogram	Histogram
-	-	-	1:Argentin,2:Espagnol,3:Portugais	1:Attaquant,2:Défenseur,3:Milieu
FCBarcelone	26.0:34.0	169.0:170.0	1(0.333)2(0.666)	1(0.333)3(0.666)
RealMadrid	29.0:32.0	186.0:188.0	3(1.0)	1(0.5)2(0.5)

(b) Fichier symbolique décrivant les équipes de football.

FIGURE 1.10 – Utilisation de ***TabSyr*** pour la création du fichier symbolique décrivant les équipes de football.

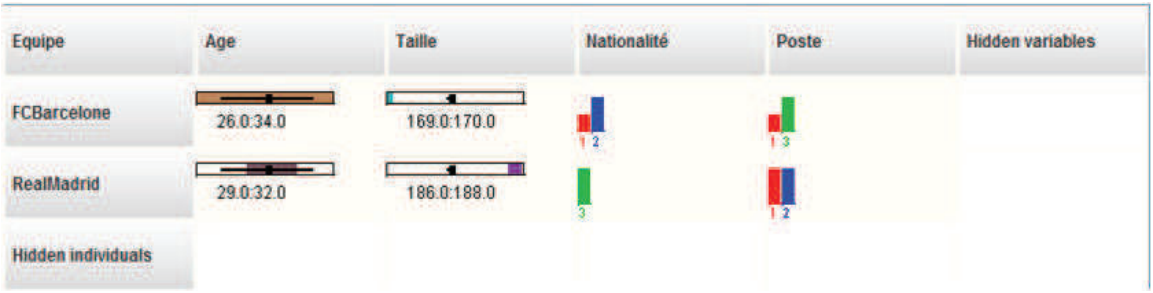
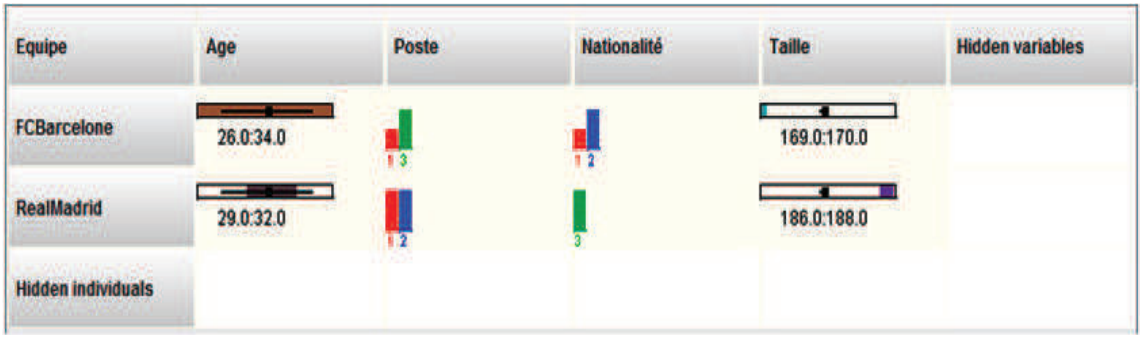
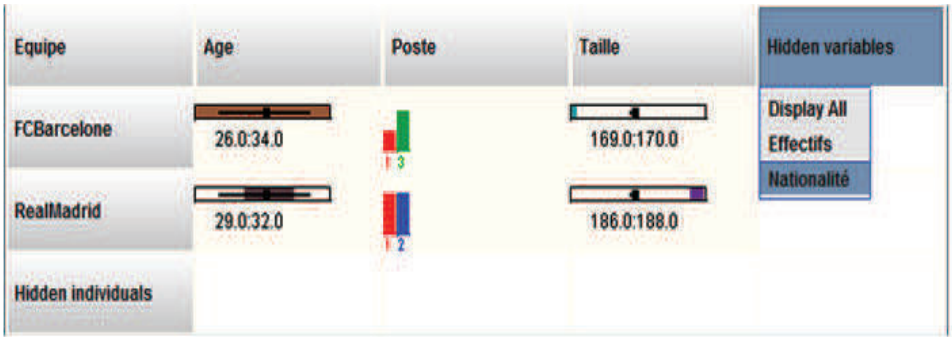


FIGURE 1.11 – Visualisation graphique du fichier symbolique des équipes de football.

L'ajout de variables descriptives se fait en fusionnant deux fichiers symboliques. En reprenant l'exemple des équipes de football, considérons que nous avons un autre fichier classique décrivant les supporters des équipes (voir tableau 1.5). Afin d'ajouter les informations des supporters de chaque équipe, nous construisons le fichier pattern décrivant les équipes en utilisant les informations de la table des supporters. Ensuite, nous ajoutons le résultat de la conversion du ficher des supporters en données symbolique au premier fichier décrivant les équipes issu de la table des joueurs. Le fichier pattern ainsi que la visualisation du fichier issue de la fusion sont représentés dans la figure 1.13.



(a) Visualisation du résultat du tri des variables descriptives par ordre croissant de discrimination.



(b) Suppression de la variable descriptive "Nationalité" en utilisant ***TabSyr***.

FIGURE 1.12 – Exemples de traitements effectués en utilisant ***TabSyr*** sur le fichier symbolique décrivant les équipes de football.

Nom	**Équipe**	**Age**	**Genre**
Louis Ferrer	Real Madrid	20	M
Francesca Rodriguez	FC Barcelone	30	F
Maria Costa	FC Barcelone	24	F
Antonio Hernandez	FC Barcelone	45	M
Rafael Nadal	Real Madrid	28	M
Isabella Ferrer	Real Madrid	50	F

Tableau 1.5 – Table des supporters des équipes de football.

4.2.2 Modules d'ADS implémentés dans SYR

Afin de couvrir tout le cycle d'ADS, le logiciel SYR offre la possibilité d'analyser les fichiers construits avec ***TabSyr*** à travers l'extension de plusieurs méthodes aux données symboliques. Les principaux modules sont : ***ClustSyr***, ***StatSyr*** et ***NetSyr***.

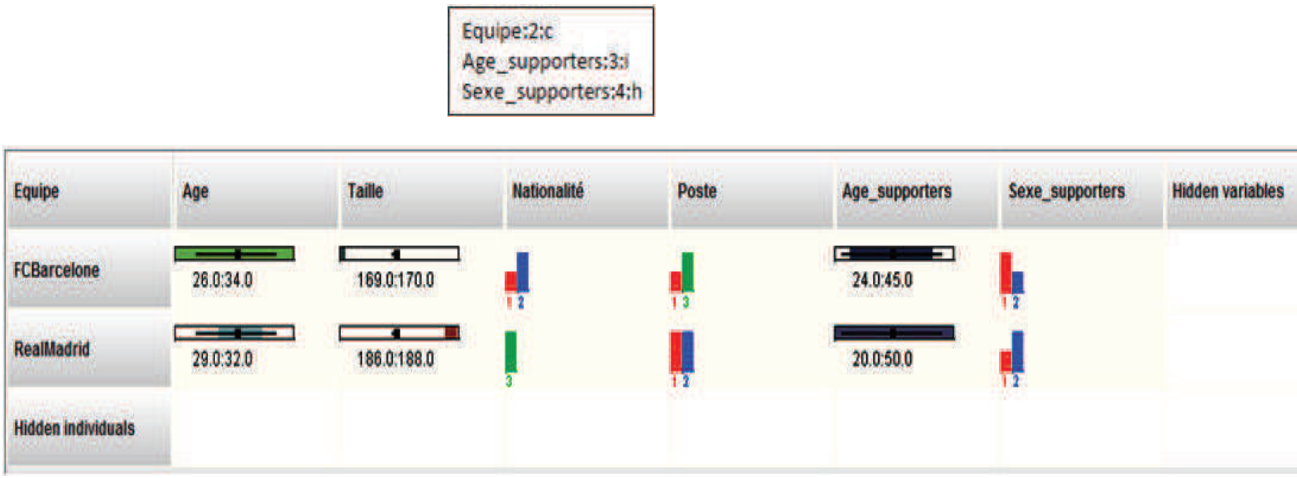


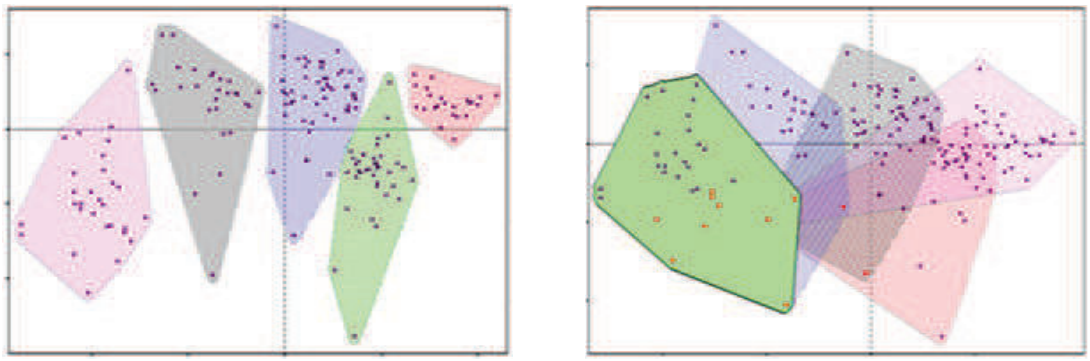
FIGURE 1.13 – Utilisation de ***TabSyr*** pour la création et la visualisation graphique du fichier Symbolique décrivant les équipes de football.

ClustSyr est un module de classification non supervisée de données symboliques. Il implémente l'extension de la méthode des nuées dynamiques[39] aux données symboliques en ayant la possibilité de mélanger différents types de variables explicatives (intervalles, histogrammes, continues, nominales, etc.). ***ClustSyr*** permet d'obtenir les classes les plus homogènes selon un critère d'inertie pour deux types de classifications :

- par partitionnement : un individu appartient à une et une seule classe. La figure 1.14a illustre le résultat d'une classification par partitionnent réalisée en utilisant ***ClustSyr***.
- par recouvrement un individu appartient à une ou plusieurs classes. La figure 1.14b illustre le résultat d'une classification par recouvrement réalisée en utilisant ***ClustSyr***.

StatSyr implémente l'extension de la statistique descriptive aux données symboliques fusionnées, agrégées et éventuellement issues d'une classification. Il propose des outils pour la recherche des variables intervalles et des catégories des variables histogrammes caractérisant les classes en visualisant leurs variations. ***StatSyr*** propose également des outils pour la recherche et la visualisation des corrélations entre variables symboliques intervalles et histogrammes.

NetSyr implémente l'extension de plusieurs méthodes d'analyse aux données symboliques. Nous citons par exemple : l'analyse en composante principale (ACP), les réseaux sociaux, la représentation de trajectoires chronologiques et la visualisation



(a) Classification par partitionnement en 5 classes. (b) Classification par recouvrement en 5 classes.

FIGURE 1.14 – Présentation des résultats de deux méthodes de classification (par partitionnement et par recouvrement) créés avec "***ClustSyr***" et visualisés avec "***NetSyr***".

des partitions et des recouvrements étendus aux données symboliques (voir figure 1.14). ***NetSyr*** propose une ACP symbolique permettant de manipuler différents types de variables symboliques (histogrammes, intervalles, etc.) et des données hétérogènes (provenant de sources diverses telles que textuelles, quantitatives, chronologiques, etc.).
NetSyr permet de visualiser un ensemble d'outils statistiques originaux et innovants :

- Projection d'un réseau sur le plan factoriel ;
- Projection des valeurs des variables symboliques de tous les types sur le plan factoriel ;
- Projection simultanée de plusieurs variables symboliques combinées pour la visualisation des corrélations ;
- Le partitionnement ou la projection d'un partitionnement des individus sur le plan factoriel grâce à une communication directe avec le module de classification ***ClustSyr*** ;
- Le recouvrement ou la projection d'un recouvrement des individus sur le plan factoriel grâce à une communication directe avec le module de classification ***ClustSyr*** ;

La figure 1.15 illustre un exemple de plan factoriel du module ***NetSyr***, avec la représentation d'un ensemble de classes (fonds d'investissement) décrits par des variables symboliques. Cette figure montre un exemple de projection d'une variable

histogramme "rendements quotidiens des fonds" sous la forme de camemberts pouvant être zoomés par des diagrammes en bâtons. Cet exemple propose la projection d'un partitionnement des classes (ici les fonds) en 3 classes.

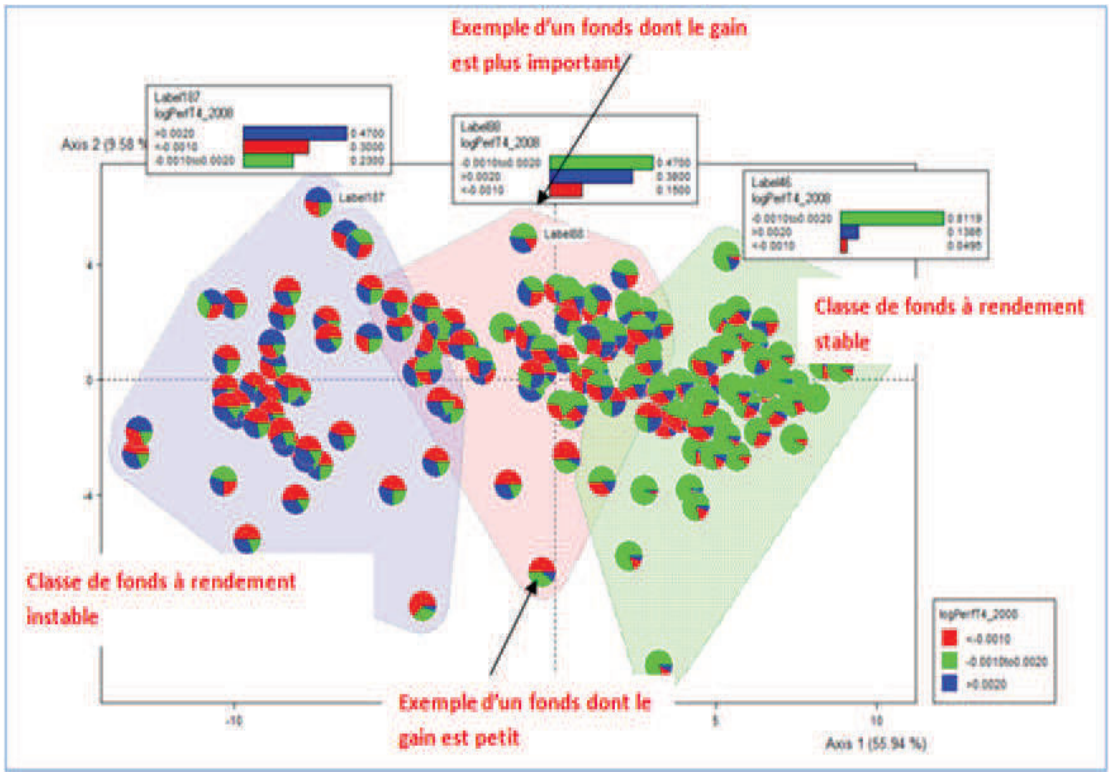


FIGURE 1.15 – Exemple de plan factoriel du module NETSYR, avec représentation d'un ensemble de classes (ex. : fonds d'investissement) décrits par des variables symboliques, + la projection d'une variable histogramme (ex. rendements quotidiens des fonds) sous la forme de camemberts pouvant être zoomés par des diagrammes en bâtons, + la projection d'un partitionnement des classes (les fonds) en 3 classes.

4.3 Les librairies d'ADS dans R

R est un logiciel de statistique [61]. Il représente à la fois un langage informatique et un environnement de travail. C'est un logiciel gratuit à code source ouvert qui sert à manipuler des données, à tracer des graphiques et à faire des analyses statistiques.
R offre la possibilité d'implémenter et d'ajouter de nouveaux modules d'analyse de données sous format de librairies *package*. Parmi ces *packages*, nous trouvons ceux dédiés à l'ADS comme ***RSDA*** [85], ***Clamix*** [6] et ***HistDAWass*** [62]. Ces *packages* sont décrits

dans le site internet du *"Comprehensive R Archive Network (CRAN)"* [5]. Les principales fonctionnalités de chaque *package* seront détaillées dans ce qui suit.

4.3.1 RSDA- R to Symbolic Data Analysis

Ce *package* de R permet de créer, d'importer et d'étudier des fichiers symboliques en utilisant quelques méthodes d'analyse de données symboliques.

Création de fichiers symboliques en utilisant *RSDA*

RSDA génère et utilise des fichiers symboliques avec un format propre à lui. Ces fichiers sont enregistrés comme étant des fichiers ".csv" avec une en-tête bien spécifique. Il intègre trois méthodes permettant la création de ce type de fichiers symboliques :

- à partir d'une table de données classiques à travers la méthode*classic.to.sym*.
- à partir d'un fichier Sodas (.sds) en utilisant la méthode *SODAS.to.RSDA*.
- à partir d'un fichier csv en utilisant la méthode *generate.sym.table*.

Des exemples d'utilisation de ces méthodes ainsi que l'explication détaillée du format utilisé se trouvent dans la documentation de ce *package* sur le site *CRAN* [84].

Méthodes d'analyse de fichiers symboliques

RSDA implémente certaines méthodes de classification automatique sur des données symboliques ainsi que quelques modèles linéaires. Il permet de traiter les variables de type intervalle et celles de type histogramme.
Pour les variables intervalles, ce *package* intègre des fonctions qui calculent : la moyenne, la médiane, la moyenne des valeurs extrêmes, l'écart type, l'écart inter-quartile et la corrélation. En plus des méthodes de calcul statistique, cette bibliothèque implémente une méthode d'ACP en proposant trois façons pour projeter les variables intervalles dans le cercle de corrélations, ce qui permet de constater la variation ou l'inexactitude des variables. Ce *package* offre aussi une fonction d'analyse multidimensionnelle des données de type intervalle nommée *"INTERSCAL"*.
Pour les variables histogrammes, ce *package* propose une méthode d'ACP traitant les histogrammes.

5. http ://cran.r-project.org

4.3.2 Clamix

Cette librairie implémente deux méthodes de classification d'Objets Symboliques (OS). Ces derniers représentent les variables sous format d'histogrammes. La création de ces OS est faite manuellement en utilisant des fonctions propres à cette librairie et en spécifiant pour chaque variable du fichier initial les modalités qui seront prises en compte. Les méthodes de classification symbolique implémentées dans cette librairie sont l'extension de deux méthodes classiques de classification : la méthode des K-Means [39] et la classification hiérarchique [107]. L'analyse des données en utilisant cette librairie suit le schéma représenté dans la figure 1.16 [6].

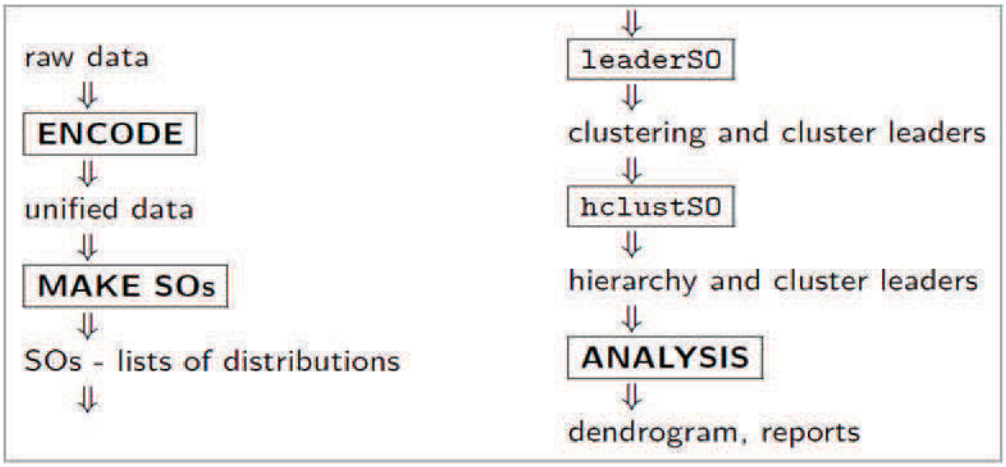


FIGURE 1.16 – Étapes d'analyse de données en utilisant la librairie "Clamix" de R [6].

4.3.3 HistDAWass : Histogram-Valued Data Analysis

Cette bibliothèque traite les données décrites par des histogrammes. Ce traitement est fait en proposant des méthodes prédéfinies permettant de calculer les valeurs statistiques pour ce type de données qui sont basées sur la distance L2 de Wasserstein [95]. Ce *package* propose aussi des méthodes de classification, de régression et des outils pour les données décrites par des histogrammes et pour les séries temporelles.

4.4 Comparaison entre les différents outils d'ADS

Nous avons présenté les principaux outils permettant de créer, de traiter et d'analyser des données symboliques. Chacun de ces outils a ses avantages et ses inconvénients que nous avons résumés dans le tableau 1.6. A partir de ce tableau, nous pouvons constater que malgré la richesse de SODAS cet outil n'est pas bien maintenu,

ce qui rend son utilisation quasi-impossible par les nouveaux utilisateurs. Cependant, le logiciel SYR ainsi que les bibliothèques R sont en évolution continue et peuvent être considérés comme l'avenir du développement des méthodes d'ADS.

Tableau 1.6 – Comparaison entre les différents outils d'ADS.

Outils	Points forts	Points faibles
SODAS	• L'outil le plus complet : il intègre le plus de méthodes d'ADS. • Il offre une interface graphique qui facilite sa manipulation avec une documentation très riche. • Toutes les méthodes sont intégrées dans le même logiciel. • Logiciel gratuit.	• Absence de maintenance du logiciel qui présente plusieurs problèmes (*Bugs*) notamment avec les nouveaux systèmes d'exploitation (comme Windows 8). • Absence de modules permettant la manipulation des bases non structurées. • Absence des nouvelles méthodes d'ADS, la méthode la plus récente date de 2003.

Tableau 1.6 – Comparaison entre les différents outils d'ADS (suite).

Outils	Points forts	Points faibles
SYR	• En évolution avec l'intégration progressive de nouvelles méthodes de création et de manipulation de données symboliques. • Des modules avec des interfaces graphiques faciles à comprendre par les nouveaux utilisateurs avec une documentation très riche. • Le module de création de données symbolique est très complet. Il permet l'extraction de données symboliques à partir de tous types de bases de données (structurées ou pas), ainsi que la conversion des fichiers ".sds" en ".syr".	• Des modules qui mettent parfois beaucoup de temps pour retourner le résultat (comme ***ClustSyr***), non adaptés à traiter les grandes bases de données. • Absence de modules qui intègrent des méthodes prédictives (comme les arbres de décision, les règles d'association, etc.) • Modules non rassemblés dans le même logiciel. Chaque module nécessite sa propre installation. • Logiciel payant.
Packages R	• Des librairies en évolution continue avec plusieurs équipes qui travaillent à intégrer de nouvelles méthodes d'ADS dans R. • Une documentation très riche. • Librairies gratuites.	• Non uniformité des formats des fichiers symboliques manipulés et générés par les différentes bibliothèques. • Absence d'interface graphique qui pourra faciliter l'utilisation et le test des différentes méthodes. • L'obligation de maitriser **R** pour pouvoir utiliser les différentes bibliothèques.

5 Conclusion

Dans ce chapitre, nous avons présenté quelques aspects théoriques de l'ADS ainsi que les principaux outils permettant la création et l'analyse de données symboliques. A partir de ce que nous avons présenté, l'ADS présente plusieurs avantages par rapport aux approches classiques qui peuvent être résumés comme suit :

- L'ADS permet de résumer les données en conservant beaucoup plus d'informations que les méthodes classiques d'agrégation (centre de gravité, moyenne, variance, etc.).
- L'étude des données peut se faire sur plusieurs niveaux de généralisation (individus $\longrightarrow$ classes d'individus $\longrightarrow$ classes de classes d'individus).
- L'approche symbolique résout le problème de confidentialité. En effet, les individus de départ (les données confidentielles) n'apparaissent plus dans la description des objets symboliques.
- L'approche symbolique permet de résoudre le problème de redondance des données, en offrant la possibilité de construire des données symboliques à partir de sources multiples sans passer par la construction d'une base regroupant toutes les informations initiales.

Dans la suite de notre travail, nous allons enrichir le logiciel SYR par trois modules. Le premier entrera dans le cadre de la première étape d'ADS en convertissant les variables continues en histogrammes. Alors que le deuxième module concernera l'ajout d'une méthode de construction, de manipulation et de test d'arbres de décision symbolique. Enfin, le troisième module présentera une extension de la création des données symboliques aux Big Data.

CHAPITRE 2

Extraction des histogrammes les plus discriminants à partir d'une variable continue (HistSyr)

1 Introduction

Le passage des données classiques aux données symboliques est une étape importante dans le processus d'ADS. Plusieurs travaux ont été effectués afin de transformer les variables classiques en symboliques. Dans la plupart de ces travaux, les variables continues ont été transformées en intervalles. Toutefois, il est intéressant de discrétiser les variables continues afin de les transformer en histogrammes lors de la création des données symboliques. Dans la littérature, aucune méthode d'ADS n'a traité la conversion automatique d'une variable continue en histogrammes. Pour remédier à ceci, nous avons créé une méthode qui automatise cette opération.

Cette méthode a été inspirée des méthodes classiques de discrétisation qui convertissent les variables du quantitatives aux qualitatives. Ces méthodes sont devenues un sujet d'étude très prisé à partir du début des années 90. Dès lors, on réalise l'importance de cette étape dans l'intégration et l'utilisation des variables quantitatives dans les différentes méthodes d'apprentissages qui n'acceptaient que les variables qualitatives.

Ce chapitre est organisé en trois sections. D'abord, nous donnons un aperçu sur les différentes méthodes classiques de discrétisation. Ensuite, dans la deuxième section, nous présentons notre méthode d'extraction d'histogrammes à partir d'une variable continue, appelée ***HistSyr***. Enfin, nous exposons les résultats de la comparaison d'HistSyr avec d'autres méthodes de discrétisation sur des données UCI[13] et sur des données issues d'études effectuées à Syrokko.

2 Discrétisation d'une variable continue : état de l'art

2.1 Discrétiser ?

Discrétiser une variable quantitative c'est transformer un vecteur de nombres réels en un vecteur de nombres entiers. Ces derniers représentent les rangs des intervalles de discrétisation. Concrètement, c'est le fait de regrouper les données en modalités, séparées par des seuils.

Soit une variable continue $y = \{y_1, y_2, .., y_n\}$ de n valeurs, ayant D_y comme domaine de définition.
Discrétiser y revient à découper D_y en K intervalles disjoints $I_k, (k = 1, .., K)$ où chaque

intervalle sera représenté par son rang :

$$\begin{aligned} I_1 &=]-\infty, d_1[\\ &\vdots \\ I_j &= [d_{j-1}, d_j[\\ &\vdots \\ I_k &= [d_{K-1}, +\infty[\end{aligned}$$

L'objectif des différentes méthodes de discrétisation est donc de trouver les $K-1$ bornes notées d_i qui optimiseront un certain critère.

2.2 Méthodes de discrétisation

Dans la littérature, nous trouvons plusieurs méthodes de discrétisation appliquées à des séries numériques. La discrétisation a été utilisée en statistiques afin de transformer une série numérique en classes. Elle a été aussi introduite dans le domaine du *data mining* afin de permettre l'utilisation des variables continues par des algorithmes qui n'acceptent que des variables discrètes en entrée.

Les méthodes de discrétisation peuvent être classifiées en méthodes supervisées et d'autres non supervisées [116].

- **Les méthodes de discrétisation non supervisées** : ne considèrent que les valeurs numériques de la variable à discrétiser. Ces méthodes sont utilisées surtout dans le domaine de la statistique.

- **Les méthodes de discrétisation supervisées** : prennent en considération les valeurs de la variable à discrétiser et celles de la classe associée à chaque valeur. Généralement, ces méthodes sont utilisées dans les algorithmes d'apprentissage supervisés.

2.2.1 Méthodes de discrétisation non supervisée

Ces méthodes prennent en entrée une série numérique et retournent un ensemble de modalités séparées par des seuils. Elles sont aussi appelées méthodes non contextuelles [116] puisqu'il n'y a aucune considération de la classe associée à chaque valeur lors du processus de discrétisation. Les méthodes les plus connues sont :

- l'algorithme de Fisher [51].

- des méthodes basiques comme la discrétisation selon des seuils observés, le découpage en classes d'égale amplitude, la discrétisation selon les quantiles, la discrétisation selon la moyenne emboîtée, etc.
- des méthodes qui prennent en compte les lois de probabilités suivi par la variable à discrétiser [69].

2.2.1.1 L'algorithme de Fisher

L'algorithme de Fisher [51] a pour but de déterminer la partition optimale, en un nombre donné de classes, d'une population décrite par une seule variable continue. La partition optimale est obtenue en minimisant la variation intra-classes (voir equation 2.1). Cet algorithme retourne une solution optimale avec une complexité quadratique par rapport au nombre des individus.

Formulation du problème

Étant donné un ensemble Ω de n individus $\{\omega_1, \ldots, \omega_n\}$ décrit par une variable continue y tel que $y(\omega_i) = v_i$. Le problème consiste à chercher la meilleure partition, notée $P^* = \{P_1^*, P_2^*, \ldots, P_k^*\}$, de Ω en k classes en minimisant l'inertie intra-classes W tel que :

$$W = \sum_{i=1}^{k} \sum_{\omega_i, \omega_j}^{\in P_i^*} D(\omega_i, \omega_j) \tag{2.1}$$

avec D est une mesure de dissimilarité (la distance euclidienne L1 par exemple).

Présentation de l'algorithme

Les principales étapes de l'algorithme sont :

1. Trier les valeurs de la variable y par ordre croissant.
2. Suivant la valeur de k (le nombre de classes) nous distinguons trois cas :
   - Si $k = 2$ (recherche de la partition optimale en 2 classes) : dans ce cas, il faut évaluer toutes les partitions en deux classes en translatant un indice $i (i = 1, \ldots, n-1)$ et retenir la partition qui minimise W (voir equation 2.1).
   - Si $k = 3$ (recherche de la partition optimale en 3 classes) : dans ce cas,nous utilisons deux indices $i (i = 1, \ldots, n-2)$ et $j (j = i+1, \ldots, n-1)$. La recherche de la meilleur partition se déroule en translatant les deux indices où pour chaque valeur de i nous calculons la sous-partition optimale en 2 classes (en

translatant j) et nous la gardons en mémoire. La partition optimale en trois classes est celle qui minimise W.

- Si $k > 3$: dans ce cas,nous utilisons deux indices $i(i = 1, \ldots, n-(k-1))$ et $j(j = i+1, \ldots, n-(k-2))$. Dans ce cas pour chaque valeur de l'indice $i(i = 1, \ldots, (n-k-1))$, il faut calculer les sous-partitions optimales en $(k-1)$ classes. Par exemple pour un $k = 4$, il faut avoir calculé, au préalable, les sous-partitions optimales en 3 classes et les garder en mémoires. Pour chaque indice j, nous utilisons les sous-partitions optimales en deux classes calculées pour $k = 3$

Propriété de l'algorithme

Il a été démontré [51] que la solution optimale P^* est constituée de classes adjacentes. Si une partition $P^* = \{P_1^*, P_2^*, \ldots, P_k^*\}$ est optimale pour la population Ω alors la partition $\{P_2^*, \ldots, P_k^*\}$ est une partition optimale pour $\Omega \setminus P_1^*$.

Exemple illustratif

Soit un ensemble de 5 individus $\{\omega_1, \ldots, \omega_5\}$ décrits par une variable y représenté par le tableau 2.1.

	ω_1	ω_2	ω_3	ω_4	ω_5
y	1	4	2.5	2	5

Tableau 2.1 – Exemple d'une population décrite par une variable continue y.

Notre objectif est d'appliquer l'algorithme de Fisher pour obtenir la partition optimale en 4 classes. Pour cela nous commençons par le tri des valeurs par ordre croissant ce qui donne : $\{\omega_1, \omega_4, \omega_3, \omega_2, \omega_5\}$ la recherche des sous partitions optimales en 3 classes : Pour $k = 3$ nous avons :

i	j	**partitions en 3 classes**	**Critère** W_3
1	2	$\{\omega_1\}, \{\omega_4\}, \{\omega_3, \omega_2, \omega_5\}$	$W = 0 + 0 + (1.5 + 2.5 + 1) = 5$
1	3	$\{\omega_1\}, \{\omega_4, \omega_3\}, \{\omega_2, \omega_5\}$	$W = 0 + 0.5 + 1 = 1.5$
1	4	$\{\omega_1\}, \{\omega_4, \omega_3, \omega_2\}, \{\omega_5\}$	$W = 0 + (0.5 + 2 + 1.5) + 0 = 4$
2	3	$\{\omega_1, \omega_4\}, \{\omega_3\}, \{\omega_2, \omega_5\}$	$W = 1 + 0 + 1 = 2$
2	4	$\{\omega_1, \omega_4\}, \{\omega_3, \omega_2\}, \{\omega_5\}$	$W = 1 + 1.5 + 0 = 2.5$
3	4	$\{\omega_1, \omega_4, \omega_3\}, \{\omega_2\}, \{\omega_5\}$	$W = (1 + 1.5 + 0.5) + 0 + 0 = 3$

Nous concluons que $\{\omega_1\}, \{\omega_4, \omega_3\}, \{\omega_2, \omega_5\}$ est la partition optimale à 3 classes. et nous gardons en mémoire les meilleurs sous partition pour chaque indice i

i	**meilleurs partitions en 2 classes**
1	$\{\omega_4, \omega_3\}, \{\omega_2, \omega_5\}$
2	$\{\omega_3\}, \{\omega_2, \omega_5\}$
3	$\{\omega_2\}, \{\omega_5\}$

i	j	**partitions en 3 classes**	**Critère** W_3
1	2	$\{\omega_1\}, \{\omega_4\}, \{\omega_3\}, \{\omega_2, \omega_5\}$	$W = 0 + 0 + 0 + 1 = 1$
1	3	$\{\omega_1\}, \{\omega_4, \omega_3\}, \{\omega_2\}, \{\omega_5\}$	$W = 0 + 0.5 + 0 + 0 = 0.5$
2	3	$\{\omega_1, \omega_4\}, \{\omega_3\}, \{\omega_2\}, \{\omega_5\}$	$W = 1 + 0 + 0 + 0 = 1$

Enfin , nous pouvons calculer la partition optimale en 4 classes :
La partition optimale en 4 classes est : $P^* = \{\{\omega_1\}, \{\omega_4, \omega_3\}, \{\omega_2\}, \{\omega_5\}\}$.

2.2.1.2 Discrétisation selon les seuils observés

Cette méthode se base sur les particularités de la distribution et sur les observations de l'utilisateur.

Présentation de l'algorithme

Pour discrétiser une série numérique en utilisant la discrétisation selon les seuils observés, il faut :

1. Construire l'histogramme des valeurs, ou le diagramme de fréquences, ou la courbe des fréquences cumulées triées par ordre croissant.
2. Déterminer les limites des classes en fonction des discontinuités apparentes sur le graphique.

Exemple illustratif : discrétisation de la taille d'une population

Soit une population $\Omega = \{\omega_1, \ldots, \omega_{15}\}$ décrite par une variable "Taille". Le tableau 2.2 contient les différentes valeurs de cette variable.

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}	ω_{11}	ω_{12}	ω_{13}	ω_{14}	ω_{15}
1.60	1.50	1.10	1.20	1.75	1.55	1.90	2.00	1.80	1.50	1.75	1.70	1.80	1.60	1.92

Tableau 2.2 – Exemple d'une population décrite par une variable continue "Taille".

D'après la figure 2.1, la discrétisation de cette variable en appliquant la méthode de discrétisation des seuils observés donnera cinq classes :

- Classe 1 : Taille $<$ 1.5
- Classe 2 : 1.5 $\leq$ Taille $<$ 1.7
- Classe 3 : 1.7 $\leq$ Taille $<$ 1.9
- Classe 4 : 1.9 $\leq$ Taille $<$ 2.0
- Classe 5 : 2.0 $\leq$ Taille

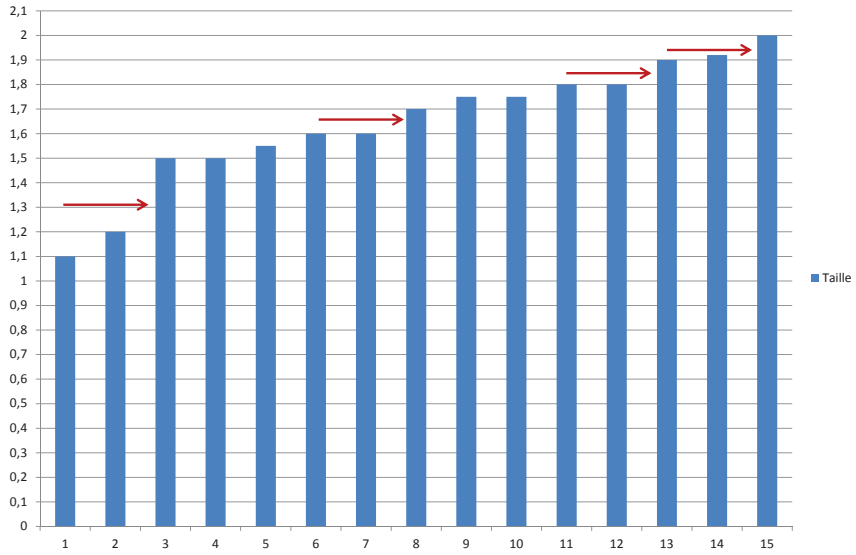


FIGURE 2.1 – Exemple de discrétisation selon les seuils observés.

2.2.1.3 La discrétisation en classes d'égale amplitude (EWD)

Cette méthode permet d'avoir des classes d'égale amplitude. L'amplitude représente la différence entre la plus grande valeur et la plus petite valeur d'un intervalle.

Présentation de l'algorithme

Pour discrétiser une variable continue en utilisant ***EWD***, il faut :

1. Calculer l'amplitude de chaque classe définit par :

$$A = \frac{max - min}{k} \tag{2.2}$$

avec min est la valeur minimale de la variable, max est la valeur maximale et k représente le nombre de classes.

2. Trier les données par ordre croissant.

3. En partant de la valeur minimale, il faut ajouter chaque fois la valeur de A pour retrouver les bornes de discrétisation.

Exemple illustratif : discrétisation de la taille d'une population

L'application de cette méthode sur les données du tableau 2.2, avec un nombre de classe $k = 5$, nous donne une amplitude $A = 0.18$ et génère les classes suivantes :

- Classe 1 : Taille $<$ 1.28
- Classe 2 : 1.28 $\leq$ Taille $<$ 1.46
- Classe 3 : 1.46 $\leq$ Taille $<$ 1.64
- Classe 4 : 1.64 $\leq$ Taille $<$ 1.82
- Classe 5 : 1.82 $\leq$ Taille

2.2.1.4 La discrétisation selon les quantiles (EFD)

Cette méthode permet d'avoir des classes d'effectifs égaux. Après l'application de cette méthode nous obtenons des classes avec le même nombre d'individus.

Présentation de l'algorithme

Pour discrétiser, en k classes ,une variable continue en utilisant la méthode ***EFD***, il faut :

1. Calculer le nombre d'individus dans chaque classe définit par :

$$A=\frac{n}{k} \tag{2.3}$$

avec n c'est le nombre des individus et k représente le nombre de classes voulues.

2. Trier les données par ordre croissant.

3. Regrouper les "nb" individus successifs dans les différentes classes.

Exemple illustratif : discrétisation de la taille d'une population

L'application de ***EFD*** sur les données du tableau 2.2, avec un nombre de classe $k = 5$, nous donne un nombre d'individus nb=3. Ce qui engendre et les classes suivantes :

- Classe 1 : Taille $<$ 1.52
- Classe 2 : 1.52 $\leq$ Taille $<$ 1.65
- Classe 3 : 1.65 $\leq$ Taille $<$ 1.77
- Classe 4 : 1.77 $\leq$ Taille $<$ 1.91
- Classe 5 : 1.91 $\leq$ Taille

2.2.1.5 La discrétisation selon la moyenne emboîtée

Pour cette méthode, le nombre de classes doit être une puissance de deux.

Présentation de l'algorithme

Pour discrétiser une variable continue en utilisant la discrétisation selon la moyenne emboîtée il faut :

1. Calculer la moyenne globale des valeurs. Cette moyenne représentera la première valeur de séparation.
2. Pour chaque sous-ensemble calculer la moyenne et prendre cette valeur comme seuil.
3. Répéter l'étape 2 jusqu'à l'obtention du nombre de classes voulu (2, 4, 8,16, etc.)

Exemple illustratif : discrétisation de la taille d'une population

L'application de cette méthode sur les données du tableau 2.2, avec un nombre de classe k=4, nous donne les classes suivantes :

- Classe 1 : Taille $<$ 1.44
- Classe 2 : 1.44 $\leq$ Taille $<$ 1.64
- Classe 3 : 1.64 $\leq$ Taille $<$ 1.83
- Classe 4 : 1.83 $\leq$ Taille

2.2.1.6 Outils implémentant des méthodes de discrétisation non supervisée

La plupart des méthodes, présentées auparavant, ont été implémentées dans des logiciels dédiés à des domaines spécifiques (comme la statistique ou la cartographie). Les outils les plus connus sont :

- Dans le domaine de la statistique [60] : de nombreux logiciels sont disponibles sur le marché. Parmi ces logiciels, nous trouvons SAS, Rstat, StatGraphics, Statistica, StatLab, StatItcf, Bmdp, Minitab, Spss, etc.

- Dans le domaine de la cartographie [71] : parmi les logiciels qui implémentent les méthodes de discrétisation, nous pouvons citer : WINCARTO, Migratio, GéoClip, Gapminder, Statplanet, etc.

2.2.2 Méthodes de discrétisation supervisée

Les méthodes de discrétisation supervisée ont été mises en œuvre afin de considérer, dans l'étape de discrétisation, les valeurs des classes associées aux différentes valeurs des variables continues. Elles ont été inventées principalement pour être utiliser dans les algorithmes de *data mining* qui n'acceptaient que les variables qualitatives.

Dans la littérature, il existe deux types de méthodes supervisées : les méthodes ascendantes et celles descendantes [116, 16]. Les méthodes descendantes partent de toutes les valeurs à discrétiser et cherchent le meilleur découpage en optimisant un critère donné à l'avance. Les méthodes de discrétisation ascendantes partent des intervalles élémentaires et cherchent d'une façon itérative la meilleure fusion de deux intervalles adjacents en optimisant un critère fixé à l'avance. La différence entre ces méthodes sont le critère qui détermine le meilleur point de coupure (ou de fusion), la condition d'arrêt du découpage (ou de la fusion) et l'ensemble initial des bornes potentielles.

2.2.2.1 Critères de découpage

Dans la littérature il existe trois types de critères de découpage [16] :

- Ceux qui favorisent le fait d'avoir des sous-intervalles homogènes par rapport à la variable à discrétiser comme la mesure de l'entropie.

- Ceux qui favorisent le fait d'avoir un nombre de valeurs suffisant dans chaque sous intervalle pour assurer une généralisation efficace comme le test de Khi2.

- Ceux qui sont sensibles aux effectifs et à la distribution de la variable à prédire comme l'indice d'impureté de Gini [53] et la mesure d'incertitude de Fusinter [115].

2.2.2.2 Conditions d'arrêt d'un algorithme de discrétisation

La condition d'arrêt d'un algorithme de discrétisation peut être

- fixée par l'utilisateur : comme le nombre d'intervalle ou le nombre minimal d'individus dans les intervalles.

- engendrée par le critère : comme l'absence de l'évolution du critère de découpage (ou de fusion).

2.2.2.3 L'ensemble initial des bornes

La plupart des méthodes existantes partent d'un sous ensemble constitué par les milieux des valeurs triées de la variable à discrétiser. Cependant nous trouvons quelques exceptions comme les méthodes *Chi-Merge* [64] et *Chi-Split* [7] qui utilisent les valeurs elles même comme ensemble de bornes initiales.

Étant donnée une variable continue triée $y = \{y_1, \ldots, y_n\}$ tel que $y_1 < y_2 < \ldots < y_n$. L'ensemble des valeurs de coupures initial noté $D = \{d_1, \ldots, d_{n-1}\}$ tel que $d_i = \frac{y_i + y_{i+1}}{2}$

Dans la littérature, plusieurs méthodes de discrétisation supervisées utilisent un sous ensemble de D, noté D^* représentant les bornes frontières. Il a été démontré par Fayyad et Irani [50] que les points de discrétisation générés par les méthodes supervisées sont toujours des éléments de D^*.

Définition de *D l'ensemble des bornes frontières**

Pour une variable continue triée $y = \{y_1, \ldots, y_n\}$ associée à une variable nominale $c = \{c_1, \ldots, c_n\}$, nous appelons ensemble de bornes frontières, noté D^*, l'ensemble des milieux de chaque deux valeurs frontières successives. Ces dernières représentent des valeurs successives de y ayant des classes différentes. La figure 2.2 illustre un exemple où nous avons trois bornes possibles.

NB : Si deux individus de classes différentes, sont décrits par la même valeur continue, alors cette valeur est considérée comme valeur frontière (voir par exemple les deux colonnes surlignées du tableau 2.3).

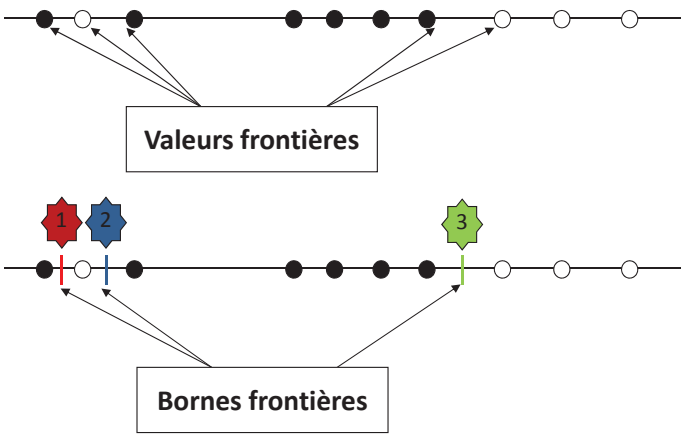


FIGURE 2.2 – Exemple de valeurs et de bornes frontières.

Exemple illustratif : Soit un ensemble de personnes décrites par deux variables : la taille et le genre (voir tableau 2.3). A partir de ce tableau l'ensemble des valeurs frontières de la variable taille est $taille^* = \{1.2, 1.5, 1.55, 1.6, 1.7, 1.75, 1.8, 1.9\}$ ce qui nous donne un ensemble de bornes frontières $D^* = \{1.35, 1.525, 1.575, 1.65, 1.725, 1.775, 1.85\}$.

Taille	1.1	1.2	1.5	1.5	1.55	1.6	1.6	1.7	1.75	1.75	1.8	1.8	1.9
Genre	F	F	F	M	F	F	M	M	M	M	F	M	M

Tableau 2.3 – Exemple d'un ensemble d'individus décrits par leurs tailles et leur genres.

2.2.2.4 Méthodes de discrétisation descendantes

Ces méthodes partent de l'intervalle entier composé par toutes les valeurs de la variable à discrétiser et cherchent les sous intervalles qui vont optimiser un critère de découpage fixé à l'avance. Ces méthodes se basent sur une recherche récursive des meilleurs sous-intervalles.

Leurs algorithmes se basent généralement sur trois étapes : (i) au début de l'algorithme elles partent de l'intervalle entier des valeurs, (ii) elles cherchent la meilleure valeur de découpage pour le subdivisé en deux sous-intervalles, (iii) le processus de recherche de la meilleure valeur de découpage se répète dans les sous-intervalles obtenus jusqu'à ce qu'une condition d'arrêt soit atteinte. La figure 2.3 illustre

un exemple d'enchaînement suivit par ce type de méthodes.

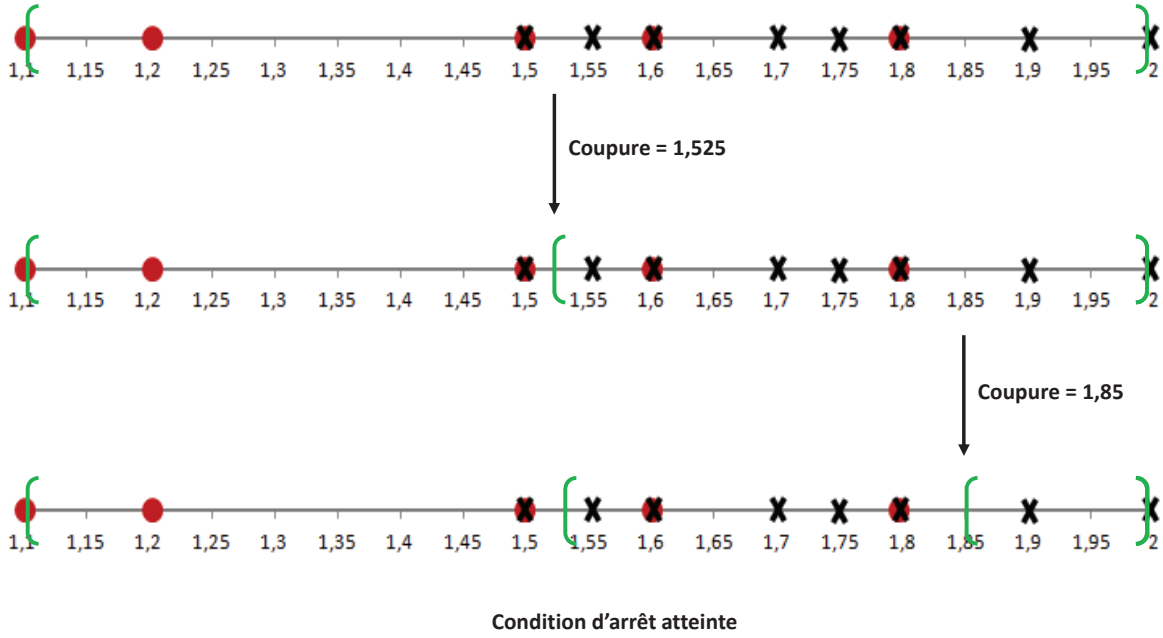


FIGURE 2.3 – Exemple d'exécution d'une méthode de discrétisation descendante.

Dans la littérature, plusieurs méthodes de discrétisation descendantes ont été implémentées et testées, parmi lesquelles nous pouvons citer :

- La méthode ***Chi-Split*** [7] : cette méthode utilise le critère de **Khi2**, appliqué à deux sous intervalles adjacents, comme critère de découpage. La condition d'arrêt est un seuil d'indépendance maximum lue dans la table du Khi2 au niveau de confiance fixé par l'utilisateur. Cette méthode utilise toutes les valeurs initiales de la variable à discrétiser comme ensemble initial de bornes possibles.
- La méthode ***D2*** [25] : elle utilise le gain informationnel basé sur l'entropie de Shannon comme critère de découpage. Il y a quatre conditions d'arrêt possibles : (i) le nombre d'exemples dans un intervalle (un effectif minimal de 14), (ii) le nombre de points de discrétisation (avec un maximum de 7), (iii) une différence négligeable entre les différents points potentiels de discrétisation, (iv) tous les

exemples d'un même sous-intervalle ont la même classe. Cette méthode utilise l'ensemble des milieux des valeurs triées de la variable à discrétiser comme ensemble initial de bornes.

- La méthode ***Multi Interval Algorithm "MIA"*** [50] : elle utilise le gain informationnel basé sur l'entropie de Shannon comme critère de découpage. Ce dernier est associé à un critère de validation basé sur le *MDLPC (Minimum Description Lenght Principle Cut)* [49]. Il y a deux conditions d'arrêt possibles : (i) l'absence de possibilités de découpage dans les sous intervalles (ii) aucune borne ne vérifie le critère du MLDPC. Cette méthode utilise l'ensemble des bornes frontières comme ensemble initial de bornes.

- La méthode ***Contrast*** [104] : cette méthode utilise la notion de "contraste" entre les valeurs de la variable à discrétiser qui est rajoutée à celle de l'entropie comme critère de découpage. Cette notion a été inspirée des méthodes de classification qui mettent deux valeurs continues proches dans le même groupe. Cette méthode utilise l'ensemble des bornes frontières comme ensemble initial de bornes.

- La méthode ***Class-attribute interdependence maximization "CAIM"*** [68] : cette méthode maximise l'interdépendance mutuelle entre la classe et la variable à discrétiser. Il y a deux conditions d'arrêt : (i) aucune amélioration dans le critère, (ii) le nombre d'intervalles est égal au nombre des valeurs de la variable classe. La deuxième condition d'arrêt permet d'avoir un minimum de valeurs de découpage sans altérer la qualité du résultat obtenu [68]. Cette méthode utilise l'ensemble des bornes frontières comme ensemble initial de bornes.

2.2.2.5 Méthodes de discrétisation ascendante

Les méthodes de discrétisation ascendantes partent des intervalles élémentaires et cherchent récursivement les meilleurs fusions entres deux intervalles adjacents en optimisant un critère donné. Dans la littérature nous trouvons deux types de méthodes : les méthodes "gloutonnes" et les méthodes optimales.

Les méthodes dites "gloutonnes" [116] utilisent des heuristiques pour trouver les meilleures fusions. Ces méthodes se basent sur une recherche récursive des meilleures fusions des sous-intervalles. Leurs algorithmes se basent sur quatre étapes : (i) le trie des valeurs de la variable à discrétiser et l'initialisation des intervalles élémentaires, (ii) l'évaluation de la partition la plus fine en calculant le critère associé à ses intervalles (iii) la recherche de la meilleur fusion de deux intervalles adjacents, (iv) répéter la troisième étape tant que la condition d'arrêt n'est pas atteinte. La figure 2.4 illustre un exemple d'exécution d'une méthode de discrétisation ascendante où les intervalles élémentaires initiaux sont délimités par les valeurs des bornes frontières.

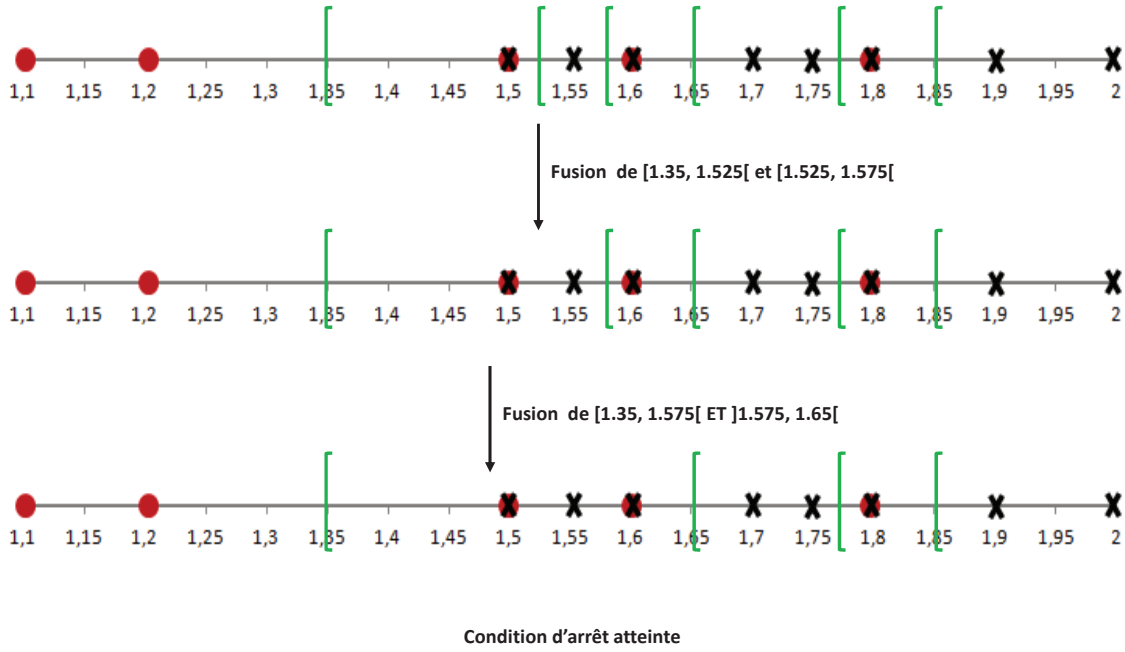


FIGURE 2.4 – Exemple d'exécution d'une méthode de discrétisation ascendante.

Parmi les méthodes de discrétisation ascendante nous citons :

- La méthode de ***ChiMerge*** [64] : elle utilise le critère de Khi2 comme critère de fusion de deux intervalles adjacents. Ce critère vérifie que les fréquences des classes dans un intervalle sont indépendantes de celles de l'intervalle adjacent. Dans ce cas ces deux intervalles peuvent être fusionnés. Le processus de fusion s'arrête dès qu'un seuil d'indépendance maximum lue dans la table du Khi2 au niveau de confiance fixé par l'utilisateur est atteint. Dans cette méthode les intervalles élémentaires sont délimités par les différentes valeurs de la variable à discrétiser.
- La méthode ***StatDisc*** [93] : elle construit hiérarchiquement les intervalles en fusionnant à chaque étape deux ou plusieurs intervalles adjacents. Le nombre maximal d'intervalles qui peuvent être fusionnés durant une étape est fixé par l'utilisateur. Cette méthode utilise une mesure de la corrélation entre deux variables nominales représentée par le coefficient Φ(Phi) pour déterminer à chaque étape de la construction hiérarchique les intervalles qui seront fusionnés.

Ces intervalles sont ceux qui ont une valeur minimal de Φ. La fusion continue jusqu'à atteindre un certain seuil de Φ. L'algorithme explore alors la discrétisation hiérarchique qui a été construite pour retourner automatiquement les meilleures bornes. Les intervalles élémentaires initiaux sont délimités par les valeurs des bornes frontières.

- La méthode ***Fusinter*** [115] : cette méthode cherche à maximiser un critère de fusion basé sur une mesure d'incertitude sensible aux effectifs. A chaque itération le gain d'incertitude est calculé pour chaque paire d'intervalles adjacents et la fusion est faite de la paire qui possède un gain positif maximal. La condition d'arrêt du processus de fusion est l'absence de fusion possible (tous les gains sont négatifs). Les intervalles élémentaires initiaux sont délimités par les valeurs des bornes frontières.

- La méthode ***Kiops*** [16] : cette méthode utilise le critère Khi2 appliqué à la distribution de l'ensemble des intervalles. Elle optimise un critère d'évaluation global de la partition du domaine en intervalles et non un critère local appliqué à deux intervalles adjacents comme Chi-Merge. Le processus de fusion s'arrête automatiquement si la probabilité d'indépendance ne décroit plus. Les intervalles élémentaires initiales sont délimités par les milieux entre deux valeurs successives.

Dans la littérature nous trouvons une méthode de discrétisation optimale [116]. Cette méthode retourne une partition optimale de la variable à discrétiser en prenant en compte les valeurs de la variable classe. Pour obtenir cette partition optimale, cette méthode implémente une extension de l'algorithme de Fisher [51] aux séquences. Ces derniers représentent la partition des individus d'une population suivant les bornes frontières (c'est-à-dire que deux individus sont dans la même séquence si leurs valeurs continues se trouvent entre deux bornes frontières successives). Cette méthode utilise le critère de Fusinter [115].

2.2.2.6 Algorithmes d'apprentissage et méthodes de discrétisation supervisée

Toutes les méthodes de discrétisation supervisée que nous avons présentées, peuvent être utilisées pour le pré-traitement des variables continues afin de permettre leurs utilisations dans les algorithmes d'apprentissage qui n'acceptent que les variables nominales. Cependant, plusieurs méthodes d'apprentissage ont intégré une méthode de discrétisation pour remédier à ce défaut. Parmi ces algorithmes nous trouvons :

- La méthode d'arbres de décision **C4.5** [89] : elle utilise le gain informationnel basé sur l'entropie de Shannon ;

- La méthode méthode d'arbres de décision **CART** [20] : elle utilise l'indice de GINI ;
- La méthode d'arbres de décision **CHAID** [63] : elle intègre la méthode ChiMerge qui utilise le test Khi2 ;
- La méthode **SIPINA** [117] : elle utilise le critère Fusinter.

Toutes les méthodes de discrétisation présentées sont des méthodes classiques. Leur objectif est de transformer une variable quantitative en une variable qualitative. Elles peuvent être utilisées pour transformer une variable quantitative en histogrammes en utilisant les deux étapes suivantes :
variable continue $\xrightarrow{discretisation}$ variable nominale $\xrightarrow{agrégation}$ variable histogramme.
L'inconvénient de cette méthode est la qualité des histogrammes obtenus : on ne peut pas garantir que ces histogrammes sont les plus discriminants pour les classes d'individus. Étant donné qu'aucune méthode n'est dédiée à l'ADS et n'a pour but de trouver les histogrammes les plus discriminants pour les classes, nous avons mis en place une méthode nommée "HistSyr". Son objectif est de transformer une variable continue en histogrammes les plus discriminants pour les classes d'individus.

3 HistSyr : conversion d'une variable continue en histogrammes les plus discriminants pour les classes d'individus

Afin de remédier à l'absence de méthodes permettant de passer d'un ensemble d'individus décrits par une variable continue à un ensemble de classes d'individus décrits par une variable histogramme, nous avons créé "HistSyr". Pour présenter cette méthode, nous commençons d'abord par la formulation de la problématique. Ensuite, nous présentons deux solutions que nous avons testées. Enfin, nous présentons de deux cas d'utilisation possibles de notre algorithme. Pour chaque cas nous donnons un exemple d'exécution et un exemple d'application sur des données réelles.

3.1 Présentation de la problématique

Étant donnés

- n individus décrits par deux variables y et c. Où y est une variable continue et c est une variable nominale. Les différentes valeurs de c représentent les classes d'individus. Nous notons nc le nombre de classes issues de la variable c.
- Une valeur entière k qui représente le nombre de modalités des histogrammes résultats.

Nous souhaitons discrétiser la variable y en un ensemble de k intervalles (modalités) dans le but de décrire chaque classe d'individus par une valeur histogramme, donnant sa fréquence dans chaque intervalle. Nous présentons, par la suite, le critère à optimiser pour cette discrétisation. Il a pour objectif la discrimination entre les descriptions des classes.

Exemple illustratif

Soit un ensemble de 9 joueurs décrits par leurs ages et leurs équipes représenté par le tableau 2.4. Dans cet exemple $y = Age$ et $c = Équipe$. Nous avons donc 3 ($nc = 3$) classes qui sont : PSG, LYON et OM.
Un partitionnement, noté I, de y en 3 intervalles est par exemple le suivant : $I = \{]-\infty, 23.5[, [23.5, 29[, [29, +\infty[\}$. En utilisant ces intervalles, nous obtenons la description des 3 classes par 3 histogrammes différents représentés par le tableau 2.5. Dans ce tableau, pour la variable "Age" chaque intervalle est associé à une modalité et il y a donc 3 modalités. Les modalités sont représentées par les rangs de leurs intervalles. Ainsi l'intervalle $]-\infty, 23.5[$ est associé à la modalité 1 et l'intervalle $[23.5, 29[$ est associé à la modalité 2.
Dans le tableau 2.5, on voit que le PSG est associé à un histogramme concentré sur la modalité 2 qui est donc de fréquence 1 et que pour l'OM la modalité 2 est de fréquence nulle.
Notre objectif est d'automatiser la recherche du partitionnement de y en k intervalles qui donnera les histogrammes les plus discriminants pour les différentes classes. De façon qu'en introduisant, par exemple, les données décrites par le tableau 2.4 et en précisant une valeur de $k = 3$ nous obtenons automatiquement les données symboliques décrites par le tableau 2.5.

Afin de mieux comprendre notre problématique, nous donnons, dans ce qui suit, les définitions d'une variable histogramme et de la notion de discrimination entre classes d'individus.

3.1.1 Une variable histogramme ?

Une variable histogramme Y est une variable modale, définie par un ensemble de k modalités. Chaque valeur Y_i d'une variable histogramme pour une classe i est représentée par la pondération de ces modalités par des fréquences $\{f_{ij}\}_{j=1,\ldots,k}$ tel que $\sum_{j=1}^{k} f_{ij} = 1$.

$$\forall i \in [1, nc], \quad Y_i = \{mod_1(f_{i1}), mod_2(f_{i2}), \ldots, mod_k(f_{ik})\};$$

avec nc : le nombre de classes d'individus(nous avons un histogramme par classe).
mod_j : représente la $j^{ème}$ modalité de l'histogramme.

Joueur	Age	Équipe
DiMaria	28	PSG
Matuidi	28	PSG
Verratti	24	PSG
Lacazette	25	LYON
Fekir	23	LYON
Tolisso	22	LYON
Diarra	31	OM
Diabi	30	OM
Rekik	21	OM

Tableau 2.4 – Exemple d'un ensemble de joueurs décrits par deux variables : l'age et l'équipe.

Équipe	Age
PSG	2(1)
LYON	1(0.667)2(0.333)
OM	1(0.333)3(0.667)

Tableau 2.5 – Tableau symbolique décrivant les "Équipe" en utilisant le résultat de la conversion de la variable Age en histogrammes les plus discriminants.

f_{ij} : représente la fréquence de la $j^{ème}$ modalité de l'histogramme décrivant la classe i.

Afin d'alléger la représentation des valeurs d'une variable histogramme, nous utilisons la représentation de **Syr** qui attribue à chaque modalité un rang. Ce rang est spécifié dans l'entête du fichier symbolique (voir par exemple la ligne 3 du tableau 2.6). Chaque valeur d'une variable histogramme sera la pondération des rangs des différentes modalités par leurs fréquences (voir par exemple la ligne 4 du tableau 2.6). Les modalités d'un histogramme peuvent être :

- Des catégories (des valeurs nominales), dans ce cas la variable histogramme est le résultat de l'agrégation d'une variable qualitative. Par exemple, la variable "Workclass" du tableau 2.6 est le résultat de l'agrégation d'une variable qualitative en histogrammes de trois modalités.

- Des intervalles, dans ce cas la variable histogramme est le résultat de l'agrégation d'une variable quantitative. Par exemple la variable "Age" du tableau 2.6 est le résultat de l'agrégation d'une variable continue en histogrammes de trois

modalités.

State	Age	Workclass
Concept	Histogram	Histogram
-	1 :<30, 2 :30to40, 3 :>40	1 :Private, 2 :Self_emp_not_inc, 3 :State_gov
moins_50K	1(0.25)2(0.5)3(0.25)	1(0.75)2(0.17)3(0.08)
plus_50K	2(0.5)3(0.5)	1(0.7)2(0.17)3(0.13)

Tableau 2.6 – Exemple de deux variables histogramme ”Age” et ”workclass” décrivant les différents statuts de travailleurs ”State”.

3.1.2 Discrimination entre les descriptions des classes d'individus

Dans ce contexte la discrimination signifie la différentiation entre les différentes classes d'individus. En effet, plus une variable est discriminante plus ses valeurs diffèrent d'une classe à une autre.

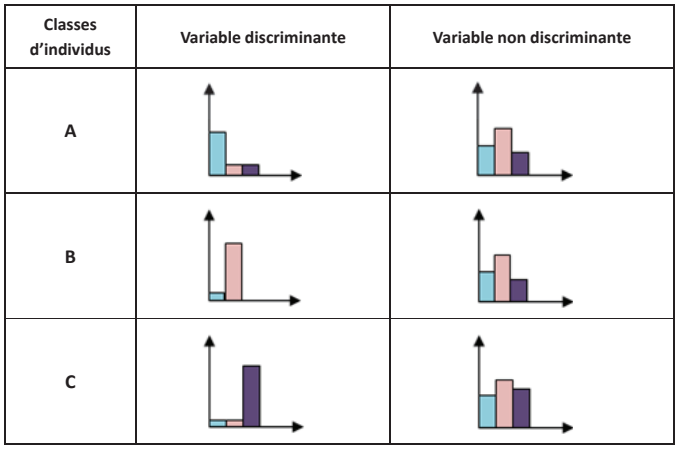


FIGURE 2.5 – Variable discriminant Vs variable non discriminante.

La figure 2.5 donne un exemple de deux variables histogrammes, la première étant discriminante alors que la deuxième ne l'est pas. A partir de cette figure nous remarquons qu'en utilisant la première variable nous pouvons différencier entre les trois concepts ”A”, ”B” et ”C”. En effet, pour cette variable la première modalité représente la classe ”A”. La deuxième représente la classe ”B” et la troisième représente la classe ”C”. Par contre en utilisant la deuxième variable, nous ne pouvons pas distinguer entre les trois classes A, B et C.

Cette notion de discrimination entre classes d'individus sera traduite mathématiquement et représentera le critère à optimiser de notre solution.

3.2 Présentation de la solution

Étant donné que nous prenons en compte les valeurs des classes associées aux valeurs continues, notre solution représentera une méthode supervisée. Pour cela il est nécessaire d'avoir un critère de découpage traduisant la notion de discrimination entre les différentes classes d'individus. A l'inverse des méthodes classiques de discrétisation nous allons évaluer un ensemble de bornes qui constituera les intervalles (les modalités de nos histogrammes) et non une seule borne à la fois.
Pour cela nous avons commencé par la mise en place du critère à optimiser. Ensuite, en s'inspirant des méthodes supervisées existantes, nous avons créé et testé deux algorithmes pour la création des histogrammes qui optimisent notre critère. Enfin, nous avons évalué la complexité de chaque solution et nous avons gardé la solution la plus rapide et la plus optimale.

3.2.1 Le critère d'HistSyr

Afin de comparer les histogrammes résultats des différents découpages, nous avons mis en place un critère de découpage. L'expression de ce critère permet d'exprimer la notion de discrimination entre classes d'individus. C'est-à-dire que plus les histogrammes sont différents plus la valeur du critère est importante.

Ce critère nommé **"score"** consiste à calculer les dis-similarités entre tous les histogrammes deux à deux. Ces dis-similarités sont définies par la somme des différences des fréquences de chaque modalité. Afin d'avoir des valeurs comprises entre 0 et 1 nous divisons cette somme par $nc * (nc - 1)$. L'expression du **"score"** d'HistSyr est représentée par l'équation 2.4[45].

$$Score = \frac{1}{nc(nc-1)} \sum_{i=1}^{nc-1} \sum_{j=1}^{k} \sum_{l=i+1}^{nc} \mid f(i,j) - f(l,j) \mid \tag{2.4}$$

avec nc : nombre de classes d'individus ;
k : nombre de modalités des histogrammes ;
$f(i,j)$: la fréquence de la modalité j de l'histogramme décrivant la classe i.
La figure 2.6 montre l'évolution de la valeur du critère d'HistSyr suivant trois cas possible :

- Si les histogrammes sont complètement différents alors le score sera égal à 1 (voir colonne 1 de la figure 2.6).

- Si histogrammes sont identiques alors le critère sera égale à (voir colonne 3 de la figure 2.6).
- Si les histogrammes sont ni identiques ni complètement différents, alors la valeur sera comprise entre 0 et 1 (voir colonne 2 de la figure 2.6).

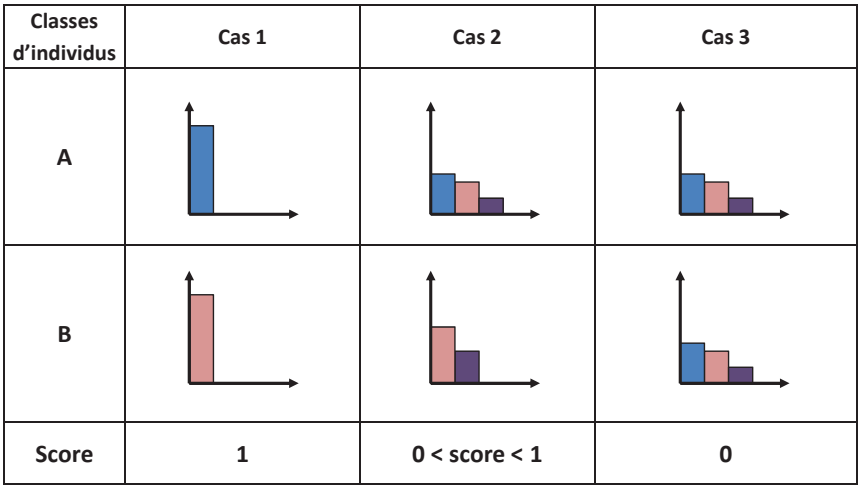


FIGURE 2.6 – Illustration de l'évolution de la valeur du critère d'HistSyr suivant trois cas possibles.

3.2.2 L'algorithme d'HistSyr

Afin de construire des histogrammes les plus discriminants pour les classes d'individus, nous avons créé et testé deux solutions.

3.2.2.1 Solution 1 : en évaluant toutes les combinaisons possibles

Dans cette première solution, l'idée initiale était d'évaluer toutes les combinaisons de $(k-1)$ bornes possibles (k étant le nombre de modalités des histogrammes). Le problème est que plus le nombre d'individus augmente plus le temps de calcul est important. En effet avec le test de toutes les combinaisons de $(k-1)$ bornes, l'opération de construction des histogrammes et de calcul du score se répète C_n^k fois. Afin de réduire le temps de calcul et d'avoir un nombre minimal d'individus dans chaque intervalle nous utilisons une liste initiale de bornes possibles représenté par le résultat d'un découpage en B classes d'effectifs égaux (B compris entre 10 et 30).

Cette première solution peut être résumée par l'algorithme 1. Cet algorithme est constitué de deux phases. Le but de la première phase est de retourner la liste initiale de bornes possibles. Les $(k-1)$ meilleurs bornes (qui maximisent le score) sont incluses dans à cette liste. La deuxième phase de notre algorithme concerne la génération et l'évaluation de toutes les combinaisons de $(k-1)$ valeurs parmi les $(B-1)$ obtenu dans la phase 1. L'évaluation de $(k-1)$ bornes revient à construire les histogrammes associés et de calculer le score d'HistSyr en utilisant ces histogrammes. A la fin l'algorithme retourne : les meilleurs $(k-1)$ bornes, les histogrammes associés à chaque classe d'individus et le score.

Interprétation de l'algorithme

Cette méthode a l'avantage de garantir d'avoir un découpage assez fin et un nombre minimal d'individus par modalité. Néanmoins, la solution retournée par l'algorithme 1 n'est pas optimale puisque la liste initiale des bornes possibles est obtenue en utilisant une méthode non supervisée (le découpage en classes d'effectifs égaux) qui ne prend pas en compte la valeur des classes d'individus.
Cette solution a une complexité très forte. En effet, le nombre de combinaisons possibles est ce l'ordre de C_{B-1}^{k-1} et pour chaque combinaison il faut construire l'ensemble des histogrammes et calculer le score associé aux histogrammes obtenus.

3.2.2.2 Solution 2 : en utilisant l'extension de l'algorithme de Fisher

En se basant sur les inconvénients de la première solution et sur les méthodes de discrétisation supervisée, nous avons mis en place un algorithme optimal beaucoup plus rapide que notre première solution.
Afin d'avoir une solution optimale, nous avons utilisé l'algorithme de Fisher [51] en optimisant le critère de HistSyr à la place de l'inertie intra-classe.
Pour réduire la complexité de l'algorithme nous nous sommes inspirés des méthodes de discrétisation supervisées en utilisant l'ensemble des bornes frontières comme liste initiale de bornes possibles. En effet, il a été démontré par Fayyad et Irani [50] que l'ensemble optimal des bornes est toujours inclus dans l'ensemble des bornes frontières.

En se basant sur les travaux de Lechevallier [72], pour utiliser l'algorithme de Fisher deux conditions doivent être vérifiées :

1. ***La propriété d'ordre*** sur l'ensemble $y = \{y_1, \ldots, y_n\}$: une partition en k intervalles possède la propriété d'ordre si pour chaque deux valeurs y_i et y_j se trouvant dans le même intervalle I_l toute valeur comprise entre y_i et y_j appartient au même intervalle, formellement :

$$\forall y_i, y_j; (y_i < y_j \; et \; y_i \in I_l et y_i \in I_l) \Longrightarrow (\forall y_h \in [y_i, y_j] \Rightarrow y_h \in I_l)$$

Algorithme 1 : Première version de la méthode HistSyr

Entrées :
Un fichier de données classiques contenant au moins une variable continue y et une variable nominale c.
Résultat :
Un fichier de données symboliques ".syr" avec les différentes valeurs de c comme classes décrites par l'agrégation de la variable continue y en histogrammes.
Données :
$y = \{y_1, \ldots, y_n\}$: une variable continue
$c = \{c_1, \ldots, c_n\}$: une variable nominale représentant les classes d'individus
k : le nombre de modalités des histogrammes résultats.
B : le nombre de classes à effectifs égaux $(10 \leq B \leq 30)$
Initialisations :
$L_{ini} \longleftarrow \emptyset$;
Phase 1 : Chercher les $(B-1)$ bornes de découpage de y en B classes d'effectifs égaux.
début
  $y1 \longleftarrow$ trier(y) ; *//trier par ordre croissant la variable y*
  $nb \longleftarrow n/B$; *//calcul du nombre d'individus dans chaque classe résultat de la classification à effectifs égaux*
  $i \longleftarrow 1$;
  tant que $(i+1) < y1.size()$ **faire**
    si *(i % nb = 0)* **alors**
      //l'ajout à la liste initiale de bornes le milieux de deux valeurs successives n'appartenant pas à la même classe.
      tant que *(y1.get(i) = y1.get(i+1))* **faire**
        $i \longleftarrow i+1$;
      fin
      L_{ini}.add($(y1.get(i) + y1.get(i+1))/2$);
    sinon
      $i \longleftarrow i+1$;
    fin
  fin
fin
Phase 2 : Chercher les meilleurs $(k-1)$ bornes parmi les $(B-1)$ valeurs résultat de la phase 1.
début
  $c1 \longleftarrow valeurs\_diff(c)$ *//recherche des différentes valeurs de c qui représenteront les classes symboliques*
  $comb \longleftarrow combinaison((k-1),(B-1))$; *// retourne toutes les combinaisons de $(k-1)$ parmi $(B-1)$*
  $max \longleftarrow 0$;
  $sc \longleftarrow 0$;
  $meil\_histo \longleftarrow \emptyset$;
  $meil\_borne \longleftarrow \emptyset$;
  pour *i = 0* **à** $comb.size()$ **faire**
    $histos \longleftarrow construction\_histo(comb.get(i), y, c, c1))$ *// calcul des histogrammes associés à la i^{eme} //combinaison*
    $sc \longleftarrow score(histo, c1.size())$; *// calcul du score des histogrammes associés à la i^{eme} combinaison*
    si *$(max < sc)$* **alors**
      $max \longleftarrow sc$;
      $meil\_histo \longleftarrow histos$;
      $meil\_borne \longleftarrow comb.get(i)$;
    fin
  fin
  retourner $max, meil\_histo, meil\_borne$;
fin

2. ***Propriété d'additivité du critère à optimiser*** : si la partition décrite par le découpage $(\{y_1, \ldots, y_i\}, I_2, \ldots, I_k)$ est optimale pour k intervalles, alors la partition décrite par le découpage $(I_2, \ldots, I_k)$ est optimale en $(k-1)$intervalles de $\{y_{i+1}, \ldots, y_n\}$.

Démonstration de l'applicabilité de l'algorithme de Fisher pour HistSyr

1. La propriété d'ordre Cette propriété est vérifiée et ne pose aucun problème puisque les valeurs de y sont inclus dans $\mathbb{R}$.

2. Propriété d'additivité du critère à optimiser : Montrons par l'absurde que si $P = \{[y_1, y_i], I_2, \ldots, I_k\}$ est le découpage optimale en k intervalles, alors la partition décrite par le découpage $\{I_2, \ldots, I_k\}$ est optimal en $(k-1)$ intervalles de $\{y_{i+1}, \ldots, y_n\}$.
Démonstration :
Si cette propriété n'est pas vérifiée, alors il existe un découpage $J = \{J_2, \ldots, J_k\}$ optimal en $(k-1)$ intervalles de $\{y_{i+1}, \ldots, y_n\}$ différent de $\{I_2, \ldots, I_k\}$. Ce découpage vérifie donc la propriété : $score(J) > score(I)$ **(ineq 1)**
La partition $Q = \{[y_1, y_i], J_2, \ldots, J_k\}$ aura comme score :

$$
\begin{aligned}
score(Q) &= \frac{1}{nc(nc-1)} \sum_{i=1}^{nc-1} \sum_{j=1}^{k} \sum_{l=i+1}^{nc} abs(mod(i,j) - mod(l,j)) \\
&= \frac{1}{nc(nc-1)} \left(\sum_{i=1}^{nc-1} \sum_{l=i+1}^{nc} abs(mod(i,1) - mod(l,1)) + \sum_{i=1}^{nc-1} \sum_{j=2}^{k} \sum_{l=i+1}^{nc} abs(mod(i,j) - mod(l,j)) \right. \\
&= score([y_1, y_i]) + score(J)
\end{aligned}
$$

De même $score(P) = score([y_1, y_i]) + score(I)$.
En reprenant l'inéquation 1 (ineq1) nous aurions alors $score(P) < score(Q)$ ce qui contredit l'hypothèse que P est la partition optimale en k intervalles.

Conclusion : La propriété d'additivité est vérifiée par le critère d'HistSyr.

Présentation de l'algorithme

Notre deuxième solution peut être résumées par les étapes suivante :

1. Trier les valeurs de la variable continue y par ordre croissant.
2. Extraire l'ensemble des bornes frontières D^*.
3. Appliquer l'algorithme de Fisher [51] modifié en optimisant le critère d'HistSyr et en partant de l'ensemble des bornes frontières D^* pour chercher les (k-1) bornes.

Dans cet algorithme, nous cherchons à chaque itération les bornes qui produisent des histogrammes qui maximisent le score d'HitSyr.

4. Retourner les (k-1) bornes, un fichier ".syr" contenant la description symbolique du résultat, le fichier pattern associé et la valeur du critère.

Cette solution est décrite par l'algorithme 2.

Interprétation de l'algorithme

La solution retournée par cet algorithme est optimale. Cependant le temps d'exécution devient important dès que la taille des données initiales dépasse quelques milliers de lignes.
Nous avons utilisé l'algorithme 2 pour tous nos tests et applications d'HistSyr.

Exemple d'application d'HistSyr

Soit un ensemble de 10 individus $\{\omega_1, \ldots, \omega_{10}\}$, décrits par deux variables y et c, représenté par le tableau 2.7.

Notre objectif est d'appliquer notre algorithme HistSyr pour construire les histogrammes , de 3 modalités, les plus discriminants pour les deux classes C1 et C2.
Pour cela nous commençons par le tri des valeurs de y par ordre croissant ce qui nous donne :
En appliquant la phase 1 de l'algorithme 2, nous obtenons un ensemble de bornes frontières $D^* = \{1.25, 2.25, 5.25\}$
L'application de la phase 2 de l'algorithme 2 est résumée par le tableau 2.8. A partir de ce tableau nous concluons que les bornes $\{2.25, 5.25\}$ représentent le meilleur découpage pour obtenir les histogrammes (de 3 modalités) les plus discriminants.

3.3 Les cas d'utilisation d'HistSyr

La méthode HistSyr peut être utilisée pour répondre à deux types de problématiques :

- La conversion d'une variable continue en histogrammes discriminants pour les

classes d'individus. Cette opération est faite lors de l'extraction de données symboliques à partir de données classiques.

- La réduction du nombre de modalités d'une variable histogramme en sélectionnant les modalités les plus discriminantes pour les classes d'individus. Cette opération est faite lors de l'étude d'un fichier symbolique contenant des variables histogrammes avec un grand nombre de modalités.

Des exemples d'application de ces deux cas d'utilisations seront présentés dans la suite.

3.3.1 Utilisation d'HistSyr pour la conversion d'une variable continue en histogrammes

Ce cas d'utilisation représente le but initial pour lequel nous avons créé HistSyr à savoir la conversion d'une variable continue en histogrammes. Il est utilisé à chaque fois où nous avons des données classiques contenant des variables continues que nous voulons convertir en données symboliques décrites par des histogrammes.

3.3.1.1 Exemple d'exécution de la méthode

Soient les données, représentées par le tableau 2.9, décrivant des personnes en utilisant deux variables la taille et le genre. Notre but est d'extraire à partir de ces données classiques un fichier symbolique décrivant la classe d'individus "Genre" en convertissant la variable continue "taille" en histogrammes les plus discriminants de 3 modalités (k=3).

Afin de convertir la variable "Taille" en histogrammes les plus discriminant pour la classe "Genre", nous appliquons les différentes étapes de l'algorithme d'HistSyr (voir algorithme 2).

1. Les valeurs de la variable "tailles" sont déjà triées par ordre croissant. A partir du tableau 2.9, l'ensemble de bornes frontières $D^* = \{1.575, 1.65, 1.775\}$

2. A partir de D^* nous allons chercher les meilleurs 2 bornes qui maximisent notre critère. Trois cas sont à étudier :

   - $I_1 = \{]-\infty, 1.575[; [1.575, 1.65[; [1.65, +\infty[\}$
     Dans ce cas, nous obtenons un histogramme par classe décrivant la variable taille. Ces histogrammes sont représentés par le tableau ci-dessous :
     A partir de ces histogrammes, nous pouvons calculer la valeur de notre

critère :

$$Score(I_1) = \frac{1}{2 \times 1} \sum_{i=1}^{1} \sum_{j=1}^{3} \sum_{l=2}^{2} |mod(i,j) - mod(l,j)|$$
$$= \frac{|mod(1,1) - mod(2,1)| + |mod(1,2) - mod(2,2)| + |mod(1,3) - mod(2,3)|}{2}$$
$$= \frac{|0 - 0.33| + |0.2 - 0.33| + |0.8 - 0.34|}{2} = \frac{0.33 + 0.13 + 0.46}{2} = \textbf{\textit{0.46}}$$

- $I_2 = \{] - \infty, 1.575[; [1.575, 1.775[; [1.775, +\infty[\}$
  De la même façon que pour I_1 nous obtenons les histogrammes suivant :
  Avec un critère qui est égal à : $Score(I_2)$ = ***0.46***
- $I_3 = \{] - \infty, 1.65[; [1.65, 1.775[; [1.775, +\infty[\}$
  De la même façon que pour I_1 nous obtenons les histogrammes suivant :
  Avec un critère qui est égal à : $Score(I_3)$ =***0.6***

3. A partir des trois découpages possibles étudiés, les meilleurs bornes qui donnent les histogrammes les plus discriminants sont $\{1.65, 1.775\}$.

Le résultat obtenu a été vérifié en utilisant le module HistSyr. Pour cela nous avons introduit le fichier initial au module en spécifiant : la classe d'individus à étudier, la variable à convertir et le nombre de modalités souhaités. La figure 2.7 représente le résultat obtenu qui coïncide avec les calculs que nous avons effectués.

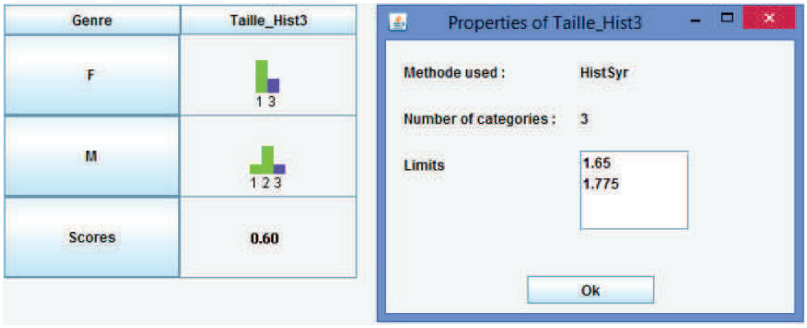


FIGURE 2.7 – Résultat de l'exécution de HistSyr sur les données des personnes décrits par leurs tailles et leurs genres en prenant le "Genre" comme classe d'individus.

3.3.1.2 Exemple d'application : les données des Iris de Fisher (données UCI)

Ces données représentent une description des iris, sous la forme d'un tableau contenant 150 individus. Chaque individu est décrit par 4 variables explicatives

continues et une variable nominale "Species". Cette dernière peut prendre 3 valeurs différentes (Setosa, Virginica et Versicolor). Le tableau 2.10 représente un extrait du fichier des Iris de Fisher.
Nous avons utilisé ces données pour tester le module HistSyr sur plusieurs variables et avec un nombre de modalités différent pour chaque variable. Le résultat est représenté par la figure 2.8.

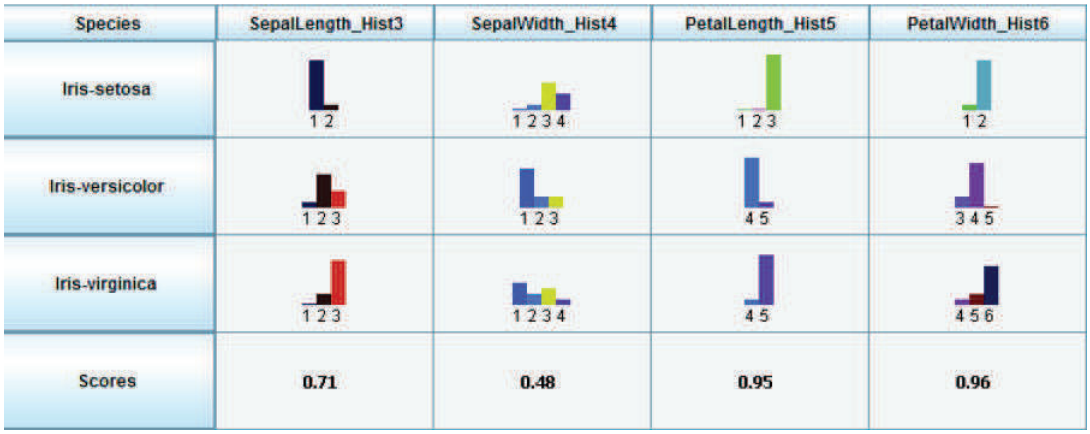


FIGURE 2.8 – Exemple d'utilisation d'HistSyr sur plusieurs variables avec un paramétrage différent pour chacune.

En plus de la création d'histogrammes, nous utilisons HistSyr pour comparer les variables et avoir une idée sur les variables les plus représentatives des différentes classes. Par exemple, pour les données des iris, nous avons utilisé HistSyr pour avoir une idée sur les variables les plus représentatives de différentes espèces d'iris. Pour cela, nous avons converti les quatre variables descriptives en histogramme de 3 modalité. Le résultat obtenu est représenté par la figure 2.9. A partir de cette figure nous constatons que les variables décrivant les pétales sont plus représentatives que celles décrivant les sépales. Nous remarquons aussi que pour les variables décrivant les sépales les histogrammes associés aux deux espèces "virginica" et "versicolor" se ressemblent et sont différents à ceux décrivant l'espèce "setosa". Nous concluons que lors de l'analyse de ces données il sera plus difficile de séparer les "virginica" et les "versicolor" que de séparer les "setosa" des autres espèces d'Iris.

3.3.2 Réduction du nombre de modalités des histogrammes

La réduction du nombre de modalités d'une variable histogramme est faite sur des données symboliques décrites par des histogrammes avec un nombre important de modalités. Une telle manipulation peut être utile dans plusieurs domaines de

Species	SepalLength_Hist3	SepalWidth_Hist3	PetalLength_Hist3	PetalWidth_Hist3
Iris-setosa				
Iris-versicolor				
Iris-virginica				
Scores	**0.71**	**0.48**	**0.95**	**0.96**

FIGURE 2.9 – Exemple d'utilisation d'HistSyr sur les données de Fisher avec 3 modalités.

data mining comme l'analyse de données textuelles *"text mining"*, l'analyse des séries temporelles, etc. L'objectif cette opération consiste à chercher parmi les m modalités d'une variable histogramme les k modalités les plus discriminantes pour chaque classe.

3.3.2.1 La méthode de réduction du nombre de modalités d'un histogramme en utilisant HistSyr

Pour cette opération nous nous sommes inspirés du principe de création d'histogrammes d'HistSyr. Sauf qu'au lieu d'évaluer la différence entre toutes les fréquences de toutes les modalités d'une variable histogramme, nous attribuons un score à chaque modalité pour chaque classe. Ensuite, nous sélectionnons les k modalités qui ont le plus grand score par classe. L'équation 2.5 donne l'expression du score d'une modalité j d'un histogramme décrivant la classe i.

$$Score = \frac{100 \times mod(i,j)}{nc-1} \sum_{l=1,\, l\neq i}^{nc} abs\Big(mod(i,j) - mod(l,j)\Big) \tag{2.5}$$

avec nc : le nombre de classes d'individus.

$mod(i,j)$: la fréquence de la modalité j de l'histogramme décrivant la classe d'individu i.

Après le calcul du score de chaque modalité et la sélection des k modalités les plus discriminantes, un nouveau fichier symbolique est créé. Dans ce cas l'utilisateur peut choisir entre :

- Supprimer toutes les autres modalités (celles classées de k+1 à m) et construire des histogrammes ne contenant que les modalités retenues. Les fréquences des modalités sélectionnées sont calculées en divisant les fréquences initiales par la somme des fréquences des modalités sélectionnées. Nous utilisons la fonction mod_1 pour calculer ces nouvelles fréquences.

$$mod_1(i,j) = \frac{mod(i,j)}{\sum_{k \in M_{sel}} mod(i,k)} \tag{2.6}$$

avec M_{sel} : représente l'ensemble des modalités sélectionnées.

- Ajouter une autre modalité nommée "other" regroupant toutes les modalités non retenues. Dans ce cas, les fréquences finales des modalités sélectionnées sont égales aux fréquences initiales. La fréquence de la modalité "other" est calculé en utilisant l'équation 2.7.

$$mod(i,other) = 1 - \sum_{k \in M_{sel}} mod(i,k) \tag{2.7}$$

- Supprimer toutes les modalités non retenues et construire des histogrammes avec des fréquences pondérées par la valeur du score de chaque modalité. Dans ce cas nous utilisons la formule de mod_2 pour calculer les nouvelles fréquences.

$$mod_2(i,j) = \frac{Score(j,i)}{\sum_{k \in M_{sel}} Score(k,i)} \tag{2.8}$$

3.3.2.2 Exemple d'exécution de la méthode de réduction du nombre de modalités d'une variable histogramme

Étant donné le fichier symbolique représenté par le tableau 2.11 décrivant trois classes d'individus (C1, C2 et C3) par une variable histogramme de 15 modalités. Notre but est d'appliquer "HistSyr" pour réduire à 3 le nombre de modalités de la variable histogramme.

Afin de réduire le nombre de modalités des histogrammes décrivant chaque classe à 3 par histogramme, nous appliquons de la méthode de réduction du nombre de modalité d'HistSyr. Cette méthode se compose de trois étapes :

1. Calcul des scores des différentes modalités des histogrammes décrivant chaque classe d'individus. Dans cet exemple, nous avons trois classes d'individus C1, C2 et C3.

- Pour la classe d'individus C1 :

$$score(A, C1) = \frac{100 \times mod(C1, A)}{3-1} \sum l = 1,\ l \neq i3 |mod(C1, A) - mod(l, A)|$$
$$= \frac{100 \times mod(C1, A) \times \Big(|mod(C1, A) - mod(C2, A)| + |mod(C1, A) - mod(C3, A)|\Big)}{2}$$
$$= \frac{100 \times 0.05 \times (|0.05 - 01| + |0.05 - 0.1|)}{2}$$
$$= 0.25$$

  De la même façon, nous avons calculé le score de chaque modalité pour la classe d'individus C1. Ce résultat est résumé dans le tableau ci-dessous.

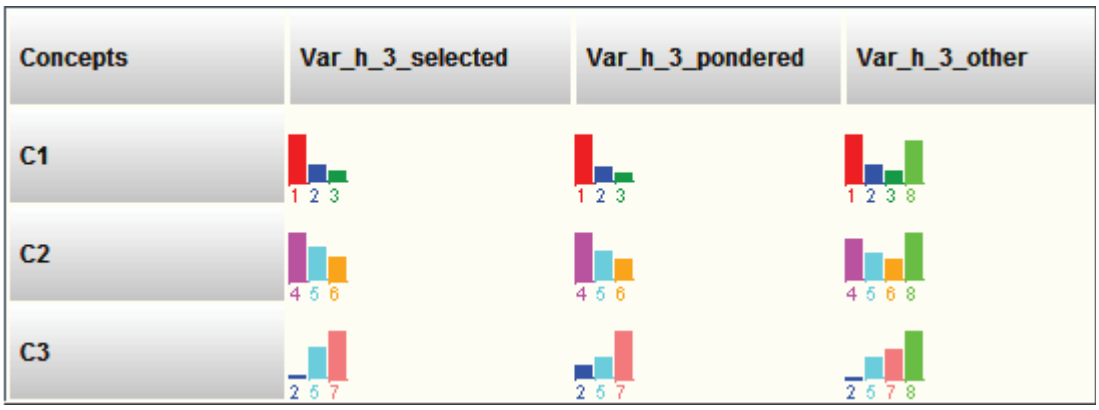
- Pour la classe d'individus C2 : après le calcul des scores des différentes modalités, de la même manière que pour C1, nous obtenons le tableau ci-dessous.
- Pour la classe d'individus C3 : après le calcul des scores des différentes modalités, nous obtenons le tableau ci-dessous.

2. La sélection des trois meilleures modalités par classe : à partir des trois tableaux représentant les scores pour les trois classes nous obtenons :
   - $M_{sel}(C1) = \{G, L, O\}$
   - $M_{sel}(C2) = \{B, E, N\}$
   - $M_{sel}(C3) = \{E, K, M\}$
3. La présentation du résultat suivant le choix de l'utilisateur :
   - Choix 1 : Utiliser les modalités retenues en recalculant leurs fréquences. Dans ce cas le résultat de la réduction du nombre des modalités de notre exemple de 15 à 3 modalité par classe est représenté par la première colonne du tableau 2.12.
   - Choix 2 : Utiliser des fréquences en se basant sur le score des modalités retenues. La deuxième colonne du tableau 2.12 illustre le résultat obtenu.
   - Choix 3 : l'ajout d'une modalité "Other" représentant toutes les modalités non sélectionnées. La troisième colonne du tableau 2.12 représente le résultat obtenu.

Le résultat obtenu peut être vérifié en utilisant l'outil HistSyr. Pour cela nous avons introduit le fichier symbolique de notre exemple à HistSyr en spécifiant le nombre de modalités souhaités ainsi que la méthode de calcul des fréquences. La figure 2.10 représente la description du fichier symbolique résultat de l'application d'HistSyr et sa représentation graphique obtenue en utilisant ***TabSyr*** (voir 4.2.1 du premier chapitre 1).

Concepts	Var_h_3_selected	Var_h_3_pondered	Var_h_3_other
Concept	Histogram	Histogram	Histogram
-	1:G,2:L,3:O,4:B,5:E,6:N,7:K	1:G,2:L,3:O,4:B,5:E,6:N,7:K	1:G,2:L,3:O,4:B,5:E,6:N,7:K,8:other
C1	1(0.62)2(0.23)3(0.15)	1(0.66)2(0.2)3(0.13)	1(0.4)2(0.15)3(0.1)8(0.35)
C2	4(0.46)5(0.31)6(0.23)	4(0.49)5(0.3)6(0.21)	4(0.3)5(0.2)6(0.15)8(0.35)
C3	2(0.04)5(0.38)7(0.58)	2(0.16)5(0.26)7(0.58)	2(0.02)5(0.2)7(0.3)8(0.48)

(a) Fichier symbolique résultat de la réduction du nombre de modalités à 3 par histogramme.



(b) Visualisation graphique du fichier résultat.

FIGURE 2.10 – Présentation du résultat de la réduction du nombre de modalités en utilisant ***HistSyr*** pour le calcul et ***TabSyr*** pour la visualisation.

3.3.2.3 Exemple d'application : données issues d'un corpus de documents issus d'appels téléphoniques [56]

Cet exemple est issue d'une étude réalisée à Syrokko[1]. Dans cette étude [56], les données initiales représentent un corpus de documents issus de la transcription de conversations téléphoniques du service client d'EDF. L'objectif de l'étude est de trouver les thématiques des conversations sans utiliser aucune analyse lexicale. HistSyr a été utilisé pour sélectionner les mots caractéristiques de chaque classe de mots. Dans cette étape, les données initiales sont représentées sous la forme d'un fichier symbolique (.syr) décrivant la classe "lema_clust_80". Ce dernier représente 80 classes de mots décrites par une variable histogramme "lema" ayant 2258 modalités (voir la figure 2.11).

Dans le but d'attribuer un thème à chaque classe de mots, nous avons utilisés "HistSyr" (avec un k =15) pour réduire le nombre de mots représentatifs de chaque

1. www.syrokko.com

classe de mots. La figure 2.12 représente le résultat obtenu après cette réduction. En utilisant ce fichier résultat et l'outil "NetSyr", nous avons pu identifier 20 classes de mots représentant des thèmes bien spécifiques [56].

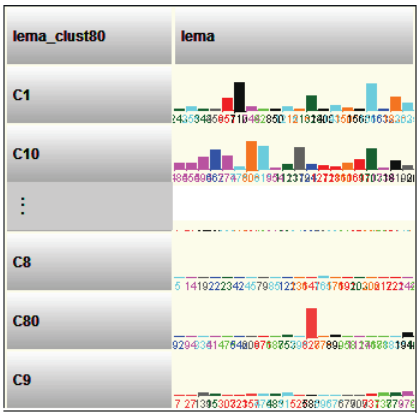


FIGURE 2.11 – Extrait du fichier initial issu de l'étude du corpus textuel.

FIGURE 2.12 – Extrait du fichier résultat de la réduction du nombre de modalités décrivant les classes de mots en utilisant HistSyr, issu de l'étude du corpus textuel.

3.4 HistSyr Vs autres méthodes de discrétisation

Afin d'évaluer la qualité des histogrammes résultats de la conversion de variables continues en utilisant HistSyr, nous les avons comparés aux histogrammes construits en utilisant des méthodes de discrétisation classiques. Ces histogrammes sont obtenus en suivant cet enchaînement "variable continue" $\rightarrow$ "variable nominale" $\rightarrow$ "variable histogramme".

Cette comparaison a été faite sur deux types de bases de données : le premier étant des données issues du répertoire UCI[13] ; alors que le deuxième représente des données de certains clients de Syrokko.

3.4.1 Tests sur des données du répertoire UCI

UCI (Blake et Merz, 1998) est un répertoire contenant des bases de données dédiées au test et à l'évaluation des différentes méthodes d'intelligence artificielle. Ses bases de données sont souvent utilisées par les scientifiques pour valider leurs méthodes de *data mining*.

Pour comparer notre méthode à d'autres méthodes classiques de discrétisation nous avons utilisé trois bases de données UCI :

- La base des "Iris de Fischer" : représente une description des fleurs d'iris. Cette base contient 150 individus. Chaque individu est décrit par 4 attributs continus et une variable classe qui peut prendre 3 valeurs différentes (Setosa, Virginica et Versicolor).
- La base "Australian" : concerne une étude sur les cartes de crédit, pour des raisons de confidentialité tous les noms et les valeurs des attributs ont été codifiés. Le nombre d'individus dans la base est égal à 690. Chaque individu est décrit par 14 attributs dont 6 continus et une variable binaire représentant la classe.
- La base "breast-cancer-wisconsin" : décrit les données recueillis sur le cancer du sein par une équipe de l'Université du Wisconsin. Elle contient 684 individus. Chaque individu est décrit par 9 attributs et une variable binaire représentant la classe.

Pour la comparaison nous avons utilisé deux méthodes de discrétisation non supervisées (en classes d'égale amplitude "EWD" et selon les quantiles "EFD") et une méthode de discrétisation supervisée (MIA [50]). Les bornes résultats la méthode MIA ont été obtenues en utilisant la boite à outil Weka [108]. Ces résultats ont été introduits à l'outil HistSyr qui implémente une méthode "prédifined limits" permettant de construire les histogrammes en introduisant les bornes de découpages. Les résultats des deux méthodes non supervisées ont été obtenu en utilisant les deux méthodes "equal areas" et "equal intervals" implémentées dans HistSyr.

Le tableau 2.13 résume tous les résultats obtenus. D'après ce tableau nous remarquons que les deux méthodes supervisées (HistSyr et MIA) sont meilleures que les deux méthodes non supervisées (EWD et EFD). Nous constatons aussi que les scores des histogrammes issus de HistSyr et de MIA sont presque égaux sauf qu'avec HistSyr nous avons moins de bornes qu'avec MIA.

Algorithme 2 : Deuxième version de la méthode HistSyr

Entrées :
Un fichier de données classiques contenant au moins une variable continue y et une variable nominale c.
Résultat :
Un fichier de données symboliques ".syr" avec les différentes valeurs de c comme classes décrites par l'agrégation de la variable continue y en histogrammes.
Données :
$y = \{y_1, \ldots, y_n\}$: une variable continue
$c = \{c_1, \ldots, c_n\}$: une variable nominale représentant les classes d'individus
k : le nombre de modalités des histogrammes résultats.
$bornes\_front$: la liste des bornes frontières.
Initialisations :
$bornes\_front \longleftarrow \emptyset$;
Phase 1 : Chercher les bornes frontières de y.
début
  $(y1, c1) \longleftarrow$ trier (y, c) ; *//trier par ordre croissant la variable y et les valeurs des classes //associées c*
  $val \longleftarrow y1.get(0)$;
  $classe \longleftarrow c1.get(0)$;
  pour *i = 1* **à** $comb.size()$ **faire**
    si $(classe \neq c1.get(i))$ **alors**
      $bornes\_front.add((val + y1.get(i))/2)$;
      $classe \longleftarrow c1.get(i)$;
    fin
    $val \longleftarrow y1.get(i)$;
  fin
fin
Phase 2 : Chercher les meilleurs $(k-1)$ bornes en utilisant l'algorithme de Fisher modifié
début
  $classes \longleftarrow valeurs\_diff(c1)$ *//recherche des différentes valeurs de c qui représenteront // les classes symboliques*
  / Utilisation de l'algorithme de Fisher modifié pour la recherche des meilleurs $(k-1)$ bornes dans la liste des bornes frontières.*/*
  $meilleur\_borne \longleftarrow fisher\_modifié(y1, c1, k, bornes\_front)$;
  $meil\_histo \longleftarrow construire_histo(y1, c1, classes, meilleur\_borne)$;
  $score\_max \longleftarrow score(meil\_histo)$;
  retourner $score\_max, meil\_histo, meilleur\_borne$;
fin

	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}
y	1	4.5	5	2.5	4	2	2	6	5.5	1.5
c	C1	C1	C1	C1	C1	C2	C2	C2	C2	C2

Tableau 2.7 – Exemple d'une population décrite par une variable continue y et une variable nominale c.

y	1	1.5	2	2	2.5	4	4.5	5	5.5	6
c	C1	C2	C2	C2	C1	C1	C1	C1	C2	C2

Bornes	Histogrammes		Score
$\{1.25, 2.25\}$	C1	1(0.2)3(0.8)	$= \frac{1}{2}(0.2 + 0.6 + 0.4)$
	C2	2(0.6)3(0.4)	$= 0.6$
$\{1.25, 5.25\}$	C1	1(0.2)2(0.8)	$= \frac{1}{2}(0.2 + 0.2 + 0.4)$
	C2	2(0.6)3(0.4)	$= 0.4$
$\{2.25, 5.25\}$	C1	1(0.2)2(0.8)	$= \frac{1}{2}(0.4 + 0.8 + 0.4)$
	C2	1(0.6)3(0.4)	$= 0.8$

Tableau 2.8 – Détails de l'exécution de la recherche des bornes qui retournent les histogrammes les plus discriminants.

Taille	Genre
1.55	F
1.6	F
1.6	M
1.7	M
1.75	M
1.75	M
1.8	F
1.8	M

Tableau 2.9 – Description d'un ensemble de personnes par leurs tailles et leurs genres.

Genre	Taille
M	2(0.2) 3(0.8)
F	1(0.33) 2(0.33) 3(0.34)

Genre	Taille
M	2(0.8) 3(0.2)
F	1(0.33) 2(0.33) 3(0.34)

Genre	Taille
M	1(0.2) 2(0.6) 3(0.2)
F	1(0.67) 3(0.33)

	Species	SepalLength	SepalWidth	PetalLength	PetalWidth
1	Setosa	5.1	3.5	1.4	0.2
2	Versicolor	7	3.2	4.7	1.4
3	Virginica	6.3	3.3	6	2.5
⋮	⋮	⋮	⋮	⋮	⋮
150	Virginica	5.8	2.7	5.1	1.9

Tableau 2.10 – Extrait des données décrivant les Iris de Fisher.

Concepts	Var_h
-	1 :A, 2 :B, 3 :C, 4 :D, 5 :E, 6 :F,7 :G, 8 :H, 9 :I, 10 :J, 11 :K, 12 :L, 13 :M, 14 :N,15 :O
C1	1(0.05)3(0.1)4(0.1)7(0.4)8(0.01)9(0.01)10(0.01)11(0.05)12(0.15)13(0.02)15(0.1)
C2	1(0.1)2(0.3)3(0.05)4(0.05)5(0.2)6(0.01)8(0.01)10(0.01)11(0.01)13(0.11)14(0.15)
C3	1(0.1)3(0.1)4(0.05)5(0.2)6(0.05)8(0.01)9(0.01)10(0.03)11(0.3)12(0.02)13(0.13)

Tableau 2.11 – Tableau de données symboliques décrivant trois classes d'individus par une variable histogramme de 15 modalités.

Modalités	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Scores	0.25	0	0.25	0.5	0	0	16	0	0.005	0.01	0.725	2.1	0.2	0	1

Modalités	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Scores	0.25	9	0.25	0.125	2	0.025	0	0	0	0.01	0.165	0	0.605	2,25	0

Modalités	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Scores	0.25	0	0.25	0.125	2	0.225	0	0	0.005	0.06	8.1	0.15	0.845	0	0

Concepts	Var_h_selected	Var_h_pondered	Var_h_other
-	1 :G,2 :L,3 :O,4 :B, 5 :E,6 :N,7 :K,8 :M	1 :G,2 :L,3 :O,4 :B, 5 :E,6 :N,7 :K,8 :M	1 :G,2 :L,3 :O,4 :B, 5 :E,6 :N,7 :K,8 :M,9 :Other
C1	1(0.62)2(0.23)4(0.15)	1(0.84)2(0.11)4(0.05)	1(0.4)2(0.15)4(0.1)9(0.35)
C2	4(0.46)5(0.31)6(0.23)	4(0.68)5(0.15)6(0.17)	4(0.3)5(0.2)6(0.15)9(0.35)
C3	5(0.32)7(0.48)8(0.20)	5(0.18)7(0.74)8(0.08)	5(0.2)7(0.3)8(0.13)9(0.37)

Tableau 2.12 – Résultats de la réduction du nombre des modalités d'une variable histogramme de 15 à 3 modalités par classe en utilisant HistSyr.

		HistSyr		EWD		EFD		MIA	
Base de données	Attribues	Bornes	Score	bornes	Score	Bornes	Score	Bornes	Score
Iris	Sepallength	5.45 ; 6.15	0.71	5.55 ; 6.7	0.66	5.4 ; 6.3	0.65	5.55 ; 6.15	0.70
	Sepalwidth	2.95 ; 3.05	0.48	2.8 ; 3.6	0.29	2.9 ; 3.2	0.44	2.95 ;3.35	0.47
	Petallength	2.45 ; 4.85	0.95	2.97 ; 4.93	0.94	2.45 ; 4.9	0.95	2.45 ; 4.75	0.95
	Petalwidth	0.8 ; 1.65	0.96	0.9 ; 1.7	0.96	0.8 ; 1.6	0.95	0.8 ; 1.75	0.96
Cancer	Clump	4.5	0.64	4 ; 7	0.60	3 ; 5	0.64	4.5 ; 6.5	0.64
	Uni_ cel_ size	2.5	0.86	3.25 ; 5.5 ; 7.75	0.82	1 ; 5	0.71	1.5 ; 2.5 ; 4.5	0.86
	Uni_ cel_ sha	2.5	0.85	3.25 ; 5.5 ; 7.75	0.82	1 ; 5	0.72	1.5 ; 2.5 ; 4.5	0.85
	Marg_ adh	1.5	0.69	4 ; 7	0.64	1 ; 3	0.68	1.5 ; 3.5	0.69
	Sing_ epit	2.5	0.81	4 ; 7	0.69	2 ; 3	0.81	2.5 ; 3.5	0.81
	Blan_ chrom	3.5	0.77	4 ; 7	0.77	2 ; 3	0.64	2.5 ; 3.5	0.77
	Norm_ nucl	2.5	0.75	4 ; 7	0.65	1 ; 2	0.71	2.5 ; 8.5	0.75
	Mitoze	1.5	0.42	5.5	0.13	1.5	0.42	1.5	0.42
Australian	A2	36.625	0.16	47	0.09	28.625	0.08	38.96	0.15
	A3	4.208	0.23	14	0.05	2.75	0.20	4.2075	0.23
	A7	1.02	0.39	14.25	0.04	1	0.36	1.02	0.39
	A10	0.5	0.46	22.33 ; 44.66	0.01	0.5 ; 9.5	0.46	0.5 ; 2.5	0.46
	A13	105	0.22	1000	0.01	160	0.15	99.5	0.21
	A14	232	0.35	50000	0.01	6	0.22	493	0.33

Tableau 2.13 – Résultats de l'application d'HistSyr, EFD, EFW and MIA aux différentes variables continue de 3 bases UCI.[45]

3.4.2 Tests sur des données issues des études de Syrokko

3.4.2.1 Étude de l'influence des conditions environnementales sur les mesures de corrosion

Cette étude entre dans le cadre d'un projet ANR APPLET "Durée de vie des ouvrages : Approche Prédictive PerformantielLE et probabilisTe " [17, 34]. Cette étude visait à apporter une réponse aux problèmes posés par la disparité des mesures de corrosion réalisées à différentes saisons. Les expériences ont été réalisées sur différents prismes de béton vieillis de façons différentes (T(Témoin), G(chlorures au gâchage), I (chlorures par immersion/séchage) et C (carbonatation)) afin de prendre en compte les divers modes de corrosion. Chaque prisme de béton est décrit par :

- L'agression : T, G, I, C
- La grandeur potentiel libre ou ***Ecorr*** qui donne une idée qualitative de la probabilité de corrosion.
- La grandeur résistance de l'enrobage, ***Re*** qui indique la chute ohmique du matériau béton armé. Plus cette valeur est importantes plus le matériau est résistant (béton plus compact, plus mouillé, non pollué par des espèces ioniques, etc...).
- La densité de courant de corrosion, ***Jcorr*** qui fourni de manière quantitative, la valeur de corrosion instantanée de l'armature dans des conditions données. Plus cette valeur est grande plus la corrosion est active c'est à dire plus l'armature se corrode.

Dans cette étude nous avons utilisés HistSyr pour transformer les trois variables continues (Ecorr, Jcorr et Re) en histogrammes en prenant l'agression comme classe d'individus. Ces histogrammes ont été par la suite comparés aux histogrammes construits à partir de seuils fixés par des experts du domaine.

La figure 2.13 représente les différents histogrammes obtenus, où Jcorr_Hist4, Re_Hist5 et Ecorr_Hist3 représentent les résultats de HistSyr et Jcorr_limit4, Re_limit5 et Ecorr_limit3 représente les histogrammes construits suivant les seuils des experts. Nous constatons que les histogrammes résultats de HistSyr sont plus discriminants que les autres. Ces histogrammes ont été utilisés dans la suite de cette étude pour décrire les agressions.

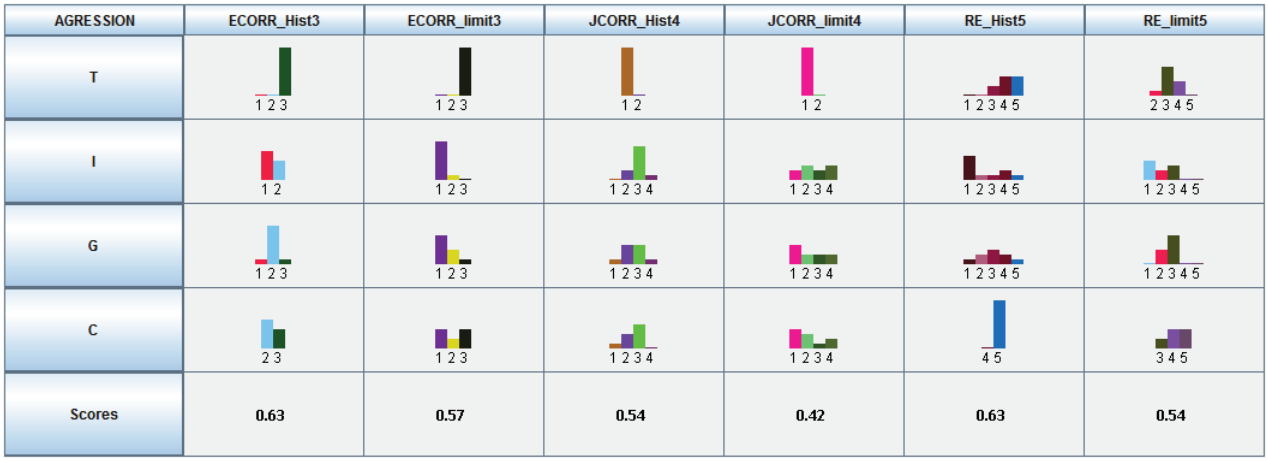


FIGURE 2.13 – Application de HistSyr sur les données de l'étude APPLET. Jcorr_Hist4, Re_Hist4 et Ecorr_Hist3 sont les résultats de l'application de la méthode HistSyr. Alors que Jcorr_limit4, Re_limit4 et Ecorr_limit3 sont issues du découpage proposé par des experts du domaine.

3.4.2.2 Étude des trajectoires de prise en charge des cancers dans la région bourgogne

Cette étude entre dans le cadre d'un contrat d'étude au titre de "Contrat Projet Etat Région (CPER) 2007-2013" du Conseil Régional de Bourgogne avec plusieurs organismes dont le CHU de Dijon. Il s'agit de l'étude des trajectoires de prise en charge des patients atteints d'un cancer [88]. Pour chaque type de cancer, 2 tables de données sont fournies :

- La table des patients où chaque ligne correspond à un patients et contient entre autres la séquence des établissements fréquentés.
- La table des séjours où chaque ligne correspond à un séjour d'un patient dans un établissement hospitalier.

Afin de répondre aux questions concernant l'identification de la relation entre les différents modes de sorties et les trajectoires des patients. Nous avons essayé de caractériser chaque mode de sortie "mod_out" par les variables explicatives existantes dans les deux fichiers de données. Puisque ces dernières contiennent des variables continues numériques (age_in et sej_len_tot, pour les fichiers des patients), nous avons utilisé HistSyr, EWD et EFD pour la construction des histogrammes en prenant la variable "mod_out" comme classe symbolique.

La figure 2.14 illustre le résultat obtenu pour les patients atteint d'un cancer

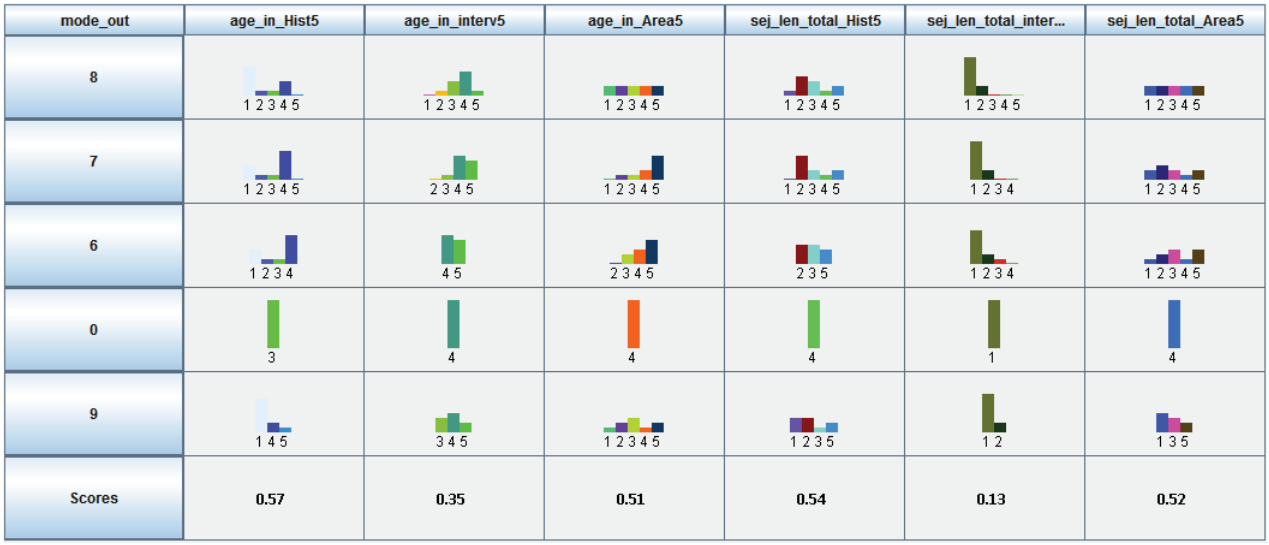


FIGURE 2.14 – Application d'HistSyr, d'EWD et d'EFD pour la conversion des variables "age_in" et "sej_len" en histogrammes dans le cadre de l'étude du mode de sortie des patients atteints d'un cancer colorectal.

colorectal. A partir de cette figure nous remarquons que les histogrammes résultats de l'application d'HistSyr sont plus discriminants que les autres histogrammes. Nous constatons aussi que l'âge du patient est la variable la plus discriminante pour le mode de sortie.

4 Conclusion

La transformation des données du classique au symbolique est une étape primordiale dans le processus de l'ADS. Cette étape se base sur le choix de la classe d'individus et sur l'agrégation des variables descriptives qualitatives et quantitatives en données symboliques. Généralement, une variable quantitative est automatiquement transformée en intervalles de valeurs. Ceci est dû au fait que l'agrégation de ce type de variable en histogrammes est une opération délicate. Afin de simplifier cette agrégation et de la rendre accessible et automatique, nous avons mis en place la méthode "HistSyr". Cette méthode a pour but de convertir une variable continue en histogrammes les plus discriminants pour les classes. Elle offre à l'utilisateur la possibilité de créer des histogrammes à partir de variables continues et d'apporter des modifications à des histogrammes existants. L'efficacité de cette méthode a été prouvée en comparant ces résultats aux histogrammes issus d'autres méthodes de discrétisation.

L'outil "HistSyr" implémentant cette méthode ainsi que d'autres méthodes de discrétisation existantes a été créé afin d'enrichir le logiciel Syr pour extraire des données symboliques conservant le maximum d'informations à partir de bases classiques.

Dans le chapitre suivant nous présentons une nouvelle méthode de construction d'arbres de décision symbolique qui entre dans le cadre de la deuxième étape d'ADS. Cette méthode a donné naissance à un nouveau module dans le logiciel Syr qui est "SyrTree".

CHAPITRE 3

Arbre de décision symbolique SyrTree

1 Introduction

Le travail présenté dans ce chapitre entre dans le cadre de la deuxième étape de l'ADS. Il a pour but d'étendre la méthode des arbres de décisions aux données symboliques. Dans la littérature, plusieurs travaux ont traités cette problématique [87, 77, 97]. La plupart de ces méthodes, à l'exception de Stree de Seck, n'acceptent pas tous types de variables symboliques. Notre objectif est de proposer un algorithme qui accepte tous types de données symboliques aussi bien pour les variables explicatives que pour la variable à expliquer.

Ce chapitre est organisé comme suit : dans la première partie, nous présentons un état de l'art des méthodes d'arbre de décision. Ensuite, nous présentons notre méthode d'arbre de décision nommée "SyrTree". Enfin, nous exposons les résultats de la comparaison entre SyrTree et d'autres méthodes d'arbre de décision.

2 Arbres de décision

Un arbre de décision est une méthode d'apprentissage supervisée qui consiste à trouver un partitionnement des individus d'une population. Ce partitionnement est représenté sous la forme d'une structure arborescente où toute la population est regroupée dans le nœud racine. L'objectif de cette méthode est de subdiviser les individus d'un nœud en deux ou plusieurs groupes homogènes par rapport à la variable à expliquer. Dans ce qui suit, nous présentons les notions de bases des arbres de décision et nous décrivons brièvement les méthodes existantes.

2.1 Définitions et notions de base

La méthode d'arbres de décision possède un vocabulaire propre. Afin de comprendre les différents termes, nous allons nous baser sur un exemple illustratif qui a été présenté dans le livre de Quinlan [89]. Dans cet exemple les données initiales représentent une table décrivant le comportement d'un ensemble d'individus par rapport au fait de jouer ou non suivant les conditions météorologiques.
Le tableau 3.1 représente la description de cette population par cinq variables : l'ensoleillement, la température, l'humidité, la présence ou pas de vent et le fait de jouer ou non.
La figure 3.1 illustre l'arbre de décision obtenu sur cet ensemble d'individus en utilisant la méthode C4.5 [89]. Cet arbre est construit à l'aide de questions binaires qui servent à répartir les individus d'un nœud dans les deux nœud fils (droite et gauche). Si pour un individu la réponse à la question binaire est "vrai" (respectivement faux) alors il appartiendra au nœud fil gauche (respectivement droit).

Ensoleillement	Température (°F)	Humidité(%)	Vent	Jouer
Soleil	75	70	Oui	Oui
Soleil	80	90	Oui	Non
Soleil	85	85	Non	Non
Soleil	72	95	Non	Non
Soleil	69	70	Non	Oui
Couvert	72	90	Oui	Oui
Couvert	83	78	Non	Oui
Couvert	64	65	Oui	Oui
Couvert	81	75	Non	Oui
Pluie	71	80	Oui	Non
Pluie	65	70	Oui	Non
Pluie	75	80	Non	Oui
Pluie	68	80	Non	Oui
Pluie	70	96	Non	Oui

Tableau 3.1 – Description de la base de données ”weather” [89].

2.1.1 Variables explicatives et la variable à expliquer

Un arbre de décision est construit à partir d'un ensemble d'individus décrits par des variables. Avant d'exécuter un quelconque algorithme de construction d'arbre de décision, il est nécessaire de définir :

- *La variable à expliquer :* appelée aussi variable ”classe” ou ”cible”. C'est la variable à prédire ou à expliquer. Généralement, le découpage d'un nœud de l'arbre a pour but d'avoir des nœuds les plus homogènes par rapport à cette variable. Dans notre exemple illustratif (figure 3.1), la variable à expliquer correspond à l'état de jouer ou non.

- *Les variables explicatives :* correspondent à l'ensemble des variables qui décrivent les individus. Elles sont utilisées pour la construction de l'arbre. A chaque étape de la construction d'un arbre de décision, elles entrent en concurrence et la variable la plus discriminante est sélectionnée et représentera la variable de

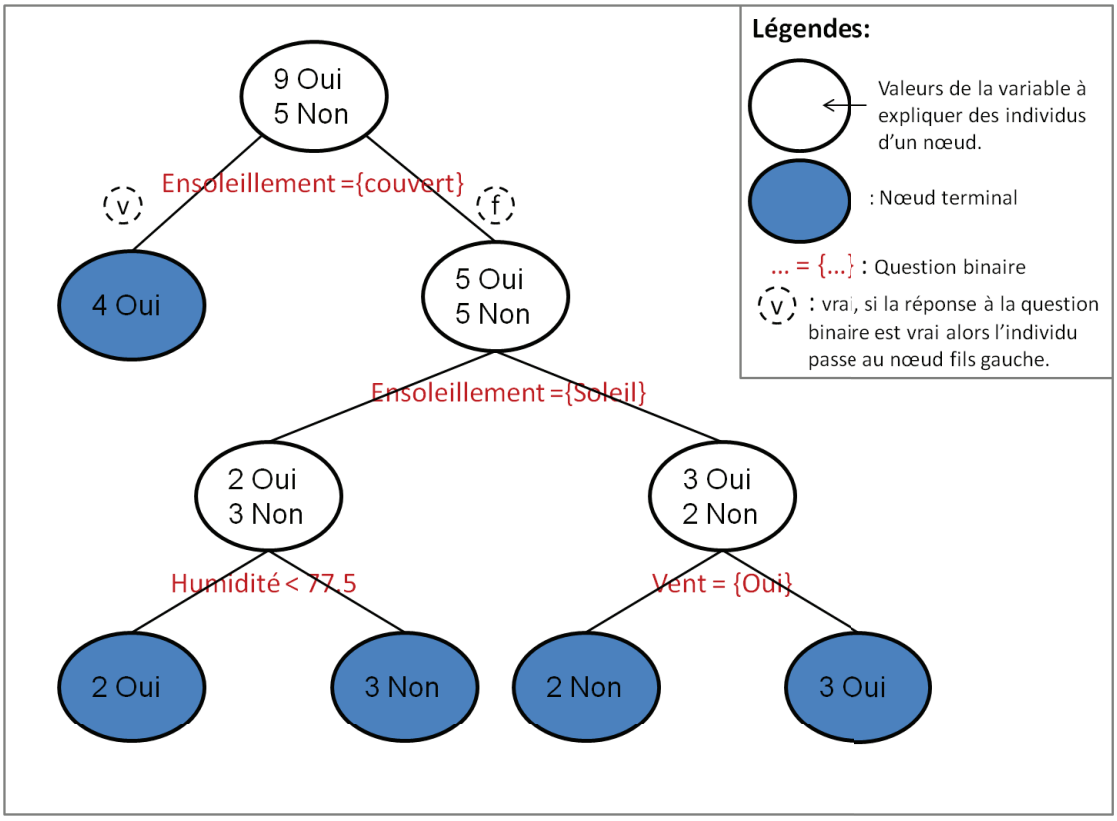


FIGURE 3.1 – Exemple d'arbre binaire de décision sur les données "weather" qui explique le fait de jouer ou non.

découpage. Dans notre exemple illustratif, toutes les variables représentant les conditions météorologiques sont des variables explicatives.

2.1.2 Les nœuds terminaux/non terminaux

Un arbre de décision est représenté par un ensemble de nœuds. Nous en distinguons deux types :

- *Les nœuds non terminaux :* correspondent à l'ensemble des nœuds intermédiaires de l'arbre. Un nœud non terminal possède au moins deux nœuds fils. Il contient un ensemble d'individus sur lesquels une question binaire est appliquée. Cette question concerne la variable explicative la plus discriminante pour la variable à expliquer. Chacun de ses nœuds fils contient un sous-ensemble d'individus correspondant à une réponse à cette question. Le nœud **racine** contient tous les individus de départ. Dans l'exemple illustratif, les 14 individus sont regroupés

dans la racine de l'arbre (figure3.1).

- *Les nœuds terminaux :* représentent l'ensemble des feuilles de l'arbre. Un nœud terminal n'a aucun fils. Chaque nœud terminal représente une classe d'affectation qui sera attribuée aux individus de test. Dans l'exemple illustratif, nous avons 5 nœuds terminaux(figure 3.1).

2.1.3 Ensembles d'apprentissage / de test

Comme toute méthode d'apprentissage, nous distinguons deux ensembles d'individus pour les arbres de décision :

- *L'ensemble d'apprentissage* qui sert à construire l'arbre de décision. La qualité de l'arbre dépend de la qualité des individus de l'ensemble d'apprentissage. Si, par exemple, l'ensemble d'apprentissage ne contenait pas l'une des valeurs possibles de la variable à expliquer alors tous les individus ayant cette valeur seront mal affectés et seront comptés comme erreur.
- *L'ensemble de test* qui sert à évaluer la pertinence de l'arbre de décision. Cet ensemble peut être identique à l'ensemble d'apprentissage ou constitué de nouveaux individus n'ayant pas participé à l'étape de construction.

2.1.4 Entrées / sorties d'un arbre de décision

2.1.4.1 Entrées

Une méthode d'arbre de décision prend en entrées :

- Un ensemble d'apprentissage (nommé aussi "growing set") composé de n individus.
- Une variable à expliquer $C = \{C_1, C_2, \ldots, C_n\}$
- Un ensemble de variables explicatives $Var\_exp = \{y_1, y_2, \ldots, y_k\}$ où $\forall i \in \{1, \ldots, k\}, y_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,n}\}$.
- Un ensemble de conditions d'arrêts que nous notons "stop_condition".

2.1.4.2 Sorties

Chaque méthode d'arbre de décision retourne :

- L'arbre de décision sous format d'un graphique (voir figure 3.1).

- Un ensemble de **règles de décision**. Chaque règle décrit le chemin de la racine de l'arbre à l'une de ses feuilles. Une règle de décision est sous la forme :

  $$\textbf{Si } (\textbf{cond}_1 \wedge \textbf{cond}_2 \wedge \ldots \wedge \textbf{cond}_l) \quad \textbf{alors } \mathbf{C_{aff}} = \textbf{classe du noeud terminal}.$$

  Généralement, la classe d'affectation ($\mathbf{C_{aff}}$) représente la modalité ayant la plus grande fréquence du nœud.

2.1.5 Construction et élagage d'un arbre

2.1.5.1 Construction d'un arbre

Toutes les méthodes d'arbre de décision suivent le même algorithme récursif de construction de leurs arbres (voir algorithme 3).

Bien que l'algorithme de construction d'un arbre de décision soit pratiquement le même, nous trouvons, dans la littérature, plusieurs méthodes d'arbre de décision. La différence entre ces méthodes réside dans leurs critères de découpage et leurs conditions d'arrêt.

a. Critère de découpage

Chaque nœud non terminal est divisé en optimisant un critère, appelé "critère de découpage". Ce dernier est calculé par rapport à la variable à expliquer. Il est différent d'un algorithme à un autre. Nous pouvons citer par exemple :

- l'entropie de Shannon, utilisée dans la méthode C4.5 [89].
- l'indice de Gini [53] utilisé dans la méthode CART [20].
- le test de Khi2 utilisé dans la méthode CHAID [63].
- le critère de Fusinter [115] utilisé dans la méthode SIPINA [116].

b. Conditions d'arrêt

Généralement ces conditions permettent de stopper la construction de l'arbre. Ces conditions peuvent être :

- *locales :* c'est-à-dire relatives à un nœud de l'arbre. Ce type de conditions décide si un nœud est terminal ou pas, nous citons par exemple :

Algorithme 3 : Construction d'un arbre de décision binaire.

Construire_Arbre (Racine, Var_{exp}, C)
début
  si *(stop_condition(Racine)= Vrai)* **alors**
    Racine $\longleftarrow$ Noeud_terminal ;
    Retourner;
  sinon
    $(y_{max}, val_{max}) \longleftarrow$ Chercher_meilleur_decoup(Racine, Var_{exp}, C);
    Partitionner (Racine, $(y_{max}, val_{max}), fils_g, fils_d$);
    Construire_Arbre $(fils_g, Var_{exp}, C)$;
    Construire_Arbre $(fils_d, Var_{exp}, C)$;
  fin
fin
Chercher_meilleur_decoup (Racine, Var_{exp}, C)
début
  $critere_{max} \longleftarrow 0$;
  pour $y_i \in Var_{exp}$ **faire**
    $val_{i,max} \longleftarrow$ meilleur_valeur de découpage de y_i ; *// celle qui maximise critère*
    si *(critere($val_{i,max}$)>* $critere_{max}$*)* **alors**
      $critere_{max} \longleftarrow critere(val_{i,max})$;
      $y_{max} \longleftarrow y_i$;
      $val_{max} \longleftarrow val_{i,max}$;
    fin
  fin
  retourner *(*y_{max}*,* val_{max}*)*;
fin

  - le fait qu'il soit pur, c'est-à-dire que tous les individus du nœud ont la même valeur de la variable à expliquer.
  - le nombre d'individus du nœud est inférieur à un seuil fixé par l'utilisateur.
  - aucune subdivision du nœud n'apporte un *gain informationnel*, c'est le cas par exemple de la méthode CHAID [63] qui se base sur le fait qu'une segmentation n'est acceptée que si le Khi2 calculé sur un sommet est supérieur à un seuil fixé.

- *globales :* c'est-à-dire relatives à l'arbre, nous citons par exemple :
  - la hauteur maximale de l'arbre.

– le nombre de feuilles dans l'arbre.

2.1.5.2 L'élagage de l'arbre

Cette opération est apparue avec la méthode CART [20]. Elle a été inventée pour remédier aux problèmes d'avoir des arbres trop petits ou trop longs. Les arbres trop petits résultaient généralement de conditions d'arrêts trop strictes, alors que ceux trop longs d'un sur-apprentissage par rapport à l'ensemble d'apprentissage.

Le principe de l'élagage est de construire, dans un premier temps, l'arbre le plus pur possible et de le réduire ensuite en utilisant un critère pour comparer des arbres de différentes tailles.
Dans la littérature deux approches d'élagage ont été proposées :

- La première propose de transformer le problème d'apprentissage en un problème d'optimisation en utilisant des formulations Bayésiennes [106]. Le critère à optimiser représente un compromis entre la complexité de l'arbre et sa représentativité des données. Cette approche est peu connue et n'est utilisée que dans certaines méthodes [65, 23].
- La deuxième approche se base sur l'utilisation d'un ensemble de validation connu aussi sous le nom *"pruning set"*. Cet ensemble est utilisé pour estimer le taux d'erreur engendré par la subdivision d'un nœud. Le *pruning set* peut être le même que l'ensemble d'apprentissage (comme pour la méthode C4.5 [89]) ou un échantillon de l'ensemble d'apprentissage non utilisé pour la construction de l'arbre (comme pour la méthode CART [20]).

2.1.6 Matrice de confusion et taux de bonne affectation

Une fois l'arbre de décision construit, il est indispensable d'évaluer sa prédiction. Dans la littérature cette évaluation peut être faite en utilisant :

- Un seul ensemble d'individus. Cet ensemble peut être :
  - L'ensemble d'apprentissage constitué par les individus ayant servit à construire l'arbre.
  - Un ensemble de test composé de nouveaux individus dont nous connaissons la valeur de la variable à prédire.
- La méthode de validation croisée qui consiste à : (i) partitionner l'ensemble d'apprentissage en k échantillons de tailles égales, (ii) répéter k fois la construction d'un arbre sur (k-1) échantillons et le tester sur l'échantillon restant et (iii) faire la moyenne des k taux d'erreurs obtenus.

Les résultats des tests sont généralement représentés sous la forme d'une matrice appelée "Matrice de Confusion" (voir tableau 3.2). Cette matrice est de taille $c \times c$, où c représente le nombre de valeurs différentes de la variable à expliquer. Chaque élément représente le nombre des valeurs de la variable à expliquer C qui ont été affectés à l'une des valeurs possibles de C, formellement : $M_{ij} = nb\Big(C_i, aff(C_j)\Big)$, où la fonction aff représente la classe d'affectation.

		Classes d'affectation			
		C_1	...	C_c	
Valeurs réelles	C_1	M_{11}		M_{1n}	Nb(C_1)
	⋮	⋮		⋮	⋮
	C_c	M_{c1}		M_{cc}	Nb(C_c)
		$Nb\_aff(C_1)$	...	$Nb\_aff(C_c)$	N

Tableau 3.2 – Matrice de confusion.

À partir de cette matrice nous pouvons extraire les informations suivantes :

- Le taux de bonne affectation représenté par l'équation 3.1.

$$\mathbf{Acc} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{c}} \mathbf{M_{ii}}}{\mathbf{N}} \tag{3.1}$$

- Le taux d'erreurs représenté par l'équation 3.2

$$\mathbf{erreur} = \mathbf{1} - \mathbf{Acc} \tag{3.2}$$

Exemple illustratif

Le tableau 3.4 illustre la matrice de confusion résultant du test de l'arbre de décision (représenté par la figure 3.1) sur l'ensemble de test (voir tableau 3.3). La classe d'affectation de chaque individu est représentée par la colonne "C_aff" du tableau 3.3. À partir de cette matrice nous avons un taux de bonne affectation :

$$Acc = \frac{2+3}{8} = 0.625.$$

Ensoleillement	Température (°F)	Humidité(%)	Vent	Jouer	C_aff
Pluie	87	85	Oui	Non	Non
Soleil	80	90	Oui	Non	Non
Pluie	85	85	Non	Non	Oui
Soleil	72	80	Oui	Oui	Non
Pluie	69	70	Non	Oui	Oui
Couvert	85	90	Oui	Non	Oui
Couvert	83	78	Non	Oui	Oui
Couvert	64	70	Oui	Oui	Oui

Tableau 3.3 – Ensemble de test de l'arbre de décision sur les données "Weather".

	Oui	Non	
Oui	3	1	4
Non	2	2	4
	5	3	8

Tableau 3.4 – Matrice de confusion de l'arbre de décision des données "Weather"

2.2 Méthodes d'arbre de décision existantes

La méthode des arbres de décision est l'une des méthodes d'apprentissage supervisé les plus connues et les plus utilisées. Dans la littérature, plusieurs travaux se sont intéressés à la création et à la manipulation d'arbres de décision. Les premiers travaux remontent aux années soixante lorsque Morgan et Sonquist ont utilisé les arbres de régression dans un processus de prédiction et d'explication "***AID*** (*Automatic Interaction Detection*)" [81]. De nos jours, de nombreuses méthodes permettent de construire des arbres de décision. Ces méthodes peuvent être divisées en deux groupes :

- **Les méthodes classiques** qui traitent les données classiques.
- **Les méthodes symboliques** qui résultent de l'extension des méthodes classiques aux données symboliques.

2.2.1 Méthodes classiques

Ces méthodes prennent en entrée une base de données classique décrite par des variables quantitatives et qualitatives. Plusieurs méthodes ont été mises en place, nous citons dans ce qui suit les plus connues classées par ordre chronologique d'apparition :

- **ID3** [90] : cette méthode produit des arbres de décision non binaires. C'est-à-dire qu'un nœud est subdivisé en autant de nœuds que de valeurs distinctes de la variable de découpage. Elle ne traite que les variables nominales. Les variables continues doivent subir un pré-traitement en utilisant une méthode de discrétisation. Le critère de découpage maximisé par ID3 est un gain informationnel qui se base sur l'entropie de Shannon.

- **CART** *"Classification And Regression Tree"* [20] : cette méthode construit des arbres de décision binaires. C'est-à-dire que chaque nœud est subdivisé en deux nœuds fils. Cette méthode accepte tous types de variables (qualitatives et/ou quantitatives). Pour traiter les variables continues,une méthode de discrétisation locale y est intégrée permettant de chercher la meilleure valeur de coupure possible. Cette méthode gère les valeurs manquantes. Le critère de découpage est la variation d'impureté basée sur l'indice de GINI.
  Cette méthode intègre une procédure d'élagage appelée *"Minimal Cost Complexity Pruning"* qui se base sur l'utilisation d'un *pruning set* pour évaluer l'erreur générée par la subdivision d'un nœud. Le découpage d'un nœud interne peut être annulé si le taux d'erreur généré par sa subdivision est supérieur au taux sans subdivision.

- **C4.5**[89] : cette méthode est une amélioration de l'algorithme ID3. En effet, elle accepte les variables continues et les valeurs manquantes. Le critère de découpage d'ID3 favorise les variables ayant plusieurs modalités lors de la sélection de la variable de coupure. Pour remédier à ce défaut, C4.5 utilise le ratio de gain d'informations comme critère de découpage d'un nœud.
  Cette méthode intègre une procédure d'élagage pour réduire l'arbre construit. Cette procédure utilise l'ensemble d'apprentissage comme *"pruning set"* pour calculer une estimation du taux d'erreur.

2.2.2 Méthodes symboliques

Les méthodes symboliques sont les résultats de l'extension des méthodes classiques aux données symboliques. Parmi ces méthodes nous trouvons :

- **La méthode TREE** [87] permet de construire un arbre de décision binaire à partir de données symboliques. La variable à expliquer doit être de type nominal. Les variables explicatives sont de type histogramme ou intervalle, cependant le mélange de différents types de variables explicatives n'est pas permis. Le critère de découpage utilisé est l'indice de Gini, l'entropie de Shannon ou la log-vraisemblance.

- **La méthode de segmentation sur un tableau de variables aléatoires**[1] permet de construire un arbre de décision binaire qui traite des variables explicatives de type histogramme. La variable à expliquer peut être de type nominal ou histogramme à deux modalités uniquement. Le critère de découpage est la variation de l'indice de Gini ou l'entropie de Shannon.

- **La méthode SYCLAD** [77] permet la description symbolique d'une classe C d'individus issus d'une population. Elle combine un critère d'homogénéité par rapport aux variables explicatives, un critère de discrimination par rapport à la variable à expliquer et un critère de débordement de la description de la classe C. Dans cette méthode la variable à expliquer est de type nominal.

- **La méthode STREE** [97] permet de construire un arbre de décision symbolique binaire. Les variables explicatives peuvent être des intervalles, des histogrammes ou un mélange des deux. La variable à expliquer peut être de type histogramme ou nominal. Le critère de découpage utilisé est l'inertie inter-classe.

Chacune de ces méthodes symboliques a ses avantages et ses limites. La plupart d'entre elles (à l'exception de STREE) n'acceptent pas de mélanger différents types de variables explicatives et ne traitent que le cas où la variable à expliquer est de type nominale (à l'exception de STREE [97] et de la méthode de segmentation d'un tableau de variables aléatoires [1]).

Afin de remédier à ces problèmes nous avons mis en place une nouvelle méthode d'arbres de décision symbolique, nommée SyrTree. C'est l'extension de la méthode CART [20] aux données symboliques. Notre méthode traite différent types de variables explicatives (histogrammes, intervalles, nominales et continues). SyrTree permet de construire des arbres en prenant comme variable à expliquer : une variable nominale, une variable histogramme ou la classe d'individus. Nous présentons, dans ce qui suit cette méthode.

3 Nouvelle méthode d'arbres de décision symbolique : SyrTree

Afin d'étendre l'algorithme de construction d'arbre de décision CART aux données symboliques, nous avons mis en place la méthode SyrTree. Cette méthode permet de créer des arbres de décision binaires à partir de données symboliques. Elle prend en entrée un fichier de données symboliques (.Syr), la spécification des variables (à expliquer et explicatives) et les conditions d'arrêts. Elle retourne l'arbre de décision, les règles de décision et la possibilité de tester l'arbre sur de nouvelles données classiques ou symboliques.

Dans cette section, nous présentons d'abord l'algorithme de notre méthode en spécifiant pour chaque type de variable à expliquer les critères de découpage utilisés. Ensuite, nous nous intéressons aux différentes stratégies possibles pour construire un arbre de décision symbolique en partant d'un ensemble de données classiques tout en donnant un exemple d'application pour chaque stratégie. Enfin, nous comparons les résultats de "SyrTree" avec d'autres méthodes d'arbres de décision.

3.1 Algorithme de construction d'un arbre de décision en utilisant SyrTree

Comme tout algorithme d'arbres de décision, "SyrTree" utilise un algorithme récursif pour la construction de l'arbre (voir algorithme 3). Pour définir notre méthode, nous avons spécifié les conditions d'arrêt du découpage d'un nœud, ainsi que les critères de découpages (qui dépendent du type de la variable à expliquer) et le traitement des variables explicatives symboliques (histogrammes et intervalles) et classiques (continues et nominales).

3.1.1 Conditions d'arrêt du découpage d'un nœud

Nous avons considéré qu'un nœud est terminal si :

- il est pur, c'est-à-dire que tous les éléments du nœud ont la même valeur de la variable à expliquer ;
- il n'y a plus de découpage possible ;
- le niveau du nœud est égal à la hauteur maximale de l'arbre, fixée par l'utilisateur ;
- ou le nombre d'individus dans ce nœud est égal au seuil fixé par l'utilisateur.

3.1.2 Critères de découpage

L'expression du critère de découpage varie selon le type de la variable à expliquer qui peut être la classe d'individus, une variable nominale ou un histogramme. Étant donné que notre but est d'étendre CART [20], nous avons mis en place des critères de découpage qui optimisent à chaque fois l'obtention des nœuds les plus purs possibles. C'est-à-dire que dans notre solution nous avons essayé de préserver l'esprit de l'indice de Gini (voir équation 3.3) utilisé comme critère de découpage dans CART.

$$\Delta I = I(N) - \left(\frac{N_d}{N} I(N_{fd}) + \frac{N_g}{N} I(N_{fg})\right) \tag{3.3}$$

où

- N_{fd} (respectivement N_{fg}) est le nœud fils droit (respectivement gauche).
- N_d (respectivement N_g) est le nombre d'individus dans le nœud fils droit (respectivement gauche).
- $I(N) = 1 - \sum_i^k f_i^2$ où k représente le nombre de classes à prédire et f_i la fréquence de la classe i dans le nœud N.

Nous présentons dans ce qui suit les critères de découpages utilisés pour chaque type de variable à expliquer.

3.1.2.1 Cas où la variable à expliquer est la classe symbolique

Dans le cas où la variable à expliquer est la classe d'individus symbolique, chaque valeur de la variable à expliquer est décrite par une seule ligne du fichier symbolique. Dans ce cas, chaque case du fichier symbolique représente la valeur de l'une des variables explicatives. Ces dernières peuvent être de type histogramme, intervalle, nominale ou continue. Le tableau 3.5 illustre un exemple d'un fichier symbolique décrivant les iris de Fisher où la variable à expliquer "Species" est la classe symbolique.

Species	PetalLength_Hist3	SepalLength	SepalWidth
Concept	Histogram	Interval	Interval
-	1 :<2.45, 2 :2.45to4.85, 3 :>4.85	-	-
Setosa	1(1.0)	4.3 :5.8	2.3 :4.4
Versicolor	2(0.92)3(0.08)	4.9 :7.0	2.0 :3.4
Virginica	2(0.06)3(0.94)	4.9 :7.9	2.2 :3.8

Tableau 3.5 – Exemple d'un fichier symbolique où la variable à expliquer représente la classe symbolique "Species".

Dans ce cas l'expression du critère de découpage et le traitement des variables explicatives dépendront du type de ces variables.

a. Critère de découpage et traitement d'une variable explicative de type histogramme

Un histogramme est représenté par m modalités où chacune est pondérée par une fréquence ($\in [0, 1]$) (voir 2.3.1.1). Pour le traitement de ce type de variable explicative,

nous nous sommes inspirés de la méthode proposée par Vrac et Diday [105] où la valeur du découpage est représentée par une modalité pondérée par une fréquence. Nous obtenons ainsi une question binaire sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp}} \leq \mathbf{freq_{decoup}} * \mathbf{mod_{decoup}}\right) \textbf{ Alors } \mathbf{noeud_{fils\_gauche}} \textbf{ Sinon } \mathbf{noeud_{fils\_droit}}$$

où :

- var_{exp} représente la variable explicative histogramme ;
- $freq_{decoup}$ représente la fréquence de découpage ;
- mod_{decoup} représente la modalité de découpage.

Pour trouver la meilleure modalité de découpage parmi les m modalités d'un histogramme, nous avons mis en place un critère qui permet de retourner la modalité qui différencie le plus l'une des classes par rapport aux autres. Par exemple, dans la figure 3.2 la variable de découpage utilisée a différencié la classe d'individus C2 par rapport aux autres classes.

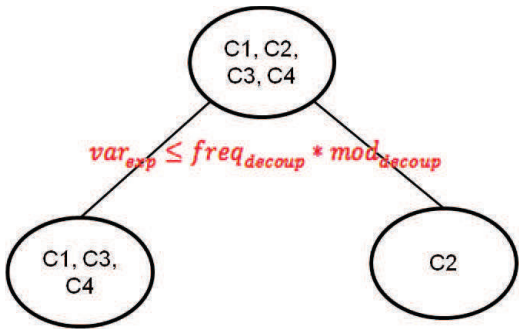


FIGURE 3.2 – Subdivision d'un nœud en utilisant une variable explicative de type histogramme, dans le cas où la variable à expliquer est la classe d'individus symbolique.

Pour traduire cette notion de discrimination d'une classe par rapport aux autres, nous avons mis en place un critère qui calcule la distance minimale entre la fréquence maximale et l'ensemble des fréquences d'une modalité. L'expression de ce critère est représentée par l'équation 3.4.

$$\mathbf{crit}(\mathbf{mod_i})_{\mathbf{i}\in\mathbf{Mod(H)}} = \mathbf{min}\left(\mathbf{max}\Big(\mathbf{f}(\mathbf{i},\mathbf{j})\Big)_{\mathbf{j}\in\mathbf{Conc}} - \mathbf{f}(\mathbf{i},\mathbf{j})\right) \tag{3.4}$$

où

- **H** représente la variable explicative de type histogramme ;
- **Mod(H)** représente l'ensemble des modalités de **H** ;
- **Conc** représente l'ensemble des classes ;
- et **f(i,j)** représente la fréquence de la $i^{ème}$ modalité de l'histogramme décrivant la $j^{ème}$ classe.

Le meilleur découpage pour une variable explicative de type histogramme coïncide avec la modalité, qui maximise le critère de découpage, pondérée par la valeur du critère. C'est-à-dire que :

- $\mathbf{freq_{decoup}} = \mathbf{crit}(\mathbf{mod_{decoup}})$
- $\mathbf{mod_{decoup}}$ est la modalité qui maximise le critère.

SyrTree permet de sélectionner la modalité la plus discriminante ou un sous-ensemble de modalités. Pour les sous ensembles de modalités, dès que le critère ne s'améliore plus nous stoppons la recherche. En d'autres termes, si le meilleur critère avec une modalité est supérieur au meilleur critère de deux modalités nous ne testons pas les sous-ensembles de taille 3. Pour la sélection des sous-ensembles de modalités nous distinguons deux cas :

- Si l'histogramme est nominal modal (les modalités sont nominales) : le sous-ensemble peut être constitué par n'importe quel combinaison des modalités de l'histogramme. En d'autres termes, si nous avons une variable avec trois modalités {A, B, C} les sous ensembles de taille deux sont inclus dans {{A, B}, {A, C}, {B, C}}.
- Si l'histogramme est ordinal modal (les modalités sont ordonnées) : le sous ensemble est constitué par des combinaisons de modalités juxtaposées. C'est-à-dire si nous avons une variable avec trois modalités {A, B, C} les sous ensembles de taille deux sont inclus dans {{A, B}, {B, C}}.

Dans le cas où nous avons un sous ensemble de modalités le critère de découpage sera :

$$\mathbf{crit}(\mathbf{ens_{mod}})_{\mathbf{ens_{mod}} \subset \mathbf{Mod(H)}} = \mathbf{min}\left(\mathbf{max}\Big(\mathbf{f}\big(\mathbf{ens_{mod}}(\mathbf{j})\big)\Big)_{\mathbf{j} \in \mathbf{Conc}} - \mathbf{f}\big(\mathbf{ens_{mod}}(\mathbf{j})\big)\right) \tag{3.5}$$

où

- $\mathbf{ens_{mod}}$ est un sous-ensemble des modalités de **H** ;
- $\mathbf{f}\big(\mathbf{ens_{mod}}(\mathbf{j})\big) = \sum_{\mathbf{k} \in \mathbf{ens_{mod}}} \mathbf{f}(\mathbf{mod_k}(\mathbf{j}))$: est la somme des fréquences des modalités constituant ens_{mod} pour la $j^{ème}$ classe.

Interprétation du critère de découpage

Notre critère retourne la distance entre les deux plus grandes fréquences d'une modalité donnée. Nous avons utilisé cette distance à la place de la distance moyenne entre toutes les fréquences car l'objectif est de déterminer la modalité possédant une fréquence (pour une classe) nettement plus grande que toutes ses autres fréquences (pour les autres classes). Cette information ne peut pas être donnée par la distance moyenne. Si nous prenons par exemple les trois classes décrites par le tableau 3.6 et les résultats des distances moyennes et de notre critère représentés dans le tableau 3.7, nous remarquons que la distance moyenne ne favorise aucune modalité par rapport aux autres alors que notre critère retournerait la modalité 2 comme meilleur valeur de découpage.

C1	1(0.5) 2(0.2) 3(0.3)
C2	1(0.2) 2(0.8)
C3	1(0.3) 3(0.7)

Tableau 3.6 – Description de trois classes d'individus C1, C2, C3 par une variable histogramme de trois modalités.

Modalité	Distance moyenne	Valeur crit(mod)
1	$\frac{(0.5+0.2+0.3)}{3} = \frac{1}{3}$	0.2
2	$\frac{(0.8+0.2)}{3} = \frac{1}{3}$	0.6
3	$\frac{(0.3+0.7)}{3} = \frac{1}{3}$	0.4

Tableau 3.7 – Résultats de calcul des distances moyennes et du critère de SyrTree appliqué à l'exemple du tableau 3.6

Exemple d'application

Soit le tableau symbolique, représenté par le tableau 3.8, où Species est la classe et SepalLength_Hist3 est une variable explicative de type histogramme. En appliquant notre critère d'évaluation (voir equation 3.4) sur les trois modalités de l'histogramme "<5.45", "5.45to6.15", et ">6.15", nous obtenons :

Species	SepalLength_Hist3
Concept	Histogram
-	1 :<5.45, 2 :5.45to6.15, 3 :>6.15
setosa	1(0.9)2(0.1)
versicolor	1(0.12)2(0.56)3(0.32)
virginica	1(0.02)2(0.2)3(0.78)

Tableau 3.8 – Exemple d'un tableau symbolique décrivant la classe d'individus "Species" par une variable de type histogramme "SepalLength_Hist3"

- crit(<5.45) = 0.9-0.12 = 0.78
- crit(5.45to6.15) = 0.56-0.2 = 0.36
- crit(>6.15) = 0.78-0.32 = 0.46

Notre exemple représente un histogramme à valeurs ordinales modales donc les sous-ensembles possibles sont {"<5.45", "5.45to6.15"} et {"5.45to6.15", ">6.15"}. En appliquant notre critère d'évaluation (voir equation 3.5) sur ces deux sous-ensembles de modalités, nous obtenons :

- crit({"<5.45", "5.45to6.15"}) = 1-0.68 = 0.42
- crit({"5.45to6.15", ">6.15"}) = 0.98-0.88 = 0.1

Dans cet exemple le meilleur découpage possible en utilisant la variable "SepalLength_Hist3" est de découper par rapport à $0.78* < 5.45$. La figure 3.3 illustre l'arbre de SyrTree sur cet exemple d'application en mettant un seul niveau de découpage.

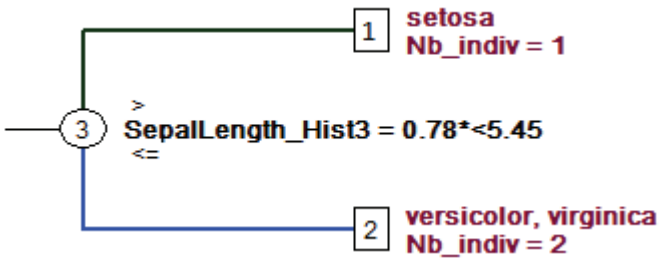


FIGURE 3.3 – Exemple du découpage de la variable "SepalLength_Hist3" décrivant les classes "Species".

b. Critère de découpage et traitement d'une variable explicative de type intervalle

Une variable explicative de type intervalle Y est définie par c valeurs Y_i où :

$$\forall i \in \{1, \ldots, c\}, Y_i = [v_{inf}, v_{sup}];\ v_{inf} \in \mathbb{R}\ et\ v_{sup} \in \mathbb{R}$$

Pour le traitement des intervalles nous nous sommes inspirés du traitement des variables continues dans le cas classique. Dans ce cas la question binaire est sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp}} \leq \mathbf{val_{decoup}}\right) \quad \textbf{Alors} \quad \mathbf{noeud_{fils\_gauche}} \quad \textbf{Sinon} \quad \mathbf{noeud_{fils\_droit}}$$

avec :

- $\mathbf{var_{exp}}$: la variable explicative intervalle ;
- $\mathbf{val_{decoup}}$: la valeur de découpage.

Pour traiter une variable explicative de type intervalle nous commençons par la recherche de l'ensemble des seuils possibles. La plupart des méthodes d'arbre de décision symboliques existantes qui traitent les variables intervalles commencent par trier l'ensemble de toutes les bornes inférieures et supérieures des intervalles et prennent l'ensemble des milieux de ces bornes comme valeurs de seuil [87, 97]. Nous avons constaté que dans le cas où les intervalles ne se croisent pas (c'est-à-dire qu'il n'y a pas d'intersection entre eux), ce choix d'ensemble de départ peut engendrer un découpage qui n'est pas optimal du point de vu pureté des nœuds. Nous utilisons l'exemple, représenté par le tableau 3.9, pour expliquer notre point de vue.

C1	[0, 1]
C2	[1, 2.5]
C3	[2, 3]

Tableau 3.9 – Exemple d'un tableau symbolique décrivant 3 classes par une variable intervalle.

L'ensemble des seuils possibles pour Stree [97] est égal à {0.5, 1.5, 2.25, 2.75}. Le problème avec cet ensemble initial est, par exemple, la perte de l'information que "dans l'intervalle [0,1] il n'y a que des individus de classe C1" et "dans l'intervalle]2.5,3] il n'y a que des individus de classe C3".

Afin de remédier à ce défaut nous prenons l'ensemble des bornes des intervalles privées de leurs valeurs minimales et maximales comme valeurs possibles de seuil.
Afin d'évaluer une coupure possible, nous avons adapté l'indice de Gini (equation 3.3)

pour qu'il prenne en compte les valeurs pondérées.
L'impureté de GINI d'un nœud N est donnée par :

$$I(N) = 1 - \sum_{i=1}^{k} \left(\frac{f_i}{eff(N)} \right)^2 \tag{3.6}$$

où

- c : nombre de classes d'individus ;
- f_i : la fréquence de la classe i dans le nœud N (avec $f_i \in \mathbb{R}^+$;
- et $eff(N) = \sum_{i \in N} f_i$.

Le Gain en impureté d'une coupure est donné par :

$$\Delta I = I(N_p) - \left(\frac{eff(N_{fd})}{eff(N_p)} I(N_{fd}) + \frac{eff(N_{fg})}{eff(N_p)} I(N_{fg}) \right) \tag{3.7}$$

où

- N_p : le nœud père ;
- et N_{fd} (respectivement N_{fg}) est le nœud fils droit (respectivement gauche).

Pour chaque valeur de coupure possible nous calculons la proportion de chaque concept qui sera assignée à l'un des 2 nœuds fils. Ces proportions représentent les f_i de chaque nœud. Ensuite nous gardons la valeur de coupure qui maximise ΔI (voir equation 3.7). $I(N_p)$ étant le même pour un nœud donné, en conséquence, il suffit de minimiser la somme pondérée des impuretés des nœuds fils :

$$\Delta S = \left(\frac{eff(N_{fd})}{eff(N_p)} I(N_{fd}) + \frac{eff(N_{fg})}{eff(N_p)} I(N_{fg}) \right) \tag{3.8}$$

Interprétation du critère

En reprenant l'exemple représenté par le tableau 3.9, l'ensemble des seuils possibles est égal à {1, 2, 2.5}. Avec cet ensemble nous réglons le problème de perte d'informations que nous avons constaté avec les autres méthodes. Cependant il faut un critère qui retourne le meilleur seuil qui donne des nœuds les plus pures possibles pour la variable à expliquer. L'utilisation du critère de Gini modifié permet d'estimer cette pureté puisqu'il prend en compte les proportions de chaque concept assignées à l'un des nœuds fils. La figure 3.4 représente les résultats des trois coupures possibles.

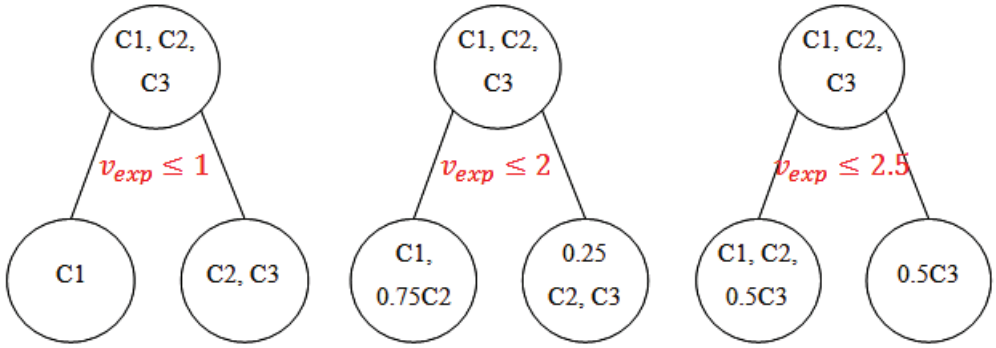


FIGURE 3.4 – Résultats des coupures possibles appliquées aux données du tableau 3.9

En se basant sur les résultats représentés par la figure 3.4, nous estimons que la meilleure valeur de coupure est égale à 1. Vérifions que notre critère retourne cette valeur de coupure.

$$S(1) = \frac{1}{3} \times \left(1 - (\frac{1}{1})^2\right) + \frac{2}{3} \times \left(1 - (\frac{1}{2})^2 - (\frac{1}{2})^2\right) = 0.33$$

$$S(2) = \frac{1.75}{3} \times \left(1 - (\frac{1}{1.75})^2 - (\frac{0.75}{1.75})^2\right) + \frac{1.25}{3} \times \left(1 - (\frac{0.25}{1.25})^2 - (\frac{1}{1.25})^2\right) = 0.42$$

$$S(2.5) = \frac{2.5}{3} \times \left(1 - (\frac{1}{2.5})^2 - (\frac{1}{2.5})^2 - (\frac{0.5}{2.5})^2\right) + \frac{0.5}{3} \times \left(1 - (\frac{0.5}{0.5})^2\right) = 0.53$$

Le calcul de S confirme que le meilleur seuil est égal à 1.

Exemple d'application

Soit le tableau Symbolique représenté par le tableau 3.10, où "Species" représente la classe et "SepalLength" est une variable explicative de type intervalle. En appliquant la méthodologie présentée auparavant, l'ensemble des valeurs de coupures possibles est $\{4.9, 5.8, 7\}$.
L'application de notre méthode découpage d'une variable intervalle, intégrée dans SyrTree, donne :

- Pour la valeur 4.9, nous obtenons deux nœuds fils composés de :
  - nœud fils gauche : 0.4* Setosa
  - nœud fils droit : 0.6 *Setosa, Versicolor, Verginica

Species	SepalLength
Concept	Interval
-	-
setosa	4.3 :5.8
versicolor	4.9 :7.0
virginica	4.9 :7.9

Tableau 3.10 – Exemple d'un tableau symbolique contenant un concept "Species" et une variable de type intervalle "SepalLength"

Le calcul de l'impureté donne :

$$\begin{aligned} I(N_{fg}) &= 1 - \left(\frac{0.4}{0.4}\right)^2 = 0; \\ I(N_{fg}) &= 1 - \left(\left(\frac{0.6}{2.6}\right)^2 + \left(\frac{1}{2.6}\right)^2 + \left(\frac{1}{2.6}\right)^2\right) = 0.65 \end{aligned} \tag{3.9}$$

La somme pondérée des impuretés des nœuds fils est donc égale à :

$$S(4.9) = \frac{0.4}{3} \times 0 + \frac{2.6}{3} \times 0.65 = 0.563$$

- De la même façon en calculant S pour 5.8 et pour 7 nous obtenons :

$$\begin{aligned} S(5.8) &= \frac{1.78}{3} \times 0.573 + \frac{1.272}{3} \times 0.495 = 0.54 \\ S(7) &= \frac{2.7}{3} \times 0.658 + \frac{0.3}{3} \times 0 = 0.592 \end{aligned}$$

Dans cet exemple, le meilleur découpage possible pour la variable intervalle "SepalLength" est la valeur "5.8". La figure 3.5 illustre le résultat de l'application de SyrTree en fixant à 1 le nombre de niveaux de coupures.

3.1.2.2 Cas où la variable à expliquer est nominale

Dans le cas où la variable à expliquer est de type nominal, nous prenons la variation de l'impureté de GINI comme critère de découpage. La seule différence avec la méthode classique CART est le traitement des variables explicatives symboliques de type histogramme et intervalle.

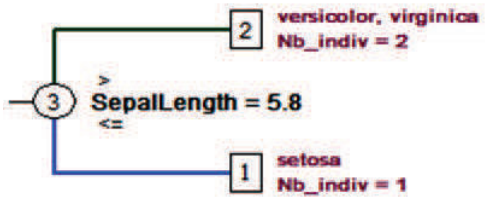


FIGURE 3.5 – Exemple du découpage de la variable intervalle "SepalLength" décrivant la variable à expliquer "Species".

a. Traitement d'une variable explicative de type histogramme

En se basant sur le cas où la variable à expliquer est la classe symbolique, la question binaire est de la forme :

$$\textbf{Si } \left(\textbf{var}_{\textbf{exp}} \leq \textbf{freq}_{\textbf{decoup}} * \textbf{mod}_{\textbf{decoup}}\right) \textbf{ Alors } \textbf{noeud}_{\textbf{fils\_gauche}} \textbf{ Sinon } \textbf{noeud}_{\textbf{fils\_droit}}$$

Dans le cas où la variable à expliquer est nominale, les valeurs de coupures possibles sont les modalités de l'histogramme pondérées par la moyenne de leurs fréquences. Le critère de découpage est la variation de l'impureté de GINI (voir equation 3.3).

Exemple d'application

Étant donné le tableau 3.11 représentant un extrait du fichier symbolique décrivant les classes "pays européens $\times$ un taux de pauvreté. Ces classes sont décrites par deux variables : "Country" qui représente les pays et "happy" qui représente la joie des personnes de classe. Notre objectif est de chercher la meilleur valeur de découpage de "happy" en prenant "Country" comme variable à expliquer.

En appliquant le traitement décrit auparavant les valeurs de coupures possibles sont : {0.076 * C, 0.252 * B, 0.671 * A}. Les résultats de découpage par rapport à ces valeurs sont représentés par la figure 3.6. D'après cette figure nous pouvons calculer S pour chaque valeur de coupure possible :

$$S(0.076 * C) = \frac{4}{6}\left(1 - \left((\frac{2}{4})^2 + (\frac{1}{4})^2 + (\frac{1}{4})^2\right)\right) + \frac{2}{6}\left(1 - \left((\frac{1}{2})^2 + (\frac{1}{2})^2\right)\right) = 0.42$$

$$S(0.252 * B) = S(0.671 * A) = \frac{4}{6}\left(1 - \left((\frac{2}{4})^2 + (\frac{2}{4})^2\right)\right) + \frac{2}{6}\left(1 - \left((\frac{2}{2})^2\right)\right) = 0.33$$

Dans ce cas, deux valeurs de coupure donnent le meilleur critère. Ainsi, notre algorithme retournera la première valeur testée qui est "0.252 * B".

Afin de vérifier nos calculs, le fichier décrit par le tableau 3.11 est entré à SyrTree. L'arbre expliquant "Country" en utilisant "happy" comme variable explicative avec un seul niveau de découpage est représenté par la figure 3.7.

Country_pov	Country	Happy
Concept	Nominal	Histogram
-	-	1 :C,2 :B,3 :A
BE_0	BE	1(0.005)2(0.084)3(0.909)
BE_1	BE	1(0.052)2(0.250)3(0.697)
BG_0	BG	1(0.060)2(0.428)3(0.510)
BG_1	BG	1(0.255)2(0.462)3(0.281)
CH_0	CH	1(0.007)2(0.102)3(0.890)
CH_1	CH	1(0.078)2(0.184)3(0.737)

Tableau 3.11 – Extrait des données décrivant les classes des pays européens croisés par un taux de pauvreté subjective [57].

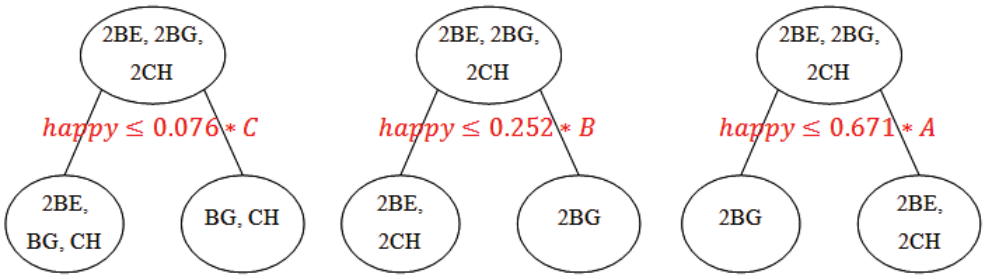


FIGURE 3.6 – Résultats des découpages de la variable "happy" du tableau 3.11

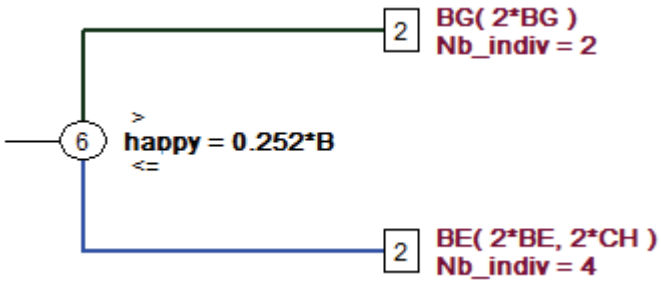


FIGURE 3.7 – Exemple d'arbre de décision, résultat de SyrTree, en utilisant une variable à expliquer nominale et une variable explicative histogramme.

b. Traitement d'une variable explicative de type intervalle

Pour traiter une variable explicative de type intervalle, nous appliquons le même principe que dans le cas où le variable à expliquer est le concept. C'est-à-dire que les

valeurs de coupures possibles sont les bornes des intervalles privées de leurs valeurs minimale et maximale. Et la question binaire sera sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp}} \leq \mathbf{val_{decoup}}\right) \quad \textbf{Alors} \quad \mathbf{noeud_{fils\_gauche}} \quad \textbf{Sinon} \quad \mathbf{noeud_{fils\_droit}}$$

Pour les mêmes raisons que dans le cas où la variable à expliquer est la classe d'individus symbolique, le critère de découpage est la variation de l'impureté de GINI modifiée (voir equation 3.7).

Exemple d'application

Étant donné le tableau 3.12 représentant un extrait du fichier symbolique décrivant les classes issues du croisement ″des pays européens″ par ″un taux de pauvreté″. Ces classes sont décrites par les deux variables : ″Country″ qui représente les pays et ″equality″ qui décrit l'égalité entre les gens du même pays. Notre objectif est de chercher la meilleur valeur de découpage de la variable ″equality″ en prenant ″Country″ comme variable à expliquer.

Country_pov	Country	Equality_i
Concept	Nominal	Interval
-	-	-
BG_0	BG	-0.536 :0.544
BG_1	BG	-0.773 :0.495
CH_0	CH	-0.006 :1.033
CH_1	CH	-0.076 :1.152

Tableau 3.12 – Extrait des données décrivant les classes ″des pays européens croisés par un taux de pauvreté subjective″ par la variable explicative ″equality″ [57].

En appliquant notre méthode de traitement de variables intervalles, l'ensemble des seuils possibles est : {-0.536, -0.076, -0.006, 0.495, 0.544, 1.033}.

En se basant sur les résultats des différents découpages de la variable ″Equality″ représentés par la figure 3.8, nous pouvons calculer la valeur de S de chaque découpage

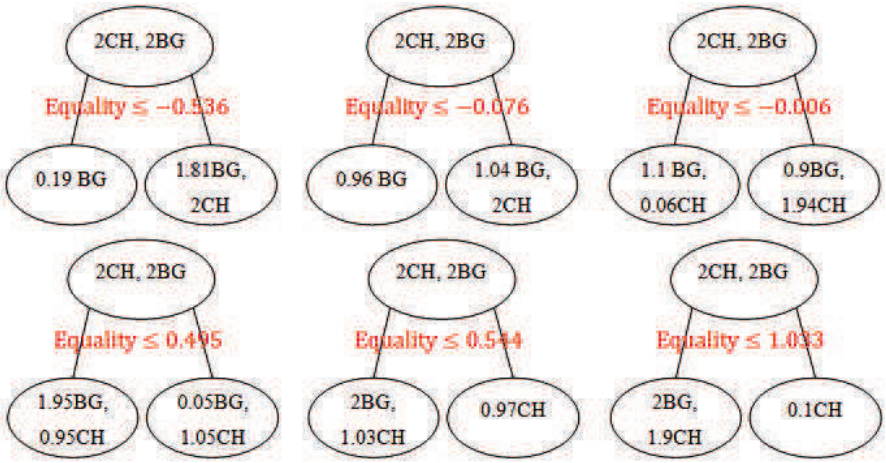


FIGURE 3.8 – Résultats des découpages de la variable "Equality" du tableau 3.12.

pour déterminer le meilleur seuil.

$$S(-0.536) = \frac{0.19}{4}\left(1 - (\frac{0.19}{0.19})^2\right) + \frac{3.89}{4}\left(1 - ((\frac{1.81}{3.98})^2 + (\frac{2}{3.98})^2)\right) = 0.5$$

$$S(-0.076) = \frac{0.98}{4}\left(1 - (\frac{0.98}{0.98})^2\right) + \frac{3.02}{4}\left(1 - ((\frac{1.02}{3.02})^2 + (\frac{2}{3.02})^2)\right) = 0.335$$

$$S(-0.006) = \frac{1.16}{4}\left(1 - ((\frac{1.1}{1.16})^2 + (\frac{0.06}{1.16})^2)\right) + \frac{2.84}{4}\left(1 - ((\frac{0.9}{2.84})^2 + (\frac{1.94}{2.84})^2)\right) = 0.337$$

$S(0.495) = 0.34$

$S(0.544) = 0.34$

$S(1.033) = 0.48$

D'après ces résultats -0.076 représente le meilleur seuil. L'arbre résultat de SyrTree, en fixant à 1 le nombre de niveaux de découpages, est représenté par la figure 3.9.

c. Traitement des variables explicatives continues et nominales

Pour le traitement des variables explicatives classiques (continues et nominales), nous utilisons la même stratégie que la méthode CART [20] où :

- Pour une variable continue : nous prenons l'ensemble des bornes frontières comme valeurs de coupures possibles. Parmi cette liste nous cherchons la valeur qui optimise la variation de GINI (voir equation 3.3).
- Pour une variable nominale : la liste initiale des découpages possibles est représentée par les différentes valeurs de cette variable. Parmi les éléments de cette liste nous cherchons la valeur qui optimise la variation de GINI.

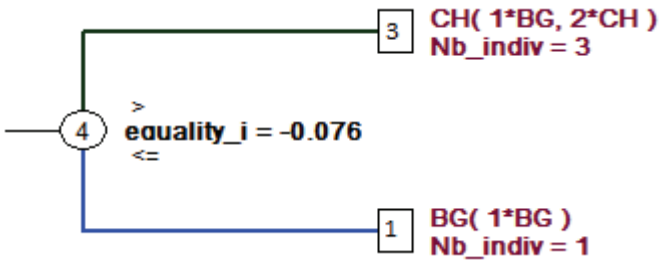


FIGURE 3.9 – Exemple d'arbre de décision en utilisant une variable à expliquer nominale et une variable explicative histogramme.

3.1.2.3 Cas où la variable à expliquer est de type histogramme

a. Critère de découpage

Expression

Dans le cas où la variable à expliquer est de type histogramme, nous avons mis en place un critère qui optimise la pureté des nœuds afin de regrouper les histogrammes qui ont les mêmes modalités dans les mêmes nœuds. L'expression de ce critère est représentée par l'équation 3.10.

$$\mathbf{Score(decoup)} = \sum_{\mathbf{i=1}}^{\mathbf{m}} \frac{\mathbf{score(m_i)}}{\mathbf{m}} \tag{3.10}$$

où :

$$\begin{aligned} \mathbf{score(m_i)} &= \left|\mathbf{moy(f_g(m_i))} - \mathbf{moy(f_d(m_i))}\right| \quad \mathbf{si} \quad \mathbf{m_i} \in \{\mathbf{noeud_g} \cap \mathbf{noeud_d}\} \\ &= \mathbf{1} \quad \mathbf{sinon} \end{aligned}$$

avec :

- m : nombre de modalités de l'histogramme
- moy : la fonction moyenne ;
- $f_g(m_i)$: les fréquences de la modalité m_i dans le nœud fils gauche ;
- $f_d(m_i)$: les fréquences de la modalité m_i dans le nœud fils droit.

Interprétation du critère

Notre critère de découpage dans le cas où la variable à expliquer est un histogramme favorise l'obtention de nœuds contenant des histogrammes décrits par

les mêmes modalités. En effet, la valeur **1** que nous donnons lorsqu'une modalité est présente dans un seul des deux nœuds fils favorise les découpages qui regroupent les histogrammes suivant l'homogénéité de leurs modalités.

b. Traitement des variables explicatives de type continue

Pour une variable explicative continue, nous commençons par trier les valeurs. Ensuite, nous prenons les milieux entre deux valeurs successives comme ensemble de seuils possibles. Enfin, nous évaluons le score de chaque découpage et nous prenons celui qui possède une valeur maximale.

Exemple d'application

Conc	X	Y
Concept	Continu	Histogram
-	-	1 :R,2 :B,3 :V
1	1	1(0.1)2(0.8)3(0.1)
2	2	1(0.9)3(0.1)
3	2	2(1)
4	3	1(0.3)3(0.7)
5	4	1(0.8)3(0.2)

Tableau 3.13 – Tableau de données avec une variable à expliquer de type histogramme et une variable explicative de type continue.

Soient les données symboliques représentés par tableau 3.13, où Y représente la variable à expliquer et X la variable explicative. Dans cet exemple nous avons trois découpages possibles pour la variable X {1.5, 2.5 et 3.5}.
L'évaluation de ces 3 découpages en utilisant le critère de SyrTree pour les variables à expliquer de type histogramme, nous donne :

- Pour le découpage "X = 1.5" nous obtenons deux nœuds fils :
  - nœuds fils gauche : (0.1R, 0.8B, 0.1V)
  - nœuds fils droit : (0.9R, 0.1V), (B),(0.3R, 0.7V), (0.8R, 0.2V)

  Le score des différentes modalités est :
  - score(R)= $|0.1 - \frac{0.9+0+0.3+0.8}{4}| = 0.4$
  - score(B)= $|0.8 - \frac{0+1+0+0}{4}| = 0.55$
  - score(V)= $|0.1 - \frac{0.1+0+0.7+0.2}{4}| = 0.15$

Le score du découpage est :

- score(1.5) = $\frac{0.4+0.55+0.15}{3}$ = 0.367

- De la même façon en calculant le score pour 2.5 et pour 3.5. Nous obtenons :
  - score(2.5) = $\frac{0.266+1+0.383}{3}$ = 0.549
  - score(3.5) = $\frac{0.475+1+0.05}{3}$ = 0.508

Dans cet exemple le meilleur seuil est égal à 2.5. Nous avons vérifié ce résultat en introduisant le fichier symbolique décrivant les données du tableau 3.13 à SyrTree. L'arbre obtenu en fixant à 1 le nombre des niveaux de découpage est représenté dans la figure 3.10.

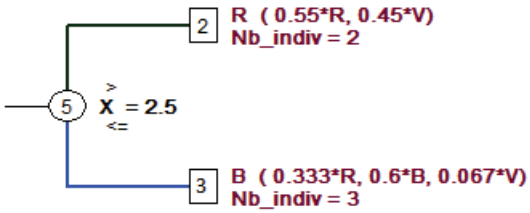


FIGURE 3.10 – Exemple d'arbre de décision, résultat de SyrTree, utilisant une variable à expliquer histogramme et une variable explicative continue.

c. Traitement des variables explicatives nominales

Pour une variable explicative nominale, l'ensemble des coupures possibles est représenté par l'ensemble des différentes valeurs de la variable explicative. Pour chaque valeur de coupure possible, nous évaluons le critère de découpage et nous prenons celle qui le maximise.

Exemple d'application

Soit les données symboliques représentées par le tableau 3.14, où Y représente la variable à expliquer et X la variable explicative. Dans cet exemple nous avons 3 valeurs de coupures possibles pour la variable X : {A, B et C}. L'évaluation de ces 3 découpages en utilisant le critère de SyrTree pour les variables à expliquer de type histogramme, nous donne :

- Pour le découpage X=A nous obtenons deux nœuds fils :
  - nœuds fils gauche : (0.1R ; 0.8B ; 0.1V), (B).
  - nœuds fils droit : (0.9R ; 0.1V), (0.3R ; 0.7V), (0.8R ; 0.2V)

Conc	X	Y
Concept	Nominal	Histogram
-	-	1 :R,2 :B,3 :V
1	A	1(0.1)2(0.8)3(0.1)
2	B	1(0.9)3(0.1)
3	A	2(1)
4	C	1(0.3)3(0.7)
5	B	1(0.8)3(0.2)

Tableau 3.14 – Tableau de données avec une variable à expliquer de type histogramme et une variable explicative nominale.

Le score des différentes modalités est

- score(R) = $|\frac{0.1}{2} - \frac{0.9+0.8+0.3}{3}| = 0.617$
- score(B) = 1
- score(V) = $|\frac{0.1}{2} - \frac{0.1+0.7+0.2}{3}| = 0.283$

Le score du découpage est :

- score(A) = $\frac{0.617+1+0.283}{3} = 0.633$

- Pour le découpage X=B nous obtenons deux nœuds fils :
  - Nœuds fils gauche : (0.9R ; 0.1V), (0.8R ; 0.2V)
  - Nœuds fils droit : (0.1R ; 0.8B ; 0.1V), (B), (0.3R ; 0.7V)

  Le score des différentes modalités est :

  - score(R) = $|\frac{0.9+0.8}{2} - \frac{0.1+0.3}{3}| = 0.717$
  - score(B) = 1
  - score(V) = $|\frac{0.1+0.2}{2} - \frac{0.1+0+0.7}{3}| = 0.117$

  Le score du découpage est :

  - score(B) = $\frac{0.717+1+0.117}{3} = 0.611$

- Pour le découpage X=C nous obtenons deux nœuds fils :
  - Nœuds fils gauche : (0.3R ; 0.7V)
  - Nœuds fils droit : (0.9R ; 0.1V), (0.1R ; 0.8B ; 0.1V), (B), (0.8R ; 0.2V)

Le score des différentes modalités est :

- score(R) = $|0.3 - \frac{0.9+0.1+0.8}{4}| = 0.15$
- score(B) = 1
- score(V) = $|0.7 - \frac{0.1+0.1+0.2}{3}| = 0.567$

Le score du découpage

- score(C) = $\frac{0.15+1+0.567}{3} = 0.572$

D'après les valeurs du critère pour les différents découpages possibles, la meilleure coupure est pour X=A. Nous avons vérifié ce résultat en introduisant le fichier symbolique représenté par le tableau 3.14 à SyrTree en fixant à 1 le nombre de niveaux de découpage. La figure 3.11 représente l'arbre obtenu.

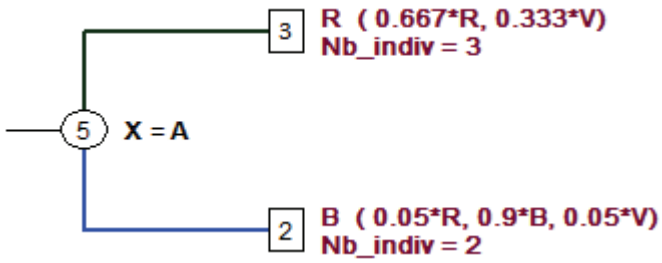


FIGURE 3.11 – Exemple d'arbre de décision en utilisant une variable à expliquer histogramme et une variable explicative nominale.

d. Traitement des variables explicatives histogrammes

Pour le cas où la variable à expliquer est un histogramme, nous traitons les variables explicatives histogrammes de la même façon que le cas où la variable à expliquer est nominale. C'est-à-dire que l'ensemble des valeurs de découpages possibles est représenté par les modalités pondérées par leurs moyennes. Après la fixation de cet ensemble, nous calculons pour chaque valeur le critère de découpage d'une variable explicative histogramme (voir equation 3.10. Finalement la meilleure valeur de découpage est celle qui maximise notre critère.

e. Traitement des variables explicatives intervalles

Pour le cas où la variable à expliquer est un histogramme, nous traitons les variables explicatives intervalles de la même façon que le cas où la variable à expliquer est nominale. C'est-à-dire que l'ensemble des valeurs de découpages possibles est

représenté par les différentes bornes des intervalles privés de leurs valeurs minimale et maximale. Pour chaque seuil possible, nous calculons le critère de découpage d'une variable explicative histogramme. Finalement la meilleure valeur de découpage est celle qui maximise notre critère.

3.2 Classe d'affectation

La classe d'affectation représente la valeur que nous associons à un nœud terminal. Cette valeur dépend du type de la variable à expliquer qui peut être :

- **La classe symbolique :** dans ce cas la classe d'affectation coïncide avec la valeur de la classe du nœud terminal.
- **Une variable nominale :** dans ce cas, nous utilisons le principe de majorité. C'est-à-dire que la classe d'affectation représentera la valeur, de la variable à expliquer, qui a le plus d'occurrences dans le nœud terminal. La description des individus présents dans le nœud est sous le format :
  $\mathbf{classe_{affectation}}(\mathbf{occ_1} * \mathbf{val_1}, \ldots, \mathbf{occ_k} * \mathbf{val_k})$ où k représente le nombre des valeurs différentes dans le nœud.
- **Une variable histogramme :** dans ce cas la classe d'affectation sera représentée par la modalité qui a la plus grande fréquence dans l'histogramme moyen. Cet histogramme est le résultat de la moyenne de tous les histogrammes du nœud terminal.

3.3 Méthode d'élagage de l'arbre SyrTree

Pour l'élagage des arbres résultats de SyrTree nous nous sommes inspirés de la procédure utilisée dans la méthode CART [20]. Cette méthode se base sur la subdivision de l'ensemble d'apprentissage en deux sous ensembles. Le premier, appelé *"growing set"*, pour la construction de l'arbre alors que le deuxième, appelé *"pruning set"*, servira à tester et à supprimer les découpages inutiles. Chaque découpage est évalué en suivant les étapes suivante :

1. Le calcul des nombres d'individus mal-classés dans les 3 nœuds : le nœud père et ses deux nœuds fils.
2. La comparaison entre le nombre de mal-classés du nœud père et la somme des mal-classés de ses fils. Si cette somme est supérieure au nombre des mal-classés du père alors le découpage est annulé et ne figurera pas dans l'arbre final.

Formellement :

$$\textbf{Si } (\mathbf{nb_{erreur}(N_p)} < \mathbf{nb_{erreur}(N_{fg}) + nb_{erreur}(N_{fd})}) \quad \textbf{alors} \quad \textbf{annulation du découpage}.$$

Nous utilisons cette procédure d'élagage dans les cas où la variable à expliquer est nominale ou histogramme. Pour le cas où c'est la classe symbolique elle est inapplicable puisque l'arbre obtenu est le plus petit arbre pouvant décrire les différentes valeurs de la variable à expliquer et chaque valeur de la variable à expliquer sera présente dans un seul nœud terminal.

3.4 Le module de test et de validation de SyrTree

Ce module de SyrTree sert à tester un arbre de décision symbolique sur un ensemble de test. Ce dernier peut être constitué par des données symboliques ou par des données classiques. Le module de test de SyrTree prend en entrée : l'arbre symbolique ou l'ensemble des règles de décision et les données de test. Il retourne : le taux de bonne affectation, la matrice de confusion et la classe d'affectation de chaque élément du fichier de test. La procédure de l'affectation d'un nouvel individu à un arbre de décision symbolique diffère suivant la nature de cet individu (classique ou symbolique).

3.4.1 Affectation d'un individu classique (de premier ordre)

Un individu classique est décrit par des variables nominales ou continues. La problématique dans ce cas est de savoir comment appliquer les tests binaires utilisant des variables symboliques (histogrammes et intervalles) sur des variables classiques ? Nous répondons à cette question dans ce qui suit.

3.4.1.1 Cas où la variable de découpage est un histogramme

Dans ce cas le test binaire est sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp} > freq_{decoup} * mod_{decoup}}\right) \quad \textbf{Alors} \quad \mathbf{noeud_{fils\_droit}} \quad \textbf{Sinon} \quad \mathbf{noeud_{fils\_gauche}}$$

Afin d'affecter un individu classique à l'un des deux nœuds fils, nous avons traduit ce test sous le format d'un test sur une variable classique. Cette traduction dépend de la nature de l'histogramme :

- Si c'est un *histogramme nominal modal* (c'est-à-dire que les modalités sont des valeurs nominales) alors le test sera traduit en :

  $$\textbf{Si } \left(\mathbf{var_{exp}\ In\ \{mod_{decoup}\}}\right) \quad \textbf{Alors} \quad \mathbf{noeud_{fils\_droit}} \quad \textbf{Sinon} \quad \mathbf{noeud_{fils\_gauche}}$$

- Si c'est un histogramme ordinal (c'est-à-dire que les modalités sont des intervalles), le test sera traduit en :

  $\mathbf{Si}\ \left(\mathbf{var_{exp}} \in \mathbf{interv_{decoup}}\right)\ \mathbf{Alors}\ \mathbf{noeud_{fils\_droit}}\ \mathbf{Sinon}\ \mathbf{noeud_{fils\_gauche}}$

  où

$$interv_{decoup} = \bigcup_{i=1}^{nb(mod\_decoup)} intervalle(mod_{decoup_i})$$

Exemple d'application

Soit l'ensemble de test extrait des données des Iris de Fisher représenté par le tableau 3.15. Dans cet exemple nous allons étudié les deux types d'histogrammes en variable de découpage. Dans chaque cas, nous présentons un exemple d'arbre de décision, l'application de la traduction des questions et les résultat de l'affectation des individus de test aux différentes arbres de décisions.

SepalLength_Hist3	SepalLength_disc	Species	C_{aff} arbre1	C_{aff} arbre2
5.1	A	setosa	setosa	setosa
4.3	A	setosa	setosa	setosa
7	C	versicolor	virginica	virginica
5.7	B	versicolor	versicolor	versicolor
7.1	C	virginica	virginica	virginica
4.9	A	virginica	setosa	setosa

Tableau 3.15 – ensemble de test d'un arbre de décision dans le cas où la variable à expliquer est un histogramme.

Cas où la variable de découpage est un histogramme nominal modal : Soit l'arbre de décision construit sur les données des "Iris" représenté par la figure 3.12. Cet arbre a été construit en prenant la classe "Species" comme variable à expliquer et l'histogramme "SepalLength_disc" comme variable explicative.
La traduction des deux questions binaires de l'arbre de la figure 3.12 donne :

- Pour le découpage de la racine :

$$Si\ \left(SepalLength\_disc\ >\ 0.78\ *\ A\right)\quad Alors\quad C_{aff} = "setosa"\quad Sinon\quad noeud_{fg}$$

$$\Updownarrow$$

$$Si\ \left(SepalLength\_disc\quad IN\quad \{A\}\right)\quad Alors\quad C_{aff} = "setosa"\quad Sinon\quad noeud_{fg}$$

- Pour le découpage du fils gauche de la racine :

$$Si\ \Big(SepalLength\_disc\ >\ 0.46\ *\ C\Big)\quad Alors\quad C_{aff} = "virginica"$$
$$Sinon\quad C_{aff} = "versicolor"$$
$$\Updownarrow$$
$$Si\ \Big(SepalLength\_disc\quad IN\quad \{C\}\Big)\quad Alors\quad C_{aff} = "virginica"$$
$$Sinon\quad C_{aff} = "versicolor"$$

Les résultats de l'affectation de cet arbre sur les données de test sont représentés par la quatrième colonne "C_{aff} arbre1" du tableau 3.15.

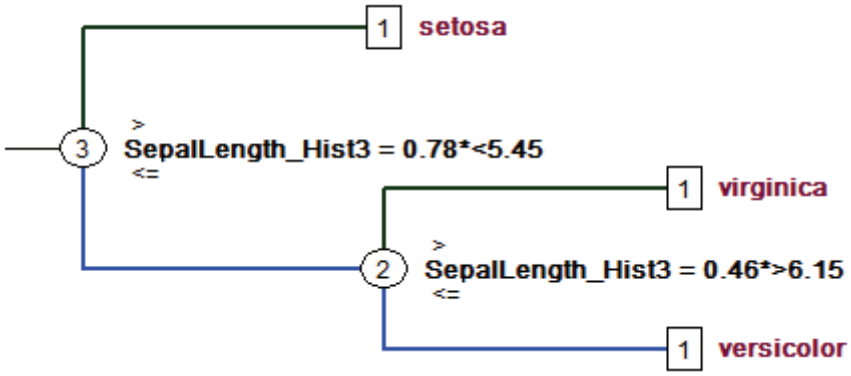


FIGURE 3.12 – Arbre de décision sur les données de Fisher en utilisant l'histogramme de SepalLength discrétisé comme variable explicative.

Cas où la variable de découpage est un histogramme ordinal : Soit l'arbre de décision représenté par la figure 3.13. Il a été construit sur les données des Iris de Fisher en prenant la classe "Species" comme variable à expliquer et la variable histogramme ordinal "SepalLength_Hist3 comme variable à expliquer.
La traduction des deux questions binaires de cet arbre nous donne :

- Pour le découpage de la racine :

$$Si\ \Big(SepalLength\_Hist3\ >\ (0.78\ *\ <5.45)\Big)\quad Alors\quad C_{aff} = "setosa"\quad Sinon\quad noeud_{fg}$$
$$\Updownarrow$$
$$Si\ \Big(SepalLength\_Hist3\quad \leq\quad \{5.45\}\Big)\quad Alors\quad C_{aff} = "setosa"\quad Sinon\quad noeud_{fg}$$

- Pour le découpage du fils gauche de la racine :

$$Si\ \Big(SepalLength\_Hist3\ >\ (0.46 *\ >\ 6.15)\Big)\ Alors\ C_{aff} = "virginica"$$
$$Sinon\ C_{aff} = "versicolor"$$
$$\Updownarrow$$
$$Si\ \Big(SepalLength\_Hist3\ >\ 6.15\Big) \qquad Alors\ C_{aff} = "virginica"$$
$$Sinon\ C_{aff} = "versicolor"$$

Les résultats de l'affectation de cet arbre sur les données de test sont représentés par la quatrième colonne "C_{aff} arbre2" du tableau 3.15.

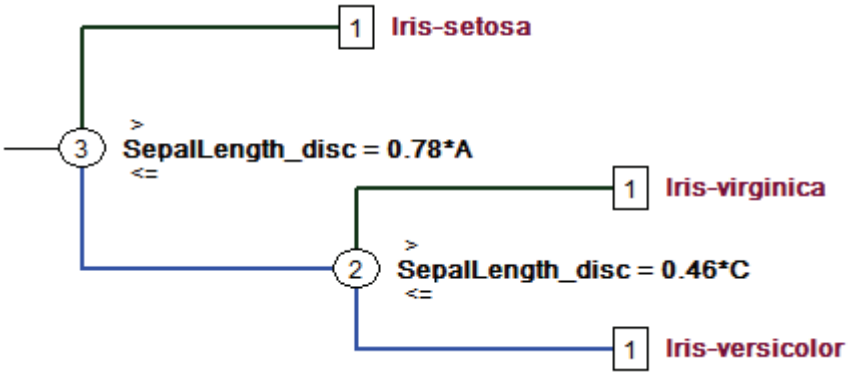


FIGURE 3.13 – Arbre de décision sur les données des Iris en utilisant l'histogramme SepalLength_Hist3 comme variable explicative.

3.4.1.2 Cas où la variable de découpage est un intervalle

Dans ce cas la question binaire est sous la forme :

$$\mathbf{Si}\ \Big(\mathbf{var_{exp}} \leq \mathbf{val_{decoup}}\Big)\ \ \mathbf{Alors}\ \ \mathbf{noeud_{fils\_gauche}}\ \ \mathbf{Sinon}\ \ \mathbf{noeud_{fils\_droit}}$$

Cette question binaire possède la même forme que celle d'une variable explicative continue donc aucune traduction à faire pour l'affectation d'un individu classique.

3.4.2 Affectation d'individus symboliques (de deuxième ordre)

Dans ce cas les individus de test sont décrits par des variables symboliques et nous devons répondre à deux questions :

1. Quel est le résultat d'une question de découpage binaire ? La réponse à cette question dépend du type de la variable explicative.
2. Est-ce une bonne affectation ou non ? La réponse à cette question dépend du type de la variable à expliquer dans le fichier de test.

3.4.2.1 Quel est le résultat d'une question de découpage binaire ?

a. Cas d'une variable explicative nominale

Dans ce cas la question binaire est sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp} = m_{decoup}}\right) \quad \textbf{Alors} \quad \mathbf{noeud_{fils\_gauche}} \quad \textbf{Sinon} \quad \mathbf{noeud_{fils\_droit}}$$

Si dans le fichier de test var_{exp} est une variable nominale, il n'y a aucun problème et il suffit de comparer sa valeur à celle de la modalité de découpage. Cependant, si cette variable est un histogramme, il faut comparer la modalité de découpage à la modalité majoritaire de l'histogramme (la modalité de la fréquence maximale). Dans ce cas la question binaire peut être traduite de cette façon :

$$\textbf{Si } \left(\mathbf{m_{max}(var_{exp}) = m_{decoup}}\right) \quad \textbf{Alors} \quad \mathbf{noeud_{fils\_gauche}} \quad \textbf{Sinon} \quad \mathbf{noeud_{fils\_droit}}$$

où m_{max} : la modalité ayant la fréquence maximale de var_{exp}. Cette fréquence doit être strictement supérieure à la moyenne.

Exemple d'application

Y	X	Classe_affectation
Concept	Histogram	
-	1 :A,2 :B,3 :C	
R	2(0.75)3(0.25)	R
B	1(1)	B
V	1(0.5)2(0.5)	R

Tableau 3.16 – Résultat du test d'une question binaire issue du découpage d'une variable nominale sur une variable histogramme.

Soit l'ensemble de test représenté par le tableau 3.16. Nous voulons tester l'arbre représenté par la figure 3.11 sur cet ensemble. Cet arbre propose une seule question binaire :

$$Si \left(X = A\right) \quad Alors \quad classe_{aff} = B \quad Sinon \quad classe_{aff} = R$$

Comme dans l'ensemble de test X est un histogramme, il faut comparer les modalités majoritaires des différentes valeurs de X à "A". Par exemple, pour le premier individu m_{max} (X) = B $\neq$ A donc sa classe d'affectation sera égale à R. De la même façon nous calculons les classes d'affectation des autres individus de test. Le résultat est représenté dans la troisième colonne du tableau 3.16.

b. Cas où la variable explicative est continue ou intervalle

Dans ce cas la question binaire est sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp}} \leq \mathbf{val_{decoup}}\right) \textbf{ Alors } \mathbf{noeud_{fils\_gauche}} \textbf{ Sinon } \mathbf{noeud_{fils\_droit}}$$

Nous distinguons deux cas :

- $\mathbf{var_{exp}}$ est une variable continue dans le fichier de test : dans ce cas, il n'y a aucun problème et il suffit de comparer sa valeur à celle du seuil de découpage val_{decoup}.
- $\mathbf{var_{exp}}$ est de type intervalle : dans ce cas, il faut comparer la valeur de découpage à la moyenne de l'intervalle. La question binaire peut être traduite de cette façon :

  $$\textbf{Si } \left(\mathbf{moy}(\mathbf{var_{exp}}) \leq \mathbf{val_{decoup}}\right) \textbf{ Alors } \mathbf{noeud_{fils\_gauche}} \textbf{ Sinon } \mathbf{noeud_{fils\_droit}}$$

  où

  – $moy(var_{exp}) = \frac{val_{max} - val_{min}}{2}$;
  – val_{max} (respectivement val_{min}) représente la borne maximale (respectivement minimale) de l'intervalle.

Exemple d'application

Y	X	Classe_affectation
Concept	Interval	
-	-	
R	[0, 1]	B
B	[3, 4]	R
V	[0.25, 2]	B

Tableau 3.17 – Résultats d'une question issue d'une variable explicative continue appliquée à une variable intervalle.

Soit l'ensemble de test représenté par le tableau 3.17. Nous voulons tester l'arbre représenté par la figure 3.10 sur cet ensemble. Cet arbre propose une seule règle de décision :

$$Si \left(X \leq 2.5\right) \quad Alors \quad Classe_{aff} = B \quad Sinon \quad Classe_{aff} = R$$

Comme l'ensemble de test X est un intervalle, nous devons comparer les moyennes des différentes valeurs de X à 2.5. Par exemple, pour le premier individu $moy(X) = 0.5$, puisque $0.5 \leq 2.5$ donc sa classe d'affectation sera égale à B. De la même façon nous calculons les classes d'affectation des autres individus de test. Le résultat est représenté dans la troisième colonne du tableau 3.17.

c. Cas d'une variable explicative histogramme

Dans ce cas le test binaire est sous la forme :

$$\textbf{Si } \left(\mathbf{var_{exp}} \leq \mathbf{freq_{decoup}} * \mathbf{mod_{decoup}}\right) \textbf{ Alors } \mathbf{noeud_{fils\_gauche}} \textbf{ Sinon } \mathbf{noeud_{fils\_droit}}$$

Suivant le type de var_{exp} dans le fichier de test, nous distinguons deux cas possibles :

- Si $\mathbf{var_{exp}}$ est une variable nominale ou continue : la transformation du test sera la même que lorsque nous affectons des individus classiques.
- Si $\mathbf{var_{exp}}$ est un histogramme : dans ce cas il suffit de comparer la fréquence de mod_{decoup} de l'individu à la fréquence de découpage pour savoir à quel nœud fils sera affecté l'individu de test.

3.4.2.2 Est-ce une bonne affectation ou non ?

Dans le cas où nous testons l'arbre sur un ensemble d'individus symboliques (ou de second ordre), la décision concernant la justesse de l'affectation dépend du type de la variable à expliquer dans le fichier de test :

- Si la variable à prédire est le concept symbolique ou de type nominal : nous considérons que l'affectation est bonne si la classe d'affectation est égale à la valeur de la variable à expliquer de l'individu de test.
- Si la variable à prédire est de type histogramme : nous considérons que l'affectation est bonne si l'histogramme décrivant le nœud terminal d'affectation est le plus proche de la valeur de l'histogramme de l'individu de test parmi toutes les valeurs des nœuds terminaux de l'arbre. En d'autres termes :

  $$\textbf{Si } D\left(var_{\acute{a}exp}(ind_{test}),\ histo(nd_{aff})\right) = min_{\{ndster\}}\left(D\Big(var_{\acute{a}exp}(ind_{test}),\ histo(nd)\Big)\right)$$

  $$\textbf{Alors} \quad l'affectation\ est\ bonne.$$

avec

- D : est la distance euclidienne entre deux histogrammes,
- $var_{áexp}$: représente la valeur de la variable à expliquer de l'individu de test,
- $histo(nd_{aff})$: représente l'histogramme moyen décrivant le nœud d'affectation,
- $\{ndster\}$: représente l'ensemble des nœuds terminaux de l'arbre,
- et histo(nd) : représente l'histogramme décrivant un nœud terminal de l'arbre.

Exemple d'application

Soit l'ensemble de test représenté par le tableau 3.18. Nous voulons tester l'arbre représenté par la figure 3.11 sur cet ensemble. Cet arbre propose une seule règle de décision :

$$Si\ (X = A) \quad Alors \quad Classe_{aff} = B(0.05 * R, 0.9 * B, 0.005 * V)$$
$$Sinon \quad Classe_{aff} = R(0.667 * R, 0.333 * V)$$

Notre objectif est d'estimer si on a une bonne affectation ou non, dans le cas où la variable à prédire est de type histogramme dans le fichier de test.

Pour savoir si les affectations sont bonnes ou pas il suffit de calculer pour chaque cas les distances entre la valeur de Y et les histogrammes décrivant les deux nœuds terminaux de l'arbre.

- Pour le premier individu de test : $Y_1 = 1(0.7)2(0.3)$.
  - $D(Y_1, C_{aff1}) = (|0.7 - 0.667| + |0.3 - 0| + |0 - 0.333|)/3 = 0.325 = D(Y_1, histo(nd_2))$
  - $D(Y_1, histo(nd_1)) = (|0.7 - 0.05| + |0.3 - 0.05| + |0 - 0.9|)/3 = 0.6$

  Puisque $D(Y_1, C_{aff1}) = min(D(y_1, histo(nd_i)))$ donc nous considérons l'affectation de cet individu de test comme étant une bonne affectation.

- Pour le deuxième individu de test : $Y_2 = 1(0.1)2(0.8)3(0.1)$.
  - $D(Y_2, C_{aff2}) = (|0.1 - 0.05| + |0.8 - 0.9| + |0.1 - 0.05|)/3 = 0.067 = D(Y_2, histo(nd_1))$
  - $D(Y_2, histo(nd_2)) = (|0.1 - 0.667| + |0.8 - 0| + |0.1 - 0.333|)/3 = 0.533$

Puisque $D(Y_2, C_{aff2}) = min(D(Y_2, histo(nd_i)))$ donc nous considérons l'affectation de cet individu de test comme étant une bonne affectation.

- Pour le troisième individu de test : $Y_3 = 2(0.6)3(0.4)$.
  - $D(Y_3, C_{aff3}) = (|0 - 0.667| + |0.6 - 0| + |0.4 - 0.333|)/3 = 0.467 = D(Y_3, histo(nd_2))$
  - $D(Y_3, histo(nd_1)) = (|0 - 0.05| + |0.4 - 0.05| + |0.6 - 0.9|)/3 = 0.233$

  Puisque $D(Y_1, C_{aff1}) \neq min(D(y_1, histo(nd_i)))$ donc nous considérons l'affectation de cet individu de test comme étant une erreur d'affectation.

conc	Y	X	Classe_affectation	Bonne affectation
Concept	Histogram	Histogram		
-	1 :R,2 :B,3 :V	1 :A,2 :B,3 :C		
1	1(0,7)2(0.3)	2(0.75)3(0.25	R (0.667*R, 0.333*V)	Oui
2	1(0.1)2(0.8)3(0.1)	1(1)	B(0.05*R, 0.9*B,0.005*V)	Oui
3	2(0,6)3(0.4)	1(0.5)2(0.5)	R(0.667*R, 0.333*V)	Non

Tableau 3.18 – Exemple de données de test où la variable à prédire est de type histogramme. Calcul et vérification de la classe d'affectation.

4 Stratégies de construction d'arbres à partir de données classiques en utilisant SyrTree

Afin de construire un arbre de décision symbolique à partir de données classiques nous avons mis en places deux stratégies :

- La première consiste à construire l'arbre en partant du fichier symbolique décrivant la variable à expliquer. C'est-à-dire que le fichier symbolique en entrée de SyrTree aura la variable à expliquer comme classe symbolique.
- La deuxième stratégie est basée sur deux étapes : (i) appliquer une méthode de classification symbolique non supervisée aux données initiales, (ii) prendre le résultat de cette classification comme entrée pour notre méthode. Dans ce cas la variable à expliquer sera sous la forme d'un histogramme.

Pour l'application de ces deux stratégies un pré-traitement sur les données classiques initiales est nécessaire. Il concerne la conversion des variables continues en histogrammes les plus discriminants en utilisant ***HistSyr*** et la classification des données en utilisant ***ClustSyr***. La figure 3.14 présente un schéma résumant les différentes étapes et outils nécessaires pour l'application des deux stratégies. Dans ce qui suit nous les décrivons en s'appuyant sur des exemples d'applications.

4.1 Stratégie 1 : la construction des arbres sur les classes d'individus symboliques

4.1.1 Les étapes

Comme nous partons d'un fichier de données classiques, la construction et le test de l'arbre de décision symbolique SyrTree sur les classes passent par plusieurs étapes :

1. Construire deux échantillons à partir du fichier initial. Le premier pour l'apprentissage et le deuxième contiendra les données de test.
2. La construction des fichiers symboliques (d'apprentissage et de test) en prenant la variable à expliquer comme classe d'individus. Dans cette étape nous utilisons le *pattern* résultat d'***HistSyr*** (voir 2.3) pour transformer les variables continues en histogrammes.
3. Utiliser ***SyrTree*** pour construire l'arbre de décision sur les concepts en prenant le fichier symbolique des données d'apprentissage en entrée.
4. Tester l'arbre sur les données classiques d'apprentissage et de test et sur les données symboliques de test.

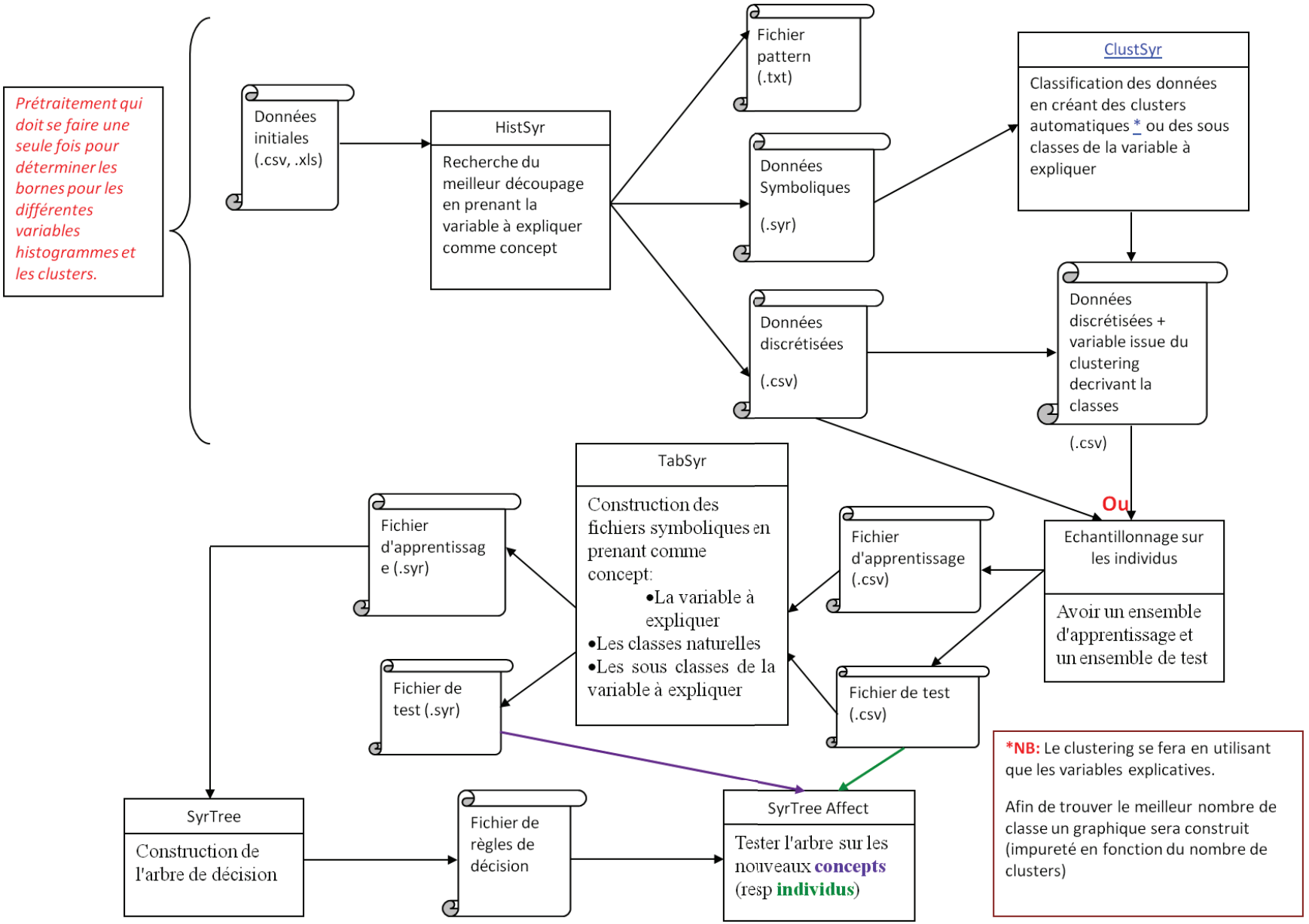


FIGURE 3.14 – Schéma des étapes de construction et de test de l'arbre de décision SyrTree à partir de données classiques.

4.2 Exemple illustratif en utilisant les données des Iris de Fisher

Les données des Iris est une base de données UCI [13] décrivant 150 iris par leurs espèces d'iris et la longueur et la largeur de leurs sépales et pétales. Le tableau 3.19 représente un extrait de ces données. Nous commençons par subdiviser ce fichier en deux fichiers l'un pour l'apprentissage et l'autre pour le test.

SepalLength	SepalWidth	PetalLength	PetalWidth	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
7	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.3	3.3	6	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica

Tableau 3.19 – Extrait des données des Iris.

En utilisant ***HistSyr*** nous transformons les 4 variables continues en histogrammes de 3 modalités. Nous obtenons alors le pattern suivant :

Species :5 :c
SepalLength_Hist3 :Group_1_$-$ $<$ 5.45 $<$ 6.15 $<$ $+$:h
SepalWidth_Hist3 :Group_2_$-$ $<$ 2.95 $<$ 3.05 $<$+ :h
PetalLength_Hist3 :Group_3_$-$ $<$ 2.45 $<$ 4.85 $<$+ :h
PetalWidth_Hist3 :Group_4_$-$ $<$ 0.8 $<$ 1.65 $<$ $+$:h
SepalLength :1 :i
SepalWidth :2 :i
PetalLength :3 :i
PetalWidth :4 :i

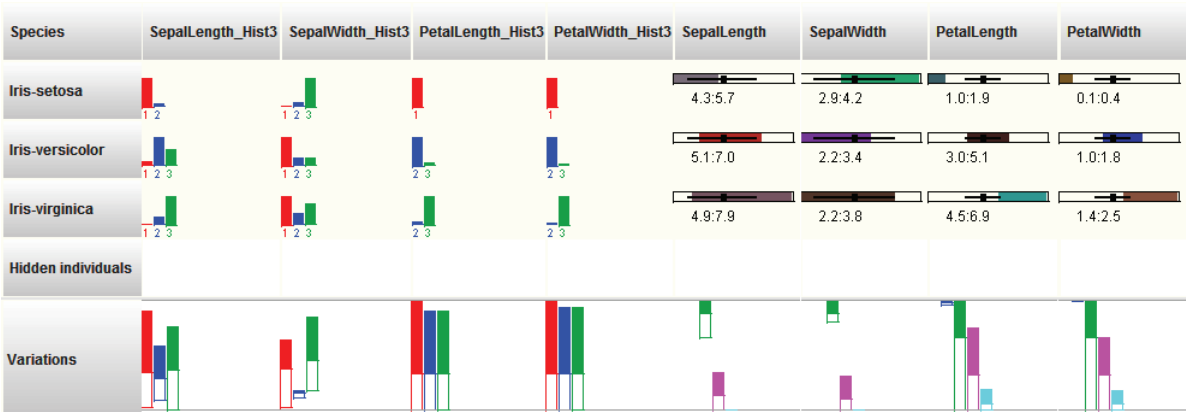
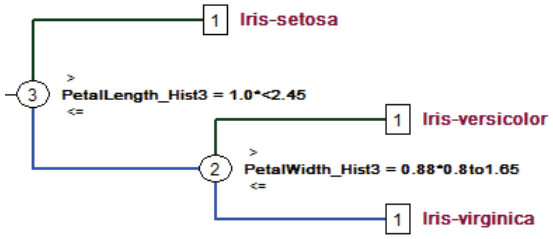


FIGURE 3.15 – Fichier symbolique des données des Iris.

Par la suite nous utilisons ***TabSyr*** pour construire à partir des fichiers classiques, les fichiers symboliques d'apprentissage et de test. La figure 3.15 représente le fichier symbolique des données d'apprentissage.

L'application de ***SyrTree*** sur les données d'apprentissage retourne l'arbre représenté par la partie a) de la figure 3.16 et les règles de décision associées à cet arbre sont illustrées par la partie b) de cette figure.



(a) Arbre de décision.

Decision rule n°1 (node n°3):

If { (PetalLength_Hist3 MORE 1.0*<2.45)}
Then Species = Iris-setosa

Decision rule n°2 (node n°4):

If { (PetalLength_Hist3 LESS 1.0*<2.45)
&
(PetalWidth_Hist3 LESS 0.88*0.8to1.65) }
Then Species = Iris-virginica

Decision rule n°3 (node n°5):

If { (PetalLength_Hist3 LESS 1.0*<2.45)
&
(PetalWidth_Hist3 MORE 0.88*0.8to1.65) }
Then Species = Iris-versicolor

(b) Règles de décision de l'arbre.

FIGURE 3.16 – Présentation de l'arbre et des règles de décisions résultants de l'application de SyrTree sur les données des Iris en prenant la classe d'individus symboliques comme variable à expliquer.

Finalement, nous avons testé l'arbre construit sur les individus d'apprentissage et de test et sur les concepts de test. Après chaque test nous avons la matrice de confusion, le taux de bonne affectation globale et les taux de bonnes affectations par règle. Le tableau 3.20 représente le résultat du test de l'arbre construit sur les classes "Species" sur les individus de l'ensemble de test. D'après ce tableau nous remarquons que sur 48 individus testés, uniquement 2 ont été mal affectés.

***** Matrice de confusion ******				
	Iris-setosa	Iris-versicolor	Iris-virginica	Sum
Iris-setosa	16	0	0	16
Iris-versicolor	0	16	0	16
Iris-virginica	0	2	14	16
Sum	16	18	14	48
***** Good Affectation rate = 95.833%**				
***** Good affectation node**				
Rule number	1	2	3	
	100.0	100.0	88.889	

Tableau 3.20 – Présentation des résultats de test de l'arbre de SyrTree construit en utilisant les classes symboliques "Species" comme variable à expliquer.

4.2.1 Avantages et inconvénients de l'arbre sur les classes d'individus symboliques

En appliquant cette stratégie de construction d'arbres SyrTree sur plusieurs jeux de données nous avons constaté que cette méthode possède des avantages et des inconvénients.

Avantages :

- Le premier avantage de l'arbre SyrTree construit sur les classes symbolique réside dans le fait de pouvoir le tester sur des individus classiques sans être obliger de créer un fichier symbolique intermédiaire, comme c'est le cas pour STREE [97] par exemple.
- Son deuxième avantage est que la méthode accepte et traite différents types de données en variables explicatives, sans exiger un seul type de données à la fois comme c'est le cas de TREE [87].
- Son troisième et plus grand avantage est la longueur de l'arbre. En effet, SyrTree sur les classes symboliques nous construit le plus petit arbre qu'on puisse obtenir avec au maximum une règle de décision par valeur de la variable à expliquer. Cet avantage s'est manifesté par exemple en testant C4.5 sur les données de corrosion présentées précédemment (voir chapitre 2, section 3.4.2.1). L'application de la méthode C4.5 implémentée dans Tanagra[91] nous a donné un arbre de 85 nœuds contenant 43 feuilles (voir figure 3.17) alors que l'arbre sur les classes symboliques de SyrTree donnera au maximum 4 feuilles (4 étant le nombre des classes). La taille de l'arbre sur les classes d'individus symboliques de SyrTree engendre une facilité à le comprendre et à l'interpréter.

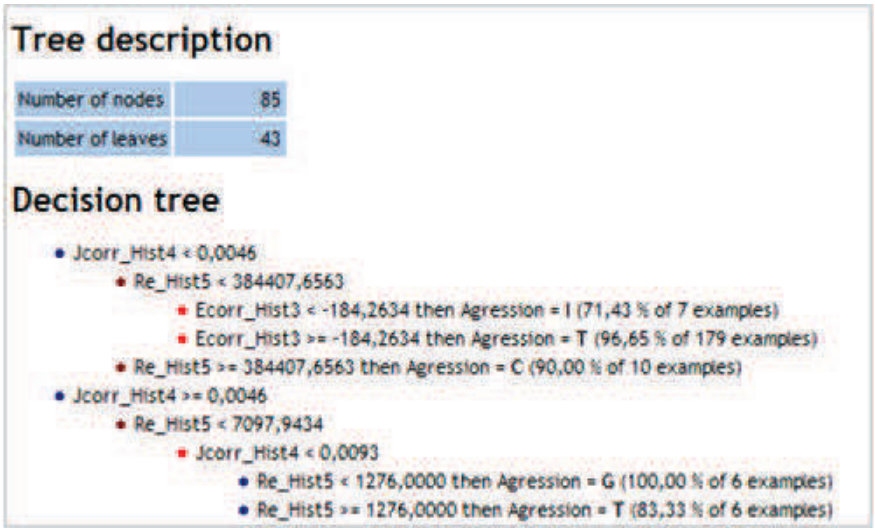


FIGURE 3.17 – Extrait de l'arbre obtenu en appliquant C4.5 sur les données de corrosion.

Inconvénients :

- Toutefois, avoir le plus petit arbre ne signifie pas forcément que c'est le meilleur de point de vue prédiction. Par exemple, si nous reprenons l'étude sur les corrosions, l'arbre

obtenu par C4.5 donne un taux de 79,5% de bonne affectation alors que celui de SyrTree est de 65,81% (voir section 6.1).

- L'arbre sur les concepts présente un autre inconvénient. Lorsque la variable à expliquer est de type binaire SyrTree sur les concepts nous donne un arbre avec deux feuilles. Cela signifie qu'il va découper par rapport à la variable explicative la plus discriminante pour les deux concepts et l'algorithme s'arrêtera (puisque les deux nœuds sont purs). La figure 3.18 représente l'arbre sur les concepts construit sur les données UCI [13] de " breast-cancer-wisconsin" qui ont une variable classe binaire.

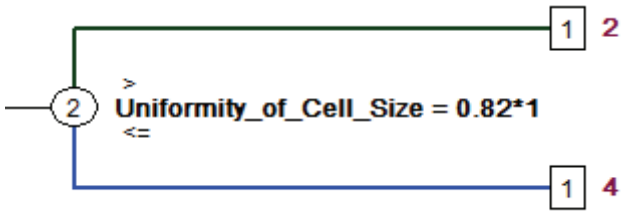


FIGURE 3.18 – Exemple d'un arbre SyrTree sur les classes symboliques : cas d'un classe binaire.

4.3 Stratégie 2 : Construction des arbres en se basant sur le résultat d'une classification

4.3.1 Les étapes

Afin de trouver une solution au cas où la variable à expliquer est binaire, nous avons mis en place une deuxième stratégie qui consiste à construire l'arbre SyrTree en prenant en entrée le résultat d'une classification automatique faite sur les données initiales. Dans ce cas la variable à expliquer est un histogramme. Cette stratégie se fait en cinq étapes :

1. Subdiviser les données initiales en deux sous-ensembles l'un pour l'apprentissage et l'autre pour le test.

2. Construire le fichier symbolique des individus d'apprentissage en utilisant le pattern construit pendant la phase de pré-traitement. Dans ce pattern il faut mettre le numéro de la ligne comme concept.

3. Lancer une classification automatique sur le fichier symbolique des individus en utilisant "ClustSyr" et en prenant toutes les variables sauf celle à expliquer qui doit être ajoutée comme variable supplémentaire. Nous avons utilisé la méthode de l'analyse de la courbe de pureté de différentes partitions pour déterminer le nombre de classes (voir Annexe 2.3 pour l'explication de cette méthode).

4. Utiliser "SyrTree" pour construire l'arbre de décision sur le fichier résultat de la classification.

5. Tester l'arbre construit sur les individus et les concepts d'apprentissage et de test.

4.3.2 Exemple illustratif sur les données UCI de la base " breast-cancer-wisconsin"

Clump_ Thickness	Uniformity_ of_Cell_Size	Uniformity_ of_Cell_Shape	...	Class	ligne
5	1	1	...	2	1
5	4	4	...	2	2
⋮	⋮	⋮	⋮	⋮	⋮
4	1	1	...	2	684

Tableau 3.21 – Extrait de la base de données "breast-cancer-wisconsin".

La base de données "breast-cancer-wisconsin" décrit les données recueillis à l'Université du Wisconsin sur le cancer du sein . Elle contient 684 individus. Chaque individu est décrit par 9 attributs et une variable binaire représentant classe à étudier. Nous avons ajouté une colonne représentant le numéro de la ligne qui sera utilisé comme classe d'individus pour construire les fichiers des individus sous format symbolique (.syr). Le tableau 3.21 représente un extrait des données initiales. Pour la construction d'un arbre symbolique à partir de ces données, nous avons appliqué la deuxième stratégie de construction d'arbre de SyrTree.

La première étape, consiste à appliquer un échantillonnage proportionnel par rapport à la variable à expliquer "Class" pour avoir les deux fichiers d'apprentissage et de test.

La deuxième étape concerne l'extraction des données symboliques décrivant les individus à partir des deux fichiers classiques. Pour cette étape nous avons mis la ligne comme concept et toutes les autres variables comme histogrammes. Cette conversion (voir figure 3.19), a été réalisée en utilisant TabSyr et le pattern suivant :

```
ligne :11 :c
Clump_Thickness :1 :h
Uniformity_of_Cell_Size :2 :h
Uniformity_of_Cell_Shape :3 :h
Marginal_Adhesion :4 :h
Single_Epithelial_Cell_Size :5 :h
Bare_Nuclei :6 :h
Bland_Chromatin :7 :h
Mitoses :9 :h
Class :10 :h
```

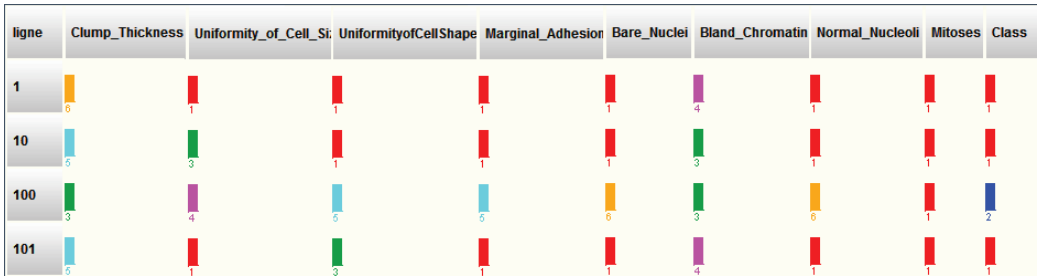


FIGURE 3.19 – Extrait du fichier symbolique décrivant les individus de la base "Breast Cancer".

Lors de la troisième étape, nous utilisons ***ClustSyr*** pour l'application de la méthode des nuées dynamiques symboliques. Pour cela nous avons fixé un intervalle de nombre de clusters qui est entre 10 et 100. Et nous avons utilisé la méthode de l'étude de pureté des clusters obtenus par rapport à la variable "Class" pour connaître le meilleur nombre de clusters qui est égal à 99. La figure 3.20 représente un extrait du fichier résultat de la classification des données en fixant à 99 le nombre de classes.

FIGURE 3.20 – Extrait du fichier symbolique résultat de l'application de "ClustSyr" sur les données "Brest-cancer" en fixant à 99 le nombre de clusters.

La quatrième étape concerne la construction de l'arbre SyrTree. L'application de SyrTree au fichier résultat de la classification en prenant "class" comme variable à expliquer et toutes les autres variables comme variables explicatives nous retourne l'arbre représenté par la figure 3.21.

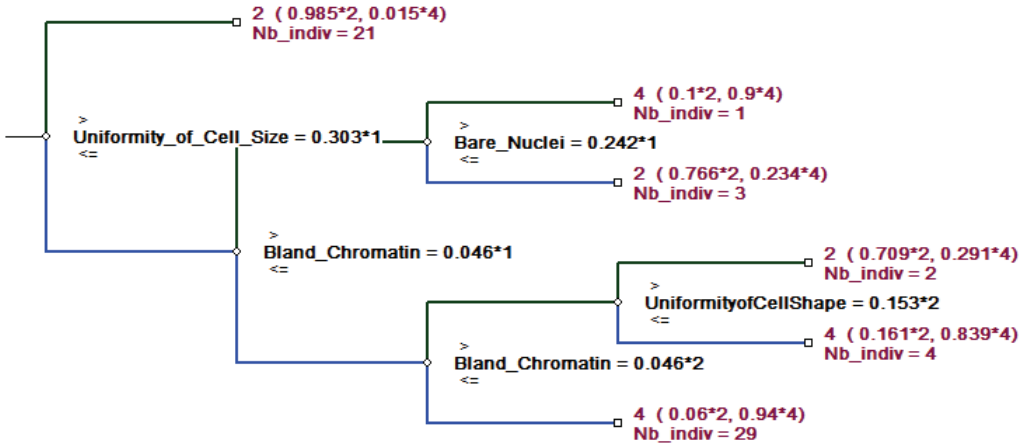


FIGURE 3.21 – Arbre de décision "SyrTree" appliqué sur le résultat de la classification de données "Breast-cancer".

Finalement, nous avons testé l'arbre obtenu sur les individus de test et sur les concepts de test. Le tableau 3.22 représente la matrice de confusion obtenue suite à ce test.

*** Matrice de confusion ****			
	2	4	Sum
2	41	3	44
4	2	22	24
Sum	43	25	68
*** Good Affectation rate = 92.647%			

Tableau 3.22 – Matrice de confusion résultat du test de l'arbre de la figure 3.21 sur les individus de test.

5 SyrTree Vs autres méthodes d'arbres de décisions

Afin de tester la fiabilité de notre méthode de construction d'arbre de décision nous avons comparé les résultats d'affectations des arbres de SyrTree (en utilisant les deux stratégies) avec ceux des méthodes C4.5[89], CART[20] et Stree[97].

Ces tests ont été réalisés sur des bases de données UCI [13] :

- "Abalone" : cette base est issue d'une étude réelle visant à prédire l'âge des ormeaux à partir de mesures physiques. Elle contient 4177 individus décrits par 8 attributs (7 continus et 1 nominal) et une variable de classe avec 3 modalités. (Il s'agit de la version où la variable à prédire est l'âge (young, adult, old) et non le nombre d'anneaux).
- "Breast Tissue" : cette base de données décrit différentes classes de tissus mammaires par différentes mesures d'impédance électrique. Elle contient 106 individus décrits par 9 attributs (continus) et une variable de classe avec 6 modalités.
- "Glass Identification" : Cette base de données contient la description de 6 types de verre en fonction de leurs teneurs en oxyde. Elle est composée de 214 individus. Chaque individu est décrit par 9 attributs de type continu et une variable classe qui peut prendre 6 valeurs différentes.
- "Seeds" : cette base décrit trois variétés de blé en utilisant des mesures des propriétés géométriques de leurs noyaux. Elle est composée de 210 individus décrits par 7 attributs (continus) et une variable de classe avec 3 modalités.
- "Breast-cancer-Wisconsin" : Cette base de données du cancer du sein a été recueillie par de l'Université du Wisconsin. Elle contient 684 individus. Chaque individu est décrit par 9 attributs de type nominal et une variable binaire représentant la classe.
- "Prima-indian-diabetes" : Cette base a été obtenue de l'Institut national du diabète et des maladies digestives et rénales. Elle décrit différentes caractéristiques de femmes qui sont diabétiques (class=1) ou pas (class=0). Elle est composée de 768 individus. Chaque individu est décrit par 8 attributs (continus) et une variable binaire représentant la classe.

- ”Ionosphere” : Cette base contient une classification des échos radar de l'ionosphère. Ces données radar ont été recueillies par un système Goose Bay, au Labrador. Cette base est composée de 351 individus. Chaque individu est décrit par 34 attributs (continus) et une variable binaire représentant la classe.

Sur chaque base de données nous avons procédé comme suit :

1. L'échantillonnage : nous avons généré cinq échantillonnages en variant le pourcentage de l'ensemble de test (10%, 15%, 20%, 25%, 30%).

2. Le test des méthodes classiques : pour les méthodes classiques (CART et C4.5), nous avons utilisé le logiciel Tanagra [91]. Chaque arbre construit sur les données d'apprentissage a été testé sur les individus d'apprentissage et de test.

3. Le test de la méthode STREE [97] : Afin de manipuler STREE, un fichier ”.sds” est obtenu à partir de chaque ”.csv” en utilisant l'outil ”csv2sds” qui donne un fichier ”.sds” contenant les variables classiques. Pour obtenir un fichier ”.sds” contenant des variables symboliques, nous avons appliqué la méthode DIV de STREE sur les différents fichiers d'apprentissage sous format ”.sds”.Nous nous sommes basés sur le nombre de classes suggéré par Seck [97]. Ensuite le résultat obtenu est enregistré sous le format ”.sds”. Ce dernier fichier est introduit dans STREE afin d'obtenir un arbre de décision symbolique. Enfin, le test de l'arbre obtenu est fait sur les ensembles d'apprentissage et de test. Nous notons ”STREE (CART)” les résultats de STREE classique et ”STREE (div)” ceux de STREE symbolique.

4. Le test de SyrTree : sur chaque fichier d'apprentissage nous avons appliqué les deux stratégies. Les arbres obtenus (notés ”SyrTree_strat1” pour la première stratégie, ”SyrTree_strat2” pour la deuxième stratégie, ”_conc” pour l'affectation des données symboliques et ”_indiv” pour l'affectation des individus classiques) ont été testés sur les fichiers classiques d'apprentissage, de test et sur les fichiers symboliques de test.

Tous les résultats sont représentés par les tableaux en Annexe 3. La figure 3.22 représente les taux d'erreurs moyens des différentes méthodes d'arbre de décision sur des données ayant une variable explicative non binaire. Ces taux d'erreurs ont été calculés sur les ensembles de test. D'après cette figure nous remarquons que

- Le taux d'erreur des arbres de SyrTree sur les données symboliques (notés SyrTree_strat1_conc ou SyrTree_strat2_con) est toujours meilleur que les autres méthodes.

- Aucune stratégie de SyrTree n'est toujours meilleure que l'autre. En effet, les arbres de SyrTree testés sur les individus issus de la première stratégie sont meilleurs que les arbres de la deuxième stratégie pour deux bases sur quatre qui sont ”Breast Tissue” et ”GlassIdentification”. Alors que les arbres testés sur les concepts issus de la première stratégie sont meilleurs que ceux de la deuxième stratégie pour trois bases sur quatre qui sont ”Abalone”, ”GlassIdentification” et ”Seeds”.

- Pour deux bases sur quatre, ” Breast Tissue” et ”Seeds”, nous avons plusieurs résultats test de SyrTree (au moins 3) qui sont meilleurs que les résultats des autres méthodes.

- L'arbre classique de STREE noté "STREE (CART)" est bien placé par rapport aux méthodes classiques CART et C4.5. Alors que l'arbre symbolique de STREE est toujours plus mauvais que les arbres de SyrTree.

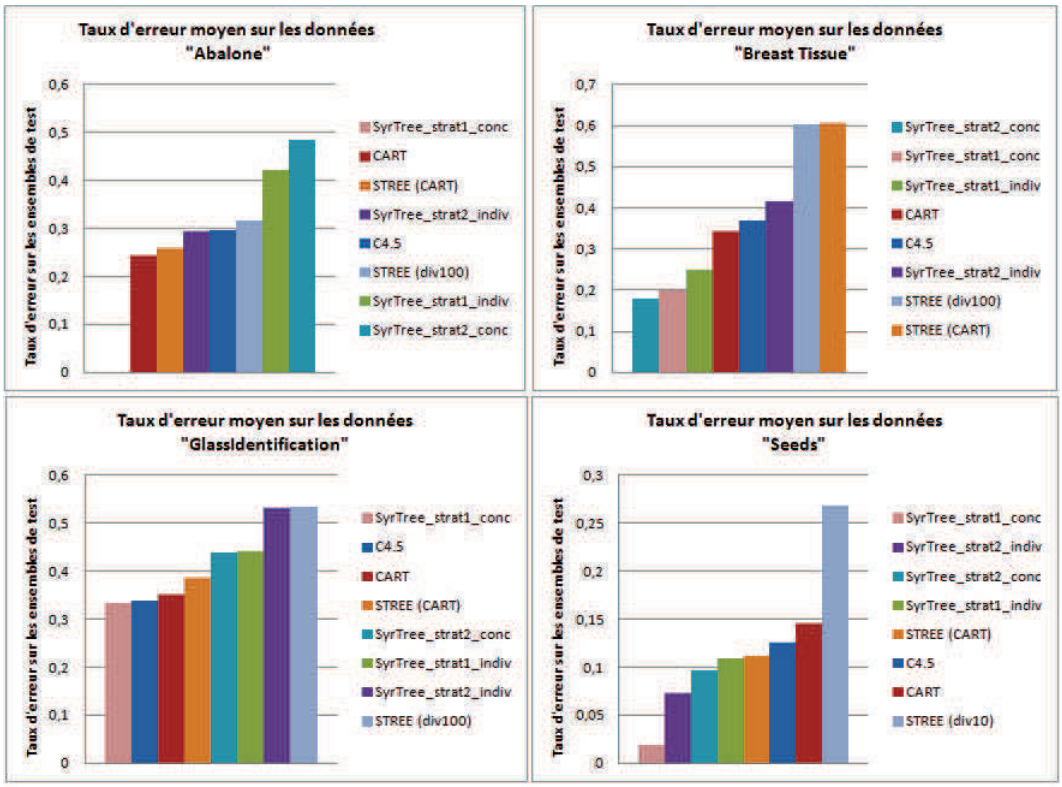


FIGURE 3.22 – Représentation des taux moyens d'erreurs d'affectation sur quatre bases UCI ayant une variable explicative non binaire.

La figure 3.23 illustre les taux d'erreurs moyens du test des différentes méthodes d'arbres de décision sur des bases de données ayant une variable à expliquer nominale. A partir de ces résultats nous constatons que :

- Les résultats de la deuxième stratégie de SyrTree pour l'affectation des individus sont meilleurs que ceux de la deuxième stratégie (sauf pour la base "prima indian diabete"). Ce qui justifie la mise en place de cette stratégie pour remédier aux problèmes d'avoir que des arbres sur les concepts surtout dans le cas où la variable à expliquer est binaire (en se basant sur la figure 3.23 pour deux bases sur trois nous avons les résultats de la première stratégie qui arrivent en dernière ou avant dernière position).

- Même si les résultats de SyrTree ne sont pas les meilleurs, à l'exception de la base "Prima Idian Diabete, ils sont très proches des résultats des autres méthodes d'arbre de décision.

- En comparant les résultats des deux stratégies de SyrTree à ceux de STREE sur les données symboliques, nous trouvons toujours au moins un des arbres de SyrTree qui est meilleur.

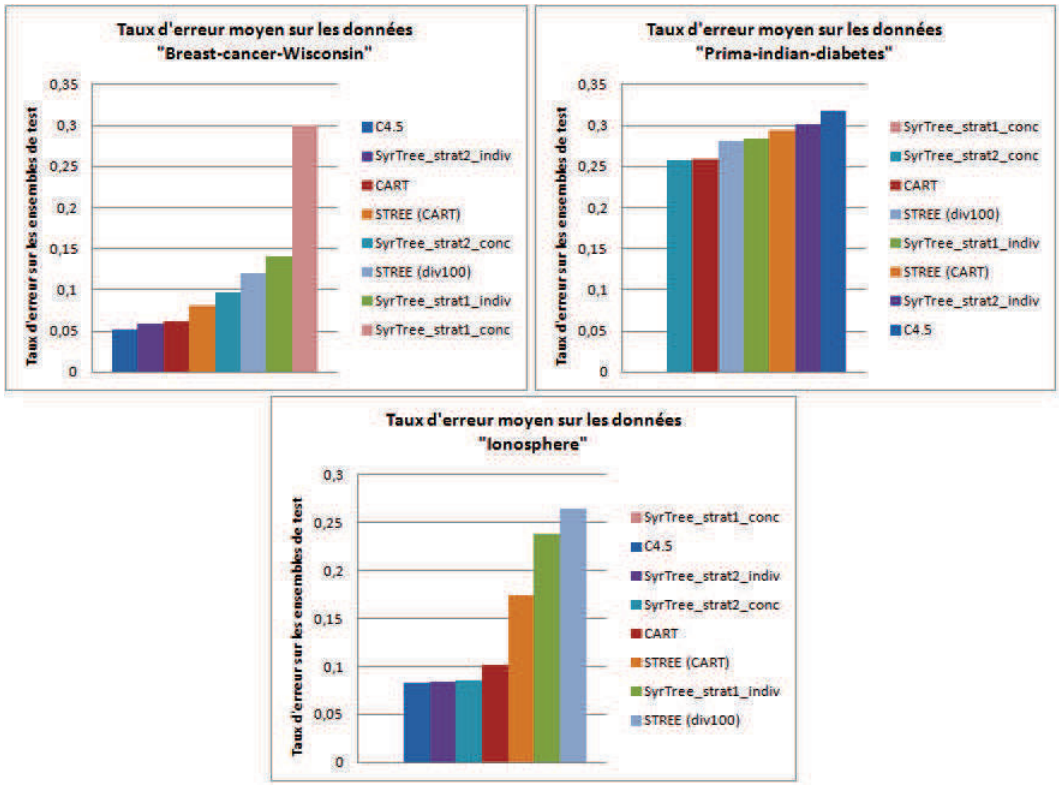


FIGURE 3.23 – Représentation des taux moyens d'erreurs d'affectation sur trois bases UCI ayant une variable explicative binaire.

En conclusion, nous pouvons dire que sur les données symboliques les arbres SyrTree ont été plus performants que ceux de STREE. Cependant par rapport aux méthodes classiques, dans certains cas SyrTree a été meilleur et dans d'autre non. Comme toutes méthodes d'analyse de données nous ne pouvons pas dire que SyrTree est la meilleure méthode d'arbre de décision, mais elle donne des résultats fiables et représente un sérieux concurrent aux autres méthodes.

6 Application de SyrTree sur des données réelles

6.1 Étude de l'influence des conditions environnementales sur les mesures de corrosion

Les données utilisées dans cette partie sont issues d'une étude de l'influence des conditions environnementales sur les mesures de corrosion. Le but étant de construire et de tester un arbre de décision décrivant des agressions subies par des prismes de béton (de valeurs T, G, I et C). Pour cela nous avons utilisé un ensemble d'apprentissage contenant les données des échéances de 1 à 6 et un ensemble de test qui représente les données de l'échéance 7. Les variables explicatives (Ecorr, Jcorr et Re) sont de type continu.

Afin d'utiliser SyrTree nous avons convertit les variables de type continu en histogrammes en utilisant "HistSyr" et en prenant la variable Agression comme classe d'individus. Ensuite nous avons utilisé le fichier obtenu comme fichier d'entrée à SyrTree. Suite au paramétrage

de la méthode de construction de l'arbre (le choix des variables explicatives et de la variable à expliquer), son exécution nous a donné l'arbre représenté sur la figure 3-27. A partir de cet arbre nous pouvons extraire une règle de décision pour chaque "Agression".

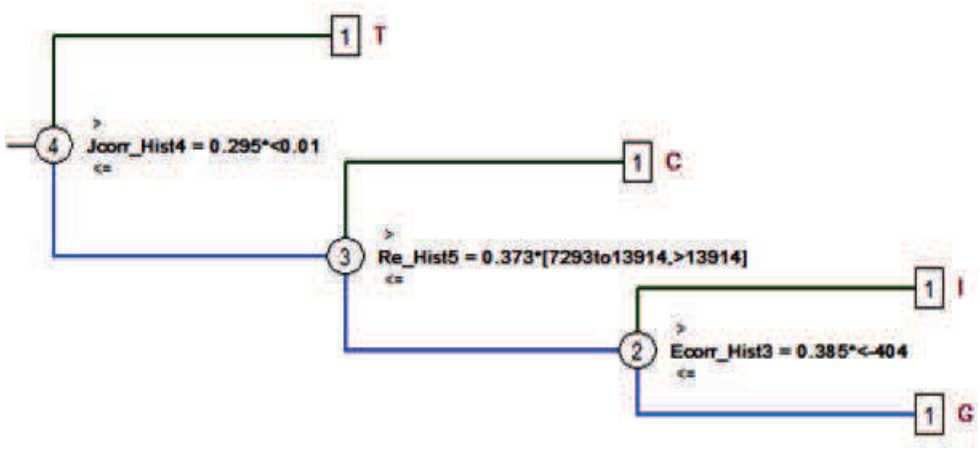


FIGURE 3.24 – Résultat de l'application de SyrTree sur les données de corrosion.

Afin d'évaluer cet arbre nous l'avons testé sur l'ensemble de test. Le tableau 3.23 représente le taux de bonne affectation et la matrice de confusion résultats de ce test. Cette matrice nous donne une idée sur la fiabilité de prédiction de notre arbre. En se basant sur cette matrice nous pouvons constater que l'arbre est performant pour les agressions de type "T" mais il est moins fiable pour les autres agressions. Par exemple pour les agressions de type "G" : 8 ont été affectées à "C" et 6 ont été affectées à "I", ce qui nous donne un taux de bonne affectation pour ce type d'agression inférieur à 50% (13/27).

***** Matrice de confusion ******					
	T	I	G	C	Sum
T	29	0	0	1	30
I	0	16	10	4	30
G	0	6	13	8	27
C	5	0	6	19	30
Sum	34	22	29	32	117
***** Good Affectation rate = 65.812%**					

Tableau 3.23 – Taux de bonne affectation et la matrice de confusion du test de l'arbre des "agressions" sur l'ensemble de test.

6.2 L'étude sur la dégradation des tours d'aéroréfrigérants d'EDF

Le module SyrTree a également été appliqué à des données issues d'une étude avec EDF-DTG. Cet exemple illustre l'intérêt de l'application de cette méthode prédictive au secteur du génie civil. Les données concernaient la surveillance de la dégradation des tours d'aéroréfrigérant des centrales nucléaires (déformation, tassement, fissures, zones de

corrosions). Ces données étaient hétérogènes par leurs sources, leurs formats, leurs volumes et leurs types de variables. L'analyse de données symboliques a permis alors de fusionner l'intégralité des données en un tableau unique afin d'analyser l'ensemble des phénomènes d'endommagement impactant les tours. Dans cette étude, les arbres de décision ont été utilisés pour la segmentation des ouvrages selon le niveau d'endommagement (voir par exemple la figure 3.25). Les règles obtenues sont des aides à la décision sur les priorités de maintenance puisque l'affectation d'un ouvrage à sa classe d'ouvrages permet de connaître les niveaux de dégradation.

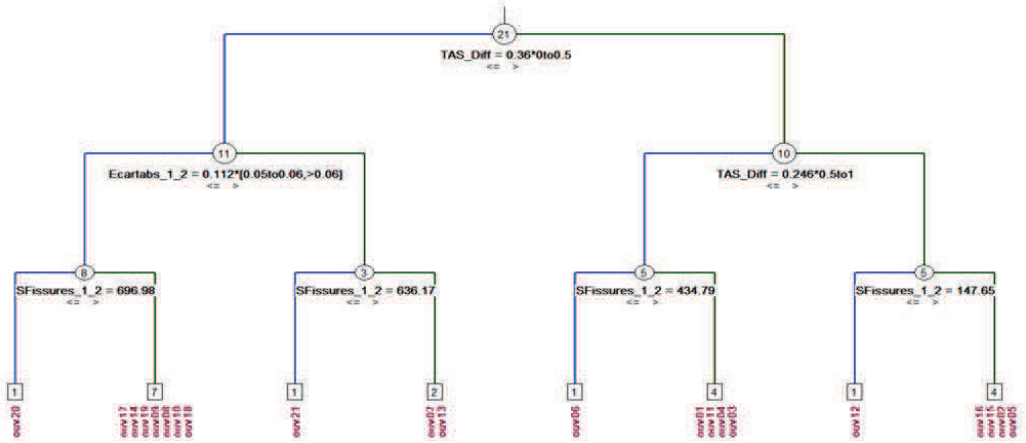


FIGURE 3.25 – Arbre de décision pour la segmentation d'ouvrages (tours d'aéroréfrigérants de centrales nucléaires) selon le niveau d'endommagement. Les règles obtenues sont des aides à la décision sur les priorités de maintenance puisque l'affectation d'un ouvrage à sa classe d'ouvrages permet de connaître les niveaux de dégradation.

7 Conclusion

Les arbres de décision représentent une méthode de data mining très connue et très utilisée par les "data miners". Ils doivent leurs succès à la facilité de leurs utilisations. Dans la littérature plusieurs travaux ont tenté d'étendre les arbres de décision aux données symboliques. Cependant la plupart de ces travaux ne traitaient pas différents types de variables explicatives et à expliquer. Pour remédier à ceci et afin d'intégrer au logiciel Syr une méthode prédictive nous avons mis en place "SyrTree". Cette méthode représente l'extension de CART [20] aux données symboliques. Tout au long de ce chapitre nous avons présenté les principes de notre méthode. En explicitant la façon dont nous traitons différents types de variable à expliquer et explicatives. Ensuite nous avons présenté les différentes stratégies de construction d'arbres de décision "SyrTree" et nous avons comparé notre méthode à d'autres méthodes d'arbres de décision classiques et symboliques. Enfin, nous avons montré l'intérêt de l'ajout de ce module au logiciel Syr en présentant son utilisation dans deux études effectuées au sein de Syrokko.

CHAPITRE 4

CloudHistSyr : extension d'HistSyr aux Big Data

1 Introduction

De nos jours la taille des données recueillies et susceptibles d'être analysées de plus en plus considérable. En effet, les "*data scientists*" sont amenés à manipuler de grandes bases de données appelées "*Big Data*". Ces données sont utilisées afin d'extraire des connaissances significatives et utiles. Étant donné que la plupart des méthodes de *data mining* classiques sont incapables de traiter de grands volumes de données (de l'ordre de To) plusieurs travaux ont été élaborés pour leurs extensions aux *Big Data*.

Parmi les solutions envisageables pour traiter ce type de données classiques, il y a celle basée sur leur conversion en données symboliques. En effet, il a été démontré que la conversion des données du classique au symbolique réduit considérablement la taille des données à étudier tout en gardant le maximum d'informations sur les données initiales (voir[14],[12] et le chapitre 1). Cependant, les outils d'ADS existants (voir chapitre 1 section 4) sont incapables d'extraire les données symboliques à partir des grandes bases de données. Pour cela et afin d'étendre l'ADS aux données scalables et distribués nous nous sommes intéressés à l'extension de la construction des données symboliques à partir des Big Data. Comme nous l'avons expliqué, dans les chapitres 1 et 2, l'étape la plus délicate pour l'extraction des données symbolique est la conversion des variables continues en histogrammes. Nous avons proposé une solution centralisée à cette problématique qui est la méthode "HistSyr" [45]. Cependant, cette solution est incapable de traiter des données dépassant quelques centaines de Ko. Nous avons alors mis en place un algorithme Map/Reduce permettant l'extension d'HistSyr aux Big Data. Cet algorithme est nommé "CloudHistSyr".

Ce chapitre est organisé comme suit : nous commençons par la présentation de l'état de l'art de l'extension des méthodes de *data mining* aux grandes masses de données et des notions de bases concernant Map/Reduce. Ensuite, nous présentons les composants Map/Reduce de notre méthode CloudHistSyr. Par la suite, nous exposons deux combinaisons de ces composants que nous avons testés pour mettre en œuvre la solution globale qui a permis d'étendre HistSyr aux Big Data. Enfin, nous présentons les résultats de CloudHistSyr sur une base de données réelles issue de l'étude du portique de Nantes.

2 Algorithmes distribués de data mining : état de l'art et présentation des principaux outils de programmation

2.1 État de l'art : Algorithmes distribués de data mining

Dans la littérature, nous trouvons plusieurs travaux qui ont traité l'extension des algorithmes d'analyse de données aux Big Data. Ces travaux peuvent être classés suivant leurs modèles de programmation ou suivant l'architecture matérielle qui a été utilisée pour leurs tests.

Concernant les modèles de programmation, nous trouvons principalement deux types de méthodes :

- Les méthodes qui utilisent des techniques classiques de parallélisation des algorithmes originaux [67] pour traiter les grandes bases de données. Parmi ces méthodes, nous

trouvons l'extension de la classification automatique [98, 110], des règles d'association [101], des SVM [103], etc.

- Les méthodes qui utilisent le modèle de programmation Map/Reduce [38, 70] pour traiter les grandes bases de données. Parmi ces méthodes nous trouvons l'algorithme Apriori [75], la classification automatique [112], la recherche des motifs fréquents [78] et l'extension de plusieurs méthodes (SVM, K-means, Naive Bayes, etc.) en utilisant MapReduce sur des machines multi-cœurs [32], etc.

Vu la variété des technologies de programmation pour étendre les méthodes de *data mining* aux grandes masses de données, nous trouvons dans la littérature diverses plateformes de test et de simulation de ces méthodes. Parmi ces architectures nous citons : les "Supers ordinateurs", les réseaux pairs à pairs ("P2P") et les nuages d'ordinateurs (le *"cloud computing"*). Dans la suite, nous présentons chacune de ces architectures en donnant quelques exemples de méthodes de *data mining* qui ont été testées dans chaque architecture.

- **Les "Super Computers"** : Ce sont des ordinateurs ayant des processeurs multi-cœurs très performants. Généralement, ils sont dotés de processeurs graphiques (GPU) beaucoup plus performants que les processeurs traditionnels (CPU). Les GPU sont équipés par des milliers de cœurs pouvant traiter des milliers de *threads* simultanément. Dans la plupart du temps, les méthodes de *data mining* ne sont pas entièrement exécutées sur les GPU. Parmi les méthodes qui ont utilisés les GPU pour accélérer leurs calculs nous trouvons : la recherche des motifs (*itemset*) fréquents [111], la méthode des k plus proches voisins [76], l'algorithme des k-means [112] et la méthode de discrétisation CAIM [24].

- **L'architecture P2P** : Dans ce cas les données sont distribuées entre les différents nœuds d'un réseau d'ordinateurs. Le but des méthodes distribuées de *data mining* implémentées pour les systèmes P2P est d'avoir le même résultat que l'algorithme centralisé sans bouger aucune données de son nœud initial [35]. Parmi ces méthodes nous trouvons : les arbres de décision [66, 9], la méthode des K-means[35] et la classification hiérarchique de documents [58].

- **Le Cloud** appelé aussi nuage d'ordinateurs : Il représente un ensemble d'ordinateurs distants communiquant par l'intermédiaire d'un réseau,généralement Internet. Il offre au client la possibilité de louer à la demande des ressources de stockage et de calcul très importantes à un coût négligeable comparé au coût de leurs achats. Dans la littérature plusieurs méthodes ont utilisé le cloud pour leurs tests sur de grandes masses de données. Parmi ces méthodes nous trouvons : la classification automatique [114], la méthode des K-means [113] et les règles d'association [74].

En étudiant les différentes extensions des méthodes de *data mining* pour traiter les grandes bases de données, nous avons remarqué que l'utilisation des *Super Computers* est très couteuse par rapport à celle du *Cloud*. Nous avons aussi remarqué que les méthodes utilisant l'architecture P2P donnent des résultats non optimaux. C'est pourquoi, nous avons choisi d'implémenter une solution qui sera testée sur le *Cloud*.

2.2 Outils de programmation

Dans cette section nous présentons les principaux environnements logiciels utilisés pour implémenter l'extension des méthodes de traitement et d'analyse aux *Big Data*.

2.2.1 Hadoop

Hadoop [1] [109] est un *Framework* java open source qui a été créé par Doug Cutting le créateur d'Apache Lucene. Il a pour objectif de faciliter la création et le test de méthodes scalables et distribuées. Il regroupe un ensemble de modules programmés en java destinés à faciliter la répartition et l'exécution des tâches sur plusieurs milliers de nœuds.
Malgré un code source écrit en java, n'importe quel langage de programmation peut utiliser Hadoop (comme Python, C++, etc).

Hadoop propose et gère son propre système de fichiers distribués *HDFS (Hadoop Distributed File System)*. Il offre l'implémentation d'un ensemble d'outils pour la manipulation et l'analyse de données de façon parallèle comme Map/Reduce [38], HBase [52], Hive [102] et Pig [2].

2.2.1.1 HDFS

HDFS [15] est un système de fichiers distribuées sur un ensemble de nœuds. Cette distribution est transparente pour l'utilisateur qui manipule ces données comme si elles étaient regroupées dans un seul fichier. HDFS prend en charge des données non structurées qui se présentent sous la forme de fichiers textes. Les avantages de HDFS sont :

- Le système gère la localisation des données lors de la répartition des tâches : un nœud donné recevra la tâche de traiter les données qu'il a pour réduire le temps de transfert de données.
- Il est *fault tolerant*, c'est à dire qu'il gère automatiquement la défaillance de ces nœuds. En effet les données sont répliquées sur plusieurs hôtes différentes pour assurer sa fiabilité.

2.2.1.2 Map/Reduce

Map/Reduce [73, 38] est un modèle de programmation qui permet le développement et le test de programmes dédiés à l'analyse des grandes masses de données distribuées sur un ensemble de nœuds. Son objectif est d'automatiser le parallélisme et la distribution du calcul sans exiger (de l'utilisateur) une supervision du système ni une expertise dans le calcul parallèle. Étant donné que le système gère l'exécution parallèle, la coordination et l'échec d'exécution des tâches, le rôle du programmeur est de définir et d'implémenter les deux fonctions Map et Reduce. La figure 4.1 [73] résume les étapes d'exécution d'un programme Map/Reduce qui sont :

1. La phase Map : chaque *Mapper* (nœud qui exécute la fonction Map) travaille sur un ou plusieurs morceaux des données initiales qui se trouvent dans son nœud. Suivant le

1. https ://hadoop.apache.org/
2. http ://pig.apache.org

traitement décrit par la fonction Map, les *Mappers* produisent des résultats sous formats de paires (clé, valeur).

2. Les paires (clé, valeur) produites par les différents *Mappers* sont regroupées et triées suivant leurs clés. Ensuite, elles sont dirigées vers les différents nœuds Reduce de façon que toutes les paires qui ont la même clé soient dans le même nœud Reduce.

3. Chaque *Reducer* traite les valeurs associées à chaque clé à la fois. Ce traitement est fixé par la fonction Reduce écrite par le programmeur.

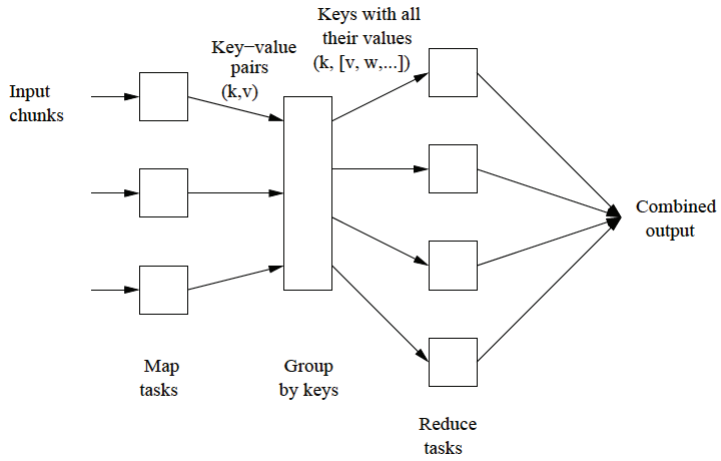


FIGURE 4.1 – Étapes d'exécution d'un algorithme Map/Reduce [73].

Pour résumer, la fonction Map doit décrire le traitement qui peut être exécuté sur des parties des données initiales indépendamment des résultats sur les autres parties. Alors que la fonction Reduce décrit comment agréger des résultats de la fonction Map pour atteindre l'objectif fixé.

2.2.1.3 HBase

HBase [52] est une base de données orientée "colonne" utilisant HDFS. Elle a été inspirée des publications de Google sur les BigTable [27]. HBase est capable de gérer d'énormes quantités de données. Elle permet de distribuer les données en utilisant le système de fichiers distribué HDFS d'Hadoop. Son fonctionnement repose sur le stockage distribué des données sur un cluster de machines physiques. Il permet de garantir la haute disponibilité et les hautes performances des bases. Les tables d'une base HBase peuvent être utilisées comme des entrées ou constituées des sorties pour les jobs Map/Reduce.

2.2.1.4 Hive

Hive [102] est une infrastructure Data Warehouse pour Hadoop. Elle offre un langage pour interroger une base Hadoop avec une syntaxe proche du SQL (*HiveQL (Hive query language)*). HiveQL offre la possibilité d'utiliser un sous ensemble de fonctions SQL comme les différents types de jointures, la fonction "Group by", les agrégations et la création de tables. Sa structuration des données est très claire grâce à l'utilisation de concepts comme les tables, les

colonnes et les lignes. Il accepte la plupart des types primitifs (les entiers, les réelles, les chaines de caractères) et les collections (comme les listes). Hive intègre un compilateur de requêtes HiveQl en les transformant en un graphe orienté de tâches Map/Reduce.

2.2.1.5 Pig

Il offre aux développeurs un langage de haut niveau dédié à l'analyse de gros volumes de données nommé *Pig Latin*. Il s'adresse aux programmeurs habitués à faire des scripts via Bash ou Python par exemple. Par ailleurs, Pig est extensible dans le sens où, si une fonction n'est pas disponible, il est possible de l'enrichir via des développements spécifiques dans un langage bas niveau (Java, Python, etc.). Les programmes écrits en Pig Latin sont analysés pour la vérification syntaxique. La sortie de cet analyseur est un plan logique représenté par un graphe orienté acyclique, permettant des optimisations logiques. Ce plan est compilé par un compilateur Map/Reduce et optimisé par un optimisateur de fonctions Map/Reduce. Le résultat est un programme Map/Reduce qui est transmis au gestionnaire de jobs Hadoop pour être exécuter.

2.2.2 Mahout

Mahout [86] est un projet de la fondation Apache. C'est une collection de méthodes programmées en java et destinées à l'apprentissage sur des architectures à mémoire distribuée. Mahout propose des programmes pour la recommandation utilisateur, la classification non supervisée par k-means, la régression logistique et les forêts aléatoires. Des compétences en Java sont indispensables pour utiliser les programmes proposés par Mahout.

2.2.3 Les librairies R

R [100] offre la possibilité de traiter des données massives en utilisant des librairies dédiées à la parallélisation des calculs et aux données massives. Parmi ces librairies nous trouvons :

- ***segue*** offre des méthodes dédiées à l'utilisation des services web d'Amazon (AWS) à partir de R.
- ***RHadoop*** : représente ensemble de librairies R d'interface avec Hadoop. Cet ensemble de librairies, développés sous licence libre, comprend :
  - *Rhdfs* : permet d'utiliser les commandes HDFS d'interrogation d'une base Hadoop à partir de R.
  - *rhbase* pour utiliser les commandes HBase d'interrogation.
  - *rmr2* permet l'exécution de jobs Map/Reduce à partir de R.

3 Composants Map/Reduce de CloudHistSyr

L'objectif de "CloudHistSyr" est d'étendre HistSyr aux *Big Data*. C'est-à-dire étant données : y, C et k. Où :

- $y = \{y_1, y_2, \ldots, y_n\}$ est une variable continue de n valeurs.
- $C = \{C_1, C_2, \ldots, C_n\}$ est une variable nominale qui représente la classe associée à chaque valeur de Y. Cette variable représentera la classe d'individus symbolique.
- k une valeur entière qui représente le nombre de modalités des histogrammes.

Notre but est de proposer une solution scalable et distribuée pour transformer y continue en histogrammes de k modalités. Ces histogrammes doivent être les plus discriminants pour les concepts.

En se basant sur les méthodes distribuées de *data mining* existantes et sur la décomposition de l'algorithme centralisé HistSyr, nous avons mis en place une méthode composée d'une succession d'algorithmes Map/Reduce. Ses principaux composants sont :

1. Un module de calcul des bornes possibles : il retourne à partir de toutes les valeurs d'une variable continue la liste des bornes frontières (voir chapitre 2). Ce module représente un algorithme Map/Reduce d'Hadoop qui a été testé en utilisant le service *Elastic Map Reduce (EMR)* d'Amazon Web Services (AWS).
2. Un module de calcul des occurrences d'un concept donné dans un intervalle défini issue d'une liste donnée. C'est un algorithme Map/Reduce qui a été testé en utilisant EMR.

3.1 Le module de calcul des bornes frontière

3.1.1 L'algorithme

L'objectif : Extraire à partir des différentes valeurs de y l'ensemble des bornes frontières. Une borne frontière est le milieu de deux valeurs frontières successives (voir la section 2.2.2.3 du chapitre 2).

Solution proposée : En utilisant Map/Reduce nous avons définit trois fonctions Map, Reduce et Combiner.

- *Fonction Map* : cette fonction sert à mettre les valeurs de la variable continue en clé et les valeurs des classes qui leurs sont associées en valeur.
  - Entrées : une partie des données initiales.
  - Traitement : pour chaque ligne des données initiales retourner la paire clé-valeur (y_i, C_i).
- Fonction Combiner : c'est un traitement qui sera exécuté au niveau des *Mappers* pour réduire le nombre de valeurs à transmettre au niveau du (des) *Reducer* (s).
  - Entrées : l'ensemble de clé-valeurs (y_i, C_i) résultats de la fonction Map.
  - Traitement : pour chaque valeur y_i nous allons regrouper les classes qui lui sont associée en supprimant la redondance. Les résultats seront de la forme $(y_i, \{C_{i1}, C_{i2}, \ldots, C_{ij}\})$ où tous éléments de la liste des valeurs sont différents.

- Fonction Reduce
  - Entrées : l'ensemble de clé-valeurs $(y_i, \{C_{i1}, C_{i2}, \ldots, C_{ij}\})$ envoyés par les *Mappers*.
  - Traitement : retourner les différentes valeurs de coupures possibles qui seront en clé des outputs associés à la valeur null.

N.B : Afin d'avoir les valeurs des y_i triées, nous exigeons l'utilisation d'un seul et unique *Reducer*. De cette façon tous les résultats des *Mappers* seront rassemblés dans un seul nœud de type *Reducer*.

3.1.2 Exemple d'application

Soit le fichier initial représenté par le tableau 4.1. Où la première colonne, représente la classe des espèces d'Iris qui peuvent prendre 3 valeurs différentes {setosa, versicolor et virginica}. Alors que la deuxième colonne représente les valeurs de la variable continue y.

C	y
setosa	5.1
setosa	4.9
setosa	4.9
versicolor	5.1
versicolor	6.4
versicolor	6.9
virginica	6.3
virginica	5.8
virginica	7.1

Tableau 4.1 – Extrait des données initiales des Iris de Fisher.

3.1.2.1 Traitement au niveau des *Mappers*

Dans cet exemple nous supposons que les données sont traitées par deux *Mappers*.

a. Résultats du premier *Mapper*

Exécution de la fonction Map : Cette fonction consiste à mettre les valeurs de la variable continue y en clés et ceux de la variable C en valeur. La deuxième colonne du tableau 4.2 représente les résultats de la fonction Map de chaque ligne.

Exécution de la fonction Combiner : Le traitement de la fonction combiner consiste à regrouper les valeurs suivant leurs clés et l'élimination des valeurs redondantes pour chaque clé. Par exemple les lignes 2 et 3 du tableau 4.2 ont la même clé 4.9 donc le combiner va regrouper ces deux lignes en une seule et renvoyer (4.9 setosa) au lieu d'envoyer 2 fois la même (clé, valeur) au *Reducer*.

Entrée Map	Résultats Map	Entrées Combiner	Résultats Combiner
setosa 5.1	5.1 setosa	4.9 (setosa, setosa)	4.9 setosa
setosa 4.9	4.9 setosa		
setosa 4.9	4.9 setosa	5.1 (setosa, versicolor)	5.1 (setosa, versicolor)
versicolor 5.1	5.1 versicolor		
versicolor 6.4	6.4 versicolor	6.4 versicolor	6.4 versicolor

Tableau 4.2 – Résumé du traitement effectué au niveau du premier *Mapper*.

La quatrième colonne du tableau 4.2 représente les résultats de la fonction Combiner. Ces champs seront envoyés depuis ce premier *Mapper* au *Reducer*.

b. Résultats du deuxième *Mapper*

De la même façon que dans le premier nœud *Mapper*, nous obtenons les résultats représentées par le tableau 4.3.

Entrée Map	Résultats Map	Entrées Combiner	Résultats Combiner
Versicolor 6.9	6.9 versicolor	5.8 virginica	5.8 virginica
Virginica 6.3	6.3 virginica	6.3 virginica	6.3 virginica
Virginica 5.8	5.8 virginica	6.9 versicolor	6.9 versicolor
Virginica 7.1	7.1 virginica	7.1 virginica	7.1 virginica

Tableau 4.3 – Résumé du traitement effectué au niveau du deuxième *Mapper*.

3.1.2.2 Traitement au niveau du *Reducer*

Pour ce module quelque soit le nombre de *Mappers* nous avons un seul *Reducer*. La fonction du *Reducer* est de déterminer si les valeurs de deux clés successives sont différentes ou pas. Si elles sont différentes, alors le programme mettra en output la moyenne de ces deux clés.

Entrées Reduce	Sortie Reduce
4.9 setosa	
5.1 setosa, versicolor	
5.8 virginica	5 null
6.3 virginica	5.45 null
6.4 versicolor	6.35 null
6.9 versicolor	7.05 null
7.1 virginica	

Tableau 4.4 – Résultat de la fonction Reduce.

Dans notre exemple nous avons les deux paires (5.8 setosa) et (5.1 (setosa, versicolor)) qui représentent deux clés successives dont les valeurs sont différentes. Ce cas générera une

première borne possible (de valeur 5 = (4.9+5.1)/2). Le tableau 4.4 illustre les résultats de l'exécution de la fonction Reduce sur notre exemple d'application.

3.2 Le module de calcul d'histogrammes

3.2.1 L'algorithme

Étant données une variable continue y, une liste de valeurs $I = \{i_1, i_2, \ldots, i_k\}$ et une variable nominale C. Cette dernière représente la classe des données symboliques. La liste **I** représente toutes les valeurs de coupures possibles résultant de l'exécution du premier module.

L'objectif : chercher pour chaque valeur de la variable y l'indice de l'intervalle auquel il appartient et calculer pour chaque classe l'occurrence de son appartenance aux différents intervalles.

Solution proposée : En utilisant Map/Reduce, nous avons défini les deux fonctions Map et Reduce :

- Fonction Map : retourne pour chaque valeur y_i de y la valeur de sa classe associée à l'indice de l'intervalle au quel elle appartient en clé et 1 en valeur.
  - Entrées : l'intervalle I et une partie des données initiales.
  - Traitement : pour chaque valeur y_i, retourner la paire clé-valeur ($C_i\_indice_i$, 1)
- Fonction Reduce : calcule les occurrences des $C_i\_indice_i$.
  - Entrées : l'ensemble de clé-valeurs ($C_i\_indice_i$, 1)
  - Traitement : pour chaque valeur $C_i\_indice_i$ nous calculons les occurrences.

N.B : Afin de réduire le nombre des champs échangés entre les **Mappers** et les *Reducers*, nous avons utilisé la fonction Reduce comme combiner.

3.2.2 Exemple d'application

Reprenons l'exemple du tableau 4.1 et le résultat du premier module (la liste des bornes frontières I = {5, 5.45, 6.35, 7.05}.
A partir de cette liste de valeurs, nous avons cinq intervalles possibles qui auront les indices suivant :

- $]-\infty, 5[$: indice 1.
- $[5, 5.45[$: indice 2.
- $[5.45, 6.35[$: indice 3.
- $[6.35, 7.05[$: indice 4.
- $[7.05, +\infty[$: indice 5.

3.2.2.1 Traitement au niveau des *Mappers*

Dans cet exemple nous supposons que les données sont traitées par deux *Mappers* différents.

a. Résultats du premier *Mapper*

Exécution de la fonction Map : L'application de la fonction Map sur notre exemple donne :

- Pour la première ligne nous avons : $5 \leq 5.1 < 5.45$, donc 5.1 appartient au deuxième intervalle. Par conséquence, $indice_{5.1} = 2$ et le *Mapper* retournera : $(setosa\_2 \quad 1)$
- pour la deuxième ligne nous avons : $4.9 < 5$ donc 4.9 appartient au premier intervalle. Par conséquence $indice_{4.9} = 1$ et le *Mapper* retournera : $(setosa\_1 \quad 1)$

La deuxième colonne du tableau 4.5 représente les résultats de la fonction Map de chaque ligne.

Entrée Map		Résultats Map		Entrées Combiner		Résultats Combiner	
setosa	5.1	setosa_2	1	setosa_2	1	setosa_2	1
setosa	4.9	setosa_1	1	setosa_1	(1, 1)	setosa_1	2
setosa	4.9	setosa_1	1				
versicolor	5.1	versicolor_2	1	versicolor_2	1	versicolor_2	1
versicolor	6.4	versicolor_4	1	versicolor_4	1	versicolor_4	1

Tableau 4.5 – Résumé du traitement effectué au niveau du premier *Mapper*.

Exécution de la fonction Combiner : Pour ce module, cette fonction a la même définition que la fonction Reduce. Dans chaque mapper elle aura comme rôle de regrouper les résultats de la fonction Map par clé et de calculer les occurrences de chaque clé. Par exemple, nous avons deux lignes qui ont la même clé setosa_1 (voir tableau 4.5) donc le combiner va regrouper ces lignes en une seule et renvoyer $(setosa\_1 \quad 2)$ au *Reducer*. La quatrième colonne du tableau 4.5 représente les résultats de la fonction Combiner. Ces champs seront envoyés depuis ce premier *Mapper* aux *Reducers*.

b. Résultats du deuxième *Mapper*

Entrée Map		Résultats Map		Entrées Combiner		Résultats Combiner	
versicolor	6.9	versicolor_4	1	versicolor_4	1	versicolor_4	1
virginica	6.3	virginica_3	1	virginica_3	(1, 1)	virginica_3	2
virginica	5.8	virginica_3	1				
virginica	7.1	virginicar_5	1	virginica_5	1	virginica_5	1

Tableau 4.6 – Résumé du traitement effectué au niveau du deuxième *Mapper*.

De la même façon que dans le premier nœud *Mapper* nous obtenons les résultats représentées par le tableau 4.6.

3.2.2.2 Traitement au niveau des *Reducers*

Pour cet exemple, nous supposons que nous avons un seul *Reducer*. Le traitement au niveau du *Reducer* sera le même que celui des Combiners. C'est-à-dire qu'il y aura un regroupement des valeurs suivant leurs clés et le calcul de la somme de ces valeurs pour chaque clé.
Dans notre exemple, la clé versicolor_3 a deux valeurs. Chacune est issue d'un *Mapper*. Dans ce cas, ces deux valeurs seront automatiquement regroupées dans une liste. Le traitement effectué consiste à calculer la somme des valeurs de chaque liste. La figure 4.2 illustre le traitement effectué au niveau du *Reducer*.

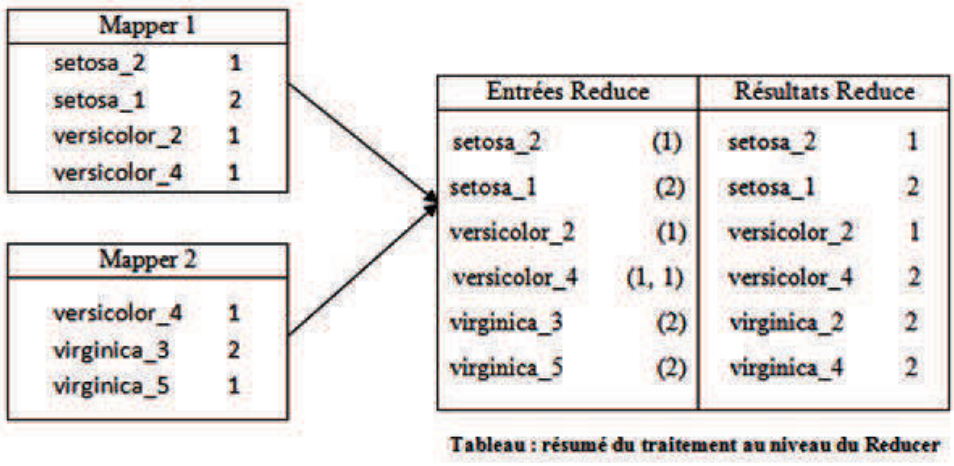


FIGURE 4.2 – Application du traitement au niveau des *Reducers* à un extrait des données de Fisher.

3.2.3 Étude de la complexité

Dans cette section, nous allons évaluer la complexité du deuxième module dans les deux cas : centralisé et distribué. Nous notons :

- N : nombre d'individus (nombre de valeurs de y) ;
- i : nombre d'intervalles, $i = nombre\_valeurs(I) + 1$;

3.2.3.1 Complexité dans le cas centralisé

Dans le cas où le calcul est fait sur une seule machine la complexité pour obtenir l'histogramme est de l'ordre de $Ni + N^2 \rightarrow \mathbf{O(Ni)} + \mathbf{O(N^2)}$

- Si $i \lll N$ donc $O(Ni)$ est négligeable par rapport à $O(N^2)$ donc nous pouvons conclure que l'algorithme est de complexité $O(N^2)$. Nous avons la même conclusion pour $i < N$.

- Si $i > N$: ce cas est aberrant. En effet, ce n'est pas logique d'avoir un nombre d'intervalles qui est supérieur au nombre de valeurs.

3.2.3.2 Complexité dans le cas distribué

En se référant aux différents travaux qui ont traités la complexité d'un algorithme Map/Reduce [73, 54], le temps de transmission dans ce type d'algorithme est plus important que le temps d'exécution des tâches Map et Reduce. Cependant, l'évaluation de la complexité d'un algorithme Map/Reduce n'a pas été standardisée et chaque papier donne ses propres expressions. Par exemple dans [73], les auteurs parlent de ***Replication rate*** qui représente le nombre de tous les clés-valeurs produits par les *Mappers* divisé par le nombre des entrées et de ***Reducer size*** qui représente la taille d'un *Reducer*. Alors que, dans [54], les auteurs parlent de ***key complexity*** et de ***Sequential complexity*** :

- *key complexity* est représenté par trois valeurs :
  - la taille maximale d'une paire clé-valeur produite ou introduite par un Mapper/Reducer ;
  - le temps maximal d'exécution pour un Mapper/Reducer pour une paire clé-valeur ;
  - et la mémoire maximale utilisé par un Mapper/Reducer pour produire une paire clé-valeur.
- *Sequential complexity* est représenté par deux valeurs :
  - la taille de toutes les paires clé-valeur produites par les *Mappers* et les *Reducers* ;
  - le temps total d'exécution de tous les *Mappers* et les *Reducers*.

L'évaluation de la complexité de notre algorithme donne :

- En utilisant la première approche :
  - *Replication rate* = N/N=1
  - *Reducer size* = $max(nc_i)$, avec nc_i c'est le nombre d'individus de la $i^{ème}$ classe.
- En utilisant la deuxième approche :
  - *Key Complexity* :
    - la taille maximale d'une paire clé-valeur = $max(nc_i)$, avec nc_i c'est le nombre d'individus de la $i^{ème}$ clé.
    - le temps maximal d'exécution = O(max(max(nci), i)) ;
    - et la mémoire maximale = O(max(max(nci), i)).
  - *Sequential complexity* :
    - la taille est de l'ordre de O(N ;
    - et le temps d'exécution est de l'ordre de O(Nk).

Suivant notre expérience, la deuxième approche de calcul de complexité d'un algorithme Map/Reduce donne meilleur idée sur les paramètres qui influencent le temps d'exécution d'un job. En effet, nous allons voir dans ce qui suit que le temps d'exécution des job de calcul d'histogrammes est proportionnel à la taille de la liste initiale des bornes.

3.2.4 Tests et étude de la scalabilité en utilisant Elastic Map Reduce (EMR) d'Amazon

Afin de tester notre programme, les données des Iris de Fisher ont été dupliquées 10000 fois. Ces données représentent 150 iris de trois classes (setosa, virginica et versicolor) décrites par une variable continue. Nous avons testé et évalué le temps d'exécution et la scalabilité du deuxième module (celui qui calcul les histogrammes) en introduisant une liste de bornes $I = \{5.45, 6.15\}$.

3.2.4.1 Test du module de calcul d'histogramme en utilisant EMR d'Amazon

Avant de commencer les tests, les données initiales ainsi que le .jar ont été chargés en utilisant "Amazon Simple Storage Service (S3)" d'Amazon. Ensuite, nous avons configuré un cluster EMR. Pour ce premier test, la plus petite configuration a été utilisée. La figure 4.3 représente le résumé de cette configuration avec un master (m1.small : ayant un processeur 64bits, 1.7Go de mémoire une faible performance réseau) et deux nœud avec la même configuration que le master.

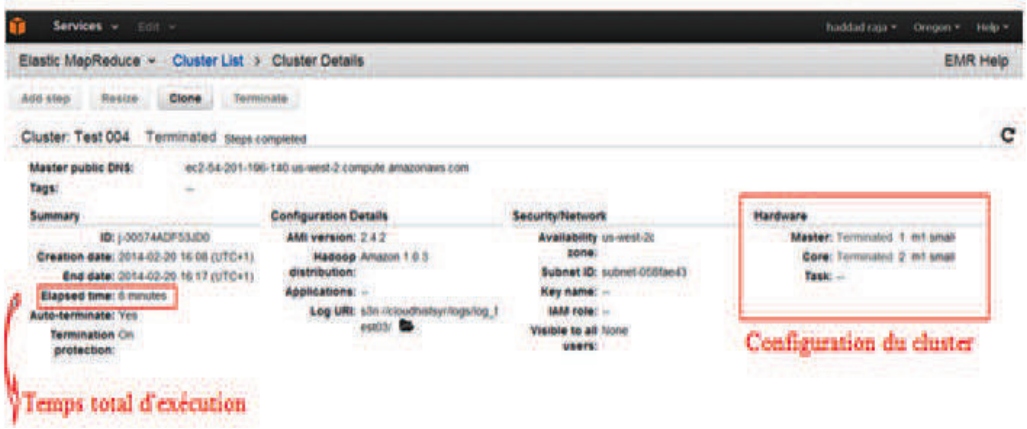


FIGURE 4.3 – Exemple de configuration d'un cluster EMR.

La préparation ainsi que la mise en place de l'environnement Hadoop sur le cluster et l'exécution du code de construction d'histogramme sur les données initiales ont été réalisé en 8 minutes (voir figure 4.3). Cependant, la compilation et l'exécution du code de construction des histogrammes n'ont nécessité que 3 minutes (voir figure 4.4).

Afin de connaitre le temps d'exécution du "Job", des instructions ont été ajoutés au code java permettant son calcul. Les résultats de ces instructions sont dans le fichier log "stdout". La figure 4.5 représente le contenu de ce fichier. Nous remarquons que le temps de configuration du Job est de $3919ms \approx 4secondes$ tandis que le temps d'exécution du job est de 141777ms = 2min 21sec 777ms.

Le Job de construction d'histogramme a été exécuté en utilisant 8 *Mappers* et 3 *Reducer* (voir figure 4.6). Ce découpage est géré automatiquement par EMR d'Amazon.

Sous le dossier des outputs, il y a création de trois fichiers générés par les trois *Reducers* (voir figure 4.7. Chaque fichier contient un sous ensemble de clés possibles associées à leurs valeurs. La fusion de ces trois fichiers est représentée par la partie gauche de la figure 4.8. Puisque les

FIGURE 4.4 – Détail du temps d'exécution des différentes étapes.

```
Start time configuration : 1392909181758
Finish time fin configuration: 1392909185677
Duration job configuration 3919
Start time job run : 1392909185677
Finish time run : 1392909327454
Duration job run 141777
```

FIGURE 4.5 – Contenu du fichier log "stdout".

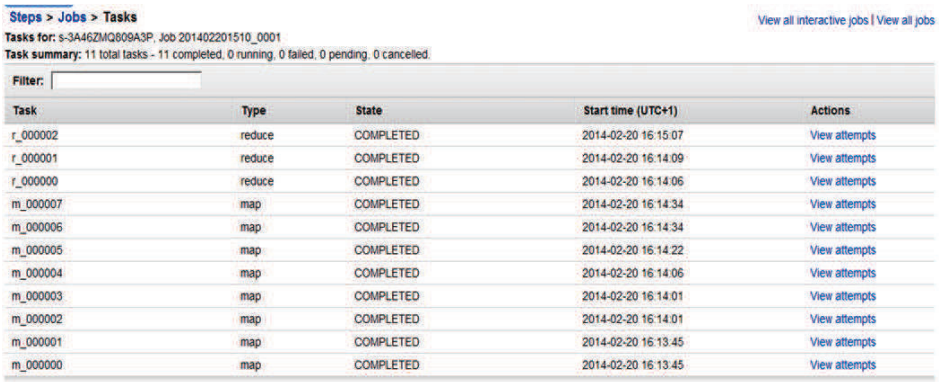


FIGURE 4.6 – Liste des tâches du job de construction d'histogramme.

données sont le résultat de la duplication des 150 iris par 10000, les résultats obtenus peuvent être vérifiées en utilisant les histogrammes obtenus à partir des 150 individus du fichier initial. La partie droite de la figure 4.8 représente les histogrammes sur les 150 individus. Le fait que les occurrences dans le grand fichier sont 10000 fois les occurrences dans le petit fichier prouve la justesse l'algorithme Map/Reduce de calcul d'histogrammes.

3.2.4.2 Étude de scalabilité de l'algorithme de calcul d'histogrammes

a. Données de test

Pour étudier la scalabilité de l'algorithme de calcul d'histogrammes, plusieurs duplications des données de Fisher ont été réalisées. Le tableau 4.7 résume la taille et le nombre de lignes de chaque fichier utilisé.

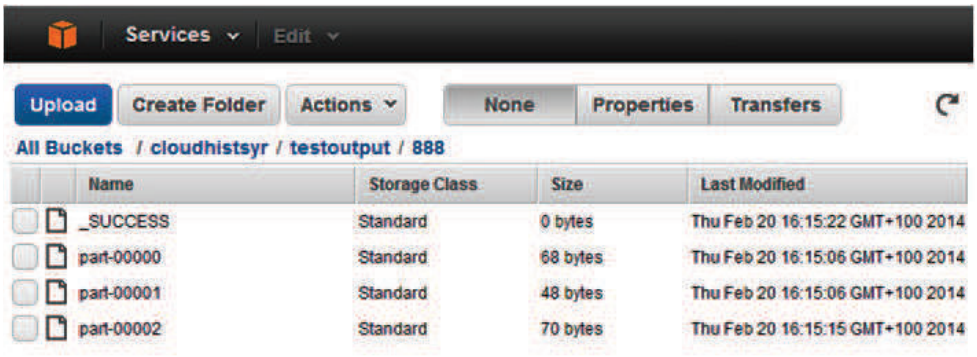


FIGURE 4.7 – Fichiers résultats du job.

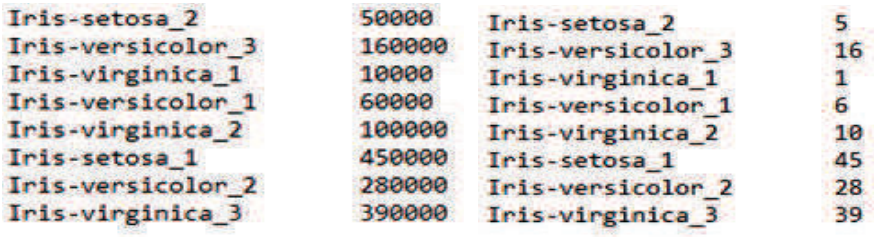


FIGURE 4.8 – Fusion des fichiers résultats de CloudHistSyr VS le résultat sur le fichier initial.

Taille données (Mo)	Nombre de lignes
27,2	1500000
271,6	15000000
543,2	30000000
1433,6	75000000
13926,4	750000000

Tableau 4.7 – Présentation de la taille et du nombre de lignes des données de test

b. Présentation des résultats

Afin d'étudier la scalabilité de l'algorithme, nous avons utilisé différentes configurations des nœuds du cluster EMR. Tous les résultats obtenus sont représentés dans l'annexe 3. Le tableau 4.8 présente les propriétés des différents composants utilisés dans ces configurations.

composant	processeur	mémoire	performances réseau
poste locale	32bits	1Go	
m1.small	32 bits	1.7Go	Faible
m1.large	64 bits	7.5Go	Modéré

Tableau 4.8 – Propriétés des composants utilisés dans les clusters EMR.

Temps total d'exécution en utilisant EMR Dans cette partie nous présentons le temps d'exécution de chaque cluster comprenant :

- Le temps d'allocation du cluster sur le cloud.
- Le temps de configuration d'Hadoop sur les différents nœuds.
- Le temps d'exécution de notre code.
- Le temps de terminaison du cluster.

Configuration	Temps d'allocation machines (minutes)	Temps de configuration d'Hadoop (seconds)	Temps de terminaison (minutes)
2m1small	4	39	3
4 m1small	5	38	4
6 m1small	4	37	4
8 m1small	3	32	3
16 m1small	4	36	2
2 m1Large	4	37	3
4 m1Large	4	60	2
6 m1Large	4	39	2
8 m1Large	4	37	2
16 m1Large	4	32	2

Tableau 4.9 – Présentation des temps d'allocation d'installation et de terminaisons des nœuds sur EMR d'Amazon.

Le tableau 4.9 représente les temps nécessaires à EMR afin d'allouer des nœuds, installer Hadoop sur ces derniers et les terminer. Nous remarquons que ce temps est presque invariable par rapport à la configuration choisie. Pour cela, afin d'évaluer notre programme Map/Reduce, nous ne prenons en compte que le temps d'exécution de notre code (voir la section suivante).

Étude du temps d'exécution du module de calcul d'histogrammes Dans cette partie, nous étudions le temps d'exécution des jobs Map/Reduce que nous avons définit. Pour cela nous commençons par l'étude du temps d'exécution en augmentant la taille des données et en utilisant différentes configurations.
La figure 4.9 présente les différentes courbes du temps d'exécution en fonction de la taille des données suivant différentes configurations. Nous remarquons que :

- L'allure de la courbe représentant l'algorithme centralisé est différente des autres courbes.
- L'augmentation du nombre de nœuds du cluster améliore le temps d'exécution.

Nous pouvons conclure que notre algorithme est scalable. Cependant, pour des données de petites tailles $< 1Go$ l'algorithme centralisé donne un temps d'exécution meilleur que celui de l'algorithme Map/Reduce. Ceci peut être expliqué par la perte de temps pendant la transmission des données entre les différents nœuds du cluster. Par ailleurs, pour un volume de

données important (13 Go), la solution Map/Reduce est meilleure que celle centralisée quelque soit la configuration utilisée.

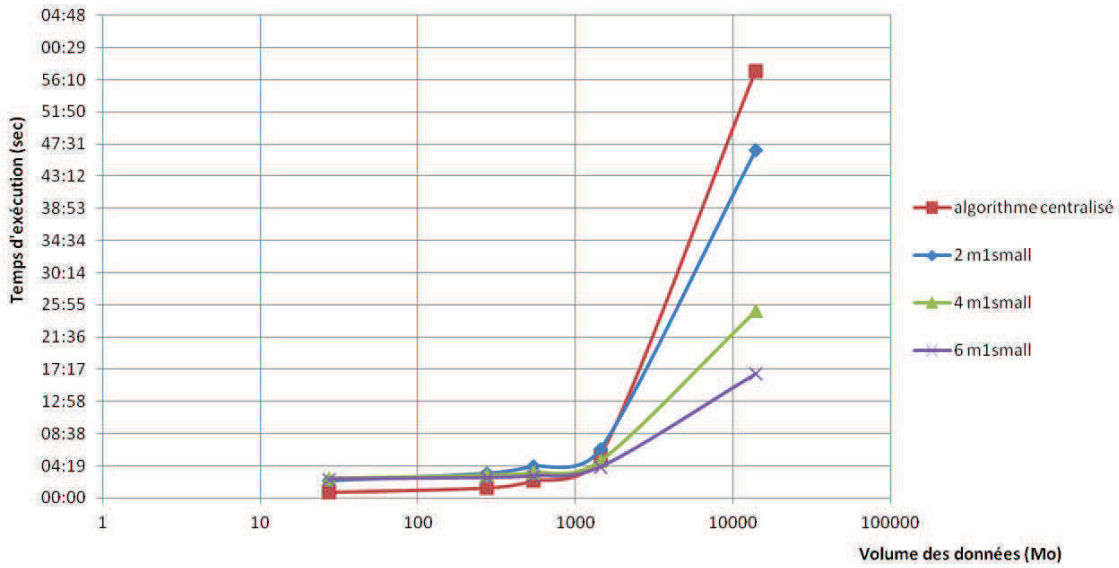


FIGURE 4.9 – Courbes des temps d'exécution en fonction de la taille des données, suivant différentes configurations.

Afin d'approfondir cette étude, les données les plus volumineuses (13,6 Go) ont été traitées en variant les configurations. La figure 4.10 présente les différentes courbes du temps d'exécution en fonction du nombre des nœuds du cluster. Nous alors constater que :

- Quelque soit les propriétés des nœuds du cluster utilisé le temps d'exécution diminue en augmentant le nombre de nœuds.
- Pour un nombre similaire de nœuds dans le cluster plus les paramètres des machines allouées sont meilleurs plus le temps d'exécution est meilleur.

4 CloudHistSyr : Implémentation et tests du programme global

Le but de CloudHistSyr est de trouver les meilleures bornes qui donnent les histogrammes les plus discriminants pour les classes d'individus symboliques. Afin d'atteindre ce but nous avons testé deux approches :

1. La première consiste à calculer toutes les combinaisons de k bornes parmi la liste des bornes possibles obtenues suite au lancement du Job "Calcul borne". Ce qui implique le lancement du job "Calcul histogramme" C_i^k fois.
2. La deuxième permet de calculer les histogrammes par rapport à la liste totale des bornes possibles obtenues comme résultat du module "Calcul borne". Ceci en laçant une seule fois le Job "Calcul Histogramme". Le résultat de ce dernier Job sera introduit à un

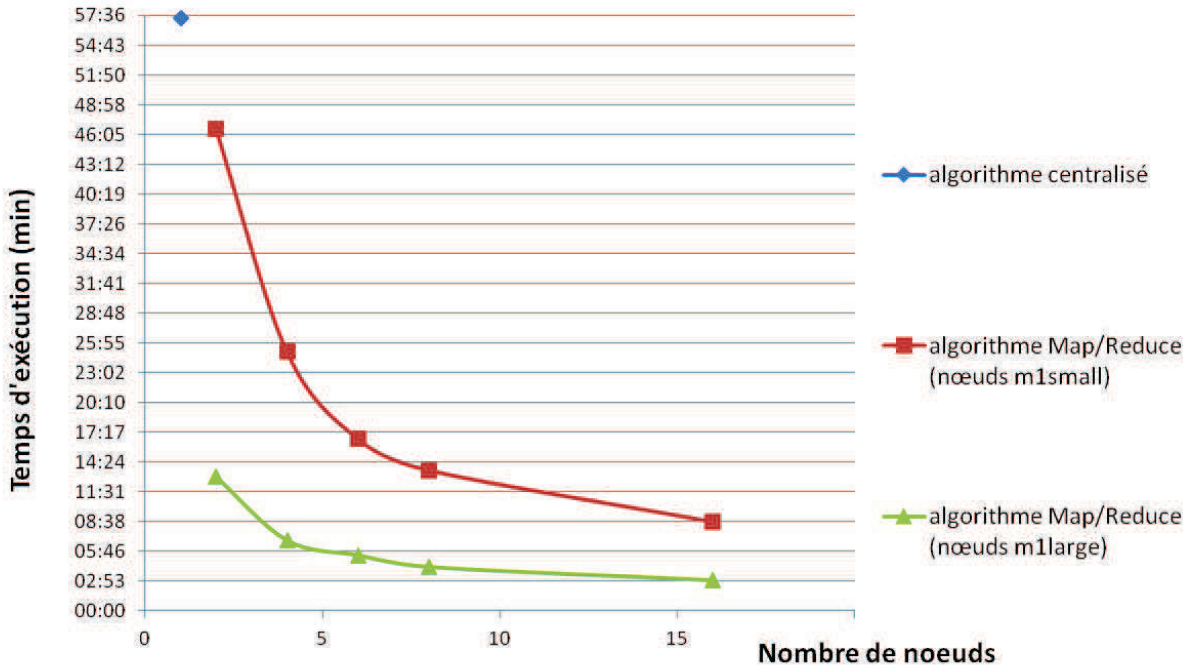


FIGURE 4.10 – Courbes des temps d'exécution de l'algorithme de calcul d'histogrammes (appliqué aux données de 13Go) en fonction du nombre de nœuds d'un cluster EMR.

programme local qui permettra la recherche des histogrammes qui discriminent mieux nos concepts.

4.1 Première approche : lancement du job "Calcul histogramme" C_i^k fois

Dans cette approche, résumée par la figure 4.11, l'idée était de :

1. Lancer le module de calcul de bornes possibles qui retournera la liste de bornes frontières, notée I (où $card(I) = i$).

2. A partir de la liste I, calculer toutes sous listes de taille k.

3. Pour chaque sous liste obtenue, lancer un job de "calcul histogramme".

4. Comparer les C_i^k résultats obtenus pour chercher les meilleurs histogrammes qui optimisent le score d'HistSyr (voir équation 2.4).

Pour le lancement des C_i^k jobs "Calcul Histogramme", deux cas se présentent : le premier est de les lancer l'un après l'autre de façon séquentielle alors que le deuxième est de les lancer de façon parallèle.

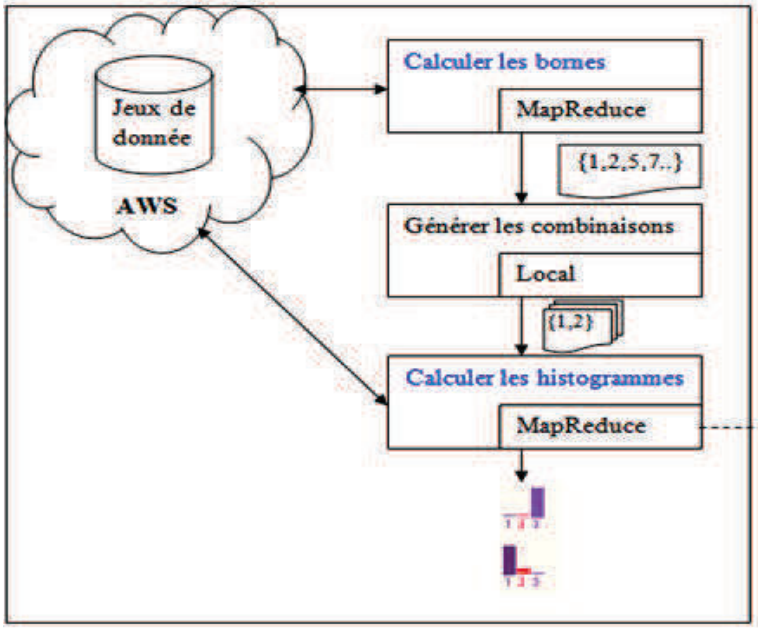


FIGURE 4.11 – Composition de CloudHistSyr suivant la première approche ;

4.1.1 Test du lancement séquentiel de jobs

Nous avons lancé les jobs de calcul d'histogramme d'une façon séquentielle sur un même cluster EMR d'AWS. Les données de test sont les données de Fisher. Nous avons donc 22 valeurs de coupure possibles. Par conséquent, nous devons lancer $C_{22}^{3} = 1540$ jobs.
Résultats : Nous avons arrêté manuellement l'exécution puisqu'au bout de 41 minutes il n'y a eu que 18 jobs exécutés sur 1540 jobs à tester. La figure 4.12 illustre le temps de début de chaque job.
Conclusion : Le lancement séquentiel de jobs sur un même cluster n'est pas performant. En effet tous ce que nous gagnons avec cette méthode c'est le temps d'allocation des différents nœuds du cluster qui restent allouées pour l'exécution de nos jobs.

4.1.2 Le test du lancement parallèle de jobs sur différents clusters

Dans cette partie nous avons testé le lancement de plusieurs clusters de façon instantanée où chaque cluster prendra en charge l'exécution d'un job en ayant une liste de bornes différente des autres.
Expérimentation : Étant donné la limitation du nombre d'instances à 20 instances (qui peuvent être augmentées sur demande), nous avons testé le lancement parallèle de 7 clusters. Chaque cluster est constitué de deux nœuds.
Résultats : Le tableau 4.10 résume les résultats de cette simulation.
Interprétation : Nous remarquons que l'exécution des 7 jobs a été faite presque au même temps. Il y a un décalage de quelques secondes entre les différents jobs qui est lié à l'allocation des nœuds des différents clusters.
Conclusion : Le lancement parallèle des jobs est beaucoup plus intéressant que le lancement séquentiel. Cependant, cette solution reste très couteuse.

Job	State	Start time (UTC+2)
job_201404021042_0018	COMPLETED	2014-04-02 13:26
job_201404021042_0017	COMPLETED	2014-04-02 13:24
job_201404021042_0016	COMPLETED	2014-04-02 13:21
job_201404021042_0015	COMPLETED	2014-04-02 13:19
job_201404021042_0014	COMPLETED	2014-04-02 13:17
job_201404021042_0013	COMPLETED	2014-04-02 13:14
job_201404021042_0012	COMPLETED	2014-04-02 13:11
job_201404021042_0011	COMPLETED	2014-04-02 13:09
job_201404021042_0010	COMPLETED	2014-04-02 13:06
job_201404021042_0009	COMPLETED	2014-04-02 13:04
job_201404021042_0008	COMPLETED	2014-04-02 13:02
job_201404021042_0007	COMPLETED	2014-04-02 12:59
job_201404021042_0006	COMPLETED	2014-04-02 12:56
job_201404021042_0005	COMPLETED	2014-04-02 12:54
job_201404021042_0004	COMPLETED	2014-04-02 12:52
job_201404021042_0003	COMPLETED	2014-04-02 12:49
job_201404021042_0002	COMPLETED	2014-04-02 12:47
job_201404021042_0001	COMPLETED	2014-04-02 12:45

FIGURE 4.12 – Lancement séquentiel des jobs de calcul d'histogrammes.

Job	Heur de début du job	Durée (minutes)
[5.05, 5.95, 6.35]	13 :04 :19	02 :23
[5.25, 5.45, 6.25]	13 :04 :25	02 :18 :104
[4.85, 5.25, 5.95]	13 :04 :37	02 :18 :785
[5.25, 5.65, 6.35]	13 :04 :38	02 :29 :904
[4.85, 5.85, 6.35]	13 :04 :39	02 :24 :526
[5.25, 5.95, 6.35]	13 :04 :46	02 :19 :653
[5.15, 5.45, 6.35]	13 :05 :04	02 :21 :493

Tableau 4.10 – Résultats du lancement parallèle de 7 clusters de deux noeuds chacun pour le calcul des histogrammes.

4.1.3 Avantages et inconvénients de la première approche

Le lancement d'autant de combinaisons que de job de calcul d'histogrammes a comme avantage de garantir un résultat optimal.

Cependant, cette approche est très couteuse. Puis, même après le lancement des C_i^k jobs de calcul d'histogrammes nous n'obtenons pas directement la solution qui optimise notre score. Il faut ajouter un petit module qui évalue ces solutions pour renvoyer celle qui maximise le score.

4.2 Deuxième approche : lancer le "Calcul Histogramme" en utilisant toutes les bornes possibles

Dans cette approche (représentée par la figure 4.13), nous traitons la problématique de recherche des histogrammes les plus discriminants de cette façon :

1. Lancer le module de calcul des bornes possibles : ce module retourne à partir de toutes les valeurs d'une variable continue la liste de valeurs frontières.

2. Lancer le module de calcul d'histogrammes en prenant la liste entière (résultat de la première étape) comme liste de bornes.

3. Utiliser un module de calcul des histogrammes de k modalités qui discriminent le mieux les différents concepts. Cet algorithme est l'adaptation de l'algorithme de Fisher afin qu'il prenne en entrée le résultat du deuxième module au lieu de travailler sur les données initiales brutes.

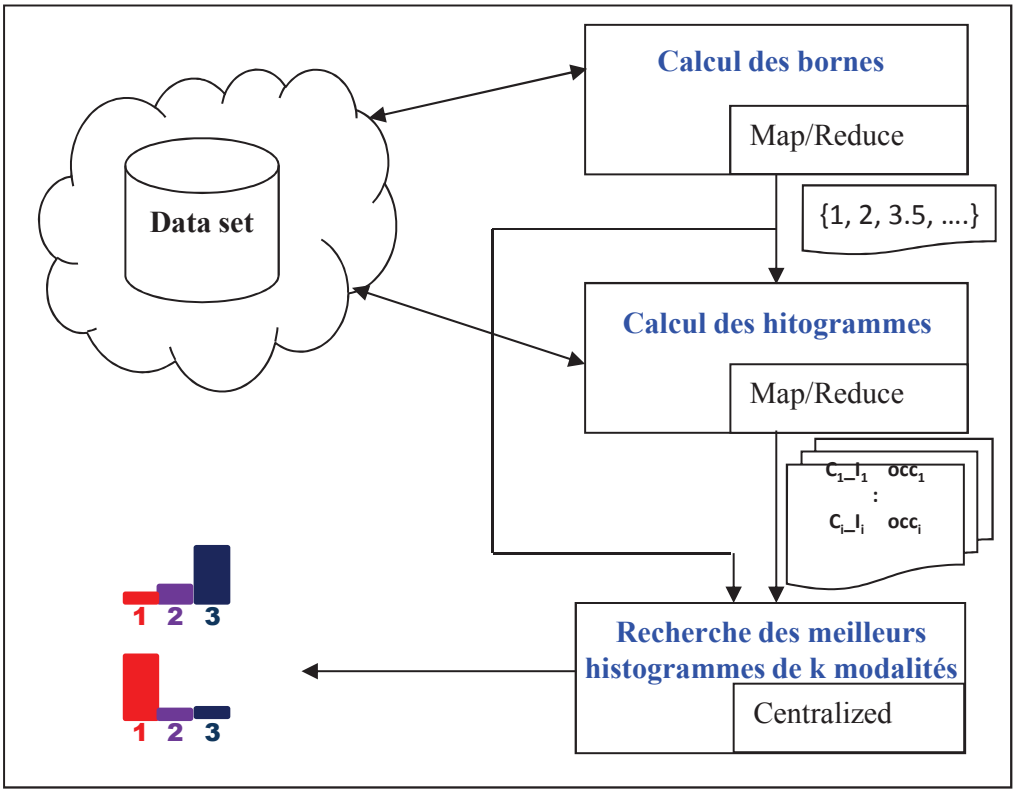


FIGURE 4.13 – Composition de CloudHistSyr suivant la deuxième approche.

4.2.1 Test de la deuxième approche sur les données des Iris

Pour tester cette deuxième approche nous avons utilisé les données des iris Fisher (de taille 13.6Go) avec un cluster EMR composé d'un master et de 16 nœuds de types m3.xlarge. L'exécution des deux jobs "Calcul bornes" et "Calcul Histogrammes " a nécessité presque 7 minutes. La figure 4.14 illustre les détails de l'exécution du job sur EMR. D'après cette figure :

- Le premier module a été exécuté au bout de 281.150 sec = 4min 42 sec .
- La liste des bornes I = {4.85 ; 4.95 ; 5.05 ; 5.15 ; 5.25 ; 5.35 ; 5.45 ; 5.55 ; 5.65 ; 5.75 ; 5.85 ; 5.95 ; 6.05 ; 6.15 ; 6.25 ; 6.35 ; 6.45 ; 6.55 ; 6.65 ; 6.75 ; 6.85 ; 6.95 ; 7.05}

```
 Etape 1 :extraction des valeurs frontieres
Start time job val front : 1421797942665
Finish time job val front : 1421798223815
Duration job val front 281150
Finished Job val front
================================
 liste
4.85;4.95;5.05;5.15;5.25;5.35;5.45;5.55;5.65;5.75;5.85;5.95;6.05;6.1
5;6.25;6.35;6.45;6.55;6.65;6.75;6.85;6.95;7.05;
We are in main debug tous
Start time job calcul histo : 1421798223945
Finish job time calcul histo : 1421798371324
Duration job calcul histo : 147379
Finished Job calcul histo :
================================
```

FIGURE 4.14 – Temps d'exécution des deux jobs de calcul des bornes suivi du calcul d'histogrammes.

- Le deuxième module a été exécuté au bout de 147.379 sec = 2 min 27 sec.

Les fichiers résultats du module de calcul d'histogrammes sont regroupés dans un même fichier. Ce dernier sera introduit avec le fichier contenant les bornes possibles au programme centralisé de recherche du meilleur histogramme. Ce programme retourne alors les histogrammes de 3 modalités les plus discriminants en 94 ms. L'ensemble des fichiers générés par ce programme sont représentés dans la figure 4.15.

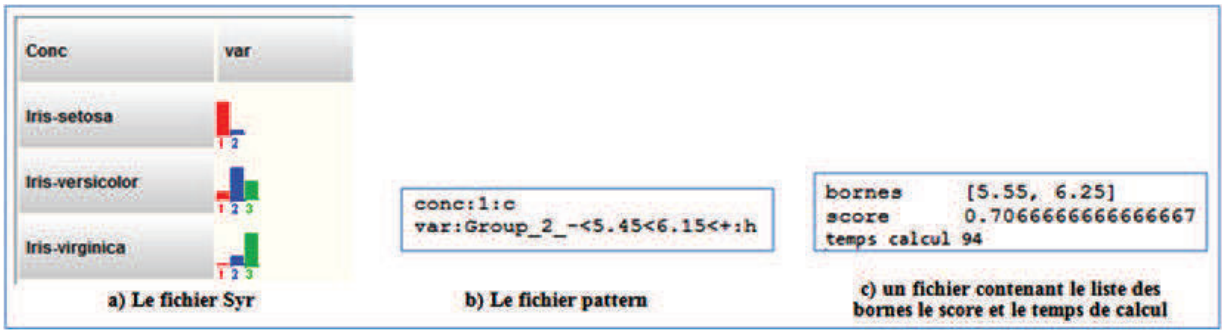


FIGURE 4.15 – Les fichiers résultats du module de recherches des histogrammes les plus discriminants.

NB : Ces résultats sont identiques à celles obtenues en lançant HistSyr sur la base de données initiale de 150 Iris. Ce qui permet de valider la justesse de l'ensemble des modules constituants CloudHistSyr.

4.2.2 Avantages et inconvénients de la deuxième approche

Avantages :

- Cette solution est moins couteuse que la première.

- Les résultats des jobs Map/Reduce sont réutilisables puisqu'ils permettent : de calculer la liste des bornes frontières et de résumer les données initiales par rapport à cette liste. Ce qui rend possible de changer le nombre des modalités des histogrammes sans refaire les tests dans le Cloud.

Inconvénients :

- Le temps d'exécution du deuxième module peut augmenter de façon considérable si le nombre de bornes possibles augmente (voir section 5).
- Le dernier composant de cette solution (l'algorithme centralisé de recherche des meilleurs bornes de découpages), peut avoir des limites avec des données ayant plusieurs milliers de bornes possibles.

La problématique du temps d'exécution du deuxième module peut être résolue soit en augmentant le nombre de nœuds dans le cluster, soit en lançant d'une façon parallèle des jobs de calcul d'histogrammes prenant des sous liste de la liste initiale des bornes. Ce qui reviendrait à subdiviser la liste des bornes et lancer un nombre négligeable de clusters en parallèle comparé à ce qui est proposé dans la première approche.

5 Application de CloudHistSyr sur les données réelles du portique de Nantes

5.1 Présentation des données initiales

Les données ont été recueillies dans le temps par 21 capteurs installés sur un portique d'autoroute « test ». Les 21 capteurs peuvent être divisés en deux groupes différents :

- 11 capteurs de changement d'état dont :
  - 3 accéléromètres ACC1, ACC2, ACC3 ;
  - 2 inclinomètres INC1, INC2
  - 6 jauges de déformation GAG1 à GAG6
- 1 station météo composée de 10 capteurs dont :
  - 5 capteurs de pression PR1 à PR5
  - 2 capteurs de températures TEMP1 et TEMP2

Trois états différents ont été simulés chronologiquement "Avant", "Pendant" et "Après" :

- Un premier état sans défaut = Avant le 20 février 2013
- Un second état dû à l'ajout de deux masses sur le portique. L'ajout de ces deux masses est considéré équivalent à un défaut = Pendant (entre le 21 février le 03 mars 2013)
- Un troisième état après le retrait des masses = Après le 4 mars 2013

Les données considérées ont été enregistrées entre le 19 décembre 2012 et le 15 avril 2013. Durant cette période, plusieurs enregistrements sont effectués chaque jour durant une durée limitée (le nombre d'enregistrements quotidiens et leurs durées ne sont pas toujours les mêmes sur la période étudiée). Ces enregistrements sont appelés "évènements dynamiques". Par exemple, "Event_2013-01-15_08" désigne l'évènement du 15 janvier 2013 numéro 8. Nous avons au total 2100 évènements dynamiques. Chaque évènement dynamique produit un fichier de données recueillant les mesures à chaque instant. Dans ce fichier, les lignes sont des instants et les colonnes représentent les valeurs relatives à chacun des 21 capteurs (changement d'état + station météo).

5.2 Présentation des tests en utilisant CloudHistSyr

Afin de pouvoir exécuter notre méthode sur les données de Sipris, nous avons

1. Mis en forme les données initiales en ajoutant à chaque ligne la variable "Etat" qui représente le concept à étudier tout au long de nos simulations.
2. Lancé l'algorithme de recherche de bornes possible pour les différents capteurs.
3. Lancé l'algorithme de construction d'histogrammes pour chaque capteur.

5.2.1 Mise en forme des données initiales

En mettant en forme les données initiales, nous obtenons des fichiers de test de l'ordre 1.75 Go pour chaque variable (acc1, acc2, etc.). Ces fichiers sont constitués de plus que 139 millions de lignes. Chaque ligne contient deux valeurs. Le tableau 4.11 représente un extrait des données décrivant les valeurs du premier accéléromètre.

	Etat	Acc1
1	Avant	3.5
2	Pendant	5
⋮	⋮	⋮
139358263	Après	-6.024

Tableau 4.11 – Extrait du fichier initial des données de Sipris représentant l'accéléromètre 1.

5.2.2 Résultats du module de calcul des bornes

Nous avons fait le test en utilisant EMR d'Amazon avec un cluster composé d'un un master et 8 nœuds (de types m3.XLarge).
Nous avons lancé le programme de calcul de bornes possibles afin d'estimer le coût total du test. Au début nous avons fait le test sans arrondir les valeurs de la variable "acc1". Nous avons alors obtenu une liste de 13857 valeurs. Ensuite, nous avons refait le test en variant à chaque fois le nombre de décimales pris en compte.

Nombre de décimales	Nombre de bornes possibles	Temps de calcul
0	51	120.306 sec = 2min
1	408	123532 sec = 2min 3sec
2	3106	125.356 sec = 2min 5sec
3	13857	133.285 sec = 2min 13sec

Tableau 4.12 – Taille de la liste des bornes possibles pour acc1 en variant le nombre de décimales.

Ce test nous a permis de voir l'impacte de la variation du nombre de décimales sur le temps d'exécution de notre algorithme. La troisième colonne du tableau 4.12 montre une petite différence (13 secondes) entre le cas où nous avons 51 bornes et celui où nous avons 13875 bornes. La figure 4.16 montre la courbe de l'évolution du temps d'exécution en fonction du nombre de décimale. Nous remarquons que plus on a de valeur plus le temps d'exécution est important. Cependant, variation n'est pas très considérable (quelques secondes).

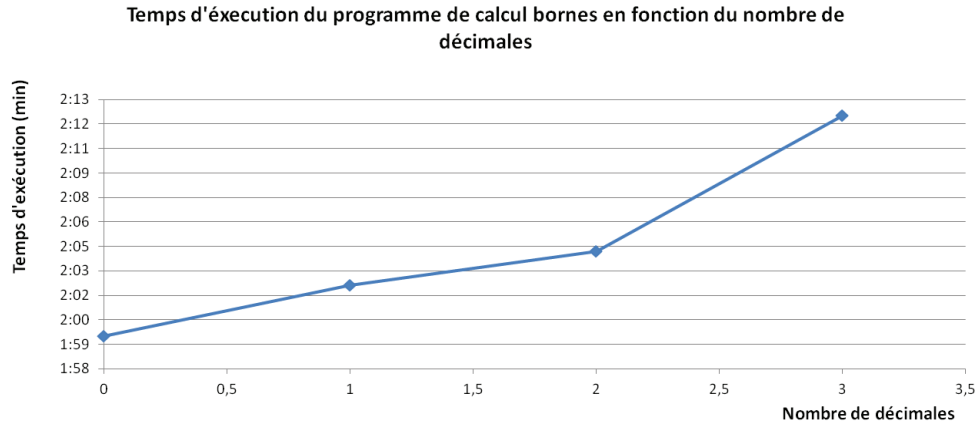


FIGURE 4.16 – Courbe de l'évolution du temps d'exécution de l'algorithme de calcul de bornes en fonction du nombre des décimales.

5.2.3 Résultats du module de calcul d'histogrammes

A partir du nombre de bornes possibles, nous avons décidé de tester les deux premier cas (avec 0 et 1 décimale). Cependant, même avec ce choix le coût du test de la première approche reste très élevé pour une seule variable. Par exemple, en choisissant de chercher les trois bornes les plus discriminants, nous devons lancer $C_3^5 1 = 20825$ clusters en parallèles. Pour toutes ces raisons, nous avons utilisé la deuxième approche pour convertir la variable acc1 en histogrammes.

5.2.3.1 Étude de l'impact de la taille des bornes sur le temps de la création d'histogrammes

Afin de connaitre la conséquence de la taille de la liste des bornes sur le temps d'exécution du module de calcul de histogrammes, nous avons extrait de la liste résultat du premier module des sous-listes de tailles différentes (5, 10, etc). Ensuite, elles ont été introduites au programme de calcul d'histogramme ce qui a donnée les résultats représentés par le tableau 4.13.

Nombre de bornes	Temps d'exécution
5	05 :14
10	05 :26
20	05 :49
40	07 :03
315	19 :54

Tableau 4.13 – Évaluation du temps d'exécution en fonction de la taille de la liste des bornes possibles.

La courbe représentée par la figure 4.17 illustre l'évolution du temps d'exécution de l'algorithme en variant la taille de liste des bornes.

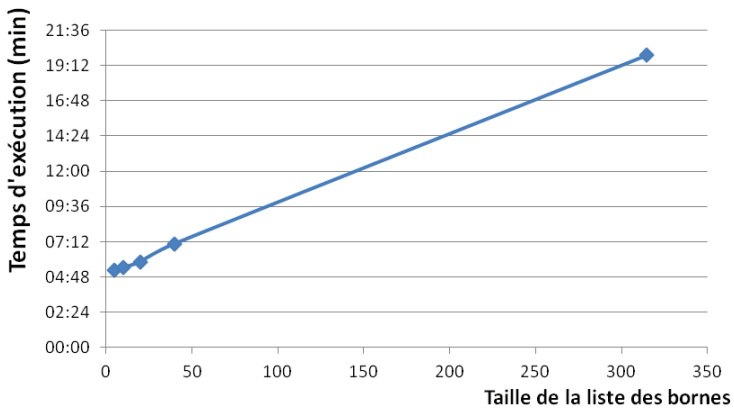


FIGURE 4.17 – Évolution du temps d'exécution du calcul d'histogrammes en fonction du nombre des bornes possibles.

La figure 4.17 montre que plus le nombre de bornes est important plus le temps d'exécution de l'algorithme de création d'histogramme est important. Ce test vient consolider l'évaluation de la complexité de notre algorithme que nous avons établie d'une façon théorique.

5.2.3.2 Traitement des 51 et 408 valeurs de bornes possibles

Dans cette partie, nous avons évalué le temps d'exécution du module de calcul d'histogramme en utilisant la liste des bornes arrondies (51 valeurs) et celle avec une décimale (408 valeurs).

Afin de traiter ces deux cas, nous avons utilisé deux configuration différentes : avec 8 et 16 nœuds. A partir du tableau 4.14, nous constatons que :

- Le temps d'exécution du module "Calcul Histogramme" est inversement proportionnel à la taille du cluster : plus le nombre de nœuds dans le cluster est grand, moins le sera le temps d'exécution.

- Le temps d'exécution du module "Calcul Histogramme" est proportionnel à la taille de la liste des bornes possibles : plus la taille de la liste des bornes est importante, plus le temps d'exécution le sera.

51 bornes possibles		408 bornes possibles	
8 nœuds	16 nœuds	8 nœuds	16 nœuds
4 min	2 min. 32 sec	16 min. 22 sec.	8 min. 40 sec.

Tableau 4.14 – Temps d'exécution du calcul d'histogrammes en utilisant deux clusters de tailles différentes et deux listes bornes différentes.

5.2.3.3 Traitement du cas avec 3106 bornes possibles

a. Présentation de la solution

Afin de traiter ce cas, nous avons subdivisé la liste des bornes (3106 valeurs) en sous-listes de taille 200. Ce découpage a été fait de façon que chaque deux sous-listes successives aient une valeur frontière commune (voir figure 4.18). En plus de la sous liste nous introduisons à chaque Job un indice de début qui sera additionné aux indices des intervalles issus des sous-listes de bornes (voir deuxième tableau de la figure 4.18). Ce paramètre "Début" a été introduit afin de pouvoir fusionner les résultats en utilisant les sous-listes. Nous obtiendrons alors exactement le même résultat que si nous avions utilisé la liste complète. Lors de la fusion pour deux valeurs associées à une même clé nous prenons la valeur minimale (voir figure 4.19).

b. Tests en utilisant AWS

Afin de créer les histogrammes en utilisant les 3106 bornes, nous avons commencé par la subdivision de cette liste en 16 sous-listes en suivant la méthode décrite dans la section précédente. Ensuite, nous avons lancé en parallèle 4 clusters de 11 nœuds de type m3.xlarge (voir figure 4.20).

Dans chaque cluster, nous avons lancé 4 jobs de création d'histogrammes qui s'exécutent de façon séquentielle (voir figure 4.21). Chacun de ces jobs a mis une minute pour retourner le résultat de création d'histogrammes sur les données initiales. En total, il a fallut 4 minutes pour avoir le résultat en utilisant toutes les sous-listes de la liste de 3106 bornes.

Après le téléchargement des fichiers résultats et leur fusion en utilisant le principe présenté précédemment, nous obtenons un fichier de 112 Ko résumant les données initiales (de 1.75 Go).

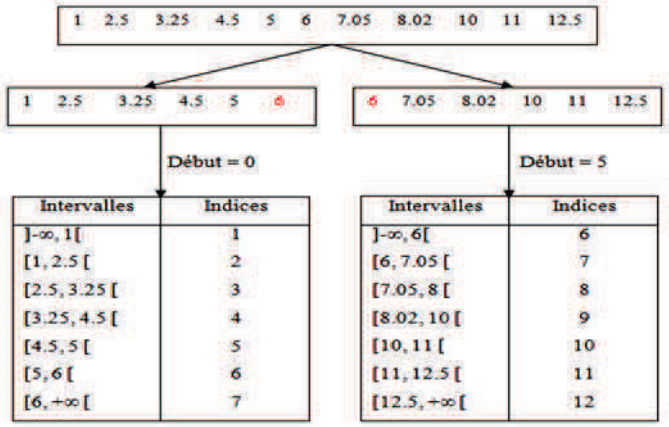


FIGURE 4.18 – Exemple de subdivision de la liste des bornes et présentation de la liste des indices.

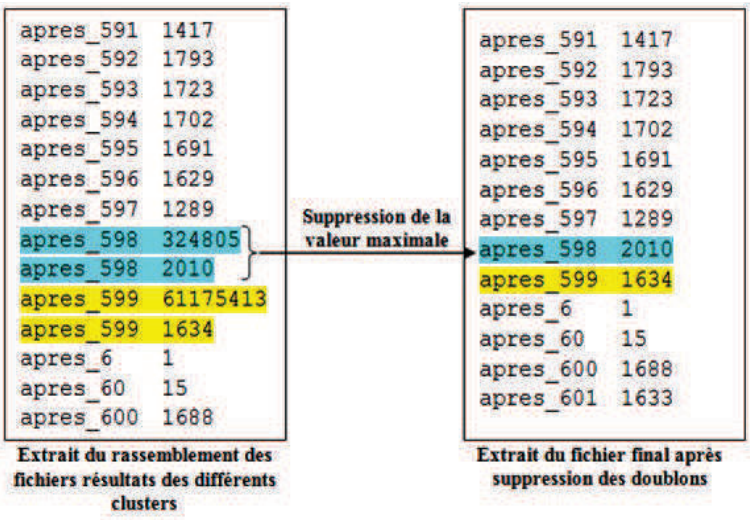


FIGURE 4.19 – Exemple illustratif du traitement de la fusion entre deux fichiers issus de deux jobs différents.

5.2.4 Résultats du module de recherche des histogrammes les plus discriminants

Après avoir récupérer les fichiers résultats des premiers modules, nous les introduisons au module de recherche des histogrammes les plus discriminants. Dans la suite nous présentons les résultats obtenus dans chaque cas en les comparant aux résultats du test de HistSyr sur un échantillon des données initiales. Cette échantillon a été construit en prenant 1 ligne sur chaque 10000 lignes.

5.2.4.1 Cas avec 51 bornes possibles

a. Présentation des résultats

La figure 4.22 nous présente les différents fichiers générés par ce module. Nous trouvons :

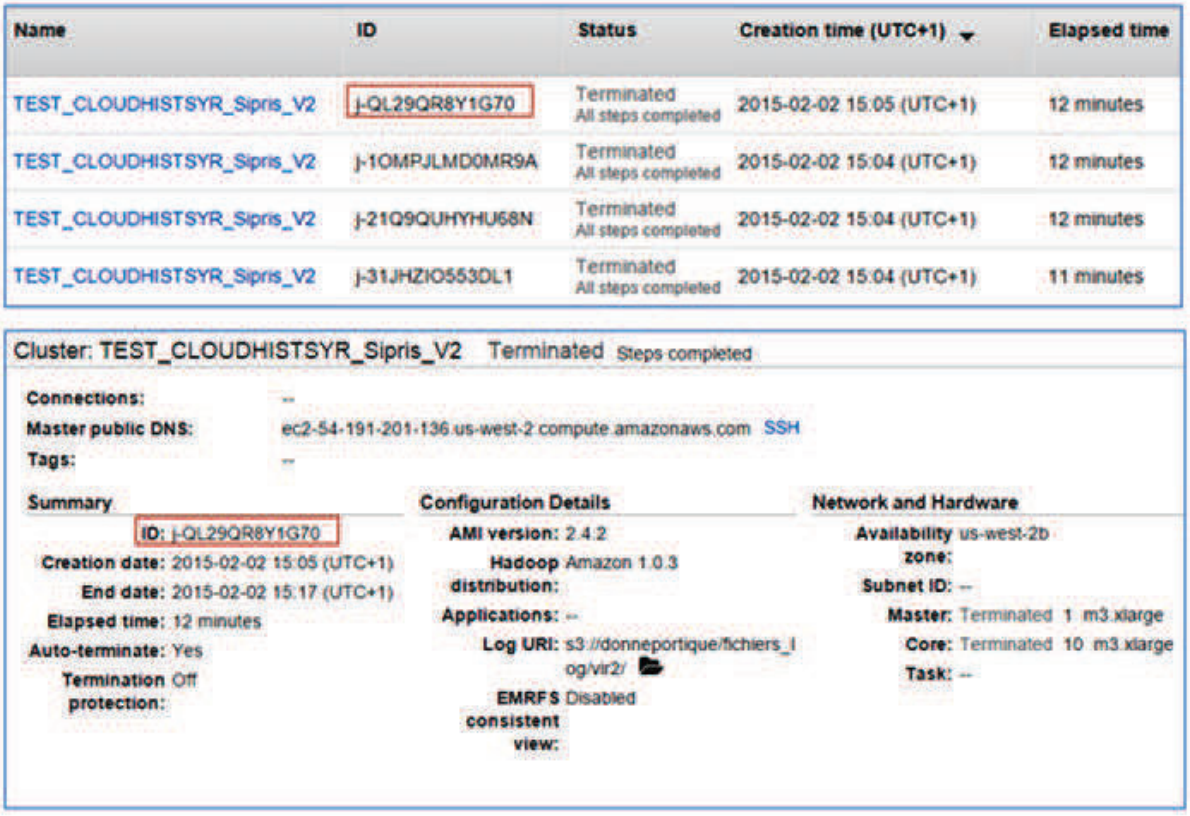


FIGURE 4.20 – Présentation des clusters utilisés et de leur configuration.

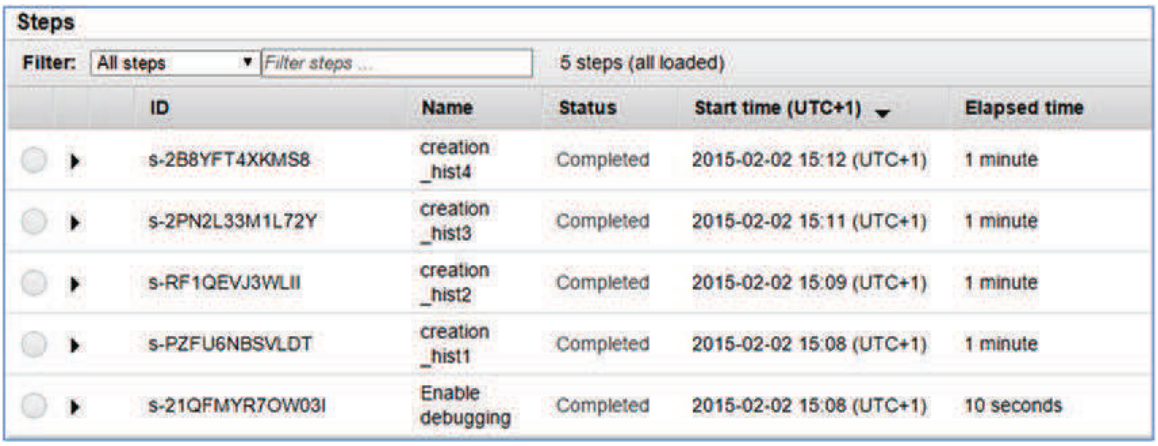


FIGURE 4.21 – Lancement des quatre jobs de calcul d'histogrammes en utilisant des listes de bornes de 200 valeurs.

- Le fichier .Syr (voir la figure 4.22.a) qui représente le fichier Symbolique résultat. A partir de cette figure nous pouvons conclure que les histogrammes représentant les concepts ″avant″ et ″après″ sont très similaires mais différents par rapport au troisième histogramme.

- Le fichier pattern (voir la figure 4.22.b) qui représente la description du fichier

symbolique en donnant le type de chaque variable et les bornes de découpage pour l'obtention des histogrammes. Dans ce cas, les meilleurs histogrammes, de trois modalités, sont obtenus en utilisant les bornes 0.5, 1.5.

- Un dernier fichier résumant les principaux résultats (voir figure 4.22.c) qui rappelle la liste de meilleures bornes, donne le score des histogrammes obtenus (qui est de 0.45) et le temps de calcul qui est de 78ms.

Les résultats de HistSyr sur les données échantillonnées (13931 lignes) sont représentés par la figure 4.23. Ce résultat a été obtenu au bout de 1152ms.

b. Comparaison et interprétation des résultats

En comparant les deux figures 4.22 et 4.23, nous remarquons que :

- Le dernier module de CloudHistSyr qui utilise les données résumées résultats des modules Map/Reduce est plus rapide que "HistSyr" qui tourne que les données échantillonnés.
- Les meilleures bornes sont identiques.

Nous avons la même interprétation pour les classes d'individus quelque soit les données utilisés car dans les deux cas les histogrammes des classes "avant" et "après" se ressemblent entre eux mais sont différents de celui du concept "pendant". Pour cette dernière classe nous avant 95% des individus qui ont une valeur du premier accéléromètre qui est strictement supérieur à "1.5". Cette remarque est logique puisque les deux états "avant" et "après" représentent où il n'y a pas de défaut sur le portique.

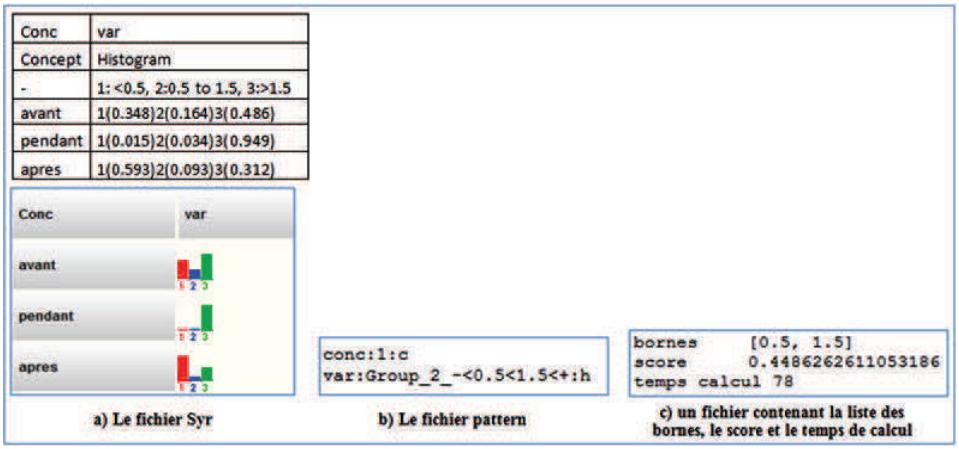


FIGURE 4.22 – Résultats du module de recherche des histogrammes les plus discriminants sur les données Sipris avec la liste de 40 bornes

5.2.4.2 Cas avec 408 bornes possibles

Nous avons procédé de la même manière qu'avec le premier cas. La figure 4.24 représente les résultats obtenus en utilisant le troisième module de CloudHistSyr. Alors que la figure 4.25 représente les résultats issus de HistSyr sur les données échantillonnées au bout de 10006 ms.

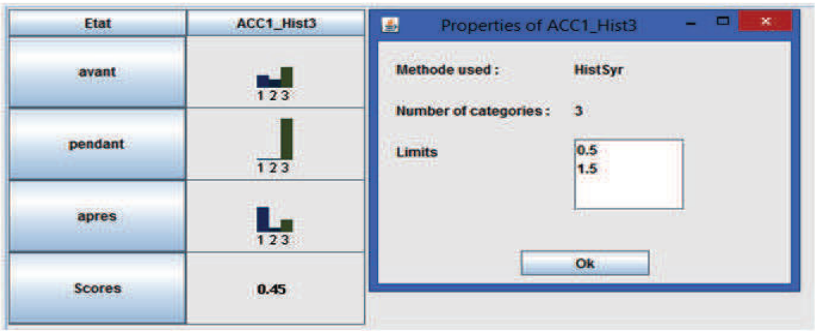


FIGURE 4.23 – Résultats de HistSyr sur les données échantillonnées avec l'arrondissement des valeurs de ″acc1″.

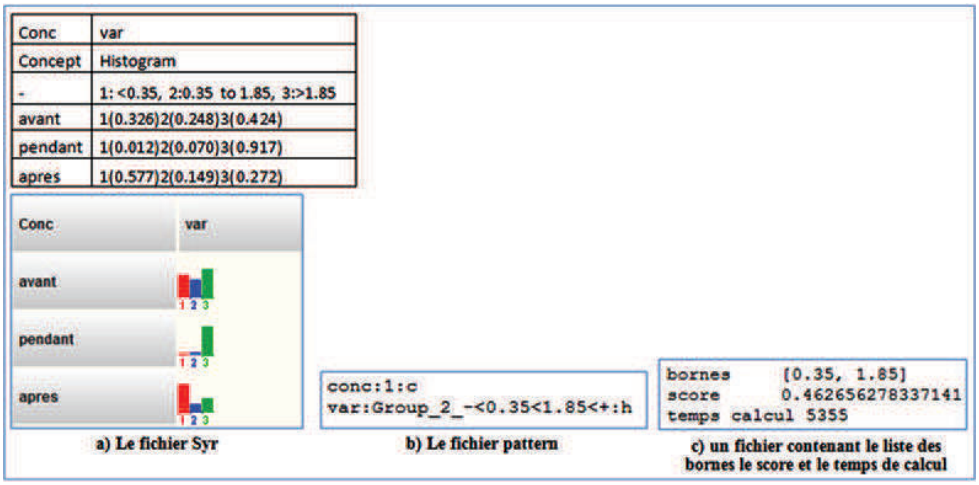


FIGURE 4.24 – Résultats du module de recherche des histogrammes les plus discriminants sur les données Sipris avec la liste des 408 bornes

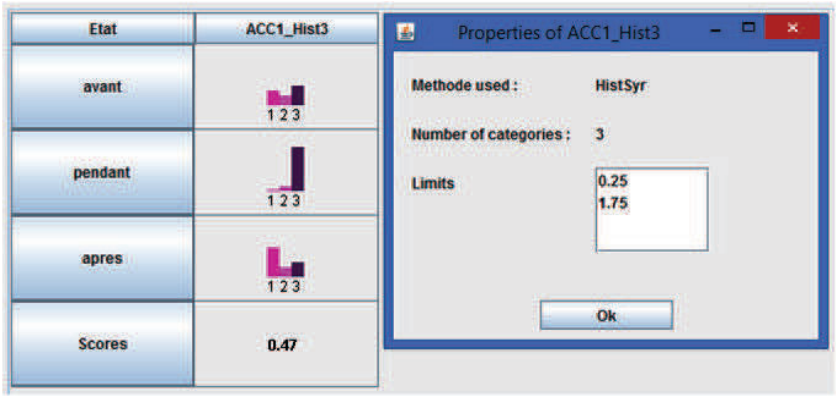


FIGURE 4.25 – Résultats de HistSyr sur les données échantillonnées avec l'arrondissement des valeurs de ″acc1″ à une seule décimale.

D'après les figures 4.24 et 4.25 nous avons les mêmes remarques et interprétations que dans le cas où nous avons une liste de 51 bornes possibles. Néanmoins, les meilleures bornes sont

différentes. Dans ce cas les meilleurs bornes en utilisant CloudHistSyr sont {0.35, 1.85} alors qu'elles sont {0.25, 1.75} pour les données échantillonnées.

5.2.4.3 Cas avec 3106 bornes possibles

Nous avons procédé de la même manière qu'avec le premier cas. La figure 4.26 représente les résultats obtenus en utilisant le troisième module de CloudHistSyr. Alors que la figure 4.27 représente les résultats issus de HistSyr sur les données échantillonnées au bout de 563902 ms = 9 min. 23 sec. 902 ms.

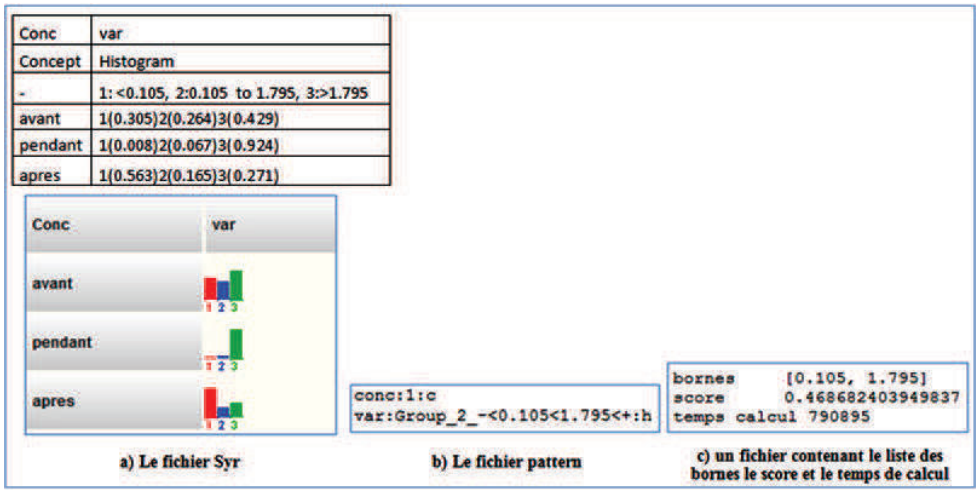


FIGURE 4.26 – Résultats du module de recherche des histogrammes les plus discriminants sur les données Sipris avec la liste des 3106 bornes.

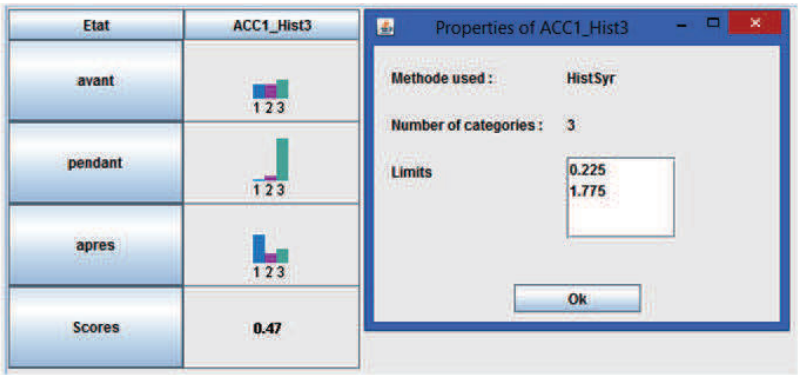


FIGURE 4.27 – Résultats de HistSyr sur les données échantillonnées avec l'arrondissement des valeurs de "acc1" à deux décimales.

Ces deux figures nous donnent les mêmes conclusions que les deux autres cas (avec 51 et 408 bornes). Les histogrammes obtenus sur les données échantillonnées et sur les résultats de CloudHistSyr ont la même allure. Cependant, dans ce cas le temps de calcul des histogrammes les plus discriminants est plus long en prenant les données issues de CloudHistSyr qu'en utilisant les données échantillonnées.

6 Conclusion

L'évolution de la taille des bases de données a augmenté d'une façon considérable (de quelques Ko à des To). Pour cela, les "*data scientistes*" se sont adaptés en étendant différentes méthodes de *data mining* aux *Big Data*.

Au sein de notre travail, nous avons étendu l'ADS aux *Big Data* en commençant par l'extension de l'extraction des données symboliques à partir de grandes bases de données. Pour cela, nous avons mis en place de l'extension d'HistSyr aux grandes bases de données. Ce qui nous a permis de passer de 57 minutes pour obtenir des histogrammes en utilisant 2 bornes à une vingtaine de minutes pour trouvez les histogrammes les plus discriminants en partant d'une liste de 3106 bornes possibles.

La solution que nous avons proposée utilise Map/Reduce comme modèle de programmation. Cependant, ça peut être intéressant de tester d'autres outils de programmation comme le langage matriciel R pour étendre HistSyr aux données scalables et distribuées et de comparer les performances de CloudHistSyr par rapport à de telles solutions.

CHAPITRE 5

Conclusion et perspectives

1 Conclusion

Les objectifs de cette thèse étaient d'enrichir les deux étapes de l'ADS et d'étendre la création de données symboliques aux *Big Data*. Afin d'atteindre ces objectifs, nous avons commencé nos travaux par l'étude des méthodes et des outils d'ADS existants. Cette étude nous a permis d'affiner nos objectifs. En effet à partir de cette étude nous avons constaté l'absence de méthodes d'ADS permettant la conversion automatique des variables continues en histogrammes. Nous avons remarqué aussi que la plupart des méthodes d'arbres de décision symboliques existantes ne traitent pas tous types de variables à expliquer et explicatives. Finalement, nous avons constaté que les outils et méthodes existantes d'ADS sont incapables de traiter de grands volumes de données. Afin de résoudre toutes ces problématiques nous avons mis en place trois méthodes "HistSyr", "SyrTree" et "CloudHistSyr".

Dans le cadre de l'enrichissement de la première étape de l'ADS, la transformation des données du classique au symbolique, nous avons créé une méthode nommée "HistSyr". Cette méthode a permis d'extraire automatiquement les histogrammes les plus discriminants à partir d'une variable quantitative. L'implémentation de cette méthode a permis à l'utilisateur du logiciel Syr de créer des histogrammes à partir de variables continues et d'apporter des modifications à des histogrammes existants. L'efficacité de cette méthode a été prouvée en comparant ces résultats aux histogrammes issus d'autres méthodes de discrétisation. Le module "HistSyr" du logiciel Syr offre l'implémentation de la méthode HistSyr ainsi que d'autres méthodes de discrétisation existantes non supervisées (discrétisation en effectifs égaux et en intervalles égaux). Il a été créé afin d'enrichir le logiciel Syr pour extraire des données symboliques conservant le maximum d'informations à partir de bases classiques.

Après la création d'HistSyr nous nous sommes intéressés à la création d'une nouvelle méthode d'arbres de décision symboliques. Pour cela, nous avons mis en place la méthode "SyrTree". Cette méthode représente l'extension de CART [20] aux données symboliques. Elle accepte tous types de variables à expliquer (intervalles, histogrammes et variables continues et nominales) et explicatives (la classe d'individus, les variables nominales et histogrammes).

Nous avons mis en place plusieurs stratégies pour la construction des arbres symboliques de ”SyrTree” à partir de données classiques. Nous avons par la suite appliqué ces stratégies sur des données issues du répertoire UCI et sur des données réelles. Les résultats de test de SyrTree ont été comparés aux résultats d'autres méthodes d'arbres de décision classiques (CART [20] et C4.5 [89]) et symbolique (Stree [97]). A partir de cette comparaison, nous avons conclu que notre méthode peut bien concurrencer les autres méthodes d'arbre de décision. En effet, les résultats de test de SyrTree sont meilleurs que ceux de l'arbre symbolique de Stree [97] et se rapprochent des résultats des arbres classiques (nous trouvons des cas où nos résultats sont meilleurs comme pour les données UCI ”Breast tissu”).

Finalement et afin d'étendre l'ADS aux données scalables et distribuées, nous avons commencé par l'extension de l'extraction des données symboliques à partir de grandes bases de données. Etant donné que la conversion des variables du classique au symbolique est l'étape la plus délicate dans la construction des données symboliques, nous nous somme intéresser à étendre la méthode HistSyr aux données scalables et distribuées. Cette extension a été réalisée en décomposant HistSyr en trois étapes : le calcul des bornes frontières, le calcul des histogrammes en utilisant toutes les bornes possibles et la recherche des histogrammes les plus discriminants en se basant sur les résultats des deux premières étapes. Les deux premières étapes ont été ré-implémentées en utilisant le modèle de programmation Map/Reduce alors que la troisième partie de CloudHistSyr représente un programme java qui tourne en local. Cette méthode nous a permise de passer de l'impossibilité d'exécuter HistSyr sur des données de 1.75Go à une vingtaine de minutes pour trouvez les histogrammes les plus discriminants à partir de ces données.

Pour conclure, le travail effectué dans cette thèse représente une première contribution à l'extension de l'analyse des données symboliques aux données massives (*Big Data*). Elle construit d'abord les données symboliques qui résument la masse des données selon des classes fournies puis étend les arbres de décisions à ce type de données.

2 Perspectives

2.1 Évaluation de l'utilité de HistSyr

Durant cette thèse nous avons mis en place une méthode qui permet de convertir les variables continues en histogrammes les plus discriminants pour les classes d'individus. Dans de futurs travaux, ça serait intéressant d'évaluer l'impacte de l'utilisation de cette description (par rapport à l'utilisation des intervalles par exemple) sur les résultats des différentes méthodes d'analyse symboliques (comme l'ACP, les arbres de décision, etc.).

2.2 Arbres de décision symboliques

Durant cette thèse nous avons testé et validé les arbres de décision de SyrTree en utilisant des matrices de confusions et les taux d'erreur. Dans la suite, il serait intéressant d'utiliser d'autres méthodes de validation comme la courbe de ROC [92].
Dans de futurs travaux il serait intéressant de tester la construction d'ensembles de modèles d'arbres SyrTree soit par bagging [19] ou par boosting [96] avec notre algorithme comme cela

à été fait dans la thèse de Seck avec son algorithme Stree [97].
En se basant sur le principe des arbres de décision "SyrTree", il serait intéressant de proposer une méthode d'arbres de régression symbolique. Cette méthode prendrait en variable à expliquer des variables continues ou des variables intervalles.

2.3 Extension de l'ADS aux Big Data

La solution que nous avons proposée pour l'extension d'HistSyr aux Big Data utilise Map-Reduce comme modèle de programmation. Cependant cela pourrait être intéressant de tester d'autres outils de programmations comme le langage matriciel R pour cette extension et de comparer ensuite les performances de CloudHistSyr par rapport à de telles solutions.
Il serait également intéressant de considérer d'autres méthodes d'analyse symboliques (comme les nuées dynamiques symboliques, l'analyse en composante principale symbolique, etc.) afin d'étendre l'ADS aux aux grandes masses de données.

Annexe A
Méthode de sélection du nombre de clusters pour la stratégie 2 de création des arbres de décision ″SyrTree″

1 La méthode

Pour connaître le meilleur nombre de clusters, pour chaque jeu de données nous procédons ainsi :

1. Lancer ClustSyr en fixant un intervalle de nombre de clusters.

2. Parmi les fichiers résultats de ClustSyr un fichier nommé "..._prototypes.syr" est alors créer. Ce fichier contient la description de différentes classes de chaque classification. Nous convertissons ce fichier en fichier texte en utilisant TabSyr. Ensuite nous supprimons toutes les variables descriptives des classes obtenues à l'exception de la variable à expliquer. Nous calculons ensuite pour chaque classification le nombre de classes décrites par des histogrammes à une seule modalité pour la variable à expliquer.

3. La classification qui a le plus de classes décrites par des histogrammes à une seule modalité est celle qui sera choisie

2 Exemple illustratif

Nous donnons un exemple sur l'enchainement suivit pour la base de données UCI Abalones. La figure 1 illustre la configuration de ClustSyr pour exécuter les nuées dynamiques symboliques avec un nombre de clusters entre 60 et 100.

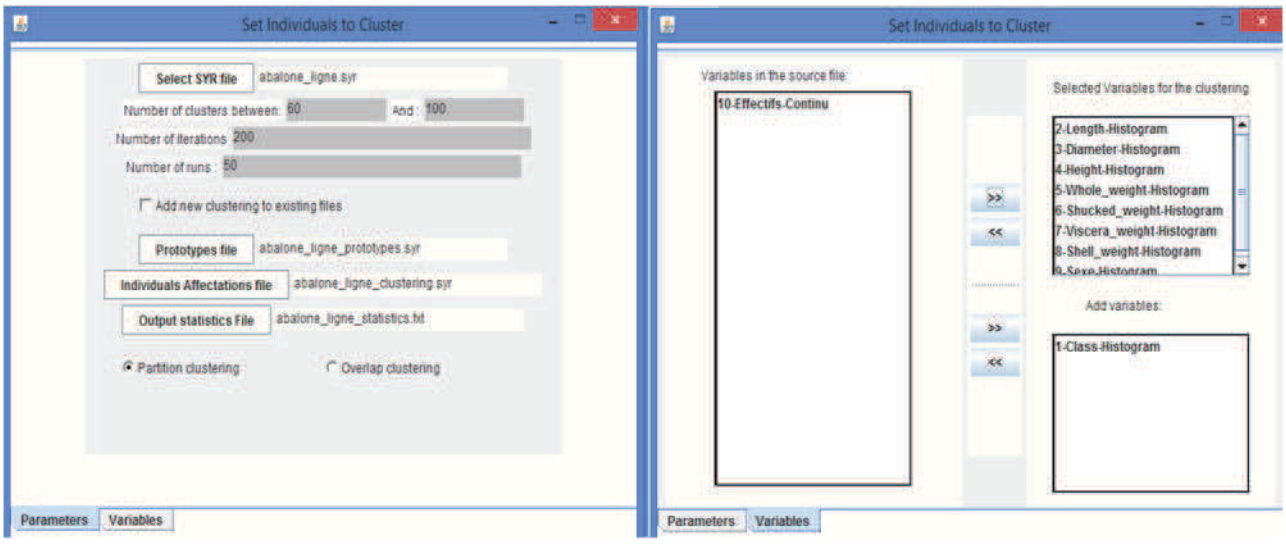


FIGURE 1 – Lancement et configuration de "ClustSyr" pour avoir le résultat de la classification en ayant entre 60 et 100 classes pour les données Abalone.

Nous obtenons alors un fichier "abalone_ligne_prototype.syr" que nous convertissons en fichier txt en utilisant TabSyr. Nous ouvrons ce dernier en utilisant Excel et nous supprimons toutes les variables descriptives des différentes classes à part la variable à expliquer (nommé "class". Le tableau 1 représente un extrait de ce fichier.

Clusters	Class		
Concept	Histogram		
-	Adult	Old	Young
Partition_60_C1	0.9697	0.0303	0
Partition_60_C2	0.95652	0	0.0435
Partition_60_C3	0.88043	0.0761	0.0435
Partition_60_C4	0	0	1
Partition_60_C5	0.13043	0.8696	0
Partition_60_C6	0	0	1
Partition_60_C7	0	1	0
Partition_60_C8	1	0	0
Partition_60_C9	1	0	0

Tableau 1 – Extrait du fichier "abalone_ligne_prototype.syr" après la suppression des variables explicatives des clusters à l'exception de la variable "class".

En utilisant ce fichier nous calculons pour chaque nombre de clusters (noté nc) le nombre de classes décrites par une variable class avec une seule modalité (noté ncp). Le tableau 2 résume les résultats obtenus. A partir de ce tableau nous constatons que le meilleur nombre de clusters à prendre est égal à 100

nc	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
ncp	23	25	22	29	25	22	28	36	29	32	26	32	38	29	35	37	37	46	42	36	39

nc	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95 96	97	98	99	100
ncp	49	40	41	45	48	58	41	45	48	60	51	52 50	55	55	57	63	61	60	67

Tableau 2 – Résultats du calcul des nombre de classes décrite par un histogramme pure pour la variable à expliquer en fonction des nombres de clusters.

3 Résultats sur les différentes bases UCI testées

Le tableau 3 représente les nombre de clusters retenus pour chaque base de données UCI étudiée.

Base de données UCI	Nombre de clusters
Abalone	100
BreastTissue	39
GlassIdentification	73
Seeds	59
Ionosphere	99
Prima-indian-diabetes	99
Breast-cancer-Wisconsin	99

Tableau 3 – Nombre de clusters retenu pour les bases de données UCI étudiées.

Annexe B
Résultats des tests des arbres de décisions sur les données UCI

Abalone															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.1833	0.2703	0.241	0.2536	0.4166	0.4282	0	0.2886	0.3062	0.41	0.47	0.2557	0.2584	0.282	0.2608
15%	0.1358	0.303	0.2403	0.2313	0.4183	0.4146	0	0.2763	0.2823	0.42	0.4941	0.2544	0.2632	0.2797	0.2807
20%	0.1338	0.2982	0.2388	0.2371	0.4144	0.4311	0	0.2832	0.2956	0.41	0.4687	0.2576	0.2503	0.2774	0.2778
25%	0.1334	0.3123	0.2359	0.2519	0.42	0.411	0	0.294	0.2931	0.45	0.4639	0.2547	0.2596	0.2745	0.2807
30%	0.1371	0.3001	0.2438	0.245	0.4256	0.423	0	0.295	0.2949	0.41	0.5204	0.2558	0.265	0.4744	0.4874
moy	0.1447	0.2968	0.2399	0.2438	0.4189	0.4216	0	0.2874	0.2944	0.42	0.4834	0.2556	0.2593	0.3176	0.3175

Breast Tissue															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.1771	0.4	0.1979	0.3	0.2187	0.2	0.33	0.4167	0.4	0.0526	0.1	0.6042	0.6	0.6042	0.6
15%	0.1868	0.3333	0.2088	0.3333	0.2088	0.26667	0.33	0.4285	0.4667	0.1667	0.333	0.6044	0.6	0.6044	0.6
20%	0.1059	0.4286	0.3647	0.2857	0.2	0.2381	0.16667	0.4	0.4285	0.147	0.3157	0.5882	0.619	0.5882	0.619
25%	0.1605	0.36	0.321	0.44	0.2099	0.28	0	0.321	0.36	0	0	0.6049	0.6	0.6049	0.6
30%	0.12	0.3226	0.3467	0.3548	0.2	0.2581	0.16667	0.4	0.4193	0.0285	0.1428	0.6133	0.6129	0.604	0.6
moy	0.1501	0.3689	0.2878	0.3428	0.2074	0.248574	0.198668	0.39324	0.4149	0.07896	0.1783	0.603	0.6064	0.60114	0.6038

GlassIdentification															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.1615	0.3182	0.276	0.4091	0.422	0.409	0.167	0.5312	0.5363	0.3611	0.4238	0.3854	0.3182	0.526	0.5455
15%	0.1639	0.3548	0.2896	0.3548	0.41	0.484	0.5	0.3442	0.4516	0.2222	0.407	0.3661	0.4516	0.5322	0.5116
20%	0.1988	0.2326	0.3158	0.2791	0.48	0.512	0.333	0.5204	0.5581	0.3571	0.475	0.386	0.3488	0.5168	0.5538
25%	0.1605	0.3269	0.2778	0.2692	0.333	0.385	0.167	0.4567	0.5384	0.2463	0.4418	0.3827	0.3654	0.5391	0.5116
30%	0.1544	0.4615	0.349	0.4462	0.329	0.416	0.5	0.5329	0.5781	0.3776	0.4477	0.349	0.4462	0.5315	0.5511
moy	0.1678	0.3388	0.3016	0.3517	0.3948	0.4412	0.3334	0.47708	0.5325	0.31286	0.43906	0.3738	0.3860	0.52912	0.5347

Seeds															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.0317	0.1429	0.0794	0.1429	0.0741	0.143	0	0.037	0.0476	0.0169	0.0769	0.0741	0.1429	0.2804	0.2381
15%	0.0278	0.1	0.0944	0.3	0.089	0.1	0	0.0445	0	0.0169	0	0.0722	0.1333	0.1944	0.2333
20%	0.0298	0.1905	0.0893	0.119	0.0715	0.119	0	0.0238	0.1428	0	0.1667	0.0714	0.119	0.3929	0.38
25%	0.0252	0.1176	0.0692	0.1176	0.0692	0.137	0	0.088	0.0784	0.0877	0.1379	0.0692	0.1176	0.2767	0.2549/
30%	0.0204	0.0794	0.0952	0.0476	0.095	0.048	0.095	0.088	0.0952	0.0754	0.1	0.0952	0.0476	0.2925	0.2381//
moy	0.027	0.1261	0.0855	0.1454	0.07976	0.1094	0.019	0.05626	0.0728	0.03938	0.0963	0.07642	0.1121	0.2874	0.2691/

Breast-cancer-Wisconsin															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.0276	0.0294	0.0569	0.0294	0.10227	0.14706	0.5	0.0634	0.0442	0.0306	0.0487	0.0714	0.0588	0.0877	0.1029
15%	0.0241	0.068	0.0345	0.0291	0.10327	0.12621	0	0.0564	0.0576	0.0421	0.1132	0.074	0.068	0.1033	0.1262
20%	0.0274	0.0441	0.0311	0.1029	0.09506	0.15328	0.5	0.0475	0.0656	0.0215	0.0793	0.0622	0.1168	0.106	0.1533
25%	0.0254	0.0468	0.0703	0.0819	0.10331	0.15205	0.5	0.0662	0.0694	0.0322	0.1442	0.0702	0.0819	0.1111	0.0936
30%	0.0376	0.0686	0.0292	0.0686	0.09812	0.12629	0	0.0489	0.0582	0.0689	0.0958	0.071	0.078	0.0981	0.1268
moy	0.0284	0.0514	0.0444	0.0624	0.1004	0.1409	0.3	0.0564	0.059	0.0390	0.0962	0.0697	0.0807	0.1012	0.1205

Ionosphere															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.0475	0.0886	0.0854	0.1714	0.20886	0.2	0	0.0919	0.086	0.0859	0.0868	0.1709	0.1143	0.2278	0.3429
15%	0.0268	0.0943	0.057	0.0943	0.20805	0.20755	0	0.0843	0.0754	0.0763	0.0833	0.1611	0.2075	0.2013	0.2453
20%	0.0391	0.0857	0.0427	0.1	0.20641	0.21429	0	0.064	0.0857	0.055	0.0851	0.1637	0.1857	0.2596	0.2143
25%	0.0417	0.069	0.1023	0.069	0.32955	0.36782	0	0.0712	0.0864	0.0674	0.0838	0.1667	0.2299	0.2311	0.2644
30%	0.0407	0.0762	0.1057	0.0762	0.21138	0.2	0	0.0741	0.089	0.0718	0.0867	0.1789	0.1333	0.2317	0.2571
moy	0.0391	0.0827	0.0786	0.1022	0.23285	0.237932	0	0.0771	0.0845	0.07128	0.08514	0.16826	0.1741	0.2303	0.2648

Prima-indian-diabetes															
	C4.5		CART		SyrTree_strat1			SyrTree_strat2				STREE		STREE (div 100)	
	app	test	app	test	indiv_app	indiv_test	conc_test	indiv_app	indiv_test	conc_app	conc_test	app	test	app	Test
10%	0.0984	0.4545	0.2301	0.2987	0.2764	0.2987	0	0.3257	0.3117	0.1919	0.2584	0.259	0.3151	0.259	0.3117
15%	0.1164	0.2435	0.242	0.2087	0.2802	0.2695	0	0.3032	0.2869	0.1818	0.2568	0.2634	0.2696	0.2542	0.2435
20%	0.114	0.2987	0.184	0.2792	0.2768	0.2857	0	0.3241	0.3046	0.1812	0.278	0.2427	0.3072	0.2655	0.2549
25%	0.1181	0.3073	0.217	0.2552	0.2795	0.276	0	0.3107	0.3062	0.1515	0.2368	0.2309	0.3089	0.2448	0.3246
30%	0.1097	0.287	0.2156	0.2522	0.2751	0.2869	0	0.3252	0.3017	0.1632	0.2560	0.2621	0.2739	0.2621	0.2739
moy	0.1113	0.3182	0.2177	0.2588	0.2776	0.28336	0	0.31778	0.3022	0.1739	0.2572	0.25162	0.2949	0.2571	0.2817

Annexe C
présentation des résultats des tests en utilisant EMR

ordinateur	taille données (Mo)	nombre de lignes	temps d'exécution (sec)	temps d'exécution (min)
1 (1Go, 32bits)	27,2	1500000	48,969	00 :49
1 (1Go, 32bits)	271,6	15000000	65,781	01 :24
1 (1Go, 32bits)	543,2	30000000	144,594	02 :24
1 (1Go, 32bits)	1433,6	75000000	341,791	05 :41
1 (1Go, 32bits)	13926,4	750000000	3442,248	57 :22
2 m1small	27,2	1500000	141,777	02 :21
2 m1small	271,6	15000000	201,225	03 :21
2 m1small	543,2	30000000	259,905	04 :19
2 m1small	1433,6	75000000	395,937	06 :35
2 m1small	13926,4	750000000	2800,17	46 :40
4 m1small	27,2	1500000	158,726	02 :38
4 m1small	271,6	15000000	183,279	03 :00
4 m1small	543,2	30000000	205,808	03 :25
4 m1small	1433,6	75000000	299,89	04 :59
4 m1small	13926,4	750000000	1506,876	25 :06
6 m1small	27,2	1500000	151,203	02 :31
6 m1small	271,6	15000000	163,333	02 :43
6 m1small	543,2	30000000	178,186	02 :58
6 m1small	1433,6	75000000	249,204	04 :09
6 m1small	13926,4	750000000	1000,767	16 :40
8 m1small	1433,6	75000000	236,112	03 :56
8 m1small	13926,4	750000000	818,112	13 :38
16 m1small	13926,4	750000000	519,398	08 :39
2 m1Large	13926,4	750000000	780,165	13 :00
4 m1Large	13926,4	750000000	410,356	06 :50
6 m1Large	13926,4	750000000	323,5	05 :23
8 m1Large	13926,4	750000000	255,519	04 :15
16 m1Large	13926,4	750000000	176,909	02 :57

Table des figures

Liste des tableaux

Bibliographie

[1] Y. J.-P. Aboa. *Méthodes de segmentation sur un tableau de variables aléatoires*. PhD thesis, Université Paris Dauphine-Paris IX, 2002.

[2] F. Afonso. Extension de l'algorithme apriori et des règles d'association au cas des données symboliques diagrammes et sélection des meilleures règles par la régression linéaire symbolique. *Revue RNTI, Classification et fouilles de données*, 2004.

[3] F. Afonso, L. Billard, E. Diday, and M. Limam. Symbolic linear regression methodology. *Symbolic Data Analysis and the SODAS Software*, pages 359–372, 2008.

[4] F. Afonso and E. Diday. Extension de l'algorithme apriori et des règles d'association aux cas des données symboliques diagrammes et intervalles. In *Revue RNTI, Extraction et Gestion des Connaissances*, pages 483–494, 2005.

[5] F. Afonso, E. Diday, and R. Haddad. Latest developments of the syr software for symbolic data analysis of complex data. 3^{rd} *Workshop on Symbolic Data Analysis*, 2012.

[6] V. Batagelj and N. Kejzar. Clamix clustering symbolic objects. *Computer software R program. Vienna : R Foundation for Statistical Computing. Available : https ://r-forge. r-project. org/projects/clamix*, 2010.

[7] P. Bertier and J.-M. Bouroche. *Analyse des données multidimensionnelles*. Presses universitaires de France, 1975.

[8] P. Bertrand and F. Goupil. Descriptive statistics for symbolic data. In *Analysis of symbolic data*, pages 106–124. Springer, 2000.

[9] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta. Distributed decision-tree induction in peer-to-peer systems. *Statistical Analysis and Data Mining*, 1(2) :85–103, 2008.

[10] L. Billard and E. Diday. Regression analysis for interval-valued data. In *Data Analysis, Classification, and Related Methods*, pages 369–374. Springer, 2000.

[11] L. Billard and E. Diday. Symbolic regression analysis. In *Classification, Clustering, and Data Analysis*, pages 281–288. Springer, 2002.

[12] L. Billard and E. Diday. Symbolic data analysis : Conceptual statistics and data mining john wiley, 2006.

[13] C. Blake and C. J. Merz. {UCI} repository of machine learning databases. Technical report, University of California, Irvine, Dept. of Information and Computer Sciences., 1998.

[14] H.-H. Bock and E. Diday. Analysis of symbolic data : Exploratory methods for extracting statistical information from complex data. 2000.

[15] D. Borthakur. Hdfs architecture guide. *HADOOP APACHE PROJECT http ://hadoop. apache. org/common/docs/current/hdfs design. pdf*, page 39, 2008.

[16] M. Boulle. Khiops : A statistical discretization method of continuous attributes. *Machine learning*, 55(1) :53–69, 2004.

[17] V. Bouteiller, C. Toque, A. Cury, J.-F. Cherrier, E. Diday, and C. Cremona. Non-destructive electrochemical characterizations of reinforced concrete corrosion : basic and symbolic data analysis. *Corrosion Reviews*, 30(1-2) :47–62, 2012.

[18] M. Bravo and J. García-Santesmases. Segmentation trees for stratified data. *Analysis of Symbolic Data. Exploratory Methods for Extracting Statistical Information from Complex Data. Springer, Heidelberg*, pages 266–293, 2000.

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

[20] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[21] P. Brito. *Analyse de données symboliques pyramides d'héritage*. PhD thesis, Université Paris Dauphine-Paris IX, 1991.

[22] P. Brito. Hierarchical and pyramidal clustering with complete symbolic objects. *Chapter 11 in Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data, (Vol.15), H.H. Bock and E. Diday, eds, Series : Studies in Classification, Data Analysis, and Knowledge Organisation.*, pages 312–324, 2000.

[23] W. Buntine. About the ind tree package. *NASA Ames Research Center, Mo ett Field, California*, 1991.

[24] A. Cano, S. Ventura, and K. J. Cios. Scalable caim discretization on multiple gpus using concurrent kernels. *The Journal of Supercomputing*, 69(1) :273–292, 2014.

[25] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *European working session on learning*, pages 164–178. Springer, 1991.

[26] P. Cazes, A. Chouakria, E. Diday, and Y. Schektman. Extension de l'analyse en composantes principales à des données de type intervalle. *Revue de Statistique appliquée*, 45(3) :5–24, 1997.

[27] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable : A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2) :4, 2008.

[28] M. Chavent. *Analyse de Données Symboliques. Une méthode divisive de classification*. PhD thesis, Université Paris Dauphine-Paris IX et INRIA Rocquencourt 78150, 1997.

[29] M. Chavent. Criterion-based divisive clustering for symbolic data. *Analysis of Symbolic Data : Exploratory Methods for Extracting Statistical Information from Complex Data*, pages 299–311, 2000.

[30] A. Chouakria. *Extension des méthodes d'analyse factorielle á des données de type intervalle.* PhD thesis, Université Paris Dauphine-Paris IX, 1998.

[31] A. Chouakria, P. Cazes, and E. Diday. Symbolic principal component analysis. *Analysis of Symbolic Data, ed. HH Bock, and E. Diday*, pages 200–212, 2000.

[32] C. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19 :281, 2007.

[33] W. S. Cleveland. Data science : an action plan for expanding the technical areas of the field of statistics. *International statistical review*, 69(1) :21–26, 2001.

[34] C. Cremona, L. Adélaide, Y. Berthaud, V. Bouteiller, V. L'hostis, S. Poyet, and J.-M. Torrenti. Probabilistic and predictive performance-based approach for assessing reinforced concrete structures lifetime : The applet project. In *EPJ Web of Conferences*, volume 12, page 01004. EDP Sciences, 2011.

[35] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4) :18–26, 2006.

[36] F. De Carvalho, E. Lima Neto, and C. Tenorio. A new method to fit a linear regression model for interval-valued data. advances in artificial intelligence. In *Proceedings of the Twenty Seventh Germany Conference on Artificial the International Intelligence, KI-04, Ulm (Germany), Biundo, S. et al. Eds, Vol. 1*. Springer, 2004.

[37] F. d. A. De Carvalho, Y. Lechevallier, and R. Verde. Clustering methods in symbolic data analysis. *Symbolic data analysis and the SODAS software*, pages 181–204, 2008.

[38] J. Dean and S. Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113, 2008.

[39] E. Diday. Une nouvelle méthode en classification automatique et reconnaissance des formes la méthode des nuées dynamiques. *Revue de statistique appliquée*, 19(2) :19–33, 1971.

[40] E. Diday. From data analysis to uncertainty knowledge analysis. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 153–160. Springer-Verlag, London, UK. ISBN 3-540-54659-6, 1991.

[41] E. Diday. De la statistique des données à la statistique des connaissances : avancées récentes en analyse des données symboliques. In *Conférence : Extraction et Gestion des Connaissances*, 2005.

[42] E. Diday. The state of the art in symbolic data analysis : overview and future. *Symbolic Data Analysis and the SODAS Software*, pages 3–41, 2008.

[43] E. Diday. Principal component analysis for bar charts and metabins tables. *Statistical Analysis and Data Mining : The ASA Data Science Journal*, 6(5) :403–430, 2013.

[44] E. Diday. Thinking by classes in data science : the symbolic data analysis paradigm. *Wiley Interdisciplinary Reviews : Computational Statistics*, 8(5) :172–205, 2016.

[45] E. Diday, F. Afonso, and R. Haddad. The symbolic data analysis paradigm, discriminant discretization and financial application. In *In Advances in Theory and Applications of High Dimensional and Symbolic Data Analysis, HDSDA 2013. Revue des Nouvelles Technologies de l'Information vol.RNTI-E-25*, pages 1–14, 2013.

[46] E. Diday, M. Noirhomme-Fraiture, et al. *Symbolic data analysis and the SODAS software*. J. Wiley & Sons, 2008.

[47] A. El Golli. *Extraction de données symboliques et cartes topologiques : Application aux données ayant une structure complexe*. PhD thesis, Université Paris Dauphine-Paris IX, 2004.

[48] F. Esposito, D. Malerba, and V. Tamma. Dissimilarity measures for symbolic objects. *Bock, HH, Diday, E.(eds.) : Analysis of Symbolic Data. Exploratory Methods for extracting Statistical Information from Complex Data, Series : Studies in Classification, Data Analysis, and Knowledge Organisation*, pages 165–185, 2000.

[49] U. M. Fayyad and K. B. Irani. On the handling of continuous valued attributes in decision tree generation. *Machine learning*, 8(1) :87–102, 1992.

[50] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *In proceedings of the 13^{th} International Joint Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufman Publishers, San Francisco, CA, 1993.

[51] W. D. Fisher. On grouping for maximum homogeneity. *Journal of the American statistical Association*, 53(284) :789–798, 1958.

[52] L. George. *HBase : the definitive guide*. O'Reilly Media, Inc., 2011.

[53] C. Gini. Measurement of inequality of incomes. *The Economic Journal*, 31(121) :124–126, 1921.

[54] A. Goel and K. Munagala. Complexity measures for map-reduce, and comparison to parallel computing. *arXiv preprint arXiv :1211.6526*, 2012.

[55] C. Guinot, D. Malvy, J.-F. Schémann, F. Afonso, R. Haddad, and E. Diday. Strategies evaluation in environmental conditions by symbolic data analysis : application in medicine and epidemiology to trachoma. *Advances in Data Analysis and Classification*, 9(1) :107–119, 2015.

[56] R. Haddad, F. Afonso, and E. Diday. Approche symbolique pour l'extraction de thématiques : Application un corpus issu d'appels tlphoniques. *Proceeding de la XVIIIème Rencontres de la Societe francophone de Classification*, pages 95–98, 2011.

[57] R. Haddad, F. Afonso, and S. Laaksonnen. Two-stage data mining for big statistical micro data. *New Techniques and Technologies for Statistics (NTTS)*, 2013.

[58] K. M. Hammouda and M. S. Kamel. Hierarchically distributed peer-to-peer document clustering and cluster summarization. *IEEE transactions on knowledge and data engineering*, 21(5) :681–698, 2009.

[59] G. Hébrail and Y. Lechevallier. Db2so : A software for building symbolic objects from databases. In *Data Analysis, Classification, and Related Methods*, pages 395–400. Springer, 2000.

[60] G. HUNAULT. Logiciels statistiques. http ://forge.info.univ-angers.fr/ gh/wstat/logiciels.pdf, 2004.

[61] R. Ihaka and R. Gentleman. R : a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3) :299–314, 1996.

[62] A. Irpino and R. Verde. Basic statistics for distributional symbolic variables : a new metric-based approach. *Advances in Data Analysis and Classification*, 9(2) :143–175, 2015.

[63] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, pages 119–127, 1980.

[64] R. Kerber. Chimerge : Discretization of numeric attributes. In *Proceedings of the tenth national conference on Artificial intelligence*, pages 123–128. Aaai Press, 1992.

[65] R. Kohavi and D. A. Sommerfield. Case studies : Public domain, multiple mining tasks systems : Mlc++. In *Handbook of data mining and knowledge discovery*, pages 548–553. Oxford University Press, Inc., 2002.

[66] K. Kubota, A. Nakase, H. Sakai, and S. Oyanagi. Parallelization of decision tree algorithm and its performance evaluation. In *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, volume 2, pages 574–579. IEEE, 2000.

[67] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing : design and analysis of algorithms*, volume 400. Benjamin/Cummings Redwood City, CA, 1994.

[68] L. A. Kurgan and K. J. Cios. Caim discretization algorithm. *IEEE transactions on Knowledge and Data Engineering*, 16(2) :145–153, 2004.

[69] J. Lafaye. Une méthode de discrétisation de variables continues. *Revue de statistique appliquée*, 27(2) :39–53, 1979.

[70] R. Lämmel. Google's mapreduce programming model-revisited. *Science of computer programming*, 70(1) :1–30, 2008.

[71] G. Lapresle. Les outils les plus utiles. Récupéré sur Cartographier à l'aide des TIC : http ://philippe.lapresle.pagesperso-orange.fr/outilsHG/carto.html, 2008.

[72] Y. Lechevallier. Recherche d'une partition optimale sous contrainte d'ordre total. Technical report, INRIA, 1990.

[73] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[74] L. Li and M. Zhang. The strategy of mining association rule based on cloud computing. In *2011 International Conference on Business Computing and Global Informatization*, pages 475–478. IEEE, 2011.

[75] N. Li, L. Zeng, Q. He, and Z. Shi. Parallel implementation of apriori algorithm based on mapreduce. In *Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on*, pages 236–241. IEEE, 2012.

[76] S. Liang, Y. Liu, C. Wang, and L. Jian. A cuda-based parallel implementation of k-nearest neighbor algorithm. In *Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009. CyberC'09. International Conference on*, pages 291–296. IEEE, 2009.

[77] M. Limam. *Méthodes de description de classes combinant classification et discrimination en analyse de données symboliques*. PhD thesis, Université Paris Dauphine-Paris IX, 2005.

[78] M. Malek and H. Kadima. Searching frequent itemsets by clustering data : Towards a parallel approach using mapreduce. In *Web Information Systems Engineering–WISE 2011 and 2012 Workshops*, pages 251–258. Springer Berlin Heidelberg, 2013.

[79] C. Mballo. *Ordre, codage et extension du critére de Kolmogorov-Smirnov pour la segmentation de données symboliques*. PhD thesis, Université Paris Dauphine-Paris IX, 2005.

[80] F. P. Miller, A. F. Vandome, and J. McBrewster. Amazon web services. 2010.

[81] J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58(302) :415–434, 1963.

[82] F. Nicolau and H. Bacelar-Nicolau. Clustering symbolic objects associated to frequency or probability laws by the weighted affinity coefficient. *Applied Stochastic Models and Data Analysis. Quantitative Methods in H. Bacelar-Nicolau, F. Nicolau and Jacques Janssen (Eds.) Business and Industry Society, INE, Lisboa, Portugal*, pages 155–158, 1999.

[83] M. Noirhomme-Fraiture and M. Rouard. Representation of sub-populations and correlation with zoom star. *Proceedings of NNTS'98*, 1998.

[84] R. Oldemar. Package 'rsda'. Récupéré sur cran.r-project : http ://cran.r-project.org/web/packages/RSDA/RSDA.pdf, 2015.

[85] R. Oldemar, J. Murillo, and J. Villalobos. An r package for symbolic data analysis. *XVIII SIMMAC*, 2012.

[86] S. Owen, R. Anil, T. Dunning, and E. Friedman. Mahout in action. 2012.

[87] E. Périnel. *Segmentation et analyse de données symboliques : application à des données probabilistes imprécises*. PhD thesis, Université Paris Dauphine-Paris IX, 1997.

[88] C. Quantin, L. Billard, M. Touati, N. Andreu, Y. Cottin, M. Zeller, F. Afonso, G. Battaglia, D. Seck, G. Le Teuff, et al. Classification and regression trees on aggregate data modeling : an application in acute myocardial infarction. *Journal of Probability and Statistics*, 2011, 2011.

[89] J. R. Quinlan. C4. 5 : Programming for machine learning. *Morgan Kauffmann*, page 38, 1993.

[90] J. R. Quinlan et al. *Discovering rules by induction from large collections of examples*. Expert systems in the micro electronic age. Edinburgh University Press, 1979.

[91] R. Rakotomalala. Tanagra : a free software for research and academic purposes. In *Proceedings of EGC*, volume 2, pages 697–702, 2005.

[92] R. Rakotomalala. Pratique de la régression logistique. *Régression Logistique Binaire et Polytomique, Université Lumière Lyon*, 2, 2011.

[93] M. Richeldi and M. Rossotto. Class-driven statistical discretization of continuous attributes. In *European Conference on Machine Learning*, pages 335–338, 1995.

[94] A. Roussot, E. Combier, G. Nuemi, J. Amat-Roze, and C. Quantin. Analyse spatiale des trajectoires de prise en charge des patients atteints de cancer primitif du poumon en région bourgogne. *Journal de gestion et d'économie médicales*, 30(2) :96–109, 2012.

[95] L. Ruschendorf. Wasserstein metric. *Encyclopedia of Mathematics (ed. M. Hazewinkel).*, 2001.

[96] R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2) :197–227, 1990.

[97] D. Seck. *Arbres de décision symbolique, outils de validation et d'aide à l'interprétation*. PhD thesis, Université Paris Dauphine-Paris IX, 2012.

[98] J. Shafer, R. Agrawal, and M. Mehta. Sprint : A scalable parallel classi er for data mining. In *Proc. 1996 Int. Conf. Very Large Data Bases*, pages 544–555. Citeseer, 1996.

[99] V. Stéphan. *Construction d'objets symboliques par synthèse des résultats de requêtes SQL*. PhD thesis, Université Paris Dauphine-Paris IX, 1998.

[100] R. C. Team. R : A language and environment for statistical computing. r foundation for statistical computing, vienna, austria. 2013, 2014.

[101] S. Thomas and S. Sarawagi. Mining generalized association rules and sequential patterns using sql queries. In *KDD*, pages 344–348, 1998.

[102] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive : a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2) :1626–1629, 2009.

[103] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines : Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr) :363–392, 2005.

[104] T. Van de Merckt. Decision trees in numerical attribute spaces. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 13, pages 1016–1016. LAWRENCE ERLBAUM ASSOCIATES LTD, 1993.

[105] M. Vrac and E. Diday. Description symbolique de classes. In *EGC*, pages 105–116, 2001.

[106] C. S. Wallace and J. Patrick. Coding decision trees. *Machine Learning*, 11(1) :7–22, 1993.

[107] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301) :236–244, 1963.

[108] M. Ware. Weka documentation. *University of Waikoto*, 2000.

[109] T. White. *Hadoop : The definitive guide*. O'Reilly Media, Inc., 2012.

[110] M. J. Zaki, C.-T. Ho, and R. Agrawal. Parallel classification for data mining on shared-memory multiprocessors. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 198–205. IEEE, 1999.

[111] F. Zhang, Y. Zhang, and J. D. Bakos. Accelerating frequent itemset mining on graphics processing units. *The Journal of Supercomputing*, 66(1) :94–117, 2013.

[112] W. Zhao, H. Ma, and Q. He. Parallel k-means clustering based on mapreduce. In *IEEE International Conference on Cloud Computing*, pages 674–679. Springer, 2009.

[113] W.-Z. Zhao, H.-F. Ma, Y.-X. Fu, and Z.-Z. Shi. Research on parallel k-means algorithm design based on hadoop platform. *Computer Science*, 38(10) :166, 2011.

[114] X. Zhengqiao and Z. Dewei. Research on clustering algorithm for massive data based on hadoop platform. In *Computer Science & Service System (CSSS), 2012 International Conference on*, pages 43–45. IEEE, 2012.

[115] D. A. Zighed, S. Rabaséda, and R. Rakotomalala. Fusinter : a method for discretization of continuous attributes. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(03) :307–326, 1998.

[116] D. A. Zighed and R. Rakotomalala. *Graphes d'induction : apprentissage et data mining*. Hermes Paris, 2000.

[117] D. A. Zighed, R. Rakotomalala, and S. Rabaseda. A discretization method of continuous attributes in induction graphs. *CYBERNETICS AND SYSTEMS RESEARCH*, pages 997–1002, 1996.

Résumé

Cette thèse a pour but l'enrichissement des méthodes supervisées d'analyse de données symboliques et l'extension de ce domaine aux données volumineuses, dites "Big Data". Nous proposons à cette fin une méthode supervisée nommée HistSyr. HistSyr convertit automatiquement les variables continues en histogrammes les plus discriminants pour les classes d'individus. Nous proposons également une nouvelle méthode d'arbres de décision symbolique, dite SyrTree. SyrTree accepte plusieurs types de variables explicatives et à expliquer pour construire l'arbre de décision symbolique. Enfin, nous étendons HistSyr aux Big Data, en définissant une méthode distribuée nommée CloudHistSyr. CloudHistSyr utilise Map/Reduce pour créer les histogrammes les plus discriminants pour des données trop volumineuses pour HistSyr. Nous avons testé CloudHistSyr sur Amazon Web Services (AWS). Nous démontrons la scalabilité et l'efficacité de notre méthode sur des données simulées et sur les données expérimentales. Nous concluons sur l'utilité de CloudHistSyr qui , grâce à ses résultats, permet l'étude de données massives en utilisant les méthodes d'analyse symboliques existantes.

Mots Clés

Analyse de Données Symboliques (ADS), Histogrammes, Arbres de décision symboliques, Big Data, Map/Reduce, Hadoop, AWS.

Abstract

This Thesis proposes new supervised methods for Symbolic Data Analysis (SDA) and extends this domain to Big Data. We start by creating a supervised method called HistSyr that converts automatically continuous variables to the most discriminant histograms for classes of individuals. We also propose a new method of symbolic decision trees that we call SyrTree. SyrTree accepts many types of inputs and target variables and can use all symbolic variables describing the target to construct the decision tree. Finally, we extend HistSyr to Big Data, by creating a distributed method called CloudHistSyr. Using the Map/Reduce framework, CloudHistSyr creates of the most discriminant histograms for data too big for HistSyr. We tested CloudHistSyr on Amazon Web Services. We show the efficiency of our method on simulated data and on actual car traffic data in Nantes. We conclude on overall utility of CloudHistSyr which, through its results, allows the study of massive data using existing symbolic analysis methods.

Keywords

Symbolic Data Analysis (SDA), Histograms, Symbolic decision trees, Big Data, Map/Reduce, Hadoop, AWS.