UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE DU PROGRAMME DE

MAÎTRE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PRÉSENTÉ PAR

ABOUBACAR BANGOURA

SYSTÈMES DE CLASSIFICATION BASÉS SUR DES RÈGLES FLOUES (FRBCS)

DEPARTEMENT DE MATHÉMATIQUES ET D'INFORMATIQUE

CANADA

MAI 2025

Résumé

Les systèmes de classification basés sur des règles floues (FRBCS) sont des modèles d'intelligence artificielle qui ont la capacité d'expliquer leur structure interne d'une manière compréhensible pour les humains, ce qui en fait un candidat potentiel pour l'intelligence artificielle explicable. En raison de leur qualité de transparence, les FRBCS sont utilisés dans de nombreux domaines, notamment dans des secteurs critiques, comme la finance, la médecine et la robotique. Cependant, l'un des points présents dans toute analyse est le nombre de règles obtenues par un classificateur flou. Dans la plupart des travaux, un nombre arbitraire de règles à retenir est défini, ou les règles sont sélectionnées sans aucune mesure raisonnable. Le problème d'une telle approche est qu'elle peut conduire à la perte de certaines règles importantes, ce qui peut affecter les performances du système. Par ailleurs, bien que certaines propositions soient raisonnables, elles souffrent du problème de bon compromis interprétabilité-précision.

Dans notre mémoire, nous proposons une méthode de minimisation d'une base de règles floues à l'aide de l'algorithme de Quine McCluskey, qui est l'une des techniques les plus populaires pour simplifier une fonction booléenne. Pour parvenir à un bon compromis précision-interprétabilité, notre méthodologie est répartie en quatre étapes. Premièrement, nous introduisons une phase de prétraitement de données dans le but d'obtenir un ensemble de données équilibré avec des caractéristiques informatives. Dans la deuxième étape, nous extrayons les règles floues à partir de données prétraitées pour obtenir une base de règles initiale. Ensuite, la troisième étape consiste à minimiser le nombre de règles dans une base de règles du système par l'algorithme de Quine McCluskey. Enfin, nous évaluons la capacité de minimisation et les performances de classification de notre système dans la quatrième et dernière étape.

Par une étude expérimentale, nous avons démontré que notre approche est capable de simplifier considérablement une base de règles floues sans altérer les performances du système.

Abstract

Fuzzy rule-based classification systems (FRBCS) are artificial intelligence models that have the ability to explain their internal structure in a way that is understandable to humans, making them a potential candidate for explainable artificial intelligence. Because of their transparent quality, FRBCS are used in many fields, including critical sectors such as finance, medicine, and robotics. However, one of the points present in any analysis is the number of rules obtained by a fuzzy classifier. In most works, an arbitrary number of rules to be retained is defined, or rules are selected without any reasonable measure. The problem with such an approach is that it can lead to the loss of some important rules, which can affect system performance. On the other hand, although some proposals are reasonable, they suffer from the problem of interpretability-accuracy compromise.

In our dissertation, we propose a method for minimizing a fuzzy rule base using the Quine McCluskey algorithm, which is one of the most popular techniques for simplifying a Boolean function. To achieve a good interpretability-accuracy compromise, our methodology is divided into four stages. Firstly, we introduce a data pre-processing phase with the aim of obtaining a balanced dataset with informative features. In the second step, we extract fuzzy rules from pre-processed data to obtain an initial rule base. Then, in the third step, we minimize the number of rules in the system's rule base using Quine McCluskey's algorithm. Finally, we evaluate the minimization capability and classification performance of our system in the fourth and last step.

By experimental study, we have demonstrated that our approach is capable of considerably simplifying a fuzzy rule base without altering the system's performance.

Remerciements

Ce document assemble quelques années de travaux menés à l'UQTR. Je dois tout d'abord exprimer mes premiers remerciements envers les enseignants du département de mathématiques et d'informatique (DMI) auprès desquels j'ai acquis les compétences nécessaires pour mener à bien ce projet.

J'aimerais exprimer mes remerciements les plus sincères à mon directeur de recherche, Ismaïl Biskri, pour avoir accepté de me diriger dans le cadre de ma maîtrise. Son soutien et ses directives ont toujours été très instructifs.

Je tiens particulièrement à exprimer ma gratitude à Ayush Varshney, du département informatique de l'Université d'Umeå, et à Leonardo Jara, du département informatique et intelligence artificielle de l'Université de Grenade. Ils ont bien voulu me fournir des détails sur leurs travaux utilisés dans le présent projet et m'ont généreusement fait part de leur expérience.

Je voulais aussi exprimer ma reconnaissance envers mes collègues de l'UQTR, en particulier ceux avec lesquels j'ai eu l'opportunité de travailler sur les projets et partager des connaissances. Je profite de l'occasion pour remercier les responsables de la gestion des laboratoires DMI pour leur disponibilité et leur aide.

Merci également à toutes les personnes que j'ai côtoyées durant toutes ces années passées à l'UQTR et avec qui j'ai eu tant de plaisir à travailler et à partager.

Enfin, je voudrais adresser mes remerciements les plus profonds à mes parents, à mes proches, et en particulier à mon frère, qui m'ont permis de vivre cette expérience, qui m'ont apporté un soutien inconditionnel et une aide très précieuse pendant mes études.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1 : Introduction

Chapitre 1

Introduction

Au fil de ces dernières années, les avancées technologiques en matière d'intelligence artificielle, en particulier les méthodes d'apprentissage automatique traditionnelles et profondes, ont connu une croissance fulgurante et significative dans la société humaine. Ces outils se distinguent par leur capacité à développer des machines capables de simuler, de remplacer, d'étendre et d'élargir une partie de l'intelligence humaine.

La disponibilité des données en masse depuis l'avènement d'Internet a considérablement augmenté la popularité des techniques d'exploration des données, qui ne cessent de croître. Ces dernières utilisent différents formats de données, notamment numériques, textuelles, d'images ou d'audio, pour apprendre et acquérir des connaissances afin d'effectuer des tâches comme le traitement automatique de données, la classification, la reconnaissance et tant d'autres.

Par ses promesses diverses et ambitieuses, l'intelligence artificielle motive une adoption à grande échelle dans divers secteurs d'activités. Bien que ces technologies offrent plusieurs opportunités, elles peuvent aussi avoir des conséquences négatives, voire irrémédiables, sur la vie humaine si elles sont exploitées de manière irrationnelle. Pour prévenir ces fléaux, quelques principes fondamentaux en ce qui concerne la normalisation de l'intelligence artificielle ont été dictés par les organisations et les experts du domaine. Nous retrouvons largement dans la littérature les principes d'interprétabilité, d'explicabilité, de transparence et autres, qui diffèrent d'un groupe à l'autre [1]. L'interprétabilité et l'explicabilité sont des facteurs les plus populaires [2]. Ces derniers ont été soulignés comme des principes très importants dans plusieurs études.

Une intelligence artificielle interprétable a la capacité de décrire la structure interne du système de manière compréhensible pour les humains [2]. Récemment, plusieurs recherches ont été entreprises sur l'intelligence artificielle explicable. Pendant que certains travaux essaient d'améliorer la transparence des modèles en boîte noire, d'autres cherchent à optimiser l'efficacité des modèles en boîte blanche.

Le système basé sur des règles est le modèle en boîte blanche qui couvre plus de principes d'intelligence artificielle explicables en termes de simplicité, d'interprétabilité, de précision et de transparence [3, 4].

Parmi ces derniers, les modèles basés sur des règles floues sont très recherchés, avec plus de 45 187 travaux dans la littérature. Ces systèmes utilisent les règles IF-THEN, qui sont formées par des variables floues linguistiques comme antécédents et conséquents pour représenter les connaissances compréhensibles par l'homme [5]. Initialement introduits par le Professeur Lotfi Zadeh [4], les systèmes basés sur la logique floue tentent d'imiter la pensée humaine, sauf qu'au lieu d'essayer de représenter l'architecture du cerveau comme vous le feriez avec un réseau neuronal, l'accent est mis sur la façon dont les humains pensent de manière approximative plutôt que précise. Cette méthode a été développée dans le but de créer une logique basée sur des règles capables de gérer l'incertitude et les valeurs de sortie dans toute la plage $\{0,1\}$.

Les systèmes basés sur des règles floues sont très recommandés dans les domaines critiques, notamment en finance [4] et en santé [3]. Cependant, ces systèmes souffrent du nombre élevé de règles qui affecte considérablement leur interprétabilité [5, 6]. Plusieurs tentatives de recherche ont été menées afin d'obtenir un nombre minimum possible de règles dans ces systèmes. Les méthodes proposées atteignent leurs objectifs en ce qui concerne la réduction du nombre de règles. Par contre, la plupart de ces solutions sont focalisées sur un choix arbitraire du nombre de règles à sélectionner, ce qui peut éliminer les règles importantes.

Le travail de cette recherche a pour objectif de développer une approche permettant de trouver un bon compromis entre précision et interprétabilité dans un système de classification basé sur des règles floues. Pour atteindre notre objectif, nous avons d'abord prétraité les données, puis nous avons extrait des règles floues à partir de ces informations. Ensuite, nous avons appliqué la méthode de Quine McCluskey pour simplifier la base de règles initialement générée.

Avant d'appliquer la méthode de Quine McCluskey à notre problématique, nous avons d'abord défini un codage approprié de l'ensemble des règles floues pour binariser les variables linguistiques, comme l'exige cette dernière. Cela nous a permis de modifier un ensemble de règles floues de base en un ensemble de règles floues étendues et courtes, comme les modèles utilisés dans [6] et [7].

Notre approche se distingue par sa capacité à minimiser de manière significative une base de règles. De plus, un ensemble de données nommé «Don't Care (DC)» est ajouté dans le cas où la taille de la base de règles initiale ne deviendrait pas trop grande. Cela élimine les variables de moindre importance, ce qui améliore davantage l'interpretabilité des règles.

Dans nos études d'expérimentations, nous avons utilisé les données financières suivantes : l'authentification des billets de banque, la vérification des enchères, le désabonnement des clients d'une société de télécommunication et la faillite d'entreprise. Dans chaque étude expérimentale, nous avons effectué une évaluation approfondie sur notre système à l'état initial et final. Ce qui nous a permis d'examiner les résultats du système avec la base de règles initiale et celles obtenues après les opérations de minimisation. Ensuite, nous en avons tiré des conclusions sur l'efficacité de notre méthode pour simplifier la base de règles du système, comme suit :

- Si les performances avant et après les opérations de réduction de la base de règles du système sont comparables, alors les données informatives ont été conservées dans la base de règles minimisée.
- Dans le cas où les performances après la réduction de la base du système diminuent considérablement par rapport aux performances avant les opérations de simplification, nous avons conclu que les données informatives ont été perdues dans le processus de simplification.

Le reste de notre mémoire est organisé de la manière suivante : dans le **deuxième chapitre**, nous présenterons les algorithmes que nous avons exploités dans nos opérations de prétraitement des données, soit l'analyse en composantes principales (ACP), la technique de suréchantillonnage minoritaire synthétique (SMOTE) et le modèle du mélange de Gaussien (GMM).

Dans **le troisième chapitre**, les concepts fondamentaux du système de classification basé sur des règles floues, avec ses composants, tels que la base de connaissances, le module de fuzzification et le moteur d'inférence, seront expliqués en détail. Nous présenterons également deux modèles d'algorithmes de génération de règles floues, avec des exemples d'illustration.

Le **quatrième chapitre** mettra de l'avant des revues littéraires sur la minimisation du nombre de règles dans une base de règles floues.

Dans le **cinquième chapitre**, nous rapporterons les détails concernant l'algorithme de minimisation des fonctions booléennes, soit la méthode de Quine McCluskey. Nous allons d'abord aborder les concepts et les principes fondamentaux de l'algorithme, puis nous présenterons un exemple de fonctionnement de la méthode.

Le **sixième chapitre** présentera notre méthodologie de minimisation du nombre de règles floues. En premier lieu, nous aborderons nos stratégies de prétraitement des données. Ensuite, notre approche pour appliquer l'algorithme de Quine McCluskey à une base de règles floues sera discutée. Enfin, nous présenterons les métriques d'évaluation des performances du système et nous terminerons la discussion sur son implémentation.

Dans le **septième chapitre**, nous examinerons de près deux groupes de métriques que nous avons nommés « Métriques globales », qui évaluent les performances du système avec un ensemble de données, et « Métriques spécifiques », qui mesurent les performances du système avec une classe de données.

Le **huitième** et **dernier chapitre** présentera la conclusion des travaux de recherche de notre mémoire et évoquera les perspectives à venir.

Chapitre 2 : L'ACP, SMOTE et GMM

Chapitre 2

L'ACP, SMOTE et GMM

2.1 Introduction

L'analyse en composantes principales (ACP), la technique de suréchantillonnage minoritaire synthétique (SMOTE) et le modèle de mélange de Gaussien (GMM) sont des algorithmes d'apprentissage automatique très répandus dans le traitement de données. Dans ce chapitre, nous introduisons les concepts fondamentaux et les principes de fonctionnement de chacun de ces algorithmes avec des exemples d'illustration.

2.2 L'analyse en composantes principales (ACP)

L'analyse en composantes principales est une méthode statistique de réduction de dimensions des données multivariées en conservant autant que possible la variation présente dans l'ensemble de données. L'ACP effectue une transformation linéaire des facteurs d'origine en un nouvel ensemble de facteurs, les composantes principales, qui ne sont pas corrélés et qui sont ordonnés de telle sorte que les premiers conservent la majeure partie de la variation présente dans toutes les variables d'origine [8].

Admettons une matrice de données multivariées X de n exemples et de p variables sur laquelle nous souhaitons effectuer une analyse en composantes principales. La matrice F des nouveaux facteurs $f_1, \ldots, f_p$, extraite par l'ACP, peut être représentée comme suit : $\quad F = X^s V$

Où X^s est la matrice des scores standardisés de X, V est une matrice des vecteurs propres qui définissent les directions des composantes principales avec lesquelles la variation de X est maximale.

L'idée principale de l'ACP est de construire de nouveaux facteurs dans le but d'atteindre les objectifs suivants :

- Chaque facteur f_j est une combinaison linéaire des facteurs initiaux $x_1^s, \ldots, x_p^s$ de la matrice de scores standardisés X^s, c'est-à-dire : $\quad f_j = v_{1j}x_1^s + \cdots + v_{pj}x_p^s = \sum_{k=1}^{p} v_{kj}x_k^s; \quad j = 1, \ldots, p;$
- Les nouveaux facteurs f_j sont non corrélés, c'est-à-dire : $\quad cov(f_i, f_j) = 0, \quad pour \quad tous \quad i \neq j;$

Pour atteindre ces objectifs, nous devons trouver la matrice V des vecteurs propres de telle sorte que la variance de F soit la matrice diagonale $\Lambda = (\lambda_1, \ldots, \lambda_p)$. Une approche simple et largement utilisée pour déterminer la matrice V consiste à effectuer la décomposition en valeurs propres (EDV) sur la matrice de corrélation R de X définie par la relation suivante [9] :

$$R = X^T X = V \Lambda V^T \tag{2.1}$$

Où Λ est la matrice diagonale dont la j-ième entrée est la j-ième valeur propre λ_j. Les valeurs propres sont positives et réelles et sont classées en ordre décroissant de valeur, de sorte que $\lambda_1 > \lambda_2 > \cdots > \lambda_p$.

Une méthode alternative à la décomposition en valeurs propres (EDV) consiste à effectuer une décomposition en valeurs singulières (SVD) sur X. La SVD factorise une matrice de données en trois autres matrices (U, Λ, V^T) et trouve les fonctions de base de l'ACP dans la matrice V tout en évitant de calculer la matrice de corrélation R [9, 10]. La SVD sur un ensemble de données X peut être définie par la relation suivante :

$$X = U \Sigma V^T \tag{2.2}$$

Où U est la matrice orthogonale dont les colonnes sont des vecteurs singuliers gauches correspondant aux vecteurs propres de XX^T, Σ est une matrice diagonale de valeurs singulières classées par ordre de valeur décroissante. La matrice V est la même que celle dans l'équation (2.1), dont les colonnes sont des vecteurs singuliers droits ou des vecteurs propres de $X^T X = R$.

Exemple de décomposition en valeurs singulières avec la matrice de données suivante :

$$\mathrm{X}=\begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}_{(2\times 3)}$$

Trouvons les valeurs propres et les vecteurs propres correspondants de la matrice XX^T.

$$\underset{(2\times 3)}{\mathrm{XX}^T} = \underset{(2\times 3)}{\begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}} \underset{(3\times 2)}{\begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}} = \underset{(2\times 2)}{\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}}$$

Les valeurs propres et les vecteurs propres d'une matrice A sont définis par la relation $A\overrightarrow{u} = \lambda\overrightarrow{u}$, et en appliquant cette dernière sur XX^T, nous avons :

$$\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Ce qui nous donne les équations suivantes :

$$(11-\lambda)x_1 + x_2 = 0$$
$$x_1 + (11-\lambda)x_2 = 0$$

Résolvons λ en mettant le déterminant de la matrice des coefficients à 0 :

$$\begin{bmatrix} (11-\lambda) & 1 \\ 1 & (11-\lambda) \end{bmatrix} = \lambda^2 - 22\lambda + 120 = 0$$

Ce qui donne les valeurs propres $\lambda = 10$ et $\lambda = 12$.

Pour $\lambda = 12$:

$$(11-12)x_1 + x_2 = 0$$
$$x_1 = x_2$$

Cela signifie que si $x_1 = 1$, alors $x_2 = 1$. Donc, $u_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. La norme de u_1 est $\overrightarrow{u}_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$.

Pour $\lambda = 10$:

$$(11-10)x_1 + x_2 = 0$$
$$x_1 = -x_2$$

Nous pouvons prendre $x_1 = 1$ et $x_2 = -1$. Avec $u_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ et sa norme $\overrightarrow{u}_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$.

Les colonnes de la matrice U sont les vecteurs propres $\overrightarrow{u}_1$ et $\overrightarrow{u}_2$.

$$\mathrm{U}=\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}_{(2\times 2)}$$

Nous devons trouver également les valeurs propres et les vecteurs propres correspondants de la matrice $\mathrm{X}^T\mathrm{X}$, comme suit :

$$\mathrm{X}^T\mathrm{X}=\underbrace{\begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}}_{(3\times 2)} \underbrace{\begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}}_{(2\times 3)} = \underbrace{\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}}_{(3\times 3)}$$

En appliquant la même procédure, nous avons obtenu les valeurs propres $\lambda = 0$, $\lambda = 10$ et $\lambda = 12$. Ce qui permet d'obtenir la matrice V.

$$\mathrm{V}=\underbrace{\begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{30}} & 0 & -\frac{5}{\sqrt{30}} \end{bmatrix}}_{(3\times 3)}$$

La matrice Σ contient en diagonale la racine carrée des valeurs propres, par ordre décroissant.

$$\text{Et} \quad \Sigma=\underbrace{\begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}}_{(2\times 3)}$$

L'ACP basée sur la SVD peut être appliquée d'une manière plus économique, particulièrement avec les données de grandes dimensions en utilisant sa version compacte [10]. Pour ce faire, il faut déterminer le rang de la matrice de données X. Le rang d'une matrice de données peut être obtenu en prenant le nombre de valeurs singulières non nulles de la matrice Σ de la SVD [11]. La SVD compacte est définie comme suit :

$$\begin{aligned} r &= rank(X); \quad r \leq min\{n, p\} \qquad (a) \\ X &= U_r \Sigma_r V_r^T \qquad (b) \end{aligned} \tag{2.3}$$

L'ACP permet non seulement de découvrir de nouvelles composantes non corrélées, mais aussi de sélectionner un sous-ensemble de composants qui conserve la grande partie de la variance totale expliquée [12]. De nombreux critères sont utilisés pour effectuer cette opération, dont celui de Jolliffe [12], qui consiste à choisir un sous-ensemble dont la proportion de la variance cumulative expliquée correspond à une proportion fixe, par exemple 90%. La valeur propre λ_j correspondante à la valeur singulière σ_j, la variance w_j expliquée par le nouveau facteur f_j et la variance cumulative w_k expliquée par les « k » premiers facteurs sont déterminées respectivement par les équations suivantes [13] :

$$\begin{aligned} \lambda_j &= \sigma_j^2/(n-1); \quad o \quad \sigma_j \in \Sigma \qquad (a) \\ w_j &= \frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j} \qquad (b) \\ w_k &= \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j} \qquad (c) \end{aligned} \tag{2.4}$$

De plus, il est recommandé de standardiser les données avant d'appliquer l'ACP dans l'objectif d'avoir une moyenne nulle et une variance unitaire pour chaque variable [14]. Cela permet d'assurer que toutes les variables aient la même échelle et la même importance pendant les opérations d'extraction des nouvelles caractéristiques. La méthode la plus classique pour effectuer une mise à l'échelle des données est la technique du z-score. La valeur standardisée z_{ij} s'obtient en soustrayant à la valeur x_{ij} la moyenne et en divisant par l'écart-type de la variable x_j, comme suit :

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \tag{2.5}$$

$$\text{Avec} \quad \bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \quad \text{et} \quad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)$$

Où $\bar{x}_j$ et s_j sont respectivement la moyenne et l'écart type de la variable x_j.

Exemple d'illustration de l'ACP basée sur la SVD avec les données du tableau 2.1[a] :

Tout d'abord, les données sont normalisées en appliquant l'équation (2.5) (voir le tableau 2.1[b]). Par la suite, la SVD est appliquée aux données standardisées de la manière suivante :

$$\mathrm{X}^s = \underset{(6\times 3)}{\begin{bmatrix} 0.12558084 & -0.60533436 & 0.28333938 \\ -0.55483066 & -0.38337625 & 0.1302568 \\ -0.474585 & 0.47119796 & -0.15595424 \\ -0.00516422 & 0.43580715 & 0.19146771 \\ 0.31666777 & -0.14795889 & -0.83320933 \\ 0.59233126 & 0.2296644 & 0.38409968 \end{bmatrix}} \underset{(3\times 3)}{\begin{bmatrix} 3.45039668 & 0. & 0. \\ 0. & 1.74165799 & 0. \\ 0. & 0. & 0.24777044 \end{bmatrix}} \underset{(3\times 3)}{\begin{bmatrix} 0.47246133 & 0.63001172 & 0.61633231 \\ 0.87872325 & -0.2827555 & -0.38457091 \\ 0.06801283 & -0.72328042 & 0.68719698 \end{bmatrix}}$$

Notons que nous avons utilisé la version compacte de SVD dans l'exemple ci-dessus avec le rang de la matrice de données d'origine égal à 3 (nombre de valeurs propres positives et non nulles).

La matrice des nouveaux facteurs s'obtient en projetant les données standardisées sur le transposé de la matrice des vecteurs singuliers droits. Les données de la matrice F sont arrondies à 4 chiffres près (voir le tableau 2.1[c]).

TABLEAU 2.1 : DONNÉES DE SIMULATION ACP-SVD

[a] X

num	x_1	x_2	x_3
1	0.38	0.44	0.48
2	0.24	0.28	0.30
3	0.56	0.26	0.25
4	0.63	0.36	0.37
5	0.50	0.46	0.45
6	0.75	0.50	0.53

[b] X^s

x_1	x_2	x_3
-0.71693115	0.52031489	0.72074997
-1.48901084	-1.04062977	-0.90093746
-0.05514855	-1.23574785	-1.35140619
0.6617826	-0.26015744	-0.27028124
0.27574275	0.91055105	0.63065622
1.32356519	1.10566913	1.1712187

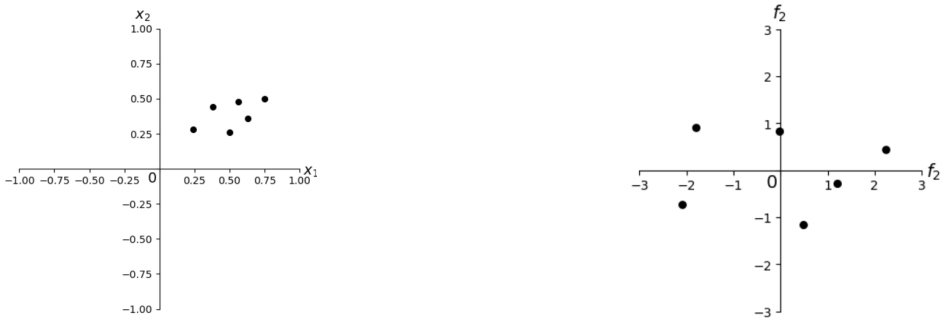
[c] F

f_1	f_2	f_3
0.4333	-1.0542	0.0702
-1.9143	-0.6677	0.0322
-1.6375	0.8206	-0.0386
-0.0178	0.7590	0.0474
1.0926	-0.2576	-0.2064
2.0437	0.3999	0.0951

Les figures 2.1[a] et 2.1[b] présentent respectivement les données d'origine du tableau 2.1[a] et les données des composantes principales du tableau 2.1[c].

FIGURE 2.1 : Visualisation des données de simulation ACP-SVD

[a] Origine **[b] PCA-SVD**



2.2.1 Algorithme d'analyse en composantes principales ACP

L'algorithme d'ACP est principalement composé d'une étape de standardisation des données qui permet de ramener les données d'origine sur la même échelle de mesure avec une moyenne nulle et une variance unitaire, une étape de détermination des vecteurs propres et des valeurs propres des données standardisées et une étape de projection des données standardisées sur les vecteurs propres pour la formation des nouveaux facteurs appelés composantes principales.

Algorithme 2.1 : Analyse en composantes principale basée sur la SVD

1: X : ensemble de données d'entrée
2: t% : proportion de sélection de la variance expliquée
3: X^s = X standardisées en utilisant (2.5)
4: Appliquer la SVD sur X^s en utilisant (2.2)
5: Déterminer le rang r de X en utilisant (2.3) (a)
6: **Pour** j de 1 à r **Faire**
7: Calculer λ_j en utilisant (2.4) (a)
8: Calculer w_j en ulisant (2.4)(b)
9: **Fin Pour**
10: Trouver les k premiers CP correspondants à t% en utilisant (2.4) (c)
11: Sélectionner V_k à partir de V^T
12: Déterminer la matrice F des composantes principales en projetant X^s sur V_k

2.3 Technique de suréchantillonnage minoritaire synthétique (SMOTE)

La technique de suréchantillonnage minoritaire synthétique (SMOTE) est un algorithme d'échantillonnage de données qui s'appuie sur l'algorithme des K-voisins les plus proches (KNN). SMOTE est l'une des techniques d'équilibrage des données les plus populaires [15]. Supposons que nous voulons équilibrer une matrice de données X composée de M échantillons pour la classe majoritaire et de N échantillons pour la classe minoritaire. Pour échantillonner un exemple dans la classe minoritaire, SMOTE commence à sélectionner un échantillon minoritaire aléatoire X_{j_min}, $j\_min = 1,\ldots,N$. Ensuite, un élément des K-voisins les plus proches X_{j_k}, $j\_k = 1,\ldots,N_K$ de l'échantillon X_{j_min} est sélectionné. Le nouvel exemple de la classe minoritaire est échantillonné entre l'échantillon minoritaire aléatoire X_{j_min} et l'échantillon des voisins les plus proches X_{j-k} par l'équation suivante [15, 16] :

$$X_{j\_new} = X_{j\_min} + rand(0,1) * (X_{j\_k} - X_{j\_min}) \tag{2.6}$$

2.3.1 Algorithme d'échantillonnage de données SMOTE

SMOTE est un algorithme qui échantillonne les données en utilisant l'algorithme des K-voisins les plus proches (KNN) pour trouver l'échantillon le plus proche de l'échantillon aléatoire sélectionné. Ensuite, un exemple est échantillonné à l'image de ces derniers.

Algorithme 2.2 : Algorithme d'échantillonnage de données SMOTE

1: X : ensemble de données déséquilibrés
2: X_{min} : échantillons de données de classe minoritaire de X
3: K : nombre de voisions les plus proches
4: $N_{X_{syn}}$: nombre d'exemples à synthétiser
5: **Pour** j de 1 à $N_{X_{syn}}$ **Faire**
6: Sélectionner un échantillon aléatoire X_{j_min} de X_{min}
7: Trouver les K voisins les plus proches de X_{j_min} avec la distance euclidienne
8: Sélectionner un échantillon aléatoire X_{j_k} des K voisins les plus proches
9: Génerer le nouvel échantillon X_{j_new} entre X_{j_min} et X_{j_k}
10: Ajouter X_{j_new} dans l'ensemble de données X
11: **Fin Pour**

2.4 Modèle de mélange de Gaussien (GMM)

Le modèle de mélange de Gaussien est un algorithme d'apprentissage automatique non supervisé et largement utilisé dans le problème de clustering. Une distribution issue d'un modèle de mélange de Gaussien est une fonction de densité de probabilité paramétrique représentée comme une somme pondérée des densités des composantes gaussiennes [17] :

$$P(x_i|\Theta) = \sum_{k=1}^{K} \pi_k p_k(x_i|\Theta_k) \tag{2.7}$$

Où x_i est un vecteur de données, π_k est le coefficient du k-ième composant de mélange, $\Theta_k = (\mu_k, \Sigma_k)$ est le paramètre du k-ième composant de mélange avec une moyenne μ_k et une matrice de covariance Σ_k, $p_k(x_i|\Theta_k)$ est la distribution gaussienne (normale) avec p dimensions du k-ième composant de mélange définie par [16] :

$$P_k(x_i|\Theta_k) = \frac{1}{\sqrt{(2\pi)^p|\Sigma|}} \exp\Big(-\frac{1}{2}(x_i-\mu_k)^T\Sigma^{-1}(x_i-\mu_k)\Big) \tag{2.8}$$

La probabilité postérieure qu'un échantillon de données x_i appartienne au k-ième composant de mélange est donnée par la relation suivante [16] :

$$P_{ik} = \frac{\pi_k p(x_i|\Theta_k)}{\sum_{k=1}^{K} \pi_k p(x_i|\Theta_k)} \tag{2.9}$$

La classe de l'échantillon x_i est déterminée comme suit [16] :

$$Class(x_i) = argmax(P_{ik});\ \ i = 1,\ldots,n;\ \ k = 1,\ldots,K; \tag{2.10}$$

Dans l'équation (2.10), pour obtenir les résultats les plus probables compte tenu du modèle, les paramètres de $\Theta = (\mu, \Sigma)$ peuvent être estimés puis maximisés [18]. L'un des estimateurs de paramètres des modèles statistiques les plus utilisés est la méthode d'estimateur du maximum de vraisemblance [18]. La fonction du maximum de vraisemblance de la distribution gaussienne est définie par l'équation suivante [19] :

$$L(\pi, \Theta|X) = \prod_{i}^{n} \sum_{k}^{K} \pi_i p(x_i|\Theta_k) \tag{2.11}$$

L'idée derrière l'estimateur du maximum de vraisemblance est de traiter la fonction L(π, Θ|X) et de trouver la valeur de Θ qui la maximise [18]. Cela peut être fait en résolvant la dérivée partielle de la fonction log de vraisemblance définie par l'équation (2.11). Pourtant, résoudre la dérivée partielle de la fonction log de vraisemblance par une méthode analytique est difficile, voire impossible [18, 19]. La solution la plus courante est l'utilisation de l'algorithme Espérance-Maximisation (EM) [18].

EM est un algorithme itératif pour l'estimation des paramètres des modèles statistiques lorsque certaines des variables aléatoires impliquées ne sont pas observées ou lorsque la dérivée partielle de l'équation (2.11) n'a pas de solution analytique [18]. En supposant que les valeurs des paramètres initiaux sont connues, la première étape consiste à initialiser les paramètres π, μ, Σ de chaque composant de mélange. Ensuite, pour chaque composant k, k = 1,..., K, l'algorithme EM maximise ses paramètres en quatre étapes suivantes :

- **Étape E :** à ce niveau, l'équation (2.9) est utilisée pour évaluer la probabilité postérieure de chaque échantillon dans chaque composant de mélange en utilisant les paramètres actuels.
- **Étape M :** dans cette étape, les paramètres π_k, μ_k, Σ_k sont recalculés en utilisant la probabilité postérieure actuelle et ils sont mis à jour par les équations suivantes [16] :

$$\begin{aligned} \mu_k^{new} &= \frac{\sum_{i=1}^{n} P_{ik}(x_i)}{\sum_{i=1}^{n} P_{ik}} && (a) \\ \sum_{k}^{new} &= \frac{\sum_{i=1}^{n} P_{ik}(x_i \mu_k^{new})(x_i - \mu_k^{new})^T}{\sum_{i=1}^{n} P_{ik}} && (b) \\ \pi_k^{new} &= \frac{1}{n} \sum_{i=1}^{n} P_{ik} && (c) \end{aligned} \tag{2.12}$$

- **Évaluation :** après chaque mise à jour des paramètres, la fonction log de vraisemblance (2.11) est évaluée pour vérifier la convergence.
- **Condition d'arrêt :** L'algorithme EM fonctionne de manière itérative, et l'étape d'évaluation est associée à une condition d'arrêt qui met fin aux opérations une fois qu'elle est satisfaite.

La condition d'arrêt de EM peut être définie par un nombre maximum d'itérations ou un changement suffisamment faible au niveau des valeurs des paramètres, c'est-à-dire lorsque $|L(\theta^{(i+1)}) - L(\theta^{(i)})| \leq \varepsilon$ [18]. Une fois que EM atteint sa convergence, la nouvelle classe de chaque échantillon x_i, $i = 1, \ldots, N$ est déterminée par l'équation (2.10).

Exemple d'illustration de clustering basé sur GMM

Dans cet exemple, nous simulons les opérations de clustering de GMM avec les échantillons de données du tableau 2.2[a] en deux clusters (K=2).

TABLEAU 2.2 : DONNÉES DE SIMULATION DU GMM

[a] Échantillons de données

num	F1	F2	Class
1	0.90	0.43	1
2	0.35	0.65	2
3	0.36	0.68	2
4	0.81	0.46	1
5	0.38	0.63	2
6	0.60	0.40	1
7	0.48	0.44	2
8	0.60	0.49	1
9	0.85	0.42	2
10	0.30	0.70	2
11	0.42	0.57	1
12	0.72	0.44	2

[b] Probabilités d'initiation

num	F1	F2	Labels
1	0.0	1.0	2
2	1.0	0.0	1
3	1.0	0.0	1
4	0.0	1.0	2
5	1.0	0.0	1
6	0.0	1.0	2
7	0.8126	0.1874	1
8	0.0	1.0	2
9	0.0	1.0	2
10	1.0	0.0	1
11	1.0	0.0	1
12	0.0	1.0	2

[c] Probabilités de convergence

num	F1	F2	Labels
1	0.0	1.0	2
2	1.0	0.0	1
3	1.0	0.0	1
4	0.0	1.0	2
5	1.0	0.0	1
6	0.0	1.0	2
7	0.0	1.0	2
8	0.0	1.0	2
9	0.0	1.0	2
10	1.0	0.0	1
11	1.0	0.0	1
12	0.0	1.0	2

En supposant que les paramètres initiaux sont connus, les coefficients des deux composants π_1 et π_2 sont initialisés de telle sorte que les densités des deux distributions soient égales. Les valeurs initiales des paramètres μ_1, Σ_1 et μ_2, Σ_2 sont obtenues à partir de K-Means et de GMM de la bibliothèque Scikit-learn.

Paramètres initiaux du composant 1 :

$$\pi_1 = 0.5; \quad \mu_1 = [0.3817, 0.6117]; \quad \Sigma_1 = \begin{bmatrix} 0.0032 & -0.0047 \\ -0.0047 & 0.0076 \end{bmatrix}$$

Paramètres initiaux de composant 2 :

$$\pi_2 = 0.5; \quad \mu_2 = [0.7467, 0.44]; \quad \Sigma_2 = \begin{bmatrix} 0.0137 & -0.0006 \\ -0.0006 & 0.0008 \end{bmatrix}$$

Maintenant, prenons le premier exemple x_1 des données du tableau 2.2[a] et déterminons sa probabilité postérieure dans chaque composant de mélange avec les paramètres initiaux. Pour ce faire, nous devons trouver sa densité de distribution dans les deux composants.

Trouvons le déterminant de la matrice de covariance Σ_1 :

$|\Sigma_1| = det(\Sigma_1)$ = (0.0032 x 0.0076) - (-0.0047 x -0.0047) = 0.00000223

Trouvons la différence entre l'exemple x_1 et le vecteur moyen μ_1 :

$x_1 - \mu_1$ = [0.90, 0.43] - [0.3817, 0.6117] = [0.5183, -0.1817]

Déterminons l'inverse de la matrice de covariance Σ_1 :

$$\text{inv}(\Sigma_1) = \frac{1}{det(\Sigma_1)} * \begin{bmatrix} 0.0076 & 0.0047 \\ 0.0047 & 0.0032 \end{bmatrix} = \begin{bmatrix} 3408.0717 & 2107.6233 \\ 2107.6233 & 1434.9776 \end{bmatrix}$$

Trouvons la partie de la fonction exponentielle dans l'équation (2.8) :

$$Exp_1 = \exp\Big(-\frac{1}{2} * [0.5183, -0.1817]^T * \begin{bmatrix} 3408.0717 & 2107.6233 \\ 2107.6233 & 1434.9776 \end{bmatrix} \Big) * [0.5183, -0.1817] \approx 0$$

Déterminons la densité de x_1 dans le premier composant en utilisant (2.8) et en remplaçant Exp_1 et $det(\Sigma_1)$ par leurs valeurs :

$$P_1(x_1|\Theta_1) = \frac{1}{(2*\pi)^{2/2}*(0.00000223)^{1/2}} * 0 = 0$$

En appliquant la même procédure, la densité de x_1 dans le deuxième composant est déterminée comme suit :

$$P_2(x_1|\Theta_2) = \frac{1}{(2*\pi)^{2/2}*(0.00001060)^{1/2}} * 0.4212 = 20.5899$$

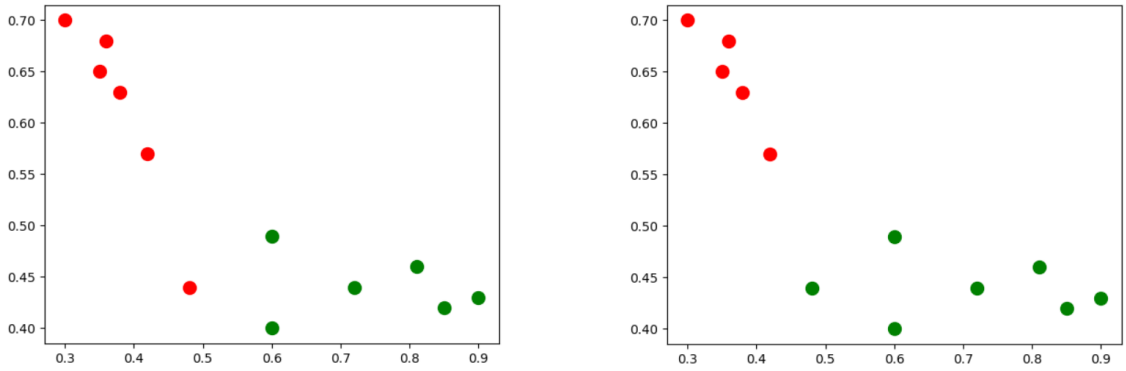
En utilisant (2.9), les probabilités postérieures de x_1 dans les deux composants sont respectivement déterminées comme suit :

$$P_{1,1} = \frac{0.5*0}{(0.5*0+0.5*20.5899)} = 0.0$$
$$P_{1,2} = \frac{0.5*20.5899}{(0.5*0+0.5*20.5899)} = 1.0$$

Ce processus est appliqué à chaque échantillon du tableau 2.2[a], et la classe de chaque échantillon avec les paramètres initiaux est déterminée, comme le montre le tableau 2.2[b]. Ensuite, l'algorithme EM est utilisé pour maximiser les paramètres de chaque composant. Les résultats obtenus après 20 itérations (convergence) sont présentés dans le tableau 2.2[c]. Les figures 2.2[a] et 2.2[b] présentent respectivement les clusters obtenus avec les paramètres initiaux et les paramètres maximisés par l'algorithme EM.

FIGURE 2.2 : Clustering basé sur GMM

[a] Initiation **[b] Convergence**



2.4.1 Algorithme de clustering GMM

Pour le problème de clustering, le modèle de mélange gaussien (GMM) classe chaque échantillon de données dans un groupe avec les valeurs initiales des paramètres. Après, une technique comme l'algorithme d'espérance-maximisation (EM) est utilisée pour obtenir les valeurs optimales des paramètres qui identifient la classe probable ou la vraisemblance de chaque échantillon de données.

Algorithme 2.3 : Clustering basé sur le GMM

X : ensemble de données d'entrée
N : nombre d'échantillons de X
K : nombre de clusters
Max-iter : nombre maximum d'itérations
Pour k de 1 à K **Faire**
    Initialiser les paramètres π_k, μ_k, Σ_k du composant k ;
Fin Pour
Pour n de 1 à Max-iter **Faire**
    Pour i de 1 à N **Faire**
        Déterminer la densité de x_i dans chaque composant avec l'équation (2.8)
        Évaluer la postérieure de x_i dans chaque composant en utilisant (2.9)
        Mettre à jour les paramères π, μ, Σ de chaque composant en utilisant (2.12)
        Évaluer la fonction log de vraisemblance en utilisant (2.11)
        Déterminer le groupe de x_i avec l'équation (2.10)
    Fin Pour
Fin Pour

2.5 Conclusion

Dans ce chapitre, nous avons présenté trois algorithmes classiques et très populaires dans le traitement et l'analyse de données, à savoir l'analyse en composantes principales (ACP), la technique de suréchantillonnage minoritaire synthétique (SMOTE) et le modèle de mélange gaussien (GMM). Nous utilisons ces algorithmes dans notre système dans le but d'améliorer la qualité de nos ensembles de données et notre système en général. Nous discuterons des détails de nos stratégies d'utilisation de ces techniques dans le sixième chapitre, intitulé « Méthodologie ».

Chapitre 3 : Systèmes de classification basés sur des règles floues (FRBCS)

Chapitre 3

Systèmes de classification basés sur des règles floues (FRBCS)

3.1 Introduction

La classification est l'une des tâches les plus importantes dans le domaine d'apprentissage automatique et profond. Prenons un ensemble de N éléments, dont les entrées sont des vecteurs de la forme $x_i = \{x_{i1}, \ldots, x_{ip}\}$, $i = 1, \ldots, N$ et un ensemble de classes de sortie $C = \{C_1, \ldots, C_K\}$. Le problème de classification consiste à trouver une fonction appropriée f de sorte que $f(x_i)$ renvoie la classe $C_i \in C$ à laquelle appartient le motif x_i, $\forall i = 1, \ldots, N$. L'objectif est d'avoir la meilleure approximation possible $\widehat{f(x)}$ de sorte que le nombre de motifs mal classés par $\widehat{f(x)}$ soit minimal par rapport à $f(x)$ [20].

Les systèmes basés sur des règles floues qui apprennent à partir d'exemples ont été introduits pour la première fois par Wang et Mendel en 1992, puis étendus pour le problème de classification par Zheru Chi, Jing Wu et Hong Yan en 1995 [20]. Les systèmes de classification des règles floues utilisent les règles floues comme support d'apprentissage. À la différence des systèmes flous classiques, les entrées sont mappées sur l'une des étiquettes de classe [5].

3.2 Définition

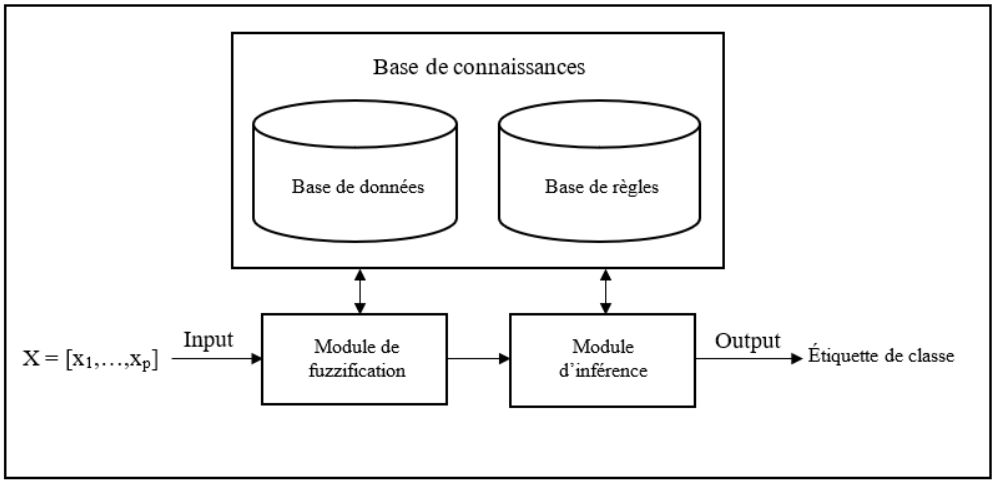
Un système de classification basé sur des règles floues (FRBCS) est un mécanisme de raisonnement basé sur le raisonnement approximatif qui a la capacité d'exprimer l'ambiguïté et la subjectivité présentes dans le raisonnement humain [21].

Les FRBCS sont constitués par une base de connaissances, un module de fuzzification et un moteur d'inférence. Ces composants transforment les données réelles en données floues pour créer des règles floues et déduire les sorties à partir de ces règles. Suivant l'approche générale de l'apprentissage supervisé, le comportement des FRBCS est défini par une fonction de mappage, comme suit :

$$\varphi : X^p \rightarrow Y$$

Où $X^p = X_1, \ldots, X_p$ sont les domaines des caractéristiques de données et Y est l'ensemble des classes de sortie possibles [22]. La Figure 3.1 présente la structure générale des FRBCS.

FIGURE 3.1 : Structure générale des FRBCS



3.2.1 Base de connaissances

La base de connaissances (KB) est une partie du système flou constituée par une base de données et une base de règles [23].

Les composants de la base de connaissances contiennent les données et les paramètres requis pour que le système apprenne automatiquement à partir d'exemples et qu' il prédise les résultats pour les nouvelles entrées.

3.2.1.1 Base de données

La base de données (DB) est le composant de la base de connaissances qui regroupe les paramètres du domaine de caractéristiques, l'ensemble des variables linguistiques floues et les fonctions d'appartenance associées pour chaque caractéristique des données [21].

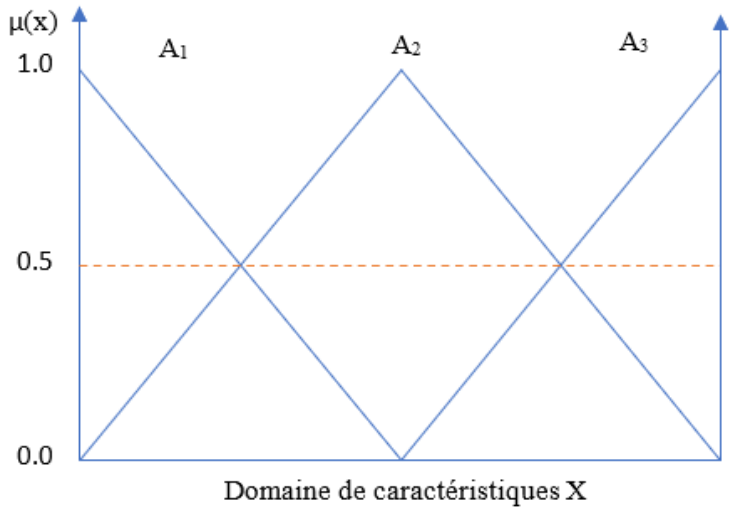
Domaine de caractéristique de données : Les systèmes flous utilisent la théorie des ensembles flous proposée par le Professeur Lotfi Zadeh [21]. Pour un ensemble d'objets X, le domaine de caractéristiques ou l'univers de discours est l'espace d'entrée de la caractéristique x_j, $j = 1, \ldots, p$ d'un exemple de X divisé en Q_j partitions floues, où $Q_j = 2n + 1$, $n \in \mathbb{N}$ [20]. Chaque partition ou ensemble flou est associé à une variable linguistique A_k,$k = 1, \ldots, Q_j$ dans le domaine X, défini par : $A_k : X \rightarrow [0, 1]$.

La division de l'espace d'entrée d'un domaine de caractéristique X est une étape cruciale dans la conception des systèmes flous. Néanmoins, l'algorithme d'origine n'impose aucune méthode de partitionnement spécifique, il propose plutôt une stratégie de partition également espacée [20]. L'espace de chaque partition floue est défini par la relation suivante [22] :

$$A_{jk} = \frac{max(x_j) - min(x_j)}{Q_j}$$

Où Q_j est le nombre de partitions de x_j, A_{jk} est la k-ième variable linguistique floue de la caractéristique x_j. La figure 3.2 illustre un exemple de domaine de caractéristique X, partitionné en trois ensembles flous et mesuré en degré d'appartenance $\mu(x)$.

FIGURE 3.2 : Exemple de domaine de caractéristiques flou



Variables linguistiques floues : Les variables linguistiques floues permettent de représenter correctement des informations imprécises et complexes à l'aide des termes linguistiques flous, facilitant ainsi le traitement de ces informations par les ordinateurs. Une variable linguistique floue peut être définie comme une variable dont les valeurs sont des mots ou des phrases en langage naturel plutôt que des nombres. Ces variables sont définies par des ensembles flous granulés sur un domaine de caractéristiques [21]. Par exemple, les trois ensembles flous définis sur le domaine X de la figure 3.2 sont respectivement associés aux variables linguistiques floues A_1, A_2 et A_3.

Fonction d'appartenance : Une fonction d'appartenance μ est une fonction qui détermine comment chaque point de l'espace d'entrée est mappé à un degré d'appartenance compris entre 0 et 1 [24]. Plusieurs types de fonctions d'appartenance existent, et le modèle triangulaire en est un des plus connus [21].

Le degré d'appartenance d'une valeur d'entrée x_{ij}, i = 1,...,N à un ensemble flou A_{jk} modélisé par une fonction d'appartenance triangulaire de paramètres a, b et c, est défini par l'équation suivante :

$$\mu_{A_{jk}}(x_{ij}) = \begin{cases} 0, & \text{Si } x_{ij} < \text{a}; \\ \frac{x_{ij}-a}{b-a}, & \text{Si a} \leq x_{ij} < \text{b}; \\ 1, & \text{Si } x_{ij} = \text{b} \\ \frac{c-x_{ij}}{c-b}, & \text{Si b} < x_{ij} \leq \text{c}; \\ 0, & \text{Si } x_{ij} > \text{c}; \end{cases} \tag{3.1}$$

La figure 3.2 présente un exemple de trois fonctions d'appartenance triangulaires associées aux ensembles de variables linguistiques floues A_1, A_2 et A_3.

3.2.1.2 Base de règles

La base de règles (RB) est l'unité de stockage des connaissances représentées au moyen des règles floues "IF-THEN" [21].

Règle floue : Une règle floue est une représentation efficace des connaissances fournies par un expert ou des données [25]. Les règles floues constituent un moyen facile d'exprimer des connaissances imprécises, ce qui confère la transparence et la compréhensibilité des systèmes flous [21]. Le modèle de règle floue utilisé dans les FRBCS est de la forme suivante :

$$\text{Règle } R_i : \text{IF } \overbrace{x_{i1} \; is \; A_{1k} \text{ AND} \ldots \text{ AND } x_{ip} \; is \; A_{pk}}^{\text{Antécédent}} \text{ THEN } \overbrace{\text{Class } = \text{ C}_i \text{ With RW}_i}^{\text{Conséquent}} \tag{3.2}$$

OÙ R_i est l'étiquette de la i-ème règle, $xi = (x_{i1},\ldots,x_{ip})$ est le vecteur de motif de p-dimension, A_{jk} est la k-ème variable linguistique du domaine de caractéristique x_j modélisée par une fonction d'appartenance, « AND » est un opérateur logique associé au connecteur de conjonction logique t-norme ($\wedge$), C_i est l'étiquette de classe et RW_i est le poids de la règle R_i.

Antécédent d'une règle floue : La partie antécédente d'une règle floue est sous forme d'énoncé conditionnel, décrivant les valeurs que les attributs antécédents doivent prendre afin de dériver le conséquent correspondant [22]. L'antécédent des règles floues est toujours une proposition floue ou une conjonction/disjonction de celles-ci. Ces propositions peuvent être atomiques (une seule variable linguistique) ou composées (conjonctions ou disjonctions de propositions atomiques) [21].

Conséquent d'une règle floue : Le conséquent du modèle des règles floues utilisé par les FRBCS est l'étiquette de classe de l'exemple correspondant, associée au poids de la règle [21].

Poids d'une règle floue : Le poids d'une règle floue est son degré d'appartenance ou sa certitude à une étiquette de classe [26]. Il est également la force de déclenchement d'une règle floue [22]. Plusieurs heuristiques sont utilisées pour déterminer le poids d'une règle floue. Le facteur de certitude pénalisé (PCF) est une méthode récente utilisée dans plusieurs travaux [6, 27]. L'efficacité de cette heuristique est due à sa capacité à prendre en compte tous les exemples de données en déterminant le poids d'une règle. Le PCF de la règle R_i d'un ensemble d'objets X est défini par la relation suivante :

$$RW_i = PCF = \frac{n^+(R_i,X) - n^-(R_i,X)}{n(R_i,X)} \tag{3.3}$$

Où le cardinal positif $n^+(R_i,X)$ est la somme des degrés de correspondance de tous les exemples de X qui dérivent de la même classe que la règle R_i. Il est défini par l'équation suivante :

$$n^+(R_i,X) = \sum_{i=1}^{N} \mu_{R_i}(x_i) \mid \text{Class}(x_i) = C_i \tag{3.4}$$

Le cardinal négatif $n^-(R_i,X)$ est la somme des degrés de correspondance de tous les exemples de X qui ne font pas partie de la même classe que la règle R_i. Il est déterminé par la relation suivante :

$$n^-(R_i,X) = \sum_{i=1}^{N} \mu_{R_i}(x_i) \mid \text{Class}(x_i) \neq C_i \tag{3.5}$$

Ensuite, le cardinal $n(R_i, X)$ est la somme totale des degrés de correspondance de tous les exemples de X. Il est défini par l'équation suivante :

$$n(R_i, X) = \sum_{i=1}^{N} \mu_{R_i}(x_i) \tag{3.6}$$

Où $\mu_{R_i}(x_i)$ est le degré de correspondance de l'exemple x_i avec la règle R_i déterminé par le connecteur de logique conjonctive t-norme, comme suit :

$$\mu_{R_i}(x_i) = \bigwedge_{j=1}^{p} \mu_{A_{jk}}(x_{ij}) \tag{3.7}$$

Où $\mu_{A_{jk}}(x_{ij})$ est le degré d'appartenance de la valeur x_{ij} à l'ensemble flou A_{jk} déterminé par (3.1).

3.2.2 Module de fuzzification

Le module de fuzzification est un composant intermédiaire qui établit une correspondance entre les entrées à valeur réelle et les données floues à l'aide d'une fonction d'appartenance [5]. Au cours des opérations de fuzzification, les entrées nettes sont transformées en valeurs d'appartenance, représentant le degré de correspondance avec les termes linguistiques dans les ensembles flous [25].

3.2.3 Moteur d'inférence

Le moteur d'inférence est la partie responsable de la classification des nouveaux exemples de données. Il déduit l'étiquette de classe des entrées selon les règles générées dans la base de règles [5]. Les FRBCS utilisent deux modèles de moteur d'inférence : l'inférence par règle gagnante et l'inférence par combinaison additive [28]. Pour classer un nouvel exemple x_{new} avec l'une de ces méthodes d'inférence, les étapes suivantes sont appliquées :

- Le degré de correspondance $\mu_{R_i}(x_{new})$ de la nouvelle instance x_{new} avec chaque règle de la base est déterminé par (3.6).
- Le degré d'association de x_{new} avec chaque règle de la base est déterminé par la relation suivante :

$$Ass_i(x_{new}) = \mu_{R_i}(x_{new}) * RW_i \tag{3.8}$$

3.2.3.1 Approche basée sur la règle gagnante

Avec cette méthode, pour chaque étiquette de classe, la règle ayant le plus grand degré d'association avec la nouvelle instance est sélectionnée et représente le degré d'association de la classe correspondante [28].

$$\text{ClassDegree}(\text{C}_i) = \max_{(R_i|Class(R_i)=C_i)}(Ass_i(x_{new})) \tag{3.9}$$

3.2.3.2 Approche basée sur la combinaison additive

Avec cette dernière, pour une étiquette de classe, toutes les règles participent à la détermination du degré d'association de la nouvelle instance par rapport à la classe correspondante [28].

$$\text{ClassDegree}(\text{C}_i) = \sum_{(R_i|Class(R_i)=C_i)}(Ass_i(x_{new})) \tag{3.10}$$

L'étiquette de classe de la nouvelle instance x_{new} est la classe ayant le degré d'association le plus élevé [28]. Elle est déterminée par l'équation (3.9) ou l'équation (3.10) comme suit :

$$\text{ClassLabel}(x_{new}) = \arg\max_{(c=1,\ldots,m)}(\text{ClassDegree}) \tag{3.11}$$

3.3 Génération des règles floues de FRBCS

La recherche croissante sur les systèmes flous a donné lieu à plusieurs algorithmes de génération de règles floues [5]. Les algorithmes Chi [6,20,27,28] et FuzzyDT [21] sont des méthodes couramment utilisées dans les recherches sur les FRBCS.

3.3.1 Algorithme Chi

La méthode de Chi constitue une extension de l'algorithme de Wang et Mendel, qui a été introduit en 1995 par Zheru Chi, Jing Wu et Hong Yan [29], puis en 1996 par Zheru Chi, Hong Yan et Tuan Pham [30]. L'algorithme est principalement composé de trois étapes. Premièrement, l'ensemble de variables linguistiques floues et les fonctions d'appartenance associées avec des paramètres nécessaires pour chaque caractéristique de données sont définis. Dans la deuxième étape, une règle floue est générée pour chaque exemple de données en respectant ces deux contraintes suivantes :

- Pour chaque valeur de l'exemple d'entrée, la variable linguistique avec le degré d'appartenance le plus élevé est sélectionnée.
- La partie antécédente de la règle est déterminée par l'intersection des variables linguistiques floues sélectionnées et la partie conséquente est une étiquette de classe. Toutes les règles auront le même nombre de variables que l'ensemble de données du problème.

La troisième et dernière étape de l'algorithme consiste à déterminer le poids de chaque règle floue générée tout en résolvant le conflit entre les règles. Parmi les règles générées par l'algorithme Chi, certaines peuvent avoir les mêmes antécédents et conséquents. Ces règles sont dites contradictoires et une seule est choisie si les poids sont égaux, sinon, c'est celle avec le poids le plus élevé qui est sélectionnée [20]. De même, certaines règles peuvent avoir les mêmes antécédents, mais des conséquents différents. Dans ce cas, la règle ayant le poids le plus élevé est sélectionnée [27].

Algorithme 3.1 : Génération des règles floues par Chi

1: X_{TR} : Données d'entraitement
2: Q : ensemble de domaines de caractéristiques
3: RB : base de règles
4: **Pour** i de 1 au nombre d'exemple dans X_{TR} **Faire**
5: **Pour** j de 1 au nombre de caractéristique de X_{TR} **Faire**
6: Sélectionner le domaine de caractéristiques Q_j
7: Règle R_i : liste des étiquettes sélectionnées
8: **Pour** k de 1 au nombre d'étiquettes floues de Q_j **Faire**
9: Sélectionner l'étiquette floue A_{jk} de x_{ij} ayant le degré le plus élevé
10: Ajouter A_{jk} dans R_i
11: **Fin Pour**
12: **Fin Pour**
13: Ajouter R_i dans RB
14: **Fin Pour**
15: **Pour** i de 1 au nombre de règles dans RB **Faire**
16: Calculer le poids RW_i de la règle R_i par (3.3), gérer le conflit entre les règles
17: **Fin Pour**

L'algorithme 3.1 ci-dessus présente la méthode de Chi. Dans les étapes 1 et 2, un ensemble de données d'entraînement et de domaines de caractéristiques sont définis. De l'étape 4 à l'étape 14, une règle floue est générée en sélectionnant l'étiquette linguistique ayant le degré d'appartenance le plus élevé pour chaque valeur du vecteur d'entrée. Enfin, le poids PCF est déterminé et la résolution de conflit entre les règles est effectuée entre les étapes 15 et 17.

Exemple de génération des règles floues par l'algorithme Chi

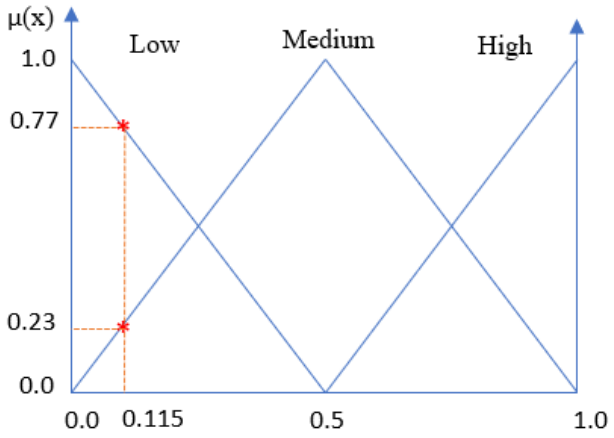
L'ensemble de données du tableau 3.1 est utilisé pour cet exemple.

TABLEAU 3.1 : DONNÉES DE GÉNÉRATION DES RÈGLES FLOUES

num	X1	X2	X3	X4	Class
1	0,115	0,5	0,055	0,045	0
2	0,42	0,835	0,035	0,045	1
3	0,615	0,335	0,615	0,585	1
4	0,4	0,7	0,09	0,045	0

Dans cet exemple particulier, les données sont normalisées sur l'intervalle de 0 et 1. Les caractéristiques de données utilisent la même base de données avec trois étiquettes linguistiques floues : Low = 0, Medium = 1, High = 2 de partitions également espacées, comme le montre la figure ci-dessous.

Domaine de caractéristiques des données du Tableau 3.1



En suivant les instructions de l'algorithme 3.1, toutes les règles sont générées. Par exemple, pour la valeur 0.115 du premier exemple de données, ses degrés d'appartenance aux ensembles flous de variables linguistiques Low, Medium et High sont respectivement de 0.77, 0.23 et 0.

Ce processus permet de fuzzifier tous les exemples et nous avons les résultats suivants :

e1 : [0.77, 0.23, 0 | 0, 1, 0 | 0.89, 0.11,0 | 0.91,0.09,0]

e2 : [0.16,0.84,0 | 0,0.33,0.67 | 0.93,0.07,0 | 0.91,0.09,0]

e3 : [0,0.77,0.23 | 0.33,0.67,0 | 0,0.77,0.23 | 0,0.83, 0.17]

e4 : [0.2, 0.8, 0 | 0 , 0.6,0.4 | 0.82,0.18,0 | 0.91, 0.09, 0]

En sélectionnant l'étiquette linguistique ayant le degré d'appartenance le plus élevé pour chaque valeur du vecteur d'entrée, les règles floues générées sont les suivantes :

R1 :[0,1,0,0] Class 0

R2 :[1,2,0,0] Class 1

R3 :[1,1,1,1] Class 1

R4 :[1,1,0,0] Class 0

Enfin, le poids PCF de chaque règle est déterminé par l'équation (3.3)

Pour R1 :[0,1,0,0], déterminons les degrés de correspondance de chaque règle en utilisant (3.7) :

R1 $\rightarrow$ R1 = [0.77, 1, 0.89, 0.91]

T-norm(R1 $\rightarrow$ R1) = 0.623, correspondance de R1 dans la classe 0.

R1 $\rightarrow$ R2 = [0.16, 0.33, 0.93, 0.91]

T-norm(R1 $\rightarrow$ R2) = 0.044, correspondance de R2 dans la classe 1.

R1 $\rightarrow$ R3 = [0, 0.67, 0, 0]

T-norm(R1$\rightarrow$ R3) = 0, correspondance de R3 dans la classe 1.

R1$\rightarrow$ R4 = [0.2 , 0.6, 0.82, 0.91] T-norm(R1$\rightarrow$ R4) = 0.089, correspondance de R4 dans la classe 0.

Déterminons les cardinaux en utilisant (3.4), (3.5) et (3.6) :

$n^{+}(R1, R4)$ = 0.712, correspondances des règles de la classe 0.

$n^{-}(R2, R3)$ = 0.044, correspondances des règles de la classe 1.

$n(R1, R2, R3, R4)$ = 0.758, correspondances de la base de règles.

$$RW_1 = \frac{0.712-0.044}{0.758} = 0.88$$

Ce processus est répété pour déterminer le PCF de chaque règle dans la base de règles et nous obtenons respectivement les poids 0.88, 0.33, 0.94 et 0.4 pour les quatre règles floues. La base de règles finale du système est la suivante :

R1 : IF X1 is Low AND X2 is Meduium AND X3 is Low AND X4 is Low THEN Class = 0 With 0.88

R2 : IF X1 is Meduim AND X2 is High AND X3 is Low AND X4 is Low THEN Class = 1 With 0.33

R3 : IF X1 is Meduim AND X2 is Medium AND X3 is Meduim AND X4 is Meduim THEN Class = 1 With 0.94

R4 : IF X1 is Meduim AND X2 is Meduium AND X3 is Low AND X4 is Low THEN Class = 0 With 0.4

3.3.2 Algorithme FuzzyDT

L'algorithme FuzzyDT est une extension de l'arbre de décision C4.5 (Quinlan, 1993), proposée en 2010 par Cintra et Camargo [21]. L'application de la méthode de FuzzyDT comprend trois étapes : la fuzzification des données, la construction de l'arbre de décision floue et l'extraction des règles floues par élagage de l'arbre. [21].

Plusieurs techniques sont disponibles pour construire un arbre de décision floue, notamment le gain d'information. Cette dernière commence par déterminer le ratio du gain d'information pour chaque caractéristique des données, puis sélectionne la caractéristique avec le ratio le plus élevé. Ce processus est ensuite itéré jusqu'à ce que toutes les caractéristiques soient utilisées ou tous les exemples soient classés [31]. Pour déterminer le ratio de gain d'une caractéristique de données, les étapes suivantes sont appliquées :

- Déterminer l'entropie I d'un ensemble de données X par l'équation suivante :

$$\mathrm{I(X)} = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

  Où m est le nombre de classes de données, p_k est la proportion de la classe C_k dans X. Si l'entropie d'un ensemble d'échantillons de données d'une classe est nul, alors tous les échantillons de cette classe sont parfaitement classées [7].

- Déterminer l'entropie I pour la caractéristique x_j :

$$\mathrm{I}(\mathrm{x}_j, X) = \sum_{k=1}^{m} \frac{|X_k|}{|X|} \mathrm{I}(\mathrm{X}_k)$$

Où X_k est un ensemble d'échantillons de X appartenant à la classe C_k pour la caractéristique x_j, $|X_k|$ est le nombre d'échantillons de X_k, $|X|$ est le nombre d'échantillons de données X.

- Déterminer le gain d'information G pour la caractéristique x_j :

$$\mathrm{G}(\mathrm{x}_j, X) = \mathrm{I}(\mathrm{X}) - \mathrm{I}(\mathrm{x}_j, X)$$

- Déterminer l'information partagée S_I pour la caractéristique x_j :

$$\mathrm{S}_I(x_j, X) = \sum_{k=1}^{m} \frac{|X_k|}{|X|} \log_2 \frac{|X_k|}{|X|}$$

- Déterminer le ratio de gain G_R pour la caractéristique x_j :

$$\mathrm{G}_R(x_j, X) = \frac{\mathrm{G}(\mathrm{x}_j, X)}{\mathrm{S}_I(x_j, X)}$$

Algorithme 3.2 : Génération des règles floues par FuzzyDT

```
X : ensemble de données
Pour i de 1 au nombre d'exemple dans X Faire
    Fuzzifier l'exemple e_i
Fin Pour
Répéter
    Pour j de 1 au nombre de caractéristiques de X Faire
        Sélectionner x_j avec le ratio de gain le plus élevé
    Fin Pour
    Définir la caractéristique sélectionnée comme nœud de test de l'arbre
Jusqu'à ce que chaque x_j de X soit utilisée ou chaque ei de X soit classé
Extraire les règles en appliquant le processus d'élagage sur l'arbre construite
Utiliser une heuristique pour déterminer le poids de chaque règle
```

Exemple d'extraction des règles floues par l'algorithme FuzzyDT

Nous utilisons les données du tableau 3.1 de l'exemple précédent et l'algorithme de Wang-Mendel pour fuzzifier les données. L'ensemble de données fuzzifié est présenté dans le tableau ci-dessous :

num	X1	X2	X3	X4	Class
1	Low	Meduim	Low	Low	0
2	Meduim	High	Low	Low	1
3	Meduim	Meduim	Meduim	Meduim	1
4	Meduim	Meduim	Low	Low	0

L'étape suivante est de construire l'arbre de décision flou.

Considérons X comme l'ensemble de données du tableau 3.1. L'entropie des données X est :

$$\text{I(X)} = -\frac{2}{4}\log_2(\frac{2}{4}) - \frac{2}{4}\log_2(\frac{2}{4}) = 1$$

Déterminons l'entropie de X1 :

$$\text{I(X1,X)} = \frac{1}{4}[-\frac{1}{1}\log_2(\frac{1}{1}) - \frac{0}{1}\log_2(\frac{0}{1})] + \frac{3}{4}[-\frac{1}{3}\log_2(\frac{1}{3}) - \frac{2}{3}\log_2(\frac{2}{3})] = 0.6887$$

Déterminons le gain d'information de X1 :

$$\text{G(X1,X)} = 1 - 0,6887 = 0.3113$$

Déterminons l'information partagée par X1 :

$$\text{S}_I\text{(X1,X)} = -\frac{1}{4}\log_2(\frac{1}{4}) - \frac{3}{4}\log_2(\frac{3}{4}) = 0.8113$$

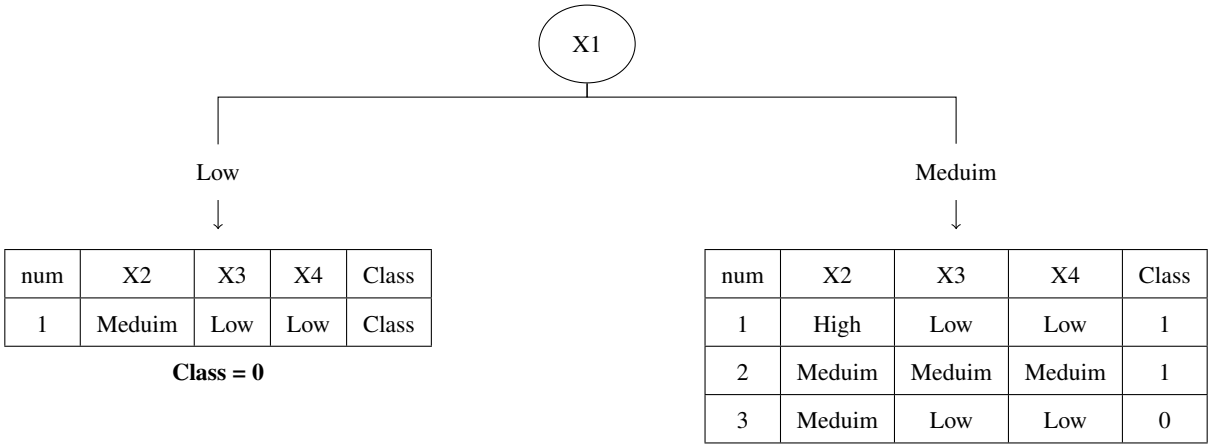
Déterminons le ratio de gain de X1 :

$$\text{G}_R\text{(X1,X)} = \frac{0.3113}{0.8113} = 0.3837$$

Pour X2, X3, X4, le même processus est appliqué et les résultats sont :

G_R(X1,X) = G_R(X2,X) = G_R(X3,X) = G_R(X4,X) = 0.3837.

Comme les ratios de gain des caractéristiques sont égaux, alors, nous pouvons utiliser un attribut de notre choix. Si nous prenons X1 par exemple, comme le nœud source de l'arbre, nous devons tracer les branches de l'arbre avec toutes les valeurs possibles de X1 et leurs données correspondantes, comme suit :

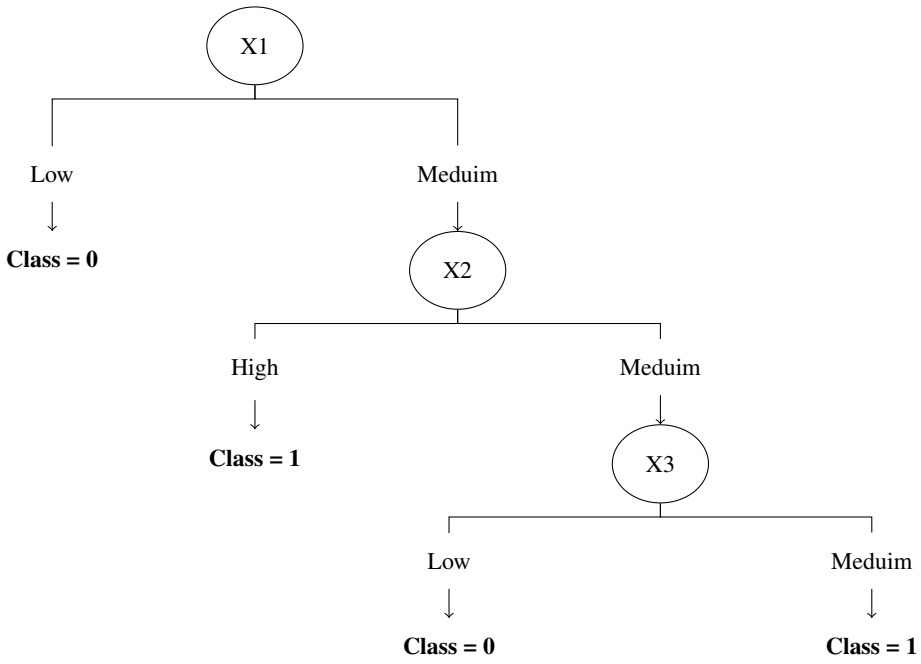


num	X2	X3	X4	Class
1	Meduim	Low	Low	Class

Class = 0

num	X2	X3	X4	Class
1	High	Low	Low	1
2	Meduim	Meduim	Meduim	1
3	Meduim	Low	Low	0

Comme nous avons une entropie I = 0 pour Low, cela signifie que le point de données de Low est parfaitement classé dans la classe 0. Pour Medium, nous avons une entropie I = 0.9183, cette valeur signifie que nous devons continuer à construire notre arbre sur cette branche en reprenant la même procédure. Après avoir classé tous les exemples de données, l'arbre de décision floue obtenu est le suivant :



Enfin, les règles extraites dans l'étape d'élagage sont les suivantes :

R1 : IF X1 is Low THEN Class = 0

R2 : IF X1 is Meduim AND X2 is High THEN Class = 1

R3 : IF X1 is Meduim AND X2 is Meduium AND X3 is Low THEN Class = 0

R4 : IF X1 is Meduim AND X2 is Meduim AND X3 is Meduim THEN Class = 1

Le poids de chacune de ces règles peut être déterminé par le PCF ou par d'autres heuristiques disponibles dans la littérature.

3.4 Conclusion

Dans ce chapitre, nous avons présenté les détails concernant les concepts fondamentaux des systèmes de classification basés sur des règles floues (FRBCS) et leurs composants. Nous avons également présenté les algorithmes Chi et FuzzyDT comme modèles de génération des règles floues dans les FRBCS. Bien que ces algorithmes soient simples, nous pouvons constater qu'ils produisent un grand nombre de règles floues complexes, en particulier lorsque la méthode de Chi est utilisée, puisqu'elle conserve le même nombre de caractéristiques dans les règles floues que dans l'ensemble de données. Ce problème peut compromettre l'un des objectifs clés des FRBCS, soit l'interprétabilité, principalement en exploitant une base de données volumineuse.

Le prochain chapitre explore diverses études dans la littérature sur la minimisation du nombre de règles floues dans les FRBCS.

Chapitre 4 : Minimisation du nombre de règles floues : revue de la littérature

Chapitre 4

Minimisation du nombre de règles floues : revue de la littérature

4.1 Introduction

Dans les classificateurs basés sur des règles floues, une quantité importante de règles est générée par les méthodes utilisées. Ce sujet a fait l'objet de plusieurs recherches. Dans ce chapitre, nous examinons plusieurs travaux dans la littérature sous différents aspects de réduction de la taille d'une base de règles floues.

4.2 Revue de la littérature

Michela, Dario et Hani [32] ont proposé une méthode basée sur l'algorithme génétique. Dans un premier temps, le nombre maximum M_{max} de règles à retenir après la minimisation de la base de règles initiale est défini. Ensuite, un vecteur de chromosome C_{RB} de paires M_{max} $p_m = (k_m, v_m)$ est généré, où k_m identifie l'index d'une règle dans la base initiale, $v_m = [v_{m,1}, \ldots, v_{m,F}]$ est un vecteur binaire de même dimension que la base de données qui indique la présence ou l'absence d'une condition dans la règle R_m.

Exemple : Prenons un modèle avec une base de règles initiale de 5 règles floues comme suit :

R_1 : IF X_1 is $A_{1,1}$ and X_2 is $A_{1,2}$ THEN Y is C_1

R_2 : IF X_1 is $A_{1,2}$ and X_2 is $A_{2,2}$ THEN Y is C_2

R_3 : IF X_1 is $A_{1,5}$ and X_2 is $A_{2,3}$ THEN Y is C_1

R_4 : IF X_2 is $A_{2,1}$ THEN Y is C_1

Supposons que nous voulons retenir 3 règles au maximum dans cette base de règles. Le vecteur de chromosome C_{RB} généré peut être supposé comme suit :

k_1	$v_{1,1}$	$v_{1,2}$	k_2	$v_{2,1}$	$v_{2,2}$	k_3	$v_{3,1}$	$v_{3,2}$
2	1	1	3	1	0	0	1	0

Le premier gène du chromosome sélectionne la deuxième règle dans la base de règles initiale $(k_1 = 2)$ avec toutes les conditions de valeurs binaires (1,1).

Le deuxième gène du chromosome sélectionne la troisième règle avec seulement la première condition de valeur binaire égale à 1 et la deuxième condition de la règle est rejetée avec la valeur binaire égale à 0.

Le troisième gène du chromosome ne sélectionne aucune règle puisque l'index du gène $(k_3 = 0)$ ne correspond à aucune règle dans la base initiale. Finalement, la base de règles finale ne contient que 2 règles suivantes :

$$R_2 : \text{IF } X_1 \text{ is } A_{1,2} \text{ and } X_2 \text{ is } A_{2,2} \text{ THEN Y is } C_1$$
$$R_3 : \text{IF } X_1 \text{ is } A_{1,5} \text{ THEN Y is } C_2$$

L'algorithme de réduction proposé par les auteurs est très efficace en ce qui concerne la minimisation du nombre de règles dans la base initiale. Cependant, les opérations génétiques utilisées génèrent les règles de manière aléatoire et le nombre de règles à retenir dans la base de règles finale est défini arbitrairement. De cette façon, certaines règles importantes de la base initiale peuvent être ignorées, ce qui peut dégrader les performances du système.

Leonardo, Antonio et Raul [6] ont introduit une technique de réduction d'une base de règles initiale en utilisant l'algorithme Quine McCluskey avec un codage spécifique. La base de règles initiale est divisée et minimisée par étiquette de classe après l'avoir générée. Avant le processus de réduction, le nombre de bits nécessaires pour coder chaque étiquette linguistique est déterminé par l'équation suivante :

$$\text{Bits} = Round(\frac{log_2(L)}{log_2(2)} + 1)$$

où L est le nombre d'étiquettes linguistiques.

Ensuite, les étiquettes linguistiques sont codées, remplacées et concaténées, tout en maintenant une différence d'un bit entre les étiquettes linguistiques contiguës. Les règles sont ensuite comparées et celles qui ne diffèrent que par un seul bit sont combinées avec un astérisque « $*$ » qui marque l'élimination d'une variable booléenne dans le processus de réduction.

En prenant les trois étiquettes linguistiques LOW, MEDIUM et LARGE utilisées par les auteurs, la forme initiale de chaque règle combinée est reconstituée en tenant compte des options suivantes :

- 0* implique l'affection à la variable LOW ou MEDUIM ;
- *1 implique l'affectation à la variable MEDUIM ou LARGE ;
- *0 et 1* n'impliquent pas de réduction, car le code binaire 10 n'est pas utilisé dans cet exemple particulier. Par conséquent, le symbole * ne peut prendre que 0 dans le premier cas, qui correspond à la variable LOW, et 1 dans le deuxième cas, qui correspond à la variable LARGE ;

Exemple : Prenons une base de règles contenant 4 règles d'une étiquette de classe. En appliquant la relation mentionnée ci-haut, le nombre de bits nécessaires pour chaque étiquette linguistique est défini à 2 et elles peuvent être représentées respectivement par les codes 00, 01 et 11. Le processus de codage et de minimisation de la base de règles peut être résumé comme suit :

IF x_1 is LOW and x_2 is LOW THEN Class = C_1 With RW_1

IF x_1 is MEDUIM and x_2 is HIGH THEN Class = C_1 With RW_2

IF x_1 is LOW AND x_2 is MEDUIM THEN Class = C_1 With RW_3

IF x_1 is LARGE AND x_2 is LARGE THEN Class = C_1 With RW_4

IF x_1 is 00 and x_2 is 00 THEN Class is C_1
IF x_1 is 01 and x_2 is 11 THEN Class is C_1
IF x_1 is 00 AND x_2 is 01 THEN Class is C_1
IF x_1 is AND 11x_2 is 11 THEN Class is C_1

0000 + 0111 + 0001 + 1111

bit 1	bit 2	bit 3	bit 4	Class
0	0	0	0	1
0	0	0	1	1
0	1	1	1	1
1	1	1	1	1

bit 1	bit 2	bit 3	bit 4	Class
0	0	0	*	1
*	1	1	1	1

IF x_1 is LOW AND x_2 is {LOW or MEDUIM} THEN Class = C_1 With RW_5

IF x_1 is {MEDUIM or LARGE} AND x_2 is LARGE THEN Class = C_1 With RW_6

La proposition des auteurs est intéressante parce qu'elle exploite la méthode d'extension des règles initiales plutôt que de sélectionner certaines règles. Cela leur a permis d'obtenir une base avec moins de règles, tout en conservant des performances comparables à celles de la base initiale. Toutefois, l'opération de réduction proposée peut devenir complexe quand le nombre de variables linguistiques augmente. De plus, l'algorithme ne peut pas complètement éliminer une variable dans une règle, même avec une base de données à faible dimension.

Chuan-Chang et Benito [33] ont suggéré de simplifier une base de règles floues à l'aide de la carte de Karnaugh. Premièrement, les variables linguistiques sont discrétisées en nombres entiers de n bits. Ainsi, chaque variable linguistique est représentée par une valeur binaire. Dans l'étape suivante, l'expression algébrique des règles est définie et elles sont représentées dans la carte de Karnaugh. Ensuite, l'expression minimale de la somme des produits (SOP) de la fonction algébrique est dérivée par l'utilisation du théorème suivant : $XY^{'} + XY = X$.

Exemple : Prenons un exemple de 4 règles dans la base de règles suivantes :

R_1 : IF A is SMALL AND B is LARGE THEN Class = C is SMALL

R_2 : IF A is SMALL AND B is SMALL THEN Class = C is SMALL

R_3 : IF A is LARGE AND B is SMALL THEN Class = C is LARGE

R_4 : IF A is LARGE AND B is LARGE THEN Class = C is LARGE

Dans cet exemple, 2 variables linguistiques SMALL et LARGE sont utilisées. Donc, chaque variable est discrétisée par 1 seul bit, et les variables sont représentées respectivement par des valeurs binaires 0 et 1. La fonction algébrique de la base de règles et la carte de Karnaugh correspondantes sont les suivantes :

$$\mathrm{C} = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2$$

			SMALL	LARGE
			A_1	A_2
			0	1
SMALL	B_1	0	0	1
LARGE	B_2	1	0	1

Les règles sont comparées les unes avec les autres sur la carte. Lorsque deux d'entre elles ont seulement un bit différent, elles sont combinées à l'aide du théorème mentionné plus haut. Ainsi, les règles $A_1B_1 = 00$ et $A_1B_2 = 01$ sont combinées en $A_1 = 0$, et les règles $A_2B_1 = 10$ et $A_2B_2 = 11$ sont combinées en $A_2 = 1$. L'expression minimale de la somme des produits de C et les règles obtenues sont les suivantes :

$$\mathrm{C} = A_1 + A_2$$

$$R_1 : \text{IF A is SMALL THEN C is SMALL}$$

$$R_2 : \text{IF A is LARGE THEN C is LARGE}$$

Il s'agit d'une proposition simple et efficace qui atteint l'objectif de réduire et d'obtenir moins de règles courtes dans la base finale. Cependant, l'algorithme ne peut traiter que 6 variables binaires (pour 2 variables linguistiques).

Caro, Simone, Marco et Uzay [34] se sont appuyés sur la théorie des graphes pour simplifier une base de règles floues. Les auteurs utilisent l'indice de Jaccard S pour mesurer la similarité entre deux ensembles flous. Tout d'abord, pour chacune des variables, l'algorithme détermine l'indice de similarité de Jaccard par paire des ensembles flous. Seules les paires avec l'indice de Jaccard supérieur à un seuil prédéfini sont stockées. Ensuite, pour chaque variable, un graphe est construit sur la base de paires des ensembles flous stockées. Dans l'étape suivante, les sous-composants complets du graphe sont cherchés et, pour chaque sous-composant, un nœud des sous-ensembles flous est choisi pour remplacer le reste.

Exemple : Supposons que nous avons 4 règles floues composées de 2 variables, A et B, qui ont chacune 4 sous-ensembles flous respectifs, (A_1, A_2, A_3, A_4) et (B_1, B_2, B_3, B_4).

$$\text{IF A is } A_1 \text{ AND B is } B_3 \text{ THEN OUTPUT is } C_1$$

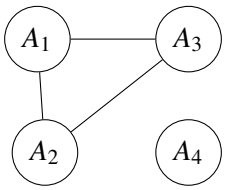
$$\text{IF A is } A_3 \text{ AND B is } B_2 \text{ THEN OUTPUT is } C_1$$

$$\text{IF A is } A_4 \text{ AND B is } B_1 \text{ THEN OUTPUT is } C_2$$

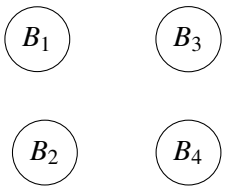
$$\text{IF A is } A_2 \text{ AND B is } B_4 \text{ THEN OUTPUT is } C_2$$

Maintenant, supposons que le seuil de similarité est fixé à S = 0.8. Pour la variable A, admettons que les indices de Jaccard soient $S(A_1, A_2) = 0.92$, $S(A_3, A_2) = 0.88$, $S(A_3, A_1) = 0.95$. Et, pour B, supposons que les sous-ensembles flous ne sont pas similaires. Les graphes correspondants à ces données sont les suivants :

(a) Forte similarité entre les sous-ensembles flous A_1, A_2, A_3



(b) Pas de similarité entre les sous-ensembles flous de la variable B



Il est clair sur les graphes qu'il y a une forte ressemblance entre les sous-ensembles flous A_1, A_2 et A_3 de la variable A. Dans ce cas, un ensemble flou peut être sélectionné pour remplacer les autres, dans cet exemple particulier, nous avons choisi A_1. Il n'y a pas de réduction de sous-ensembles flous pour la variable B, car il n'y a pas de ressemblance entre ses sous-ensembles flous. Enfin, les règles peuvent être réécrites comme suit :

IF A is A_1 AND B is B_3 THEN OUTPUT is C_1

IF A is A_1 AND B is B_2 THEN OUTPUT is C_1

IF A is A_4 AND B is B_1 THEN OUTPUT is C_2

IF A is A_1 AND B is B_4 THEN OUTPUT is C_2

Le problème avec cette proposition est qu'elle n'est associée à aucune stratégie permettant d'éliminer entièrement une règle. Les auteurs ont seulement cherché à réduire le nombre de sous-ensembles flous pour une variable donnée, ce qui la rend moins efficace en termes de minimisation du nombre de règles. Plus encore, la définition du seuil de similarité est arbitraire.

Arshia et Mohammad [35] ont développé une méthode de simplification des règles floues basée sur la valeur d'équilibre. Pour commencer, les auteurs attribuent une valeur d'appartenance à chaque étiquette linguistique des variables d'entrées dans la plage de $[-1, 1]$. Dans l'étape suivante, les règles sont regroupées par classe de sortie et la forme booléenne de chaque règle est déterminée. La dernière étape consiste à déterminer la valeur moyenne des formes booléennes de chaque classe qui serviront à calculer la moyenne finale de la base de règles comme seuil de sélection des règles importantes.

Exemple : Prenons une partie de l'exemple utilisé dans l'article d'origine concernant le contrôle d'erreur de sortie dans un contrôleur à logique floue. Nous avons 2 variables d'entrées E (erreur), CE (erreur de changement) qui ont 5 étiquettes linguistiques chacune : LN (Large Negative), MN (Medium Negative), Z

(Zero), MP (Medium Positive) et LP (Large Positive). La variable de sortie U a 3 étiquettes linguistiques (LN, Z, MP). Les degrés d'appartenance attribués sont les suivants : LN = -1, MN = -0.67, Z = 0, MP = 0.67 et LP = 1. Le processus est résumé dans le tableau ci-dessous :

0/P	Collection de règles	Formes booléennes	Avg
LN	If e is LN and ce is LN then u is LN	e*ce = 1	
	If e is LN and ce is MN then u is LN	e*ce = 0.67	0.56
	If e is LN and ce is Z then u is LN	e*ce = 0	
Z	If e is Z and ce is Z then u is Z	e*ce = 0	
	If e is LP and ce is LN then u is Z	e*ce = 1	0.5
MP	If e is MP and ce is MP then u is MP	e*ce = 0.4489	
	If e is LP and ce is MP then u is MP	e*ce = 0.67	0.56

La valeur d'équilibre ou la moyenne finale des étiquettes LN, Z et MP est $(0.56+0.5+0.56)/3 = 0.54$. Dans le tableau ci-dessus, seules les règles dont les deux entrées sont supérieures à la valeur d'équilibre (0.54) seront déclenchées.

La stratégie de réduction proposée par les auteurs semble intéressante dans le contexte où un nombre minimum de règles peut être obtenu. Mais une fois encore, les valeurs d'appartenance des étiquettes linguistiques sont arbitrairement définies.

Amel, Rim et Hayfa [36] adoptent une méthode de regroupement d'attributs pour réduire le nombre de règles dans un classificateur de règles floues. La première étape de la méthode proposée consiste à diviser les exemples de données en sous-ensembles de classe et à déterminer la matrice de corrélation des données de chaque sous-ensemble de classe. Pour chaque classe de données, les caractéristiques fortement corrélées sont regroupées pour former un sous-ensemble. Enfin, la base de règles finales est générée par la combinaison des bases de règles formées par différents sous-ensembles de caractéristiques. Ainsi, le nombre de règles dans la base finale des sous-ensembles de caractéristiques se trouve inférieur au nombre de règles dans la base des caractéristiques d'origine. Le nombre de règles d'une base est défini par K^n, où K est le nombre de partitions floues, n est le nombre de caractéristiques.

Exemple : Considérons les matrices de corrélation R_1, R_2, ainsi que leurs versions arrondies présentées ci-dessous. Les auteurs ont arrondi chaque valeur des matrices à 1 si elle est supérieure au seuil de $\theta = 0.7$ prédéfini, sinon elle est arrondie à 0.

$$R_1 = \begin{bmatrix} 1 & 0.8 & 0.1 & 0.2 & 0.5 \\ 0.8 & 1 & 0.9 & 0.1 & 0.1 \\ 0.1 & 0.9 & 1 & 0.3 & 0.2 \\ 0.2 & 0.1 & 0.3 & 1 & 0.8 \\ 0.5 & 0.1 & 0.2 & 0.8 & 1 \end{bmatrix} \iff R_\theta^1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0.3 \\ 0.8 & 1 & 0.4 & 0.1 & 0.1 \\ 0.5 & 0.4 & 1 & 0.3 & 0.2 \\ 0.2 & 0.1 & 0.3 & 1 & 0.8 \\ 0.3 & 0.1 & 0.2 & 0.8 & 1 \end{bmatrix} \iff R_\theta^2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Dans les matrices R_θ^1, nous pouvons constater qu'il y a une forte corrélation entre les caractéristiques X_1, X_2, X_3 qui composent l'ensemble $\{X_1, X_2, X_3\}$. Les caractéristiques X_4, X_5 sont également corrélées et elles forment l'ensemble $\{X_4, X_5\}$. Pour la matrice R_θ^2, le premier ensemble de caractéristiques corrélées est $\{X_1, X_2\}$, le deuxième ensemble $\{X_3\}$ ne contient qu'une seule caractéristique, X_3, qui n'est corrélée à aucune des autres caractéristiques et le troisième ensemble est $\{X_4, X_5\}$.

Avec 3 partitions floues, le nombre de règles à générer dans la base avec l'ensemble de caractéristiques d'origine est de $K^n = 3^5 = 243$ règles. Tandis que le nombre de règles obtenues par les sous-ensembles de caractéristiques est de $(3^3 + 3^2 + 3^2 + 3^1) = 48$ règles.

L'apprentissage de sous-ensembles de caractéristiques suggéré dans cet article est très utile, car il réduit considérablement le nombre de règles. Il permet aussi d'obtenir des règles courtes et faciles à lire grâce au regroupement d'attributs. Toutefois, les auteurs n'ont effectué aucune évaluation de la base de règles de leur système par rapport à la base de règles des caractéristiques d'origine.

Strehl, Moraga, Temme et Stancović [37] utilisent une approche d'optimisation d'une base de règles floues basée sur le diagramme de décision flou FuDD. FuDD est construit de sorte que tous les nœuds d'une couche donnée soient associés à la même variable et qu'aucune répétition de variables n'apparaît le long d'un chemin allant d'une feuille à la racine. Les variables sont ordonnées dans FuDD comme présentées dans les règles d'origine.Pour chaque variable linguistique, le nombre d'arêtes sortantes correspond au nombre d'étiquettes linguistiques sans répétition. Puisque les étiquettes linguistiques de chaque variable sont censées être ordonnées, certains termes linguistiques liés à l'opérateur T-conorm sont utilisés pour compacter la base de règles initiale.

Exemple : Prenons une base de règles à 2 entrées et une sortie comme suit :

If X_1 is T_{11} and X_2 is T_{22} then X_3 is T_{33}

If X_1 is T_{11} and X_2 is T_{23} then X_3 is T_{33}

If X_1 is T_{11} and X_2 is T_{24} then X_3 is T_{33}

If X_1 is T_{12} and X_2 is T_{23} then X_3 is T_{34}

If X_1 is T_{12} and X_2 is T_{24} then X_3 is T_{34}

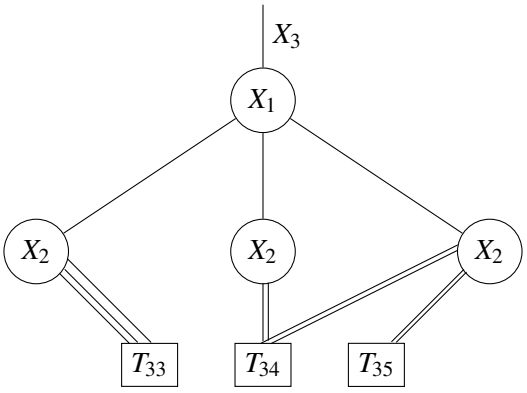
If X_1 is T_{13} and X_2 is T_{21} then X_3 is T_{35}

If X_1 is T_{13} and X_2 is T_{22} then X_3 is T_{35}

If X_1 is T_{13} and X_2 is T_{23} then X_3 is T_{34}

If X_1 is T_{13} and X_2 is T_{24} then X_3 is T_{34}

Le graphe FuDD de cette base de règles est construit comme suit :



En utilisant les étiquettes linguistiques spécifiques « smaller than or equal » et « larger than or equal » associées à l'opérateur T-conorm comme dans l'article d'origine, la base de règles finale à partir du graphe ci-haut est la suivante :

If X_1 is T_{11} and X_2 is Larger than or equal T_{22} then X_3 is T_{33}

If X_1 is T_{12} and X_2 Larger than or equal is T_{23} then X_3 is T_{34}

If X_1 is T_{13} and X_2 Larger than or equal is T_{23} then X_3 is T_{34}

If X_1 is T_{13} and X_2 is smaller than or equal T_{22} then X_3 is T_{35}

Le travail réalisé dans ce document obtient un bon taux de réduction d'une base de règles floues. Cependant, les auteurs n'ont effectué aucune étude expérimentale pour évaluer les capacités du système obtenu après la réduction des règles.

Marco, Luca, Beatrice, Francesco [38] propose une nouvelle méthode de simplification d'une base de règles appelée synthèse interactive séquentielle multivaluée MVSIS. Les auteurs commencent à construire une table de vérité des règles générées, suivie par une carte de Karnaugh. Pour trouver la forme minimale de la fonction algébrique de chaque classe, le principe de simplification de la carte de Karnaugh est pratiqué. Une règle est choisie arbitrairement pour régler les conflits entre règles identiques, qu'elles soient de la même classe ou non, dans la base de règles. Ensuite, les formes minimales obtenues sont transcrites sous forme de règle "IF-THEN ", en considérant l'absence d'une variable comme étant une variable DC (Don't Care).

Exemple : Prenons l'exemple de la base de règles de l'article et la table de vérité associée.

$$RB = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

X_1	X_2	X_3	X_4
0	0	0	0
0	1	0	0
0	1	0	1
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	1
1	0	1	0

Les cartes de Karnaugh avant et après la résolution du conflit entre la deuxième et la troisième règle en supposant que la deuxième a été sélectionnée au hasard sont les suivantes :

X_3 \ X_1X_2	00	01	11	10
0	0	0.1	1	1
1	0	−	1	0

X_3 \ X_1X_2	00	01	11	10
0	0	0	1	1
1	0	−	1	0

Pour la classe 1 par exemple, la forme minimale de la fonction algébrique est obtenue comme suit :

$$\begin{aligned} X_4^1 &= X_1X_2\neg X_3 + X_1\neg X_2\neg X_3 + X_1X_2X_3 \\ &= X_1\neg X_3(X_2 + \neg X_2) + X_1X_2(X_3 + \neg X_3) \\ &\Longrightarrow X_4^1 = X_1\neg X_3 + X_1X_2 \end{aligned}$$

Les règles extraites de la fonction X_4^1 sont les suivantes :

If X_1 is $A_{1,1}$ and X_2 is dc and X_3 is $A_{3,0}$ then X_4 is C_1

If X_1 is $A_{1,1}$ and X_2 is $A_{2,1}$ and X_3 is dc then X_4 is C_1

Dans cet article, les auteurs exploitent les règles redondantes pour simplifier la base de règles initiale. Par contre, l'algorithme pose certaines limites, à savoir le nombre d'étiquettes linguistiques, qui est limité à 2 pour chaque variable d'entrée, et la résolution arbitraire des conflits entre les règles.

Raffaele, Ciro, Corrado et Anna [39] introduisent une stratégie d'amélioration de l'interprétabilité d'un système de classification basé sur des règles floues en se basant sur le concept de cointension sémantique. Cela implique l'application des opérateurs de la logique booléenne T-norme et T-conorme pour transformer les règles de la base initiale en des règles étendues tout en préservant la sémantique des règles d'origine. Par conséquent, le nombre de règles initiales est réduit, ce qui améliore l'interprétabilité de la base finale.

Exemple : Démontrons le processus avec une base de règles initiale de 5 règles.

R_1 : IF Temperature IS Low AND Himidity IS Low then Normal

R_2 : IF Temperature IS High AND Himidity IS Low then Normal

R_3 : IF Temperature IS High AND Himidity IS High then Warning

R_4 : IF Temperature IS Low AND Himidity IS High then Warning

R_5 : IF Temperature IS Meduim AND Himidity IS High then Warning

Avec les opérateurs de la logique booléenne, les règles initiales sont réécrites en forme étendue sans altérer la sémantique de ces dernières. Les règles réécrites de la base initiale sont les suivantes :

R_1 : IF (Temperature IS Low OR Temperature is High) AND Himidity IS Low then Normal

R_2 : IF (Temperature IS High OR Temperature is Low OR Temperature is Meduim) AND Himidity IS High then Warning

Il ne fait aucun doute que la suggestion des auteurs préserve la logique et la sémantique des règles initiales dans les règles simplifiées. Toutefois, les performances du système final restent à vérifier et à comparer à celles de la base initiale.

Tamrat et László [40] ont proposé une méthode hybride pour simplifier une base de règles. Les auteurs exploitent à la fois la technique d'interpolation et de combinaison des règles pour réaliser leur opération hybride.

Pour réduire une base de règles, la technique d'interpolation est appliquée dans un premier temps à toutes les règles dans lesquelles la condition d'ordonnancement complet est respectée, soit les règles dont les variables floues, les univers d'entrée et de sortie sont bornés et graduels. Cela permet de garantir l'ordre complet dans ces règles. Ensuite, les règles qui ne respectent pas cette condition et qui ont un certain niveau de similarité sont minimisées par la technique de combinaison qui utilise les opérateurs de la logique booléenne.

Exemple : En s'appuyant sur les exemples d'illustration des auteurs, le fonctionnement des deux techniques est le suivant :

Pour la technique d'interpolation des règles :

D_{ht} / Q_g	Zero	Small	Meduim	Large	Very Large
Zero	meduim	meduim	meduim	long	very long
small	short	meduim	meduim	long	very long
meduim	short	short	meduim	long	very long
large	short	meduim	meduim	long	very long
very large	short	meduim	meduim	long	very long

Dans le tableau ci-dessus, seules les règles et les variables associées qui respectent les conditions de raisonnement progressif et d'ordonnancement complet sont sélectionnées, les résultats obtenus sont présentés dans le tableau suivant :

D_{ht} / Q_g	Zero	Meduim	Very Large
Zero	meduim	meduim	very long
meduim	short	meduim	very long
very large	short	meduim	very long

Pour la technique de combinaison des règles :

D_{ht} / Q_g	z	sh	m	l	vl
z	g	g	g	g	g
sh	r	g	g	g	g
m	r	r	g	g	g
l	r	r	r	g	g
vl	r	r	r	r	g

Dans le tableau ci-dessus, les règles qui ont certaines conditions et sorties identiques sont combinées par les opérateurs «AND» et «OR».

Q_g \ D_{ht}	z	z/sh/m/l/vl	sh	sh/m/l/vl	m	m/l/vl	l	l/vl	vl
z		g							
sh				g					
sh/m/l/vl	r								
m						g			
m/l/vl			r						
l								g	
l/vl					r				
vl							r		g

La proposition hybride des auteurs est très efficace pour simplifier une base de règles floues avec les règles qui respectent les conditions d'interpolation utilisées et qui ont un degré de similarité important. Ce qui est peu réaliste dans la plupart des situations concrètes.

A.Y. Khedr, Ramadan et Abdel-Mageid [7] introduisent une nouvelle version de la méthode Quine McCluskey pour réduire la complexité d'une base de règles. Premièrement, les étiquettes linguistiques des règles sont codées par les valeurs binaires. Ensuite, toutes les règles en forme binaire sont transformées en forme décimale, ce qui permet aux auteurs d'ajouter l'ensemble de données "Don't Care (DC)" afin de réduire davantage une base de règles. Finalement, la base de règles est divisée en groupes de règles de même classe, et la méthode Quine McCluskey est appliquée sur chaque groupe.

Exemple : Supposons une base de 4 règles appartenant à la même classe comme suit :

Temperature	Humidity	AC control
Low	Meduim	High
Low	High	High
Meduim	Meduim	High
High	Low	High

Les règles dans le tableau ci-dessus sont formées par trois étiquettes linguistiques : Low, Medium et High. Le nombre de deux bits est défini pour coder les étiquettes linguistiques comme suit : 00, 01 et 10. Dans le tableau de forme binaire des règles ci-dessous, les variables sont codées par T_1T_0 pour Temperature et H_1H_0 pour Humidity.

T_1	T_0	H_1	H_0	AC control
0	0	0	1	1
0	0	1	0	1
0	1	0	1	1
1	0	0	0	1

Les fonctions d'entrées de High et DC en valeur décimale et binaire sont présentées ci-dessous :

$$\begin{aligned} &\text{High}(T_1,T_0,H_1,H_0) = \Sigma(1,2,5,8) \\ &\text{DC}(T_1,T_0,H_1,H_0) = \Sigma(0,3,4,6,7) \end{aligned} \iff \begin{aligned} &\text{High}(T_1,T_0,H_1,H_0) = \Sigma(0001,0010,0101,1000) \\ &\text{DC}(T_1,T_0,H_1,H_0) = \Sigma(0000,0011,0100,0110,0111) \end{aligned}$$

En bref, l'application de Quine McCluskey débute par la formation de groupes de règles en fonction du nombre de bits de 1, suivi de la comparaison des groupes adjacents. Lors de cette opération, les règles avec une différence d'un bit sont combinées par un astérisque ($*$). Ce processus se poursuit jusqu'à ce qu'il n'y ait plus aucune combinaison possible. Les combinaisons impossibles dans le processus sont des impliquants premiers marqués en vert.

Group	Minterms
G_1	0000
G_2	0001
	0010
	0100
	1000
G_3	0011
	0101
	0110
G_4	0111

$\Longrightarrow$

Group	1-Cube
G_5	000*
	00*0
	0*00
	*000
G_6	00*1
	0*01
	001*
	010*
	01*0
G_7	0*11
	01*1
	011*

$\Longrightarrow$

Group	2-cube
G_8	00**
	0*0*
	0**0
G_9	0**1
	0*1*
	01**

$\Longrightarrow$

Group	3-Cube
G_{10}	0***

Dans le tableau ci-dessous, la cellule d'intersection de chaque règle et de son IP dérivée est marquée par un (x).

IP \ M	0001	0010	0101	1000
1000				x
*000				x
0***	x	x	x	

Pour trouver les impliquants premiers essentiels, les auteurs utilisent une opération itérative qui permet de sélectionner tous les termes qui couvrent la fonction d'entrée. Dans chaque itération, le terme qui couvre plus de règles initiales est sélectionné. Sur ce, nous pouvons voir que le terme $0***$ couvre le plus grand nombre de règles, suivi du terme $*000$. Ces deux termes sont suffisants pour couvrir la fonction d'entrée. Les règles extraites à partir de ces deux impliquants premiers sont les suivantes :

$$\text{High}(T_1, T_0, H_1, H_0) = \neg T_1 + \neg T_0 \neg H_1 \neg H_0$$

$$\Downarrow$$

R_1 : IF (Temperature, Low) THEN (AC Control, High)

R_2 : IF (Temperature, Medium) THEN (AC Control, High)

R_3 : IF (Temperature, High) AND (Himidity, Low) THEN (AC Control, High)

L'approche introduite dans ce travail est très intéressante dans le contexte où elle est capable d'éliminer une variable dans une règle. Toutefois, L'implication des données DC augmente considérablement la taille de la fonction d'entrée pour une base de données de grande dimension.

Alonso, Magdalena et Guillaume [41] exploitent les opérateurs de la logique booléenne pour minimiser la base de règles floues d'expert. Tout d'abord, une base de règles élaborée par des experts est analysée pour identifier les règles redondantes. On parle de règles qui ont la même prémisse et la même conclusion ou qui couvrent un espace d'entrée déjà couvert par une règle existante. Ces règles redondantes sont supprimées dans la base initiale. Ensuite, les règles de même classe et qui partagent certaines prémisses identiques sont combinées par les opérateurs mentionnés ci-dessus.

Exemple : Supposons que nous avons une base de règles de 3 règles floues obtenues par un expert :

R1 : IF Input1 is A1 AND Input2 is B1 THEN Ouput is C1

R2 : IF Input1 is A2 AND Input2 is B2 THEN Ouput is C1

R3 : IF Input1 is A1 AND Input2 is B2 THEN Ouput is C2

Nous pouvons constater que les premières prémisses des règles R1 et R2 sont identiques et qu'elles appartiennent à la même classe. Avec l'opérateur "OR", ces deux règles sont combinées comme suit :

R12 : IF Input1 is A1 OR A2 AND Input2 is B1 OR B2 THEN Ouput is C1

R3 : IF Input1 is A1 AND Input2 is B2 THEN Ouput is C2

L'intérêt de la technique proposée est sa simplicité. En plus, elle peut obtenir une bonne performance de réduction d'un ensemble de règles. Toutefois, dans une base de règles où la similarité entre les règles est faible, le taux de réduction sera également très faible.

Feijun et Samuel [42] proposent de minimiser le nombre de règles en utilisant les indices d'importance des règles de la base initiale. Pour une règle R_i, $i = 1,\ldots,N$, où N est le nombre de règles, un indice de mesure est défini comme suit : $I_i = \sum_{j=1}^{m} \beta_i^j$, où β_i^j est la force de déclenchement de la règle R_i dans la cellule X_j. Ensuite, les règles sont classées selon l'importance des indices prédéterminés. Les premières règles importantes sont ensuite sélectionnées par l'utilisateur. Une fois encore, une opération de sélection arbitraire est effectuée dans ce travail, même si la stratégie adoptée par les auteurs semble intéressante.

Chen et Linkens [43] réduisent la complexité d'une base de règles floues en simplifiant les ensembles flous de l'univers de discours et le nombre de règles. Pour ce faire, les auteurs commencent à examiner le degré de similarité $S(A_{i,j},U_j)$, $j = 1,\ldots,S$; $i = 1,\ldots,C$, entre l'univers de discours U_j et l'ensemble flou A_{ij}. Si le degré de similarité dépasse le seuil $\lambda_y = 0.9$, alors l'ensemble flou A_{ij} est supprimé. De même, deux ensembles flous A_{ij} et A_{kj}, $i \neq k$ de l'univers de discours U_j sont combinés dans le cas où leur degré de similarité $S(A_{ij},A_{k,j}$ est supérieur au seuil λ_r. Ensuite, les opérations de réduction se poursuivent en éliminant toute règle floue dont le poids est proche de 0 et en combinant les règles dont la similarité $S_a(A_{ij},A_{kj})$ atteint une certaine valeur de référence γ_a. Une fois encore, la stratégie de minimisation suggérée est associée à la définition arbitraire d'un seuil de sélection.

Janos, Johannes, Szeifert [44] proposent également une approche de mesure de similarité basée sur les opérateurs logiques. Pour réduire le nombre de règles à générer, les auteurs exploitent la théorie des ensembles flous en mesurant la similarité $S(A_{ij}, U_j)$ entre l'ensemble flou A_{ij} et l'univers de discours U_j, ainsi que la similarité $S(A_{ij}, A_{kj})$ entre les ensembles flous A_{ij} et A_{kj}. Les ensembles flous ayant une similarité supérieure au seuil $\theta = 0,5$ sont combinés, tandis que ceux dont le degré de similarité dans leurs univers de discours est supérieur au seuil $\theta = 0,8$ sont éliminés.

4.3 Conclusion

Selon nos recherches, la plupart des méthodes de réduction des règles floues dépendent d'un choix subjectif. Ce qui élimine certaines règles de la base de connaissances du système sans aucune justification rationnelle. En outre, d'autres méthodes manquent d'efficacité en ce qui concerne le nombre de règles obtenues après la simplification. Vu l'intérêt croissant pour les systèmes flous dans le domaine d'intelligence artificielle explicative, il est impératif de développer une solution de minimisation de bases de règles floues de ces derniers, qui soit à la fois rationnelle et robuste.

Chapitre 5 : Algorithme Quine McCluskey

Chapitre 5

Algorithme Quine McCluskey

5.1 Introduction

L'algorithme Quine McCluskey, aussi connu sous le nom de méthode tabulaire, est un outil courant et largement utilisé dans la conception de logique numérique. Développée indépendamment par Quine (1955) et McCluskey (1966), la méthode QM permet de trouver l'expression minimale sous forme normale disjonctive correspondant à toute fonction booléenne [45]. Cette dernière a l'avantage de traiter un nombre quelconque de variables et de ne pas exiger de paramètre pour son application. De plus, elle est déterministe, c'est-à-dire qu'elle est capable de vérifier que la forme minimale d'une fonction booléenne a été atteinte [46].

Dans ce chapitre, nous allons explorer l'algorithme Quine McCluskey en définissant ses concepts et principes fondamentaux. Nous démontrerons ensuite les étapes de fonctionnement de l'algorithme à l'aide d'un exemple d'illustration détaillé.

5.2 Terminologies et principes fondamentaux

Quine McCluskey est un algorithme procédural qui repose sur un ensemble de concepts et de principes de base expliquant son fondement et son fonctionnement.

5.2.1 Terminologies de Quine McCluskey

Littéral : Un littéral est une variable de logique booléenne (A) ou son complément ($\bar{A}$) [6].

Minterms : Un minterm est un produit de n littéraux pour n variables où chaque variable apparaît exactement une fois, soit sous forme de vérité (A), soit sous forme de complément ($\bar{A}$) [6].

Par exemple, dans le minterm $A\bar{B}C\bar{D}$, les variables A et C apparaissent sous forme de «vérité», tandis que les variables B et D apparaissent sous forme de «complément».

Impliquant premier : L'impliquant premier d'une fonction est le produit qui ne peut être combiné avec un autre terme de la fonction pour éliminer une variable afin de simplifier davantage [6].

Supposons la fonction F = $ABC + \bar{A}BC + A\bar{B}\bar{C}$. Nous pouvons constater que le terme $A\bar{B}\bar{C}$ contient au moins deux variables d'état différent par rapport aux variables des autres termes de F. Par conséquent, le terme $A\bar{B}\bar{C}$ est un impliquant premier de F, car il ne peut être combiné à aucun des autres termes.

Impliquant premier essentiel : C'est tout impliquant premier d'une fonction capable de couvrir au moins une sortie de cette fonction qui n'est pas couverte par un autre impliquant [6].

Supposons la fonction simplifiée $F' = \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}C$ de la fonction F = $\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}C$. Dans la fonction simplifiée, l'impliquant premier $\bar{B}C$ est le seul à dériver le terme $A\bar{B}C$. Par conséquent, il est l'impliquant premier essentiel de F, car il est le seul à couvrir la sortie $A\bar{B}C$ de F.

5.2.2 Principes de base de Quine McCluskey

Les opérations de réduction de QM s'appuient sur trois lois fondamentales de l'algèbre booléenne [6] :

- Loi complémentaire : $A + \bar{A} = 1$; (1)
- Loi distributive : $A(B + C) = AB + AC$; (2)
- Loi idempotente : $A + A = A$; (3)

La méthode QM exige également que les variables d'entrée d'une fonction à minimiser soient converties en valeurs binaires. Par une table de vérité, le code binaire de chaque variable de la fonction est généré. Pour les fonctions dont les termes sont formés par deux variables ou plusieurs variables et leurs compléments, un codage simple peut être appliqué avec les valeurs binaires 0 et 1 [46]. Par exemple, les termes formés par les variables A et B, ou par les variables A, B et C, ainsi que leurs compléments ($\bar{A}$, $\bar{B}$ et $\bar{C}$) peuvent être codés en utilisant un seul bit pour chaque variable. Par contre, si le nombre de variables composant les termes est supérieur à 2 et que ce ne sont pas les formes de vérité des variables et de leurs compléments, alors une définition du nombre de bits et un codage spécifique sont nécessaires pour binariser ces termes. Par exemple, pour coder les termes composés par les variables A, B et C, il faut au moins deux bits composés par les valeurs binaires pour chacune des variables [6].

5.3 Exemple d'application de l'algorithme Quine McCluskey

La procédure d'application de la méthode QM comprend deux étapes principales [46] :

5.3.1 Première étape : recherche des impliquants premiers

La méthode de QM réalise une recherche itérative pour déterminer tous les impliquants premiers de la fonction à simplifier. Pour y parvenir, les opérations suivantes sont effectuées.

Binarisation des variables des termes de la fonction d'entrée : supposons la fonction ci-après.

$$F = \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + A\bar{B}CD + ABC\bar{D} + ABCD$$

Dans cet exemple particulier, nous avons les termes composés des variables A, B et C et leurs compléments $\bar{A}$, $\bar{B}$ et $\bar{C}$. Alors, nous allons simplement les remplacer chacun par sa valeur binaire correspondante afin de binariser chaque minterm de la fonction F, comme le montre le tableau 5.1.

TABLEAU 5.1 : TABLE DE VÉRITÉ DES MINTERMS DE LA FONCTION F

№	Minterms	Valeurs Binaires des variables			
		A	B	C	D
1	$\bar{A}B\bar{C}\bar{D}$	0	1	0	0
2	$A\bar{B}\bar{C}\bar{D}$	1	0	0	0
3	$A\bar{B}\bar{C}D$	1	0	0	1
4	$A\bar{B}C\bar{D}$	1	0	1	0
5	$AB\bar{C}\bar{D}$	1	1	0	0
6	$A\bar{B}CD$	1	0	1	1
7	$ABC\bar{D}$	1	1	1	0
8	$ABCD$	1	1	1	1

Formation des groupes de minterms : Former les groupes de termes est l'opération suivante de la première étape qui est obligatoire pour le processus de comparaison et de combinaison. Les groupes de termes sont formés en fonction du nombre de bits 1, puis classés par ordre croissant du nombre de bits 1. Le tableau 5.2 ci-dessous répertorie le rang, le groupe, le minterm, la forme décimale et binaire des groupes de termes de la fonction F.

TABLEAU 5.2 : GROUPE DE MINTERMS PAR NOMBRE DE BITS DE « 1 »

№	Groupes	Minterms	Décimales	Binaires
1		$\bar{A}B\bar{C}\bar{D}$ $\surd$	4	0100
2	1	$A\bar{B}\bar{C}\bar{D}$ $\surd$	8	1000
3		$A\bar{B}\bar{C}D$ $\surd$	9	1001
4		$A\bar{B}C\bar{D}$ $\surd$	10	1010
5	2	$AB\bar{C}\bar{D}$ $\surd$	12	1100
6		$A\bar{B}CD$ $\surd$	11	1011
7	3	$ABC\bar{D}$ $\surd$	14	1110
8	4	$ABCD$ $\surd$	15	1111

Processus de comparaison et de combinaison : il s'agit d'une série d'opérations où les termes de deux groupes adjacents sont comparés et combinés selon les principes de la méthode QM. Cela signifie que chaque terme du groupe G_k est comparé à tous les termes du groupe G_{k+1}. Ainsi, les termes qui diffèrent d'un seul bit sont combinés en remplaçant les bits de différence par un astérisque ($*$) dans le nouveau terme formé.

Exemple : première itération

La première itération est effectuée sur le tableau 5.2 et les résultats sont présentés dans le tableau 5.3.

TABLEAU 5.3 : MINIMISATION DE LA FONCTION F (première itération)

№	Groupes	Minterms	Décimales	Binaires
1		$B\bar{C}\bar{D}$	4,12	$*100$
2		$A\bar{B}\bar{C}$ $\surd$	8,9	$100*$
3		$A\bar{B}\bar{D}$ $\surd$	8,10	$10*0$
4	5	$A\bar{C}\bar{D}$ $\surd$	8,12	$1*00$
5		$A\bar{B}D$ $\surd$	9,11	$10*1$
6		$A\bar{B}C$ $\surd$	10,11	$101*$
7		$AC\bar{D}$ $\surd$	10,14	$1*10$
8	6	$AB\bar{D}$ $\surd$	12,14	$11*0$
9		ACD $\surd$	11,15	$1*11$
10	7	ABC $\surd$	14,15	$111*$

Pendant les opérations de comparaison et de combinaison de chaque itération, si deux termes se combinent, la «loi distributive» est appliquée pour isoler les variables et les bits de différence dans les termes correspondants. Ensuite, la «loi complémentaire» est appliquée pour éliminer les variables et bits isolés dans les nouveaux termes formés. Pour toute répétition de termes, la «loi idempotente» est appliquée afin d'éliminer les termes redondants. Les termes qui participent à au moins une opération de combinaison sont marqués d'un ($\surd$) (voir les tableaux 5.2 et 5.3). Les termes non marqués n'ont pas pu être combinés (voir les tableaux 5.3 et 5.4). Ils constituent les impliquants premiers de la première phase de l'algorithme QM.

Lorsque nous comparons les termes, toute opération de combinaison possible entre deux groupes adjacents entraîne la création d'un nouveau groupe de termes. Ce processus est répété jusqu'à ce qu'il n'y ait plus de combinaisons possibles entre les termes des derniers groupes formés. Pour notre exemple, les tableaux ci-dessous présentent le reste des opérations dans lesquelles les mêmes principes sont appliqués.

Exemple : deuxième itération

TABLEAU 5.4 : MINIMISATION DE LA FONCTION F (deuxième itération)

№	Groupes	Minterms	Décimales	Binaires
1		$A\bar{B}$	8,9,10,11	$10**$
2		$A\bar{B}$	8,9,10,11	$10**$
3		$A\bar{D}$	8,10,12,14	$1**0$
4	8	$A\bar{D}$	8,10,12,14	$1**0$
5		AC	10,11,14,15	$1*1*$
6	9	AC	10,11,14,15	$1*1*$

Exemple : troisième itération

TABLEAU 5.5 : MINIMISATION DE LA FONCTION F (troisième itération)

№	Groupes	Minterms	Décimales	Binaires
1		$A\bar{B}$	8,9,10,11	$10**$
2	8	$A\bar{D}$	8,10,12,14	$1**0$
3	9	AC	10,11,14,15	$1*1*$

Exemple : quatrième itération

En effectuant la quatrième itération sur le tableau 5.5, nous pouvons voir que les termes des deux groupes restants ne peuvent se combiner selon les principes de QM. Par conséquent, ces termes constituent les impliquants premiers de la fonction minimisée suivante :

$$F' = B\bar{C}\bar{D} + A\bar{B} + A\bar{D} + AC$$

5.3.2 Deuxième étape : recherche des impliquants premiers essentiels

Tout comme dans la première étape, les impliquants premiers essentiels d'une fonction d'entrée sont obtenus par une recherche itérative. Plusieurs techniques sont développées pour réaliser cette opération. Pour notre exemple, nous allons présenter une méthode basée sur le travail proposé dans [47]. Pour commencer, nous allons établir un tableau dont les lignes sont les termes en forme décimale de F' et les colonnes sont les valeurs décimales de F. Dans chaque cellule d'intersection entre un terme de F' et une valeur de F, si le terme couvre la valeur, alors un astérisque est marqué, comme indiqué dans le tableau 5.6.

TABLEAU 5.6 : RELATION ENTRE IMPLIQUANTS PREMIERS ET MINTERMS

Minterms / Impliquants	4	8	9	10	11	12	14	15
4,12	*					*		
8,9,10,11		*	*	*	*			
8,10,12,14		*		*		*	*	
10,11,14,15				*	*		*	*

Maintenant, nous devons sélectionner le terme de F' qui couvre le plus de valeurs de F dans le tableau 5.6, comme le montre le tableau 5.7 ci-dessous.

TABLEAU 5.7 : RECHECHE D'IMPLIQUANTS PREMIERS ESSENTIELS (itération 1)

Minterms / Impliquants	4	8	9	10	11	12	14	15
4,12	*					*		
8,9,10,11		*	*	*	*			
8,10,12,14		*		*		*	*	
10,11,14,15				*	*		*	*

Dans le tableau 5.7, nous pouvons constater que $(8,9,10,11)$ est le premier terme qui couvre le plus de valeurs de F'. Par conséquent, il est sélectionné comme un impliquant premier essentiel de F. La technique utilisée exige la suppression du terme sélectionné et de ses valeurs dans les ensembles F' et F. Ce qui nous permet de réduire les champs de recherche des impliquants premiers essentiels.

Après avoir retiré le terme sélectionné dans F′ et les valeurs décimales couvertes par ce dernier dans F, le tableau des termes restants est présenté ci-dessous.

TABLEAU 5.8 : RECHECHE D'IMPLIQUANTS PREMIERS ESSENTIELS (itération 2)

Minterms / Impliquants	4	12	14	15
4,12	*	*		
8,10,12,14		*	*	
10,11,14,15			*	*

Les mêmes principes sont appliqués au tableau 5.8, où le terme $(4, 12)$ est sélectionné comme celui qui couvre davantage les valeurs de F. Après la suppression de ce dernier et de ses valeurs décimales, le tableau 5.9 présente le reste des termes de F′.

TABLEAU 5.9 : RECHECHE D'IMPLIQUANTS PREMIERS ESSENTIELS (itération 3)

Minterms / Impliquants	14	15
8,10,12,14	*	
10,11,14,15	*	*

Après avoir sélectionné le terme $(10, 11, 14, 15)$ dans F′, nous pouvons voir que toutes les valeurs décimales de la fonction d'entrée F sont maintenant couvertes. La forme minimale de F après la deuxième étape est la suivante :

$$F'' = A\bar{B} + B\bar{C}\bar{D} + AC$$

5.4 Algorithme de Quine McCluskey

La méthode de QM est essentiellement composée de deux étapes :
La première étape consiste à chercher tous les impliquants premiers d'une fonction d'entrée après les opérations préliminaires, soit la binarisation et la formation des groupes de termes.

Dans la deuxième étape, les impliquants premiers essentiels sont également recherchés et sélectionnés, c'est-à-dire l'ensemble minimal des impliquants premiers qui couvrent toutes les sorties de la fonction d'entrée.

Algorithme 5.1 : Binarisation des termes de la fonction d'entrée

```
1: F : fonction des termes d'entrées
2: CB : liste de codes des variables
3: TB : liste de stockage des termes binarisés
4: Pour chaque terme dans F Faire
5:     Pour chaque variable dans le terme Faire
6:         Remplacer la variable par son code dans CB
7:     Fin Pour
8:     Ajouter terme dans TB
9: Fin Pour
```

Dans l'algorithme 5.1 ci-dessus, nous avons en entrée une fonction F des termes à minimiser et une liste de code binaire CB pour les variables des termes de F. De l'étape 4 à 13, un terme de F est binarisé en remplaçant chaque variable par son code binaire défini. Ensuite, le terme codé est ajouté dans la liste des termes binarisés TB créée à l'étape 3.

Algorithme 5.2 : Regroupement des minterms par nombre de bits «1»

```
1: TB : liste de termes binarisés
2: G : tableau de stockage des groupes de minterms
3: Pour chaque minterm dans TB Faire
4:     Nb = nombre de bits 1 dans minterm
5:     Si groupe Nb est dans G Alors
6:         Ajouter minterm dans le groupe Nb de G
7:     Sinon
8:         Créer groupe Nb dans G
9:         Ajouter minterm dans le groupe Nb de G
10:    Fin Si
11: Fin Pour
```

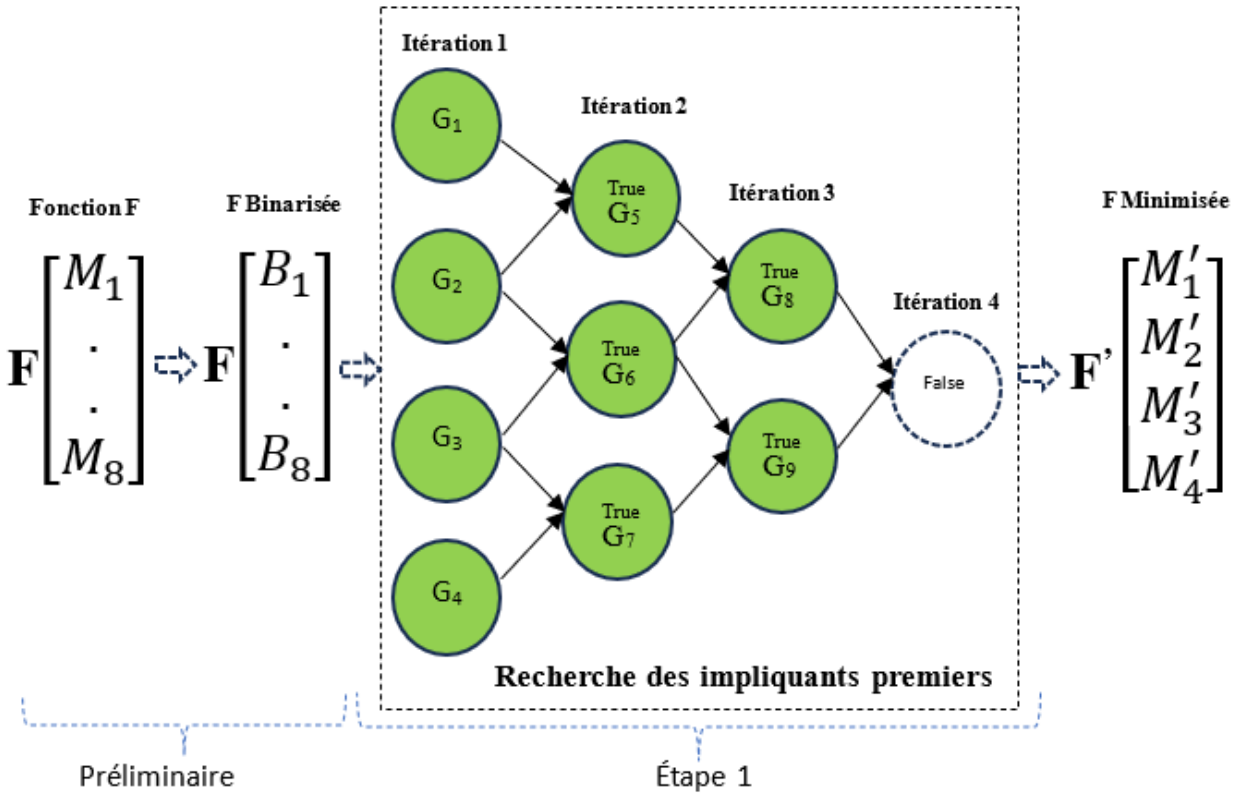
L'algorithme 5.2 a comme entrée une liste de termes binarisés TB d'une fonction à simplifier. De l'étape 3 à 11, chaque terme TB est ajouté dans un groupe G correspondant au nombre de bits 1 dans le terme. Si un terme n'a pas de groupe correspondant à son nombre de bits 1 , un nouveau groupe est créé pour ce dernier avant de l'ajouter.

Algorithme 5.3 : Recherche des impliquants premiers (combinaison)

```
1:  G : groupes de minterms
2:  IP : ensembe pour ajouter les impliquants premiers
3:  Tant que vraie Faire
4:      GN = copie de G
5:      L = nombre de groupes dans G
6:      G = tableau de stockage des nouveaux groupes
7:      Stop = true ;
8:      TC = ensemble pour ajouter les termes combinés
9:      j = 0
10:     Pour k de 0 à L Faire
11:         Pour chaque m_k dans GN_k Faire
12:             Pour chaque m_{k+1} dans GN_{k+1} Faire
13:                 Comparaison = comparer m_k et m_{k+1}
14:                 Si Comparaison est vraie Alors
15:                     Combinaison = combiner m_k et m_{k+1}
16:                     Créer groupe j dans G
17:                     Ajouter Combinaison dans le groupe G_j
18:                     Ajouter m_k et m_{k+1} dans TC
19:                     Stop = false
20:                 Fin Si
21:             Fin Pour
22:         Fin Pour
23:         j = j + 1
24:     Fin Pour
25:     NC = différence entre GN et TC
26:     Ajouter NC dans IP
27:     Si Stop est true Alors
28:         Arrêter la comparaison
29:     Fin Si
30: Fin Tant que
```

L'algorithme 5.3 reçoit un tableau G contenant des groupes de termes en entrée. De l'étape 3 à 9, une copie actuelle du tableau G est récupérée dans la variable GN et sa taille est réinitialisée. De l'étape 10 à 24, le tableau GN est parcouru en fonction du nombre de groupes L, puis chaque terme m_k d'un groupe G_k est comparé à chacun des termes m_{k+1} du groupe adjacent G_{k+1}. Si les deux sont combinables, ils seront combinés et ajoutés automatiquement au groupe G_j créé dans le tableau G. Ensuite, les termes combinés sont ajoutés à la liste TC et la variable Stop est définie sur « faux ». Les termes qui n'ont pas pu être combinés sont récupérés par la variable NC et seront ensuite ajoutés à l'ensemble IP. Enfin, la variable Stop est vérifiée pour mettre fin aux opérations si sa valeur actuelle est « true », sinon les opérations se poursuivent. Le fonctionnement du processus de recherche des IP est représenté par la figure 5.1 ci-dessous.

FIGURE 5.1 : Quine McCluskey (recherche des impliquants premiers)



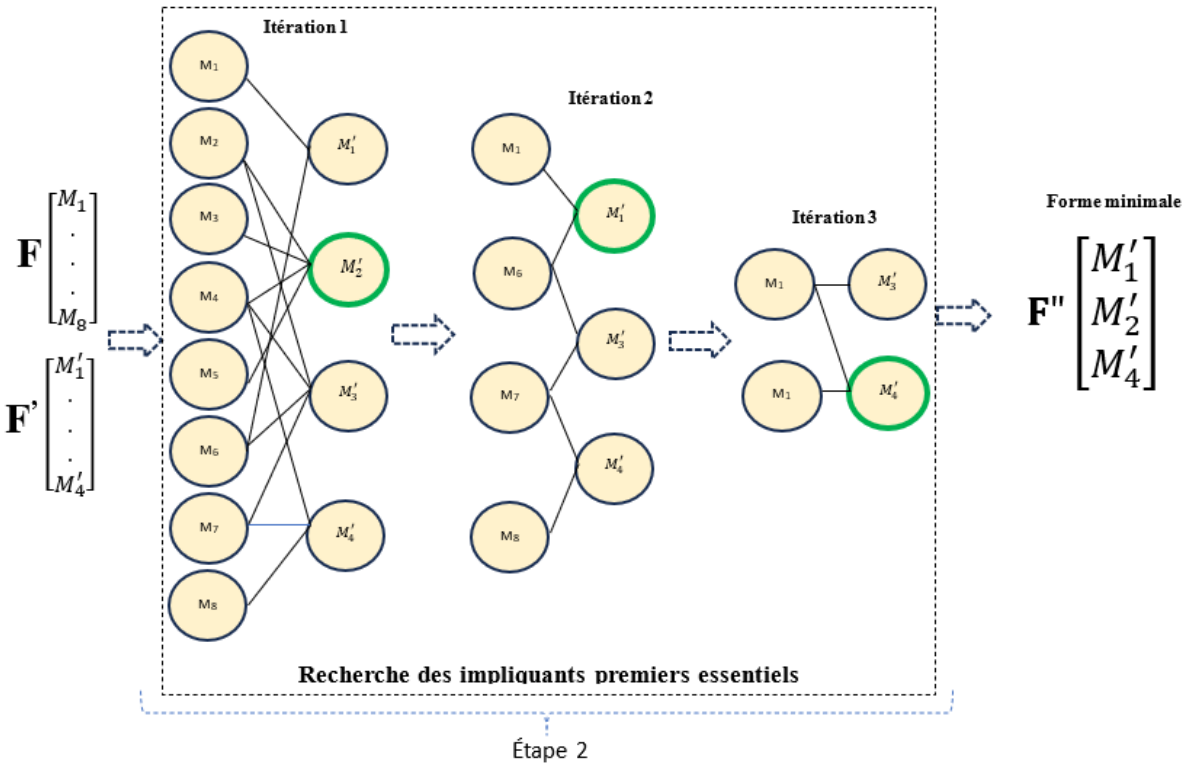
Algorithme 5.4 : Recherche des impliquants premiers essentiels (sélection)

```
1:  IP : ensemble des impliquants premiers en forme décimale
2:  TD : ensemble de termes d'entrée en forme décimale
3:  IPE : une liste pour ajouter les impliquants premiers essentiels
4:  Victim : ensemble pour ajouter chaque valeur unique des IPE sélectionnés
5:  Max-len : 1
6:  Tant que Max-len est différent de 0 Faire
7:      Max-len = 0
8:      Pour chaque prime dans IP Faire
9:          primeVal = valeurs communes dans prime et TD
10:         Si primeVal est égal TD Alors
11:             Temp = prime
12:             Arrêter la recherche
13:         Sinon
14:             Sélectionner les valeurs de primeVal qui ne sont pas dans Victim
15:             Si la taille de primeVal est supérieure à Max-len Alors
16:                 Max-len = taille de primeVal
17:                 Temp = prime
18:             Fin Si
19:         Fin Si
20:     Fin Pour
21:     Ajouter Temp dans Victim
22:     Ajouter Temp dans IPE
23:     Supprimer Temp dans IP
24: Fin Tant que
```

Dans l'algorithme 5.4, nous avons en entrée les ensembles IP et TD des impliquants premiers et des formes décimales des termes d'entrée. Dans les étapes 6 à 20, l'impliquant premier qui couvre le plus de valeurs de TD est sélectionné. Pour y arriver, les valeurs communes entre chaque «prime» de IP et l'ensemble TD sont récupérées par la variable «primeVal». Si les valeurs de « primeVal » sont identiques aux valeurs TD, cela signifie que le « prime » actuel couvre la fonction d'entrée et il est sélectionné. Sinon, les valeurs de TD sélectionnées et ajoutées à la variable «Victim» sont retirées de «primeVal». Ce qui permet de trouver un nouveau «prime» qui couvre le plus de valeurs de TD. Ensuite, le «prime» sélectionné est ajouté à l'ensemble «Victim» et à la liste des impliquants premiers essentiels IPE, puis supprimé de l'ensemble IP. La valeur 0 de la variable «Max-len» à la sortie valide la couverture de la fonction et l'arrêt des opérations.

La figure 5.2 ci-dessous représente le fonctionnement du processus de recherche des impliquants premiers essentiels.

FIGURE 5.2 : Quine McCluskey (impliquants premiers essentiels)



5.5 Conclusion

Cette partie de notre travail était consacrée à la présentation de l'algorithme Quine McCluskey pour le problème de recherche de la forme minimale d'une fonction booléenne. Nous avons commencé à présenter l'historique et les principes fondamentaux de l'algorithme, notamment ses concepts de base et ses principes de fonctionnement. Ensuite, nous avons expliqué la procédure d'application de l'algorithme en nous appuyant sur un exemple pratique. Comme le montre notre exemple, il s'agit d'une méthode très simple et facile à développer. En plus, l'algorithme est extrêmement efficace pour minimiser une fonction. Il ne requiert aucune définition de seuil ni de paramètre arbitraire pour atteindre son but.

Dans le prochain chapitre, nous présenterons nos approches de minimisation d'une fonction booléenne à partir de l'algorithme Quine McCluskey, puis nous les appliquerons à un ensemble de règles floues.

Chapitre 6 : Méthodologie

Chapitre 6

Méthodologie

6.1 Introduction

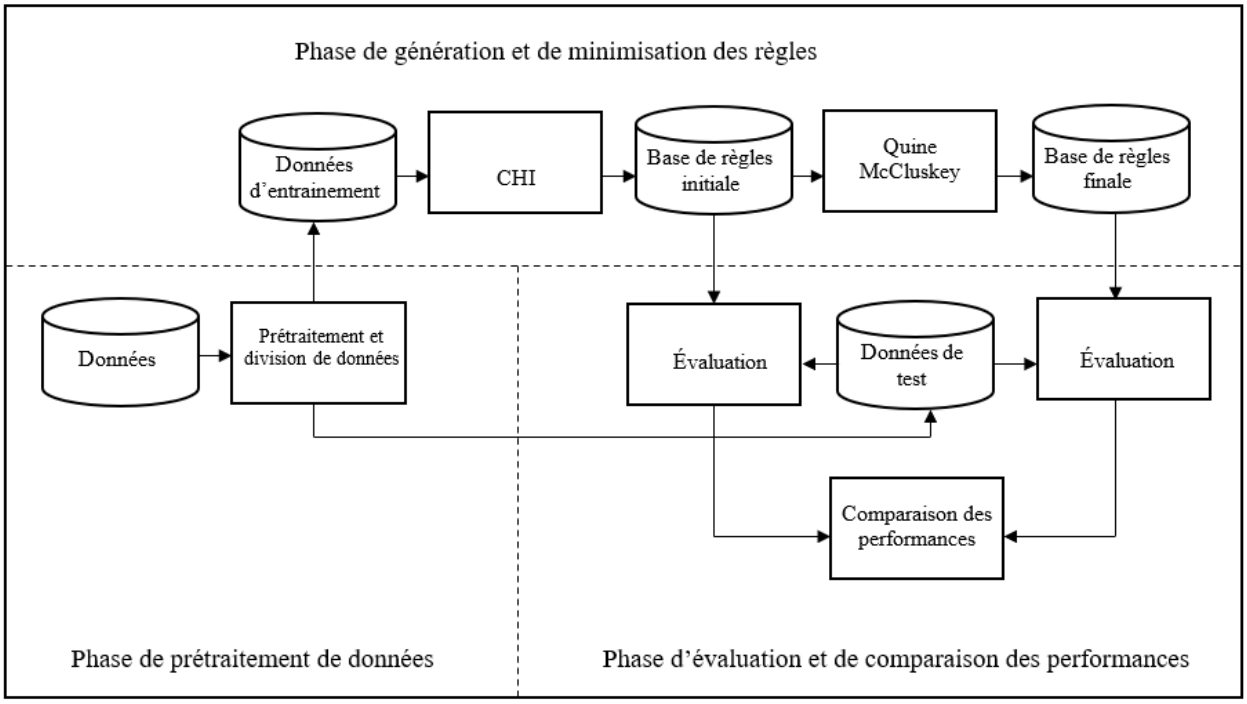
Ce chapitre explore notre plan méthodologique. Premièrement, nous allons présenter la structure générale et la procédure d'exécution du système proposé. Ensuite, nous allons détailler nos approches de prétraitement de données et de minimisation d'une base de règles floues. Nous terminerons par les étapes d'évaluation et de mise en œuvre du système.

6.2 Structure de notre système

La structure générale de notre système est composée des phases suivantes (Voir la figure 6.1) :

- Une phase de prétraitement des données ;
- Une phase d'extraction et de minimisation des règles floues ;
- Et une phase d'évaluation, de comparaison et de validation du système ;

FIGURE 6.1 : Structure générale de notre système



Notre système (Voir la figure 6.1) accepte en entrée une matrice de données numériques X uniquement pour le problème de classification. La première tâche du système est le prétraitement des données, à savoir la sélection des caractéristiques informatives et l'augmentation des données des classes minoritaires pour équilibrer le jeu de données. Ces opérations sont réalisées par les algorithmes d'analyse en composantes principales (ACP) et SMOTE. La deuxième tâche consiste à extraire des règles floues à partir des données traitées X_{tr} dans la base de règles initiale RB_i du système et à minimiser RB_i en une base de règles finale RB_f respectivement par les algorithmes CHI et QM. Les données de test X_{tst} sont utilisées par la suite pour évaluer RB_i et RB_f. L'algorithme 6.1 présente la procédure d'exécution du système.

Algorithme 6.1 : Programme du système

Procédure : Program(X)

Variables

X : ensemble de données

X_f : données des fonctionnalités sélectionnées

X_g : données générées

X_{tr} : données d'entrainement

X_{tst} : données de test

P_i, P_f : performances

Début

F_k = selectFeatures(X, t%) algorithme 6.2

$X_f = X[F_k]$

IF X_f est déséquilibré **Alors**

X_{eq} = dataGeneration(X_f, k_1, k_2, max_iter) algorihme 6.3

Fin Si

X_{tr}, X_{tst} = Division(X_{eq})

$Rb_i = CHI(X_{tr})$ algorihme 3.1

$Rb_f = QM(Rb_i)$ algorithme 6.4 et 6.5

P_i = évaluation(X_{tst}, Rb_i)

P_f = évaluation(X_{tst}, Rb_f)

Comparer(P_i, P_f)

Fin

6.3 Phase de prétraitement de données

Malgré l'efficacité des modèles d'apprentissage automatique, face à certains défis liés aux données, ces algorithmes auront du mal à produire des résultats fiables. Le déséquilibre et la dimensionnalité élevée des données sont deux problèmes de données très connus. Étant donné que notre système utilise les données pour extraire les règles floues initiales, il est alors essentiel d'améliorer la qualité de ces données. Notre stratégie de prétraitement des données est composée de deux phases utilisant respectivement les algorithmes d'ACP et SMOTE présentés dans le chapitre 2.

Sélection de fonctionnalités basée sur l'ACP-SVD

Le but de l'ACP est d'extraire de nouvelles caractéristiques dans une matrice de données numériques. Néanmoins, cela va à l'encontre de l'un des facteurs clés recherchés dans les FRBCS, soit l'interprétabilité. Au lieu de cela, nous avons adopté la technique proposée dans [48], qui consiste à modifier l'opération d'extraction de nouvelles caractéristiques en une opération de sélection des fonctionnalités informatives. Comme exemple d'illustration, nous allons utiliser les matrices de valeurs singulières Σ et de vecteurs singuliers droits V obtenus par ACP-SVD de l'exemple présenté dans le chapitre 2.

Exemple :

Reprenons les matrices des valeurs singulières Σ ainsi que les vecteurs singuliers droits V.

$$\Sigma = \begin{bmatrix} 3.45039668 & 0. & 0. \\ 0. & 1.74165799 & 0. \\ 0. & 0. & 0.24777044 \end{bmatrix} \qquad V = \begin{bmatrix} 0.47246133 & 0.63001172 & 0.61633231 \\ 0.87872325 & -0.2827555 & -0.38457091 \\ 0.06801283 & -0.72328042 & 0.68719698 \end{bmatrix}$$

À partir de la matrice Σ, déterminons le vecteur des valeurs propres $\Sigma^{'}$ correspondant aux valeurs singulières en utilisant l'équation (2.4)(a).

$$\Sigma^{'} = [2.38104745, 0.60667451, 0.01227804]$$

Ensuite, déterminons le vecteur des variances expliquées W par les composantes principales en appliquant l'équation (2.4)(b) sur le vecteur $\Sigma^{'}$.

$$W \approx [0.79368248, 0.20222484, 0.00409268]$$

Nous allons maintenant construire la matrice des poids F_w, où chaque ligne i est associée à la caractéristique j des données d'origine [48].

$$F_w = W * V^T = \begin{bmatrix} 3.74984284e{-}01 & 1.77699665e{-}01 & 2.78354705e{-}04 \\ 5.00029268e{-}01 & -5.71801847e{-}02 & -2.96015482e{-}03 \\ 4.89172161e{-}01 & -7.77697895e{-}02 & 2.81247691e{-}03 \end{bmatrix}$$

Déterminons ensuite le vecteur de score des variables d'origine en faisant la somme des colonnes de la matrice de poids, puis la valeur absolue des valeurs obtenues [48]. Le vecteur de scores déterminé est le suivant :

$$F_w = [0.5529623, 0.43988893, 0.41421485]$$

Le vecteur de score F_W est ensuite trié par ordre décroissant des valeurs de scores. À partir du vecteur de scores trié, les fonctionnalités d'origine les plus informatives, correspondant aux k premiers composants, sont sélectionnées [48].

Équilibrage de données basé sur GMM-SMOTE

SMOTE est une technique de génération de données très populaire. Toutefois, l'algorithme a tendance à propager les données bruyantes. Ce qui peut affecter la qualité d'un jeu de données mal distribué. Pour prévenir ces lacunes de SMOTE, nous utilisons une méthode de filtrage des données basée sur le travail proposé dans [16].

Bien avant le filtrage des données, nous analysons tout d'abord chaque classe de données et effectuons une synthèse initiale dans toute classe dont le taux de déséquilibre est extrêmement élevé par rapport à la classe majoritaire. Le but de cette opération est d'éviter l'élimination de la classe correspondante par les opérations de filtrage. Nous supposons qu'une classe est fortement déséquilibrée si le nombre d'échantillons de cette dernière est inférieur à la moitié de celui de la classe majoritaire. Dans ce cas, nous augmentons les données de la classe concernée à la moitié de la classe majoritaire en utilisant l'équation suivante [16] :

$$\left|\left|X\right|\right|_{syn\_min} = \sum_{j=1}^{N} \left|\left|X\right|\right|_{j\_min} * OR \tag{6.1}$$

Où $\left|\left|X\right|\right|_{syn\_min}$ est le nombre d'échantillons à synthétiser, $\left|\left|X\right|\right|_{j\_min}$ est le nombre d'échantillons minoritaires et *OR* est le taux d'échantillonnage de données défini comme suit :

$$OR = \left(\frac{\sum_{j\_maj}^{M} \left|\left|X_{j\_maj}\right|\right| / 2}{\sum_{j\_min}^{N} \left|\left|X_{j\_min}\right|\right|} - 1 \right) * 100\% \tag{6.2}$$

Où $\left|\left|X\right|\right|_{j\_maj}$ est le nombre d'échantillons majoritaires.

Exemple :

Supposons que nous avons une base de données de 20 échantillons, dont 16 pour la classe majoritaire (M) et 4 pour la classe minoritaire (N). Le nombre d'échantillons minoritaires à synthétiser pour obtenir la moitié de la classe majoritaire est :

$$\begin{aligned} OR &= \left(\frac{16/2}{4} - 1 \right) * 100\% = 100\% \\ \left|\left|X\right|\right|_{syn\_min} &= 4 * 100\% = 4 \end{aligned}$$

Après cette première évaluation, nous utilisons le modèle de mélange de Gaussien (GMM) pour classer les données en K clusters. Ensuite, nous filtrons deux groupes de données dans l'ensemble de données d'origine (voir la figure 6.2[a]), comme suit :

1. Les échantillons de classe majoritaire dans un cluster où les échantillons de classe minoritaire sont les plus nombreux sont définis comme les données bruyantes (voir la figure 6.2[b]). Ces échantillons sont filtrés et supprimés dans les clusters (voir la figure 6.2[c]). L'équation suivante est appliquées [16] :

$$\begin{aligned} \text{Si} \quad Min\left(\sum_{j=1}^{M_1} x_{j-maj}^{1} \right) &< \sum_{j=1}^{N_k} x_{j-min}^{k} \quad alors \\ X_{bruyant} &= x_{j-maj}^{1}\Big|_{j=1}^{M_1}; \quad k = 1, \ldots, K \end{aligned} \tag{6.3}$$

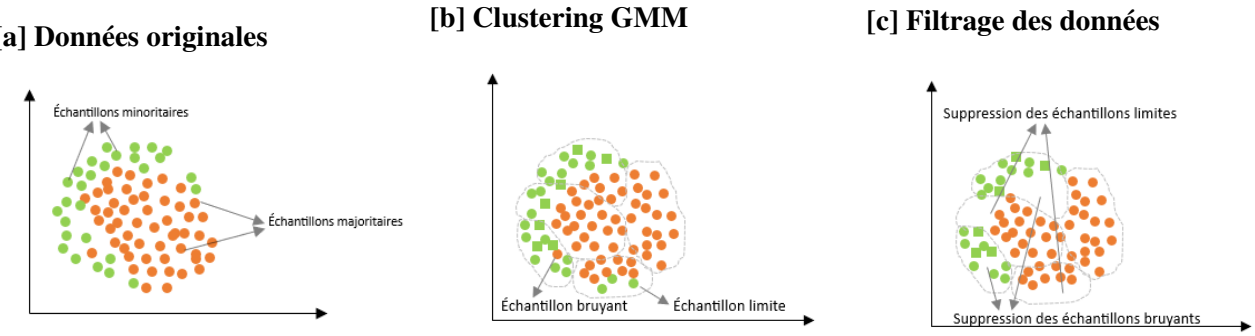
   Où $x_{j\_maj}^{1}$ est un échantillon majoritaire d'origine, $x_{j\_min}^{k}$ est un échantillon minoritaire d'origine, M_1 et N_k sont respectivement les nombres d'échantillons majoritaires et minoritaires dans les K clusters.

2. De même, les échantillons de classe minoritaire dans un cluster où les échantillons majoritaires sont les plus nombreux sont définis comme les données limites (voir la figure 6.2[b]). Elles sont également filtrées et supprimées dans les K clusters (voir la figure 6.2[c]). Pour cette tâche, la relation suivante est utilisée [16] :

$$\begin{aligned} \text{Si} \quad Min\left(\sum_{j=1}^{N_1} x^1_{j\_min}\right) &< \sum_{j=1}^{M_k} x^k_{j\_maj} \quad alors \\ X_{limites} &= x^1_{j\_min}\Big|_{j=1}^{N_1}; \quad k = 1,\ldots,K \end{aligned} \tag{6.4}$$

Où $x^1_{j\_min}$ est un échantillon minoritaire d'origine, $x^k_{j\_maj}$ est un échantillon majoritaire d'origine, N_1 et M_k sont respectivement les nombres d'échantillons minoritaires et majoritaires dans les K clusters.

FIGURE 6.2 : Filtrage des données par GMM



Après le filtrage, les échantillons majoritaires $X^k_{j\_maj}$ des K sous-classes sont combinés dans leur classe d'origine par l'équation suivante :

$$\begin{aligned} \text{Si} \quad Class_{j\_maj} &= Class_{j\_k} \quad alors \\ X_{j\_maj}\Big|_{j=1}^{M_{del}} &= x^k_{j\_maj}\Big|_{j=1}^{M_k}; \quad k = 1,\ldots,K \end{aligned} \tag{6.5}$$

Où $Class_{j\_maj}$ est l'étiquette de classe majoritaire d'origine, $Class_{j\_k}$ est l'étiquette de classe de l'échantillon majoritaire dans la sous-classe k, M_{del} est le nombre d'échantillons majoritaires après la combinaison, M_k est le nombre d'échantillons majoritaires dans la sous-classe k avant la combinaison. Après la combinaison des échantillons majoritaires, les échantillons restants sont des échantillons minoritaires et ils sont aussi combinés dans leur classe d'origine.

Exemple de filtrage de données basé sur GMM

Soit un ensemble de données déséquilibré avec 11 exemples et 2 classes. La classe 2 est la classe majoritaire, car elle a plus d'exemples que la classe 1.

$$\begin{bmatrix} e_1 & 1 \\ e_2 & 2 \\ e_3 & 1 \\ e_4 & 1 \\ e_5 & 2 \\ e_6 & 1 \\ e_7 & 1 \\ e_8 & 2 \\ e_9 & 2 \\ e_{10} & 2 \\ e_{11} & 2 \end{bmatrix}$$

Supposons que les échantillons sont classés en trois clusters par GMM dans les tableaux suivants :

$$\begin{bmatrix} e_4 & 1 & 1 \\ e_9 & 2 & 1 \\ e_1 & 1 & 1 \\ e_6 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} e_3 & 1 & 2 \\ e_{10} & 2 & 2 \\ e_5 & 2 & 2 \end{bmatrix} \qquad \begin{bmatrix} e_{11} & 2 & 3 \\ e_2 & 2 & 3 \\ e_7 & 1 & 3 \\ e_8 & 2 & 3 \end{bmatrix}$$

Pour filtrer les échantillons bruyants et limites dans ces données, chaque cluster est analysé en utilisant l'équation (6.3) ou (6.4). Par exemple, dans le premier cluster, il est clair que les échantillons minoritaires (e_4, e_1, e_6) sont plus nombreux que l'échantillon majoritaire (e_9). Dans ce cas, l'exemple (e_9) est défini comme un échantillon bruyant. Dans le deuxième cluster, l'exemple (e_3) est aussi défini comme l'échantillon limite, car les exemples de la classe majoritaire sont les plus nombreux dans ce cluster. En effectuant les mêmes opérations sur le troisième cluster, les données après le filtrage sont les suivantes :

$$\begin{bmatrix} e_4 & 1 & 1 \\ e_1 & 1 & 1 \\ e_6 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} e_{10} & 2 & 2 \\ e_5 & 2 & 2 \end{bmatrix} \qquad \begin{bmatrix} e_{11} & 2 & 3 \\ e_2 & 2 & 3 \\ e_8 & 2 & 3 \end{bmatrix}$$

Ensuite, l'équation (6.5) est utilisée pour combiner les classes de données comme suit :

$$\begin{bmatrix} e_4 & 1 \\ e_1 & 1 \\ e_6 & 1 \\ e_{10} & 2 \\ e_5 & 2 \\ e_{11} & 2 \\ e_2 & 2 \\ e_8 & 2 \end{bmatrix}$$

Maintenant que les données sont filtrées, le nombre d'échantillons à synthétiser pour chaque classe est déterminé par rapport au nombre d'échantillons majoritaire d'origine. Pour ce faire, un nouveau taux dynamique est défini comme suit :

$$OR_{new} = \left(\frac{\sum_{j\_maj}^{M} \left|\left| X_{j\_maj} \right|\right|}{\sum_{j\_min}^{N_{del}} \left|\left| X_{j\_min} \right|\right|} - 1 \right) * 100\% \tag{6.6}$$

Le nombre d'échantillons à synthétiser dans chaque classe de données est alors déterminé par la relation suivante :

$$\left|\left|X\right|\right|_{syn\_min} = \sum_{j=1}^{N} \left|\left|X\right|\right|_{j\_min} * OR_{new} \tag{6.7}$$

Exemple :

Reprenons l'exemple ci-dessus, les données d'origine contiennent 11 exemples avec 6 pour la classe majoritaire. Après le filtrage, il ne reste que 5 échantillons de la classe majoritaire et 3 pour la classe minoritaire. Maintenant, nous devons mettre à niveau chaque classe de données par rapport à la classe majoritaire d'origine. Pour la classe majoritaire :

$$\begin{aligned} OR_{new} &= \left(\frac{6}{5} - 1\right) * 100\% = 20\% \\ \left|\left|X\right|\right|_{syn\_min} &= 5 * 20\% = 1 \end{aligned}$$

Comme nous pouvons le voir, il nous manque un échantillon à synthétiser dans la classe majoritaire pour la remettre à niveau. Par la même procédure, 3 échantillons minoritaires seront synthétisés pour mettre à niveau la classe minoritaire. En combinant les échantillons filtrés et synthétisés de chaque classe de données, nous obtiendrons précisément 6 échantillons pour la classe minoritaire (3 + 3) et 6 échantillons pour la classe majoritaire (5 + 1). Ce qui assure l'équilibre du jeu de données initial. Toutes les données sont synthétisées par l'algorithme SMOTE présenté dans le chapitre 2.

6.4 Phase de génération et de minimisation du nombre de règles floues

Lors de la génération d'une base de règles initiale par l'algorithme Chi, nous avons utilisé trois étiquettes linguistiques associées à trois fonctions d'appartenance triangulaire pour chaque caractéristique de données. Par conséquent, une base de règles est formée par des groupes de règles et chaque groupe est constitué par un ensemble de règles de même conséquent. Sur ce, nous utilisons une stratégie de minimisation comme celle proposée dans [6], soit la réduction des règles ayant le même conséquent. La procédure d'application de notre approche comprend trois points essentiels :

1- Codage des étiquettes linguistiques. L'objectif de codage est de transformer les règles en forme binaire, soit le modèle accepté par l'algorithme de Quine McCluskey. Tout d'abord, le nombre de bits nécessaires pour chaque étiquette linguistique est déterminé par la relation suivante :

$$\text{Différence} = \text{Round}\Big(\log_2(L)+1\Big); \text{L est le nombre de variable linguistiques.} \tag{6.8}$$

Pour 3 étiquettes linguistiques, le nombre de bits pour coder chacune de ces étiquettes est :

$$\text{nombre de bits} = \text{Round}\Big(\log_2(3)\Big)+1 = 2 \text{ (arrondi à la baisse)}$$

Pour garantir une simplification efficace, nous avons codé deux étiquettes contiguës en gardant un seul bit de différence. Les étiquettes linguistiques et codes binaires utilisés dans ce mémoire sont les suivants :

Étiquettes	Index	Codes
Low	0	00
Medium	1	01
High	2	11

Par exemple, la règle [Medium, High, Low] est codée comme suit :

$$[\text{Medium, High, Low}] \rightarrow [1,2,0] \rightarrow [01,11,00] \rightarrow 011100$$

2- Minimisation d'un ensemble de règles de même conséquent. Dans le but d'accélérer les opérations de réduction, nous travaillons avec les valeurs décimales des règles binarisées, comme le modèle proposé dans [49]. La technique de combinaison basée sur les termes décimaux est expliquée par les étapes suivantes :

A- Deux termes de valeurs décimales appartenant à des groupes adjacents peuvent être combinés si leur différence correspond à une puissance de 2, soit 2^M, où $M \in \mathbb{N}$. La combinaison des deux termes est une paire dont les valeurs sont en ordre croissant. Par exemple : 8 et 9 forment la paire (8,9), car $8-9 = |1| = 2^0$.

B- De même, deux paires de deux groupes adjacents peuvent être combinées si les différences de leurs valeurs conjointes sont identiques et correspondent à une puissance de 2 (2^M). La combinaison des deux paires est un Quad de valeurs en ordre croissant. Par exemple, (0,1) et (8,9) forment le Quad (0,1,8,9), car $0-8=|8|=2^3$ et $1-9=|8|=2^3$.

C- L'étape B est répétée jusqu'à ce qu'il n'y ait aucune combinaison possible.

Nous avons également utilisé deux approches de minimisation en fonction de la taille de la base de règles, soit la réduction simple, soit la réduction par ajout de données.

- La réduction simple est basée sur le modèle proposé dans [6]. Il s'agit d'une technique qui utilise uniquement les données de la fonction d'entrée. Nous utilisons cette stratégie particulièrement avec les bases de données de dimensions importantes afin de réduire la complexité des opérations.

- La réduction par ajout de données s'inspire du travail proposé dans [7]. Avec cette dernière, la taille de la fonction d'entrée est modifiée en ajoutant les données "Don't Care (DC)". Cela permet de simplifier encore davantage la base de règles en éliminant certaines variables dans les règles qui n'affectent pas leurs poids, ce qui améliore la lisibilité de ces dernières. Nous utilisons particulièrement cette méthode avec les bases de données de faibles dimensions pour éviter l'explosion de la base de règles.

3- Revenir à la forme d'origine des règles après la minimisation. Il s'agit d'une opération de décodage des règles combinées en utilisant les codes suivants :

- $0*$, l'affectation à la variable est Low ou Medium ;
- $*1$, l'affectation à la variable est Medium ou High ;
- $*0$ et $1*$ n'impliquent pas de réduction, ils correspondent aux variables Low et High ;
- $**$ implique l'élimination d'une variable de la règle ;

D'un cas à l'autre, les codes de codage et de décodage sont utilisés pour remplacer les étiquettes linguistiques dans les règles combinées. Sur cette base, nous arrivons à la formulation des règles de forme d'origine simple ou étendue.

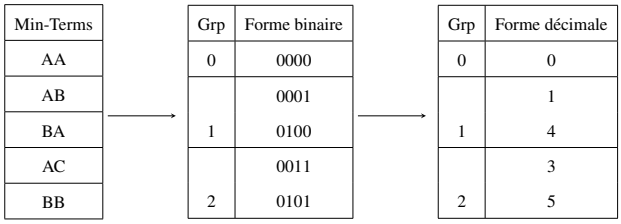
Exemple de minimisations proposées

Supposons une fonction suivante avec un ensemble de règles de même conséquent :

$$F = AA + AB + BA + AC + BB$$

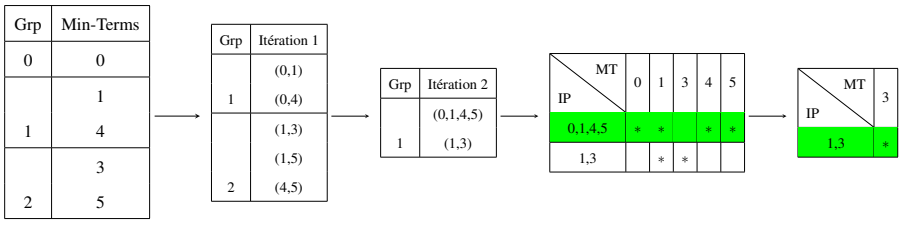
Chaque étiquette est codée comme suit : A = 00, B = 01 et C = 11.

Les étapes de codage et de transformation en forme décimale sont les suivantes :



Modèle de réduction simple

Seules les données de la fonction d'entrée sont exploitées. La procédure de minimisation est la suivante :



La dernière étape consiste à décoder les impliquants premiers essentiels pour revenir à la forme initiale des règles simplifiées de la manière suivante :

$$(0,1,4,5) \rightarrow (0000,0001,0100,0101) \rightarrow 0*0*$$

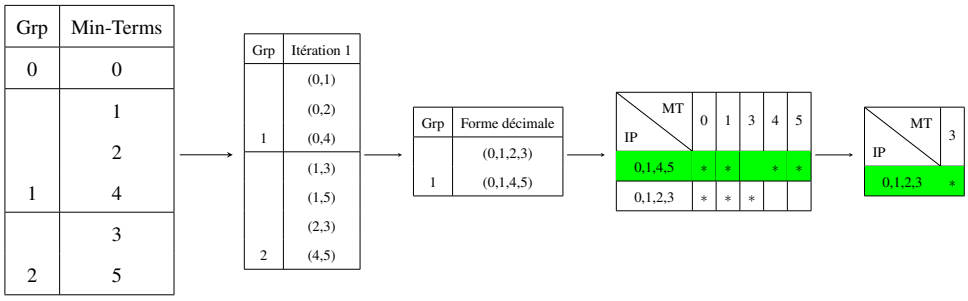
$$(1,3) \rightarrow (0001,0011) \rightarrow 00*1$$

F = [A or B][A or B] + A[B or C]

Modèle de réduction par ajout de données

L'ensemble "Don't Care (DC)" contient les données qui se situent hors de la base de règles, mais qui sont dans l'intervalle des valeurs minimale et maximale des données de la base.

Dans cet exemple particulier, il ne manque que la valeur 2 entre 0 et 5.



La phase de décodage et la fonction simplifiée sont les suivantes :

$$(0,1,2,3) \rightarrow (0000,0001,0010,0011) \rightarrow 00**$$

$$(0,1,4,5) \rightarrow (0000,0001,0100,0101) \rightarrow 0*0*$$

F = A + [A or B][A or B]

6.5 Phase d'évaluation des performances du système

L'évaluation des performances d'un modèle d'apprentissage automatique est une étape essentielle. Cela permet non seulement de vérifier les capacités du modèle, mais aussi de valider ses performances. Traditionnellement, le modèle est soumis à un test avec les nouvelles données, appelées données de test.

Une autre situation où les modèles d'apprentissage automatique sont sensibles est le changement d'état. Pour diverses raisons, l'état initial d'un modèle est souvent modifié. Dans ce cas, il est fortement recommandé de réévaluer les performances afin de s'assurer qu'elles ne sont pas négativement affectées.

Ce dernier cas correspond parfaitement à notre travail de recherche, qui vise à réduire la base de règles initiale, modifiant ainsi l'état du système. Par conséquent, nous devons évaluer les performances du système avant et après les opérations de réduction, puis les comparer dans le but de valider notre approche (voir la figure 6.1). Par un ensemble de données de tests (X_{tst}), nous évaluons la base de règles initiale (RB_i) et finale (RB_i) de notre système en suivant l'approche décrite ci-dessous.

$$X_{tst} \rightarrow S \begin{cases} RB_i \rightarrow \text{Nombre de règles, performances initiales} \\ RB_f \rightarrow \text{Nombre de règles, Performances finales} \end{cases}$$

Par la suite, nous nous servons des performances des états (initial et final) pour tirer des conclusions sur l'efficacité de notre approche de minimisation des règles floues, de la manière suivante :

- Si Performance$_i \approx$ Performance$_f$, alors la base de règles du système n'a pas été affectée négativement ;
- Si Performance$_f \ll Performance_i$, cela indique que la base de règles du système a été négativement affectée ;

6.6 Phase d'implémentation

- Langage de programmation utilisé

Dans la mise en œuvre du système, nous avons utilisé le langage de programmation Python. La raison de notre choix est que Python est un langage simple, open source et portable. En plus, Python fournit des packages et bibliothèques très riches pour l'apprentissage automatique. Nous utilisons essentiellement les bibliothèques suivantes :

1- Pandas, pour la manipulation et le traitement des tableaux de données multidimensionnels ;

2- Numpy, pour les opérations algébriques linéaires sur des matrices de données ;

3- Matplotlib, pour la visualisation des données ;

4- Scikit-learn, pour modéliser la tâche de classification ;

- Scripts Python de la méthodologie proposée

Il s'agit des fonctions Python pour la sélection de caractéristiques informatives d'un jeu de données, l'augmentation des données déséquilibrées et la réduction d'un ensemble de règles de même conséquent.

selectFeatures.py : La fonction a comme entrée une matrice de données numériques X et un taux de proportion t%. Après l'application de l'algorithme ACP_SVD, elle retourne une liste d'index des caractéristiques sélectionnées F_k.

Algorithme 6.2 : Fonction de sélection des fonctionnalités

Fonction :selectFeatures(X : tableau multidimensionnel, t% : réel)

Variables

X : matrice de données

t% : proportion

Début

X_r, Σ_r, V_r = ACP-SVD(X, t%)

Pour chaque valeur singulière σ_j dans Σ_r **Faire**

$\Sigma_r^{'}$[j] = λ_j (Équation 2.4(a))

W_r[j] = w_j (Équation 2.4(b))

Fin Pour

k = nombre de composants correspondant à $w_k \geq$ t% (Équation 2.4(c))

$F_w = V_r \times W_r$

Pour chaque attribut i dans X_r **Faire**

F_w[i] = $|\sum_{j=1}^{k} f_{i,j}|, \quad f_{i,j} \in F_w$

Fin Pour

F_w = Tri_decroissant_par_index(F_w)

F_k = select(F_w[k])

Retourner F_k

Fin

dataGeneration.py : Cette fonction prend comme entrée une matrice de données déséquilibrées X, le nombre « K_1 » de voisins proches pour l'algorithme SMOTE, le nombre « K_2 » de clusters ainsi que le nombre maximal d'itérations « max_iter » pour l'algorithme GMM. Elle retourne la matrice de données équilibrée X_{eq}.

Algorithme 6.3 : Fonction de génération des données

Fonction :dataGeneration(X : tableau, k_1 : entier, k_2 : entier, max_iter : entier)
Variables
X : ensemble de données
K_1 : nombre de KNN
K_2 : nombre de clusters
max_iter : nombre d'itérations maximal
Début
$Class_M$ = nombre d'échantillons M de classe majoritaire
Pour Chaque classe m dans X **Faire**
$Class_N$ = nombre d'échantillons N de la classe m
Si $Class_N \neq Class_M$ et N < $\frac{M}{2}$ **Alors**
t% = OR (Équation 6.2)
X_{syn} = N + N× t%
X[m] = SMOTE(X[m], X_{syn}, K_1)
Fin Si
Fin Pour
X_clusters = GMM(X, K_2, max_iter)
Pour k de 1 à K_2 **Faire**
X_clusters[k] = x_{filter} (Équation 6.3 ou Équation 6.4)
Fin Pour
X_filters = X_clusters (Équation 6.5)
Pour chaque classe m dans X_filters **Faire**
N = nombre d'échantillons de la classe m
Si N < M **Alors**
t'% = OR_{new} (Équation 6.6)
X_{syn} = N + N× t'%
X_{eq} = X_filters[m] + SMOTE(X_filters[m], X_{syn}, k_1)
Fin Si
Fin Pour
Retourner X_{eq}
Fin

ruleMinimization.py : Ce script réduit un ensemble de règles en recevant une liste de codes d'étiquettes, une base de règles initiale « Rb_i » et la base de règles de même conséquent « Rb_c ».

Algorithme 6.4 : Fonction de minimisation des règles ayant le même conséquent

Fonction :ruleMinimisation(Rb_i :liste, Rb_c :liste, Codes : liste)

Variables

Rb_i : liste de toutes les règles initiales

Rb_c : liste de règles de même conséquent

Codes : liste de codes des étiquettes

add_DC : booléenne

Début

Si add_DC est vrai **Alors**

DC : liste

Pour i de 1 à max(Rb_i) **Faire**

Si i est $\neq$ de Rb_i[i] **Alors**

DC.ajouter(i)

Fin Si

Fin Pour

Rb_c = Rb_c + DC

Fin Si

TB = Binarisation(Rb_c)

G = Regroupement(Rb_c)

IP = combinaison(G)

IPE = sélection(IP)

$Rb_c^{'}$ = IPE

Retourner $Rb_c^{'}$

Fin

getRuleForm.py : Dans cette fonction, la forme d'origine d'une règle est obtenue. La fonction reçoit en entrée une liste d'étiquettes des codes binaires et une liste de règles minimisées en forme binaire.

Algorithme 6.5 : Fonction de recherche de forme initiale d'une règle

Fonction :getRuleForm(Étiquettes : liste, $Rb_c^{'}$: liste)

Variables

Étiquettes : liste d'étiquettes des codes

$Rb_c^{'}$: liste des règles minimisées en forme binaire

Début

règle_binaire = combinaison($Rb_c^{'}$)

Règle : chaine

Pour chaque code dans règle_binaire **Faire**

Pour chaque étiquette dans Étiquettes **Faire**

Si code correspond à étiquette **Alors**

Règle = Règle + étiquette

Fin Si

Fin Pour

Fin Pour

Retourner Règle

Fin

6.7 Conclusion

Dans ce chapitre, nous avons présenté notre méthodologie de travail. Nous avons d'abord démontré nos approches de prétraitement des données, à savoir la sélection des caractéristiques informatives et l'équilibrage des classes d'un jeu de données. Par la suite, nous avons détaillé notre procédure d'application de l'algorithme Quine McCluskey, en employant deux approches : la réduction simple et la réduction par ajout de données.

Les résultats de nos études expérimentales seront présentés dans le chapitre suivant.

Chapitre 7 : Résultats d’expérimentations

Chapitre 7

Résultats d'expérimentations

7.1 Introduction

Dans ce chapitre, nous examinons en détail les résultats de nos études expérimentales. Nous avons exploré quatre jeux de données liés à la finance pour le problème de classification. Les données sont également déséquilibrées et la dimensionnalité de certaines est très élevée. C'est pourquoi nous avons introduit une phase de prétraitement des données dans le but d'améliorer leurs qualités.

Les ensembles de données exploités concernent respectivement les problèmes d'authentification des billets de banque, de vérification des enchères d'une action commerciale, de désabonnement des clients d'une société de télécommunication et de la prévision de faillite d'entreprise.

7.2 Expérimentation

Toutes nos expériences ont été réalisées au Laboratoire d'Intelligence Artificielle Appliquée (LI2A) de l'Université du Québec à Trois-Rivières sur un ordinateur équipé d'un processeur Intel Core i7-8700 CPU @ 3.20 GHz et de 8 Go de mémoire, fonctionnant sous le système d'exploitation Windows 10 x64 64.

Premièrement, nous avons effectué une analyse préliminaire des données prétraitées concernant le nombre de caractéristiques sélectionnées et le nombre d'exemples avant et après les opérations. Ensuite, nous avons examiné en profondeur les variables cruciales de notre travail de recherche, c'est-à-dire le nombre de règles et les performances du système avant et après les opérations de minimisation.

7.3 Descriptions des jeux de données d'expérimentations

Authentification de billets de banque : L'ensemble de données d'authentification des billets de banque provient de la bibliothèque d'apprentissage automatique de l'UCI. Les données sont obtenues par la photographie des spécimens des billets de banque authentiques et falsifiés. Les caractéristiques de ces images sont extraites à l'aide de divers outils de transformation en ondelettes. Cet ensemble de données est créé par Volker Lohweg (Université des sciences appliquées, Ostwestfalen-Lippe). Cependant, d'après notre connaissance, aucune information n'est disponible sur l'origine des billets de banque utilisés par l'auteur.

Le jeu de données comporte 5 attributs : la variance, l'asymétrie, le kurtosis et l'entropie de l'image, ainsi que la classe cible. Les 4 premières caractéristiques sont extraites de l'image et la cinquième est la classe cible qui comporte deux étiquettes : 0 pour les billets authentiques et 1 pour les billets falsifiés. Il y a environ 1 372 exemples de données, dont 762 billets authentiques et 610 billets falsifiés.

Lien : https://archive.ics.uci.edu/dataset/267/banknote+authentication

Vérification des enchères : Cet ensemble de données vient du travail proposé dans [50]. Les caractéristiques des données sont extraites d'un modèle de processus de l'enchère allemande du spectre 4G, la vente des bandes passantes de la bande 800 MHz. L'enchère compte 4 soumissionnaires et 6 produits. Chaque soumissionnaire dispose d'une capacité individuelle. L'objectif est de remplacer la vérification coûteuse des modèles de processus complexes par un modèle prédictif afin de détecter les résultats indésirables. L'ensemble de données comprend 130 292 lignes et 30 colonnes. Toutefois, les auteurs exploitent des sous-ensembles de ces données dans trois scénarios de prédiction différents : le résultat de la vérification pour la classification, le temps de vérification et le chiffre d'affaires pour la régression.

Dans ce travail, nous utilisons le premier scénario qui convient à notre étude. Ce sous-ensemble de données comprend 2 043 instances. Chaque instance de données est une série de vérifications évaluant la possibilité d'un prix donné pour un produit et, dans certains cas, déterminant le soumissionnaire potentiel gagnant à ce prix. Deux groupes de caractéristiques composés de 7 attributs décrivent les données : les capacités des soumissionnaires (capacité 1, capacité 2, capacité 3, capacité 4) et les propriétés actuellement vérifiées (prix, produit, gagnant). Le résultat de la vérification est la variable cible. Elle présente 1 781 cas positifs et 262 cas négatifs. L'ensemble de données a été rendu public au référentiel d'apprentissage automatique de l'UCI.

Lien : https://archive.ics.uci.edu/dataset/713/auction+verification

Désabonnement des clients de télécommunication : Les échantillons de données proviennent d'une collecte aléatoire dans la base de données d'une entreprise de télécommunication iranienne pour un groupe de 3 150 clients. Les données ont été collectées sur une période de 12 mois (de septembre 2006 à septembre 2007) de telle sorte que les clients sélectionnés n'aient pas connu de désabonnement au cours des deux premiers mois. La fin de la période d'observation pour chaque client correspond au mois au cours duquel il se désabonne dans les 10 derniers mois restants. Plus de 14 facteurs influençant la satisfaction et l'insatisfaction des clients sont analysés, notamment le nombre d'appels échoués, la durée de l'abonnement, les réclamations des clients, le montant des frais, la durée de tous les appels, le nombre d'appels, la fréquence des SMS, le nombre d'appels distincts, le type de service et l'âge du client. Le désabonnement du client est la variable cible, avec 2 655 abonnés et seulement 495 désabonnements.

Lien : https://archive.ics.uci.edu/dataset/563/iranian+churn+dataset

Prévision de faillite d'entreprise : L'ensemble de données provient du Taiwan Economic Journal pour la période de 1999 à 2009 avec 6 819 entreprises. La base de données comprend plus de 95 caractéristiques de ratios financiers. La variable cible des données est la faillite, avec 6 599 entreprises stables et 220 entreprises en difficulté financière. Les informations détaillées sur les attributs de données sont accessibles sur Kaggle ou l'UCI (voir le lien ci-dessous).

Lien : https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction

7.4 Résultats de prétraitement de données

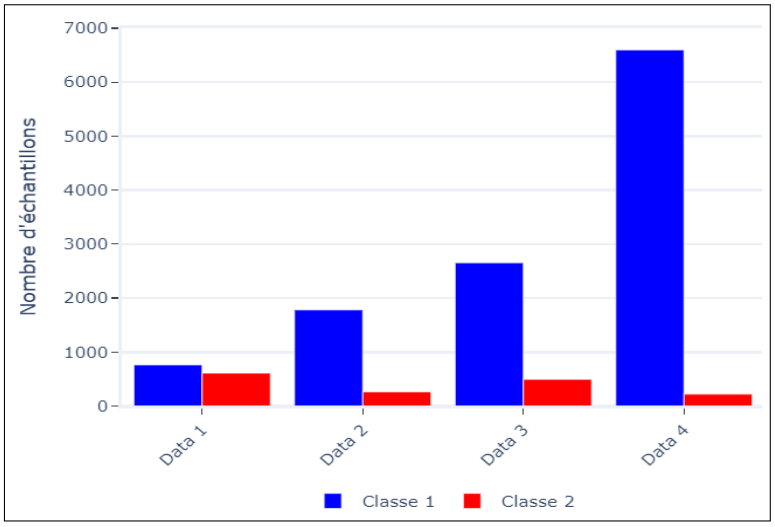
Le tableau 7.1 présente les informations sur les données, notamment le rapport entre les classes, le nombre d'exemples, le nombre de caractéristiques avant et après les opérations de prétraitement, ainsi que le taux de déséquilibre des données.

TABLEAU 7.1 : TABLEAU DES RÉSULTATS DE PRÉTRAITEMENT DE DONNÉES

Ensembles de données	Classes de données	Données avant le prétraitement			Données après le prétraitement			Taux de déséquilibre
		Rapport entre les classes	Nombre d'exemples	Nombre de caractéristiques	Rapport entre les classes	Nombre d'exemples	Nombre de caractéristiques	
Autentification des billets de banque	2 Authentique/Falsifié	762 / 610	1372	4	762 / 762	1 524	3	1.25
Vérification des enchères	2 True/False	1 781 / 262	2 043	9	1 781 / 1 781	3 562	8	6.8
Désabonnement des clients de tétécommunication	2 Oui/Non	495 / 2 655	3 150	13	2 655 / 2 655	5 310	9	5.36
Faillite d'entreprise	2 Stable/Faillite	6 599 / 220	6 819	95	6 599 / 6 599	13 198	9	30

Dans le tableau 7.1, le taux de déséquilibre de chaque jeu de données exprime à quel point la classe majoritaire est supérieure à la classe minoritaire (voir la figure 7.1). Pour les données de faillite d'entreprise, par exemple, nous pouvons constater que les classes sont fortement déséquilibrées avec un taux de 30%. Après les opérations de prétraitement, nous pouvons également remarquer que la classe minoritaire de chacun des ensembles de données a été remise au même niveau que la classe majoritaire.

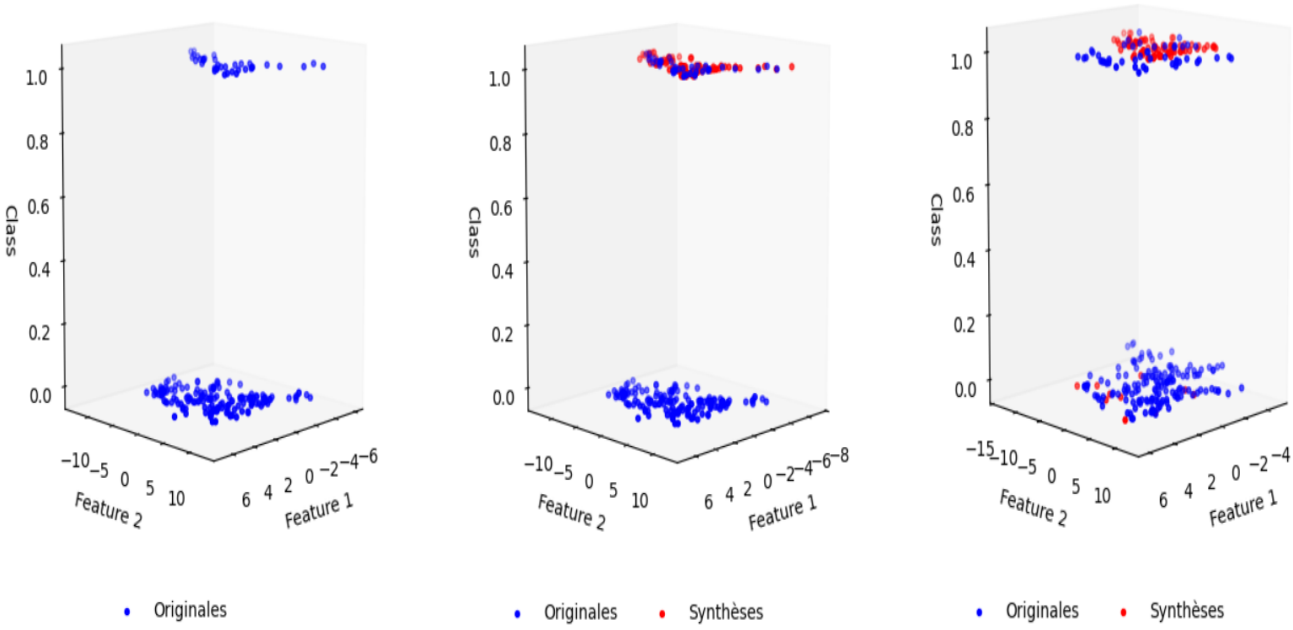
FIGURE 7.1 : Classes de données d'origine



Nous proposons également une visualisation des données synthétisées par l'algorithme SMOTE-GMM. Par exemple, pour les données d'authentification des billets de banque, nous pouvons remarquer quelques « échantillons bruyants » qui s'éloignent nettement de la distribution normale des données dans les deux classes (voir la figure 7.2[a]). Comme nous l'avons mentionné précédemment, l'algorithme SMOTE ne peut traiter ce problème. En effet, il introduit davantage de « données bruyantes » et de « données limites » (voir la figure 7.2[b]). Toutefois, l'algorithme SMOTE-GMM atténue ce problème en supprimant ces deux groupes de données avant de synthétiser les nouveaux points de données (voir la figure 7.2[c]).

FIGURE 7.2 : Visualisation des données générées par SMOTE-GMM

[a] Données d'origne **[b] SMOTE** **[c] SMOTE-GMM**



7.5 Évaluation et validation de notre système

Pour un examen fiable et exhaustif de notre système, nous analysons les résultats des métriques sur l'ensemble de données que nous désignons « Métriques globales », les résultats des métriques sur chaque classe de données que nous désignons également « Métriques spécifiques » et le comportement des courbes d'apprentissage.

1- **Analyse des métriques globales :** il s'agit du taux de minimisation et de l'exactitude. Les résultats de ces derniers évaluent les performances globales du système. Ce qui nous permet de mesurer la capacité et l'efficacité du système dans son intégralité.

- **Taux de minimisation :** il exprime la capacité de la méthode de réduction proposée en termes du nombre de règles avant et après les opérations de minimisation. Le taux de minimisation est défini par la relation suivante :

$$\text{Taux de minimisation (TM)} = 1 - \frac{\text{nombre de règles après la minimisation}}{\text{nombre de règles avant la minimisation}}$$

Supposons un ensemble de 15 règles minimisé en 9 règles. Le taux de minimisation de cette base de règle est :

$$\text{TM} = 1 - \frac{9}{15} = 40\%$$

- **Exactitude :** Elle mesure la proportion des exemples correctement classés par rapport à l'ensemble des cas. L'exactitude utilise quatre (4) variables à savoir :

(a) **Vrai positif (VP) :** nombre d'instances positives correctement prédites ;

(b) **Faux positif (FP) :** nombre d'instances positives incorrectement prédites ;

(c) **Faux négatif (FN) :** nombre d'instances négatives incorrectement prédites ;

(d) **Vrai négatif (VN) :** nombre d'instances négatives correctement prédites ;

La proportion d'exactitude est déterminée par la relation suivante :

$$\text{Exactitude} = \frac{\text{(VP + VN)}}{\text{(VP + VN + FP + FN)}}$$

Admettons la classification de 225 échantillons de données, dont 115 pour la classe 0 et 110 pour la classe 1. Supposons qu'un modèle a prédit les résultats suivants : VP = 106, FP = 10, FN = 9, VN = 100. L'exactitude du modèle est donc :

$$\text{Exactitude} = \frac{(106+100)}{(106+100+10+9)} = 0.92$$

2- **Analyse des métriques spécifiques :** Ce sont des métriques qui nous permettent de mesurer les performances spécifiques pour chaque classe de données.

- **Matrix de confusion :** La matrice de confusion est un excellent outil d'évaluation des performances d'un modèle prédictif pour le problème de classification. Elle sert non seulement à effectuer une analyse approfondie, mais aussi à déterminer d'autres métriques intéressantes pour une évaluation détaillée. Dans la matrice de confusion, les lignes sont des résultats prédits et les colonnes sont des résultats réels. Pour un problème de classification binaire, la matrice de confusion peut être représentée sous la forme du tableau ci-dessous.

		Valeur réelle Positive	Valeur réelle Négative
Valeur prédite	Positive	VP	FP
	Négative	FN	VN

La matrice de confusion de l'exemple précédent est la suivante :

		Valeur réelle 0	Valeur réelle 1
Valeur prédite	0	106	10
	1	9	100

- **Précision :** La précision exprime la proportion des cas positifs correctement reconnus comme positifs par rapport à l'ensemble des cas classés comme positifs et elle est déterminée par la formule suivante :

$$\text{Précision} = \frac{\text{VP}}{(\text{VP} + \text{FP})}$$

La précison de l'exemple précédent de la classe 0 est :

$$\text{Précision} = \frac{106}{(106 + 10)} = 0.91$$

- **Rappel :** Le rappel exprime la proportion des cas positifs correctement reconnus comme positifs par rapport à l'ensemble des cas positifs réels et il est déterminé par l'équation suivante :

$$\text{Rappel} = \frac{\text{VP}}{(\text{VP} + \text{FN})}$$

Le rappel de l'exemple précédent de la classe 0 est :

$$\text{Rappel} = \frac{106}{(106 + 9)} = 0.92$$

- **F1-Score :** c'est une métrique de mesure équilibrée des performances d'un modèle, car elle combine à la fois la précision et le rappel en une seule valeur, la moyenne harmonique de la précision et du rappel. Il est défini par l'équation suivante :

$$\text{F1-Score} = \frac{(2 \times \text{Précision} \times \text{Rappel})}{(\text{Précision} + \text{Rappel})}$$

Le F1-Score de l'exemple précédent de la classe 0 est le suivant :

$$\text{F1-Score} = \frac{(2 \times 0.91 \times 0.92)}{(0.91 + 0.92)} = 0.91$$

3- **Analyse du comportement des courbes d'apprentissage :** La courbe d'apprentissage est une approche standard pour évaluer le sur- ou sous-ajustement lié au problème de biais-variance, dans le but de minimiser l'erreur et d'obtenir un classificateur fiable. La courbe d'apprentissage d'un classificateur révèle son comportement et son degré de fiabilité par les trois indicateurs suivants :

- **Surajustement :** le surajustement peut être défini comme un problème de généralisation, c'est-à-dire un modèle qui a de bonnes performances avec les données d'entrainement, mais de faibles performances avec les données de test. Cela est associé à un faible biais et une variance très élévée. Ce problème se manifeste par une divergence significative entre les courbes d'apprentissage et de validation.

- **Sous-ajustement :** le sous-ajustement est un problème où le modèle a de performances de test nettement supérieures à ses performances d'entrainement, avec un biais très élévé et une faible variance. Cela peut être observé sur les courbes d'apprentissage, par une tendance à la hausse ou à la baisse continue.

- **Bon ajustement :** c'est la situation désirée où le modèle a de bonnes performances d'entrainement et de test, une généralisation parfaite, avec un biais faible (moins d'erreurs) et une variance faible (moins de variabilité). Sur les courbes d'apprentissage, nous pouvons constater le bon ajustement d'un modèle par la fusion et le chevauchement éventuels à mesure que la taille des données d'entrainement augmente.

7.6 Résultats et interprétation

Les résultats de nos expériences, issus des quatre bases de règles que nous avons désignées RB1, RB2, RB3 et RB4, sont récapitulés dans le tableau 7.2.

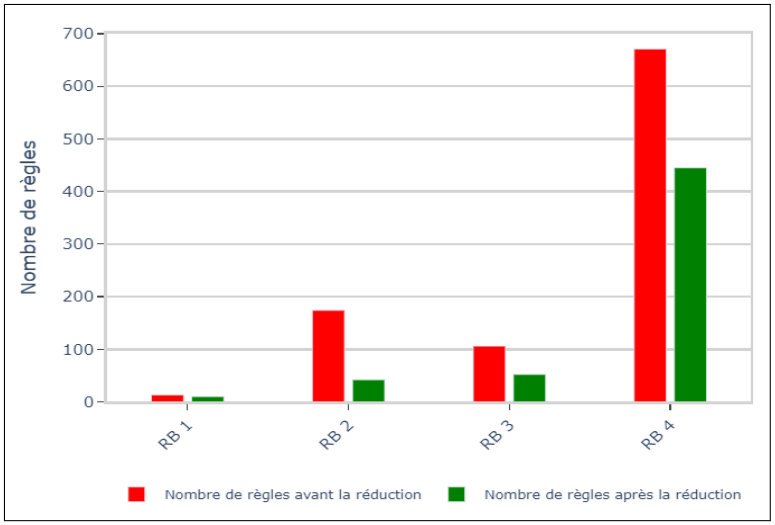
TABLEAU 7.2 : TABLEAU DES RÉSULTATS D'EXPÉRIMENTATIONS

Bases de règles	**Classes de données**	**Performances avant la réduction**							**Performances après la réduction**							**Taux de réduction**
		Matrix de confusion		Exactitude	Précision	Rappel	F1-Score	Nombre de règles	Matrix de confusion		Exactitude	Précision	Rappel	F1-Score	Nombre de règles	
RB1	0	281	21	94.10%	94.93%	93.05%	93.98%	13	286	16	96.72%	98.62%	94.70%	96.62%	10	
	1	15	293		93.31%	95.13%	94.21%		4	304		95.00%	98.70%	96.82%		
Moyenne				94.10%	94.12%	94.09%	94.10%	13	Moyenne		96.72%	96.81%	96.70%	96.72%	10	23%
RB2	0	737	0	100%	100%	100%	100%	174	737	0	98.18%	96.59%	100%	98.27%	42	
	1	0	688		100%	100%	100%		26	662		100%	96.22%	98.07%		
Moyenne				100%	100%	100%	100%	174	Moyenne		98.18%	98.30%	98.11%	98.17%	42	76%
RB3	0	1 091	0	100%	100%	100%	100%	106	1 091	0	100%	100%	100%	100%	52	
	1	0	1 033		100%	100%	100%		0	1 033		100%	100%	100%		
Moyenne				100%	100%	100%	100%	106	Moyenne		100%	100%	100%	100%	52	51%
RB4	0	2 526	108	97.35%	98.75%	95.90%	97.30%	671	2 551	83	97.82%	98.76%	96.85%	97.80%	445	
	1	32	2 614		96.03%	98.79%	97.39%		32	2 614		96.92%	98.79%	97.85%		
Moyenne				97.35%	97.39%	97.35%	97.35%	671	Moyenne		97.82%	97.84%	97.82%	97.82%	445	34%
Moyenne des règles								**241**	**Moyenne des règles**						**137**	

- Taux de minimisation des règles floues :

Comme nous pouvons le voir dans le tableau 7.2, nous avons obtenu une réduction de 23% des règles de RB1, passant de 13 à 10. De même, plus de 76% de taux de réduction ont été obtenus avec RB2. Pour les bases de règles RB3 et RB4, nous avons respectivement obtenu un taux de minimisation de 51% et de 34%. La figure 7.3 présente le degré de simplification de chacune de ces quatre bases de règles.

FIGURE 7.3 : Nombre de règles avant et après la réduction



- Exactitude du système avant et après la réduction des règles (voir la figure 7.4)

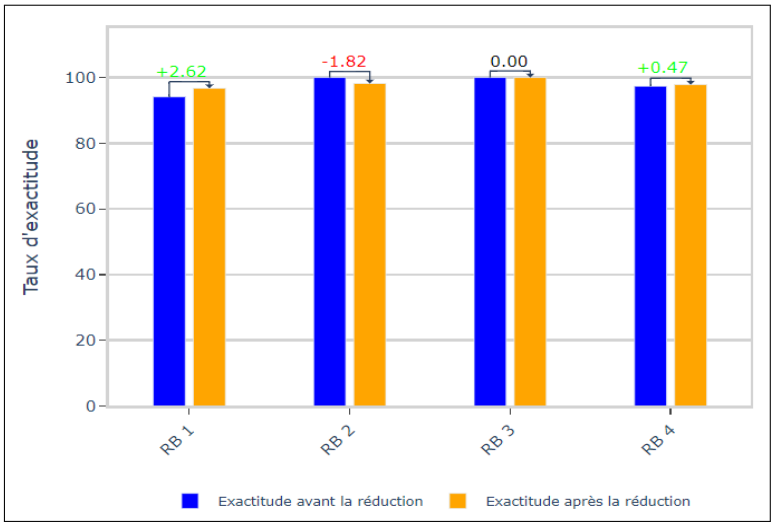
RB1 : 94.10% pour la base initiale, contre 96.72% après la réduction, avec une amélioration de 2.62% ;

RB2 : 100% pour la base initiale et 98.18% pour la base finale avec une baisse de 1.82% d'exactitude ;

RB3 : 100% avant et après la réduction. Les performances initiales ont été conservées ;

RB4 : 97.35% avant la réduction et 97.82% après la réduction avec une amélioration de 0.47% d'exactitude ;

FIGURE 7.4 : Taux d'exactitude avant et après la réduction



- Matrice de confusion avant et après la réduction des règles :

Les résultats de prédiction obtenus par la matrice de confusion de chacune des bases de règles sont détaillés ci-après.

RB1 : Dans les 610 échantillons de tests de RB1 avant la réduction, 281/302 billets authentiques et 293/308 billets falsifiés ont été correctement classés. Alors que 21/302 billets authentiques et 15/308 billets falsifiés ont été incorrectement classés. Après la réduction, la classification correcte des classes s'est améliorée, atteignant 286/302 pour les billets authentiques et 304/308 pour les billets falsifiés. La classification erronée est également réduite, avec 16/302 pour les billets authentiques et 4/308 pour les billets falsifiés.

RB2 : Avant la réduction, tous les échantillons de test de vérification d'enchères ont été parfaitement classés, soit 737 enchères vérifiées et 688 enchères non vérifiées. Après la réduction, les 737 enchères vérifiées ont été parfaitement classées. Néanmoins, seulement 662/688 enchères non vérifiées sont correctement reconnues, avec une classification erronée de 26/688 enchères non vérifiées.

RB3 : Les 2124 échantillons de tests sur la prédiction du désabonnement des clients de télécommunication sont parfaitement classés avant et après la minimisation des règles, soit 1091/1091 clients abonnés et 1033/1033 clients désabonnés.

RB4 : Avant la réduction, parmi les 5280 échantillons de tests sur la chute des entreprises, 2526/2634 cas stables et 2614/2646 cas en faillite ont été correctement reconnus. En revanche, 108/2634 cas stables et 32/2646 cas en faillite ont été prédits à tort. Après la réduction des règles, la prédiction des cas stables s'est améliorée avec 2551/2634 cas contre 83/2634. Toutefois, les résultats de prédiction des entreprises en faillite n'ont pas changé.

- Précision du système avant et après la réduction des règles (voir la figure 7.5)

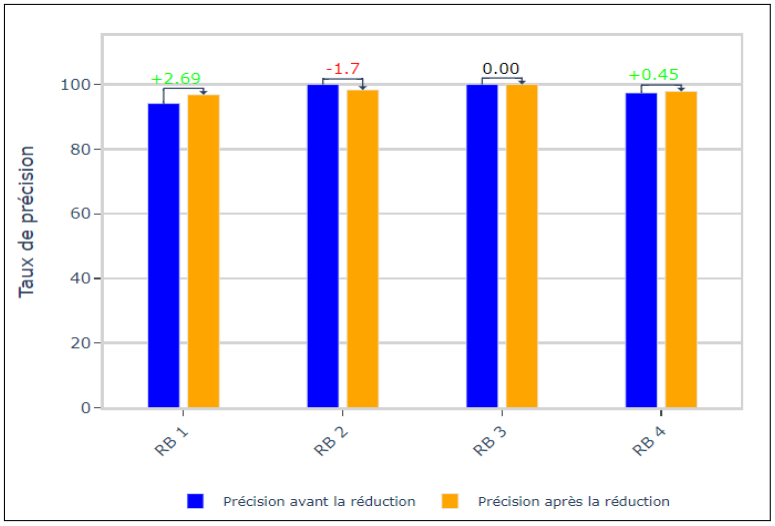
RB1 : un taux de 94.12% avant la réduction et 96.81% après, soit une amélioration de 2.69% ;

RB2 : 100% pour la base initiale et 98.30% pour la base finale avec une perte de 1.7% ;

RB3 : Les performances initiales ont été conservées, soit 100% ;

RB4 : 97.39% avant la réduction et 97.84% après, une amélioration de 0.39% ;

FIGURE 7.5 : Taux de précision avant et après la réduction



Rappel du système avant et après la réduction des règles (voir la figure 7.6)

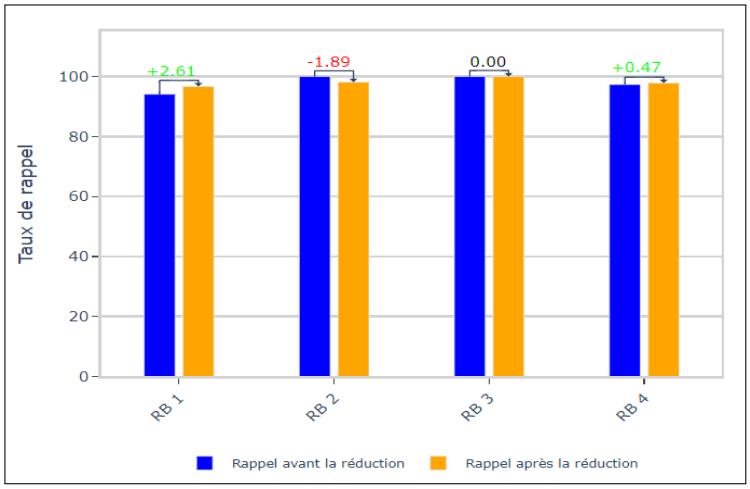
RB1 : un taux de 94,09% pour la base initiale et 96,70% pour la base finale, soit 2,61% d'amélioration ;

RB2 : 100% avant la réduction et 98.11% après la réduction, une baisse de 1.89% ;

RB3 : les taux de rappel sont les mêmes, soit 100% ;

RB4 : 97.35% contre 97.82% pour la base initiale et finale, soit 0.47% d'amélioration ;

FIGURE 7.6 : Taux de rappel avant et après la réduction



F1-Score du système avant et après la réduction des règes (voir la figure 7.7)

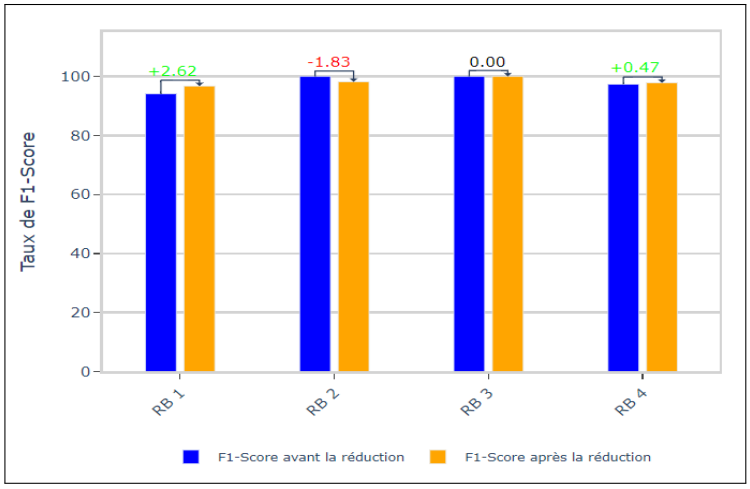
RB1 : pour les deux états du système, les taux sont 94.10% et 96.72% avec une amélioration de 2,62% ;

RB2 : nous avons 100% pour la base initiale contre 98.17% pour la base minimisée, soit une perte de 1.83% ;

RB3 : un taux de 100% pour les deux états du système ;

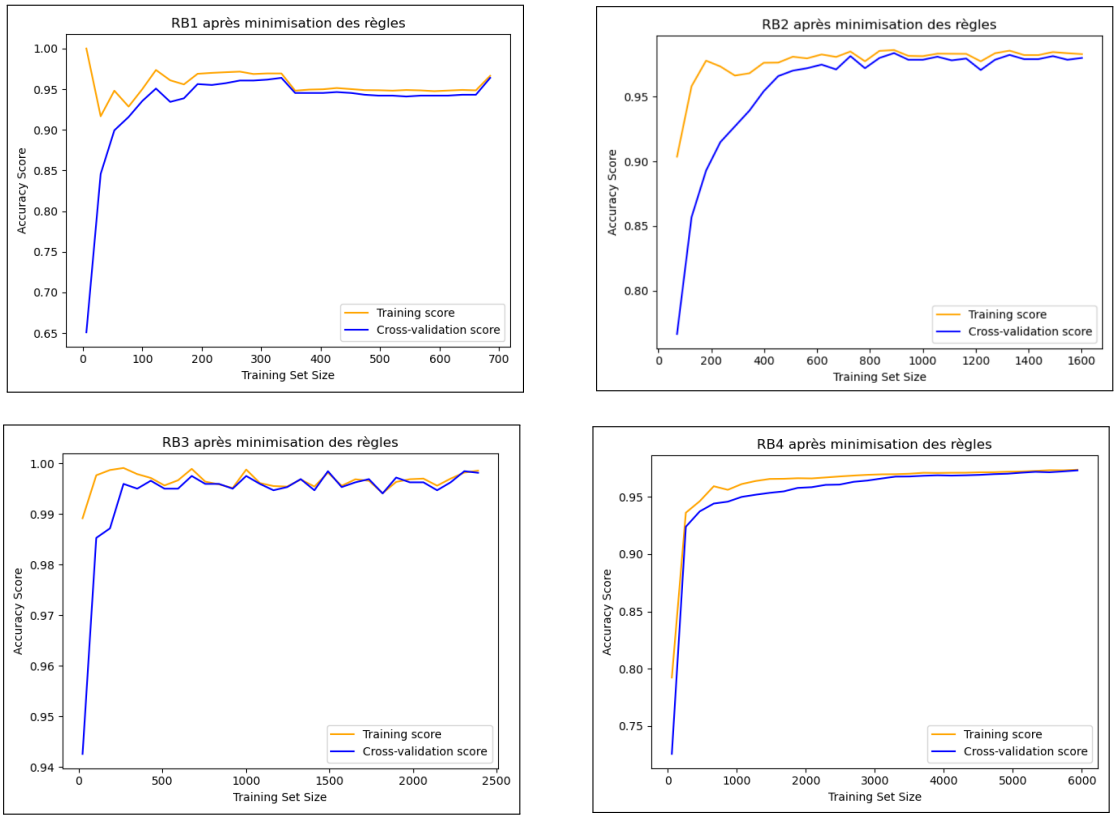
RB4 : 97.35% contre 97.82% respectivement pour la base initiale et finale avec 0.47% d'amélioration ;

FIGURE 7.7 : Taux de F1-Score avant et après la réduction



Courbes d'apprentissage du système après la réduction des règles : La figure 7.8 présente les courbes d'apprentissage de chacune de nos bases de règles après les opérations de simplification dans le système. D'après les images, nous pouvons clairement voir que les courbes d'apprentissage et de validation s'ajustent parfaitement au fur et à mesure que la taille des données d'entraînement augmente. Cela confirme les résultats obtenus par les métriques étudiées, tout en attestant la robustesse du système final. Par conséquent, nous pouvons en conclure que le système ne souffre pas de problèmes de surajustement ou de sous-ajustement et qu'il a un bon compromis biais-variance.

FIGURE 7.8 : Visualisation des courbes d'apprentissage du système final



7.7 Discussion

D'après nos analyses précédentes, il est évident que notre approche de minimisation a contribué à l'optimisation du système à tous les niveaux de notre recherche, soit :

- **La minimisation des règles :** pour la réduction de chaque base de règles, un taux de minimisation positif a été atteint. Ce qui nous a permis d'atteindre notre premier objectif en ce qui concerne la réduction de la complexité de la base de règles initiale.

- **Les performances :** nous avons constaté une amélioration nette des performances du système final pour toutes les métriques d'évaluation des bases de règles RB1 et RB4. Pour la base de règles RB3, nous avons observé les mêmes performances. En revanche, nous avons remarqué une légère diminution des performances finales de la RB2 par rapport à l'état initial du système. Cela corrobore les remarques des auteurs de l'article de la base de données originale [50], qui soulèvent les problèmes de distribution asymétrique et de limite de la qualité représentative des données.

  Nous avons également évalué et confirmé la fiabilité du système final en examinant les courbes d'apprentissage. Ces résultats contribuent à l'atteinte de notre deuxième objectif, qui est la conservation des performances comparables entre les états du système.

7.8 Conclusion

Les résultats de nos études expérimentales montrent une amélioration du système avec chaque jeu de données exploité. Pour chaque base de règles, le nombre de règles initiales a été réduit pour obtenir une base de règles finale moins complexe. En plus, l'approche de minimisation proposée a conservé les données informatives dans chaque base de règles. Ce qui a permis au système de maintenir des performances comparables dans ses deux états, avant et après la réduction des règles. Par conséquent, nous pouvons conclure que notre méthode de minimisation peut simplifier une base de règles floues initiale sans compromettre ses performances.

Chapitre 8 : Conclusion

Chapitre 8

Conclusion

Les systèmes de classification basés sur des règles floues (FRBCS) font partie de la grande famille des systèmes à base de règles floues (FRBS). Ce modèle présente un avantage considérable par rapport à ses homologues algorithmiques dans la mesure où il est capable de dériver des interprétations compréhensibles par l'homme. Par conséquent, ils sont largement utilisés, en particulier dans des secteurs critiques pour les humains, comme la santé, la justice, la finance, etc. Ces dernières années, avec la demande croissante de l'intelligence artificielle explicable, beaucoup de travaux ont été consacrés sur le développement des systèmes de classification basés sur des règles floues.

Dans ce mémoire, nous avons présenté les principaux composants utilisés dans la mise en place d'un système de classification basé sur des règles floues, ainsi que le modèle de données exploitées par ce dernier. Ensuite, nous avons présenté les algorithmes CHI et FuzzyDT comme exemples de génération de règles floues à partir d'un jeu de données en utilisant le facteur de certitude pénalisé (PCF) comme heuristique de détermination du poids des règles. Par des exemples d'application, nous avons constaté le défi soulevé dans plusieurs études, c'est-à-dire le nombre de règles produites par un classificateur flou.

Notre travail de recherche, présenté dans ce mémoire, vise à contribuer à la résolution de cette problématique relative aux systèmes flous. En effet, nous avons pensé que l'utilisation d'une méthode de minimisation efficace de la base de règles de ces derniers peut non seulement réduire sa complexité, mais aussi améliorer ses performances. Sur ce, nous avons utilisé l'algorithme de minimisation de la fonction booléenne, soit la méthode Quine McCluskey.

Avant l'application de notre approche de minimisation, nous avons d'abord introduit une phase de prétraitement de données afin d'améliorer la qualité de nos bases de données. Cela nous a aidés à réduire la taille d'un ensemble de données en sélectionnant les attributs pertinents grâce à l'analyse en composantes principales (ACP) basée sur la décomposition en valeurs singulières (SVD). De plus, nous avons également équilibré une base de données en augmentant les échantillons d'une classe minoritaire grâce à l'algorithme SMOTE, qui s'appuie sur le modèle de mélange de Gaussien (GMM).

Nous avons généré des règles floues avec les étiquettes linguistiques «Low : 0, Medium : 1, High : 2» à partir des données prétraitées. Nous avons également déterminé le nombre de bits nécessaires pour chacune de nos variables linguistiques. Ensuite, nous avons défini un codage approprié comme suit : «Low : 00, Medium : 01, High : 11» pour adapter les règles à la forme de données utilisée par l'algorithme Quine McCluskey, soit la forme binaire. Nous avons défini les codes des étiquettes de sorte qu'il y ait un seul bit de différence entre deux étiquettes contiguës. Ce qui garantit une simplification efficace. Dans une fonction de règles minimisée par l'algorithme Quine McCluskey, un astérisque ($*$) désigne l'élimination d'une variable booléenne, tandis que deux astérisques ($**$) désignent l'élimination d'une variable linguistique. Nous avons exploité les valeurs décimales dans la première étape de notre méthode de réduction dans l'objectif d'accélérer les opérations. De plus, nous avons proposé deux types de minimisation selon la taille de la fonction d'entrée : soit la minimisation simple qui n'utilise que les données d'entrée, soit la minimisation par ajout de données qui autorise les données "Don't Care (DC)" pour une réduction optimisée.

Par une étude expérimentale, les résultats prouvent que notre approche de minimisation est capable de simplifier efficacement une base de règles, tout en conservant les données informatives dans la base de règles minimisée. Elle atteint également l'objectif d'obtenir un bon compromis entre la précision et l'interprétabilité. Cela est démontré par une analyse globale et détaillée des résultats de plusieurs métriques d'évaluation des algorithmes d'apprentissage automatique. En examinant les résultats avec chaque ensemble de données, il est clair que la méthode proposée a obtenu une base de règles beaucoup plus compacte que celle générée initialement, et ce, avec des performances comparables pour les deux états du système. Ce qui confirme l'efficacité de l'algorithme Quine McCluskey, qui fonctionne plutôt sur un ensemble de principes et de logiques que sur des définitions arbitraires.

Comme travaux futurs, nous souhaitons développer un modèle hybride constitué par l'apprentissage profond et le système de classification basé sur des règles floues, puis appliquer la méthode de Quine McCluskey. Nous pensons que l'association d'un modèle d'apprentissage profond à un système flou utilisant la méthode tabulaire peut non seulement améliorer la transparence du modèle, mais aussi optimiser la lisibilité des règles floues extraites par le système hybride.

Bibliographie :

[1] Scott Thiebes, Sebastian Lins, and Ali Sunyaev. Trustworthy artificial intelligence. *Electronic Markets : The International Journal on Networked Business*, 31(2) :447–464, 2020.

[2] Natalya Shevskaya. *Explainable Artificial Intelligence Approaches : Challenges and Perspectives*, pages 540–543. 2021. (ITQMIS).

[3] Roohallah Alizadehsani, Solomon Sunday Oyelere, Sadiq Hussain, Senthil Kumar Jagatheesaperumal, Rene Ripardo Calixto, Mohamed Rahouti, Mohamad Roshanzamir, and Victor Hugo C De Albuquerque. Explainable artificial intelligence for drug discovery and development-a comprehensive survey. *IEEE Access*, 2024.

[4] Janet Adams and Hani Hagras. *A Type-2 Fuzzy Logic Approach to Explainable AI for regulatory compliance, fair customer outcomes and market stability in the Global Financial Sector*, pages 1–8. 2020. (FUZZ-IEEE).

[5] Ayush Varshney and Vicenç Torra. Literature review of the recent trends and applications in various fuzzy rule-based systems. *International Journal of Fuzzy Systems*, 25(6) :2163–2186, 2023.

[6] Leonardo Jara, Antonio Gonzalez, and Raul Perez. *A preliminary study to apply the Quine McCluskey algorithm for fuzzy rule base minimization*, pages 1–6. 2020. (FUZZ-IEEE).

[7] Ahmed Khedr, Rabie Ramadan, and Salah Abdel-Mageid. Qmr : Quine-mccluskey for rule minimization in rule-based systems. *international Journal of Intelligent Computing and Information Science*, 2012.

[8] Robert Frost. Eigenvectors from eigenvalues sparse principal component analysis (eespca). *Journal of computational and graphical statistics : a joint publication of American Statistical Association, Institute of Mathematical Statistics, Interface Foundation of North America*, 31(2) :486, 2021.

[9] Frank Mauldin, Dan Lin, and John Hossack. The singular value filter : A general filter design strategy for pca-based signal separation in medical ultrasound imaging. *IEEE transactions on medical imaging*, 30(11) :1951–1964, 2011.

[10] Antonella Falini. A review on the selection criteria for the truncated svd in data science applications. *Journal of Computational Mathematics and Data Science*, 5 :100064, 2022.

[11] Yousef Jaradat, Mohammad Masoud, Ismael Jannoud, Ahmad Manasrah, and Mohammad Alia. A tutorial on singular value decomposition with applications on image compression and dimensionality reduction. In *2021 international conference on information technology (ICIT)*, pages 769–772. IEEE.

[12] Ian Jolliffe. Principal component analysis and factor analysis. *Principal component analysis*, pages 150–166, 2002.

[13] Ansar Khan, Mrityunjoy Jana, Subhankar Bera, and Arosikha Das. Retracted article : Subject choice in educational data sets by using principal component and procrustes analysis. *Modeling Earth Systems and Environment*, 2 :1–5, 2016.

[14] Marina V Polyakova and Viktor N Krylov. Data normalization methods to improve the quality of classification in the breast cancer diagnostic system. , 5(1) :55–63, 2022.

[15] Ying Zhang, Li Deng, Hefeng Huang, and Bo Wei. An improved smote based on center offset factor and synthesis strategy for imbalanced data classification. *The Journal of Supercomputing*, 80(15) :22479–22519, 2024.

[16] Zhaozhao Xu, Derong Shen, Yue Kou, and Tiezheng Nie. A synthetic minority oversampling technique based on gaussian mixture model filtering for imbalanced data classification. *IEEE Transactions on Neural Networks and Learning Systems*, 35(3) :3740–3753, 2022.

[17] Xiang Lu, Yaoxiang Li, and Tanzy Love. On bayesian analysis of parsimonious gaussian mixture models. *Journal of Classification*, 38(3) :576–593, 2021.

[18] Mehran Azimbagirad and Luiz Otavio Murta Junior. Tsallis generalized entropy for gaussian mixture model parameter estimation on brain segmentation application. *Neuroscience Informatics*, 1(1-2) :100002, 2021.

[19] Mateusz Przyborowski and Dominik Ślęzak. Approximation of the expectation-maximization algorithm for gaussian mixture models on big data. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 6256–6260. IEEE.

[20] Diego Alvarez-Estevez and Vicente Moret-Bonillo. Revisiting the w ang–m endel algorithm for fuzzy classification. *Expert Systems*, 35(4) :e12268, 2018.

[21] Marcos Evandro Cintra and Heloisa de Arruda Camargo. Fuzzy rules generation using genetic algorithms with self-adaptive selection. In *2007 IEEE International Conference on Information Reuse and Integration*, pages 261–266. IEEE.

[22] Tianhua Chen, Changjing Shang, Pan Su, Elpida Keravnou-Papailiou, Yitian Zhao, Grigoris Antoniou, and Qiang Shen. A decision tree-initialised neuro-fuzzy approach for clinical decision support. *Artificial Intelligence in Medicine*, 111 :101986, 2021.

[23] Luis Magdalena and Juan Velasco. Fuzzy rule-based controllers that learn by evolving their knowledge base. *Genetic Algorithms and Soft Computing*, 8 :172–201, 1996.

[24] Siti Hajar Khairuddin, Mohd Hilmi Hasan, Manzoor Ahmed Hashmani, and Muhammad Hamza Azam. Generating clustering-based interval fuzzy type-2 triangular and trapezoidal membership functions : A structured literature review. *Symmetry*, 13(2) :239, 2021.

[25] Jin Cao, Ta Zhou, Shaohua Zhi, Saikit Lam, Ge Ren, Yuanpeng Zhang, Yongqiang Wang, Yanjing Dong, and Jing Cai. Fuzzy inference system with interpretable fuzzy rules : Advancing explainable artificial intelligence for disease diagnosis—a comprehensive review. *Information Sciences*, 662 :120212, 2024.

[26] Michela Antonelli, Dario Bernardo, Hani Hagras, and Francesco Marcelloni. Multiobjective evolutionary optimization of type-2 fuzzy rule-based systems for financial data classification. *IEEE Transactions on Fuzzy Systems*, 25(2) :249–264, 2016.

[27] Leonardo Jara, Antonio Gonzalez, and Raul Perez. Qchi : A faster classification algorithm based on wang–mendel algorithm. *IEEE Transactions on Fuzzy Systems*, 32(4), 2024.

[28] Ayush Varshney and Vicenç Torra. Designing distributed chi-fuzzy rule based classification system. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE.

[29] Zheru Chi, Jing Wu, and Hong Yan. Handwritten numeral recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, 28(1) :59–66, 1995.

[30] Zheru Chi, Hong Yan, and Tuan Pham. *Fuzzy algorithms : with applications to image processing and pattern recognition*, volume 10. World Scientific, 1996.

[31] Xiaolong Wang, Zhou, and Xiaolong Xu. Application of c4. 5 decision tree for scholarship evaluations. *Procedia Computer Science*, 151 :179–184, 2019.

[32] Michela Antonelli, Dario Bernardo, Hani Hagras, and Francesco Marcelloni. Multiobjective evolutionary optimization of type-2 fuzzy rule-based systems for financial data classification. *IEEE Transactions on Fuzzy Systems*, 25(2) :249–264, 2016.

[33] Chuan-Chang Hung and Benito Fernandez. Minimizing rules of fuzzy logic system by using a systematic approach. In *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*, pages 38–44. IEEE.

[34] Caro Fuchs, Simone Spolaor, Marco Nobile, and Uzay Kaymak. A graph theory approach to fuzzy rule base simplification. In *International conference on information processing and management of uncertainty in knowledge-based systems*, pages 387–401. Springer.

[35] Arshia Azam and Mohammad Haseeb Khan. Reduced rule fuzzy logic controller for performance improvement of process control. In *2013 IEEE Conference on Information Communication Technologies*, pages 894–898. IEEE.

[36] Amel Borgi, Rim Kalai, and Hayfa Zgaya. Attributes regrouping in fuzzy rule based classification systems : an intra-classes approach. In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–7. IEEE.

[37] Karsten Strehl, Claudio Moraga, K-H Temme, and Radomir S Stankovic. Fuzzy decision diagrams for the representation, analysis and optimization of rule bases. In *Proceedings 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000)*, pages 127–132. IEEE.

[38] Marco Cococcioni, Luca Foschini, Beatrice Lazzerini, and Francesco Marcelloni. Complexity reduction of mamdani fuzzy systems through multi-valued logic minimization. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 1782–1787. IEEE.

[39] Raffaele Cannone, Ciro Castiello, Corrado Mencar, and Anna Maria Fanelli. A study on interpretability conditions for fuzzy rule-based classifiers. In *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 438–443. IEEE.

[40] Tamrat Delessa Chala and László Kóczy. Intelligent fuzzy traffic signal control system for complex intersections using fuzzy rule base reduction. *Symmetry*, 16(9) :1177, 2024.

[41] Jose Maria Moral Alonso, Luis Magdalena Layos, and Serge Guillaume. Linguistic knowledge base simplification regarding accuracy and interpretability. *Mathware Soft Computing. 2006, vol. 13, núm. 3*, 2006.

[42] Fuijun Song and Samuel Smith. A simple weight based fuzzy logic controller rule base reduction method. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 5, pages 3794–3799. IEEE.

[43] Min-You Chen and Derek Linkens. Rule-base self-generation and simplification for data-driven fuzzy models. *Fuzzy sets and systems*, 142(2) :243–265, 2004.

[44] Janos Abonyi, Johannes Roubos, and Ferenc Szeifert. Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. *International journal of approximate reasoning*, 32(1) :1–21, 2003.

[45] Ronaldo Vigo. A note on the complexity of boolean concepts. *Journal of Mathematical Psychology*, 50(5) :501–510, 2006.

[46] Bohdan Borowik, Mykola Karpinskyy, Valery Lahno, and Oleksandr Petrov. *Theory of digital automata*, volume 63. Springer Science Business Media, 2012.

[47] Hoang-Gia Vu, Ngoc-Dai Bui, and Anh-Tu Nguyen. Performance evaluation of quine-mccluskey method on multi-core cpu. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 60–64. IEEE.

[48] Aminata Kane and Nematollaah Shiri. Selecting the top-k discriminative features using principal component analysis. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 639–646. IEEE.

[49] Alak Majumder, Barnali Chowdhury, Abir J Mondai, and Kunj Jain. Investigation on quine mccluskey method : A decimal manipulation based novel approach for the minimization of boolean function. In *2015 International Conference on Electronic Design, Computer Networks Automated Verification (EDCAV)*, pages 18–22. IEEE.

[50] Elaheh Ordoni, Jakob Bach, and Ann-Katrin Fleck. Analyzing and predicting verification of data-aware process models–a case study with spectrum auctions. *IEEE Access*, 10 :31699–31713, 2022.