République Algérienne démocratique et populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université YAHIA FARES De Médéa

Faculté des Sciences

Projet de fin d'étude présenté en vue de l'obtention du diplôme de Master en informatique

Spécialité : ***Ingénierie des systèmes et technologie du web***

Thème :

Segmentation hybride à base de différentes techniques de clustering

---

Proposé et dirigé par :

Mr ***BOUDJEBBOUR Karim***, Encadreur

Mr ***HAYI Mohamed Yassine***, Co-encadreur

Réalisé et présenté par :

Mr ***BOUCHENAFA Islem***

Melle ***BOURACHEDI Assia***

Soutenu le 15-09-2020 devant le jury composé de :

Madame ***SADOUKI Fatiha***, Présidente

Mr ***SAHMADI Brahim***, Examinateur

PROMOTION
2019 - 2020

« Remercîments »

*En premier lieu, nous adressons nos sincères et chaleureux remerciements à **DIEU** le tout puissant qui nous a donné la volonté, le courage ainsi que la patience afin de réaliser ce travail, et nous a aidé tout au long de notre cursus.*

*Nous exprimons notre gratitude et nos remercîments à notre encadreur **Mr, BOUDJEBBOUR Karim**, Enseignant chercheur à l'université de **Yahia Fares – Médéa**, ainsi qu'à notre Co-encadreur **Mr, HAYI MohamedYassine**, Doctorant à l'université de **Akli Mohand Oulhadj - Bouira**. Merci pour leur aide, leur encouragement, leurs conseils, ainsi que leurs intérêts qu'ils ont porté dans ce projet.*

Nous exprimons notre sincère gratitude aux membres de jury pour nous avoir fait l'honneur d'accepter d'évaluer notre modeste travail.

Nous adressons également notre sincère reconnaissance à toute personne qui nous a appris avec amour et sincérité tout au long de notre cursus du primaire à l'université.

*Nous remercions en particulier **Mr REGAI**, Enseignant à l'université Yahia Fares – Médéa. La plus sincère et la plus sympathique personne que nous avons l'honneur de rencontré.*

Nous présentons nos chaleureux remercîments à nos chers parents, nos frères et sœurs. Merci d'être toujours à coté de nous pour nous encourager. Merci à tous nos familles, amis et collègues pour leurs aides et leurs énormes soutiens.

Résumé

Le clustering, l'un des processus majeurs dans le domaine de la science de données. Plusieurs techniques ont été proposée pour réaliser ce processus où chacune a été proposée pour des cas ou des types de données spécifiques. Dans ce projet, nous allons proposer et implémenter une nouvelle technique de clustering hybride basée sur des techniques existantes afin de les adapter et généraliser leurs domaines d'application. Cette hybridation est basée essentiellement sur deux grandes techniques de clustering, La technique par densité et la technique hiérarchique agglomérative, avec l'utilisation d'autres paramètres afin de la rendre une approche efficace.

Mots clés : Science de données, clustering, méthode hiérarchique agglomérative, méthode par densité, algorithme, méthode hybride.

Sommaire

Résumé ----------3

Figures ----------7

Tableaux ----------9

Introduction générale

Introduction générale : ----------11

Chapitre 1: État de l'art

1. Introduction : ----------14

2. Définition : ----------14

3. Objectif : ----------16

4. Principe de fonctionnement : ----------16

5. Avantages de clustering : ----------17

6. Quelques notions préliminaires : ----------18

6.1. Type et échelle de données : ---------- 18

6.1.1. Les types : ----------18

6.1.2. Les échelles : ----------18

6.2. La matrice de données (Jeu de données) : ---------- 18

6.3. La matrice de proximité : ---------- 19

6.4. L'indice de ressemblance 'Similarité' : ---------- 20

6.5. La fonction de distance : ---------- 20

1- Données numériques : ----------21

2- Données binaires : ----------22

3- Données qualitatives : ----------23

4- Données textuelles : ----------24

5- Données mixtes : ----------24

6- Distance entre clusters : ----------25

7. Les méthodes de clustering : ----------26

7.1. Technique par partitionnement : ---------- **27**

7.1.1. Principe : ----------27

7.1.2. L'algorithme K-means : ----------28

7.1.3. L'algorithme k-medoïd :----------30

7.2. Technique hiérarchique :---------- **33**

7.2.1. Les avantages : ----------34

7.2.2. Les inconvénients----------34

7.2.3. L'approche divisive :----------34

7.2.3.1. L'algorithme DIANA :----------35

7.2.4. L'approche agglomérative : ----------36

7.2.3.1. L'algorithme Lance-Williams : ----------36

7.3. Technique par densité :---------- **37**

7.3.1. Principe----------38

7.3.2. L'algorithme de DBSCAN : ----------38

7.4. Technique par grille : ---------- **40**

7.4.1. Avantage----------41

7.4.2. Inconvénients ----------42

7.4.3. Principe----------42

8. Conclusion :----------*42*

Chapitre 2: Méthodologie

1. Introduction :----------*44*

2. Objectif : ----------*45*

3. Méthodologie et choix d'algorithmes de regroupement : ----------*45*

4. Méthode hybride proposée : ----------*50*

4.1. Préparation de données et calcul de la matrice de similarité : ---------- **51**

4.2. Algorithme de clustering hybride :---------- **52**

4.2.1. Principe de fonctionnement : ----------53

4.2.2. Choix d'indice d'agrégation :----------53

4.2.2.1 Principe de fonctionnement de la méthode des barycentres : ----54

4.2.3. Algorithme AggloDens : ----55

5. Conclusion :----*58*

Chapitre 3: Implémentation et évaluation

1. Introduction :----*60*

2. Outils utilisés :----*60*

2.2. Matériels :---- 60

2.2. Langage de programmation : ---- 61

2.3. Environnement de développement :---- 61

2.4. Bases de données utilisées : ---- 61

3. Implémentation de la méthode AggloDens : ----*64*

3.1. Calcul de la matrice de distance : ---- 64

3.2. L'algorithme AggloDens :---- 67

3.3. Évaluation des résultats :---- 70

4. Application de l'algorithme AggloDens :----*70*

5. Résultats obtenus :----*73*

6. Étude comparative :----*76*

6. Conclusion :----*80*

Conclusion générale et perspective

Conclusion générale et perspective : ----*82*

Bibliographies :----*83*

Figures

Figure 1.1 : Méthode de Clustering 15
Figure 1.2 : Les différentes étapes pour appliquer un clustering 17
Figure 1.3 : Single linkage 25
Figure 1.4 : Complete linkage 26
Figure 1.5 : Exemple de clustering par partitionnement 27
Figure 1.6 : Exemple du clustering par k-means 29
Figure 1.7: Exemple du clustering par k-medoïd 32
Figure 1.8 : Les deux méthodes de la technique hiérarchique 33
Figure 1.9 : Approche divisive 35
Figure 1.10 : Approche agglomérative 36
Figure 1.11 : Exemple d'un dendrogramme issu de la classification des données par CAH ... 37
Figure 1.12 : Formation du cluster avec DBSCAN 39
Figure 1.13 : Exemple d'un cluster dont les points sont densité-accessibles 40
Figure 1.14 : Grille bidimensionnelle pour la détection de clusters 41
Figure 2.15 : Exemple d'implémentation de l'algorithme de Lance Williams 46
Figure 2.16 : Exemple d'implémentation de l'algorithme de DBSCAN 47
Figure 2.17 : Schéma montrant le processus global de la méthode AggloDens 49
Figure 2.18 : Exemple d'un dendrogramme résultant de l'application de la méthode AggloDens 50
Figure 2.19 : Exemple d'un ensemble de clusters à une étape quelconque 54
Figure 2.20 : L'ensemble de clusters après un nouveau fusionnement 55
Figure 2.21 : Organigramme de l'algorithme AggloDens proposé 56
Figure 2.22 : Exemple du déroulement de la méthode AggloDens 57
Figure 2. 23 : Dendrogrammes résultants de l'implémentation d'un exemple avec l'algorithme de Lance Williams et AggloDens 58
Figure 3.24 : Un échantillon de la base de données « Mall customers » 62
Figure 3.25 : Un échantillon de la base de données « Country data » 63
Figure 3.26 : Un échantillon de la base de données « Utilisateurs de services web » 63
Figure 3.27 : Ensemble de fonctions pour le calcul de la similarité entre des données textuelles 65

Figure 3.28 : Résultats de l'application des fonctions pour le calcul de la similarité entre données textuelles 65
Figure 3.29 : Résultats de l'application des fonctions crées pour le calcul de la similarité entre données textuelles 66
Figure 3.30 : Échantillon de la matrice de proximité de la base de données « Mall customers » 67
Figure 3.31 : Le changement du nombre de clusters en fonction de (Eps, MinPts) 74
Figure 3.32 : Le changement de l'inertie intra-classe en fonction de (Eps, MinPts) 75
Figure 3.33 : Le changement de l'inertie intra-classe en fonction du nombre de clusters 76
Figure 3.34 : Comparaison des valeurs de l'inertie intra-classe résultantes des deux méthodes AggloDens et CHA 78

Tableaux

Tableau 1.1 : Matrice de données ---------- 19
Tableau 1.2 : Matrice de proximité ---------- 19
Tableau 1.3 : Matrice de contingence ---------- 22
Tableau 3.4 : Les valeurs utilisées dans l'application de l'algorithme AggloDens sur les bases de données ---------- 72
Tableau 3.5 : Comparaison des valeurs de l'inertie intra-classe résultantes des deux méthodes AggloDens et CHA ---------- 77
Tableau 3.6 : La taille des clusters résultants de l'application de l'algorithme AggloDens et Lance Williams sur l'ensemble « Country data » ---------- 79
Tableau 3.7 : Le nombre d'itérations et le temps d'exécution résultants de l'application de l'algorithme AggloDens et Lance Williams sur un ensemble de 200 instances. ---------- 80

Introduction générale

Introduction générale :

En Apprentissage Automatique (en anglais ***Machine Learning***), il existe deux grands types de classement de données, le classement supervisé et le classement non supervisé.

Dans le premier cas (dit ***classification*** en anglais), l'objectif est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets. Les méthodes s'étendent souvent à des variables Y quantitatives. On dispose au départ d'un échantillon dit d'apprentissage dont le classement est connu. Cet échantillon est utilisé pour l'apprentissage des règles de classement. Il est nécessaire d'étudier la fiabilité de ces règles pour les comparer et les appliquer, évaluer les cas de sous apprentissage ou de sur apprentissage (complexité du modèle). On utilise souvent un deuxième échantillon indépendant, dit de validation ou de test.

Autrement dit, La classification supervisée nécessite de fournir à l'algorithme une population déjà partitionnée. L'idée est de lui apprendre quelle est la loi qui régit la répartition afin qu'il puisse affecter un nouvel individu à sa classe d'appartenance.

Parmi les incontournables domaines d'application de ce type de classification, on cite notamment la biologie, champ d'études très vaste et bien connu de l'analyste, à travers par exemple l'établissement de diagnostics médicaux : catégorisation de la nature de tumeur associée au cancer du sein (maligne ou bénigne), stade de maladie d'un patient, etc.

Tandis que le classement non supervisé (en anglais dit ***Clustering***) est une approche importante en analyse exploratoire de données non étiquetées, mais reste un problème difficile. A partir d'une population, il s'agit d'extraire des classes ou groupes (***clusters***) d'individus présentant des caractéristiques communes [1].

Un ***cluster*** peut être défini comme un ensemble de données similaires entre elles et peu similaires avec les données appartenant à un autre cluster (homogénéité interne et séparation externe). Les ***clusters*** peuvent aussi être décrits comme des régions de l'espace de représentation des données contenant une densité relativement élevée de points de données, séparées entre elles par une zone de densité relativement faible. La détection de ces regroupements joue un rôle indispensable pour la compréhension de phénomènes variés décrits par un ensemble d'observations [2].

Ce type est utilisé dans divers domaines, on peut citer le domaine biomédical comme l'un des champs d'applications étendus de ces méthodes de classification, à travers, entre autres, le regroupement de gènes différentiels selon leur profil d'expression dans un phénomène biologique au cours du temps. Ces algorithmes peuvent aussi être utilisés dans une problématique de recommandation, afin de répartir différents articles en clusters et proposer à l'utilisateur un article qui lui convient.

Ces dernières années les volumes de données de toutes sortes croît de plus en plus. Avec tant de données disponibles, il est nécessaire de développer des algorithmes qui peuvent extraire des informations significatives à partir de ces vastes volumes. C'est pourquoi nous voulons proposer une nouvelle méthode de clustering hybride pouvant traiter tout type et taille de données. Afin de réaliser cet objectif, il nous a fallu en premier lieu de faire une recherche exploratoire des techniques et algorithmes de clustering existants et de mener une étude de synthèse analytique et comparative pour pouvoir extraire une approche hybride combinant entre des techniques de clustering existantes.

Ce travail s'articule sur trois chapitres, á savoir :

1- Clustering, ce chapitre est une étude sur le clustering est ses différentes techniques et algorithmes.

2- Conception de la méthode hybride, ce chapitre est consacré à l'approche hybride proposé. Il présente le principe de fonctionnement, la méthodologie et choix d'algorithmes.

3- Expérimentation et évaluation de la méthode, ce chapitre est consacré à l'implémentation de cette approche et son évaluation afin de la rendre une approche réelle.

Nous terminerons par une conclusion générale résumant le travail avec les résultats obtenus.

Chapitre I :

État de l'art

1. Introduction :

Le clustering, un processus d'analyse exploratoire qui divise les données en un ensemble de clusters, grâce à ces derniers, il est possible d'expliquer des caractéristiques de la distribution de données sous-jacente. Il s'agit d'une démarche très courante qui permet de mieux comprendre l'ensemble analysé. Le clustering est reconnu comme une tache de challenge, dont la difficulté est causée par un manque de définition unique et précise d'un cluster. Mais ce manque n'est pas un inconvénient, il a l'avantage de donner naissance à de multiples facettes du problème basées sur plusieurs définitions différentes d'un cluster.

Dans ce chapitre. Nous allons définir qu'est-ce que le clustering, l'objectif principale de clustering, les avantages et les inconvénients du clustering, et les différentes techniques de clustering avec quelques exemples d'algorithmes les plus connus de chacune.

2. Définition :

Clustering, en français segmentation ou regroupement de données, une technique d'apprentissage automatique non supervisé, on se place donc dans le cadre où les caractéristiques communes de ces groupes (appelés ***clusters***) au sein de l'échantillon ne sont pas connus á priori mais plutôt ils sont á découvrir automatiquement. Le pari est fait qu'au sein du jeu de données d'intérêt, des concentrations de données similaires vont se démarquer des autres amas de données par leur dissimilarité. Ainsi on va chercher à catégoriser une collection d'objets de telle manière que des observations appartenant à un même groupe soient plus similaires entre elles qu'avec tout autre élément appartenant à d'autres groupes, et dissimilaires quand ils appartiennent à des groupes différents. En d'autres termes le ***clustering*** vise à diviser les observations en différents groupes, homogènes au sein du groupe et hétérogènes entre groupes. On peut souvent lire que pour obtenir un bon partitionnement, il est nécessaire à la fois de minimiser l'inertie intra-classe (variance au sein d'un groupe) pour obtenir des ***clusters*** les plus homogènes possibles, et maximiser l'inertie inter-classe (variance entre les groupes) afin d'obtenir des groupes distincts. Un partitionnement idéal est constitué de groupes bien séparés et compacts [3].

Cette homogénéité est très souvent évaluée selon une distance ou plus largement une mesure de similarité dont le choix revient à l'expert et peut plus ou moins fortement influencer sur la qualité des partitions, en induisant un biais potentiel. Cette mesure de similarité peut se traduire tout

simplement par une distance euclidienne si les données s'y prêtent (repère orthonormé) ou se baser sur une mesure propre au problème et définie au préalable par l'utilisateur [3].

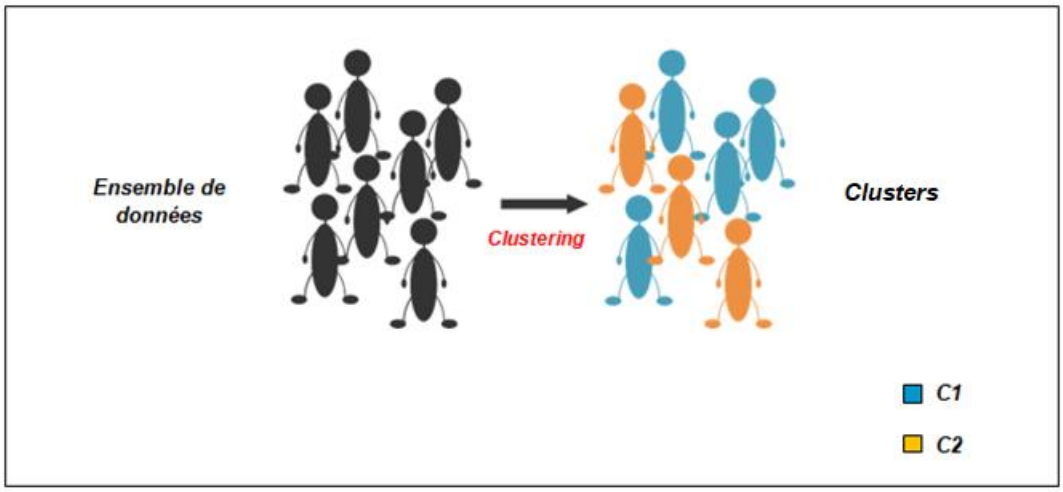


Figure 1 : Méthode de Clustering

La qualité du ***clustering*** dépend donc de la mesure de similarité utilisée par la méthode et de son implémentation, ainsi que le type de données á traiter.

Dans le premier cas selon qu'il y ait chevauchement entre les classes ou pas, on parlera de classification ***dure***, ***douce*** ou ***floue***. Dans le cas où des objets sont trop différents des autres, il s'agit de classification ***dure partielle***. Si la contrainte de dissimilarité est relâchée, un objet peut appartenir à différentes classes, on parle de classification ***douce***. Mais aussi de classification ***douce partielle*** s'il existe des objets qui n'appartiennent à aucune classe. Enfin, dans le cas où chaque objet appartient à toutes les classes avec un degré d'appartenance, on va vers la notion de classification ***floue***. Elle aboutit souvent vers une classification dure après une ***defuzzification*** des résultats par maximum d'appartenance par exemple, ou encore vers une classification douce ou les classes sont ordonnées par ordre décroissant du degré d'appartenance. Chaque objet est alors affecté aux ***clusters*** dont il vérifie les conditions [4].

Le clustering est :

- La méthode courante d'analyse des données statistiques utilisée dans de nombreux domaines.
- L'une des techniques les plus populaires en science de données « ***Data Analysis*** ».

- L'une des techniques de modélisation subjective largement utilisée dans de nombreux domaines, notamment l'apprentissage automatique, l'analyse d'images, la recherche d'informations, la bioinformatique, la compression de données et l'infographie.

3. Objectif :

L'objectif de cette méthode d'apprentissage débordant le cadre strictement exploratoire n'est pas de faire des prédictions à partir des valeurs d'entrée non encore vues mais principalement utilisée pour segmenter ou classifier une ***base de données*** afin d'être utilisée dans la classification supervisée, c'est-à-dire la recherche d'une typologie, ou segmentation en répartissant les données en classes (***clusters***) homogènes, ou catégories. Cet objectif est à distinguer des procédures de discrimination, ou encore de classement pour lesquelles une typologie est a priori connue pouvant également être utilisée dans la classification supervisée, au moins pour un échantillon d'apprentissage.

4. Principe de fonctionnement :

En générale, un algorithme de ***clustering*** contient les étapes suivantes :

- Le choix d'une mesure de proximité appropriée au domaine des données. Cette mesure doit assurer que tous les attributs des données contribuent également au calcul de la mesure de proximité et il n'y a aucun attribut qui domine d'autres.
- La définition d'un critère de ***clustering*** qui peut être exprimé par l'intermédiaire d'une fonction objective ou d'un autre type de règles.
- Une représentation simple et compacte de l'ensemble de données et l'ensemble des clusters.
- Le ***clustering*** : cette étape peut être réalisée de nombreuses façons, selon la technique adoptée.
- La validation des résultats : la qualité des résultats d'un algorithme de ***clustering*** est vérifiée en employant des mesures de validité de clusters. La qualité de chaque cluster doit être jugée non seulement par l'algorithme de ***clustering*** (la fonction objective) qui l'a produit, mais aussi selon un critère d'évaluation externe.

Le processus de ***clustering*** se divise en trois étapes majeures :

- La préparation des données ;

- L'algorithme de ***clustering*** ;
- Et l'exploitation des résultats de l'algorithme [5].

Nous discutons ici des problématiques générales liées à chacune de ces étapes.

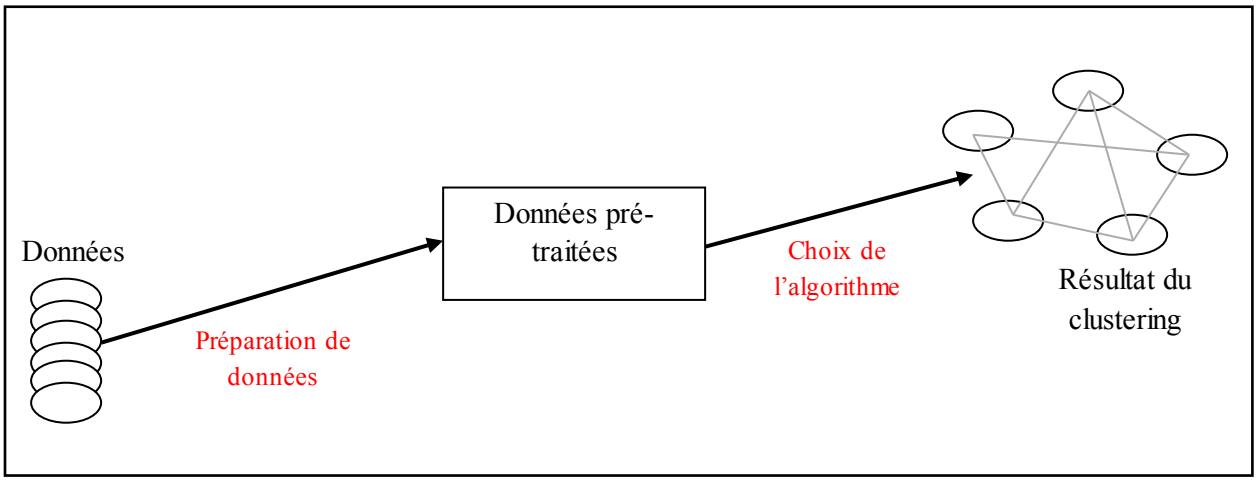


Figure 2 : Les différentes étapes pour appliquer un clustering

Les algorithmes de clustering peuvent être classifiés selon [6] :

- Le type des données en entrée.
- Les critères de ***clustering*** définissant la similarité entre les points de données.
- La théorie et les concepts fondamentaux sur lesquels les techniques de clustering sont basés (par exemple, la théorie floue, les statistiques). Ainsi, selon la méthode adoptée pour définir des clusters.

5. Avantages de clustering :

- Prendre en compte différents types de données : numériques, symboliques…
- Prendre en compte différentes dimensions de données (nombre d'attributs et d'instances).
- Générer des formes quelconques de clusters, pas seulement des formes convexes.
- Exploration des modèles de similitude à partir des ensembles de données non classés.
- Exploitation des résultats dans la classification supervisée (les résultats du clustering seront utilisés dans la classification supervisée pour classer des nouveaux individus dans l'ensemble de données).

7. Quelques notions préliminaires :

7.1. Type et échelle de données :

La mesure de similarité et le type de ***clustering*** utilisé dépendent des types et échelles de données á étudier [7].

7.1.1. Les types :

- ***Binaire :*** deux valeurs, vrai ou faux (1 ou 0).
- ***Discret :*** un nombre fini des valeurs ou des entiers.
- ***Continu :*** un nombre infini des valeurs ou les réels.

7.1.2. Les échelles :

- ***Quantitative :***

✓ ***Nominal :*** les valeurs sont juste des noms différents. Par exemple les codes postaux, les couleurs, le sexe.

✓ ***Ordinal :*** les valeurs reflètent un ordre, rien plus. Par exemple bon, meilleur, mieux ou couleurs ordonnées par le spectre.

- ***Qualitative :***

✓ ***Intervalle :*** la différence entre les valeurs est significative par exemple, l'intervalle de température

✓ ***Ratio :*** rapport entre deux grandeurs. Par exemple : les quantités monétaires, comme le salaire et le bénéfice et beaucoup de quantités physiques comme courant électrique, pression, etc.

7.2. La matrice de données (Jeu de données) :

Les objets (échantillons, mesures, modèles, événements) sont habituellement représentés comme des points (vecteurs) dans un espace multidimensionnel, où chaque dimension représente un attribut distinct (variable, mesure) décrivant l'objet. Ainsi, un ensemble d'objets est représenté comme une matrice $M \times N$, avec M lignes, une pour chaque objet et N colonnes, une pour chaque attribut. Cette matrice est appelée ***matrice de données*** (ou encore ***jeu de données***).

Points	x	y
P1	0	2
P2	2	0
P3	3	1
P4	5	1

Tableau 1 : Matrice de données

7.3. La matrice de proximité :

Plusieurs algorithmes de ***clustering*** utilisent la matrice de données originale et beaucoup d'autres emploient une matrice de similarité, ou une matrice de dissimilarité. Pour la convenance, les deux matrices sont généralement mentionnées comme une matrice de proximité P. Une matrice de proximité, P, est une matrice $M \times M$ contenant les valeurs de toutes les dissimilarités où les similarités entre les objets considérés. Si P_i et P_j sont le $i^{ème}$ et le $j^{ème}$ objets, respectivement, alors l'entrée à la $i^{ème}$ ligne et la $j^{ème}$ colonne de la matrice de proximité est la similarité, ou la dissimilarité, entre P_i et P_j.

	P1	P2	P3	P4
P1	0	2.8	3.2	5.1
P2	-	0	1.4	3.2
P3	-	-	0	2
P4	-	-	-	0

Tableau 2 : Matrice de proximité

7.4. L'indice de ressemblance 'Similarité' :

La similarité entre les données à étudier est l'information principale (et souvent unique) permettant à l'algorithme de classification de partitionner le jeu de donnée. Le plus souvent, cette similarité est calculée selon une mesure de "***distance***" adaptée au type de données et au problème à résoudre. Une distance, noté de manière générale ***d(i, j)***, est une mesure de similarité qui satisfait certaines propriétés mathématiques :

- $d(i, j) \geq 0$ … (***contrainte de positivité***) ;
- $d(i, j) = 0$ … si et seulement si $x_i = x_j$;
- $\forall x_i, x_j, d(i, j) = d(j, i)$ … (***contrainte de symétrie***) ;
- $\forall x_i, x_j, x_z, d(i, j) \leq d(i, z)+d(z, j)$ … (***contrainte d'inégalité triangulaire***).

7.5. La fonction de distance :

Il est souvent utile d'avoir une représentation géométrique afin de fournir une première vue des données dans le but de repérer des groupes d'objets naturels ou des objets isolés. Chaque attribut représente alors un axe de coordonnées. L'espace des données est donc un espace euclidien à ***M*** dimensions dans lequel chaque objet est représenté par un point.

Dans la littérature, il est aisé de trouver plusieurs définitions de métrique de distance. Cependant, toutes ces mesures ont une interprétation identique : il est très probable que des points proches dans l'espace de projection représentent des données d'un même groupe (la distance est alors proche de 0), alors que des points éloignés représentent, quant à eux, des données appartenant à des groupes différents (la distance tend vers l'infini) [8].

La formule de calcul de distance est utilisée d'après le type de données. Il existe plusieurs formules de calcul de distance pour chaque type, et les plus utilisées sont :

1- Données numériques :

A. La distance de Minkowski : avec ***q*** un entier positif non nul.

$$\mathbf{d}(i,j) = (\sum_{k=1}^{d} | \mathbf{i_k} - \mathbf{j_k} |^{q})^{1/q}$$

A.1. La distance euclidienne : est la plus connue et la plus utilisée. Elle peut être vue comme un cas particulier de la distance de ***Minkowski*** pour ***q = 2***.

$$\mathbf{d}(i,j) = \sqrt{\sum_{k=1}^{d} (\mathbf{i_k} - \mathbf{j_k})^2}$$

A.2. La distance de Manhattan : est également un cas particulier de la distance de ***Minkowski*** avec ***q = 1*** (également appelée ***Métrique Absolue***) :

$$\mathbf{d}(i,j) = \sum_{k=1}^{d} | \mathbf{i_k} - \mathbf{j_k} |$$

B. La distance de Sebestyen : pour donner un poids différent aux attributs :

$$\mathbf{d(i, j) = (i - j)^t\ W\ (i - j)}$$

Avec ***W*** matrice diagonale de pondération.

C. La distance de Mahalanobis : si les variables corrélées prennent trop d'importance, on peut normaliser la distance euclidienne par la covariance.

$$\mathbf{d(j, j) = (i - j)^t\ C^{-1}(i - j)}$$

Avec ***C*** matrice de variance-covariance.

2- Données binaires :

Pour les données binaires, il faut tout d'abord la construction de la matrice de contingence, qui contient quatre indices (***a, b, c*** et ***d***), telle que :

- ***a :*** nombre de caractéristiques communes aux deux objets (i=1, j=1);
- ***b :*** nombre de caractéristiques possédées par ***i*** et non par ***j*** (i=1, j=0);
- ***c :*** nombre de caractéristiques possédées par ***j*** et non par ***i*** (i=0, j=1);
- ***d :*** nombre de caractéristiques possédées ni par ***i*** ni par ***j*** (i=0, j=0);
- Et **p = a + b + c + d**.

		Objet j		
		1	0	sum
Objet i	1	***a***	***b***	***a + b***
	0	***c***	***d***	***c + d***
	sum	***a + c***	***b + d***	***p***

Tableau 3 : Matrice de contingence

Ensuite appliquer l'une des formules de calcul de la distance en utilisant cette matrice.

- ***Coefficient simple d'appariement*** (similarité invariante, si la variable binaire est symétrique) : $\mathbf{d(i,\ j) = \frac{b\ +\ c}{p}}$

- ***Jaccard*** (similarité non invariante, si la variable binaire est asymétrique) :

$$\mathbf{d(i,j) = \frac{b\ +\ c}{a\ +\ b\ +\ c}}$$

- ***Concordance :*** $\mathbf{d(i,j) = \frac{a+b}{p}}$

- ***Dice :*** $\mathbf{d(i,j) = \frac{2a}{2a+\ b\ +\ c}}$

- ***Autres distances souvent utilisées :***

  - $\mathbf{d(i,j) = \frac{a}{p}}$
  - $\mathbf{d(i,j) = \frac{a}{a+b+c}}$
  - $\mathbf{d(i,j) = \frac{a}{a+2\,(b+c)}}$

3- Données qualitatives :

Pour les données qualitatives, il y a plusieurs méthodes de normalisation afin de calculer la similarité. Voici en quelques :

- ***Correspondance simple :*** $\mathbf{d(i,j) = \frac{p - m}{p}}$ **.**

D'où :

- ***p :*** nombre total de variables.
- ***m :*** nombre de correspondances.

- ***Généralisation des variables binaires :*** pour chaque attribut (avec plus de 2 états), on crée une variable binaire pour chaque modalité. Par ex on a l'attribut '***couleur'*** comme suit :

Instance	***Couleur***
1	*Rouge*
2	*Bleu*
3	*Vert*
4	*Bleu*

v = 1

f = 0

Instance	***Bleu***	***Rouge***	***Vert***
1	f	v	f
2	v	f	f
3	f	f	v
4	v	f	f

Puis utiliser une des distances du cas binaire.

4- Données textuelles :

Pour calculer la similarité entre deux textes (ou chaine de caractères), tout d'abord il faut construire un vecteur global contenant tous les termes du 1er et 2eme texte.

Puis transformer chaque texte en vecteur binaire en comparant avec le vecteur global (tel que ***1*** indique la présence du terme dans le texte, et ***0*** indique l'absence du terme), ou en vecteur ***TF*** (***Natural Frequence***) contenant les fréquences des termes dans le texte.

Ensuite utilisé une formule convenable pour le calcul de la similarité, ***Jaccard*** avec le vecteur binaire, ***Euclidienne*** avec le vecteur TF, ou encore la formule de Cosinus avec les deux types de vecteur. Telle que : $\boldsymbol{Cosinus}(x, y) = \frac{\sum(x_i * y_i)}{\sqrt{\sum x_i^2 * \sum y_i^2}}$

5- Données mixtes :

Pour le cas de données mixtes (***Multi-types***), il faut utiliser la distance appropriée pour chaque groupe d'attribut puis utiliser une distance numérique (ex : la ***sommation***).

Prenant l'exemple suivant :

Personne	***Sexe***	***Age***	***Niveau scolaire***	***Salaire (DA)***
Assia	F	25	Licence	45000
Islem	H	29	PhD	63000
Karim	H	34	Magister	58000

Attribut numérique = Age, Salaire. Utiliser l'une des formules numériques (eg : Euclidienne).

Attribut qualitative (nominal) = Niveau scolaire. On peut le transformer en binaire comme suit :

Personne	***Licence***	***PhD***	***Magister***
Assia	1	0	0
Islem	0	1	0
Karim	0	0	1

Attribut binaire = Sexe, licence, PhD, Magister. Pour l'attribut Sexe on considère F = 1 et H= 0, puis utiliser l'une des formules binaires (***similaritéBinaire***).

Et à la fin sommer les valeurs obtenues avec chaque formule :

Similarité(Assoa, Islem) = euclidienne(Assia, Islem) + similaritéBinaire(Assia, Islem)

6- Distance entre clusters :

Il existe plusieurs méthodes pour le calcul de la distance entre clusters, citant quelques-unes :

- ***Saut minimum (Single Linkage) :*** La distance entre deux custers est représentée par la plus petite distance entre les éléments des deux classes.

$$D(c_1, c_2) = \min_{x_1 \in c_1,\ x_2 \in c_2} D(x_1, x_2)$$

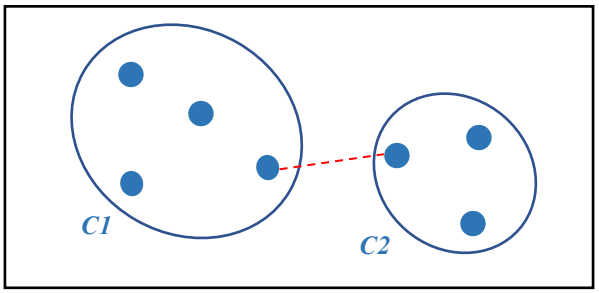


Figure 3 : Single linkage

- ***Saut maximum (Complete Linkage) :*** La distance entre deux custers est représentée par la plus grande distance entre les éléments des deux classes.

$$D(c_1, c_2) = \max_{x_1 \in c_1,\ x_2 \in c_2} D(x_1, x_2)$$

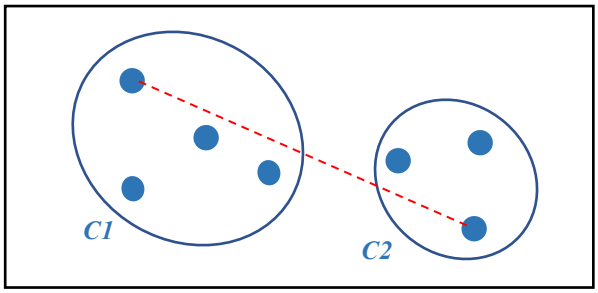


Figure 4 : Complete linkage

- ***Méthode de Ward :*** Supposant que C_1, C_2 et C_3 trois clusters avec N_1, N_2 et N_3 le nombre d'observations (instances) dans ces clusters respectivement, et $N = N_1 + N_2 + N_3$.

C_1 et C_2 sont les clusters fusionnés, la distance entre C_3 et l'union $(C_1 \cup C_2)$ selon la méthode de Ward est déterminée par la formule suivante :

$$d_{c_1 \cup c_2,\ c_3} = \alpha_1\, d_{C_1,C_3} + \alpha_2\, d_{C_2,C_3} + \beta\, d_{C_1,C_2}$$

D'où :

$$\alpha_1 = {}^{N_1 + N_3}/_{N} \quad \text{et} \quad \alpha_2 = {}^{N_2 + N_3}/_{N} \quad \text{et} \quad \beta = {}^{-N_3}/_{N}$$

8. Les méthodes de clustering :

Une multitude de techniques de ***clustering*** est développée. Les techniques se diffèrent dans leurs principes, propriétés, paramètres et formes générales du partitionnement généré. La catégorisation de ces techniques peut être réalisée selon plusieurs aspects : la mesure de proximité utilisée entre les données, la théorie ou les concepts fondamentaux sur lesquels se basent les techniques, la nature des données manipulées et beaucoup d'autres critères. En réalité, la catégorisation de ces techniques n'est ni directe, ni canonique car les catégories se chevauchent. Enfin en fonction de l'approche choisie pour la modélisation (paramétrique / non paramétrique) et le type de données étudiées, on se place dans une famille particulière d'algorithmes de ***clustering*** répartissant en quatre essentiaux types :

1- Technique par ***partitionnement***.

2- Technique ***hiérarchique***.

3- Technique par ***densité***.

4- Technique par ***grille***.

8.1. Technique par partitionnement :

Cette technique cherche à décomposer directement la base de données dans un ensemble de clusters disjoints. Plus spécifiquement, elle essaie de déterminer un nombre de partitions qui optimisent certains critères (***fonction objectif***). La fonction objectif peut mettre en valeur la structure locale ou globale des données et son optimisation est une procédure itérative [8].

Les techniques par partition créent un partitionnement des points de données, d'un seul niveau. Si ***k*** est le nombre désiré de clusters, alors les approches par partitionnement trouvent typiquement tous les ***k*** clusters immédiatement.

8.1.1. Principe :

- ***Initialisation :*** création d'une partition de ***k*** clusters.
- ***Itérations :*** déplacer un objet entre clusters pour optimiser la fonction objectif.

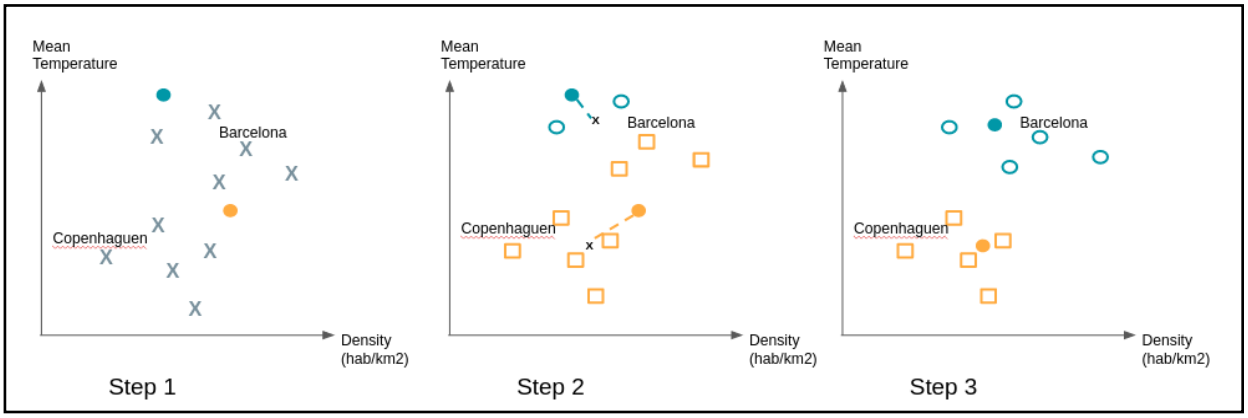


Figure 5 : Exemple de clustering par partitionnement

En fonction de la méthode de calcul des prototypes, les méthodes de partitionnement par optimisation itérative sont divisées en sous-catégories principales [9] :

- Les algorithmes basés sur les ***centroïdes*** (***K-means***).

- Les algorithmes basés sur les ***médoïdes*** (***K-medoïde***).

8.1.2. L'algorithme K-means :

Dans le cas de ***K-means***, un cluster est représenté par son centre, qui est une moyenne (d'habitude c'est une moyenne pondérée) des instances affectées au cluster [10].

Ce type d'algorithme ne fonctionne pas très bien pour les attributs catégoriels, par contre il est très adapté aux cas d'attributs numériques. D'une autre part, les centroïdes ont l'avantage d'avoir un sens géométrique et statistique assez clair.

L'algorithme ***K-means*** est l'outil de classification le plus utilisé dans les applications scientifiques et industrielles. La somme des dispersions (distances) entre un point et son centroïde, exprimée par une distance appropriée, est utilisée comme fonction objectif.

Par exemple, la fonction objectif basée sur la norme ***R2***, la somme des carrées des erreurs (en Anglais ***Sum Of Squared Errors*** ou ***SSE***) entre les points x_i et les centroïdes w_j correspondant, est égale à la variance intra-cluster totale [11] :

$$E(C) = \sum_{i=1}^{k} \sum_{x_i \in A} |x_i - w_j|^2$$

8.1.2.1. Principe :

1. Une classe est représentée par son centre de gravité.

2. Un objet appartient à la classe dont le centre de gravité lui est le plus proche.

8.1.2.2. Algorithme :

1. Choisir k individus au hasard (comme centre des classes initiales).

2. Affecter chaque individu au centre le plus proche.

3. Recalculer le centre de chacune de ces classes.

4. Répéter l'étape (2) et (3) jusqu'à stabilité des centres (aucun point ne change de cluster).

5. Éditer la partition obtenue [12].

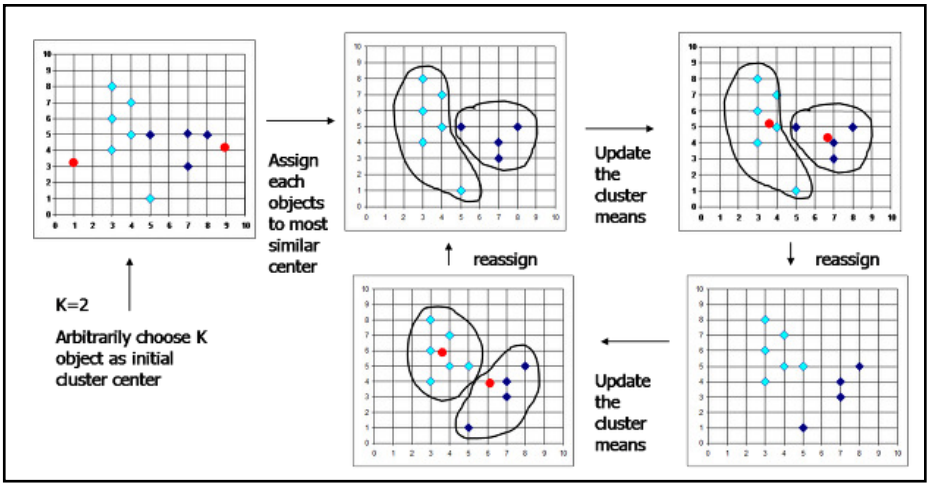


Figure 6 : Exemple du clustering par k-means

Le choix des centroïdes initiaux appropriés est l'étape clef de l'algorithme de ***K-means***. Il est facile de choisir les centroïdes initiaux aléatoirement mais l'inconvénient est que la partition finale dépend de la partition initiale.

8.1.2.3. Avantages :

- Le grand avantage de l'algorithme de K-means est sa complexité temporelle, qui le rend efficace dans le traitement de grands jeux de données.
- Simple à implémenter et à comprendre [12].

8.1.2.4. Inconvénients :

- Le résultat dépend fortement de la conjecture initiale de centroïdes.
- Le processus est sensible aux bruits et aux anomalies.
- Seulement des attributs numériques sont couverts.
- Il tend à trouver des clusters sphériques de tailles égales.
- Nécessite de spécifier le nombre de classes à l'avance.

- Utilisable seulement quand la notion de moyenne existe (donc pas sur des données nominales) [12].

Plusieurs variantes de l'algorithme de ***K-means*** existent dans la littérature. Certains d'entre eux essayent de choisir une bonne partition initiale pour que l'algorithme trouve la valeur du minimum global. Une autre variante consiste à fractionner ou fusionner les clusters résultants. Typiquement un cluster est fractionné quand sa variance est au-dessus d'un seuil pré spécifié et deux clusters sont fusionnés quand la distance entre leur centroïdes est au-dessous d'un autre seuil pré spécifié [13].

L'algorithme le plus connu qui utilise cette technique de fusion et de fractionnement de clusters c'est ***ISODATA***, qui a été est utilisé particulièrement dans le traitement d'image [14].

K-Means est largement utilisé pour de nombreuses applications :

- Segmentation d'image.
- Regroupement des données de segmentation des gènes.
- Regroupement d'articles.
- Clustering de langues.
- Regroupement des espèces.
- Detection d'une anomalie.

8.1.3. L'algorithme k-medoïd :

Dans l'approche ***K-médoïde***, un cluster est représenté par un de ses points. Ce point représentatif est appelé ***médoïde***, ce dernier est un point qui existe réellement et qui correspond à l'observation qui minimise sa distance avec l'ensemble des autres points. D'autre manière, c'est le point le plus placé au centre en tenant en compte quelques mesures, comme par exemple, la distance.

$$M = \min_{e} \sum_{i=1}^{n} d(i, e)$$

Avec ***e = 1... n***, tout élément dans le cluster peut être ***médoïde***.

Le ***médoïde*** est l'élément dont la variance est minimale telle que :

$$E(C) = \sum_{j=1}^{k} \sum_{x_i \in A} |x_i - M_j|^2$$

Avec : - x_i est un élément du cluster j (avec $i = 1... N_j$, d'où N_j le nombre d'élément dans le cluster j).

- M_j le ***médoïde*** du cluster **j** [15].

8.1.3.1. Principe :

Il a le même principe avec le k-means sauf que :

1. Chaque cluster est représenté par un des objets du cluster.
2. Quand les medoïdes sont sélectionnés, les classes sont définies comme des sous-ensembles de points proches des medoïdes respectifs, et la fonction objectif est définie comme la distance moyenne ou une autre mesure de dissimilarité entre un point et son medoïde.

8.1.3.2. Algorithme :

1. Choisir k points (médoïdes) initiaux.
2. Affecter chaque individu à la classe dont le médoïde est le plus proche.
3. Recalculer les médoïdes des classes à partir des individus rattachés.
4. Répétée l'étape (2) et (3) jusqu'á atteindre le nombre d'itérations fixé, ou lorsque les médoïdes M_k sont stables (convergence).
5. Éditer la partition obtenue [12].

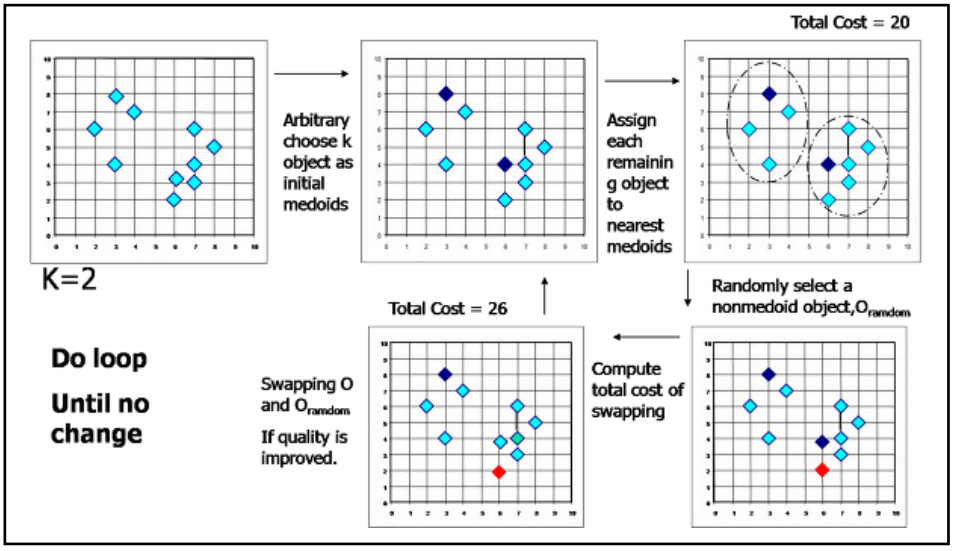


Figure 7: Exemple du clustering par k-medoïd

Cette approche n'est pas limitée aux espaces euclidiens. En outre, l'utilisation de médoïdes pour définir des clusters rend cette méthode résistante contre les bruits dans les données [16].

La dégradation est faite dans les étapes 2 et 3 de l'algorithme, puisque la découverte d'un meilleur médoïde exige l'essai de tous les points qui ne sont pas médoïdes.

8.1.3.3. Avantages :

- Pas de limites par rapport au type d'attributs.
- Le choix du medoïde est dicté par la location d'une fraction des points prédominants dans le cluster et ainsi est moins sensible à la présence des données aberrantes.

8.1.3.4. Inconvénients :

- Très coûteux en temps de calcul.
- La complexité de calcul est particulièrement dissuasive.

8.2. Technique hiérarchique :

Le résultat de ce type d'algorithmes est un arbre de clusters, appelé ***dendrogramme***, qui montre comment les clusters sont organisés. En coupant le ***dendrogramme*** au niveau désiré, une classification des données dans des groupes disjoints est ainsi obtenue. Une telle approche permet d'explorer les données à travers différents niveaux de granularité. Les méthodes de la classification hiérarchique sont divisées en deux types d'approches [17] :

1. Ascendante (***CHA***), dite ***agglomérative***. La plus couramment utilisée consiste à construire la hiérarchie à partir des objets (au départ on a un objet par classe), puis à fusionner les classes les plus proches, afin de n'en obtenir plus qu'une seule contenant tous les objets.

2. Descendante (***CHD***), dite ***divisive***. Moins utilisée, consiste à construire la hiérarchie à partir d'une seule classe regroupant tous les objets, puis à partager celle-ci en deux groupes. Cette opération est répétée à chaque itération jusqu'à ce que toutes les classes soient réduites à des singletons. Le processus continu jusqu'à ce qu'un critère d'arrêt (d'habitude c'est le nombre k de clusters demandé) soit atteint.

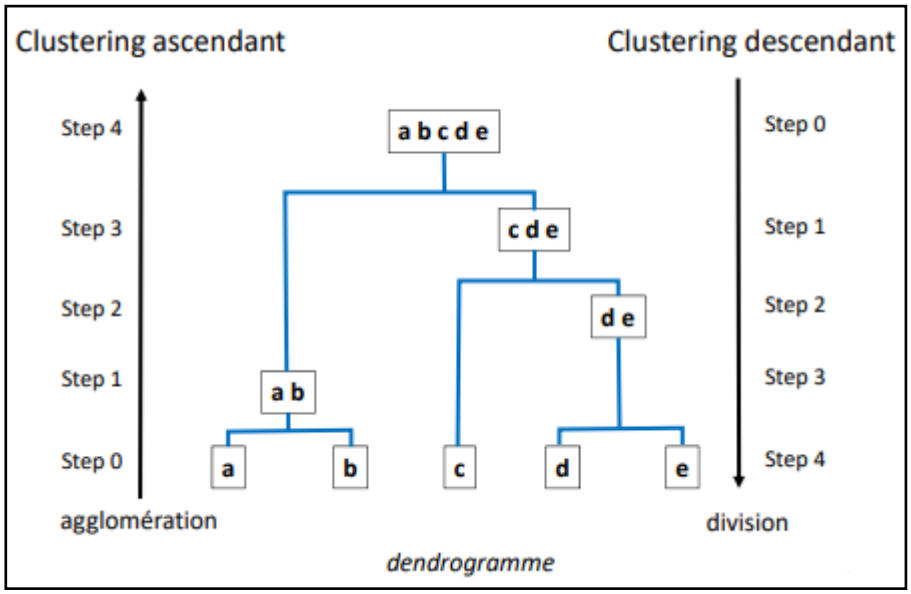


Figure 8 : Les deux méthodes de la technique hiérarchique

8.2.1. Les avantages :

- Facilité pour traiter différentes formes de similarité ou de distance entre objets.
- Applicable aux différents types d'attributs.
- Une flexibilité en ce qui concerne le niveau de granularité.
- Facile à implémenter.
- Fournit une structure (préférable pour une analyse détaillée qu'une méthode de partitionnement).

8.2.2. Les inconvénients :

- Choix du critère d'arrêt qui reste vague.
- Fait que la plupart des algorithmes hiérarchiques ne revoient pas les clusters intermédiaires qu'une fois qu'ils sont construits pour les améliorer.
- Les méthodes de classification hiérarchique basées sur des métriques de liens donnent comme résultats des clusters de formes convexes.
- La matrice de distance N × N, qui calcule les similarités ou dissimilarités entre les données est trop grande. Pour surmonter cette limite, plusieurs heuristiques sont possibles pour réduire la taille de cette matrice. Cela peut être réalisé en omettant des entrées inférieures à un certain seuil, ou en utilisant seulement un certain sous-ensemble de représentants des données, ou en gardant pour chaque individu qu'un certain nombre des plus proches voisins [18].

8.2.3. L'approche divisive :

L'approche divisive la plus simple et la plus utilisée est donné par les étapes suivantes [19] :

1. Calculer l'***arbre de couverture minimum*** (***MST*** ou ***Minimum Spanning Tree***) du graphe de proximité.
2. Créer un nouveau cluster en enlevant le lien correspondant à la plus grande distance.
3. Répéter l'étape (2) jusqu'à ce que seulement les clusters de ***singleton*** restent.

8.2.3.1. L'algorithme DIANA :

DIANA (***Divisive ANAlysis***), un algorithme hiérarchique qui procède par division. Son fonctionnement repose sur un principe simple permettant de grouper les éléments d'un ensemble selon leurs similitudes.

8.2.3.1.1. Principe :

1. Au départ, tous les objets appartiennent à un seul et unique groupe.
2. Un critère de partitionnement est, ensuite, utilisé pour diviser le groupe en deux sous-groupes et ensuite pour diviser chaque sous-groupe récursivement jusqu'à ce que chaque groupe contienne un seul élément.

8.2.3.1.2. Algorithme :

1. Initialisation á 1 cluster.
2. Sélection du cluster avec diamètre maximum.
3. Identifier dans le cluster l'objet ***x*** ayant la plus faible similarité moyenne avec les autres objets.
4. Initialisation d'un nouveau cluster avec ***x***.
5. Ajouter au nouveau cluster l'objet dont la valeur maximale de dissimilarité est positive. Répéter jusqu'á valeur maximale de dissimilarité < 0.
6. Répéter 2 á 5 jusqu'á obtenir des clusters singleton seulement [20].

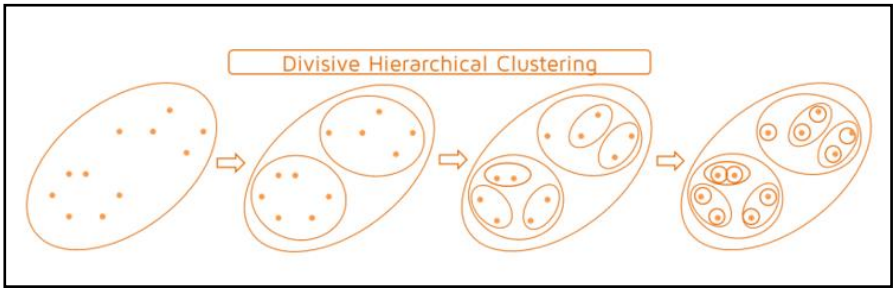


Figure 9 : Approche divisive

8.2.4. L'approche agglomérative :

Cette méthode est l'inverse de la méthode divisive, on la résume par les étapes suivantes :

1. Calculer la matrice de proximité.
2. Au début, chaque instance est un cluster.
3. Regrouper les deux éléments les plus proches (c-à-d. les deux éléments possédants la distance la plus petite dans la matrice de proximité).
4. Mettre à jour la matrice de proximité en remplaçant les deux éléments regroupés par le nouveau cluster.
5. Répéter (3) – (4) jusqu' à obtenir un seul cluster contenant toutes les instances.

8.2.3.1. L'algorithme Lance-Williams :

L'algorithme de Lance-Williams est un algorithme de clustering hiérarchique agglomératif. C'est l'un des meilleurs algorithmes pouvant traiter de gros jeux de données

8.2.3.1.1. Principe :

1. Calculer la ***matrice de proximité***.
2. Fusionner les deux clusters les plus proches (les plus similaires).
3. Mettre à jour la matrice de proximité pour refléter la proximité entre le nouveau cluster et les clusters originaux.
4. Répéter les étapes (3) et (4) jusqu'à ce que seulement ***un seul cluster*** reste [21].

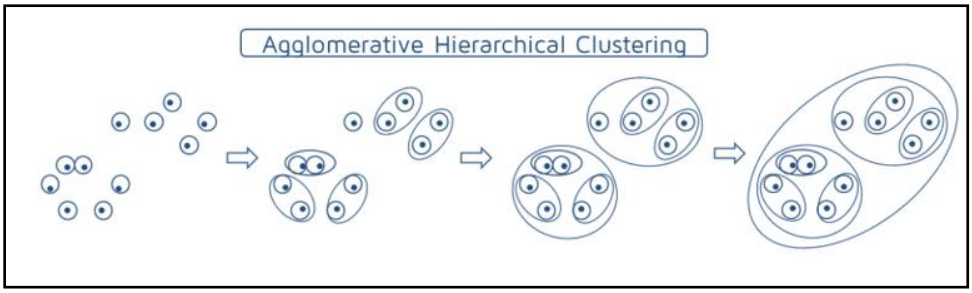


Figure 10 : Approche agglomérative

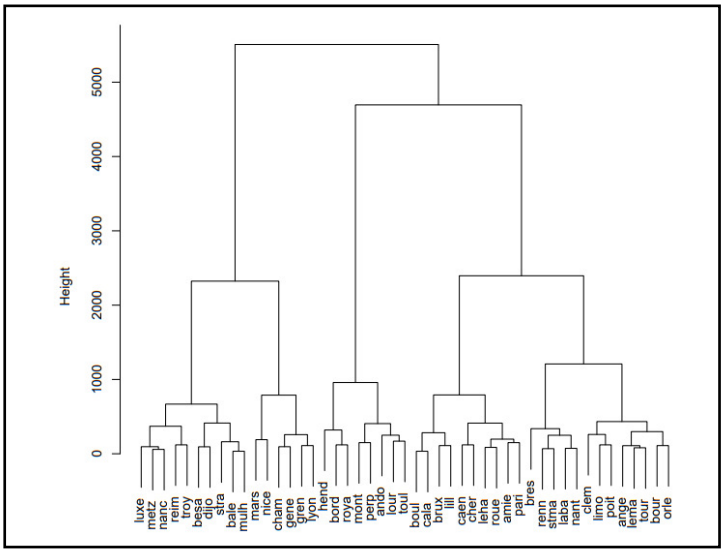


Figure 11 : Exemple d'un dendrogramme issu de la classification des données par CAH

8.3. Technique par densité :

L'idée principale de ce type de classification est de grouper des objets voisins de l'ensemble des données en classes suivant une ***fonction de densité***.

Un ensemble ouvert dans l'espace euclidien peut être divisé en un ensemble de ses composantes connectées. L'implémentation de cette idée pour le partitionnement d'un ensemble fini de points à besoin de tels concepts comme la ***densité***, la ***connectivité*** et les ***limites*** (***extrémité***, ***bordure***, ***frontière***) plutôt que la distance. Ils sont fortement reliés aux plus proches voisins d'un point. Un cluster, défini comme une composante densément connectée, évolue dans n'importe quelle direction où la densité le mène. Par conséquent, les algorithmes à base de densité sont capables de découvrir des clusters de formes arbitraires. Ainsi, ce principe fournit une protection naturelle contre les données aberrantes. Les clusters ne sont pas nécessairement convexes (entièrement contenus/remplis). Le bruit et les données isolées sont identifiés. Le nombre de classes n'a pas besoin d'être prédéfini préalablement. Un seul parcours de la base de données est effectué pour la réalisation de ces clusters.

Les algorithmes basés sur la densité considèrent les clusters comme des régions de points denses dans l'espace de données qui sont séparées par des régions de faible densité [22].

8.3.1. Principe :

1. Un point est voisin d'un autre point s'il est à une distance inférieure à une valeur fixée (ε), cette valeur est à choisir selon l'échelle de l'étude et l'exploitation de la solution recherchée.

2. Un point est dense si le nombre de ses voisins dépasse un certain seuil (***MinPts***). Ce critère est aussi défini par l'utilisateur selon les objectifs de son étude.

Il existe deux types d'approches à base de densité :

- Le premier type d'approche attache la densité a un point qui appartient aux données d'apprentissage. Il compte des algorithmes spécifiques comme ***DBSCAN***, ***GDBSCAN***, ***OPTICS*** et ***DBCLASD***.
- Le deuxième type attache la densité à un point dans l'espace d'attributs. Il inclut l'algorithme ***DENCLUE*** [23].

Des concepts majeurs de cette section sont la ***densité*** et la ***connectivité***, les deux mesurées en termes de distribution locale des plus proches voisins.

Un cluster basé sur la densité est maintenant défini comme un ensemble de points ***densité-connectés***.

8.3.2. L'algorithme de DBSCAN :

L'un des algorithmes les plus connus de cette catégorie. Avec cet algorithme, la découverte d'un groupe se fait en deux étapes :

1. Tout d'abord, un point dense est choisi aléatoirement,
2. Puis tous les points qui sont accessibles à partir de ce point, selon le critère de densité, forment le groupe.

Deux paramètres indiqués par l'utilisateur entrent en jeu :

1. ***Epsilon*** (ε), le rayon de voisinage,
2. Et ***MinPts***, le seuil de densité qui correspond au nombre minimal d'objets dans le voisinage d'un point.

8.3.2.1. Principe :

1. Choisir un point arbitraire ***x***.
2. Trouver tous les points qui sont ***densité-accessibles*** de ***x*** (en réalisant ces deux conditions : il faut qu'il y ait au moins un certain nombre de points, déterminé par ***MinPts***, à une distance de voisinage inférieure à ***ε***. Si ce point ne réalise pas ces conditions, il est considéré comme ***bruit***). Si ***x*** est un point ***noyau*** alors nous avons formé un cluster. Si ***x*** est un point ***bordure*** alors aucun point n'est densité-accessible de x.
3. Répétition pour le prochain point dans les données [24].

8.3.2.2. Algorithme :

1. Entrer ***eps*** et ***MinPts***.
2. Prendre un point ***x*** qui n'a pas été visité.
3. Construire ***N*** = voisinage(E***ps***, ***x***).
4. Si |***N***| < ***MinPts*** Marquer ***x*** comme bruit, sinon initaliser **C** = {**x**} et agrandir_cluster(***C***, ***N***).
5. Ajouter ***C*** à la liste des clusters.
6. Marquer tous les points de ***C*** comme visités.
7. Répéter 2 á 6 jusqu'á tous les points ont été visités [24].

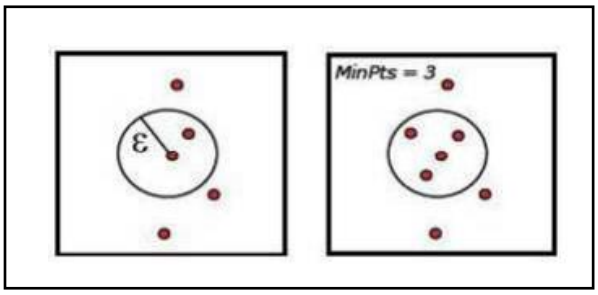


Figure 12 : Formation du cluster avec DBSCAN

Le cluster formé contient alors l'ensemble des points à l'intérieur du voisinage du noyau. Il opère de la même manière pour tous les autres points appartenant au cluster, augmentant ainsi le

nombre de points du cluster (voir image ci-dessous). On dit que tous les points de ce cluster sont densité-accessibles. Il s'arrête lorsque les points ne respectent plus les critères ε ou ***MinPts*** [24].

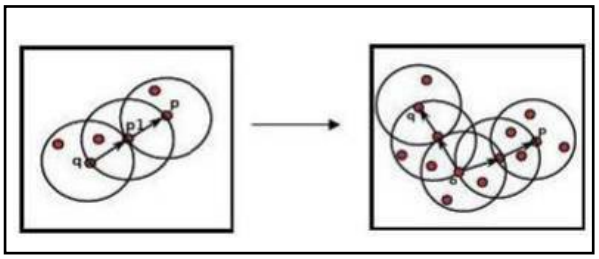


Figure 13 : Exemple d'un cluster dont les points sont densité-accessibles

8.3.2.3. Avantages :

- ***DBSCAN*** a le grand avantage d'être efficace en temps de calcul sans requérir de prédéfinir le nombre de clusters.
- Il permet de trouver des clusters de forme arbitraire.

8.3.2.4. Inconvénients :

- L'inconvénient majeur est que les clusters doivent avoir un minimum de points ***MinPts***. Ceci rend impossible pour ***DBSCAN*** de trouver les très petits clusters dans de grands jeux de données.
- L'algorithme ***DBSCAN*** est difficile à utiliser en très grand jeu de données.
- Le choix des paramètres ε et ***MinPts*** peut aussi être délicat.

8.4. Technique par grille :

Ces algorithmes commencent par partitionner l'espace de données à un nombre fini de cellules et accomplir ensuite des opérations exigées sur l'espace partitionné.

Les cellules qui contiennent plus qu'un certain nombre de points sont traitées comme denses et les cellules denses sont connectées pour former les clusters [25].

L'espace de données multidimensionnel est divisé en cellules, et le calcul de la densité des régions (taille du voisinage) n'est plus centré sur les instances mais effectué pour chaque cellule. Chaque variable est discrétisée en largeur (intervalles égaux), telle que la largeur des intervalles de discrétisation détermine la taille et le nombre de cellules. Les données bruitées et les valeurs aberrantes sont implicitement ignorées car elles sont contenues dans des cellules faiblement peuplées.

Ce type de méthodes est conçu pour des données spatiales. En fait, avec une telle représentation des données, au lieu de faire la classification dans l'espace de données, on le fait dans l'espace spatial en utilisant des informations statistiques des points dans la cellule. Les algorithmes les plus connus sont ***STING***, ***CLIQUE*** et ***WaveCluster***.

résultent de l'espace de partitionnement, alors que l'espace de partitionnement est basé sur les caractéristiques de la grille extraites des données d'entrée [26].

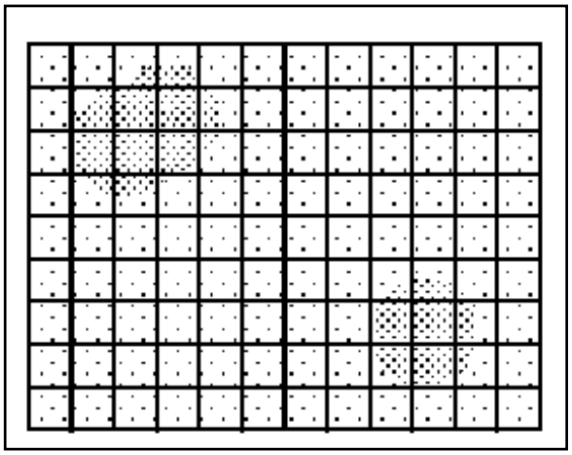


Figure 14 : Grille bidimensionnelle pour la détection de clusters

8.4.1. Avantage :

- L'accumulation des données sous forme de grille rend les techniques à base de grille indépendante par rapport à l'ordre (rangement) des données.
- Fonctionnent avec des variables de différents types.
- Dans une certaine mesure, les méthodes à base de grille reflètent un point de vue technique. Cette famille d'approches est éclectique : elle contient les deux types d'algorithmes : hiérarchiques et de partitionnement [27].

8.4.2. Inconvénients :

- Le principal inconvénient de ces méthodes est de choisir convenablement les paramètres liés à la taille des cellules. En effet, des cellules de trop grande taille reprendront beaucoup d'objets qui ne sont pas forcément bien uniformément répartis. On pourra donc avoir des clusters peu homogènes, nécessitant d'être subdivisés par la suite. Il y a donc sous partitionnement. Par contre, des cellules de petite taille seront généralement très denses à partir du moment où elles contiennent des objets. On aura alors tendance à détecter des frontières qui n'auraient pas lieu d'être et à avoir un sur-partitionnement [27].

8.4.3. Principe :

1. Diviser l'espace sur lequel les données s'étendent en cellules rectangulaires, par exemple, on divise l'intervalle de valeurs de chaque dimension en cellules de taille égales.
2. Rejeter les cellules de grille à densité basse. Ceci assume une définition de cluster basée sur la densité, c'est-à-dire, que les régions à haute densité représentent des clusters, alors que les régions à basse densité représentent le bruit. Cette supposition est souvent bonne, bien que les approches basées sur la densité puissent avoir des difficultés quand il y a des clusters de différentes densités
3. Combiner des cellules adjacentes à haute densité pour former des clusters [27].

9. Conclusion :

Dans ce chapitre, nous avons définir le clustering comme méthode d'analyse non supervisée utilisée pour organiser des données brutes en silos homogènes. A l'intérieur de chaque cluster, les données sont regroupées selon une caractéristique commune. L'outil d'ordonnancement est un algorithme qui mesure la proximité entre chaque élément à partir de critères définis (similarité). Puis nous avons expliqué comment préparer les données afin qu'on puisse explorer et analyser des modèles de similitude. Ensuite nous avons détaillé les différentes techniques de clustering avec leurs principes et quelques exemples d'algorithme

Chapitre II :

Méthodologie

1. Introduction :

Les algorithmes de clustering sont utilisés par les chercheurs de divers domaines pour explorer et analyser les modèles de similitude dans leurs données, puis appliquer ces modèles dans différents domaines, tel que la classification supervisée, le traitement d'image, le web mining, etc.

Malgré la diversité de ces techniques de clustering et le grand nombre de leurs algorithmes, toutes ces méthodes ont des limitations, parce que le problème de clustering est si gros. C'est pourquoi des analystes ont pensé à trouver des nouvelles solutions et ils ont proposé quelques approches hybrides.

Une ***approche hybride*** (dit aussi ***clustering hybride***) est une méthode combinant entre deux ou plusieurs techniques de clustering. Plusieurs méthodes hybrides ont été proposées, telle que chacune et son cas d'utilisation, et elle se base sur des catégories précises. L'objectif des hybridations se diffère d'une approche à une autre, mais le global est l'amélioration des résultats de clustering et l'adaptation des méthodes de base combinés au cas étudié.

De notre part, notre travail consiste á proposer une nouvelle approche hybride en se basant sur deux grandes techniques de clustering :

- La technique ***hiérarchique agglomérative*** ;
- Et la technique par ***densité***.

La technique hiérarchique agglomérative, comme décrit auparavant, c'est une technique ascendante qui tient comme paramètre d'entrée un groupe de données, ensuite elle agrège les données les plus similaires et refait le processus jusqu'à l'obtention d'un singleton.

Tandis que, la technique par densité nécessite au démarrage deux paramètres comme condition, ***Eps*** qui détermine le rayon de voisinage, et ***MinPts*** qui détermine le nombre des éléments d'un cluster au minimum. Puis former des clusters en respectant ces deux conditions.

Dans ce chapitre, nous allons présenter notre méthode proposée, notre objectif, les méthodes de clustering qu'elle combine et comment les combiner et l'enchaînement de l'algorithme.

2. Objectif :

La technique ***AggloDens*** proposée vise à :

- Améliorer la qualité de clustering et minimiser l'inertie intra-classe en se basant sur la méthode des barycentres.

- Définir la distance entre les éléments du même cluster (***Eps***) et même le nombre de ces éléments (***MinPts***).

- Optimiser le temps d'exécution surtout avec les données larges, au lieu de former un cluster par un soit pour le ***DBSCAN*** ou ***Lance Williams***, former plusieurs clusters á la fois á chaque agglomération, donc minimiser le nombre d'itération.

- Utiliser avec n'importe type et taille de données contrairement à d'autre méthode qui reste spécifiques aux données numériques, binaires, etc.

- Déterminer le meilleur nombre de clusters qui minimisent le critère de performance d'une méthode de clustering.

- Pas besoin de définir le nombre de clusters, ce qui veut dire éviter le paramètre le plus délicat dans le clustering.

- Avoir une segmentation équilibrée.

3. Méthodologie et choix d'algorithmes de regroupement :

La nouvelle méthode que nous allons proposer ici se base essentiellement sur deux grandes techniques de clustering, la première est la technique hiérarchique ascendante (CHA), et la deuxième est la technique par densité, en combinant deux algorithmes de ces catégories, l'algorithme de ***Lance Williams et*** le ***DBSCAN***.

L'algorithme de ***Lance Williams***, une approche hiérarchique ascendante où chaque objet commence comme son propre cluster et à chaque étape (agglomération), les clusters sont fusionnés en fonction d'une mesure de distance (similarité) entre leurs membres les plus proches, ce processus s'exécute jusqu'à arriver à un singleton (un seul cluster).

En appliquant l'algorithme de Lance Williams :

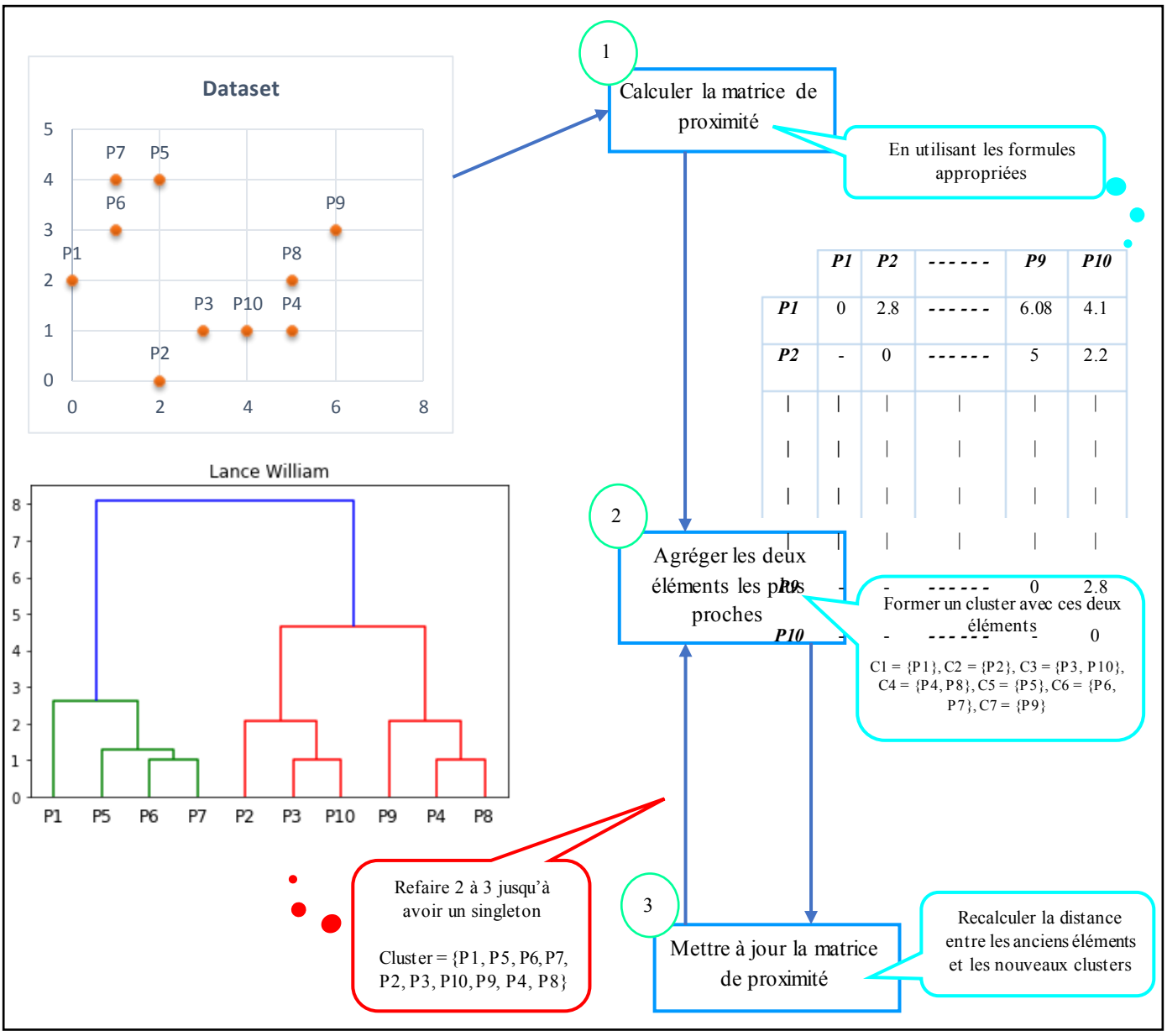


Figure 15 : Exemple d'implémentation de l'algorithme de Lance Williams

Tandis que l'algorithme de DBSCAN prend en entrée deux paramètre à définir, ces deux paramètres sont un peu délicats, car il est difficile de trouver les bonnes valeurs dès le premier essaye. Le premier paramètre est utilisé pour définir la distance minimale entre le noyau et les points densité-accessibles de celui-ci. Le deuxième paramètre est utilisé pour définir le nombre minimal des éléments d'un cluster.

En appliquant le DBSCAN :

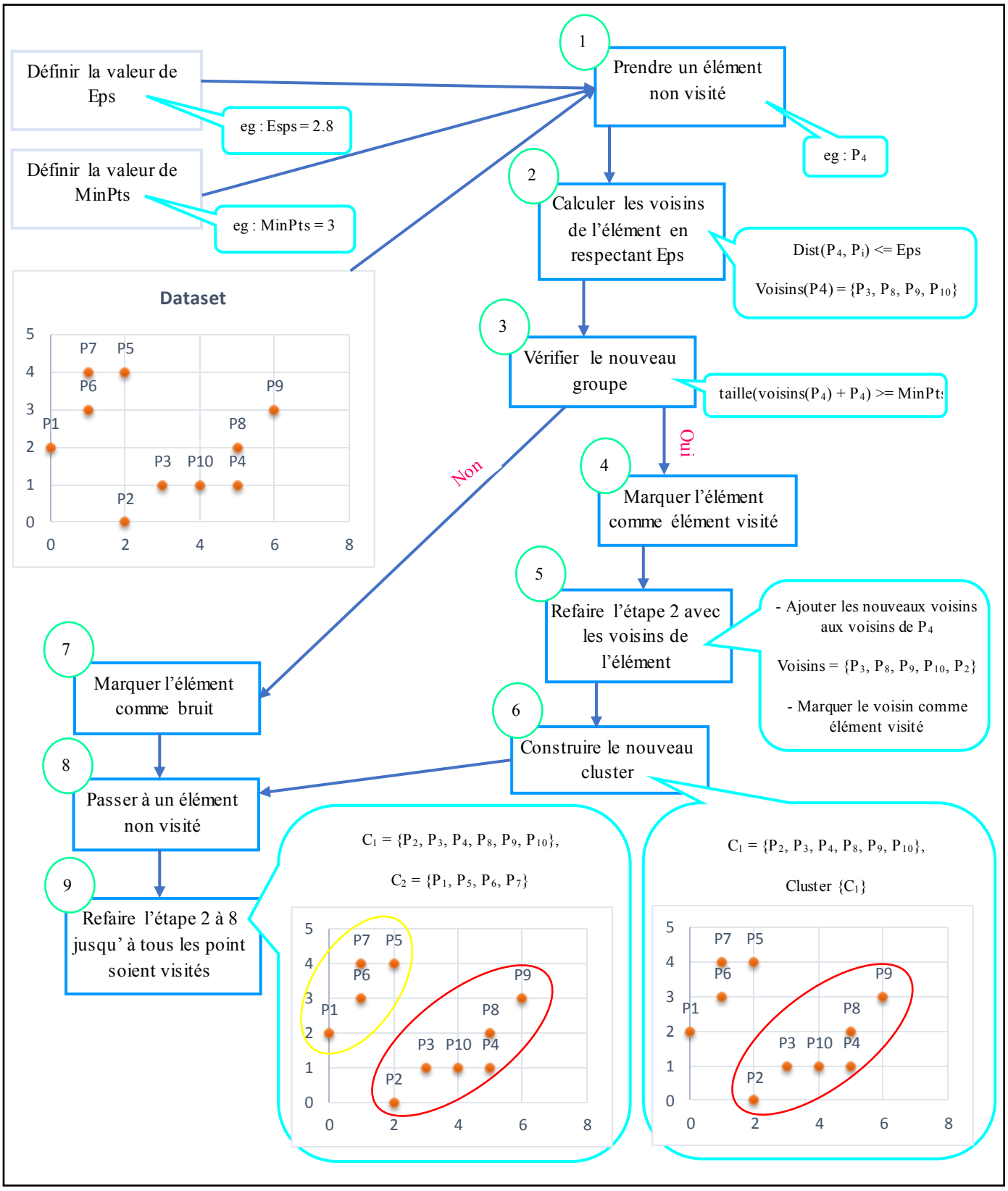


Figure 16 : Exemple d'implémentation de l'algorithme de DBSCAN

D'après ces deux algorithmes, l'idée globale de notre approche hybride (que nous appelons ***AggloDens***) est d'intégrer les paramètres du clustering par densité à la méthode hiérarchique agglomérative, tel que :

1- Initialiser chaque élément comme un cluster.

2- À chaque agglomération, fusionner tous les clusters dont la distance entre eux est inférieure à ***Eps***, au lieu de fusionner seulement les clusters les plus proches. Cette contrainte accélère le processus, mieux former plusieurs grappes en même temps que d'en former un seul.

3- Mettre à jour la matrice de proximité.

4- Les clusters doivent avoir un ***MinPts*** éléments au minimum. Cette contrainte est considérée comme ***condition d'arrêt***.

Nous avons ajouté une nouvelle contrainte sous forme d'un pourcentage ***P*** où l'algorithme s'arrête lorsque le nombre des clusters possédants au moins MinPts éléments atteint ***P%*** du nombre global des clusters actuels. Cette contrainte est utilisée en cas d'avoir quelques clusters (un nombre très petit par rapport au nombre global des clusters actuels) de taille inférieure à ***MinPts*** même si le processus continue à s'exécuter ils ne seront plus fusionnés avec d'autre. Ou encore en cas d'un ensemble de données très larges pour diminuer le temps d'exécution.

Prenant un exemple, supposant que ***MinPts*** = 15. A une étape quelconque nous avons 12 clusters de taille (10, 23, 31, 16, 21, 29, 38 ,50 ,41, 17, 25, 35). Nous remarquons ici qu'il existe un cluster (c'est le premier cluster) de taille inférieure à 15. Même si nous continuons l'exécution du processus ce cluster ne sera plus fusionné avec les autres. Donc pour arrêter le processus nous utilisons le pourcentage ***P*** (avec eg : ***P = 90%***) tel que ***12 * 90% = 10.8 ≈ 11*** où :

- 12, nombre de clusters actuels.

- 11, nombre de clusters possédants plus de 15 éléments.

Voici un schéma décrivant le processus général de cette hybridation :

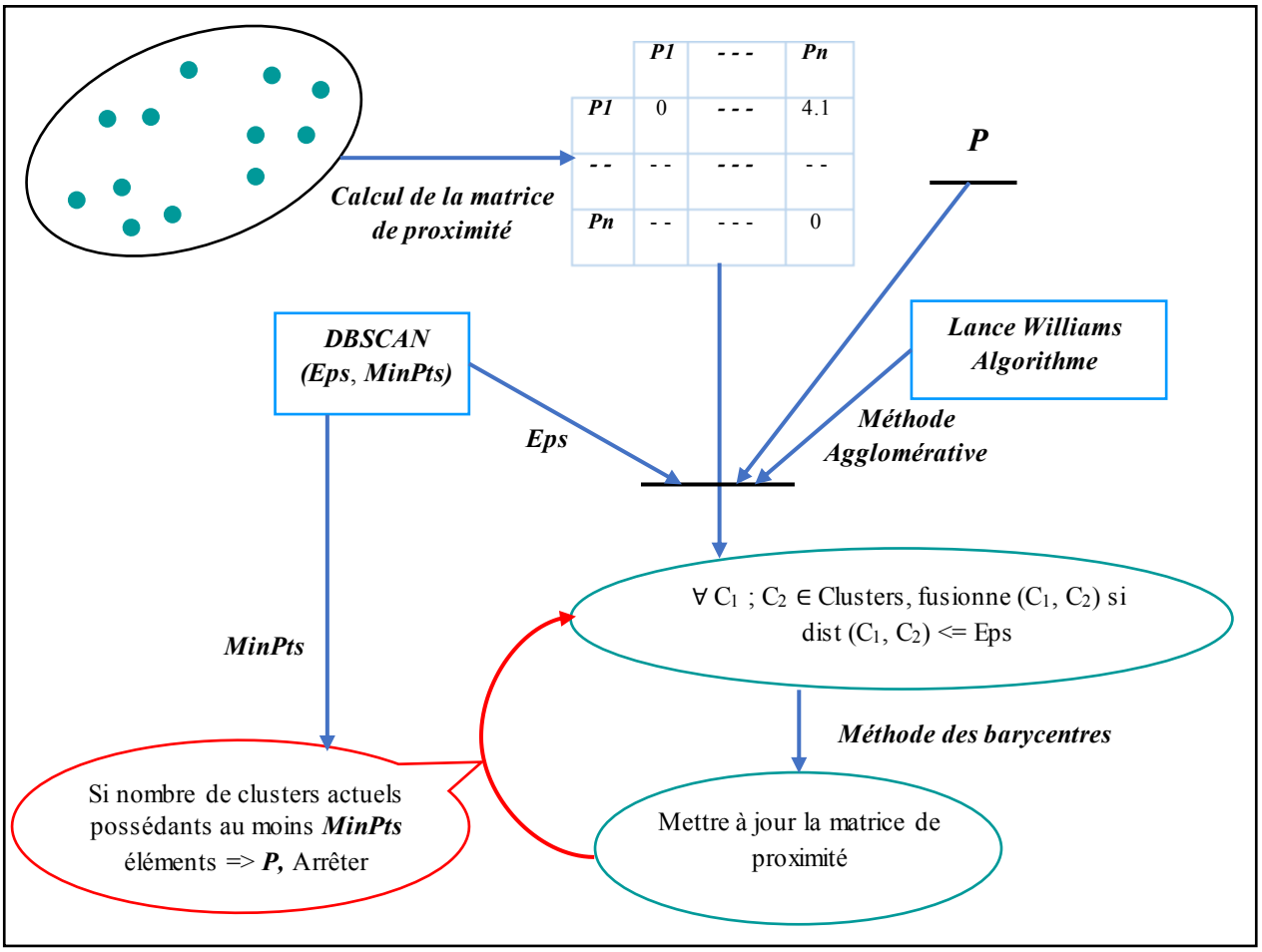


Figure 17 : Schéma montrant le processus global de la méthode AggloDens

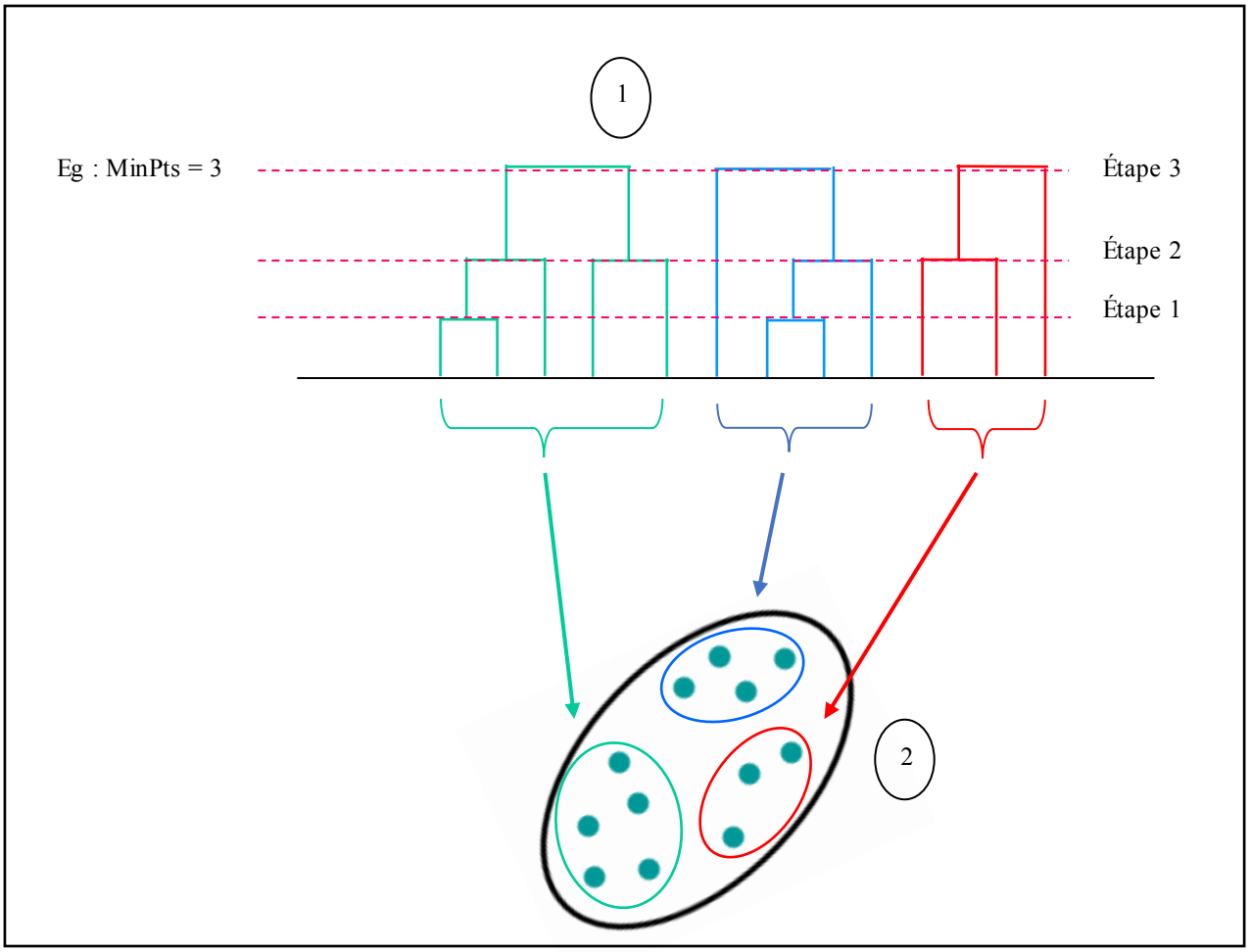


Figure 18 : Exemple d'un dendrogramme résultant de l'application de la méthode AggloDens

L'arbre (1) est un ***dendrogramme*** qui est utilisé pour illustrer l'arrangement de groupes générés. Ce dendrogramme montre comment cette hybridation forme tous les clusters possibles (plusieurs clusters à la fois) en respectant le ***Eps*** à chaque agglomération. Et lorsqu'ils atteignent le ***MinPts*** l'exécution s'arrête et on obtient le résultat de (2).

4. Méthode hybride proposée :

La solution hybride que nous proposons ici pourrait être généralisée pour partitionner des ensembles de données si volumineux et même multi-types. On peut la résumer en quatre étapes majeures :

1- Préparation de données.

2- Calcul de la matrice de proximité.

3- Algorithme de clustering hybride.

4- Exploitations des résultats.

4.1. Préparation de données et calcul de la matrice de similarité :

Avant d'appliquer n'importe quelle méthode de clustering, il faut préparer les données afin de les partitionner. Pour chaque type de données, il existe plusieurs formules pour calculer la similarité. Puisque le principe de clustering est de fusionner les éléments les plus proches (dont la distance entre eux est plus petite), dans notre approche hybride nous utilisons les formules les plus convenables au clustering. Nous utilisons les formules assurant que plus les données sont similaires, plus la valeur de distance est faible, et si les données sont identiques la distance égale à 0.

a. Donnée numériques :

Comme cité au premier chapitre, pour calculer la similarité entre données numériques, il faut utiliser soit la distance de Minkowski, la distance de Manhattan, ou la distance euclidienne ou bien d'autres, et celle que nous utilisons dans cette hybridation est la distance euclidienne,

$$\boldsymbol{d(i,j)} = \sqrt{\sum_{k=1}^{d} (\, \boldsymbol{i_k} - \boldsymbol{j_k} \,)^2}$$

C'est la formule la plus utilisée pour ce type de données, car elle donne de bonnes valeurs de similarité.

b. Données binaires :

Ce type de données nécessite tout d'abord le calcul de la matrice de contingence avec ses quatre essentiaux indices ; ***a***, b, c et d. tel que :

- ***a*** est le nombre de données communes entre deux objets.

- ***b*** nombre de données possédées par l'un des objets, et ***c*** nombre de données possédées par l'autre.

- ***d*** le nombre du reste de données.

- $p = a + b + c + d$

Ensuite utiliser le coefficient simple d'appariement $d(i, j) = \frac{b + c}{p}$.

c. Données nominales :

Pour ce type de donnée, il faut une normalisation tout d'abord afin de calculer la similarité, en se basant sur la correspondance simple dans notre approche, telle que $d(i, j) = \frac{p - m}{p}$, avec p le nombre total de variables, et m nombre de correspondance.

d. Données textuelles :

Pour ce type de données, il y a une grande diversité de formule pour calculer la similarité, notamment la distance de Hamming, Jaccard, Levenshtein, Sorensen, etc. Et là nous nous basons sur la distance euclidienne avec le vecteur ***TF*** (***Natural Frequence***), car c'est la méthode qui respecte le principe selon lequel plus les données sont similaires, plus la valeur de similarité est petite.

Après cette étape préparative, nous calculons la matrice de proximité en se basant sur les formules choisisses précédemment et en appliquant la sommation tel que :

$$Distance_{totale} = (distance_{numérique} + distance_{binaire} + distance_{nominale} + distance_{textuelle})$$

4.2. Algorithme de clustering hybride :

Dans cette approche ***AggloDens***, nous combinons deux techniques différentes pour avoir une technique hybride, une technique principale non paramétrique, c'est la technique hiérarchique agglomérative et précisément l'***algorithme de Lance William***, et l'autre paramétrique, c'est la méthode par densité qui nécessite deux paramètres en entrée ***Eps*** et ***MinPts***, précisément l'algorithme de ***DBSCAN***.

4.2.1. Principe de fonctionnement :

Le processus de cette hybridation se base essentiellement sur une hiérarchie ascendante. D'autre manière, cette approche fonctionne de la même façon que l'algorithme de Lance William qui forme des clusters en construisant un arbre des instances agrégées (regroupées) à chaque étape, commençant par un groupe de clusters contenant une seule instance (feuille) jusqu'à arriver au résultat final (un groupe de clusters). Durant chaque étape (agglomération), on fusionne les éléments dont la distance entre eux est inférieure ou égale à ***Eps***. La condition d'arrêt de ce processus est la taille des clusters où chacun contient au minimum un ***MinPts*** instances, au lieu d'exiger le nombre de cluster qui est le plus délicat paramètre dans le clustering sans rien savoir à priori. Alors le ***MinPts*** est définie comme la taille minimale des clusters exigée. En utilisant ces deux paramètres, on assure la notion de densité dans chaque cluster, c'est-à-dire les clusters qui résultent soient denses.

Brièvement, On peut décrire le déroulement du processus que nous parlons ici dans ces deux essentiaux points :

- Prendre de la technique agglomérative la manière comment former les clusters (regroupement hiérarchique ascendant).

- Prendre de la technique par densité ses paramètres comme conditions exigées pour valider ces clusters, ***Eps*** la distance au maximum entre les éléments d'un cluster et ***MinPts*** la taille minimale d'un cluster.

Dans chaque étape (agglomération), il y trois choses élémentaires à faire :

1- Agréger les éléments dont la distance entre eux est inférieure ou égale à **Eps**.

2- Recalculer la distance entre les anciens éléments et les nouvelles grappes formées en utilisant la méthode des barycentres.

3- Vérifier si tous les clusters ont au moins un ***MinPts*** instances, si oui arrêter le processus a cette étape courante, Sinon aller à l'étape (1).

4.2.2. Choix d'indice d'agrégation :

Une étape essentielle dans cette méthode hybride est la mise à jour de la matrice de proximité après chaque agrégation. Pour assurer de bon résultat en baissant l'inertie intra-classe, nous

appliquons ici une technique basée sur la distance entre les barycentres des clusters, Le critère de choix de la paire de clusters à fusionner à chaque étape est basé sur la distance minimale entre les barycentres.

4.2.2.1 Principe de fonctionnement de la méthode des barycentres :

Cette technique consiste à réunir les deux clusters dont le regroupement fera le moins baisser l'inertie intra-classe. La distance entre deux classes est celle de leurs barycentres. Cette technique tend à regrouper les petites classes entre elles, tel que plus le nombre d'instances d'une classe est petit plus sa valeur d'inertie intra-classe est faible.

Le barycentre peut être complètement artificiel, ne correspondant pas à la réalité des données. On préfère dans certains cas la notion de médoïde (l'élément le plus placé au centre d'un cluster), un point qui existe réellement et qui correspond à l'observation qui minimise sa distance avec l'ensemble des autres points [28].

Supposant à une étape quelconque, on a un ensemble de quatre clusters C1, C2, C3 et C4 tel que :

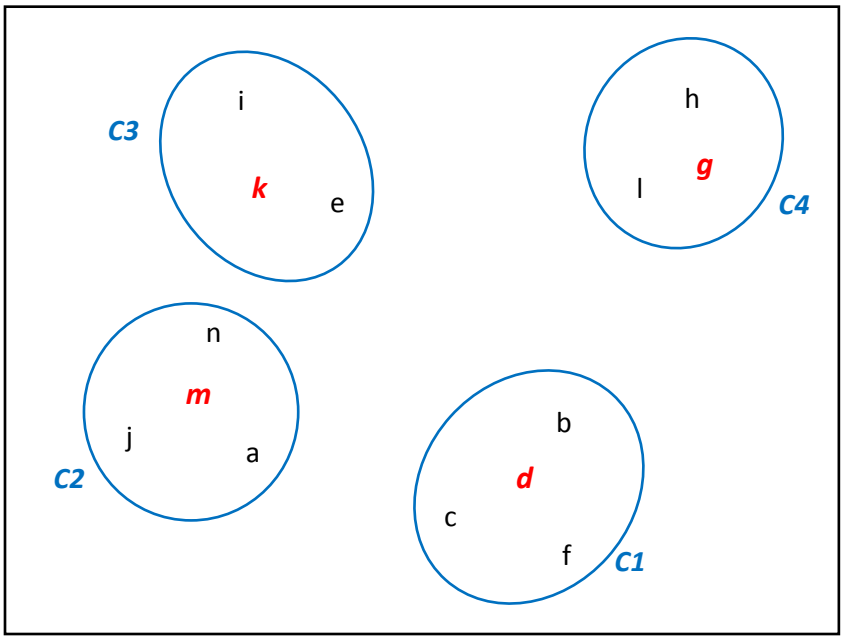


Figure 19 : Exemple d'un ensemble de clusters à une étape quelconque

La prochaine itération, C2 et C3 sont les plus proches (leurs barycentres ***k*** et ***m*** sont plus proches par rapport aux autres), donc ils vont être fusionnés en un seul cluster, et l'élément ***n*** devient le barycentre du nouveau cluster C' (car il est le plus centré dans C').

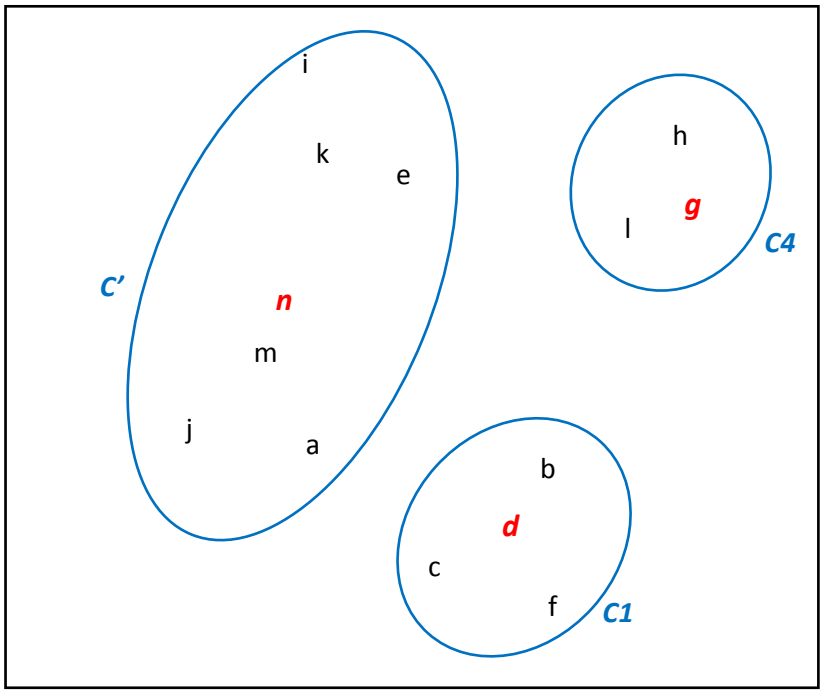


Figure 20 : L'ensemble de clusters après un nouveau fusionnement

4.2.3. Algorithme AggloDens :

1- Initialiser toute instance en un cluster.

2- Définir ***Eps***, ***MinPts*** et ***P***.

3- Fusionner tous les clusters dont la distance entre eux est inférieur ou égale à ***Eps***.

4- Mettre à jour la matrice de proximité en se basant sur les ***barycentres***.

5- refaire 3 à 4 jusqu'à où le nombre de clusters actuels possédant au moins ***MinPts*** éléments égale ou supérieur à ***P***.

Le résultat de cette méthode hybride peut être sous forme de dendrogramme comme celui de l'algorithme de Lance Williams, donc il est possible de l'analyser et de modifier les paramètres exigés (***Eps***, ***MinPts***) si nécessaire pour améliorer les grappes.

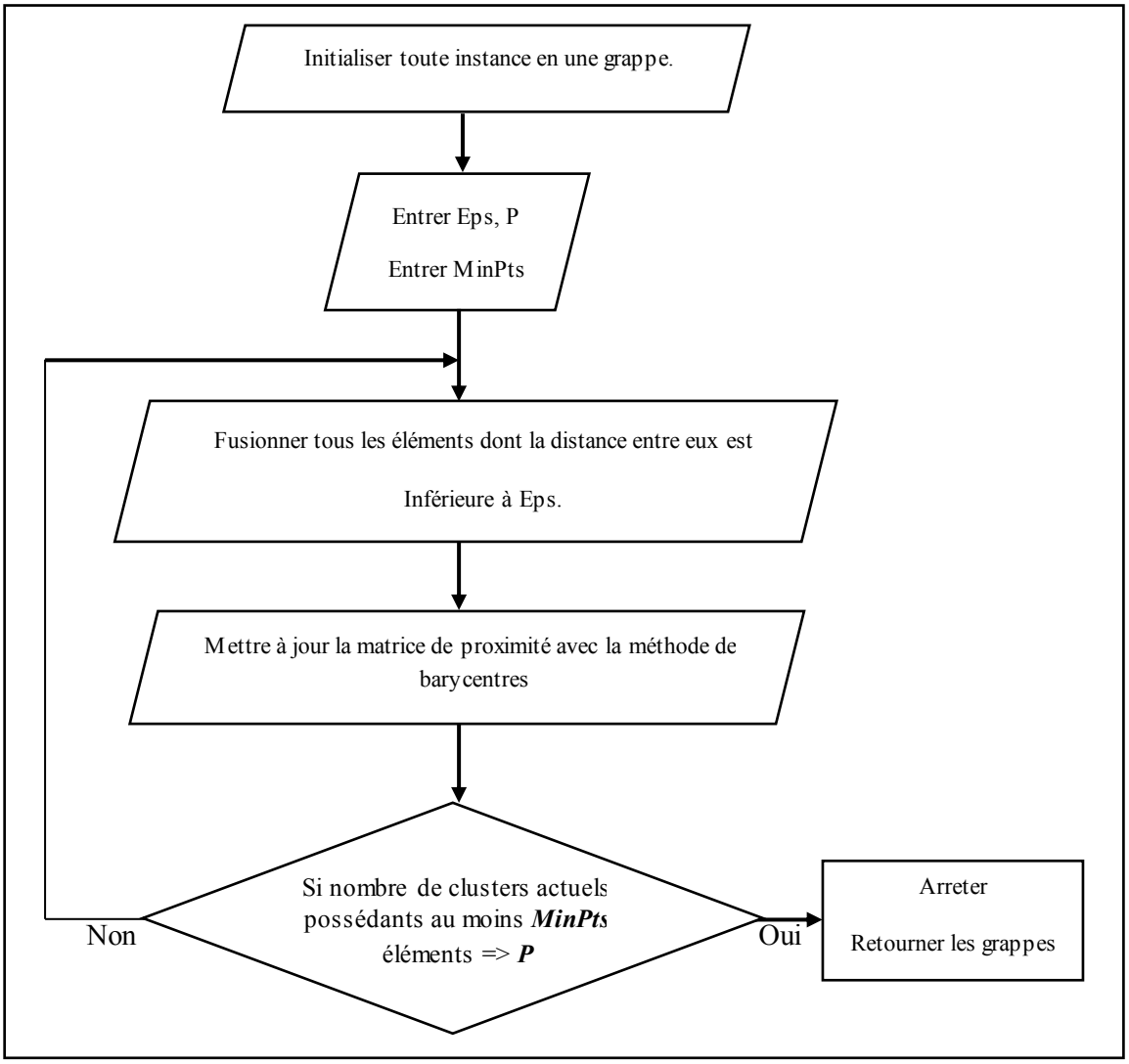


Figure 21 : Organigramme de l'algorithme AggloDens proposé

Le schéma ci-dessous illustre le déroulement de notre méthode AggloDens :

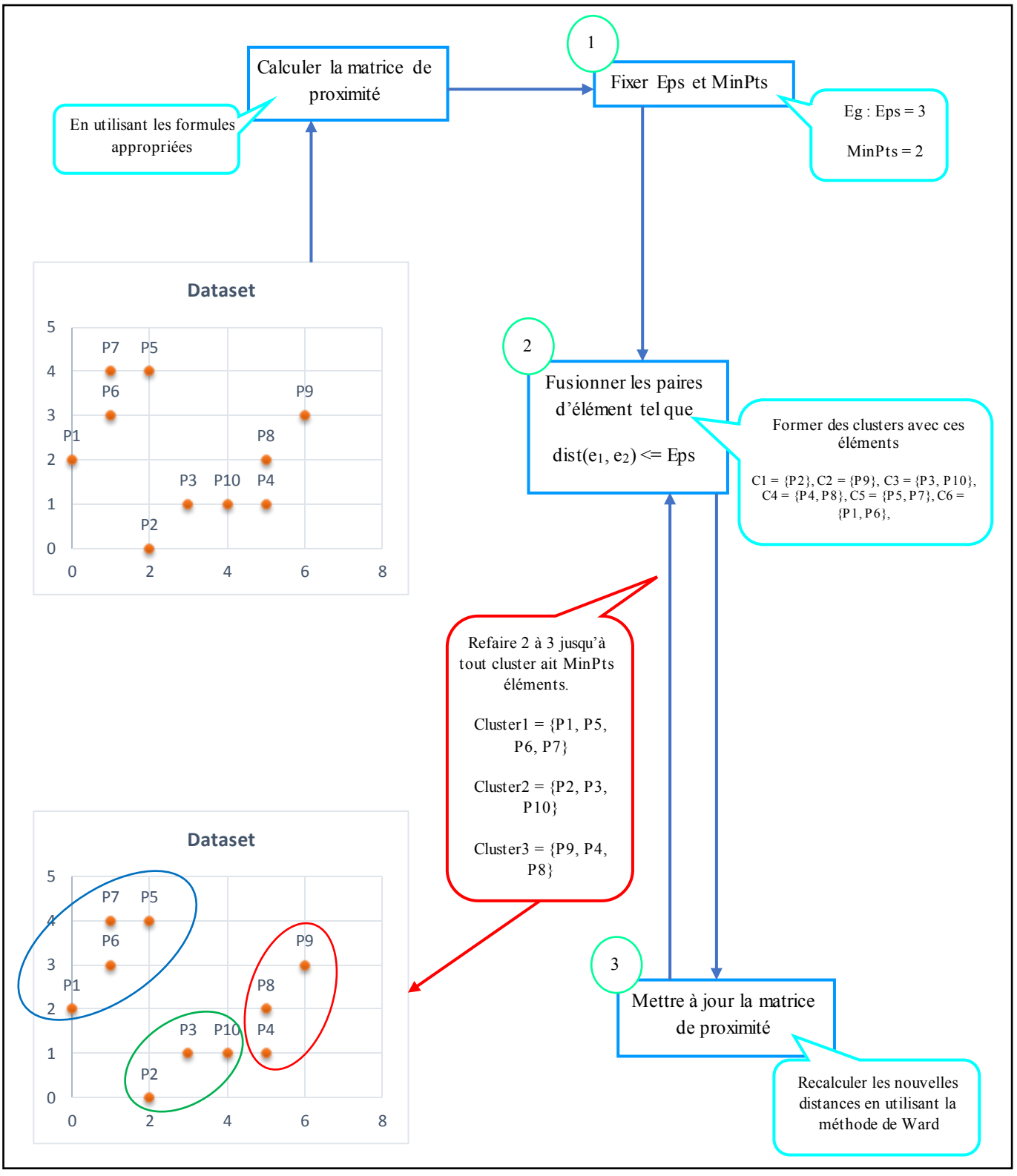


Figure 22 : Exemple du déroulement de la méthode AggloDens

En comparant les résultats obtenus ci-dessus avec ceux de l'algorithme de ***Lance Williams***, on trouve que pour avoir les mêmes résultats (***Clusters {[P1, P6, P5, P7], [P3, P10, P2], [P8, P4, P9]}***), le nombre d'itérations dans la première implémentation est plus grand (quatre itérations) par rapport à notre méthode hybride (deux itérations). Un gain de deux itérations avec un petit ensemble de données, donc plus l'ensemble est grand, plus le nombres d'itérations éliminés est grand.

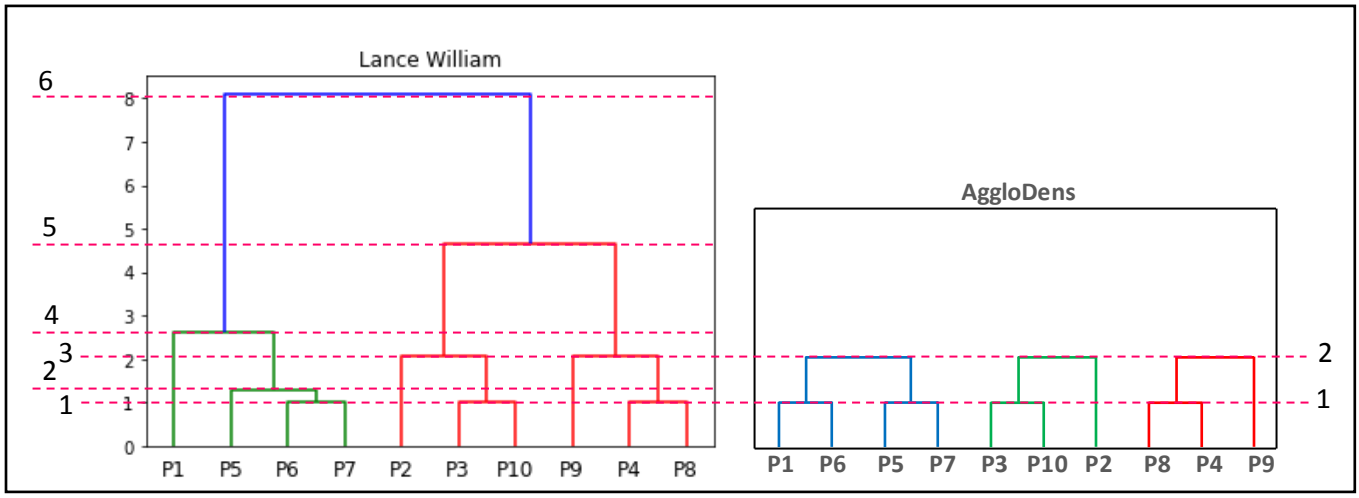


Figure 23 : Dendrogrammes résultants de l'implémentation d'un exemple avec l'algorithme de Lance Williams et AggloDens

5. Conclusion :

A travers ce chapitre, nous avons détaillé la conception et le processus de notre approche hybride proposée qui combine entre deux grandes techniques de clustering de base, la technique hiérarchique agglomérative, et celle par densité. Ainsi que nous avons présenté les points forts de cette hybridation par rapport aux autres méthodes existantes. Nous pouvons dire également qu'elle est une approche généralisée qui peut être utilisée avec tout type et taille de données. Le chapitre suivant nous allons l'implémenter pour voir réellement comment elle fonctionne.

Chapitre III :

Expérimentation et évaluation

1. Introduction :

Dans le chapitre précédent, nous avons parlé de la conception de la méthode de clustering ***AggloDens*** proposée qui combine la technique hiérarchique agglomérative et la technique par densité. Nous avons détaillé le principe de fonctionnement, les méthodes sur lesquelles elle se base, avec des schémas illustrant ce processus.

Afin de contribuer à une meilleure compréhension de cette méthode et de l'impact sur les résultats de la distribution des données, il faut l'implémenter et la comparer avec d'autres méthodes existantes. Cette réalisation vient couronner le travail de l'étude conceptuelle. Elle présente la dernière étape, et elle est très importante, car elle dessine les points forts et faibles de la nouvelle méthode par rapport aux autres méthodes.

Pour mettre en œuvre notre solution hybride, nous allons parler tout d'abord dans ce chapitre des outils nécessaires à l'implémentation de ***AggloDens*** ainsi que d'autres méthodes existantes, en l'occurrence le langage de programmation et les bases de données utilisées. Puis nous présenterons le processus implémenté ainsi que les différentes méthodes et formules utilisées. Ensuite nous passerons à l'exécution de quelques modèles avec des différentes bases de données, et l'évaluations des résultats obtenus. Enfin nous comparons les résultats obtenus avec d'autres des méthodes combinées.

2. Outils utilisés :

Afin de rendre cette méthode proposée réelle, nous avons l'implémenté en utilisant ***Python***, le langage le plus utilisé dans le domaine de la Science de Données et le Machine Learning. Le choix d'un bon environnement de programmation est très important pour le développement des projets.

2.2. Matériels :

Nous avons exploité et implémenté notre programme sous le système d'exploitation Windows version 10, processeur Intel Core i5-6300 CPU @ 2.4 GHz, 4.00 Go de Ram.

2.2. Langage de programmation :

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes. L'interpréteur Python et sa vaste bibliothèque standard sont disponibles librement [29].

2.3. Environnement de développement :

Et comme environnement de développement, nous avons utilisé ***Spyder***, un environnement scientifique puissant écrit en Python, pour Python. Il offre une combinaison unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec l'exploration de données, l'exécution interactive, l'inspection approfondie et les superbes capacités de visualisation d'un package scientifique [30].

2.4. Bases de données utilisées :

Pour appliquer notre programme, nous avons utilisé trois ensembles de données de différent type et taille, au format ***CSV*** (***Comma-Separated Values***).

Mall customers [31] :

Cet ensemble de données est créé uniquement à des fins d'apprentissage des concepts de segmentation client, également appelés analyse du panier de marché constitué de 200 instances. 5 attributs ont été tirés :

- ***Attributs numériques :*** clientID, age, annualIncome (revenue annuel), spendingScore (Le score de dépense est un score attribuer au client en fonction des paramètres définis tels que le comportement du client et les données d'achat).

- ***Attribut binaire :*** gender.

Le résultat de l'application du clustering sur cet ensemble sert à déterminer les groupes de clients en fonction de leurs comportements de dépenses, afin de pouvoir connaitre qui sont les clients cibles avec lesquels il est possible de lancer une stratégie marketing.

```
import pandas as pd
pd.read_csv('/content/mallCustomers.csv')
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

Figure 24 : Un échantillon de la base de données « Mall customers »

Country data [32] ***:***

Cet ensemble de données contient les profils des pays pauvres dans le monde entier. Il est créé pour pouvoir aider l'organisation de « ***HELP international*** [1] » à choisir les pays qui ont le plus besoin d'aide. Il est constitué de 168 instances. Le résultat de l'application du clustering sur cet ensemble sert à classer les pays en utilisant des facteurs socio-économiques et sanitaires qui déterminent le développement global du pays.

[1] HELP International est une organisation qui s'est engagée à lutter contre la pauvreté et à fournir aux populations des pays arriérés des équipements de base et des secours en temps de catastrophes et de calamités naturelles

Out[2]:

country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	1220

Figure 25 : Un échantillon de la base de données « Country data »

Utilisateurs de services web :

Cet ensemble de données résume l'intérêt des utilisateurs de différents services web. Il est constitué de 4000 instances et 38 attributs de différent type (26 attributs binaires, 7 attributs numériques, 2 attributs nominaux et 1 attribut de type texte). Nous avons éliminé deux attributs qui ne servent à rien (***userID***, ***userName***) dans cette opération.

L'objectif est de grouper les utilisateurs similaires d'apres les détails de leurs profils.

	id	name	sexe	School level	Chinese	English	French	German	Italian	Japanese	Russian	Spanish	Turkish	Gaming	Social media	Economic	Education	movies	culture	description	capacite	performance	fiabilite	Dispo
0	1	User1	Male	None	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	Online messaging offering a large storage spac...	1813	21.83	60.42	
1	2	User2	Male	Higher than professor	1	1	1	1	1	0	1	0	1	1	1	0	1	0	1	Search engine for computer code.	841	43.35	86.97	
2	3	User3	Female	None	0	0	1	0	1	0	0	1	0	0	0	0	0	1	1	Paying service offering the ability to create...	6008	17.43	17.54	
3	4	User4	Female	Baccalaureate	1	0	0	0	1	1	0	0	1	0	0	0	1	0	1	Research in online scientific publications	4022	10.81	55.73	
4	5	User5	Female	Non-university diploma	1	1	1	0	0	1	1	1	0	1	1	0	1	0	0	Online music service	4471	52.75	66.78	
5	6	User6	Female	Non-university diploma	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	Search for products that can be purchased online	3942	26.61	65.24	
6	7	User7	Male	Secondery	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0	Contact geolocation service	7911	96.14	64.49	

Figure 26 : Un échantillon de la base de données « Utilisateurs de services web »

3. Implémentation de la méthode AggloDens :

En général, la réalisation de notre méthode ***AggloDens*** se divise en trois étapes majeures :

1. Implémentation.
2. Exécution et exploration des résultats.
3. Évaluation et validation des bons résultats.

Nous avons adopté une implémentation naïve pour notre méthode ***AggloDens*** en se basant essentiellement sur les algorithmes de ***Lance Williams*** et le ***DBSCAN***.

Le programme est divisé en 3 sous-programmes :

a- Le premier consiste à calculer la matrice de similarité (distance).

b- Le 2em est consacré pour l'algorithme ***AggloDens***.

c- Et le dernier pour l'évaluation de la méthode.

3.1. Calcul de la matrice de distance :

Cette partie contient les différentes fonctions qui calculent la distance entre les données de différent type. Afin d'avoir les bonnes similarités, nous avons utilisé plusieurs fonctions pour un même type et après l'application nous avons adopté celle qui nous a assuré que plus les données sont similaires, plus la valeur de distance est faible.

Prenant l'exemple des données textuelles, il existe de nombreuses fonctions prédéfinies sous la bibliothèque ***textdistance*** de Python. Parmi eux que nous avons essayé :

```
102
103 import textdistance
104
105
106
107 ''' Distance between string data using Hamming Distance '''
108 def hammingDist(text1, text2):
109     return textdistance.hamming.normalized_similarity(text1, text2)
110
111
112
113 ''' Distance between string data using Jaccard Distance '''
114 def jaccardDist(text1, text2):
115     return textdistance.jaccard(text1.split(), text2.split())
116
117
118
119 ''' Distance between string data using Levenshtein Distance '''
120 def levenshteinDist(text1, text2):
121     return textdistance.levenshtein.normalized_similarity(text1, text2)
122
123
124
125 ''' Distance between string data using Sorensen Distance '''
126 def sorensenDist(text1, text2):
127     return textdistance.sorensen(text1.split(), text2.split())
128
129
```

Figure 27 : Ensemble de fonctions pour le calcul de la similarité entre des données textuelles

En les appliquant sur les deux textes suivants :

```
In [54]: text1 = 'Search engine for computer code.'

In [55]: text2 = 'Search engine dedicated to music.'
```

Nous obtenons :

```
In [68]: sorensenDist(text1, text2)
Out[68]: 0.4

In [69]: sorensenDist(text1, text1)
Out[69]: 1

In [69]:

In [70]: levenshteinDist(text1, text2)
Out[70]: 0.5151515151515151

In [71]: levenshteinDist(text1, text1)
Out[71]: 1.0
```

```
In [64]: hammingDist(text1, text2)
Out[64]: 0.48484848484848486

In [65]: hammingDist(text1, text1)
Out[65]: 1.0

In [65]:

In [66]: jaccardDist(text1, text2)
Out[66]: 0.25

In [67]: jaccardDist(text1, text1)
Out[67]: 1
```

Figure 28 : Résultats de l'application des fonctions pour le calcul de la similarité entre données textuelles

Nous remarquons qu'avec les résultats ci-dessus, nous ne sommes pas arrivés à notre objectif. Tel que la similarité entre deux textes identiques (Dans l'exemple nous avons utilisé le même texte -text1- pour le test) égale à 1 au lieu 0 que ce soit la méthode utilisée, Hamming ou Jaccard... C'est pourquoi nous avons pensé à créer des fonctions naïves. Nous avons créé trois fonctions :

- La première se base sur la normalisation binaire et la formule de cosinus.
- La deuxième se base sur la normalisation en TF (Natural Frequence) et la formule de cosinus.
- La dernière se base sur la normalisation en TF et la distance euclidienne.

Ensuite nous avons adopté celle qui donne la meilleure valeur de similarité, c'est la 3eme fonction.

En l'appliquant, nous obtenons :

```
  * text1+text2 = {'engine', 'music', 'code', 'dedicated', 'for', 'to', 'computer', 'search'}
   * text1 =  search engine for computer code
    [1, 0, 1, 0, 1, 0, 1, 1]
   * text2 =  search engine dedicated to music
    [1, 1, 0, 1, 0, 1, 0, 1]
Out[98]: 2.449489742783178
In [99]: textDataDist3(text1, text1)
  * text1+text2 = {'engine', 'code', 'for', 'computer', 'search'}
   * text1 =  search engine for computer code
    [1, 1, 1, 1, 1]
   * text2 =  search engine for computer code
    [1, 1, 1, 1, 1]
Out[99]: 0.0
```

Figure 29 : Résultats de l'application des fonctions crées pour le calcul de la similarité entre données textuelles

Nous remarquons ici que la similarité (distance) entre deux texte identiques (text1 – text1) égale à 0. C'est le résultat qui nous aide à réaliser la première étape correctement.

Après le choix des plus convenables fonctions, en les appliquant nous créons la matrice de proximité sous forme d'un ***DataFrame***.

Pour le cas de données multi-type, nous utilisons la méthode de sommation pour calculer la distance entre ces données.

```
similarity = round(binaryDataDist(binaryData[i],binaryData[j])
                   +numericalDataDist(numericalData[i],numericalData[j])
                   +nominalDataDist(nominalData[i],nominalData[j])
                   +textDataDist3(stringData[i],stringData[j]), 3)
```

```
pd.read_csv('/content/mallCustomers2.csv')
```

	Name	customer1	customer2	customer3	customer4	customer5	customer6	customer7	customer8	customer9	customer10	customer11	customer12	customer13	customer14	customer15	customer16	custome
0	customer1	NaN	42.048	34.030	39.223	13.207	38.175	37.797	56.227	57.767	36.014	54.268	63.225	47.065	39.652	32.016	40.423	18.
1	customer2	NaN	NaN	76.013	5.583	43.249	6.477	77.354	14.491	89.157	14.342	81.370	24.152	76.829	8.071	70.036	5.477	49.
2	customer3	NaN	NaN	NaN	71.063	35.749	70.036	15.133	88.074	45.204	66.821	48.770	94.250	39.256	71.225	19.815	74.137	33.
3	customer4	NaN	NaN	NaN	NaN	37.868	1.732	72.035	17.117	85.652	9.110	77.903	25.239	71.309	4.123	66.635	5.583	43.
4	customer5	NaN	NaN	NaN	NaN	NaN	37.108	34.249	54.599	50.619	32.078	45.452	59.169	36.919	37.776	28.821	41.137	7.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
195	customer196	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
196	customer197	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
197	customer198	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
198	customer199	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
199	customer200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

200 rows × 201 columns

Figure 30 : Échantillon de la matrice de proximité de la base de données « Mall customers »

3.2. L'algorithme AggloDens :

Après calculer la matrice de proximité, nous passons à l'algorithme de la méthode ***AggloDens***.

1- Initialisation des clusters : Tout d'abord initialiser des simples clusters (single clusters),

```
Clusters =  {'user1': 'user1', 'user2': 'user2', 'user3': 'user3', 'user4': 'user4',...}
```

2- Recherche des distances inférieures à Eps : Chercher toutes les distances inférieures à ***Eps*** en les retournant dans une liste, chacune avec les indices des éléments à fusionner,

```
distanceWithinEps = distWithinEps(simDataCopy, eps)
```

Pour éviter les cas d'ambiguïté, par exemple on a trois éléments a, b et c, la distance entre a et b est inférieure à Eps (avec distance(a, b) = 1,2), et même celle entre a et c est inférieure à Eps (avec distance(a, c) = 0,91), donc pour savoir quels éléments à fusionner a et b ou a et c, il vaut mieux trier cette liste par ordre croissant, puis fusionner les éléments dont la distance entre eux est la plus petite, donc fusionner a et c.

```
distanceWithinEps = sorted(distanceWithinEps, key=lambda x: x[0])
```

3- Construction des nouveaux clusters : Après avoir la liste des distances inférieures à Eps, fusionner les éléments de cette liste deux par deux et mettre à jour la matrice de proximité en :

- Supprimant les lignes et les colonnes des éléments fusionnés.

```
similarityData = similarityData.drop([newIdxJ, newIdxK], axis=0)
similarityData = similarityData.drop([jElement, kElement], axis=1)
```

- Ajoutant une nouvelle colonne pour le nouveau cluster en se basant sur la distance entre les barycentres des clusters.

```
similarityData.insert(len(similarityData), 'cluster'+str(newID), newClusterSim)
```

Dans cette étape, il faut chercher tout d'abord les barycentres de tous les clusters actuels. Ensuite calculer la distance entre les anciens clusters et le nouveau en calculant la distance entre leurs barycentres.

```
def getBarycenter(cluster, similarityData):
```

- Mettre à jour la liste des clusters en ajoutant le nouveau cluster et en supprimant les clusters fusionnés.

```
clusters['cluster'+str(newID)] = clusters[jElement] + clusters[kElement]
clusters.pop(jElement)
clusters.pop(kElement)
```

Pour éviter les cas d'ambigüité ci-dessus, utiliser une liste additionnelle,

```
elementLabel = [0]*len(similarityData)
```

Où ***elementLabel*** est une liste de taille égale au nombre de clusters au courant, contenant des 0 et 1 tel que :

- 0, signifie que l'élément n'est pas fusionné encore avec d'autre.

- 1, signifie que l'élément est fusionné déjà avec un autre, donc ne pas pouvoir de le fusionner à nouveau.

Initialiser cette liste additionnelle à 0. Et après chaque fusion il faut la mettre à jour tel que la valeur de chaque élément fusionné devient 1.

4- Condition d'arrêt : Pour arrêter l'algorithme, nous vérifions la taille des clusters actuels si elle est supérieure à MinPts. Sinon nous réexécutons le processus commençant par l'étape 2 jusqu'à avoir des clusters de taille égale ou supérieure à MinPts.

```
if lenClusters(clusters, MinPts) != True:
```

En cas de ne plus trouver des distances inférieures à Eps, et la taille des clusters actuels est inférieure à MinPts, nous quittons la boucle et nous redéfinissons la valeur de Eps ou de MinPts ou des deux ensembles.

```
if lenClusters(clusters, MinPts) != True:
    print('\n   Oops! Change Eps or MinPts value.. \n')
```

En cas de données assez larges ou déséquilibrées [2], nous utilisons un pourcentage (***p***) tel que si la taille de ***p*** des clusters est supérieure ou égale à MinPts, nous quittons la boucle de l'algorithme. C'est-à-dire, a une étape quelconque, lorsque nous avons un nombre très petit de clusters dont la taille est inférieure à MinPts, il suffit d'exiger un ***p*** (avec eg : ***p = 95%***) de clusters ayant au moins MinPts éléments pour valider le résultat final.

[2] Les données déséquilibrées sont les données quand on les répartit avec n'importe quelle méthode de clustering, les clusters résultants soient de taille divergente.

```
165
166 ''' Percentage of clusters possess at least minPts elements '''
167 def lenClusters(clusters, minPts):
168     nbrValidatedClusters = 0
169     for i in clusters:
170         if len(clusters[i]) >= minPts:
171             nbrValidatedClusters += 1
172     if nbrValidatedClusters >= round(len(clusters)*0.95, 2):
173         return True
174     else:
175         return False
176
```

L'utilisation du pourcentage p nécessite la vérification si toute instance appartient à une classe afin de valider les résultats.

3.3. Évaluation des résultats :

Après implémenter l'algorithme ***AggloDens***, il faut l'appliquer sur l'ensemble de données plusieurs fois en changeant les valeurs de Eps et MinPts pour avoir un bon clustering. Afin de valider les bons résultats, il existe de nombreux critères d'évaluation. De notre part, nous avons appliqué la notion de ***l'inertie intra-classe*** (vue en premier chapitre). À chaque exécution, nous calculant l'inertie intra-classe, puis nous validons les résultats qui minimise le maximum la valeur de cette inertie.

```
221 ''' Inertia intra-classe'''
222 def inertiaIntraClass(clusters, simData):
223     inertiaIntraClass = 0
224     for cluster in clusters:
225         totalDistBarycenter = getBarycenter(clusters[cluster], simData)[1]
226         inertiaIntraClass +=  totalDistBarycenter
227     return inertiaIntraClass
228
```

4. Application de l'algorithme AggloDens :

Pour appliquer l'algorithme AggloDens, il faut définir trois paramètres Eps, MinPts et le pourcentage P. Afin de savoir comment choisir les valeurs de ces paramètres, nous avons exécuté plusieurs modèles en changeant ces valeurs à chaque exécution et en analysant et évaluant les résultats obtenus. Voilà un tableau résumant les valeurs utilisées avec les résultats obtenus pour les trois ensembles de données vus au début de ce chapitre :

		Entrée			Sortie		
Les datasets	**N Exécution**	**MinPts**	**Eps**	**P**	**Nombre de clusters**	**Nombre d'itération**	**Inertie intra-classe**
Utilisateurs de services web	1	80	870	95%	**8**	**9**	**679015**
	2	80	1200	95%	**12**	**8**	**551256**
	3	80	3000	95%	**9**	**8**	**787099**
	4	100	1000	95%	**8**	**9**	**679015**
	5	100	1500	95%	**6**	**9**	**898821**
	6	100	2000	95%	**9**	**8**	**747872**
	7	100	4000	95%	**9**	**8**	**787099**
	8	100	5000	95%	**16**	**7**	**754309**
	9	140	3150	95%	**5**	**9**	**1167195**
	10	140	6000	95%	**8**	**8**	**1212907**
	11	150	2300	95%	**5**	**9**	**1099230**
	12	200	1200	95%	**6**	**10**	**898821**
	13	200	2000	95%	**5**	**9**	**1099230**
	14	330	2300	95%	**4**	**9**	**1315225**
	15	400	3500	95%	**3**	**10**	**1875459**
	16	500	5500	95%	**2**	**10**	**2517717**
	17	230	1000	95%	**Pas de résultat !**		
Counntry data	1	8	55000	95%	**22**	**3**	**617358**
	2	10	60000	95%	**11**	**4**	**898250**
	3	12	55000	95%	**7**	**7**	**898573**

	4	16	1300	95%	***Pas de résultat !***		
	5	16	5000	95%	***12***	***4***	***738850***
	6	16	20000	95%	***7***	***7***	***929337***
	7	16	80000	95%	***6***	***5***	***1125198***
	8	20	60000	95%	***3***	***6***	***1363260***
	9	20	75000	95%	***3***	***6***	***1363260***
Mall customers	1	8	20	95%	***17***	***7***	***2513***
	2	8	68	95%	***14***	***4***	***2638***
	3	10	35	95%	***10***	***5***	***3032***
	4	10	45	95%	***6***	***6***	***3550***
	5	15	45	95%	***Pas de résultat !***		
	6	15	60	95%	***4***	***6***	***4718***
	7	15	80	95%	***7***	***5***	***4194***
	8	20	45	95%	***3***	***9***	***5194***
	9	24	50	95%	***3***	***9***	***6051***
	10	32	65	95%	***4***	***6***	***4718***
	11	32	72	95%	***2***	***7***	***6413***

Tableau 4 : Les valeurs utilisées dans l'application de l'algorithme AggloDens sur les bases de données

Tout d'abord, nous avons commencé avec des petites valeurs (tout dépend la taille de l'ensemble de données utilisé et les valeurs de la matrice de proximité). Ensuite nous avons fait un changement (changement par augmentation ou diminution) pour ces valeurs petit à petit. Pour le pourcentage P, nous avons utilisé une valeur élevée (95% pour toutes les exécutions) et cela n'a pas affecté le résultat.

5. Résultats obtenus :

Avec ces exécutions, nous avons obtenu des résultats divers où la valeur de l'inertie intra-classe, le nombre d'itération et le nombre de clusters résultants se diffèrent d'une exécution à une autre tout dépend les valeurs des paramètres exigés.

Nous illustrerons les résultats sous forme de graphes afin que nous puissions les analyser. Un graphe pour le changement du nombre de clusters résultant à chaque itération. Un deuxième pour le changement de l'inertie intra-classe en fonction de Eps et MinPts. Et un troisième pour le changement de l'inertie en fonction du nombre de clusters.

Voici les graphes illustrant les résultats obtenus sur les différents ensembles de données utilisés :

Nous observant ici que le nombre de clusters résultants se change par diminution ou augmentation d'un modèle à un autre. Et puisqu'il y a trois contraintes exigées ensemble, il n'y a pas de règle pour dire le nombre de clusters s'augmente ou se diminue en fonction d'un paramètre ou d'un autre.

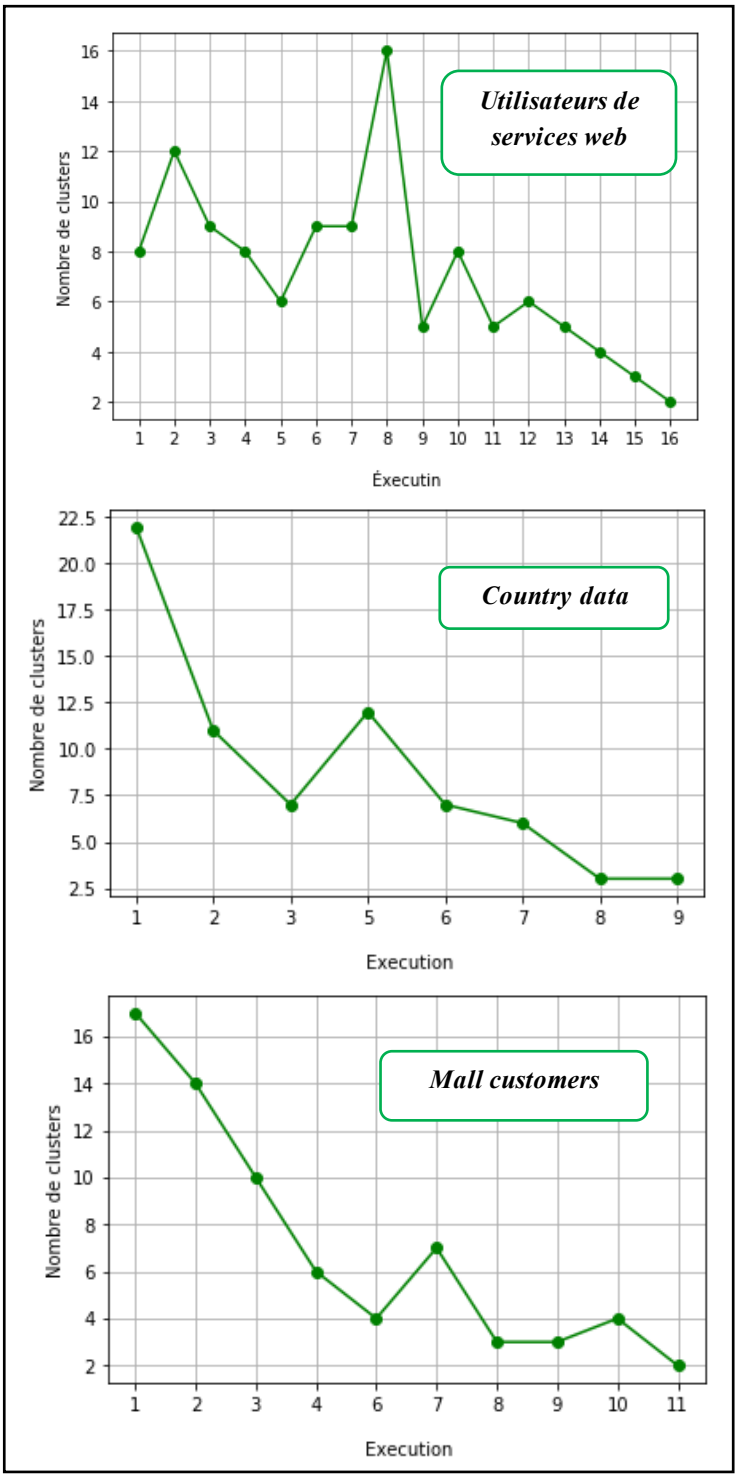


Figure 31 : Le changement du nombre de clusters en fonction de (Eps, MinPts)

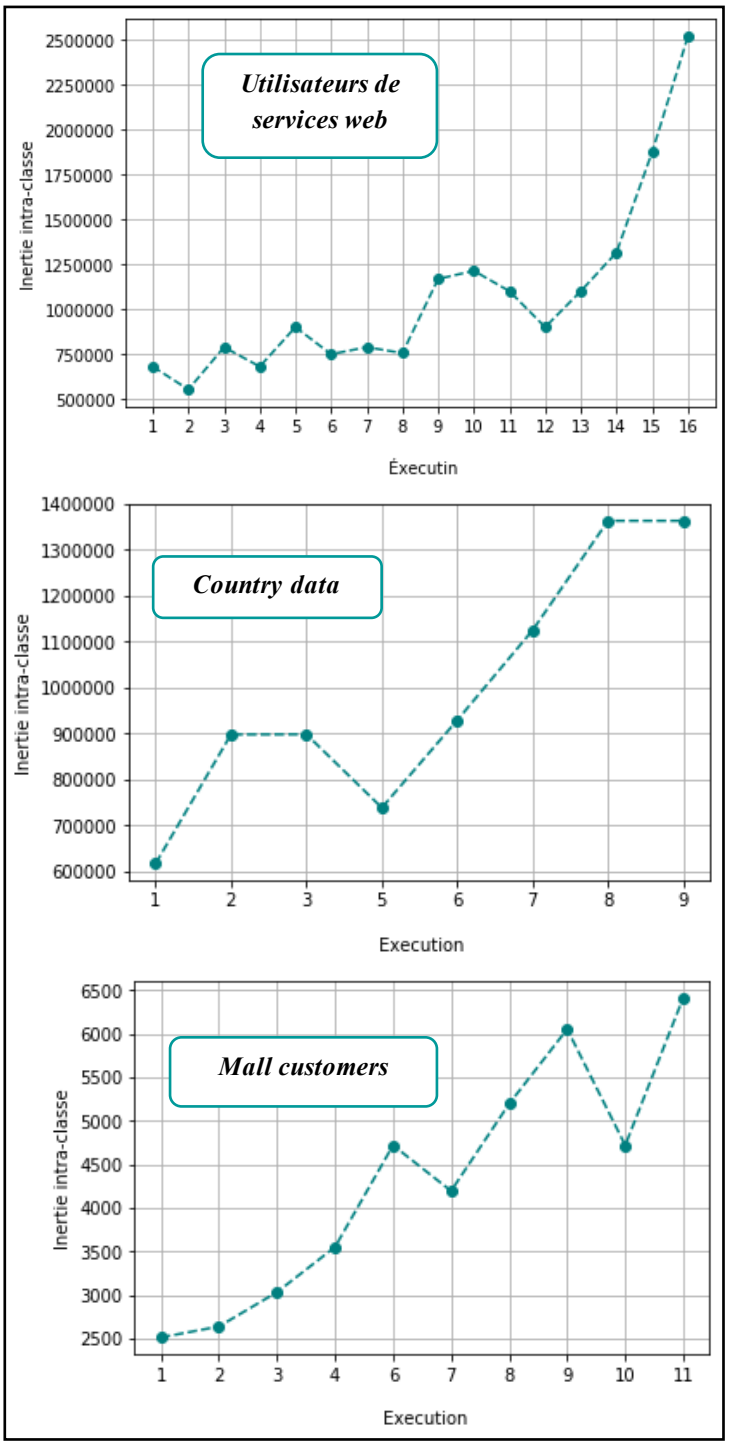


Figure 32 : Le changement de l'inertie intra-classe en fonction de (Eps, MinPts)

Pour les valeurs de l'inertie inta-classe, nous observant qu'en général plus les valeurs de Eps et MinPts sont petites, plus la valeur de l'inertie intra-classe et faible, et vice versa. Ainsi que l'inertie intra-classe atteint la valeur maximale lorsque les valeurs de Eps et MinPts sont grandes.

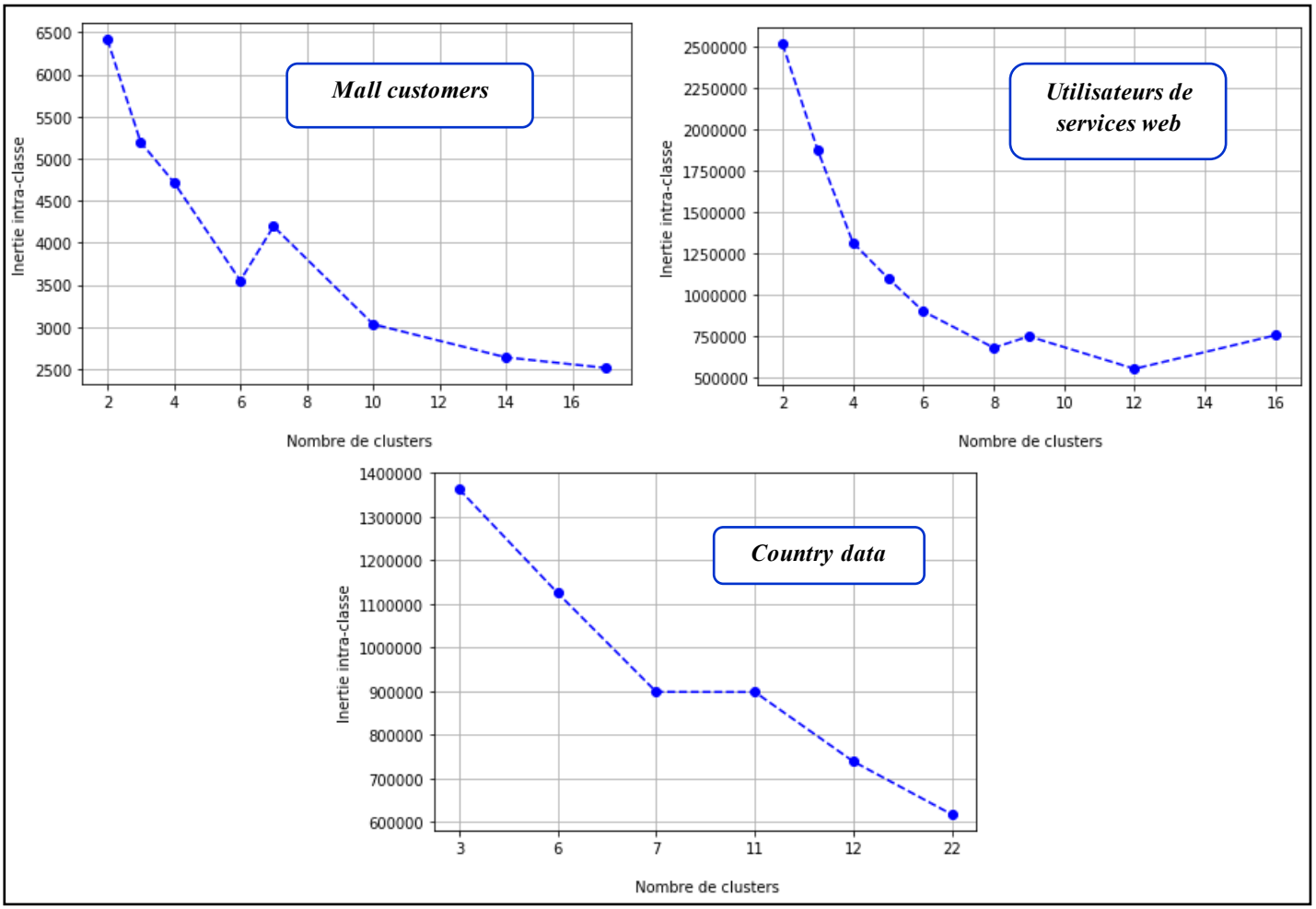


Figure 33 : Le changement de l'inertie intra-classe en fonction du nombre de clusters

En comparant ces graphes, pour tous les ensembles de données utilisés nous observons que plus le nombre de clusters est petit plus la valeur de l'inertie intra-classe est élevée, et vice versa. Tel qu'avec 2 clusters l'inertie intra-classe atteint le seuil maximum.

6. Étude comparative :

Pour comparer les résultats obtenus de l'implémentation de notre méthode ***AggloDens*** avec d'autres des méthodes existantes (ici nous avons appliqué l'algorithme de Lance Williams et le DBSCAN sur les bases de données précédentes, puis nous avons utilisé les résultats comme références), nous avons basé sur quatre choses :

- L'inertie intra-classe.
- L'équilibrage du répartissement.
- Le nombre d'itérations et le temps d'exécution.

1- Commençons par ***l'inertie intra-classe***, nous avons la calculé pour les résultats des deux techniques, ***AggloDens*** et ***CHA***. Voici un tableau comparant les valeurs de l'inertie intra-classe des deux techniques :

Base de données	Nombre de clusters	L'inertie intra-classe	
		AggloDens	CHA
Utilisateurs des services web	4	***1315225.79***	1396710.61
	5	***1099230.92***	1101461.49
	6	***898821.77***	914476.83
	8	***679045.49***	700154.56
	9	747872.17	641017.07
	12	551256.37	488653.91
Country data	3	***1363260.41***	1490565.29
	6	***1125198.3***	1286396.7
	7	***898573.57***	1215815.79
	11	898250.86	761670.71
	12	738850.07	536306.95

Tableau 5 : Comparaison des valeurs de l'inertie intra-classe résultantes des deux méthodes AggloDens et CHA

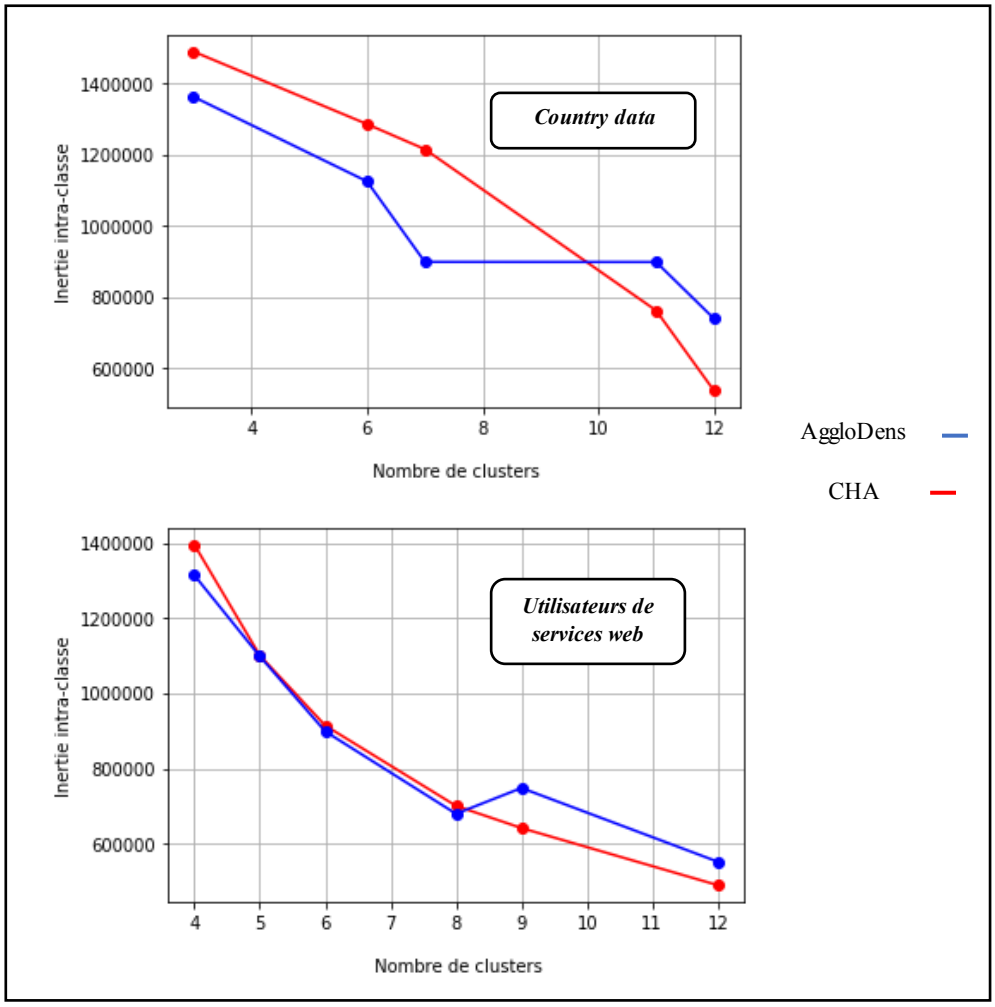


Figure 34 : Comparaison des valeurs de l'inertie intra-classe résultantes des deux méthodes AggloDens et CHA

Nous avons trouvé dans la plupart des exécutions que l'inertie intra-classe résultante de la technique ***AggloDens*** est plus petite par rapport à celle de la technique CHA. Et comme dit auparavant, l'objectif du clustering est de baisser le maximum l'inertie intra-classe.

2- Ainsi qu'avec la méthode hybride AggloDens, nous avons obtenu un clustering plus équilibré par rapport aux autres méthodes. Nous avons exécuté l'algorithme de Lance Williams et AggloDens sur l'ensemble de données (***Country data***) plusieurs fois, le tableau ci-dessous montre les tailles des clusters résultants par les deux algorithmes :

Nombre de clusters	***AggloDens***		***Lance Williams***
	La taille des clusters exigée (MinPts)	***La taille des clusters résultants***	***La taille des clusters résultants***
3	38	C1=64, C2=64, C3=39	C1=4, C2=34, C3=129
6	16	C1=16, C2=32, C3=32, C4=32, C5=32, C6=23	C1=31, C2=2, C3=129, C4=1, C5=3, C6=1
7	12	C1=4, C2=12, C3=32, C4=32, C5=32, C6=32, C7=23	C1=129, C2=2, C3=18, C4=1, C5=3, C6=1, C7=13
11	12	C1=16, C2=16, C3=16, C4=16, C5=16, C6=16, C7=16, C8=16, C9=15, C10=12, C11=12	C1=112, C2=17, C3=13, C4=2, C5=3, C6=1, C7=3, C8=1, C9=8, C10=6, C11=1

Tableau 6 : La taille des clusters résultants de l'application de l'algorithme AggloDens et Lance Williams sur l'ensemble « Country data »

Nous observons ici que les clusters résultants de la méthode AggloDens sont plus équilibrés (mieux répartis), par exemple avec $k = 3$ (où k est le nombre de clusters), la taille des trois clusters est proche (64, 64, et 39). Tandis que les clusters résultants de l'algorithme de Lance Williams sont mal repartis, où avec $k = 3$. Il y a des clusters qui contiennent une seule instance, tandis que d'autre contient 129 instances.

3- D'autre part, en utilisant la méthode CHA, le nombre d'itération est assez grand par rapport à celui de la méthode AggloDens. À chaque exécution nous parcourons toute la matrice de proximité et nous extrayons une liste de distance pour construire plusieurs clusters à la fois avec la méthode AggloDens. Par contre avec la méthode CHA nous extrayons la distance minimale pour former un seul cluster, donc à chaque fois nous parcourons un ensemble de données large pour extraire une seule valeur, ce qui augmente le nombre d'itération donc augmente le temps d'exécution. Voici un tableau

résumant les résultats de l'exécution de quelques modèles avec les deux algorithmes AggloDens et Lance Williams sur un ensemble de 200 instances :

Nombre de clusters	AggloDens		Lance Williams	
	Nombre d'itérations	Temps d'exécution (en seconde)	Nombre d'itérations	Temps d'exécution (en seconde)
2	***7***	***36,82***	198	414.45
4	***6***	***35,8***	196	254,23
7	***5***	***35,95***	193	403.03
8	***5***	***34,33***	192	255,72
9	***5***	***33,17***	191	389,24

Tableau 7 : Le nombre d'itérations et le temps d'exécution résultants de l'application de l'algorithme AggloDens et Lance Williams sur un ensemble de 200 instances.

7. Conclusion :

D'après notre expérimentation, nous pouvons conclure que :

- Pour avoir un bon clustering en appliquant cette méthode, il faut connaitre comment bien choisir les valeurs des paramètres exigés (Eps et MinPts).

- Plus la taille des clusters (MinPts) est petite et la distance de voisinage (Eps) est proche des distances les plus fréquentes, meilleur est le clustering résultant.

- Notre méthode ***AggloDens*** résulte de bon clustering grâce à un répartissement de données équilibré, une basse inertie intra-classe et moins d'itération ce qui signifie un gain de temps.

Conclusion générale et perspective

Conclusion générale et perspective :

La classification automatique non supervisée (***Clustering***) des données est un concept important qui fait partie du processus d'analyse de données. C'est également l'une des tâches majeures dans le domaine de la science de données. Elle vise à découvrir la structure intrinsèque d'un ensemble d'objets en format de regroupement homogènes significatifs appelés clusters, et qui partagent des caractéristiques similaires. Pour réaliser ce processus, diverses techniques ont été proposées et appliquées.

Au cours de ce projet, nous avons proposé une nouvelle approche hybride de clustering (***AggloDens***). L'objectif principal était d'adapter et généraliser l'utilisation des méthodes existantes combinées. Ainsi que d'améliorer la qualité des résultats et minimiser le temps d'exécution, en se basant essentiellement sur deux grandes techniques de base de clustering, la ***technique par densité*** et celle ***hiérarchique agglomérative***.

Afin de mieux implémenter et tester l'efficacité de notre méthode proposée, nous avons utilisé trois différents ensembles de données, un ensemble numérique, un ensemble binaire-numérique et un ensemble multi-type. Les valeurs de l'inertie intra-classe basses, les différents répartissements équilibrés et le temps d'exécution optimisé résultants montrent l'avantage majeur de cette approche hybride. Ainsi que son efficacité avec différent type et taille de données.

Comme perspective à ce travail, il est possible d'ajouter au futur la notion de ***densité-connectivité*** à cette méthode pour avoir des clusters plus denses est plus similaires. Aussi appliquer la méthode de Ward pour la mise à jour de la matrice de proximité, car c'est une méthode qui minimise le maximum la valeur de l'inertie intra-classe.

Bibliographies :

[1] Guillaume Cleuziou, Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information, décembre 2004, Université d'Orléans.

[2] Guénaël CABANES, Classification non supervisée à deux niveaux guidés par le voisinage et la densité, 03/12/10, Université Paris 13.

[3] Thèse, Ballstering: un algorithme de clustering dédié à de grands échantillons.

[4] Bounneche Meriem Dorsaf, réduction de données pour le traitement d'images, 2009, Université Mentouri Constantine.

[5] M. H. C. Law, A. P. Topchy and A. K. Jain, "Multiobjective Data Clustering",In the proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol 2, pp 424-430, 2004.

[6] M. Halkidi, Y. Batistakis, M. Vazirgiannis, "On Clustering Validation Techniques", Intelligent Information Systems Journal, Kluwer Pulishers, vol.17, n°2-3, pp. 107-145, 2001.

[7] A.K. Jain et R.C. Dubes. "Algorithms for Clustering Data", Prentice Hall advanced reference series,1988.

[8] M. A. NEMMICH. Algorithme de Clustering Spectral pour la mémoire de Master Saida, Département d Algorithme de Clustering Spectral pour la Classification Textuelle. Université de SAIDA, 2014.

[9] C. RAMDANE. Le clustering des données : une nouvelle approche évolutionnaire quantique. Uniersité de constantine, 2006.

[10] Nicoleta ROGOVSCHI, Classification à base de modèles de mélanges topologiques des données catégorielles et continues, Université Paris 13 - Insitut Galilée Laboratoire d'Informatique de Paris Nord UMR 7030 du CNRS.

[11] Spath H. Clustering algorithms. John Wiley and Sons, New York, NY, 1975.

[12] D. Arbia, " Métaheuristiques appliqués à la classification non supervisée de données", Université Mohamed Boudiaf M'Sila, 2019.

[13] P. Berkhin, "Survey of Clustering Data Mining Techniques", Technical report, Accrue software, San Jose,California, 2002.

[14] A.K. Jain, M.N. Murty et P.J. Flynn. "Data Clustering: A Review", ACM Computing Surveys, Vol. 31, No. 3,pp 264- 323 , 1999.

[15] R. RAKOTOMALALA, ‘’Algorithme des k-médoïdes, Classification automatique, Méthode de réallocation’’, Université Lumière Lyon 2.

[16] C. RAMDANE. Le clustering des données : une nouvelle approche évolutionnaire quantique. Université de constantine, 2006.

[17] M. Halkidi, Y. Batistakis, M. Vazirgiannis, “On Clustering Validation Techniques”, Intelligent Information Systems Journal, Kluwer Pulishers, vol.17, n°2-3, pp. 107-145, 2001.

[18] Jalam, Radwan. (2003) "Apprentissage automatique et catégorisation de textes multilingues". Thèse de doctorat, Université Lumière Lyon 2.

[19] Jalam, Radwan. (2003) "Apprentissage automatique et catégorisation de textes multilingues". Thèse de doctorat, Université Lumière Lyon 2.

[19] R.J.Wilson et J.J.Watkins. "Graphs : an introductory approach : a first course in discrete mathematics. John Wily and Sons, New York, NY,1990

[20] Une méthode de classification non-supervisée pour l’apprentissage de règles et la recherche d’informatique. Guillaume Cleuziou. 2006

[21] G.Lance et W.Williams. "A general theory of classification sorting strategies". Computer Journal, 9, 373-386, 1967

[22] M. Halkidi, Y. Batistakis, M. Vazirgiannis, “On Clustering Validation Techniques”, Intelligent Information Systems Journal, Kluwer Pulishers, vol.17, n°2-3, pp. 107-145, 2001

[23] Sylvain Grassin & Pauline Regnault, Classification par une méthode de Clustering basée sur la densité, Département d’Enseignement et de Recherche en Informatique & Technologies Urbaines, Ecole d’ingénieurs de la ville de Paris

[24] RAMDANE CHAFIKA Le clustering des données : une nouvelle approche évolutionnaire quantique-2006

[25] S.B. Kotsiantis, P. E. Pintelas, "Recent Advances in Clustering: A Brief Survey", WSEAS Transactions on Information Science and Applications, Vol 1(1), pp.73–81,2004

[26] Cours de clusternig des données. Nicolas PASQUIER. Université NICE Sophia-Antipolis

[27] Sylvain Grassin & Pauline Regnault, Classification par une méthode de Clustering basée sur la densité, Département d‟Enseignement et de Recherche en Informatique & Technologies Urbaines, Ecole d‟ingénieurs de la ville de Paris.

[28] R. RAKOTOMALALA, ‘’Algorithme des k-médoïdes, Classification automatique, Méthode de réallocation’’, Université Lumière Lyon 2.

[29] https://docs.python.org/fr/3/tutorial/

[30] https://www.spyder-ide.org/

[31] https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python , 29-03-2020

[32] https://www.kaggle.com/rohan0301/unsupervised-learning-on-country-data , 02-08-2020