Résumé DM

Le Data mining désigne la méthode qui extrait et exploite de la connaissance ou des modèles réels à partir de grandes quantités de données. Les étapes de ce processus sont : *Data cleaning, Data integration, Data selection, Data transformation, Data mining algorithms, Pattern evaluation, Knowledge presentation.*

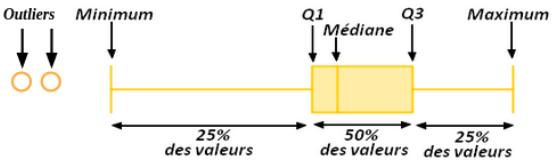
1- Analyse et nettoyage des données

Pour un ensemble de n données $X = x_1, x_2, \dots, x_n$. La première étape est d'analyser les données afin de procéder à l'élimination et/ou transformation de certaines données.

Note: Il est nécessaire de commencer par ordonner l'ensemble de données avant de calculer certaines des mesures présentées ci-dessous. C'est pourquoi il est préférable de commencer par cette étape de manière automatique.

1.1- Analyse des données

Moyenne : $\overline{x}$	$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$
Médiane : (Med ou Q2)	La valeur se situant au milieu de l'ensemble de données ordonné. (N+1/2) ème élément si N est impaire. moyenne de (N/2) et (N/2)+1 ème élément si N est pair.
Mode	La valeur de l'ensemble de données ayant la plus grande fréquence.
Type de modalité	Unimodal, bimodal, trimodal, … dépend du nombre de Mode que possède l'ensemble de données.
Symétrie des données	Moyenne≃Médiane≃Mode : Symétrique Moyenne > Médiane >Mode : Positivement Moyenne < Médiane < Mode : Négativement Pour les données unimodales légèrement asymétriques on a: mean - mode ≈ 3 * (mean - median)
midrange (milieu de gamme)	Moyenne de la valeur la plus grande et la plus petite de l'ensemble de données. $\frac{min + max}{2}$
étendue	Différence entre la plus grande et la plus petite valeur de l'ensemble de données. $max - min$
Quartile Q1 (méthode simple)	Le 1er quartile est la $(\frac{n}{4})$ème valeur de l'ensemble de

	données ordonné. (25% des données sont inférieur à Q1)
Quartile Q3 (méthode simple)	Le 3ème quartile est la $(\frac{3 \cdot n}{4})$ème valeur de l'ensemble de données ordonné. (75% des données sont inférieur à Q3)
Résumé des 5 nombres	(MIN, Q1, Q2, Q3, MAX)
Écart interquartile : IQR	IQR = Q3 - Q1
Boxplot **(Boite à moustache)**	
Outliers (Valeurs aberrantes)	Les valeurs > $Q3 \ + \ 1.5 \ * \ IQR$ Les valeurs < $Q1 \ - \ 1.5 \ * \ IQR$
Scatter plot **Nuage de points**	Il permet d'identifier toute particularité de la forme de la distribution des données, qui peut être symétrique ou inclinée vers des valeurs supérieures ou inférieures. (trouver une relation entre 2 variables de natures différente)
Quantile	Les quantiles sont des points pris à intervalles réguliers dans une distribution de données, la divisant en ensembles consécutifs de taille essentiellement égale.
Quantile plot	permet d'évaluer à la fois le comportement global et les occurrences inhabituelles. Représentation des x_i ordonnées en fonction f_i (%), tel que : $f_i = \frac{i - 0.5}{n}$ (i est le range de la valeur X_i)
quantile-quantile plot	Il permet de comparer deux distributions de deux sources de données S1 (axe des x) et S2 (axe des y). Une ligne (y = x) peut être ajoutée, les points situés au-dessus de cette ligne indiquent que les valeurs de la source S2 sont plus élevées que celles de S1 et vis versa. Ordonner séparément chaque source de donnée puis représenter par un point l'intersection de chaque paire de valeur de même rang. (Comparaison de 2 variables de même nature).
Standard deviation : σ **(Ecart-type)**	$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$

1.2- Nettoyage des données

<table>
<tr><td rowspan="6">Traiter les valeurs manquantes</td><td colspan="2">ignorer les instances contenant des valeurs manquantes.</td></tr>
<tr><td colspan="2">remplir manuellement les valeurs manquantes.</td></tr>
<tr><td colspan="2">Étiquette “unknown” ou nouvelle classe.</td></tr>
<tr><td colspan="2">Mesure de tendance centrale (moyenne, médiane,..)</td></tr>
<tr><td colspan="2">Moyenne des instances appartenant à la même classe</td></tr>
<tr><td colspan="2">Valeur la plus probable: (Formule bayésienne ou arbre de décision)</td></tr>
<tr><td rowspan="7">Traiter les valeurs aberrantes (outliers) & Corriger les données incohérentes</td><td rowspan="4">Binning</td><td>lisser par bacs de fréquences égales</td></tr>
<tr><td>lisser par bacs de moyenne</td></tr>
<tr><td>lisser par bacs de médiane</td></tr>
<tr><td>lisser par limites d’intervalles des bacs.</td></tr>
<tr><td colspan="2">Régression linéaire ou polynomiale</td></tr>
<tr><td colspan="2">Clustering pour détecter et supprimer les outliers</td></tr>
<tr><td colspan="2">Inspection manuelle ou automatique selon les connaissances sur le domaine.</td></tr>
<tr><td>Éliminer la redondance</td><td>cause : intégration de données de différentes sources / natures</td><td>Solution : vérification informatique + manuelle et suppression d’attribut et/ou d’instances.</td></tr>
</table>

2- Intégration, sélection et transformation des données

La phase d’intégration et de transformation est cruciale afin de combiner les données de plusieurs sources de manière à limiter les redondances et incohérences. Elle regroupe : le lissage des données, agrégation et construction de data cube, normalisation des données, ...

min-max Normalisation	$X' = Min_{new} + \frac{(X - Min_{old})\,(Max_{new} - Min_{new})}{Max_{old} - Min_{old}}$
z-score Normalisation	$X' = \frac{x - \bar{x}}{\sigma}$

<table>
<tr><td rowspan="3">Coefficient de corrélation : τ (de deux variable A et B)

⇒ détection de l'existence d'attributs redondants.</td><td>Données numériques : $\tau_{A,B} = \frac{(\sum_{i=1}^{n} A_i.B_i) - n.\overline{A}.\overline{B}}{(n-1)\,\sigma_A\sigma_B}$</td></tr>
<tr><td>Données en Catégories: avec |A|=n , et |B|=m.
$x^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{(Observed_{ij} - Expected_{ij})^2}{Expected_{ij}}$</td></tr>
<tr><td>τ > 0 : Les deux variables sont positivement corrélées (évoluent de manière proportionnelle).
τ ≃ 0 : Les deux variables ne sont pas corrélées.
τ < 0 : Les deux variables sont négativement corrélées (évoluent de manière inversement proportionnelle).</td></tr>
<tr><td rowspan="5">Réduction des données</td><td>Agrégation des Data cube : faire une projection d'un minimum de dimension nécessaire.</td></tr>
<tr><td>Réduction de la dimensionnalité : éliminer les attributs à faible variance ou Utilisation du coefficient de corrélation.</td></tr>
<tr><td>Compression de données: String/Audio/video compression,</td></tr>
<tr><td>Numerosity reduction : réduire le nombre d'instances.</td></tr>
<tr><td>Discretization and concept hierarchy generation</td></tr>
<tr><td rowspan="3">Méthodes de Discrétisation</td><td>Calcule du nombre d'intervalles à former:
Brooks-Carruthers ⇒ K = 5*log(n,base=10)
Huntsberger ⇒ K = 1 + $\frac{10}{3}$*log(n,base=10)
Sturges ⇒ K = log(n+1,base=2)
Amplitude des intervalles :
w = (max - min + degrée de précision) / k
Intervalles : [min, min+w[, [min+w, min+2w[, … , [min+(k-1)*w, min+k*w[.</td></tr>
<tr><td>Intervalle de fréquence égal : tous les intervalles doivent contenir approximativement le même nombre de valeurs.</td></tr>
<tr><td>ChiMerge Algorithme.</td></tr>
</table>

2- Algorithmes de Data Mining

2.1- Extraction de motifs fréquents, règles d'associations et corrélations

Ce type de technique de Data Mining a pour but de trouver des régularités inhérentes aux données, c'est-à-dire des associations ou relations courantes dans un ensemble de données.

Algorithme : Apriori

Entrée : T : Table de transactions (Matrice) ; Min_{Sup} : le support minimum
Sortie : L: l'ensemble des motifs fréquents;
Var
C_1 , C_2 , … , C_k : Matrice des candidats (itemsets - support)
L, L_1 , L_2 , … , L_k : Tableaux des motifs fréquents (itemsets)

Début
/*Construction des candidats C_1 (1-itemsets)*/
Calculer pour chaque item le nombre d'apparitions (support);

/*Réduction des candidat C_1 en liste de motifs fréquents L_1 */
Pour chaque candidat c **de** C_1 **faire**
Si support(c) ≥ Min_{Sup} Alors $L_1 \leftarrow L_1 \cup c$; FinSi;
Fait;

/*Réitération du processus avec les 2-itemsets, 3-itemsets, … , k-itemsets*/
$k \leftarrow 1$;
Tant que L_k != ϕ**faire**
$L \leftarrow L \cup L_k$; /*sauvegarde de l'ensemble des motifs fréquents*/
$k \leftarrow k + 1$;

/*Générer toutes les combinaisons possibles formant des k-itemsets*/
Générer les candidats C_k à partir de la jointure de L_{k-1} * L_{k-1}

/*Extraction des k-itemsets fréquents : L_k à partir de C_k */
Pour chaque candidat c **de** C_k **faire**
Si support(c) ≥ Min_{Sup} Alors $L_k \leftarrow L_k \cup c$; FinSi;
Fait;

Fait;

retourner L;
Fin.

Algorithme : ECLAT

Entrée : T : Table de transactions (Matrice) ; Min_{Sup} : le support minimum
Sortie : L: l'ensemble des motifs fréquents;
Var
C_1 , C_2 , … , C_k : Matrice des candidats (itemsets - support)
L, L_1 , L_2 , … , L_k : Tableaux des motifs fréquents (itemsets)
tableVerticale : matrice booléenne.

Début
tableVerticale ← Représentation verticale des données de transaction

/*Construction des candidats C_1 (1-itemsets)*/
Calculer le support de chaque itemset à partir de *tableVerticale* ;

/*Réduction des candidat C_1 en liste de motifs fréquents L_1 */
Pour chaque candidat c **de** C_1 **faire**
Si support(c) ≥ Min_{Sup} Alors L_1 ← L_1 ∪ c ; FinSi;
Fait;

/*Réitération du processus avec les 2-itemsets, 3-itemsets, … , k-itemsets*/
k ← 1 ;
Tant que L_k != ϕ **faire**
L ← L ∪ L_k ; /*sauvegarde de l'ensemble des motifs fréquents*/
k ← k + 1;

/*Générer toutes les combinaisons possibles formant des k-itemsets*/
Générer les candidats C_k à partir de la jointure de L_{k-1} * L_{k-1}
/*Calculer pour chaque itemset de C_k le nombre de transaction en commun*/
Calculer le nombre de 1 résultant de l'intersection des items le composant;

/*Extraction des k-itemsets fréquents : L_k à partir de C_k */
Pour chaque candidat c **de** C_k **faire**
Si support(c) ≥ Min_{Sup} Alors L_k ← L_k ∪ c ; FinSi;
Fait;

Fait;

retourner L;
Fin.

TID	Itemsets
1	a,b,c
2	a,b
3	a,c,d
4	a,b,d
5	a,c

représentation
⇒
verticale

Itemsets	TID set				
	1	2	3	4	5
a	1	1	1	1	1
b	1	1	0	1	0
c	1	0	1	0	1
d	0	0	1	1	0

Algorithme : FP-Growth

Entrée : T : Table de transactions (Matrice) ; Min_{Sup} : le support minimum
Sortie : L: l'ensemble des motifs fréquents;
Var
L_1 : Tableaux des items fréquents (1-itemsets);
T' : Table de transaction ordonnée;
ListePath : liste de chemins;
CFPT : liste des Conditional FP-Tree;

Début
L_1 ← 1-itemset fréquents(T);
Sorte(L_1); //ordre décroissant de leur support
T' ← Ré-ordonner les items de chaque transaction;
Construction du FP-Tree;

Pour chaque item i de L_1 **faire** //prendre les items dans l'ordre croissant
/*Conditional Pattern Base*/
ListePath ← récupérer le chemin allant de la racine à l'item i;
Marquer chaque chemin par son support; //support de l'item i dans le chemin traité.

/*Conditional FP-Tree*/
CFPT ← Combiner les items ayant des débuts de chemins en commun;
Mettre à jours leur support; //faire la somme
Supprimer les itemsets ayant un support < Min_{sup}

/*Fréquent Patterns Generation*/
L ← Recombiner les différents items de chaque CFPT avec l'item i;
Garder le support minimal de chaque combinaison;
Fait;
L ← L U L_1 ;
retourner L;
Fin.

Calcul de la confidence d'une règle d'association (A ⇒ B) :

$$\textbf{Confidence (A} \Rightarrow \textbf{B) = P(B/A) = } \frac{Support\ (A\ U\ B)}{Support\ (A)}$$

2.2- Classification et Prédiction

Il s'agit des méthodes d'apprentissage supervisé telles que les données d'apprentissage (observations, Exemple) sont pré-étiquetées (label indiquant leur classe). Le modèle de classification est construit sur la base de ces données d'apprentissage. Puis de nouvelles données sont classées à travers le modèle obtenu.

Parmi les techniques de classification les plus connus : Arbres de Décision (ID3, C4.5), k-NN, Forêt d'arbres décisionnels (random forest), Naïve Bayes, Réseaux de Neurones, … etc

2.2.1- Arbres de Décision

Les attributs doivent être catégoriques (Nominaux ou Quantifiées). S'ils sont à valeur continue, ils doivent être préalablement discrétisés.

Les attributs à tester sont sélectionnés sur la base d'une mesure heuristique ou statistique (ex: gain d'informations).

Algorithmes : ID3

```
fonction ID3(dataset, attributCible, attributs)
Début
    Si dataset est vide  /* Nœud terminal */
    Alors retourner un nœud Erreur; //ou bien voir la classe dominante
    Sinon
        Si attributs est vide  /* Nœud terminal */
        Alors retourner un nœud ayant la classe C la plus représentée pour attributCible;
        Sinon
            Si tout le dataset a la même classe C de l'attributCible /* Nœud terminal */
                alors retourner un nœud ayant cette classe C;
                sinon /* Nœud intermédiaire */
                    attributSélectionné = attribut maximisant le gain d'information parmi attributs;
                    attributsRestants = suppressionListe(attributs, attributSélectionné);
                    newNode = nœud étiqueté avec attributSélectionné;

                    pour chaque valeur V de attributSélectionné faire
                        dataFiltrés_V = getAttributValeur(dataset, attributSélectionné, V);
                        newNode->fils(V) = ID3(dataFiltrés_V, attributCible, attributsRestants);
                    finpour

                    retourner newNode;
Fin.
```

Calcule du Gain d'information

L'**entropie** est une mesure de l'incertitude:

$$\text{Entropie}(D) = -\sum_{i=1}^{k} p_i * \log_2(p_i)$$

$p_i = n_i / n$: n_i est le nombre d'instances de la classe i; n est le nombre d'instances du dataset.

Le **Gain d'information** de D relatif à l'attribut A est la réduction d'incertitude prévue grâce au partitionnement selon l'attribut A:

$$\text{Gain(D, A)} = Entropie(D) \; - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \; * \; Entropie(D_v) \right]$$

⇒ choisir l'attribut qui permet de maximiser le gain.

Calcule du Gain Ratio

L'**$SplitInfo_A$** mesure l'intérêt de séparer les données selon l'attribut A:

$$\text{SplitInfo}_A(D) = - \sum_{v \in values(A)} \frac{|D_v|}{|D|} * \log_2\left(\frac{|D_v|}{|D|} \right)$$

$p_i = n_i / n$: n_i est le nombre d'instances de la classe i; n est le nombre d'instances du dataset.

Le **Gain Ratio** de D relatif à l'attribut A est donné par l'équation suivante:

$$\text{GainRatio(D, A)} = \frac{Gain(D,A)}{SplitInfoA(D)}$$

⇒ choisir l'attribut qui permet de maximiser le gain ratio.

2.2.2- Classification Bayésienne naïve

Cette méthode est basée sur le calcul des probabilité. Pour cela il est à noter que :

$P(a_i)$: la probabilité d'avoir la valeur a_i pour l'attribut i.
$P(a_i / b_j)$: la probabilité d'avoir la valeur a_i pour l'attribut i, sachant qu'on a la valeur b_j pour l'attribut j.

Etant donné une instance $X = \{x_1, x_2, \ldots, x_n\}$ à classer. La classification bayésienne naïve cherche à calculer pour chaque classe C_i :
$P(C_i / X)$: la probabilité d'avoir une classe C_i sachant une donnée X .
avec:

$$\mathbf{P(C_i / X) = P(C_i) * P(x_1 / C_i) * P(x_2 / C_i) * P(x_3 / C_i) * \ldots * P(x_n / C_i)}$$

Puis, il faut choisir la classe **C_i** qui maximise cette probabilité **$P(C_i / X)$.**

$$\mathbf{y = argmax_{Ci}\ P(C_i) * \prod_{j=1}^{n} P(x_j / C_i)}$$

Remarque: afin d'éviter le problème de la probabilité zéro, nous pouvons utiliser l'***Estimateur de Laplace.*** Incrémenter de 1 le nombre d'instances de chaque probabilité et le nombre d'instances global.

2.2.3- k-NN: k Nearest Neighbours

Le classificateur des k-plus proches voisins est considéré comme classificateur paresseux, car il s'inspire de la classe des instances voisines. Afin d'estimer le voisinage d'une instance, il est nécessaire de définir une mesure de similarité.

Distance euclidienne:	$d(A , B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$
Distance de Manhattan :	$d(A , B) = \sum_{i=1}^{n} \lvert A_i - B_i \rvert$
Distance de Hamming :	$d(A , B) = \#\{ i : A_i \neq B_i \}$

Algorithme : k-NN

Entrée : *D* : Dataset ; *k* : nombre de voisins à considérer ; *Inst* : l'instance à classifier ;
Sortie : y: la classe de l'instance à classifier
Var
dist : tableau de [1..N] de paire (instance , distance) ; //avec N la taille de D = |D|
knn : tableau de [1..k] d'instances ;
Début

Pour chaque instance X de D **faire**
dist[X] ←Calculer la distance entre *X* et *Inst* ;
Fait;
dist ←Trier dist dans l'ordre croissant des distances; *
knn ←les *k* premières instances de *dist* ; *
y ←La classe dominante dans *knn* ;
retourner y;
Fin.

Note: * *ou bien juste récupérer dans la variable knn directement les k instances ayant une distance minimale.*

2.3- Clustering

Il s'agit des méthodes d'apprentissage non-supervisé telles que les données d'apprentissage (observations, Exemple) ne sont pas pré-étiquetées. La méthode de clustering vise à regrouper ensemble les objets similaires et séparer les objets dissimilaires.

Il existe 4 principales[1] catégories d'algorithmes de clustering comme représentés ci-dessous.

a- Méthodes de partitionnement :

Algorithme : k-means

Entrée : *D* : Dataset ; *k* : le nombre de cluster à former ;
Sortie : D: Dataset étiqueté **;**
Début
  Choisir aléatoirement k instances comme centroïdes.
  Répéter
    Calculer la distance entre chaque instances D[i] et les k centroïdes ;
    Affecter chaque instances D[i] au groupe dont il est le plus proche de son centre;
    Calculer le nouveau centre de chaque cluster et modifier le centroïde ;
  Jusqu'à D[i] identique à D[i-1] ;
  Retourner D ;
Fin.

Algorithme : k-medoids (PAM : Partition Around Medoids)

Entrée : *D* : Dataset ; *k* : le nombre de cluster à former ;
Sortie : D: Dataset étiqueté **;**
Début
  Choisir aléatoirement k instances comme medoids ;
  Répéter
    Assigner chaque instance au medoid le plus proche ;
    Choisir aléatoirement une instance non-medoid D[random];
    Calculer le coût total S de la permutation medoid$_i$ avec D[random] ;
    $S = coutAprèsPermutation - coutAvantPermutation$;
    Si S < 0 **Alors** permuter medoid$_i$ avec D[random] pour mettre à jours les medoids ; **FinSi** ;

[1] Il existe plus de 4 catégories, mais celles-ci sont les plus répondues.

Jusqu'à aucun changement ;
Retourner D ;
Fin.

Formule du coût total : $Cout = \sum_{i=1}^{k} \sum_{p \in C_i} dist(medoid_i, p)$

a- Méthodes hiérarchique:

AGNES (AGglomerative NESting) est une méthode de clustering hiérarchique agglomératif (bottom-up).

Algorithme : AGNES (AGglomerative NESting)

Entrée : *D* : Dataset ;
Sortie : D: Dataset étiqueté **;**
Début
Répéter
Calculer la distance entre chaque pair de clusters avec une méthodes d'agglomération ;
Fusionner la pair de clusters ayant la distance minimale ;
Jusqu'à ce que les données forment un seul cluster ;
Déterminer ou couper la hiérarchie pour obtenir les clusters voulus ;
Retourner D ;
Fin;

Les méthodes d'agglomération :

- Maximum or complete linkage: (Lien maximum ou complet) La valeur maximale de toutes les distances par paires entre les éléments du cluster C1 et les éléments du cluster C2.

$$dist(C1, C2) = Max\,(\,dist(e1, e2)\,,\ e1 \in C1\ et\ e2 \in C2\,).$$

- Minimum or single linkage: (Liaison minimale ou unique) La valeur minimale de toutes les distances par paires entre les éléments du cluster C1 et les éléments du cluster C2.

$$dist(C1, C2) = Min\,(\,dist(e1, e2)\,,\ e1 \in C1\ et\ e2 \in C2\,).$$

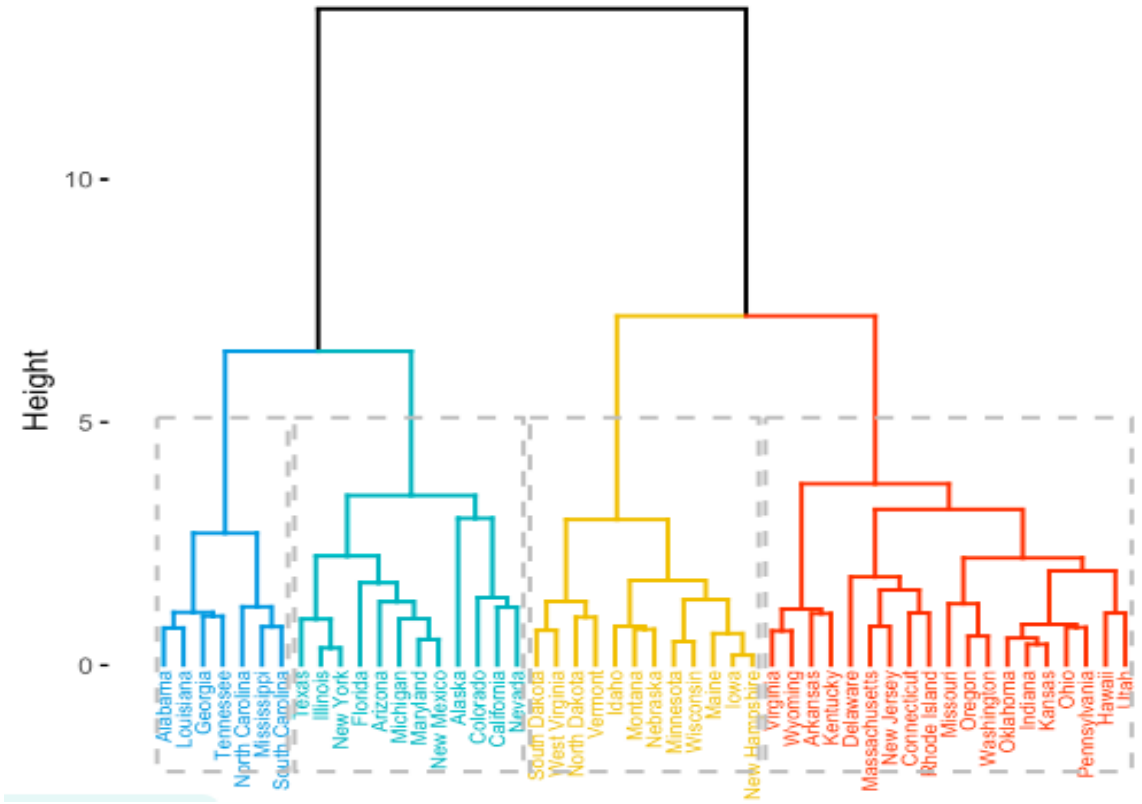
- Mean or average linkage: (Lien moyen) La distance moyenne entre les éléments du cluster C1 et les éléments du cluster C2.

$$dist(C1, C2) = \frac{1}{size(C1) * size(C2)} \sum_{e1 \in C1} \sum_{e2 \in C2} dist(e1, e2).$$

- Centroid linkage: (Lien centroïde) La distance entre deux clusters est définie comme la distance entre le centroïde du cluster C1 et le centroïde du cluster C2.

Représentation:

La représentation graphique du déroulement de l'algorithme AGNES se fait à travers un arbre appelé "*dendrogramme*".



DIANA (DIvisive ANAlysis) est une méthode de clustering hiérarchique divisante (top-down).

Algorithme : DIANA (DIvisive ANAlysis)

1. Les instances sont toutes regroupées au sein d'un même cluster.
2. Dans le clusters C présentant la plus grande dissimilarité entre deux instances, on sépare ces instances en deux clusters A et B.
3. Les instances du cluster C scindée en deux sont affectés à l'un ou l'autre des deux clusters A ou B créées suivant l'algorithme suivant :
   - A est au départ constitué de toutes les instances de C, B est vide.
   - Pour chaque instance i de A, on calcule la dissimilarité moyenne aux autres instances de A. On affecte l'instance x ayant la plus forte dissimilarité moyenne dans le groupe B. On a alors A= A \{x} et B={x}
   - Pour chaque instance de A, on calcule la dissimilarité moyenne à A et à B. L'instance ayant la plus forte différence dist(i,A)-dist(i,B) est affectée au groupe B si la différence est positive sinon l'algorithme s'arrête.

c- Méthodes basées densité:

DBSCAN : *Density-Based Spatial Clustering of Applications with Noise,* est une méthode basée sur la densité des instances. Elle à la particularité de détecter les données bruitées et outliers.

Algorithme : DBSCAN

Entrées : *D*: Dataset, ***eps***: distance minimum de voisinage, ***MinPts***: nombre minimum d'instance dans un cluster;
Sortie : *D*: Dataset étiqueté **;**

```
C = 0 ;
Pour chaque point P non visité du dataset D Faire
    P.status = visité ; //Marquer P comme déjà visité
    PtsVoisins = epsilonVoisinage(D, P, eps) ;
    Si tailleDe(PtsVoisins) < MinPts Alors
        P.cluster = BRUIT; //Ajouter P à la liste des données bruitées
    sinon
        C++ ;//new cluster
        etendreCluster(D, P, PtsVoisins, C, eps, MinPts) ;
    FinSi;
Fait;
Fin.
```

Fonction : etendreCluster

Entrées : *D*: Dataset, ***eps***: distance minimum de voisinage, ***MinPts***: nombre minimum d'instance dans un cluster, ***P***: une instance de D, ***PtsVoisins***: liste des voisins de P, ***C***: numéro du cluster en cours de création ;
Sortie : *D* : Dataset étiqueté **;**

```
Début
    P.cluster = C; // Ajouter P au cluster C
    Pour chaque point P' de PtsVoisins Faire
        Si P'.status <> visité Alors
            P'.status = visité ; // Marquer P' comme visité
            PtsVoisins' = epsilonVoisinage(D, P', eps) ;
            Si tailleDe(PtsVoisins') >= MinPts Alors
                PtsVoisins = PtsVoisins U PtsVoisins' ; // Inclure les voisins des voisins à la liste
            FinSi;
        FinSi;
        Si P'.cluster == 0 Alors //P' n'est membre d'aucun cluster
            P'.cluster = C ; // Ajouter P' au cluster C
        FinSi;
    Fait;
Fin.
```

Note : On appelle le "epsilon Voisinage" d'une instance P toutes les instances de D qui sont à une distance inférieure ou égale à epsilon de P.

Algorithme	k-means	k-medoide	CLARANS	ANGES	DIANA	DBSCAN
Complexité	O(n*k*t)	O(k * $(n-k)^2$ * t)	O(n^2)	O(n^3)	with heuristic O(n^3) without O(2^n)	O(n^2)

Metrics:
Distance intra-cluster : la distance entre les instances d'un même cluster. Mesure à minimiser.
Distance inter-clusters : la distance moyenne entre les différents clusters. Mesure à maximiser.

Remarque : utiliser la distance de Manhattan pour l'EMD.