IFT870/BIN710
Forage de données

Thème 5 : Fonctions descriptives

Aida Ouangraoua
Département d'informatique

Université de Sherbrooke

Fonctions descriptives

- **Rappel : fonctions descriptives**
- **Optimisation d'un modèle de clustering**
- **Comparaison et évaluation de modèles de clustering**

Fonctions descriptives

- **Transformation :** une fonction qui transforme l'ensemble des valeurs d'un attribut en un nouvel ensemble de valeurs

- **Réduction de dimension :** sélection d'attributs ou construction de nouveaux attributs expliquant (séparant) au mieux les données

- **Segmentation (clustering) :** estimation d'une mesure de similarité entre les données, et regroupement des données de sorte que les données similaires sont dans les mêmes groupes, tandis que les données dissimilaires sont dans des groupes différents.

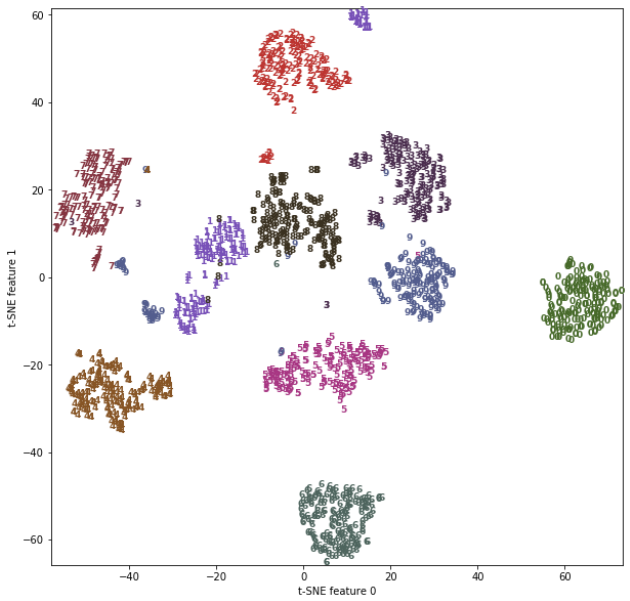
  - Utile pour analyse de la distribution d'un ensemble de données
  - Utile pour l'étape de pré-traitement avant analyse de données
  - Utile pour la détection de données aberrantes

Rappel: quelques algorithmes

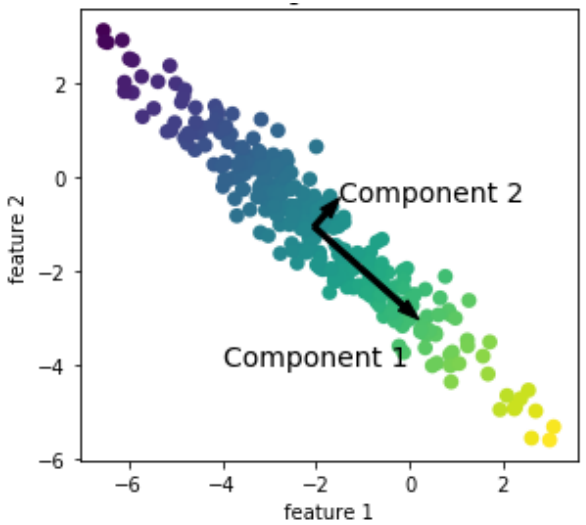
❑ Transformation

- ❖ Transformation non-linéaire
  log, exp, sin, cos
- ❖ Normalisation
  Min-max, Z-score

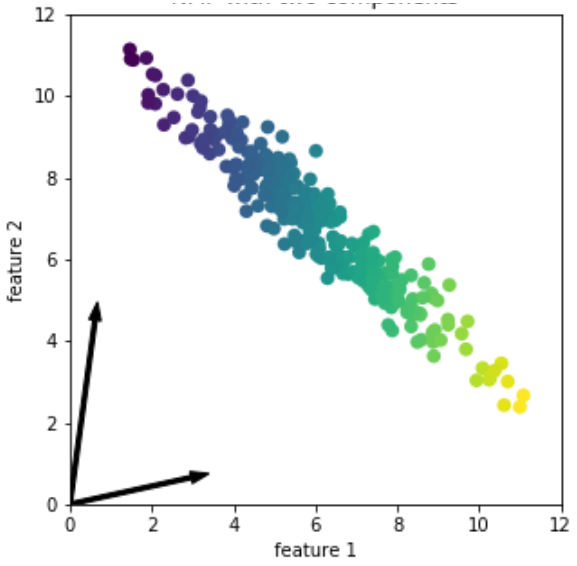
❑ Réduction par Manifold (2 dimensions)



❑ Réduction par décomposition (PCA)

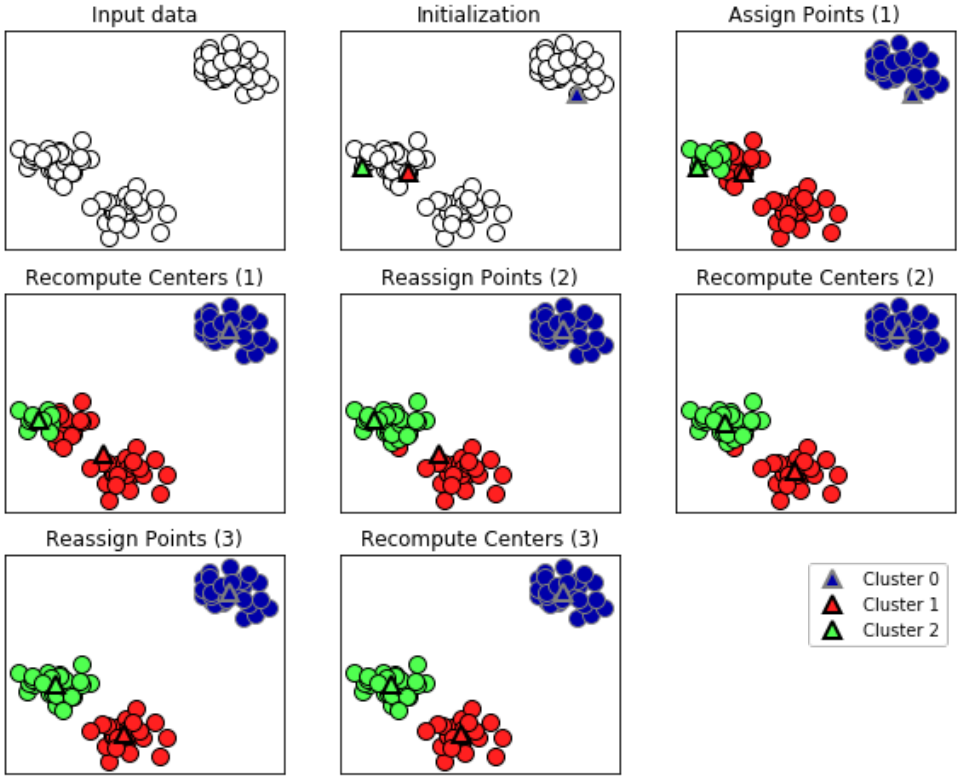


❑ Réduction par décomposition (NMF)

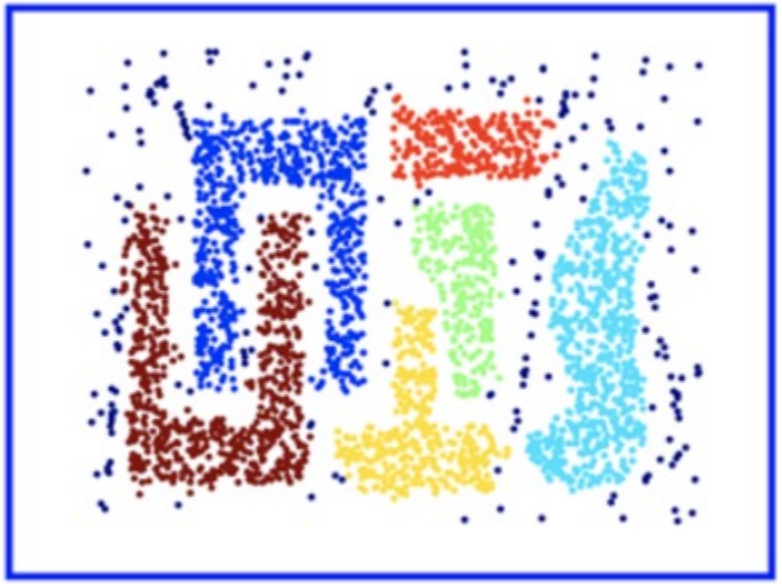


Rappel: quelques algorithmes

- **Clustering par partitionnement: (ex. K-Means (K-Moyennes))**



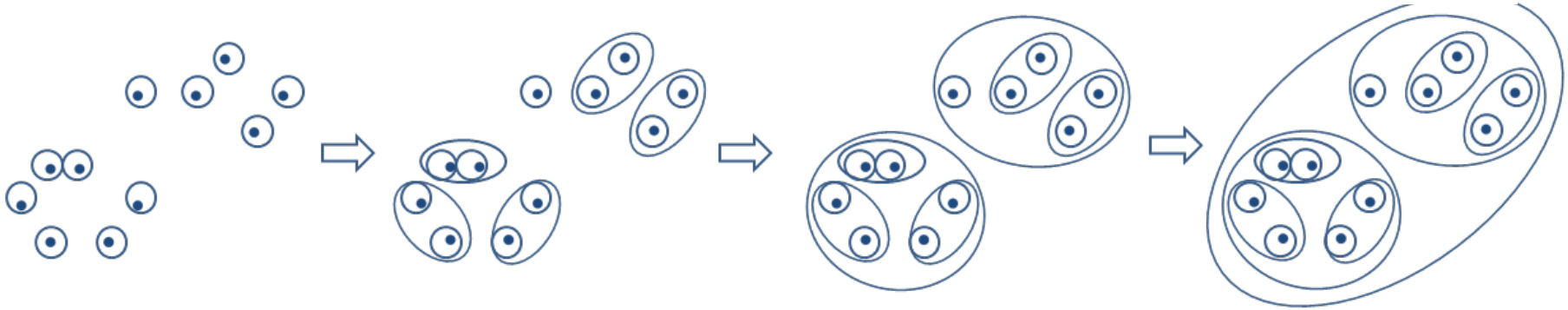
- **Clustering basé sur la densité: (ex. DBSCAN)**



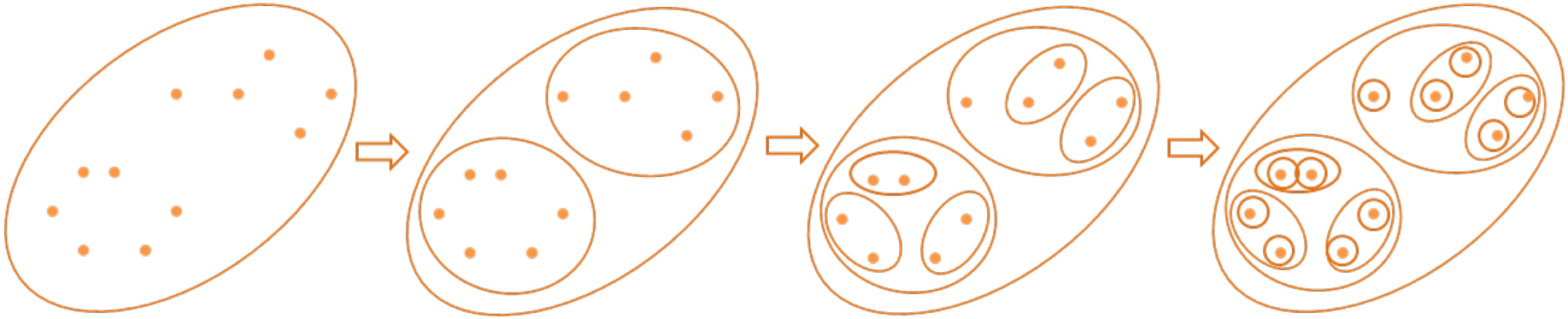
Rappel: quelques algorithmes

- **Clustering hiérarchique**

Méthodes agglomératives (ex. UPGMA, AGNES)



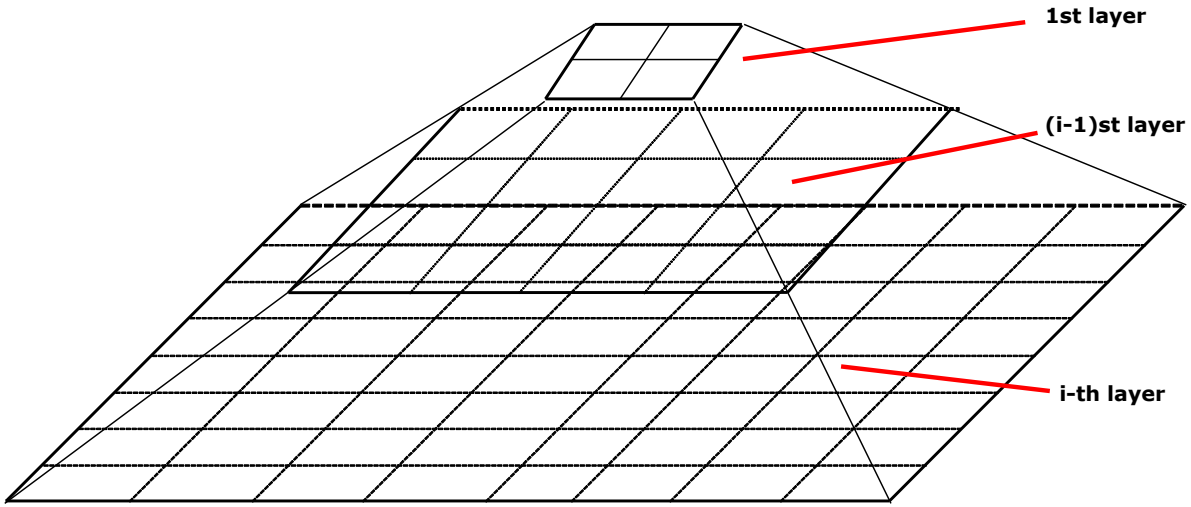
Méthodes divisives (ex. DIANA)



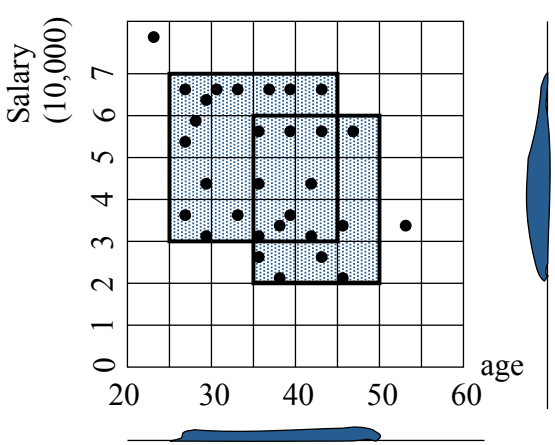
Rappel: quelques algorithmes

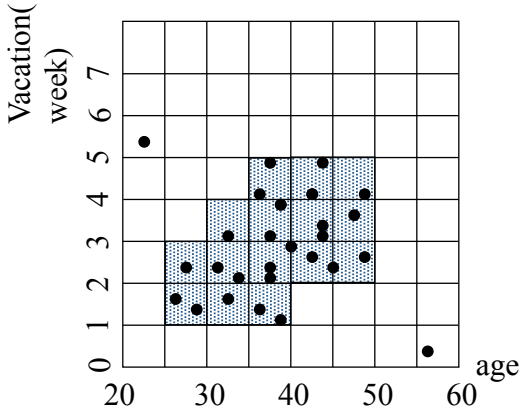
- **Clustering basé sur des grilles : utilisation de structures de données de grille.**

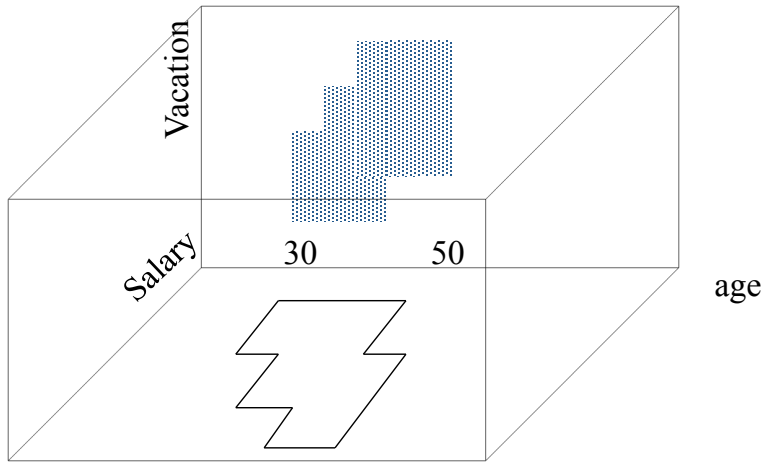
Ex. STING (grille à multiples niveaux de résolution)



Ex. CLIQUE (division de chaque dimension en intervalles)







Fonctions descriptives

- **Rappel : fonctions descriptives**
- **Optimisation d'un modèle de clustering**
- **Comparaison et évaluation de modèles de clustering**

Optimisation d'un modèle de clustering

- **Avant tout, vérifier si les données sont aléatoirement distribuées**
  (si oui, la tendance des données à être segmentées est quasi-nulle)

- **Test statistique de Hopkins:**
  - Comparer les données à des données aléatoires
  - Si les données sont fortement segmentées ➔ valeur ≈ 1
  - Si les données sont aléatoires ➔ valeur ≈ 0.5
  - Si les données sont uniformément distribuées ➔ valeur ≈ 0
  - Étant donné un ensemble de données *D*:
    - *X* = un échantillon de *m* éléments de *D* tirés aléatoirement
      {*x* = *min*{dist(*p*, *q*)} | *p* in *X* et *q* in *D*-*X*}
    - *Y* = un échantillon de m données générés aléatoirement
      {*y* = *min*{dist(*p*, *q*)} | *p* in *Y* et *q* in *D*}

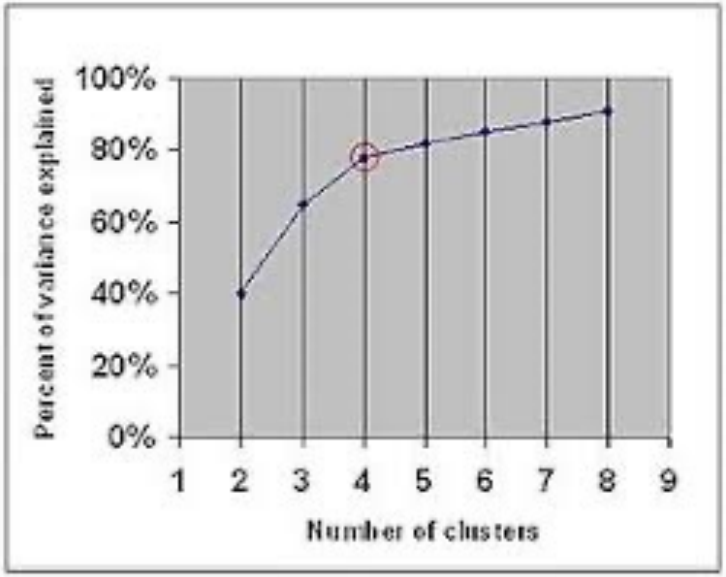
  - Statistique de Hopkins : $$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$

Optimisation d'un modèle de clustering

- **Détermination du nombre de clusters**
  (pour les modèles qui ne sont pas basés sur la densité)

  - **Méthode du coude (Elbow method) :**
    - Faire varier le nombre de clusters (*k*) et générer une courbe de mesure de qualité versus nombre de clusters
    - Exemple de score utilisé:
      Pourcentage de variance expliquée (F-test) =
      variance entre-groupe / variance globale
    - Nombre de clusters optimal (*k_opt*) correspond au point d'angle maximum sur la courbe

    - Méthode subjective et peu fiable

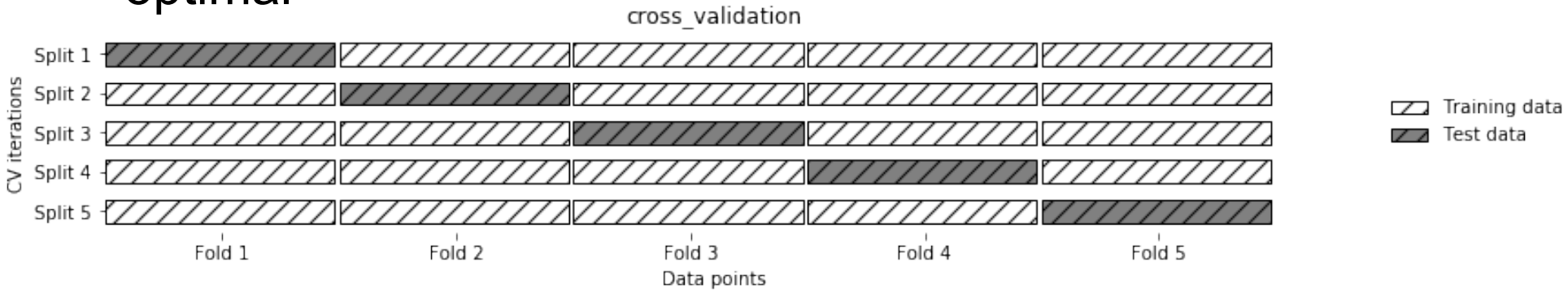


Optimisation d'un modèle de clustering

❑ Détermination du nombre de clusters

❖ Méthode de validation croisée (cross-validation) :

- Diviser les données en *m* parties (*part_1*, *part_2*, … *part_m*)
- Faire varier le nombre de clusters (*k*) et pour chaque *k* répéter pour *i* de *1* à *m* : entraîner et évaluer un modèle en utilisant *part_i* comme données de test et le reste comme données d'entraînement, puis calculer la moyenne des scores pour les *i* itérations.
- Exemple de score à chaque itération : Somme des distances au carré des données de test à leurs plus proches centroïdes
- Nombre de clusters optimal (*k_opt*) correspond au score optimal



Optimisation d'un modèle de clustering

❑ Détermination du nombre de clusters

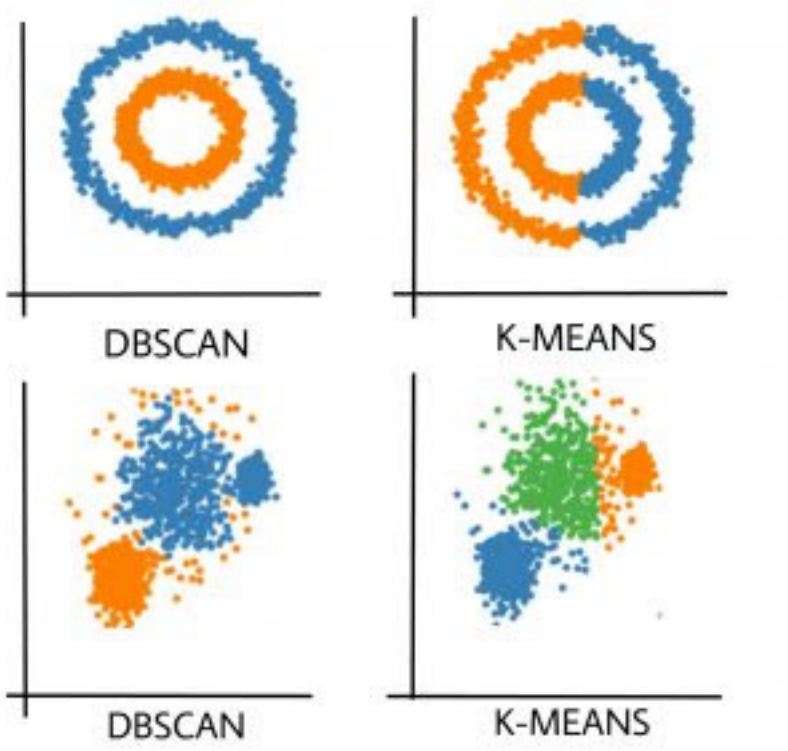
❖ Méthode de la silhouette (Silhouette method) :

- Faire varier le nombre de cluster (*k*) et calculer le coefficient de silhouette pour chaque *k*, puis choisir *k_opt* de coefficient maximum.

- Coefficient de silhouette: évalue la qualité d'un clustering en se basant sur la cohésion (forte similarité à l'intérieur des groupes) et la séparation (faible similarité entre groupes) (voir définition dans partie suivante sur l'évaluation des modèles)

Fonctions descriptives

- **Rappel : fonctions descriptives**
- **Optimisation d'un modèle de clustering**
- **Comparaison et évaluation de modèles de clustering**



Comparaison et évaluation de modèles de clustering

- **Propriété d'un clustering de bonne qualité**
  - **Cohésion** : forte similarité intra-groupe
  - **Séparation :** faible similarité inter-groupe
- **Facteurs influençant la qualité**
  - **Mesure de similarité** utilisée (combinaisons de différentes mesures de similarité/dissimilarité sur différents types d'attributs
    Reflète-elle des aspects de similarité intéressants, cachés ?
- **Évaluation de la qualité: deux cas de figures**
  - Méthodes extrinsèques : avec des données réelles (clustering réel connu)
  - Méthodes intrinsèques : sans données réelles

Méthodes extrinsèques d'évaluation

❑ Critères de qualité

❖ **Homogénéité** : cluster pur (contenant uniquement des données du même groupe)

Catégorie « Autres (Non groupés) » : privilégie l'inclusion d'une donnée dans la catégorie « Autres » par rapport à son inclusion erronée dans un cluster pur.

```
from sklearn.metrics.cluster import homogeneity_score
ari = homogeneity_score (y_test, y_pred)
```

❑ **Complétitude** : cluster complet (contenant toutes les données du même groupe)

Complétitude des petits clusters : pénalise plus la division des petits clusters que la division des grands clusters.

```
from sklearn.metrics.cluster import completeness_score
ari = completeness_score (y_test, y_pred)
```

Méthodes extrinsèques d'évaluation

- **Matrice de confusion :** utile si chaque cluster réel peut être associé à un cluster prédit (Exemple: même noms de clusters)

  Nombre de données dans l'intersections des clusters réels et prédits

Cluster réel\prédit	**C1**	**C2**	**C3**
C1	190	6	4
C2	0	150	0
C3	5	0	145

Homogénéité ?
Complétude ?

Nombre total de données = 200 (C1 réel) + 150 (C2 réel) + 150 (C3 réel) = 500
= 195 (C1 prédit) + 156 (C2 prédit) + 149 (C3 prédit) = 500

```
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(y_test, y_pred)
```

Méthodes extrinsèques d'évaluation

- **Matrice de confusion :** utile si chaque cluster réel peut être associé à un cluster prédit (Exemple: même noms de clusters)

- **Accuracy- Error rate**

Cluster réel\prédit	**C1**	**C2**	**C3**
C1	190	6	4
C2	0	150	0
C3	5	0	145

Nombre total de données
N = 500

Accuracy: pourcentage de données regroupées correctement
Accuracy = (190+150+145)/ 500 = 0.97
Taux d'erreur (Error rate):
Error rate = 1- Accuracy = 0.03

Méthodes extrinsèques d'évaluation

- **Matrice de confusion :** utile si chaque cluster réel peut être associée à un cluster prédit (Exemple: même noms de clusters)

- **Accuracy- Error rate**

Cluster réel\prédit	X1	X2	X3	X4	X5	X6
C1	100	0	0	100	0	0
C2	0	75	0	0	75	0
C3	0	0	75	0	0	75

Nombre total de données
N = 500

Accuracy = (0)/ 500 = 0.0
Error rate = 1.0

Pourtant tous clusters prédits sont purs, même si les clusters initiaux sont divisés en deux

Méthodes extrinsèques d'évaluation

- **Matrice de confusion**

- **Mutual information (*MI*) :** mesure du chevauchement entre deux clustering *C* et *X*

$$MI(C,X) = \sum_{i=1}^{3} \sum_{j=1}^{6} P(Ci,Xj) * log(P(Ci, Xj) / P(Ci) * P(Xj))$$

P(*x*, *y*) = probabilité qu'une donnée soit dans l'intersection des clusters *x* et *y*

P(*x*) = probabilité qu'une donnée soit dans le cluster *x*

Cluster réel\prédit	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**
C1	100	0	0	100	0	0
C2	0	75	0	0	75	0
C3	0	0	75	0	0	75

Nombre total de données N = 500

MI(*C*,*X*) = 100/500 * *log*((100/500) / ((200/500) * (100/500)) + …

Méthodes extrinsèques d'évaluation

- **Matrice de confusion**

- **Mutual information (*MI*) :** mesure du chevauchement entre deux clustering *C* et *X*

$$MI(C,X)=\sum_{i=1}^{3}\sum_{j=1}^{6} P(Ci,Xj) * log(P(Ci, Xj) / P(Ci) * P(Xj))$$

- **Normalized Mutual Information (*NMI*) : «** Mutual information » normalisé par l'entropie des deux clusterings

$$NMI(C,X)= MI(C,X) / ((H(C) + H(X))/2) \quad ;$$

$$H(C) = -\sum_{i=1}^{3} P(Ci) * log(P(Ci)$$

Méthodes extrinsèques d'évaluation

- **Matrice de confusion**

- **Mutual information (*MI*) :** mesure du chevauchement entre deux clustering *C* et *X*

$$MI(C,X)=\sum_{i=1}^{3}\sum_{j=1}^{6} P(Ci,Xj) * log(P(Ci, Xj) / P(Ci) * P(Xj))$$

- **Adjusted Mutual Information (*AMI*) : «** Mutual information » corrigé pour tenir de la similarité attendue par chance pour toute comparaison de clusterings générés par un modèle aléatoire.

$$AMI(C,X)= (MI(C,X) – E\{MI(C,X)\}) / (max\{H(C),H(X)\} – E\{MI(C,X)\}$$

$E\{MI(C,X)\}$: *MI* attendu pour deux clusterings aléatoires

Méthodes extrinsèques d'évaluation

- **Matrice de confusion**

- **Mutual information (*MI*) :** mesure du chevauchement entre deux clustering *C* et *X*

$$\sum_{i=1}^{3}\sum_{j=1}^{6} P(Ci,Xj) * log(P(Ci, Xj) / P(Ci) * P(Xj))$$

- **Adjusted Mutual Information (*AMI*) : «** Mutual information » corrigé pour tenir de la similarité attendue par chance pour toute comparaison de clusterings générés par un modèle aléatoire.

```
from sklearn.metrics.cluster import adjusted_mutual_info_score
ami = adjusted_mutual_info_score (y_test, y_pred)
```

Méthodes extrinsèques d'évaluation

❑ Matrice de contingence :

m11 : nombre de paires de données dans le même cluster pour clustering réel (C) et prédit (P)
m00 : nombre de paires de données dans des clusters différents pour C et P
m10 : nombre de paires de données de même cluster pouur C et de clusters différents pour P
m01 : cas symmétrique à m10

Cluster réel\prédit	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**
C1	100	0	0	100	0	0
C2	0	75	0	0	75	0
C3	0	0	75	0	0	75

Nombre total de données
N = 500

réel\prédit	**Même cluster**	**Clusters différents**
Même cluster	m11 = 21000	m10 = 21250
Cluster différents	m01 = 0	m00 = 82500

Nombre total de paires =
N*(N-1)/2 = (500*499)/2 =
124 750

```
from sklearn.metrics import contingency_matrix
contingency = contingency_matrix(y_test, y_pred)
```

Méthodes extrinsèques d'évaluation

- **Matrice de contingence :**

- **Précision: m11/ (m11 + m01) = 21000 / 21000 = 1**
  mesure l'homogénéité (si tous les clusters prédits sont purs, prec = 1)

- **Recall: m11/ (m11 + m10) = 21000 / 42250 = 0.497**
  mesure la complétude (si tous les clusters prédits sont complets, recall = 1)

- **F-score (Moyenne harmonique) : 2*Precision*Recall / Precision + Recall = 0.66**

réel\prédit	**Même classe**	**Classes différentes**
Même classe	m11 = 21000	m10 = 21250
Classes différentes	m01 = 0	m00 = 82500

Nombre total de paires =
N*(N-1)/2 = (500*499)/2 = 124 750

Méthodes extrinsèques d'évaluation

- **Matrice de contingence :**

- **Coefficient de Jaccard :**
  m11 / (m11 + m01 + m10) = 21000 / 42250 = 0.497
  accorde une importance à l'homogénéité et à la complétude, et néglige les paires de données de classes différentes

- **Rand index :**
  (m11 + m00) / (m11 + m01 + m10 + m00) = 103500/124750 = 0,829
  accorde une importance à l'homogénéité et la complétude

réel\prédit	**Même classe**	**Classes différentes**
Même classe	m11 = 21000	m10 = 21250
Classes différentes	m01 = 0	m00 = 82500

Nombre total de paires =
N*(N-1)/2 = (500*499)/2 = 124 750

Méthodes extrinsèques d'évaluation

- **Matrice de contingence :**

- **Adjusted Rand Index (ARI) : «** Rand index » corrigé pour tenir de la similarité attendue par chance pour toute comparaison de clusterings générés par un modèle aléatoire.

```
from sklearn.metrics.cluster import adjusted_rand_score
ari = adjusted_rand_score (y_test, y_pred)
```

Méthodes intrinsèques d'évaluation (sans données réelles)

- **Nécessité**
  - En pratique, pas de données réelles pour évaluation
  - Méthodes extrinsèques utiles pour le développement de méthodes
  - Méthodes intrinsèques nécessaires pour évaluer les applications (si le clustering était connu, alors on pourrait utiliser la classification)

- **Critères de qualité**
  - **Cohésion (Compactness)** : similarité entre les données du même cluster
    (une faible variation intra-cluster correspond à une bonne cohésion)
  - **Séparation** : dissimilarité entre différents clusters (une grande distance entre les clusters correspond à une bonne séparation)
  - **Connectivity (Connectivity)** : données regroupées avec leurs plus proches voisins (des clusters qui sont des figures fermées dans l'espace correspondent à une bonne connectivité)

Méthodes intrinsèques d'évaluation

❑ Coefficient de Silhouette (Silhouette score)

- ❖ Mesure pour chaque donnée, à quel point elle est proche des données dans le même cluster (cohésion), et distante des données dans les clusters voisins (séparation).

- ❖ Pour toute donnée i, **S(*i*) = (b*i* – a*i*)/ max(a*i*, b*i*)** tel que **a*i*** est la distance moyenne de *i* aux données du même cluster que *i*, et **b*i*** est la distance moyenne minimum de i aux données d'un cluster différent.
  - Si S(*i*) ≈ 1, S(*i*) est très bien placé
  - Si S(*i*) ≈ 0, S(*i*) est entre deux clusters
  - Si S(*i*) ≈ -1, S(*i*) est mal placé

```
from sklearn.metrics.cluster import silhouette_score
ss = silhouette_score(X, y_pred)
```

Méthodes intrinsèques d'évaluation

❑ Score de Davies-Bouldin

- ❖ Mesure la similarité moyenne entre chaque cluster et son plus proche cluster voisin. Le score minimum est 0, et S ≈ 0 correspond à un bon clustering.

```
from sklearn.metrics import davies_bouldin_score
dbs = davies_bouldin_score(X, y_pred)
```

❑ Index de Dunn

- ❖ Combinaison de la cohésion et la séparation calculée comme suit: **DI = (min_sep/max_cohes)** tel que min_sep est la plus petite distance entre deux données placées dans des clusters différent, et max_cohes est la distance maximum entre deux données placées dans le même cluster. Un DI maximum correspond à un bon clustering.

Autres Méthodes intrinsèques d'évaluation

- **Évaluation de la robustesse du modèle**
  - Ajouter du bruit aux données ou faire varier les paramètres du modèle et comparer les résultats. Un modèle robuste aux légères perturbations permet d'avoir plus confiance aux résultats.

- **Analyse manuelle de la signification sémantique**
  - Toutes les méthodes d'évaluation intrinsèques dépendent fortement de la mesure de distance utilisée pour comparer les données.
  - Un bon score d'une méthode d'évaluation intrinsèque ne signifie pas que le clustering calculé a une signification sémantique utile ou non-triviale (cachée). Une analyse manuelle du résultat pour déterminer les caractéristiques communes et spécifiques aux objets de chaque classe est toujours nécessaire.

Références

[1] PEDREGOSA et al. : *Scikit-learn : Machine Learning in Python*. JMLR 12, pp. 2825-2830. (User guide and API : https ://scikit-learn.org/stable/), 2011.

[2] Jiawei HAN, Micheline KAMBER, Jian PEI. *DataMining: Concepts and Techniques (Third edition)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[3] Adreas C. MÜLLER et Sarah GUIDO : *Introduction to Machine Learning with Python*. O'Reilly Media, Inc., Sebastopol, CA, 2017.

[4] Pang-Ning TAN, Michael STEINBACH et Vipin KUMAR : *Introduction to Data Mining, (Second Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2018.