



DATA SCIENCE AVEC PYTHON

Année académique 2024-2025

Master: X-MAS-DE-4

Enseignant : David Rhenals

Clause de confidentialité : Ce cours est à usage unique des étudiants de l'EFREI, la diffusion externe de tout contenu de ce cours est strictement interdite sans l'autorisation écrite préalable de l'enseignant.

Techniques de machine learning pour la modélisation de données

- Qu'est ce que le Machine Learning
- Classification des techniques de Machine Learning
 - Apprentissage supervisé
 - Apprentissage non supervisé
- Principales bibliothèques disponibles sur python (Scikit-Learn – Pycaret)
- L'avant entraînement des modèles de Machine Learning
 - Prétraitement des données avant entraînement (Nettoyage de données, Imputation, Recodage (variables qualitatives))
 - Quelques exemples d'application de prétraitement en python avec scikit-learn
- Apprentissage Supervisé
 - Synthèse de techniques d'apprentissage supervisé
 - Méthodes de rééchantillonnage et synthèse de métriques d'évaluation de performance (Validation croisée, Bootstrap)
 - Modèles de régression et classification (Arbres de régression, Random-Forest, KNN)
- Apprentissage non supervisé
 - Classification hiérarchique (classification automatique)
 - Clustering k-means
- Exemples pratiques d'utilisation de Scikit-Learn et PyCaret (Arbres de Régression, Random Forest)
- Travaux pratiques sur quelques techniques de Machine Learning.

Qu'est ce que le Machine Learning?




- **Le Machine Learning**, ou apprentissage automatique en français, est un sous-domaine de l'intelligence artificielle (IA) qui se concentre sur le développement d'algorithmes capables d'apprendre à partir des données. Contrairement aux systèmes programmés explicitement pour accomplir une tâche précise, les algorithmes de Machine Learning utilisent des données historiques pour **détecter des motifs, générer des prédictions, ou prendre des décisions** sans être explicitement programmés pour chaque situation.
- **Composants essentiels du Machine Learning :**
 - **Données** : Ensemble d'exemples (souvent sous forme de tableaux, textes, images, etc.) utilisé pour entraîner un modèle.
 - **Modèle** : Représentation mathématique générée par un algorithme d'apprentissage à partir des données pour faire des prédictions ou des classifications.
 - **Algorithme d'apprentissage** : Méthode ou processus utilisé pour entraîner un modèle sur des données. Exemples : régression linéaire, arbres de décision, réseaux de neurones.
 - **Métrique, Fonction de perte (ou d'erreur)** : Mesure utilisée pour évaluer la performance d'un modèle pendant son entraînement. L'objectif est de minimiser cette fonction pour améliorer la précision du modèle.
 - **Généralisation** : Capacité du modèle à bien performer sur de nouvelles données qu'il n'a pas vues pendant l'entraînement.

Classification des techniques de Machine Learning

- Apprentissage supervisé : L'algorithme apprend à partir de données étiquetées. Chaque entrée est associée à une étiquette ou une sortie (ex. : classification, régression)
- Apprentissage non supervisé : L'algorithme apprend à partir de données non étiquetées et doit identifier des structures ou des motifs (ex. : clustering, réduction de dimensionnalité).
- Apprentissage par renforcement : L'algorithme apprend en interagissant avec un environnement, recevant des récompenses ou des punitions en fonction des actions qu'il entreprend (ex. : jeu, contrôle de robots).

Principales bibliothèques disponibles sur python (Scikit-Learn – Pycaret)

Tableau introductif aux bibliothèques de Machine Learning Python (Scikit-Learn – Pycaret)

Type de tableau	Lien html
Modules et fonctionnalités Scikit-Learn (Documentation)	
Modules et fonctionnalités Pycaret	
Comparatif Scikit-Learn - Pycaret	

L'avant entraînement des modèles de Machine Learning

- Le prétraitement des données est une étape cruciale dans le développement de modèles de Machine Learning. Elle consiste à préparer les données brutes afin de les rendre adaptées à l'apprentissage automatique.
- Les principales objectifs de la phase de prétraitement :
 - Nettoyage des données
 - Permet d'éliminer les données manquantes qui peuvent fausser les résultats. Les imputer (remplacer) ou les supprimer permet d'avoir un jeu de données complet
 - Permet d'identifier et de corriger les erreurs de saisie, les doublons ou les incohérences afin de garantir l'intégrité des données
 - Transformations nécessaires (mise à l'échelle des variables numériques et encodage des variables catégorielles). La sélection de variables pourrait également faire partie de cette phase, même si certains modèles permettent d'obtenir l'importance des variables dans les prédictions de la variable cible.
 - Permet l'entraînement des estimateurs et peut améliorer les performances du modèle

L'avant entraînement des modèles de Machine Learning

■ Nettoyage des Données

■ Gestion de données manquantes

- Imputation moyenne, médiane, valeur constante ou KNN pour des variables numériques
- Imputation pour la mode (valeur la plus fréquente dans la variable) ou valeur constante pour les variables catégorielles
- Suppression des valeurs manquantes (à utiliser avec attention!)
- Utilisation de modèles prédictifs (prédire les valeurs manquantes)





■ Suppression de doublons (réduction du nombre des lignes à traiter pour le modèle)

■ Transformation des Données

- Normalisation: Mise à l'échelle des variables prédictives dans une intervalle spécifique (Généralement dans l'intervalle $[0, 1]$)
- Standardisation: Transformation des variables prédictives à des nouvelles variables centrées-réduites (variables supposées normalement distribuées $\mu = 0$ et $\sigma = 1$).
- Encodage des variables catégorielles
 - Encodage One-Hot : Pour chaque catégorie de la variable, celui-ci crée une colonne binaire
 - Encodage Ordinal : Attribue des entiers sur chaque catégorie basé dans la relation d'ordre de la variable catégorielle.

Quelques exemples d'application de prétraitement en python avec scikit-learn

Tableau d'exemples introductifs au pré-traitement avec scikit-learn

Exemple	Lien html
Synthèse de méthodes de transformation de données pendant la phase de pré-traitement	
Imputation de données numériques et catégorielles ordinales et nominales	
Encodage de variables catégorielles nominales et ordinales	
Normalisation de variables numériques	

Apprentissage Supervisé (Synthèse de techniques de régression et classification)

Tableau de synthèse des principales techniques de machine learning supervisé

Méthode	Description	Type de problème
Régression linéaire	Modèle prédictif qui suppose une relation linéaire entre les variables indépendantes (features) et la variable dépendante (target). Utilisé pour prédire des valeurs continues.	Régression
Régression logistique	Utilisé pour les problèmes de classification binaire. Modélise la probabilité qu'une observation appartienne à une classe, avec une sortie entre 0 et 1.	Classification
Machines à vecteurs de support (SVM)	Cherche à trouver une hyperplan qui sépare les différentes classes avec un maximum de marge. Utilisé pour des problèmes de classification et de régression.	Classification / Régression
Arbres de décision	Utilise une série de questions basées sur les caractéristiques pour prédire une classe ou une valeur. Facile à interpréter, mais susceptible au sur-apprentissage.	Classification / Régression
Forêts aléatoires	Un ensemble d'arbres de décision. Chaque arbre est construit sur un sous-ensemble des données, et la prédiction est obtenue par un vote majoritaire (classification) ou la moyenne (régression).	Classification / Régression
K plus proches voisins (KNN)	Prédit la classe d'un point en fonction des classes des K voisins les plus proches. Non paramétrique et simple à implémenter, mais coûteux en calcul.	Classification / Régression
Naive Bayes	Basé sur le théorème de Bayes et l'hypothèse d'indépendance entre les caractéristiques. Utilisé principalement pour la classification de texte ou les données catégorielles.	Classification
Réseaux de neurones (perceptron)	Modèle inspiré du cerveau humain, composé de couches de neurones. Utilisé pour des tâches complexes de classification et de régression. Peut être très puissant mais nécessite beaucoup de données.	Classification / Régression
Gradient Boosting Machines (GBM)	Technique d'ensemble qui construit des modèles prédictifs en combinant plusieurs modèles faibles (souvent des arbres de décision) de manière itérative, pour corriger les erreurs des précédents.	Classification / Régression
XGBoost	Une implémentation optimisée du gradient boosting, souvent utilisée en compétition de machine learning, offrant une haute performance et un contrôle fin sur les hyperparamètres.	Classification / Régression
LightGBM	Variante de XGBoost, plus rapide, optimisée pour de très grands ensembles de données. Utilise une approche basée sur l'histogramme pour accélérer l'apprentissage.	Classification / Régression
CatBoost	Algorithme de gradient boosting développé par Yandex, spécialement conçu pour gérer efficacement les variables catégorielles.	Classification / Régression

Méthode basée sur des estimations paramétriques

Méthodes basées sur des règles conditionnelles de décision

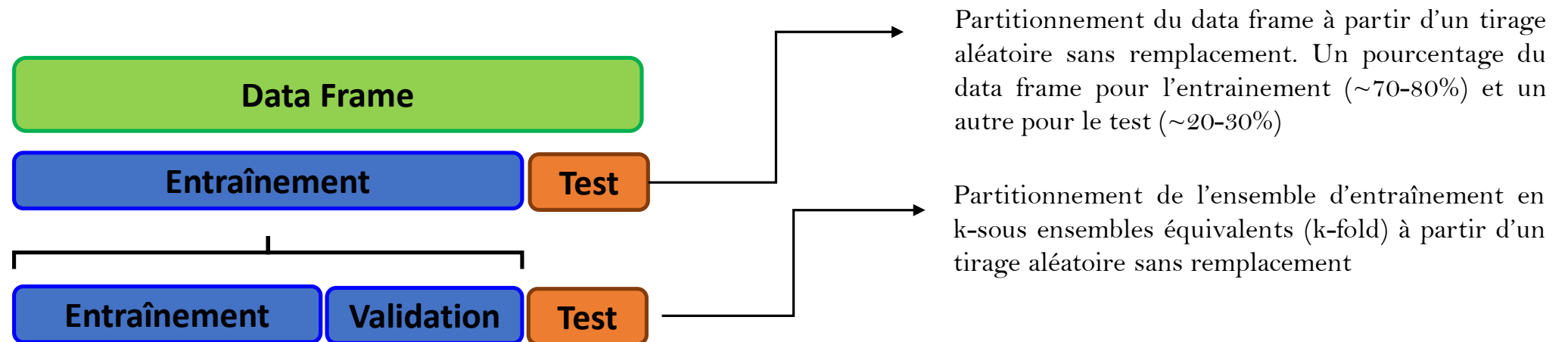
Méthode basée sur la similarité des individus

Techniques de rééchantillonnage

- Les **techniques de rééchantillonnage** (ou **resampling** en anglais) sont des méthodes statistiques qui consistent à manipuler ou à réorganiser un échantillon de données pour estimer la variabilité d'un modèle ou d'un estimateur. Ces techniques sont largement utilisées en statistique, en apprentissage automatique et en analyse de données pour améliorer la précision des résultats ou pour évaluer la stabilité de certains modèles.
- **Ces techniques permettent:**
 - L'estimation de la variabilité d'un estimateur, autrement dit, sa dispersion ou son incertitude (par exemple, les intervalles de confiance d'un estimateur statistique (Bootstrap))
 - L'évaluation de la performance des modèles de Machine Learning de manière plus fiable, et ainsi d'éviter des éventuels problèmes de surapprentissage (validation croisée).
- Dans le contexte du Machine Learning, le bootstrap et la validation croisée sont deux des techniques de rééchantillonnage les plus couramment utilisées pour évaluer et améliorer la performance des modèles.

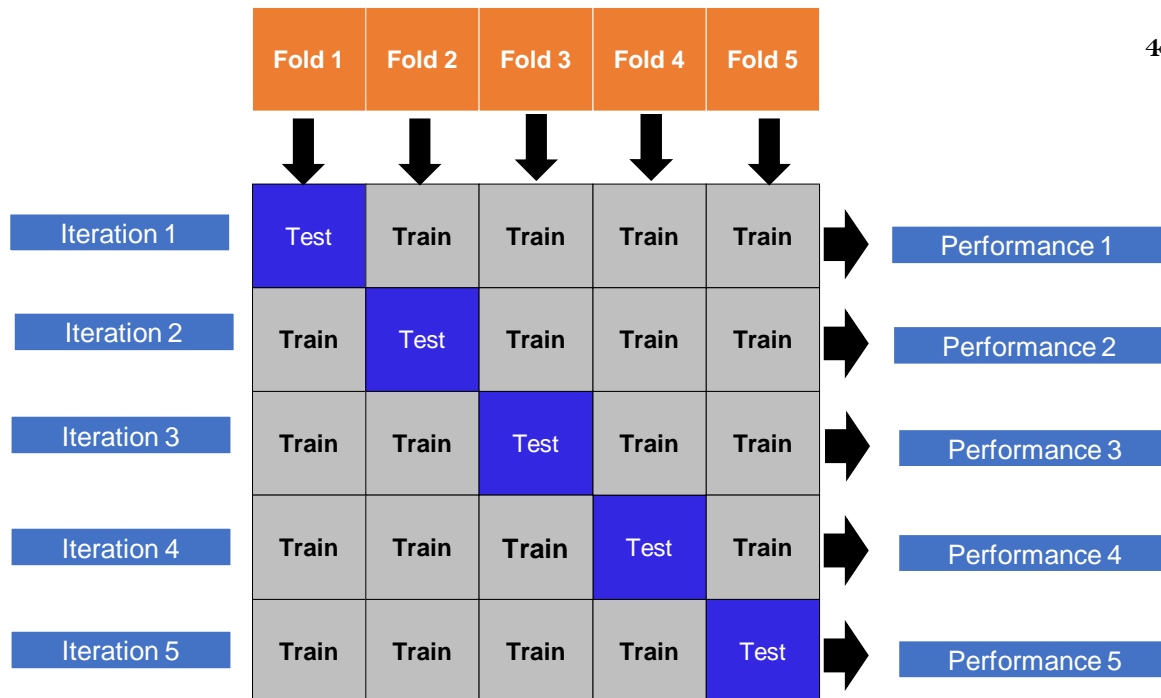
Validation Croisée – Définitions de concepts fondamentaux

- **Ensemble d'entraînement (Training set)** : C'est un sous-ensemble des données utilisées pour entraîner un modèle de Machine Learning. Cet ensemble permet au modèle d'apprendre les relations entre les caractéristiques (variables d'entrée) et les cibles (variables de sortie) afin de pouvoir effectuer des prédictions sur de nouvelles données.
- **Ensemble de test (Test set)** : C'est un sous-ensemble des données qui n'est pas utilisé pendant l'entraînement du modèle. Il sert à évaluer la performance du modèle après qu'il a été entraîné. L'ensemble de test permet de mesurer la capacité du modèle à généraliser sur des données qu'il n'a jamais vues auparavant, ce qui est essentiel pour éviter le surapprentissage (overfitting)



Validation Croisée (k-fold) - Principe de Fonctionnement

1. **Division des données**: L'ensemble de données est partitionnée en k sous-ensembles égaux (ou presque égaux). Par exemple, ci-dessous, nous avons choisis k=5, on obtiendra 5 sous-ensembles.
2. **Entraînement et validation**: Pour chaque itération (il y en a donc k itérations): un sous-ensemble est choisi pour l'utiliser comme ensemble de validation (test). Les k-1 autres sous-ensembles sont utilisés pour entraîner le modèle
3. **Évaluer la performance**: Après chaque itération, la performance du modèle est évaluée sur le sous-ensemble de validation (en calculant des métriques comme la précision, l'accuracy, etc.).
4. **Estimation de la performance globale**: Une fois toutes les itérations terminées, la moyenne des scores obtenus sur chaque sous-ensemble de validation est calculée pour obtenir une estimation globale de la performance du modèle.

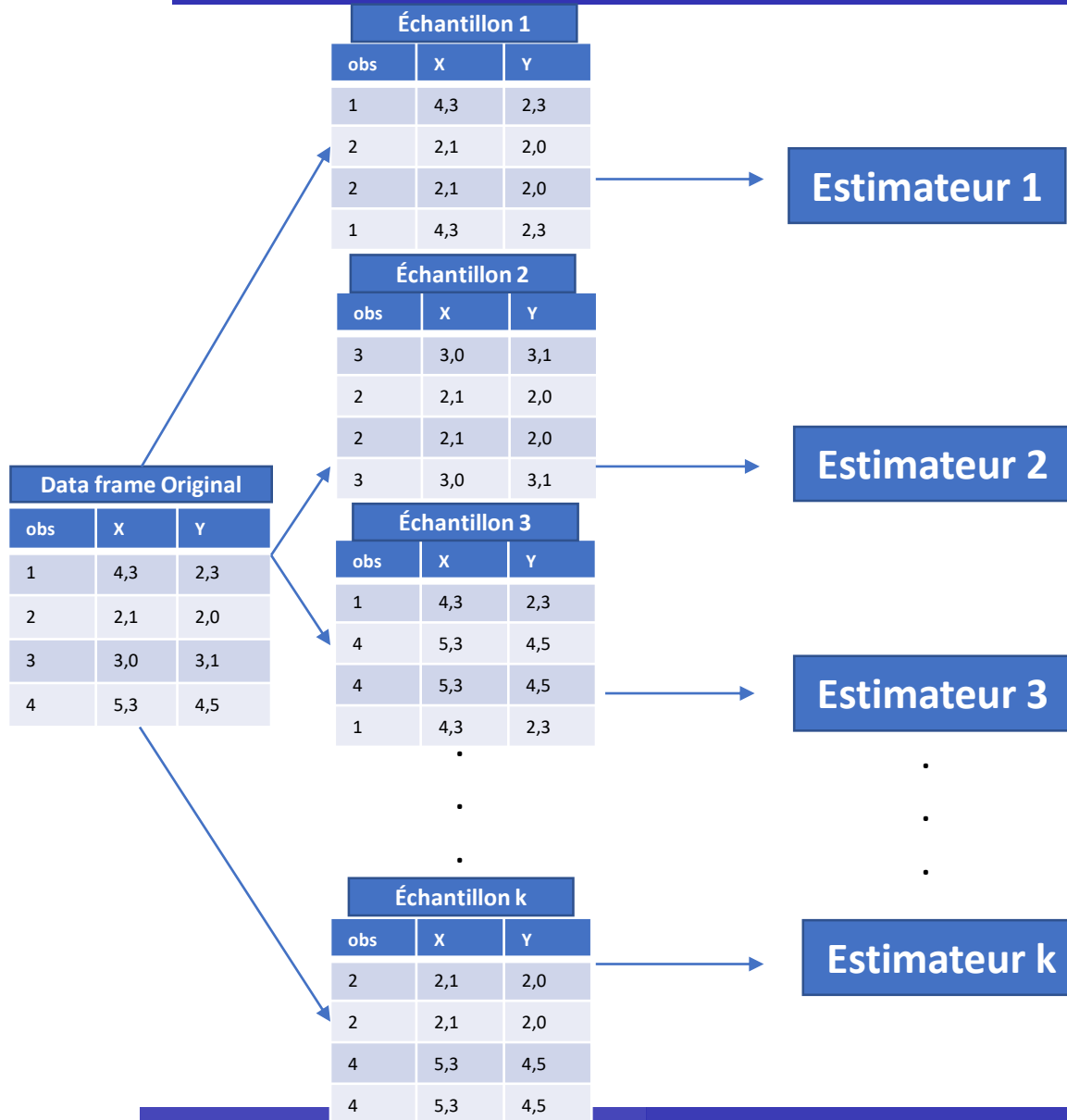


$$Performance\ globale = \frac{1}{n} \sum_{i=1}^n Perf_i$$

Comment mesurer la performance d'un modèle?

Cliquer [ici](#) pour la définition des métriques

Bootstrap – Principe de Fonctionnement



1. Rééchantillonnage avec remplacement :

À partir du data frame original, plusieurs nouveaux échantillons (par exemple, k nouveaux échantillons) sont générés, chacun de taille n.

2. Entraînement du modèle ou calcul de l'estimateur d'intérêt :

Pour chaque sous-ensemble rééchantillonné, l'estimateur d'intérêt est calculé (moyenne, médiane, écart-type, etc.).

3. Distribution de l'estimateur d'intérêt :

Les valeurs des statistiques d'intérêt sont obtenues fournissant une distribution qui permet d'estimer des intervalles de confiance et de quantifier l'incertitude.

Comment mesurer la performance d'un modèle?

Cliquer [ici](#) pour la définition des métriques

Synthèse comparative (k-fold / Bootstrap)

- Quelques différences fondamentales entre le Bootstrap et la validation croisée (k-fold), peuvent être mises en lumière en termes de fonctionnement, d'applications et de coût.

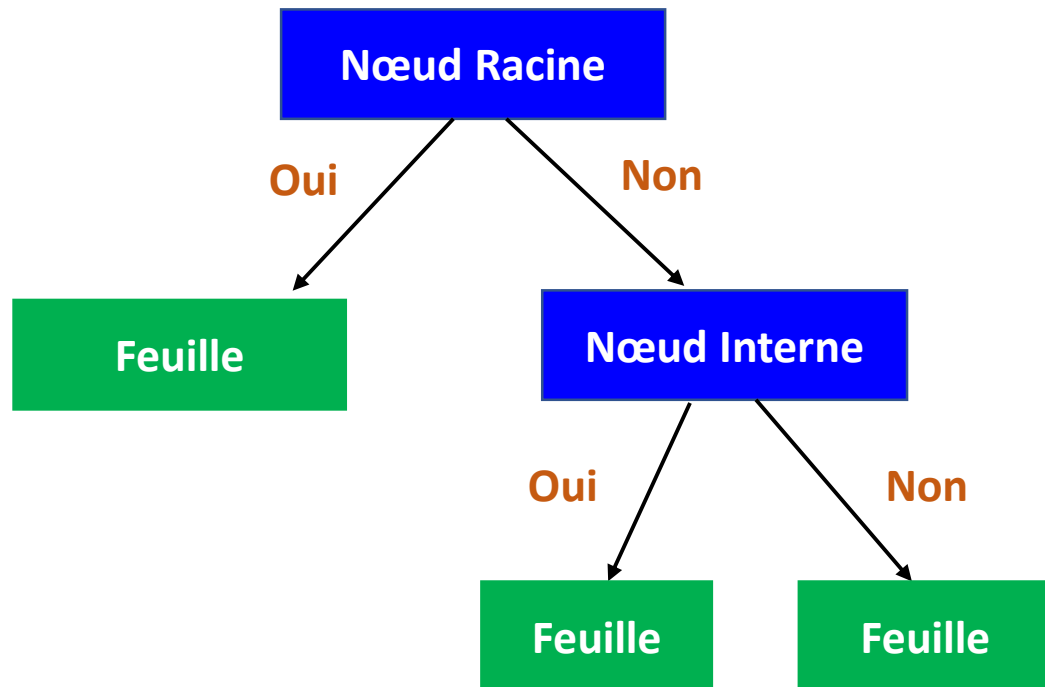
Tableau Comparatif des techniques de rééchantillonnage (Validation croisée (k-fold) - Bootstrap) 11

Critère	Bootstrap	Cross-validation
Principe de rééchantillonnage	Rééchantillonnage avec remplacement (échantillons dupliqués possibles)	Rééchantillonnage sans remplacement (chaque observation appartient à un seul fold)
Nombre d'échantillons dans chaque sous-ensemble	Sous-ensembles d'entraînement de même taille que les données d'origine, avec répétitions possibles	Sous-ensembles d'entraînement excluant un fold, chaque fold étant utilisé une fois pour la validation
Applications typiques	Estimation de l'incertitude, bagging (ex. : Random Forest), évaluation de modèles	Évaluation de la performance générale, tuning des hyperparamètres
Estimation de la performance	Peut sous-estimer l'erreur sur les données de test, car certaines données sont utilisées pour l'entraînement et la validation	Estimation plus robuste, car chaque donnée est utilisée pour l'entraînement et la validation sans chevauchement
Complexité et coût	Moins coûteux en calcul (rééchantillonnage avec remplacement)	Plus coûteux en calcul (formation k fois pour la k-fold cross-validation)

Arbres de décision (CART-Décisions Binaires)

Qu'est ce qu'un arbre de décision: Un arbre de décision est un modèle graphique utilisé pour prendre des décisions en fonction de conditions successives.

Composantes fondamentales d'un arbre de décision



Nœuds : Chaque nœud interne représente une décision basée sur un attribut de la donnée (exemple: Le prix est-il inférieur à 10 000 € ?)

Branches : Représentent les décisions qui mènent aux sous-ensembles de données, illustrant le chemin de décision (exemple: [oui/non.])

Feuilles : Les nœuds terminaux, ou feuilles, indiquent la classe (dans le cas de la classification, exemple: [acheter ou non]) ou la valeur prédite (dans le cas de la régression).

Comment construire un arbre de décision? [cliquer ici](#)

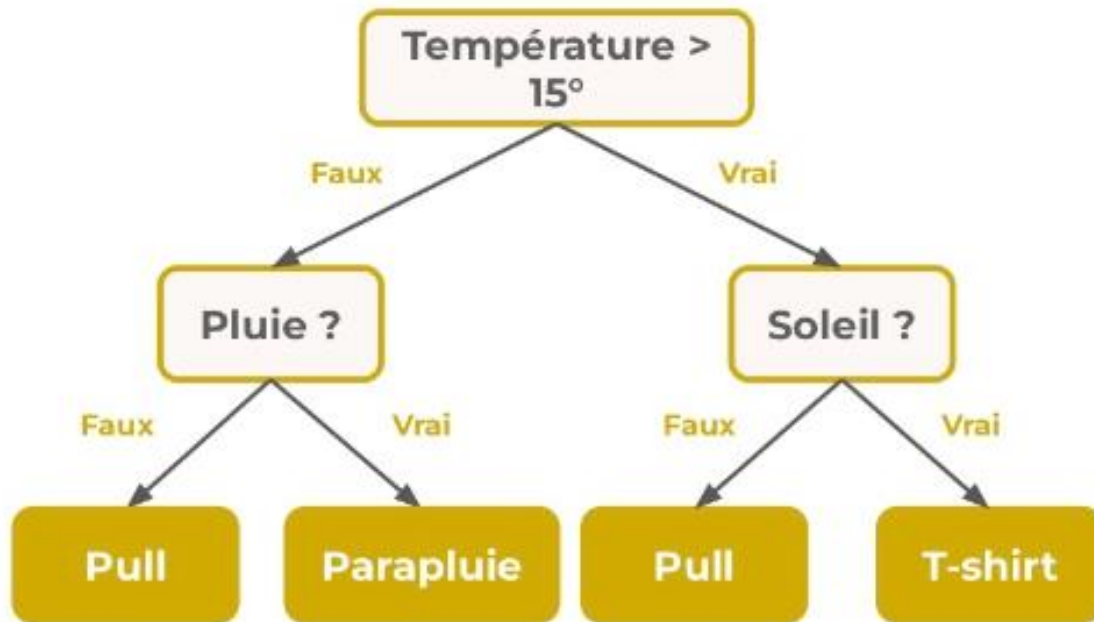
Arbre de décision à travers un exemple

✓ Une personne souhaite savoir comment s'habiller et quels accessoires utiliser en fonction des conditions météorologiques. Nous supposons que, à partir d'un ensemble de données météorologiques simples, nous pouvons entraîner un modèle de prise de décision basé sur l'arbre binaire ci-dessous.

✓ Comment lire l'arbre ?

✓ Comment devrait s'habiller et quel accessoire utiliser sous les conditions suivantes :

C1=(Température=13, Temps=Pluie) , C2=(Température=25, Temps=Pluie), C3=(Température=20, Temps=Soleil)



Lecture conditionnelle de l'arbre

```
if Température > 15:
    if Temps == 'Soleil':
        print('Vous devriez porter un T-shirt')
    else:
        print('Vous devriez porter un Pull')
else:
    if Temps == 'Pluie':
        print('Vous devriez prendre un Parapluie')
    else:
        print('Vous devriez porter un Pull')
```

Prédiction de l'arbre

Prédiction(C1)=(‘Vous devriez prendre un **Parapluie**’)

Prédiction(C2)=(‘Vous devriez porter un **Pull**’)

Prédiction(C3)=(‘Vous devriez porter un **T-shirt**’)

Forêts aléatoires (Random Forest)

- **Qu'est-ce que Random Forest:** Random Forest est un algorithme d'apprentissage automatique supervisé basé sur des arbres de décision. C'est une méthode d'**ensemble learning (Bagging - Leo Breiman [1996])** qui combine plusieurs arbres de décision pour améliorer la précision des prédictions.
- **L'ensemble learning** (ou apprentissage par agrégation de modèles) est une approche en apprentissage automatique qui consiste à combiner plusieurs modèles d'apprentissage pour obtenir une prédiction plus robuste et précise qu'un seul modèle. L'idée principale est que plusieurs modèles, même faibles individuellement, peuvent être combinés pour améliorer la performance globale
- **Basé sur la loi des grands nombres:** Ce principe stipule que, dans un grand ensemble de modèles ou d'observations, les erreurs individuelles (ou "bruits") des modèles ont tendance à se compenser lorsqu'elles sont moyennées. Ainsi, si chaque modèle dans l'ensemble fait des erreurs indépendantes, les erreurs totales du système combiné tendent à diminuer au fur et à mesure que l'on augmente le nombre de modèles. Cela permet de réduire la **variance** du modèle global

Description Graphique du fonctionnement d'un Random Forest

1. Bootstrap

Data frame Original		
obs	X	Y
1	4,3	2,3
2	2,1	2,0
3	3,0	3,1
4	5,3	4,5

Échantillon 1		
obs	X	Y
1	4,3	2,3
2	2,1	2,0
2	2,1	2,0
1	4,3	2,3

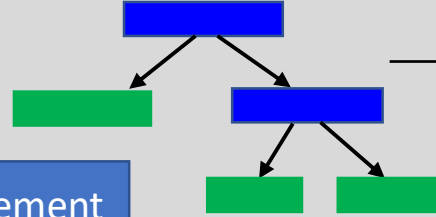
Échantillon 2		
obs	X	Y
3	3,0	3,1
2	2,1	2,0
2	2,1	2,0
3	3,0	3,1

Échantillon 3		
obs	X	Y
1	4,3	2,3
4	5,3	4,5
4	5,3	4,5
1	4,3	2,3

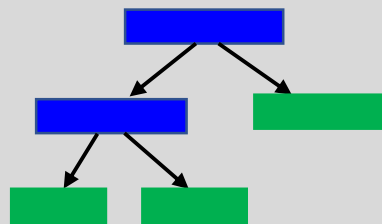
Échantillon 4		
obs	X	Y
2	2,1	2,0
2	2,1	2,0
4	5,3	4,5
4	5,3	4,5

2. Entraînement

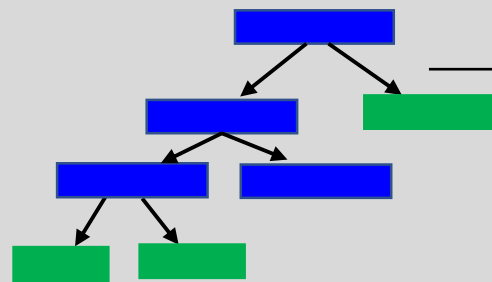
Arbre 1



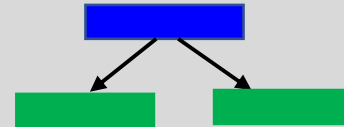
Arbre 2



Arbre 3



Arbre 4



Out of Bag (OOB) 1

obs	X	Y
3	3,0	3,1
4	5,3	4,5

OOBE 1

3. Évaluation de l'erreur (OOBE) individuelle

Out of Bag 2 (OOB)

obs	X	Y
1	4,3	2,3
4	5,3	4,5

OOBE 2

Out of Bag 3 (OOB)

obs	X	Y
2	2,1	2,0
3	3,0	3,1

OOBE 3

Out of Bag 4 (OOB)

obs	X	Y
1	4,3	2,3
3	3,0	3,1

OOBE 4

4. Erreur Ensemble Out of Bag Error (OOBE)

$$OOBE = \frac{1}{n} \sum_{i=1}^n OOBE_i$$

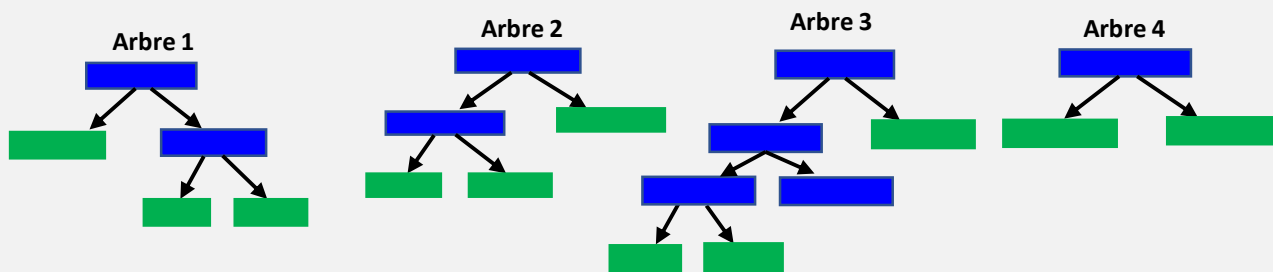
Classification

OOBE { Accuracy
Recall
Precision

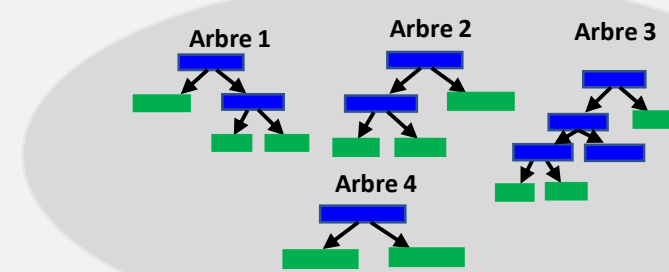
Régression

OOBE { RMSE
MSE

Description Graphique du fonctionnement d'un Random Forest (Prédiction- Continuation)



1	✓	1	✓	1	✓	1	✗
2	✗	2	✓	2	✓	2	✓
3	✓	3	✓	3	✗	3	✓
4	✓	4	✗	4	✓	4	✓
5	✓	5	✓	5	✗	5	✗
6	✓	6	✓	6	✓	6	✗
7	✗	7	✗	7	✓	7	✗
8	✗	8	✓	8	✓	8	✓
9	✓	9	✗	9	✓	9	✓
10	✓	10	✓	10	✗	10	✓
70%		70%		70%		60%	



**Classification
(Règle de la majorité)**

$$y_{pred} = \underset{c}{\operatorname{argmax}} \left(\sum_i^T y_{pred_t} \right)$$

Régression

$$y_{pred} = \frac{1}{T} \sum_i^T y_{pred_t}$$

1	✓
2	✓
3	✓
4	✓
5	✗ ✓
6	✓
7	✗
8	✓
9	✓
10	✓

$y_{pred} \geq 80\%$

Ligne 5: prédiction aléatoire catégories équivalentes. La prédiction d'ensemble peut être ✗ ou ✓

Caractéristiques importantes

- Les résultats des modèles sont divers. Autrement dit, les modèles ne se trompent pas de la même manière pour les mêmes individus
- La combinaison des prédictions des modèles individuels s'améliore de manière significative

Forêts aléatoires (Random Forest) – Avantages et inconvénients

■ Avantages:

- **Moins sensible au sur-apprentissage** En combinant plusieurs arbres, Random Forest réduit la variance globale, ce qui permet d'obtenir des modèles plus robustes et moins susceptibles de s'adapter excessivement aux données d'entraînement.
- **Peut gérer des données très complexes**: Random Forest peut gérer des données avec un grand nombre de caractéristiques (variables) et est performant même si les données sont bruitées.
- **Estimation de l'importance des caractéristiques** : Random Forest permet de déterminer quelles variables sont les plus influentes dans la prédiction, ce qui peut être utile pour l'interprétation des résultats.
- **Flexibilité**: Random Forest est peu sensible à l'échelle des variables et n'exige pas une mise à l'échelle des données, contrairement à d'autres modèles comme les régressions linéaires ou les SVM.

Forêts aléatoires (Random Forest) – Avantages et inconvénients

- **Inconvénients:**

- **Modèle "boîte noire":** La combinaison d'un grand nombre d'arbres rend l'interprétation du modèle plus difficile. Contrairement aux arbres de décision simples, il est plus compliqué de comprendre exactement comment une décision a été prise par l'ensemble des arbres.
- **Consommation de mémoire et temps de calcul élevé:** L'entraînement de multiples arbres de décision sur de grands ensembles de données peut être coûteux en termes de temps et de ressources informatiques, notamment en raison de la nécessité d'effectuer plusieurs itérations sur de larges volumes de données.
- **Pas idéal pour les données très déséquilibrées** : Si les classes sont fortement déséquilibrées (par exemple, une classe majoritaire et une classe minoritaire), Random Forest peut avoir tendance à prédire systématiquement la classe majoritaire. Des techniques comme l'**ajustement des poids** ou l'**échantillonnage** peuvent être nécessaires pour compenser ce biais.
- **Dépendance à la randomisation** : L'algorithme repose fortement sur la randomisation (par exemple, sélection aléatoire des sous-ensembles de données et des caractéristiques), ce qui peut rendre les résultats légèrement variables d'une exécution à l'autre. Cependant, cela peut être atténué en fixant une **seed** pour garantir la reproductibilité des résultats

K-Plus Proches Voisins (KNN)

- **Définition** : KNN (K-Nearest Neighbors) est un algorithme d'apprentissage supervisé basé sur des calculs de similarité (distance) utilisé pour des tâches de classification et de régression. En termes de distance, deux tuples (A,B) d'un data frame ou d'une matrice sont égaux si la distance entre les deux est nulle ($d(A, B) = 0$)
- **Principe de base** : KNN fonctionne en trouvant les K voisins les plus proches d'un point donné et en prenant une décision basée sur la majorité (classification) ou la moyenne (régression) des voisins.
- **Applications** :
 - Classification d'images
 - Reconnaissance de texte
 - Recommandation de produits

K-Plus Proches Voisins (KNN) – Comment ça fonctionne?

1. **Choix du paramètre K** : K est le nombre de voisins à considérer. Par exemple, K = 3 signifie que l'algorithme regarde les 3 voisins les plus proches pour prendre une décision.
2. **Calcul de la distance** : L'algorithme mesure la distance entre le point à classer et les autres points. Les distances couramment utilisées sont :

✓ Distance Euclidienne :

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

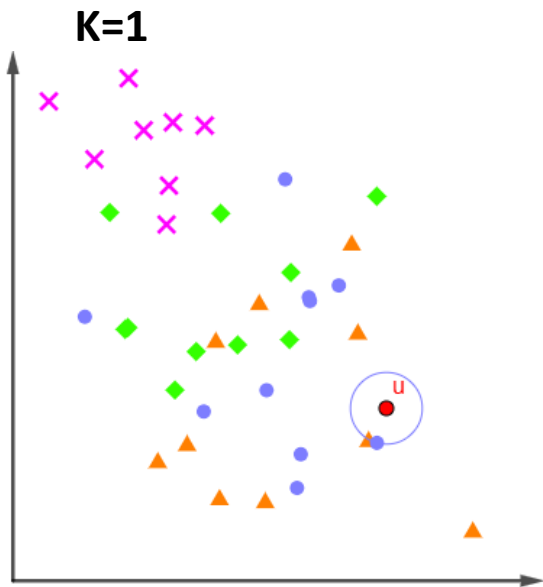
✓ Distance de Manhattan :

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

3. **Identification des voisins** : L'algorithme identifie les K points les plus proches du point à classer.
4. **Décision** :
 1. **Classification** : Le point est classé dans la classe majoritaire des K voisins.
 2. **Régression** : La valeur prédite est la moyenne des valeurs des K voisins.

K-Plus Proches Voisins (KNN) – Fonctionnement graphique de l'algorithme

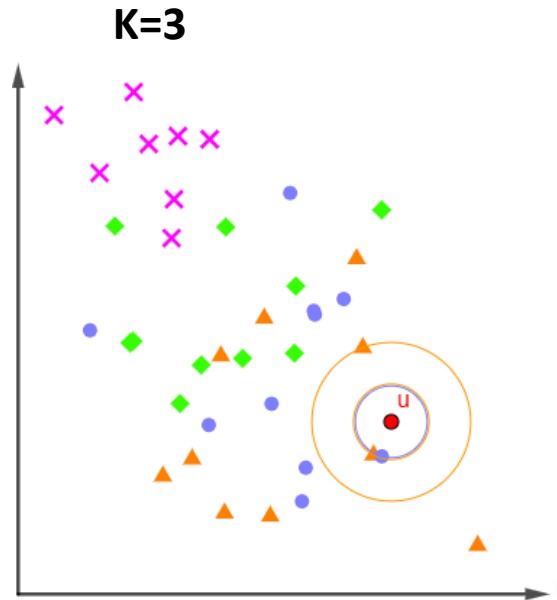
- ✓ Contexte: Nous voulons prédire la classe d'un nouveau point (tuple) **u** en utilisant la méthode KNN. Nous disposons d'un ensemble de données étiquetées à 4 catégories (**x**, ●, ◆, ▲). Les graphiques représentent la distribution spatiale de l'ensemble de données par rapport au point **u**. Quelle serait la prédiction obtenue par le KNN pour les différentes valeurs de K?



Distance minimale ($k=1$)=1 ●

Prédiction ($k=1$)= classe ●

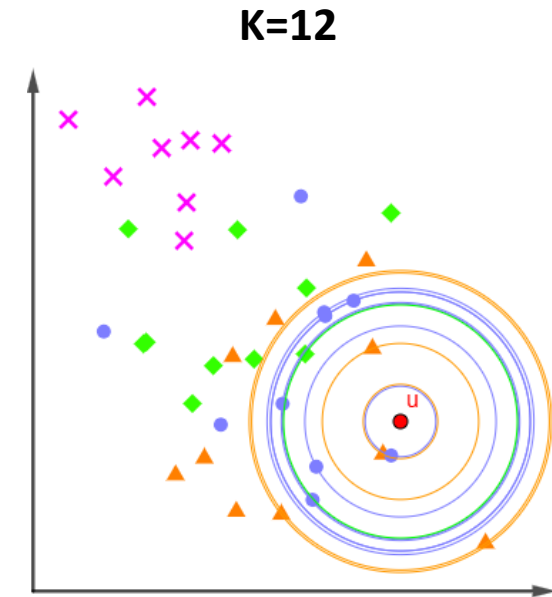
Prédiction_reg ($k=1$)= Valeur ●



Distances minimale ($k=3$)=1 ● , 2 ▲

Prédiction ($k=3$)= classe majoritaire ▲

Prédiction_reg ($k=3$)= moyenne valeurs(● , ▲)



Distances minimale ($k=12$)=7 ● , 4 ▲ , 1 ◆

Prédiction ($k=12$)= classe majoritaire ●

Prédiction_reg ($k=12$)= moyenne valeurs(● , ▲ , ◆)

K-plus proches voisins – Avantages et limites

■ **Avantages :**

- **Simplicité** : Facile à comprendre et à implémenter.
- **Non-paramétrique** : Aucun modèle prédéfini n'est nécessaire. Il ne fait aucune hypothèse sur la distribution des données.
- **Flexible** : Peut être utilisé pour la classification et la régression.

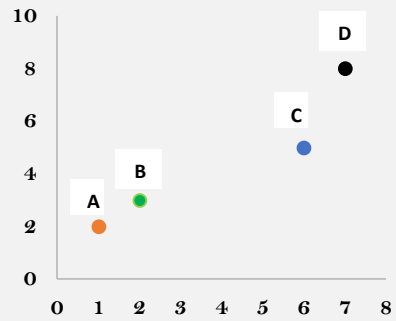
■ **Limites :**

- **Coût computationnel élevé** : Le calcul des distances pour chaque prédiction peut être lent, surtout avec de grands ensembles de données.
- **Sensibilité aux valeurs aberrantes** : Les points extrêmes peuvent influencer fortement la classification.
- **Choix de K crucial** : Le choix du bon K (trop grand ou trop petit) peut affecter la performance du modèle.

Clustering Hiérarchique Ascendant

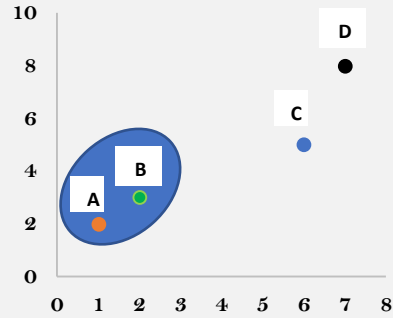
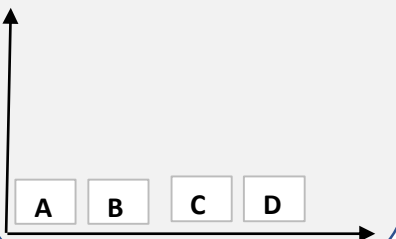
- **Définition** : La classification hiérarchique ascendante (HAC) est une méthode de clustering qui construit une hiérarchie de clusters en fusionnant les plus proches à chaque étape.
- **Objectif** : Créer une hiérarchie de clusters qui peut être utilisée pour extraire un nombre spécifique de groupes à partir de la structure finale.
- **Principales caractéristiques** :
 - Méthode basé sur la ressemblance des individus (ou calcul de distance (Euclidienne, Manhattan, etc))
 - **Méthode ascendante** : commence avec chaque point comme un cluster individuel et les fusionne progressivement.
 - Aucune spécification du nombre de clusters à l'avance. Construit des clusters de manière spontané
 - Produit une structure arborescente (nommée dendrogramme) montrant les relations entre les clusters.

Apprentissage non supervisée (Clustering Hiérarchique Ascendant –Principe de fonctionnement)



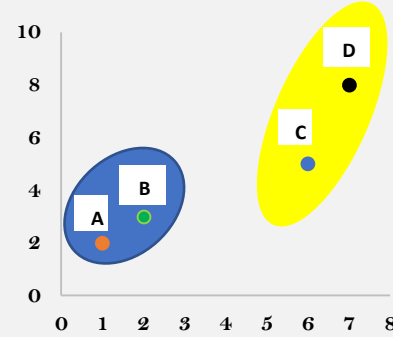
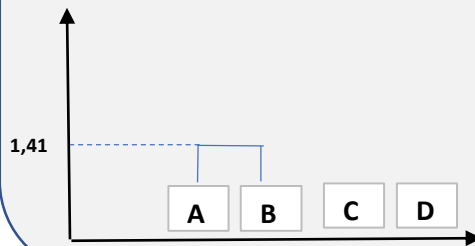
	A	B	C	D
A	0	1,41	5,83	8,49
B	1,41	0	4,47	7,07
C	5,83	4,47	0	3,16
D	8,49	7,07	3,16	0

$d_{\min}(A,B) = 1,41$, conformation du premier cluster remplaçant les individus A,B



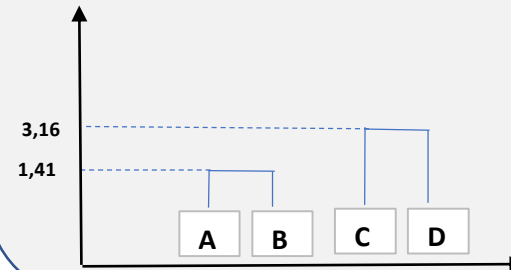
	{A,B}	C	D
{A,B}	0	4,47	7,07
C	4,47	0	3,16
D	7,07	3,16	0

La matrice de distance est recalculée. D'après le critère du saut minimum: $d(A,C) = 5,83$, $d_{\min}(B,C) = 4,47$, $d(A,D) = 8,49$, $d_{\min}(B,D) = 7,07$, alors $d_{\min}(\{A,B\},C) = 4,47$ et $d_{\min}(\{A,B\},D) = 7,07$



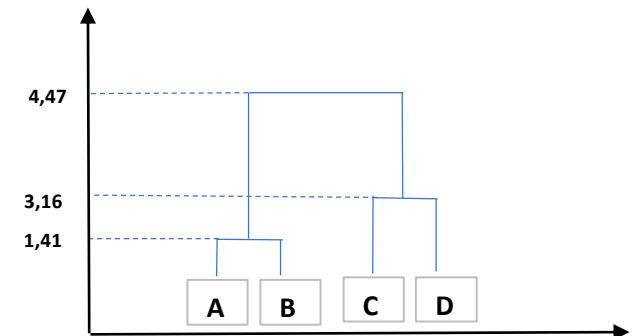
	{A,B}	{C,D}
{A,B}	0	4,47
{C,D}	4,47	0

Regroupement de {C,D} à distance minimale de 3,16. Le processus s'arrête car il n'y a plus d'individus à clustériser. $d(\{A,B\},\{C,D\}) = 4,47$



- Calcul des distances initiales:** L'algorithme calcule la matrice de distance pour l'ensemble d'individus, chacun étant considéré comme un cluster individuel
- Regroupement d'individus :** L'algorithme procède ensuite à un premier regroupement des individus à distance minimale (le regroupement est placé à cette distance). Ce nouveau cluster remplace les individus regroupés.
- Recalcul de distance :** La matrice de distances est recalculée en prenant en compte la distance entre les individus et le regroupement à l'aide du critère du saut minimum
- Répétition:** L'algorithme répète les étapes 2 et 3 jusqu'à ce qu'il ne reste plus qu'un seul cluster ou cluster o que le nombre souhaité de cluster soit atteint

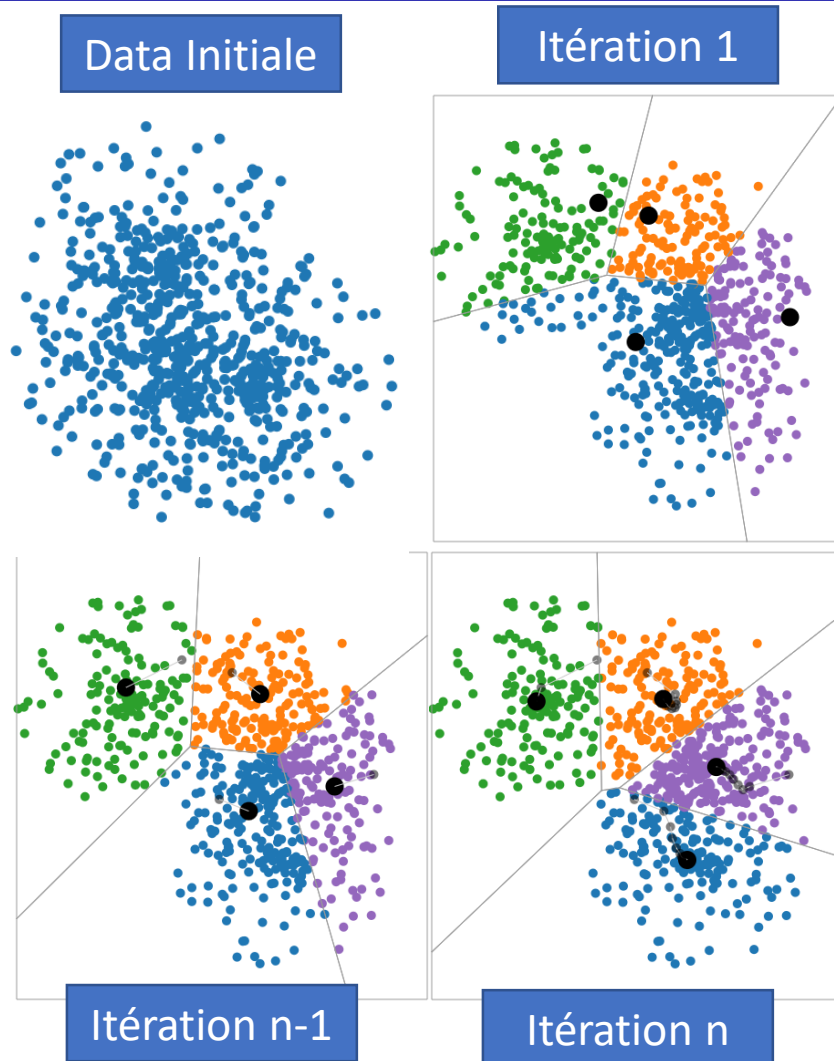
Dendrogramme obtenu



Clustering Hiérarchique Ascendant (Quelques avantages et inconvénients)

- **Avantages :**
 - **Pas de besoin de définir k (nombre de cluster) :** contrairement à l'algorithme k-means, il n'est pas nécessaire de spécifier le nombre de clusters à l'avance.
 - Produit une **hiérarchie** qui peut être utilisée pour explorer différentes granularités de clustering.
 - Peut être utilisé avec des données de type **non linéaire** et **de forme irrégulière**.
 - Possède multiples applications dans des différents domaines (exploration de données, bio-informatique, analyse de similarité de documents, etc.)
- **Inconvénients:**
 - **Coût computationnel élevé :** nécessite $O(n^2)$ (Par exemple, si l'on duplique le nombre d'individus, le temps de calcul quadruple.) opérations pour calculer la matrice des distances, ce qui peut devenir coûteux pour des jeux de données de grande taille.
 - **Sensibilité aux bruits :** les petits écarts ou les valeurs aberrantes peuvent affecter les résultats de manière significative.


Apprentissage non supervisé – Kmeans (Principe de fonctionnement)



1. **Choix du nombre de clusters:** Définition du nombre de clusters (K) souhaité.
2. **Sélection aléatoire des centroïdes initiaux:** L'algorithme Sélectionne aléatoirement k individus dans l'ensemble de données comme centres ou moyennes initiales des clusters.
3. **Assignment des individus aux clusters :** L'algorithme assigne chaque observation au centroïde le plus proche, en fonction de la distance euclidienne entre l'objet et le centroïde.
4. **Actualisation des centroïdes des clusters:** L'algorithme met à jour le centroïde des clusters en calculant les nouvelles moyennes de toutes les données qu'ils contiennent.
5. **Minimisation de somme totale des carrés intra-cluster:** L'algorithme minimise de manière itérative la somme totale des carrés intra-cluster. Autrement dit, répète les étapes 2 et 3 jusqu'à ce que les assignments de clusters cessent de changer ou que le nombre maximum d'itérations soit atteint.

Kmeans (Optimisation du nombre de clusters k)

- L'optimisation du nombre de clusters est une étape cruciale dans l'application de l'algorithme K-means, permettant de déterminer le nombre optimal de groupes dans lesquels les données doivent être segmentées. Plusieurs méthodes peuvent être utilisées pour cette optimisation, notamment la **méthode du coude**, le **score de silhouette**, et l'indice de **Davies-Bouldin**, qui permettent d'évaluer la qualité du clustering en fonction de critères de cohésion et de séparation des clusters.
- Le problème pourrait alors se réduire à la recherche itérative du nombre de clusters et ainsi choisir des valeurs optimales suivant l'une de ces méthodes. Une synthèse de la description de ces trois méthodes est présenté dans le tableau ci-dessous:




Tableau	Lien html
Synthèse des critères d'optimisation du nombre de clusters	

Apprentissage non supervisé – Kmeans (Quelques avantages et des inconvénients)

- **Avantages :**
 - **Simple** et **rapide** pour des grands ensembles de données.
 - Efficace pour des données bien séparées et des clusters convexe.
 - Un spectre large d'utilisation en plusieurs secteurs (par exemple, segmentation de clients (marketing), classification de gènes en groupes similaires (biologie), identification de comportements similaires de transactions bancaires (finance), etc).
- **Inconvénients:**
 - Nécessite de définir un **nombre de clusters K** défini à l'avance.
 - Sensible aux **centroïdes initiaux** (peut converger vers un minimum local).
 - Ne fonctionne pas bien avec des **clusters de formes irrégulières** ou des données avec du bruit.

Exemples pratiques d'utilisation de Scikit-Learn et PyCaret

Tableau d'exemples introductifs au Machine Learning

Exemple	Lien html
Prédiction du salaire des nouveaux membres d'une équipe de baseball américain à l'aide d'un arbre de décision (PyCaret)	
Prédiction des salaires des professionnels dans différents secteurs à l'aide du modèle Random Forest (SciKitLearn)	
Application de la méthode du coude (elbow) pour déterminer le nombre optimal de clusters à partir de données synthétiques	

Synthèse du Processus général du Machine Learning

- **Collecte des données** : L'étape initiale où des données pertinentes sont rassemblées.
- **Prétraitement** : Nettoyage, normalisation et transformation des données pour les rendre utilisables par les algorithmes.
- **Sélection du modèle** : Choix d'un algorithme approprié pour le problème donné.
- **Entraînement** : Utilisation des données pour ajuster les paramètres du modèle.
- **Évaluation** : Validation de la performance du modèle sur un ensemble de données de test à l'aide de métriques comme RMSE, précision, AUC, etc.
- **Mise en production** : Utilisation du modèle pour faire des prédictions ou décisions dans des environnements réels.