Principe et Applications du Machine Learning (1)

Corentin Allaire

Remerciement à **David Rousseau** et **Françoise Bouvet** pour une partie des figures

À quoi s'attendre

Machine Learning :

- Concepts fondamentaux de « L'IA »
- Les différentes techniques de Machine Learning :
  - Supervisé
  - Non-supervisé
- Application à la recherche du boson de Higgs

Deep Learning :

- Brève histoire du réseau de neurones
- Les différentes techniques de Deep Learning :
  - Perceptron multi-couche
  - L'auto-encoder
  - Réseaux à base de graphe
- Application à la reconstruction d'objet en physique des hautes énergies

À quoi s'attendre

Machine Learning :

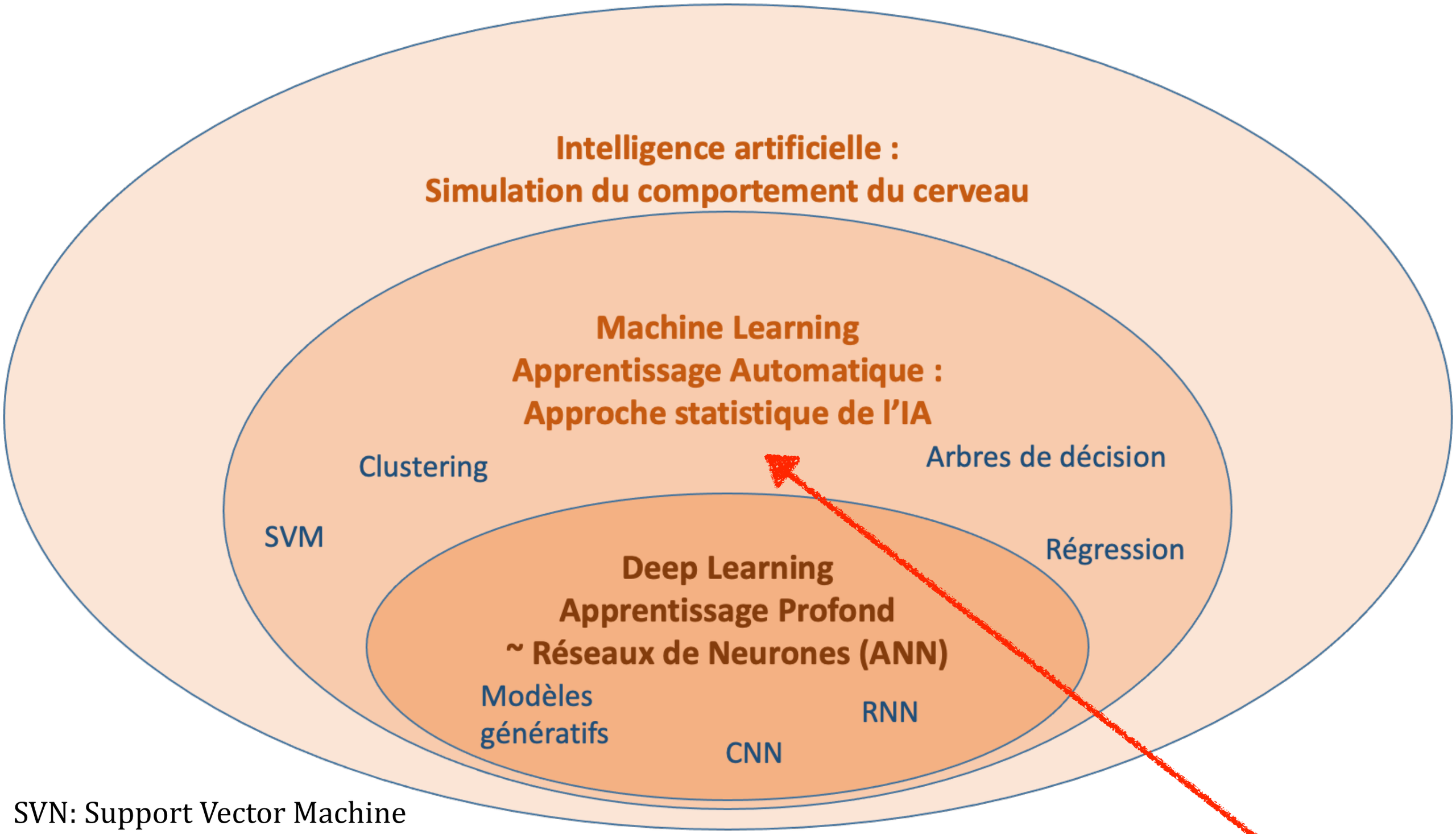
- Concepts fondamentaux de « L'IA »
- Les différentes techniques de Machine Learning :
  - Supervisé
  - Non-supervisé
- Application à la recherche du boson de Higgs

Deep Learning :

- Brève histoire du réseau de neurones
- Les différentes techniques de Deep Learning :
  - Perceptron multi-couche
  - L'auto-encoder
  - Réseaux à base de graphe
- Application à la reconstruction d'objet en physique des hautes énergies

Aujourd'hui

Une question de nomenclature



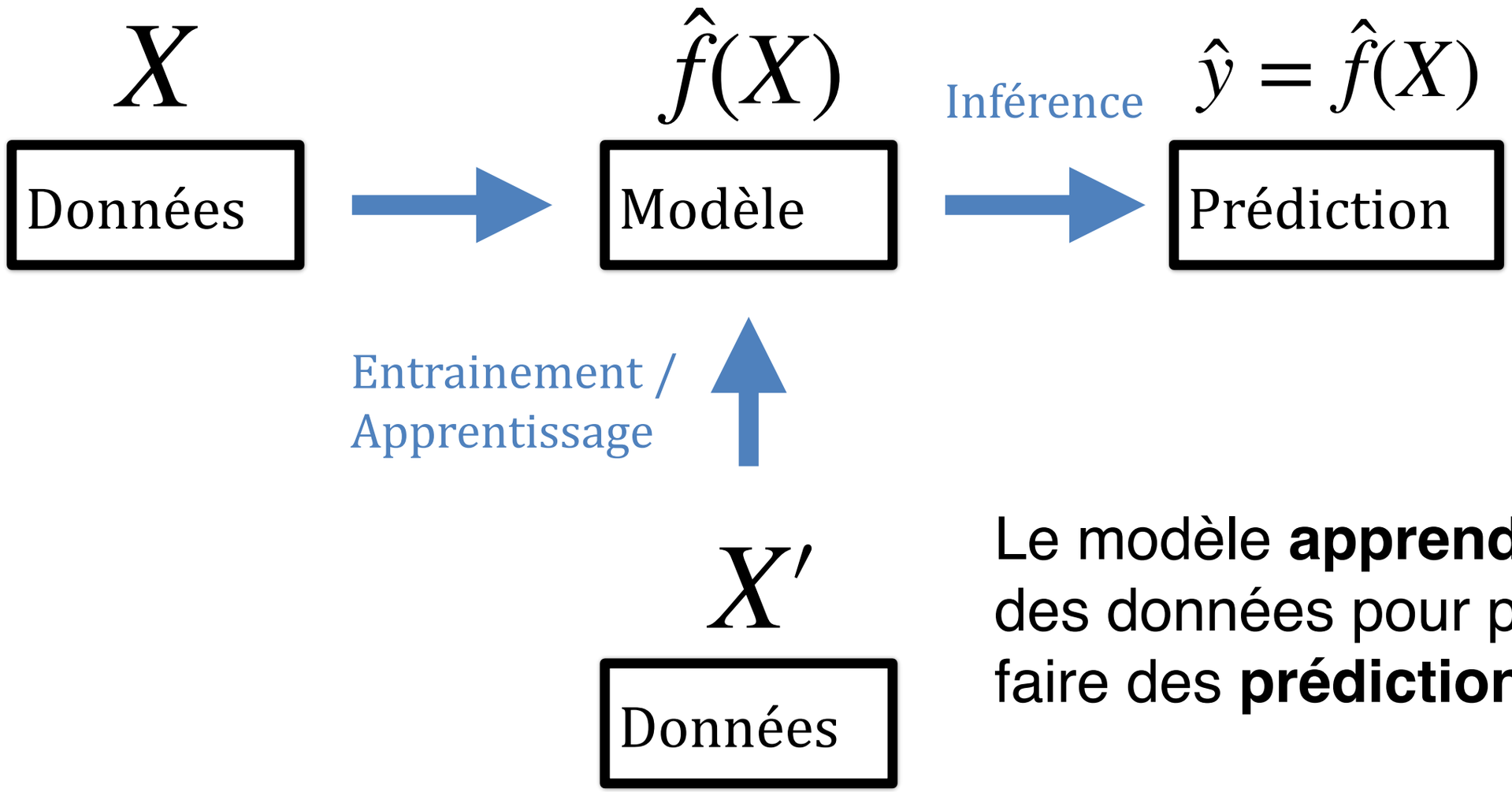
SVN: Support Vector Machine
CNN: Convolution NN
RNN: Recurrent NN

Le Machine Learning, qu'est ce que c'est ?

Machine Learning : Extraire de l'information de (grandes quantités de) **données** à l'aide d'un modèle

Réalité :

$$y = f(X)$$



Le modèle **apprend** à partir des données pour pouvoir faire des **prédictions**

Différents types d'apprentissage possibles

Apprentissage supervisé :

- On dispose de donnée **étiquetées**
- On souhaite effectuer une **prédiction** sur de nouvelle donnée inconnues

Exemples :

- Classification ←
- Régression
- Classement
- Génération de texte, d'image

Chats

Chiens

?

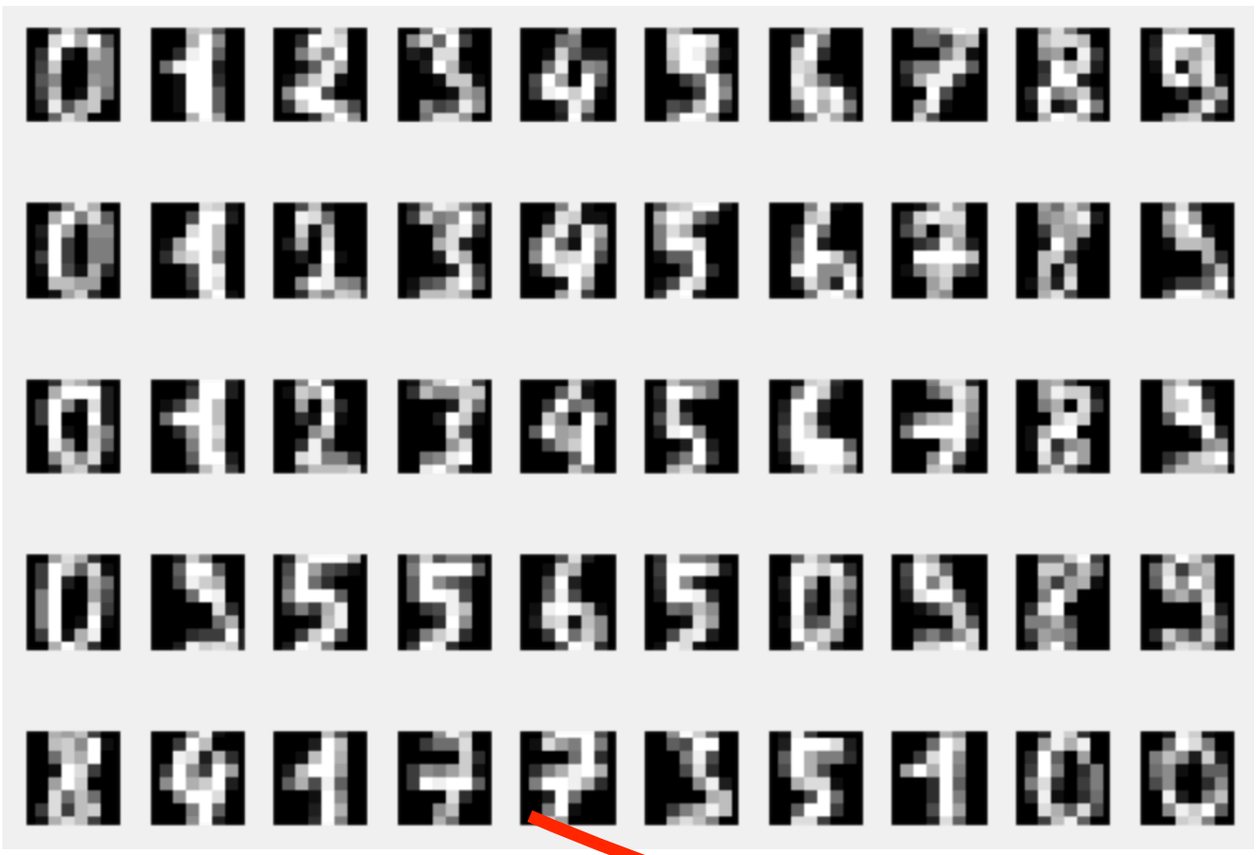
Différents types d'apprentissage possibles

Apprentissage non supervisé :

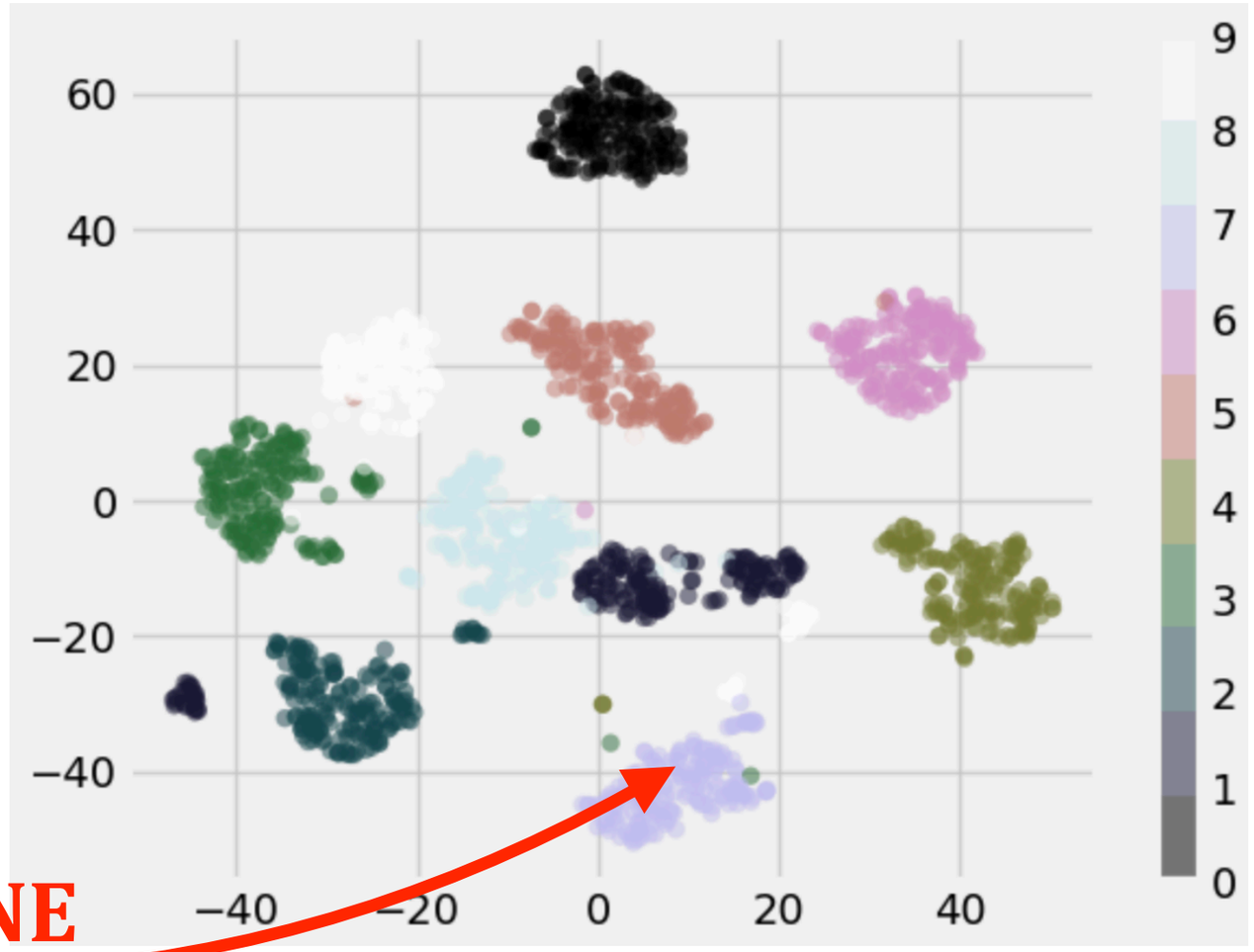
- On dispose de données **non-étiquetées**
- On souhaite obtenir de **nouvelles informations** sur ces données

Exemples :

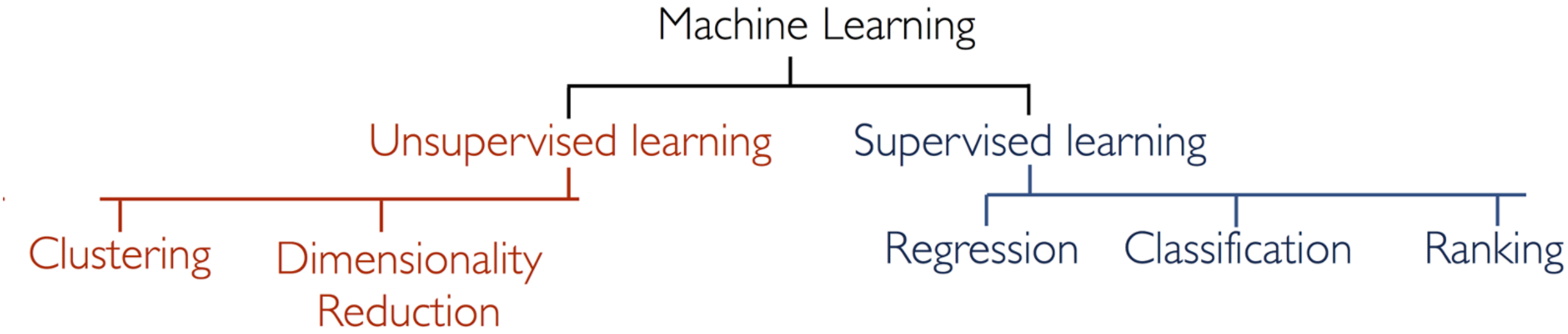
- Clustering ←
- Reduction de dimension ←
- Détection d'anomalie
- Génération de texte, d'image



t-SNE



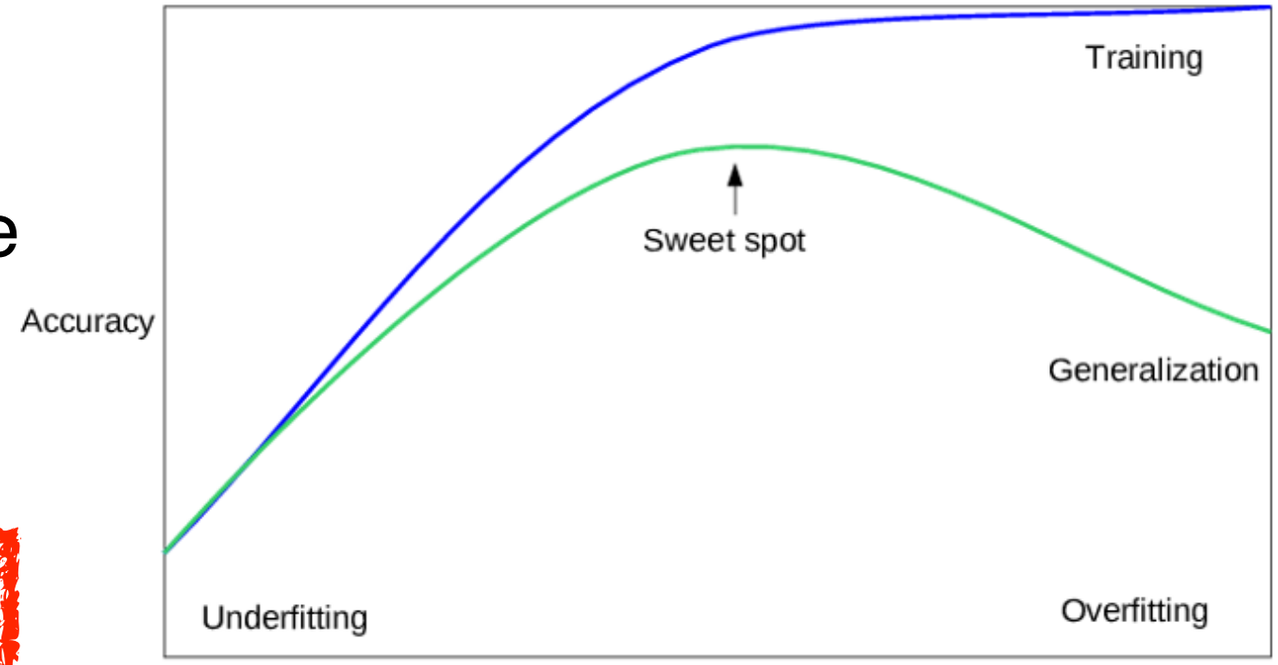
Différents types d'apprentissage possibles



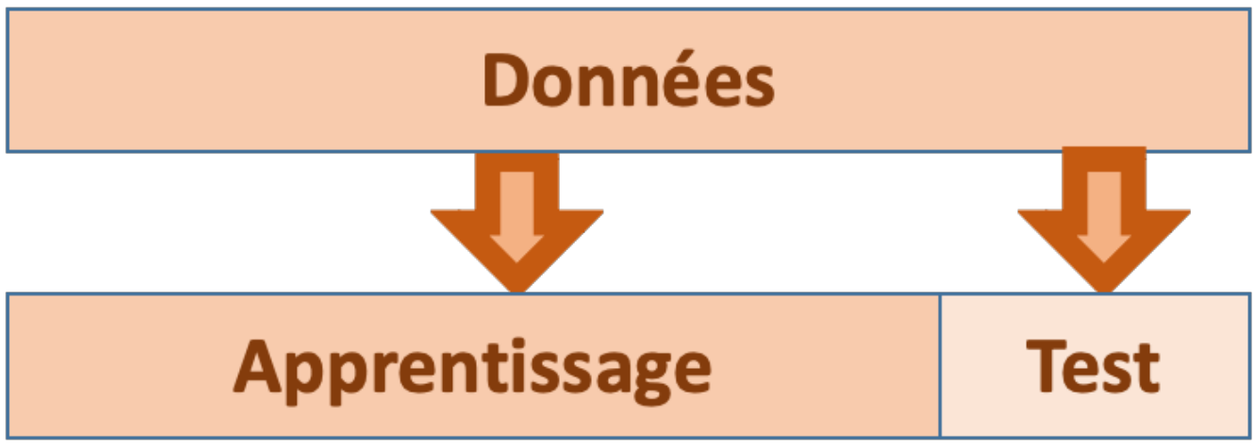
La première étape avant d'utiliser le ML est toujours **d'identifier** le type de problème auquel on est confronté pour déterminer quel type **d'algorithme** utiliser

La gestion des données

Le **succès** d'un algorithme de ML repose sur la bonne utilisation de ses données **d'entraînement** et de **test**



Entrainement supervisé : Ne **<u>jamais tester</u>** un modèle avec les données **d'apprentissage**



Overfitting : Le modèle apprend **par cœur** les données d'entraînement, mais ne sait pas généraliser

La gestion des données

Principaux défis :

- Suffisamment de données doivent être disponible
- Les données doivent être **représentatives**
- Uniquement des données **pertinentes** pour le problème
- Pour la classification, les différentes classes doivent être **équilibrées**

L’évaluation du modèle

Un critère d’évaluation est indispensable pour répondre à la question : **quel est le meilleur modèle ?**

Le critère dépendra du problème :

- **Régression** : Erreur aux moindres carrés

$$\mathrm{MSE}(y,\hat{y}) = \frac{1}{N_{samples}} \sum_{i=0}^{N_{samples}-1} (y_i - \hat{y}_i)^2$$

- **Classification** : Précision, Sensibilité, Spécificité, Aire sous la courbe…

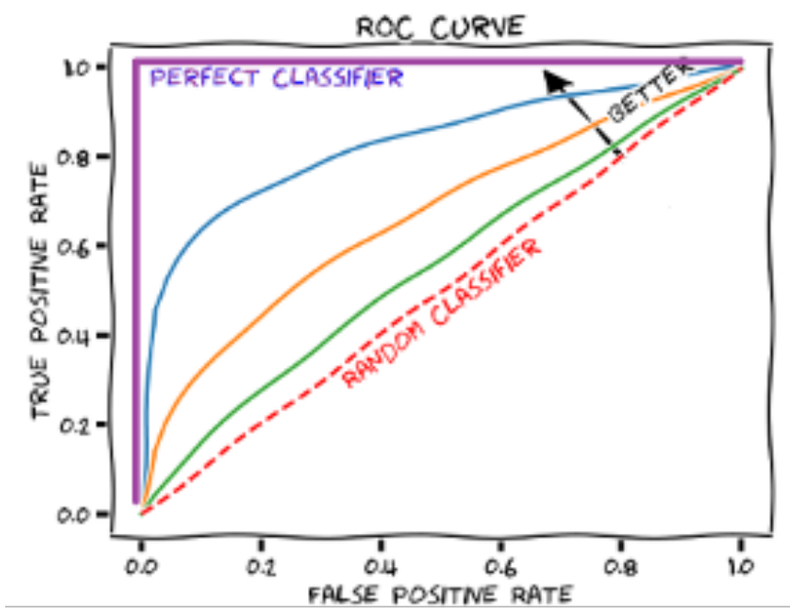
Prédiction

Réalité	FAUX	VRAI
FAUX	VN	FP
VRAI	FN	VP

$$Precision = \frac{VP}{VP + FP}$$

$$Sensibilite = \frac{VP}{VP + FN}$$

$$Specificite = \frac{VN}{VN + FP}$$



Un mot sur scikit-learn

https://scikit-learn.org/

- Bibliothèque **python** pour le Machine Learning
- Inclus :
  - Processing des données
  - Entrainement supervisé
  - Entrainement non supervisé
  - Visualisation de donnée
- Principalement maintenue par **l'Inria** (sur le plateau)

Quelques exemples : non supervisé

Reduction de dimension: **t-SNE** (t-distributed stochastic neighbour embedding)

Extension des algorithmes de SNE :

- Dans **P** : calcule la probabilité que i et j sois choisis comme **voisin** (choix de voisin basé sur une distribution gaussienne)

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

Quelques exemples : non supervisé

Reduction de dimension: **t-SNE** (t-distributed stochastic neighbour embedding)

Extension des algorithmes de SNE :

- Dans **P** : calcule la probabilité que i et j sois choisis comme **voisin** (choix de voisin basé sur une distribution gaussienne)
- Même chose dans **Q**

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

Quelques exemples : non supervisé

Reduction de dimension: **t-SNE** (t-distributed stochastic neighbour embedding)

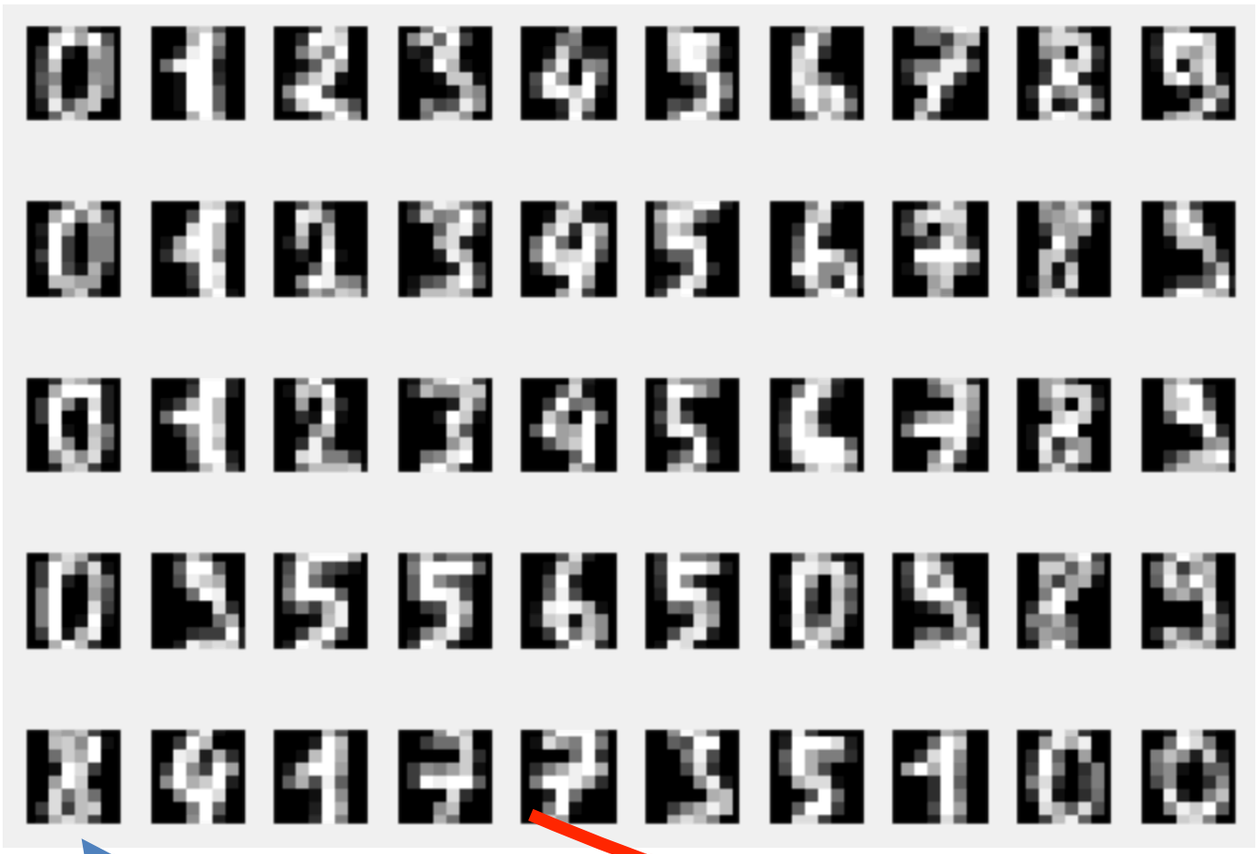
Extension des algorithmes de SNE :

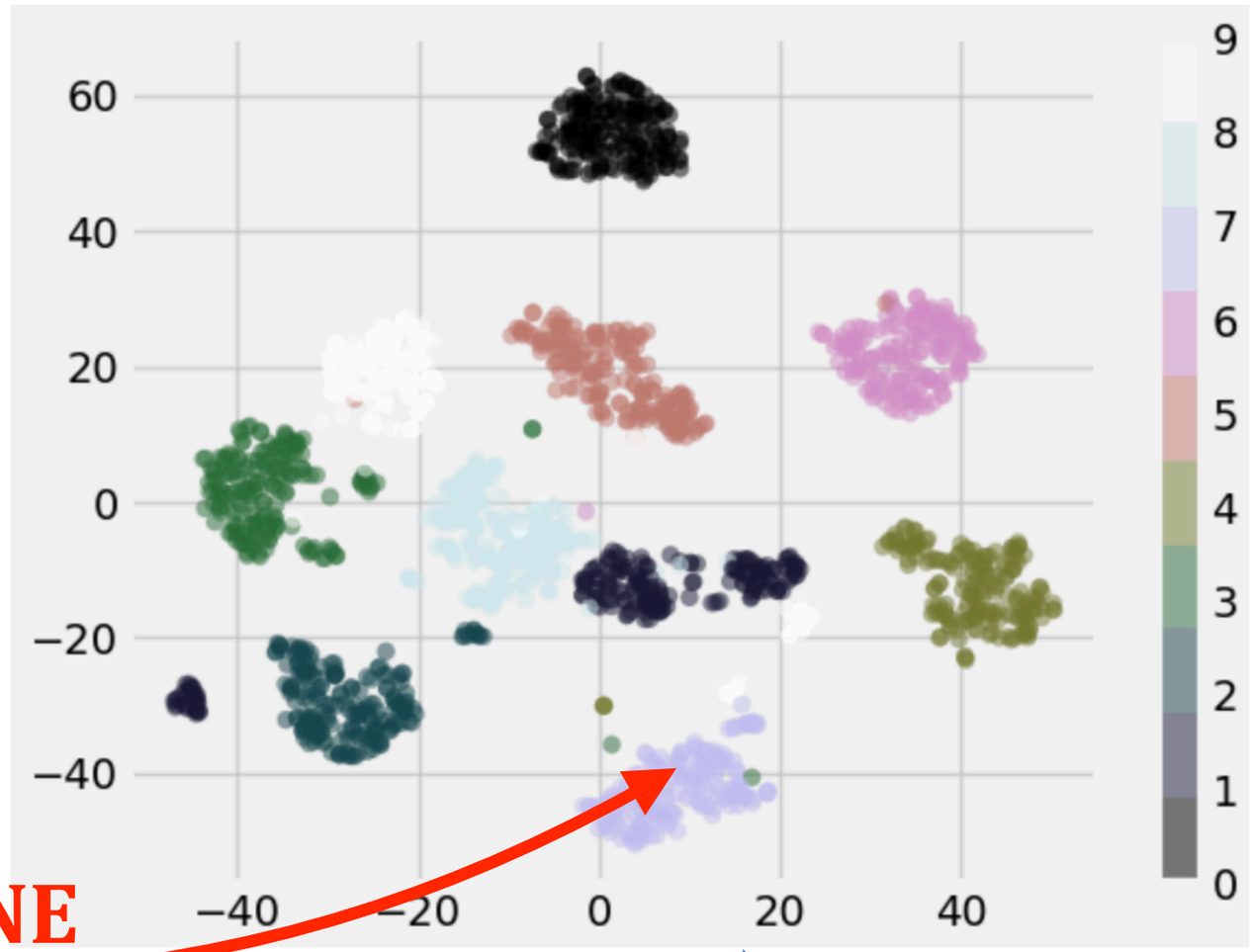
- Dans **P** : calcule la probabilité que i et j sois choisis comme **voisin** (choix de voisin basé sur une distribution gaussienne)
- Même chose dans **Q**
- Si Q est une bonne représentation de P, alors $\frac{p_{j|i}}{q_{j|i}}$ tends vers 1
- On trouve les positions dans Q par **descente de gradient**

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Quelques exemples : non supervisé

Exemple avec des chiffres





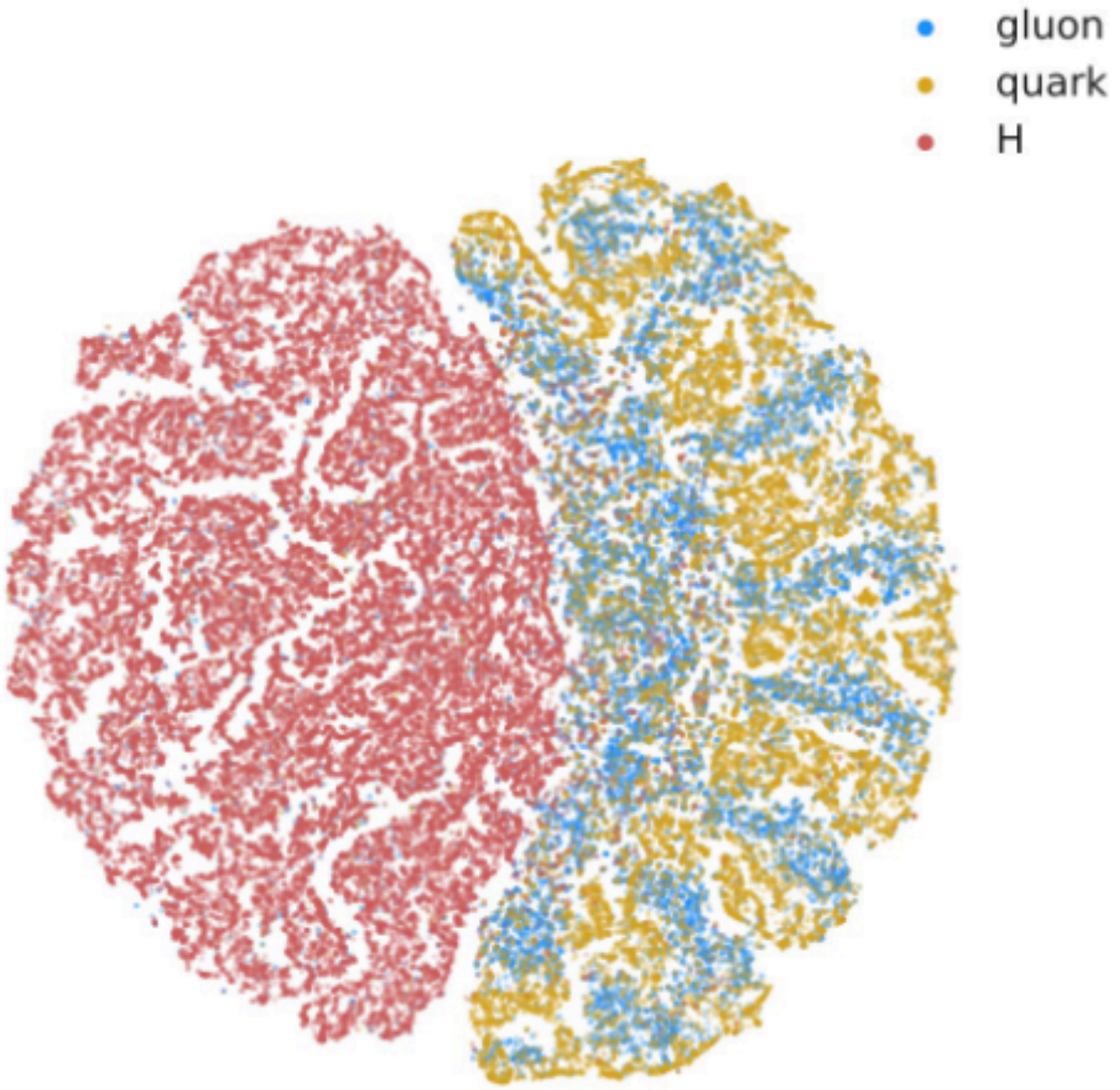
t-SNE

Image 8*8 = 64D

2D plot

t-SNE pour l'explicabilité des réseaux de neurones

- Les **réseaux de neurones** (cf demain) sont aussi utilisés pour la classification des **particules**
- Beaucoup plus complexe (projection en dimension > 100)
- L'utilisation d'algorithme de **t-SNE** nous permet de « **visualiser** » partiellement le fonctionnement de notre algorithme
- Visualisation de **l'encodage** de jet de particules en fonction de leurs sources



Quelque exemples : non supervisé

Clustering : **kMean**

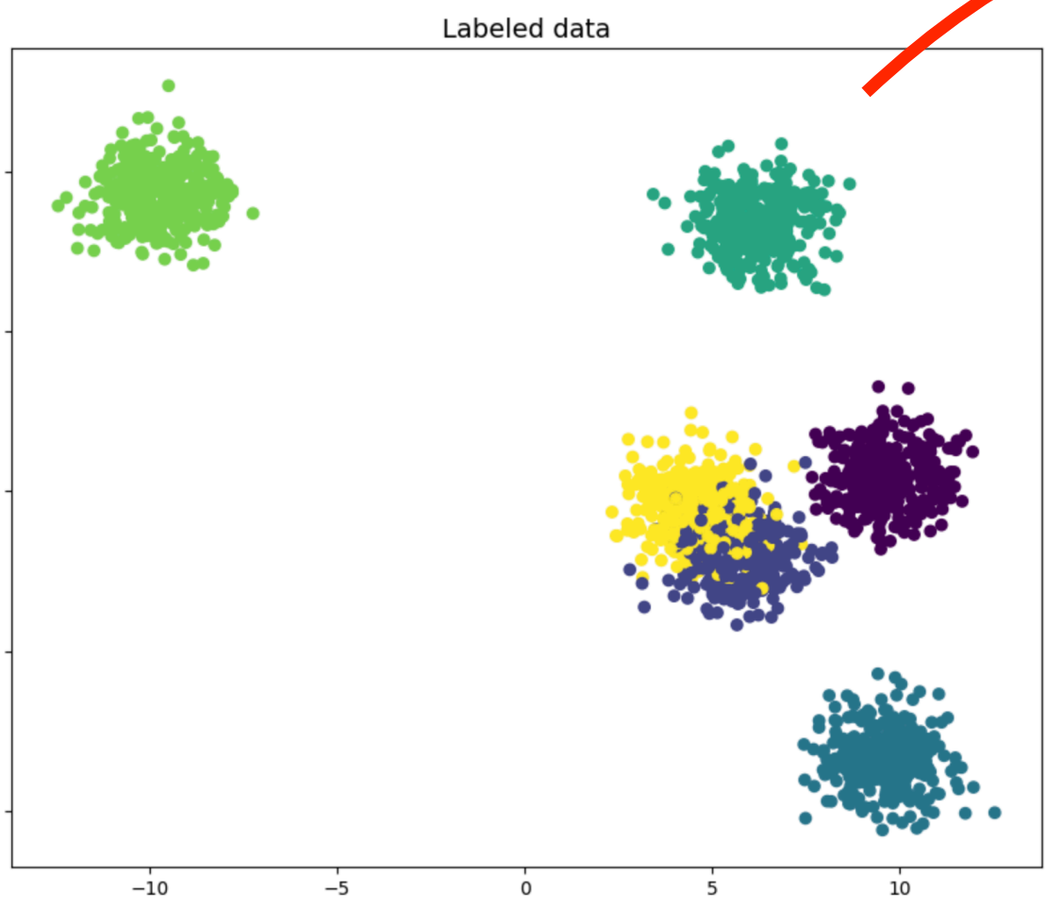
Divise les N données en **K clusters**

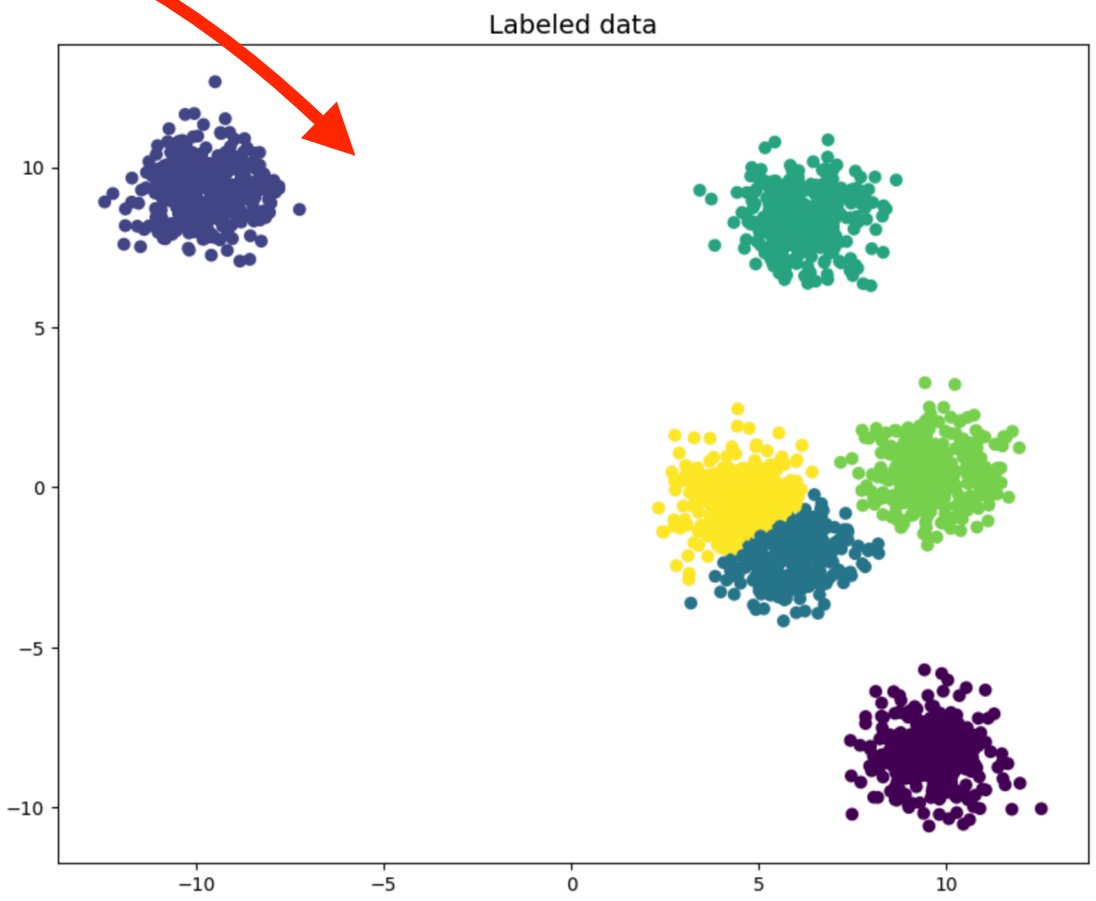
Extrêmement **rapide et efficace**

Nombreuses applications

Spécifié initialement par l'utilisateur !

K = 6





Quelque exemples : non supervisé

Clustering : **kMean**

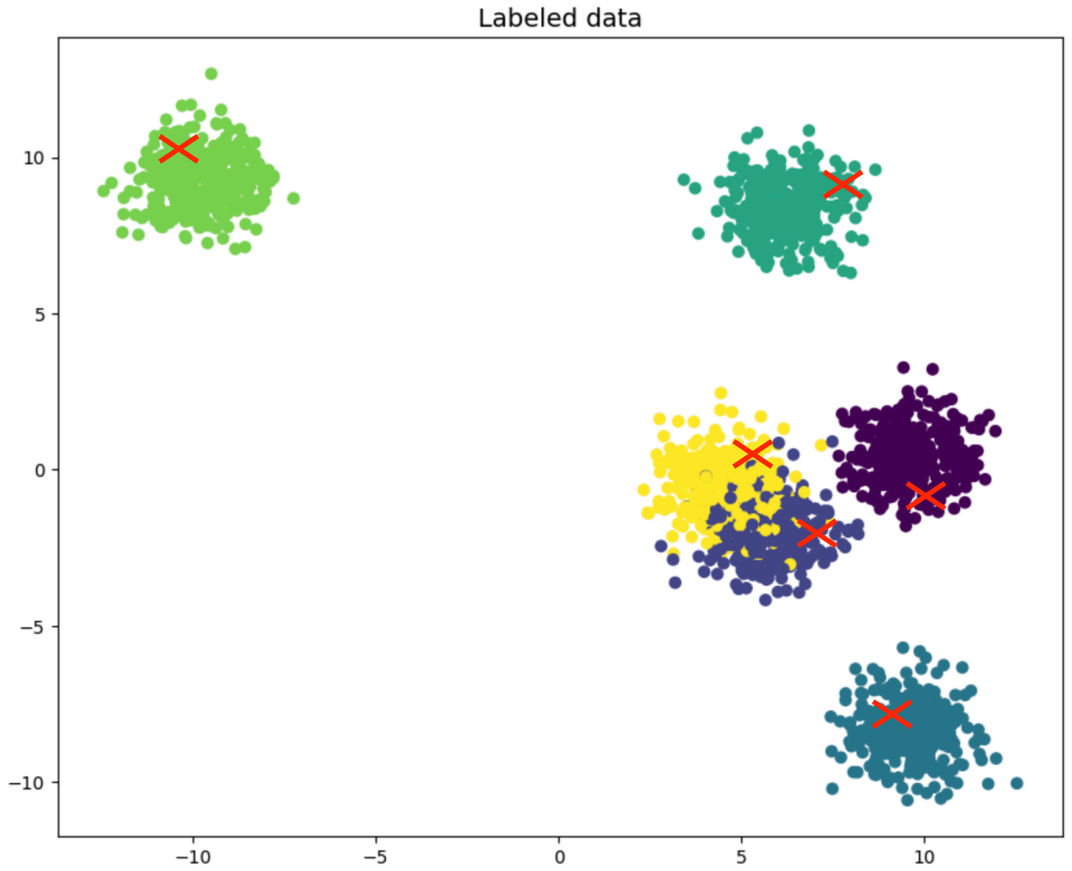
Comment ça marche ?

L'algorithme recherche K centroids minimisant **l'inertie** du système

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

Trois étapes :

- Sélection de la position initiale des centroids (K points **aléatoires**)



Quelque exemples : non supervisé

Clustering : **kMean**

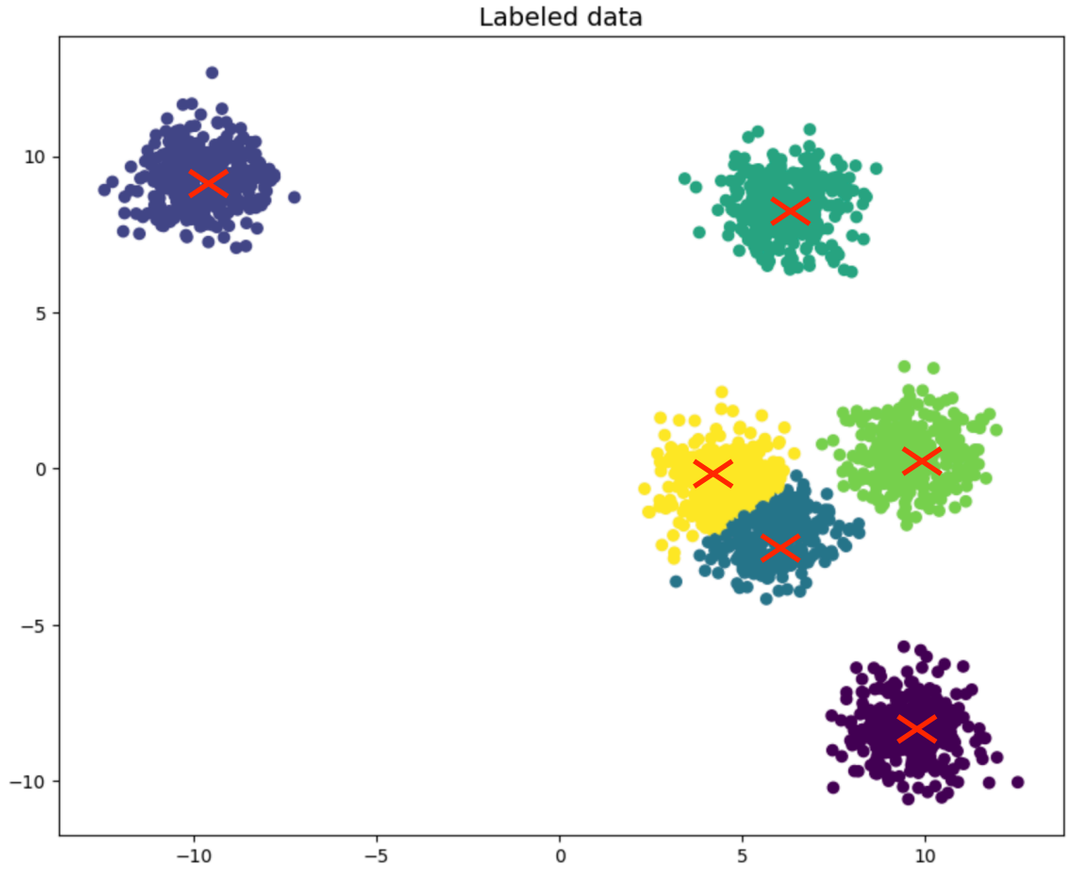
Comment ça marche ?

L'algorithme recherche K centroids minimisant **l'inertie** du système

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

Trois étapes :

- Sélection de la position initiale des centroids (K points **aléatoires**)
- Associe les points au centroid le plus proche
- Crée de nouveau centroid en prenant la valeur moyenne de chaque cluster



Quelque exemples : non supervisé

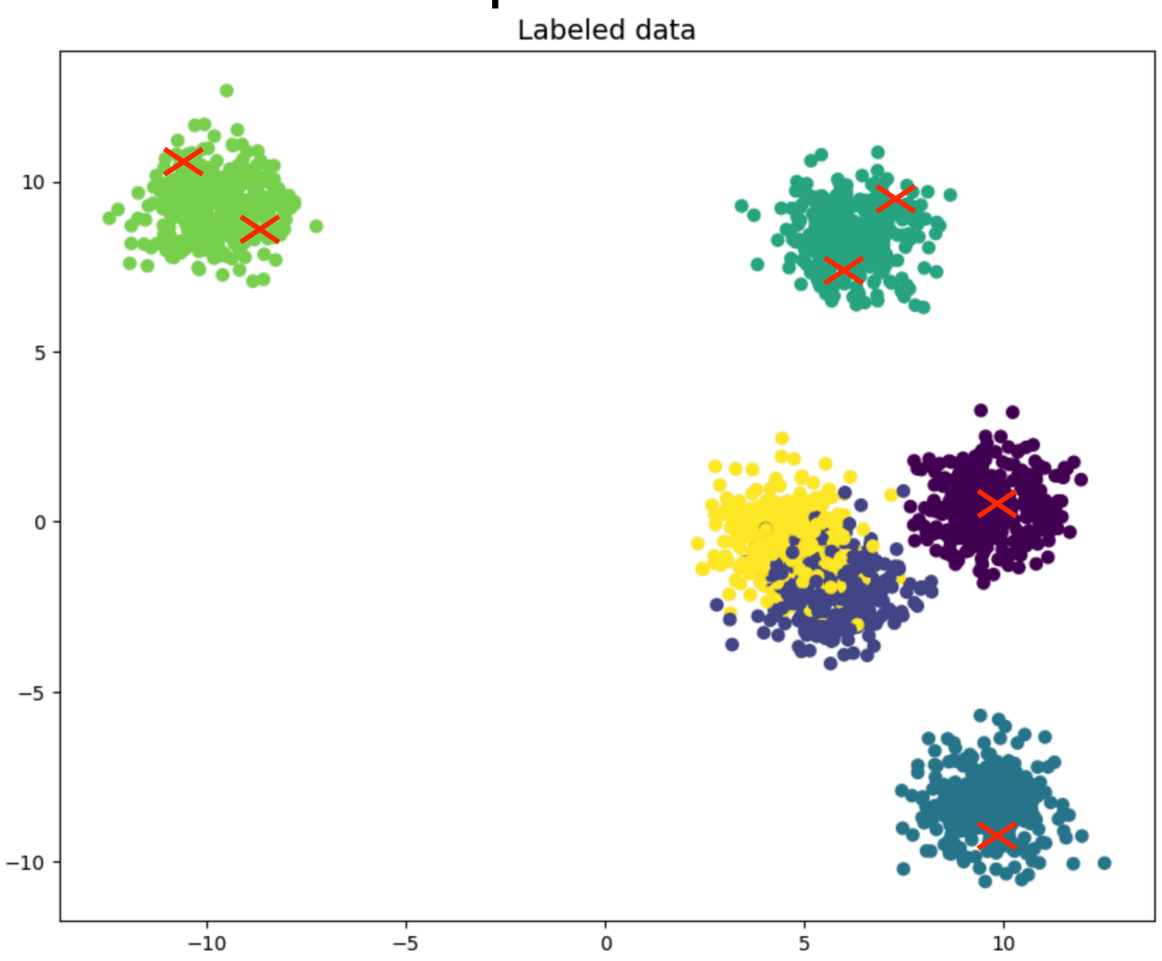
Clustering : **kMean**

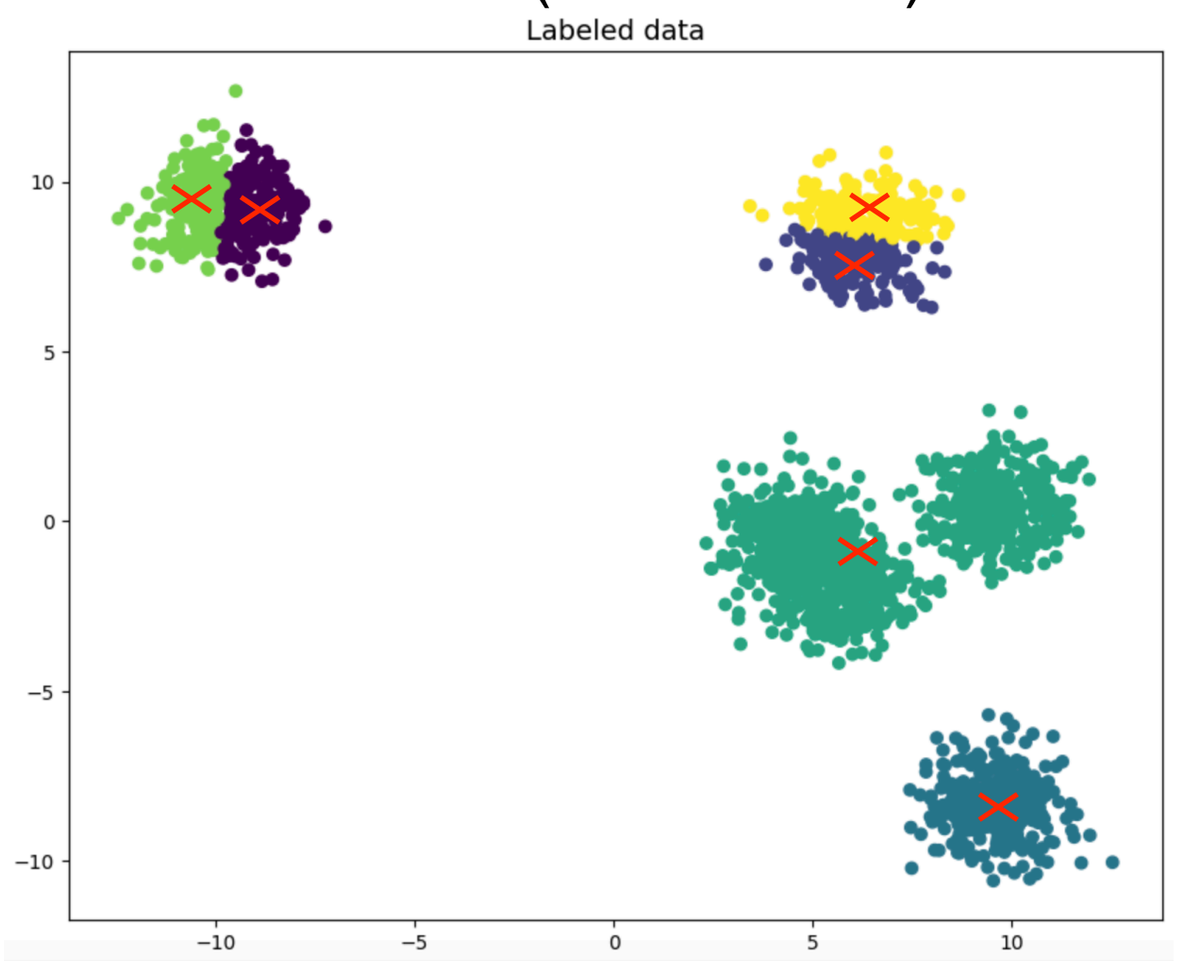
Convergence ?

Le résultat dépend grandement des point choisit pour l'initialisation et la structure des données

L'algorithme étant rapide, il est **répété n fois** avec différentes initialisations

Des techniques d'initialisation différentes existent aussi (k-means++)

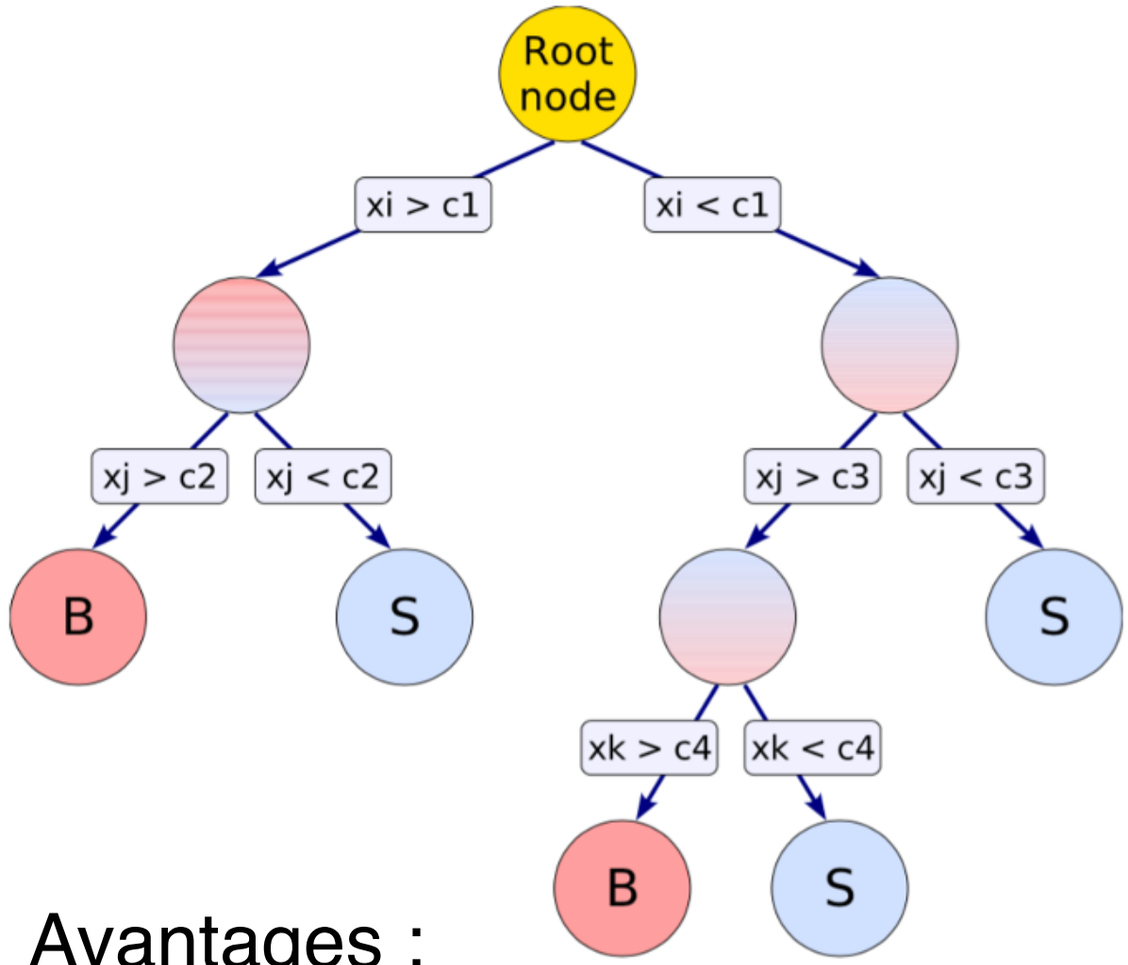




Quelque exemples : Supervisé

Classification : **Arbre de décision**

- Outil central de la physique des particules depuis les années 90
- Classifie des objets décris par N **variables** x_i



Construction de l'arbre :

- **Coupures** successives sur les variables pour maximiser la « pureté » des nœuds suivants
- Répète l'opération jusqu'à atteindre une pureté (ou un nombre de nœuds) cible

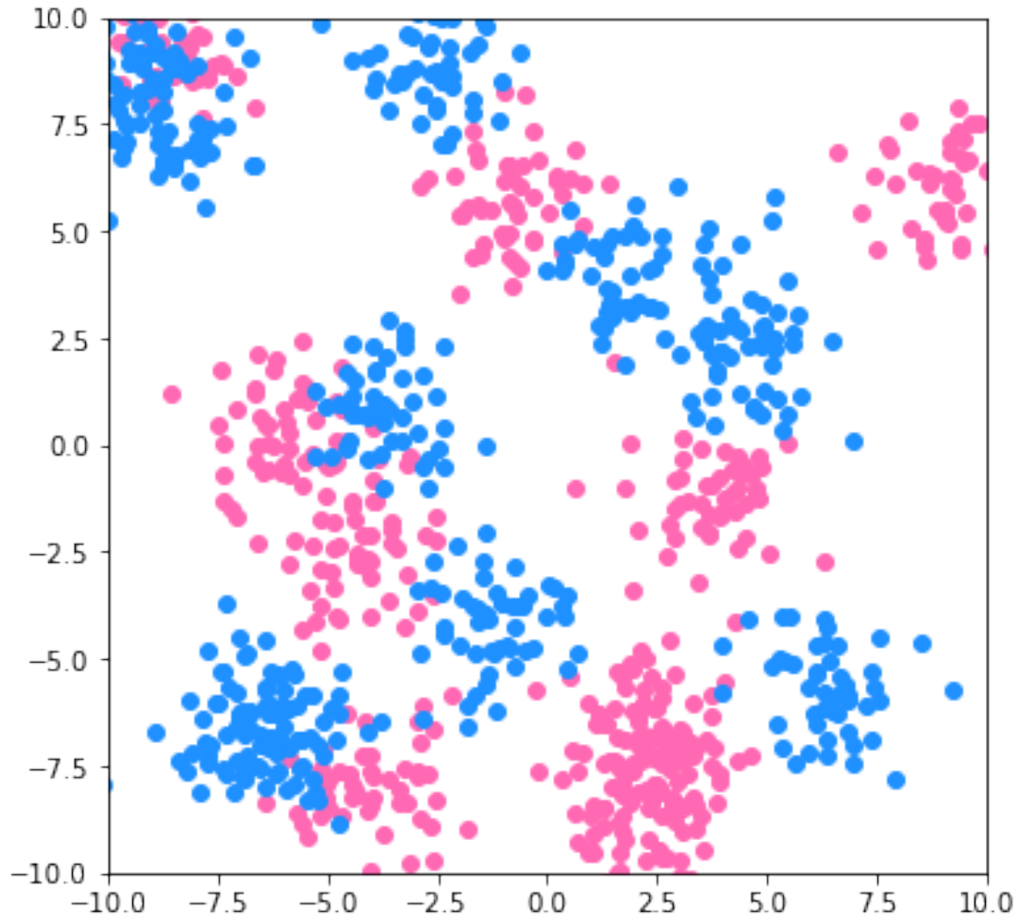
Avantages :

- Simple à **interpréter**
- Demande peu de données

- Désavantages :
  - Risque d'overfit
  - Création de l'arbre instable

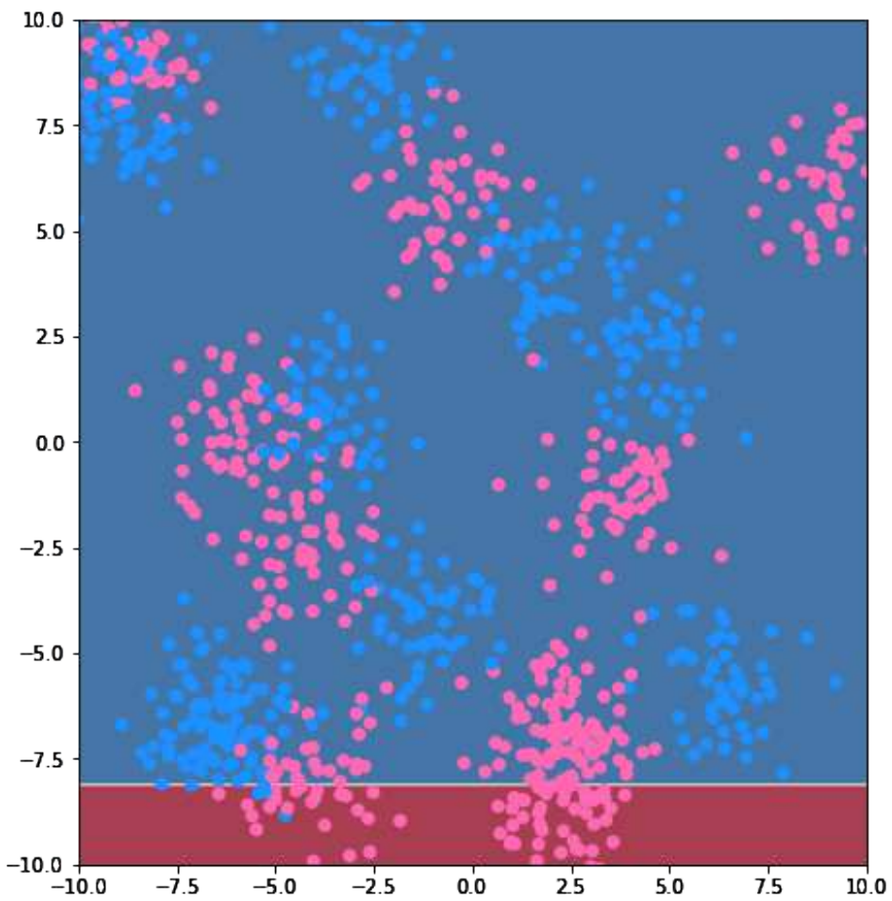
Quelque exemples : Supervisé

Classification : **Arbre de décision**



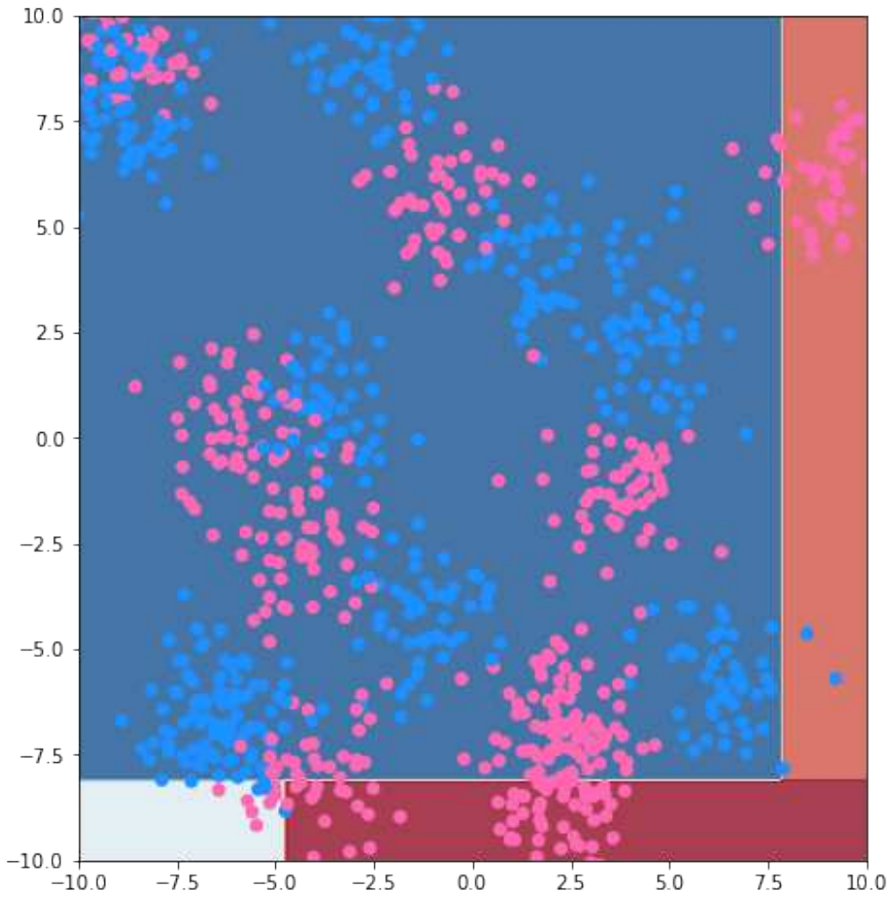
Quelque exemples : Supervisé

Classification : **Arbre de décision**



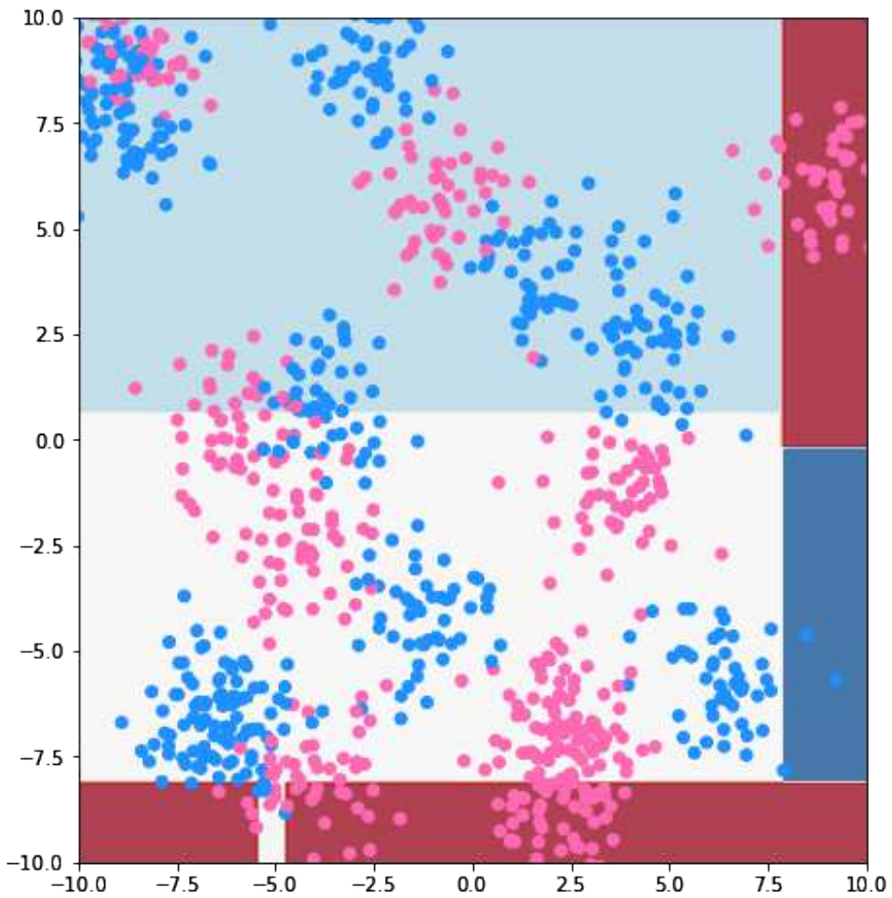
Quelque exemples : Supervisé

Classification : **Arbre de décision**



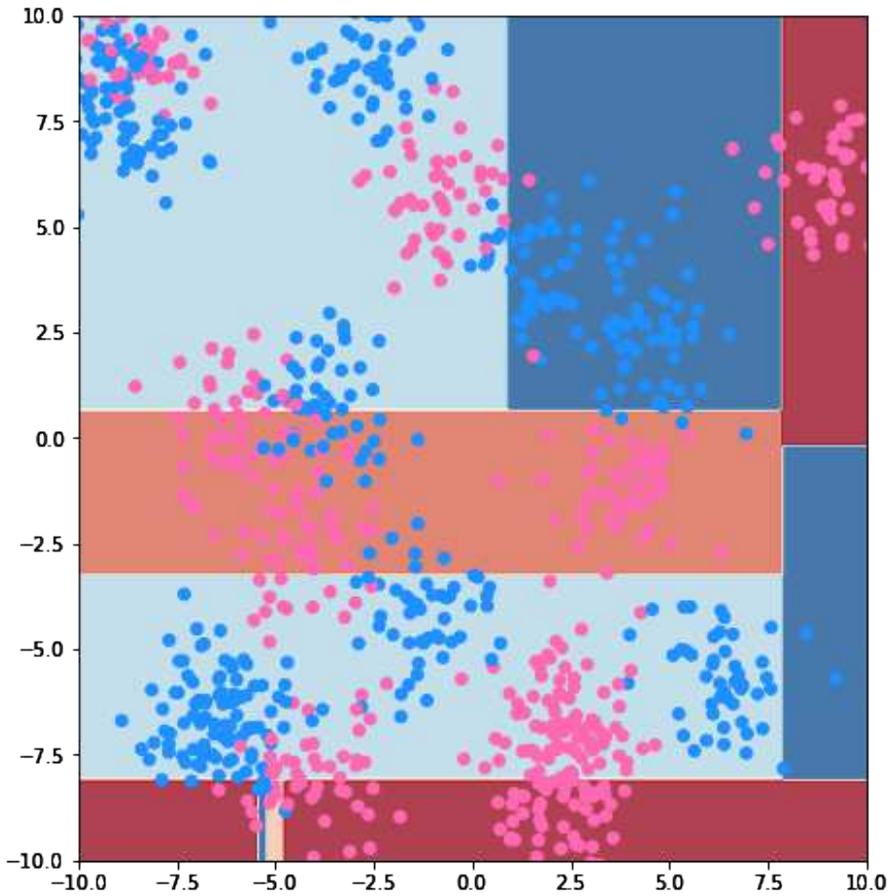
Quelque exemples : Supervisé

Classification : **Arbre de décision**



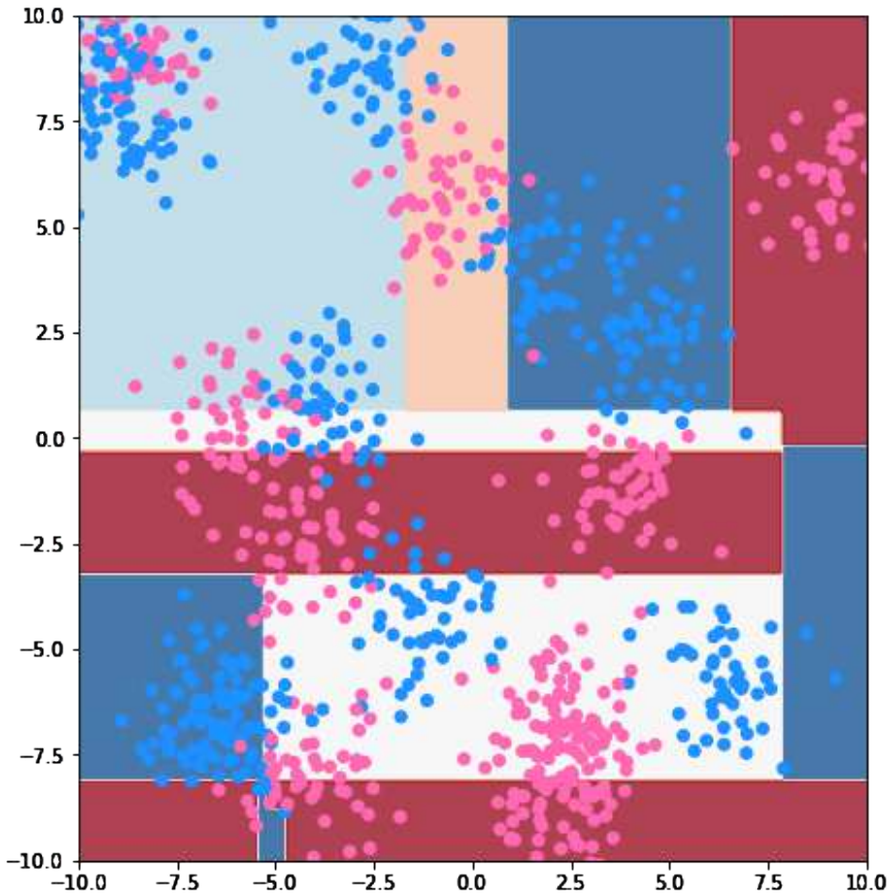
Quelque exemples : Supervisé

Classification : **Arbre de décision**



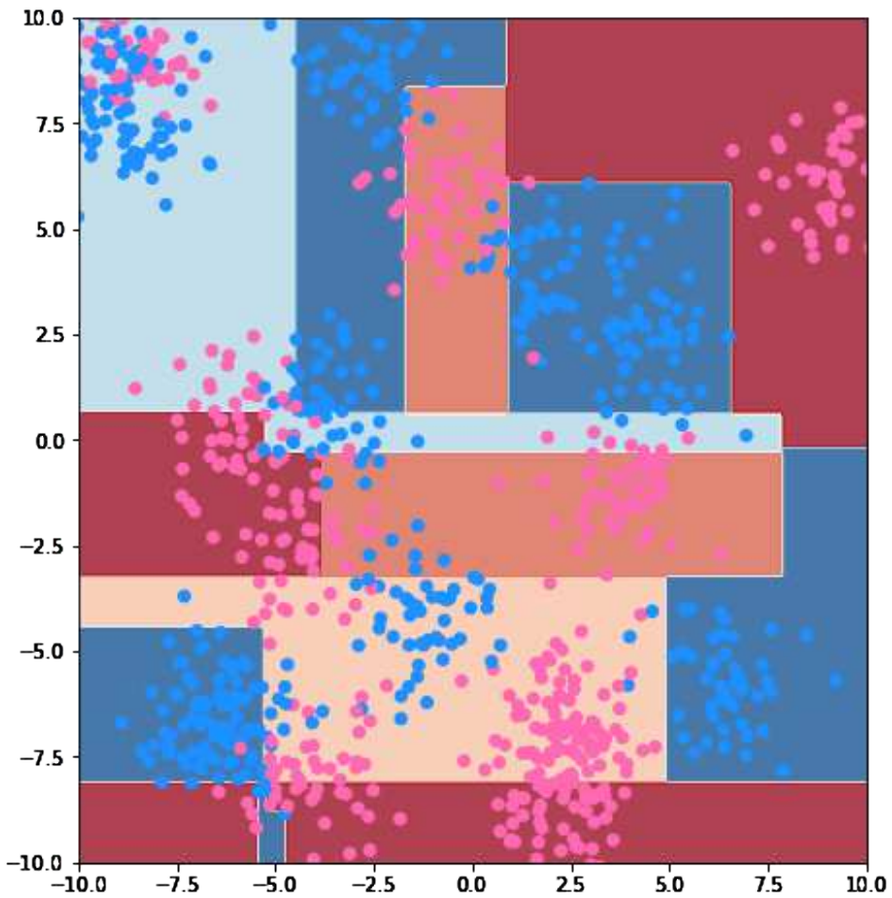
Quelque exemples : Supervisé

Classification : **Arbre de décision**



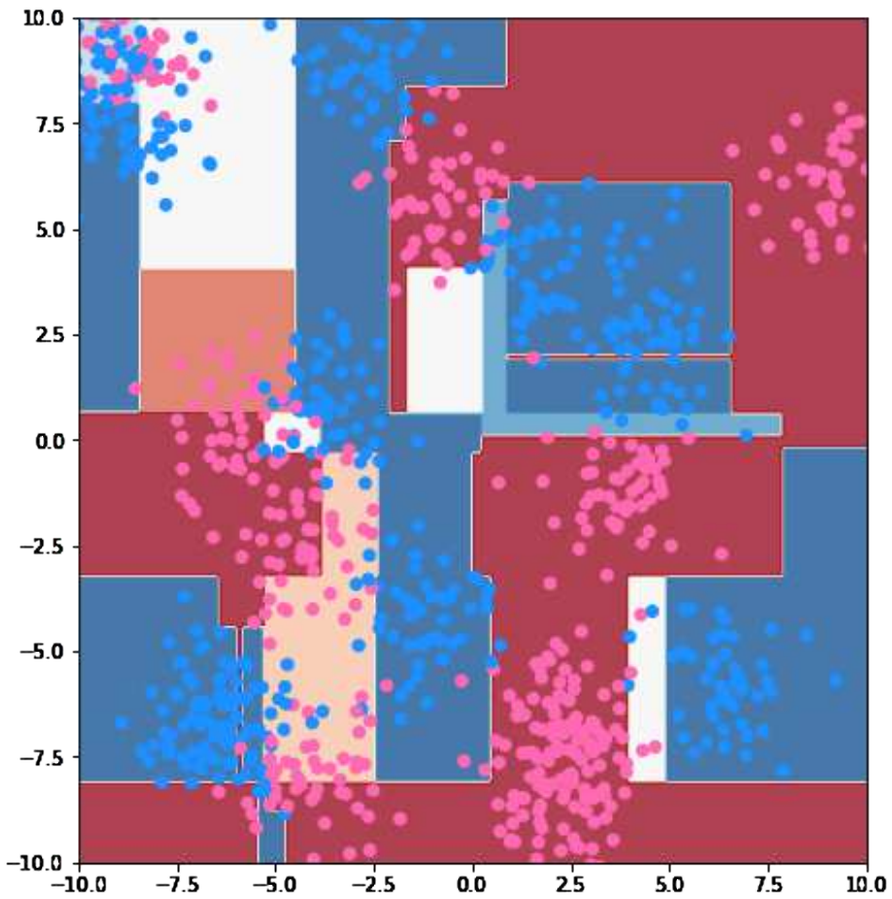
Quelque exemples : Supervisé

Classification : **Arbre de décision**



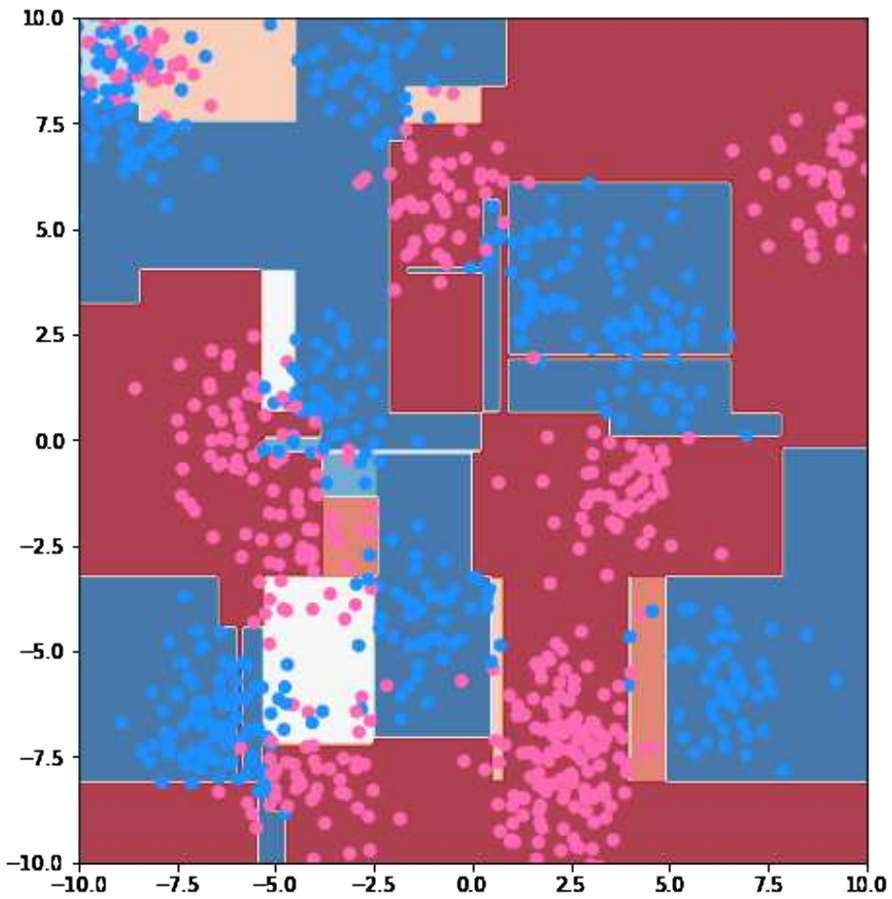
Quelque exemples : Supervisé

Classification : **Arbre de décision**



Quelque exemples : Supervisé

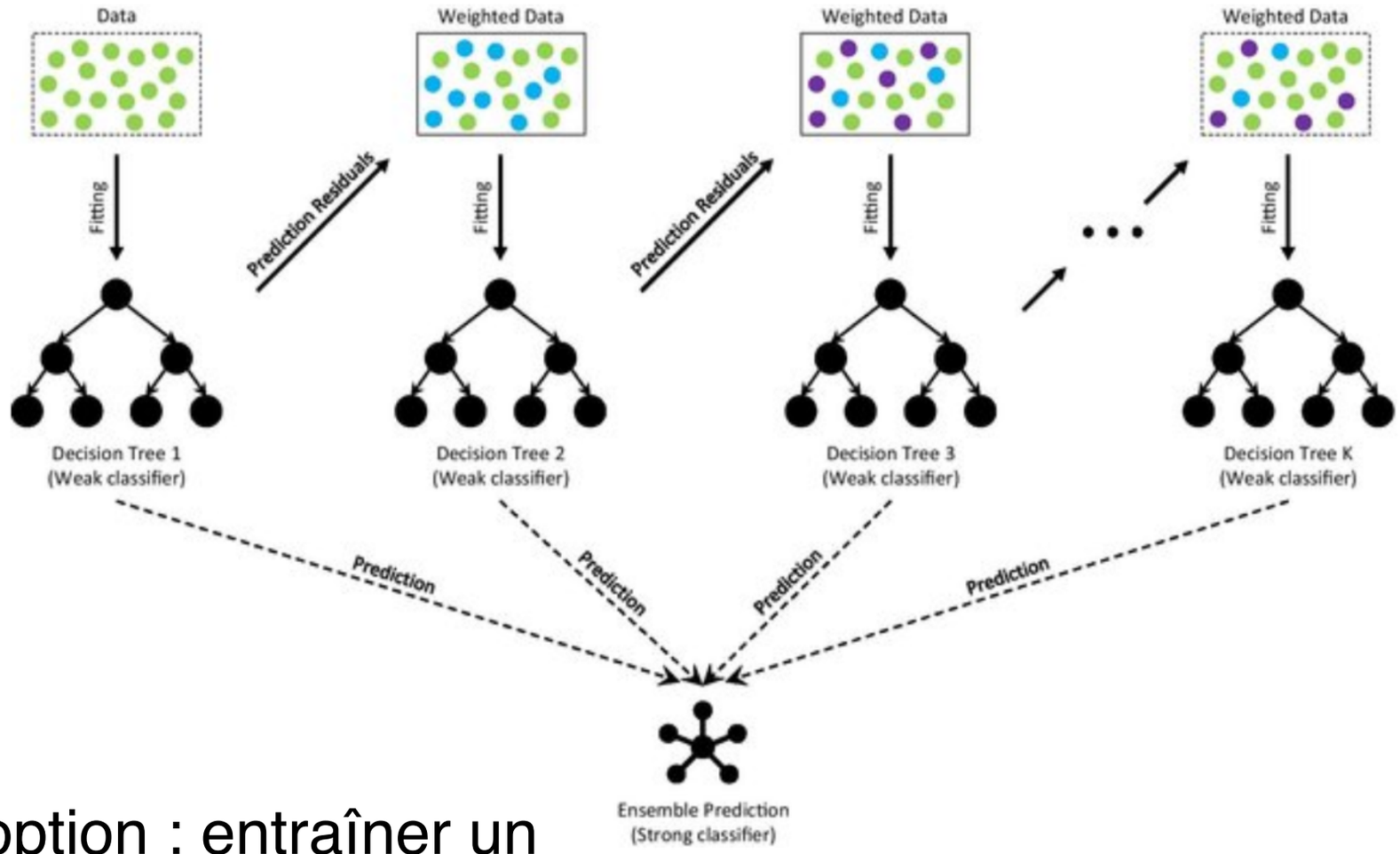
Classification : **Arbre de décision**



Quelque exemples : Supervisé

Classification : **BDT** (boosted decision tree)

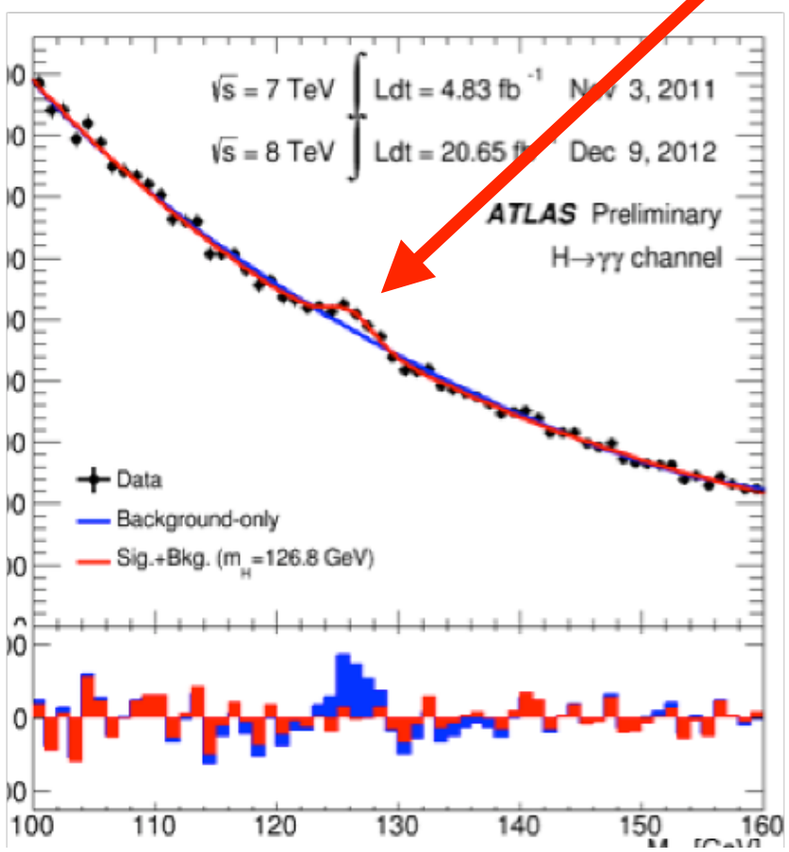
- Individuellement, les arbres peuvent **overfit** et sont très **instables**
- Utilise une **série d'arbres** pour optimiser la précision du système
- Solution idéale pour les problèmes à moins de 100 variables

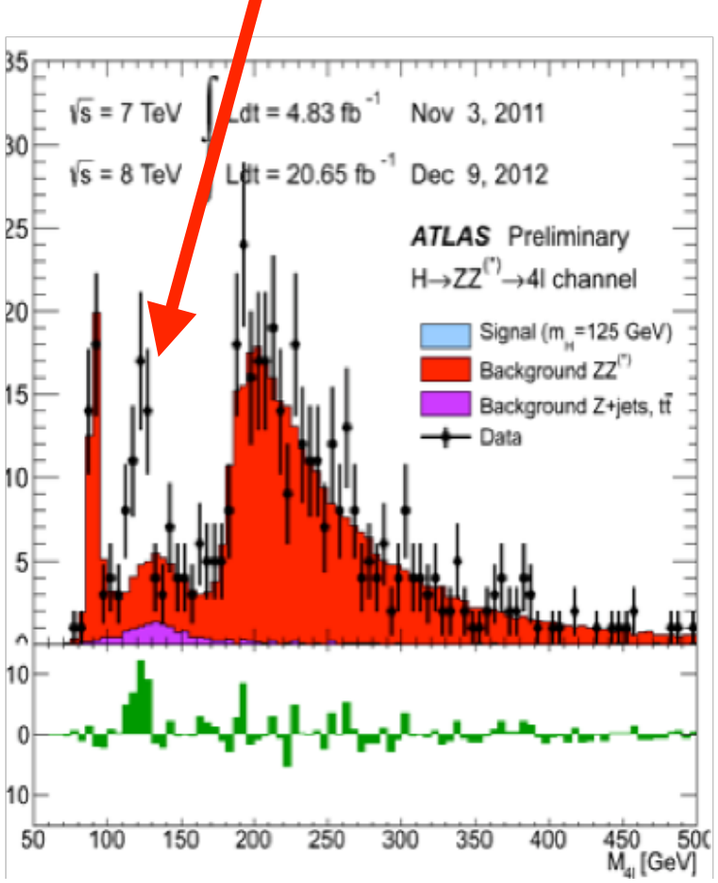


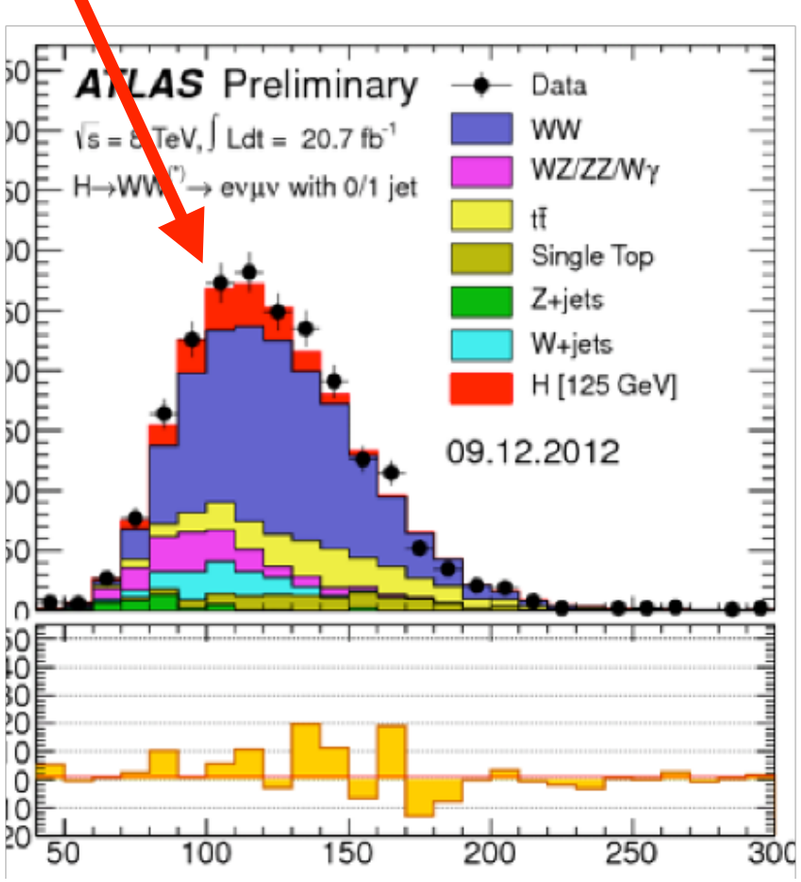
Autre option : entraîner un grand nombre d'arbres et effectuer un **vote**

Application à la recherche du Higgs

- Observation du boson de Higgs : Comparaisons de la **distribution** (histogramme) d'une variable d'intérêt observé au LHC (**signal**) avec sa valeur attendue en absence de Higgs (**bruit de fond**)
- La présence d'un **excès** dénote l'existence d'une nouvelle particule

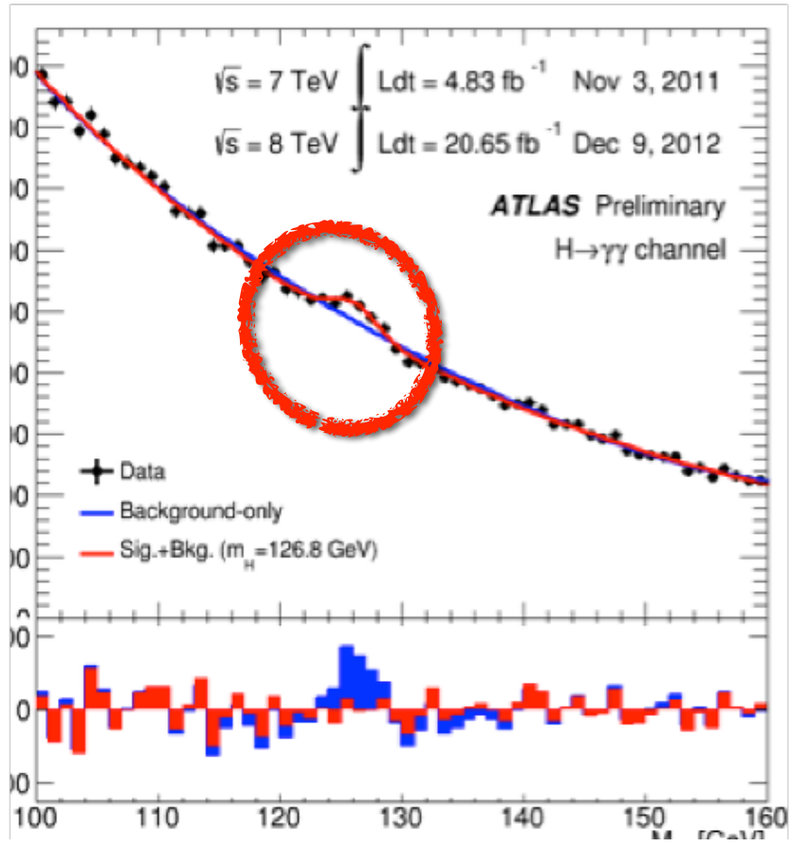


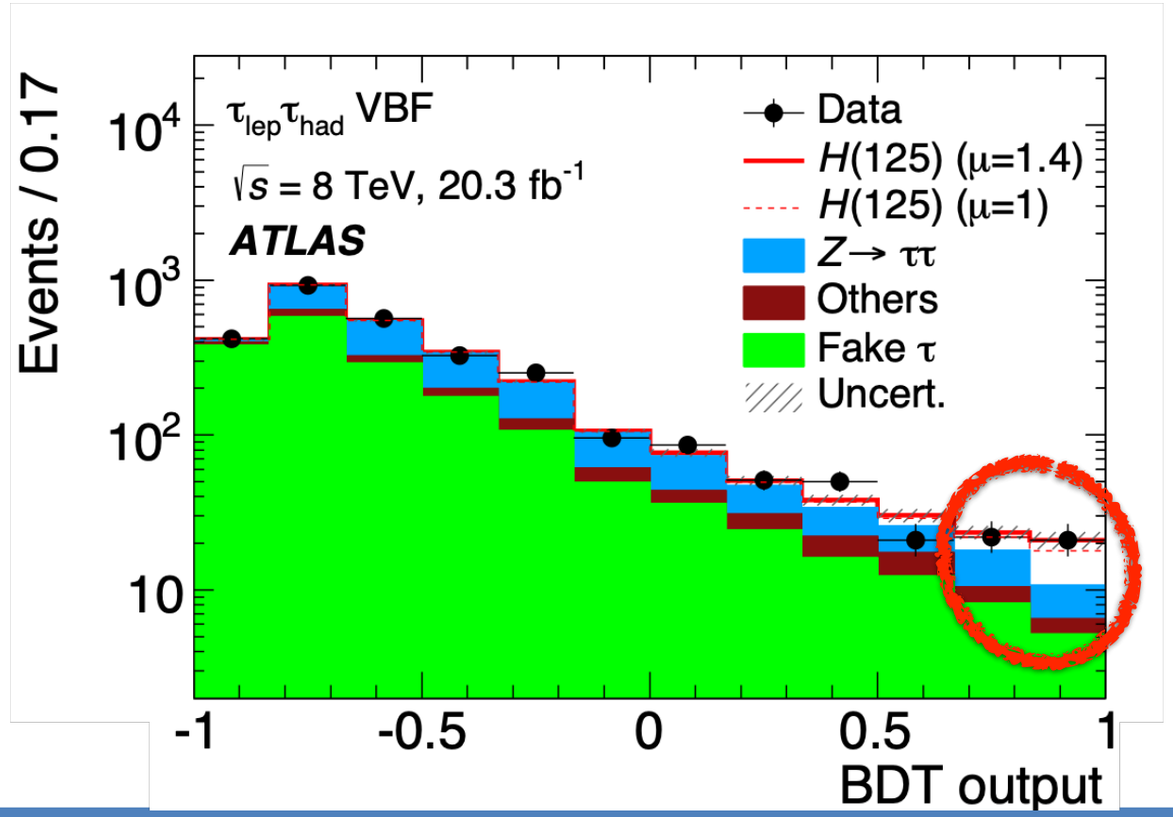




Choix de la variable d'intérêt

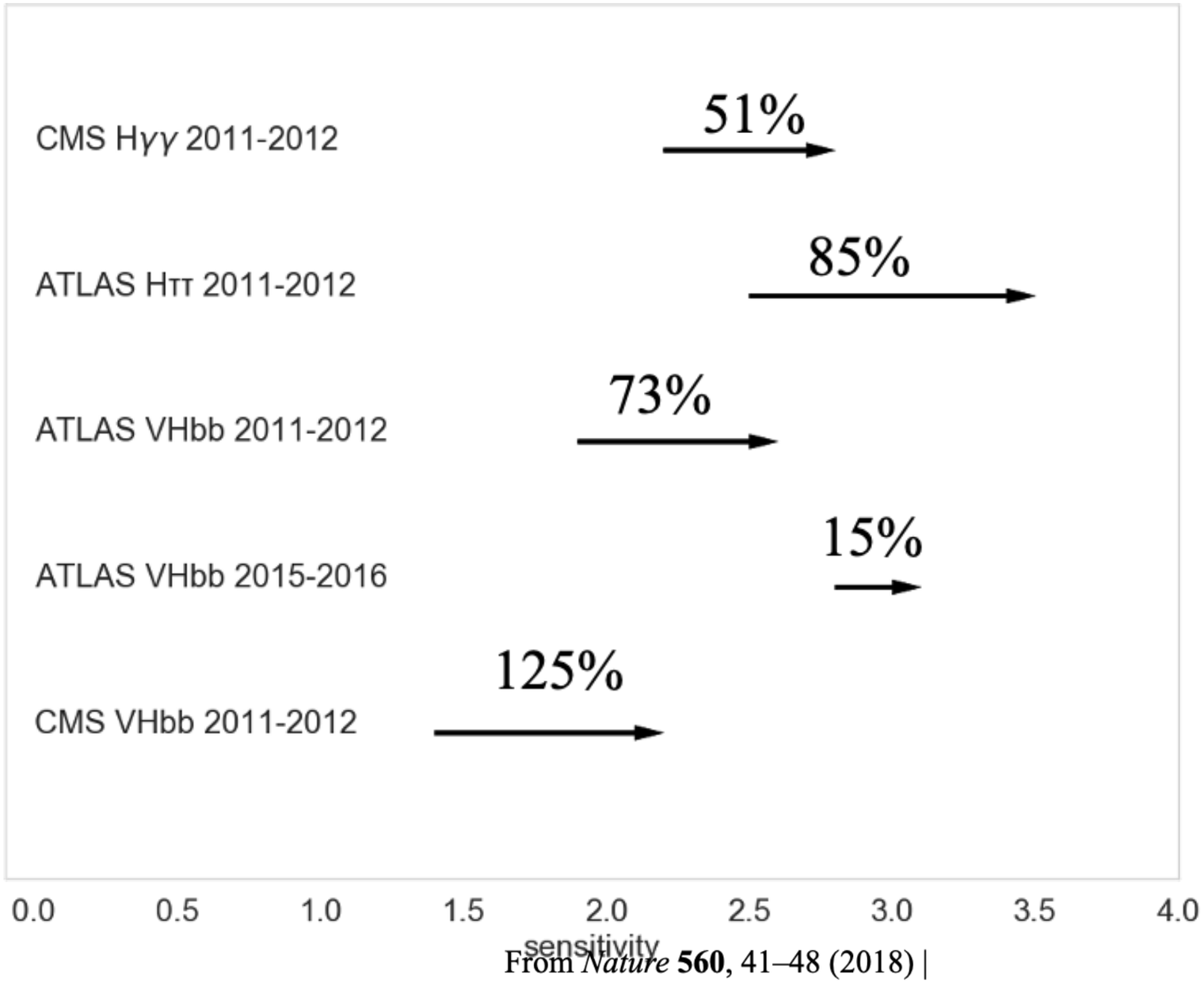
- Option évidente : **masse résonante** de production du **Higgs**
- « Facile » à reconstruire à partir de la cinématique des particules produite
- Nous donne une information sur les **propriétés** de la particule reconstruite





- Autre option : score de classification **BDT** des événements
- Le BDT essaie de séparer Higgs / non Higgs
- Idéale pour les **signatures complexes**
- Tire parti de plus de paramètres de l'événement

Pourquoi utiliser du ML ?



From *Nature* **560**, 41–48 (2018) |

- Principalement des **BDT** à **~10 variables**
- En moyenne, l'utilisation de BTD nous permet d'atteindre des **sensibilités** à l'existence du Higgs **~50%** plus élevés
- Maximisation de notre usage du LHC

Conclusion

- Le Machine Learning est un outil central pour les physiciens des particules
- Longue histoire d'utilisation : parmi les premiers à utiliser les BDT
- Demain : Le Deep Learning (ou apprentissage profond), toutes les techniques basées sur des réseaux de neurones pour extraire de l'information des données

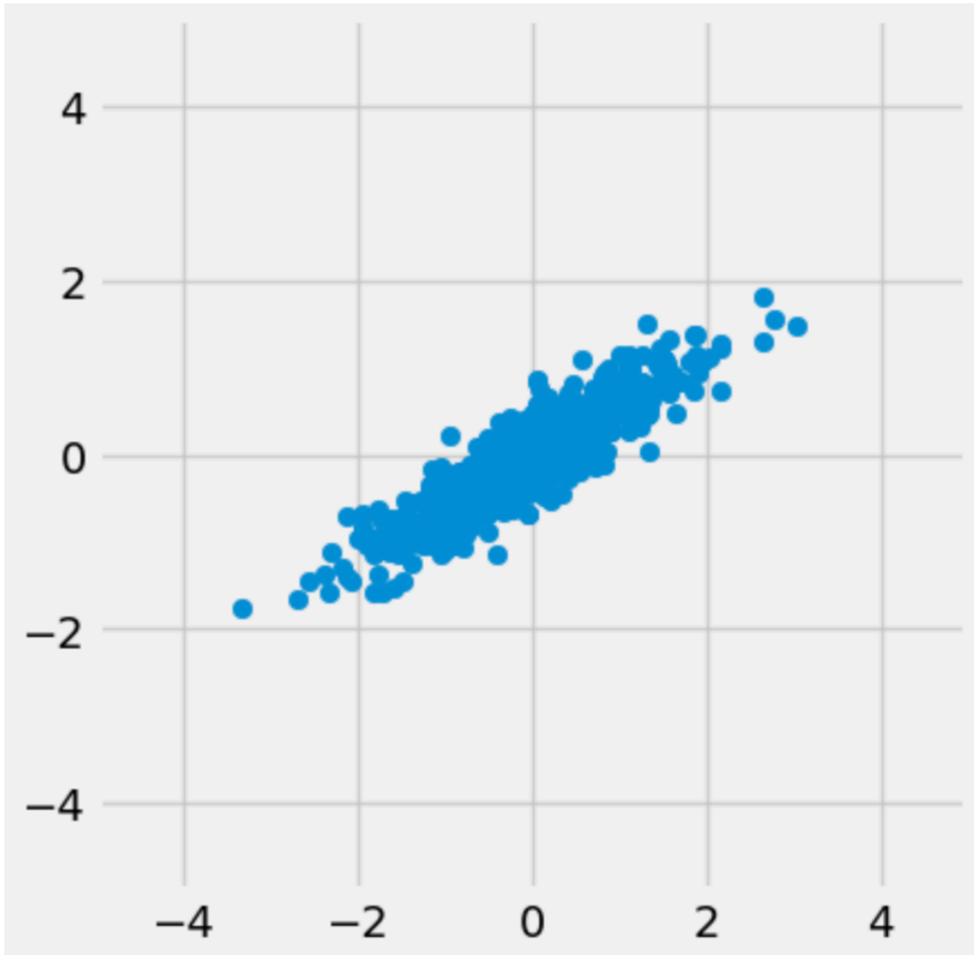
Pour aller plus loin

Analyse en composantes principales

En anglais : principal component analysis (**PCA**)

Objectif : détermination des axes favorables qui maximise la variance de nos données

➧ Trouve le premier axe avec la variance la plus large



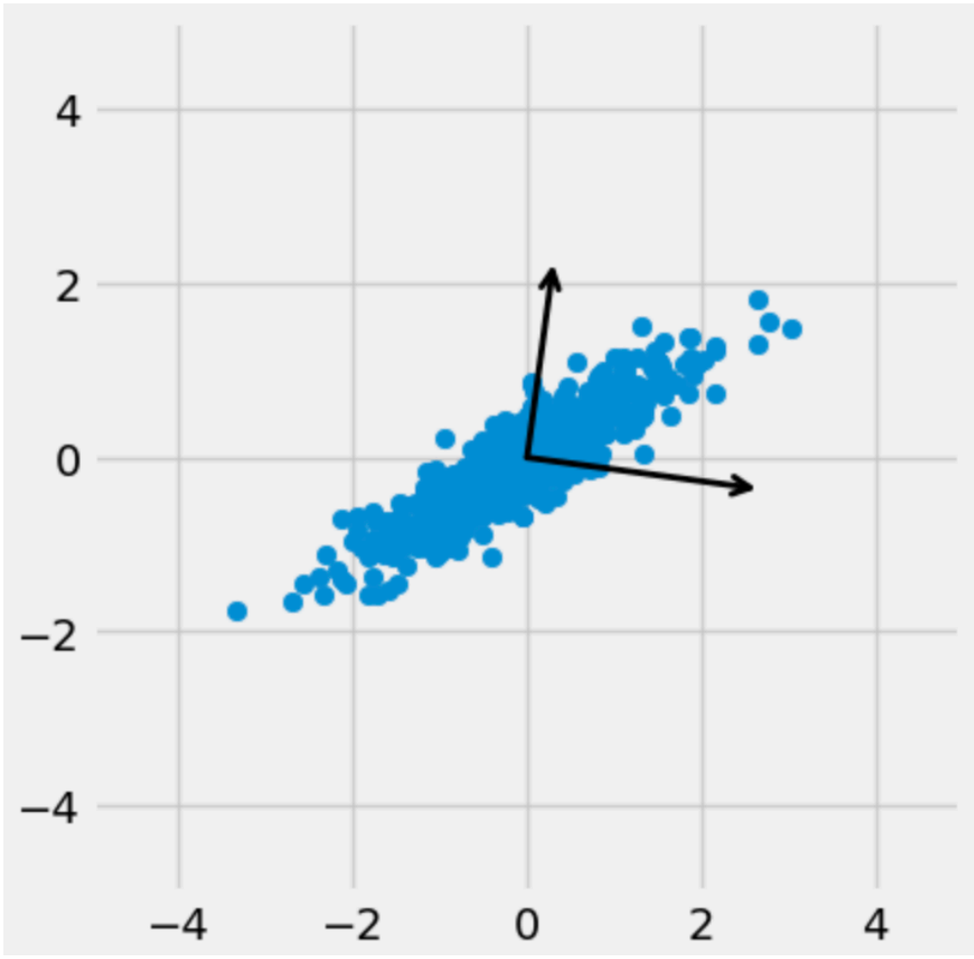
Analyse en composantes principales

En anglais : principal component analysis (**PCA**)

Objectif : détermination des axes favorables qui maximise la variance de nos données

➧ Trouve le premier axe avec la variance la plus large

➧ Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large

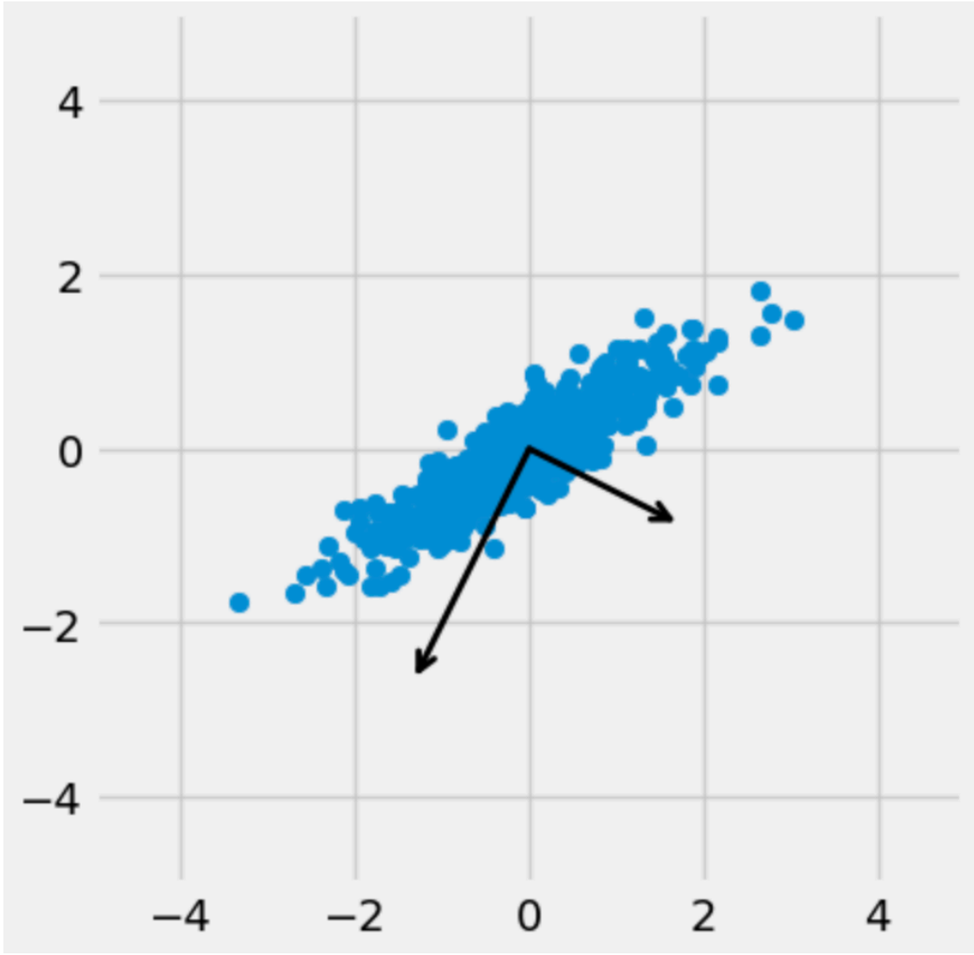


Analyse en composantes principales

En anglais : principal component analysis (**PCA**)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- Trouve le premier axe avec la variance la plus large
- Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- Continue jusqu'à avoir fait tous les axes

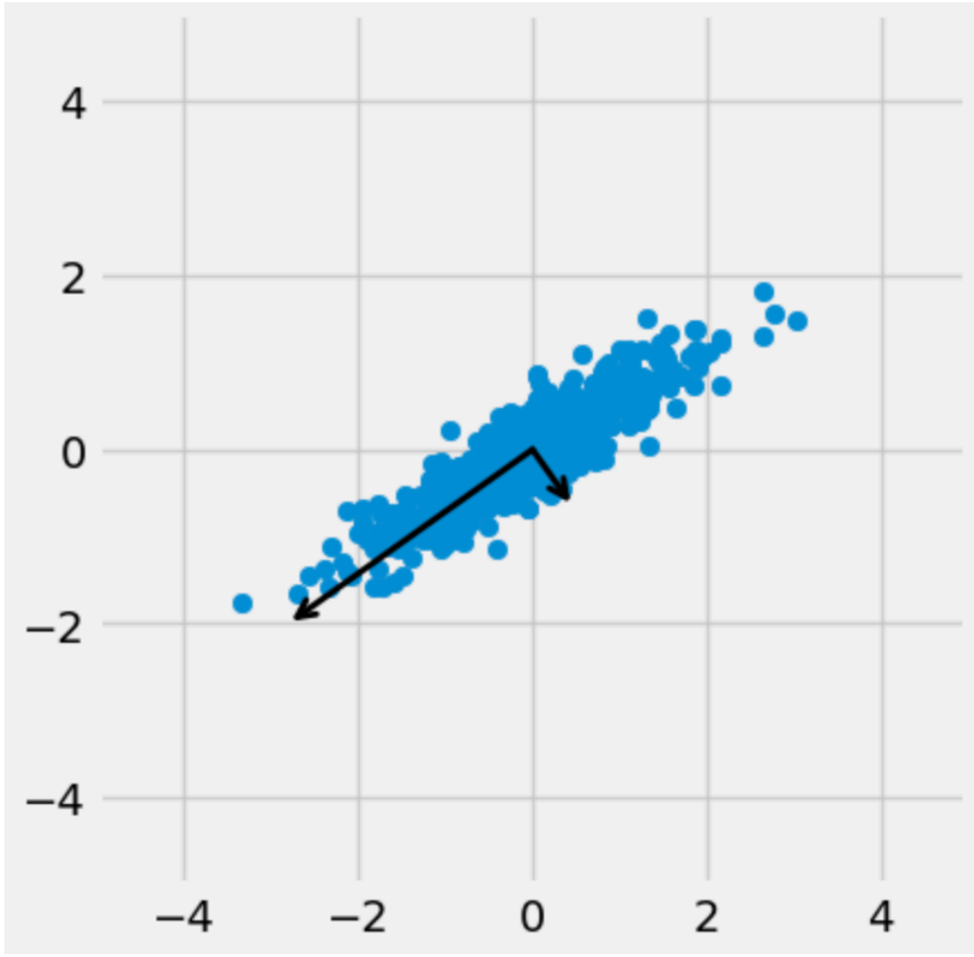


Analyse en composantes principales

En anglais : principal component analysis (**PCA**)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- Trouve le premier axe avec la variance la plus large
- Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- Continue jusqu'à avoir fait tous les axes

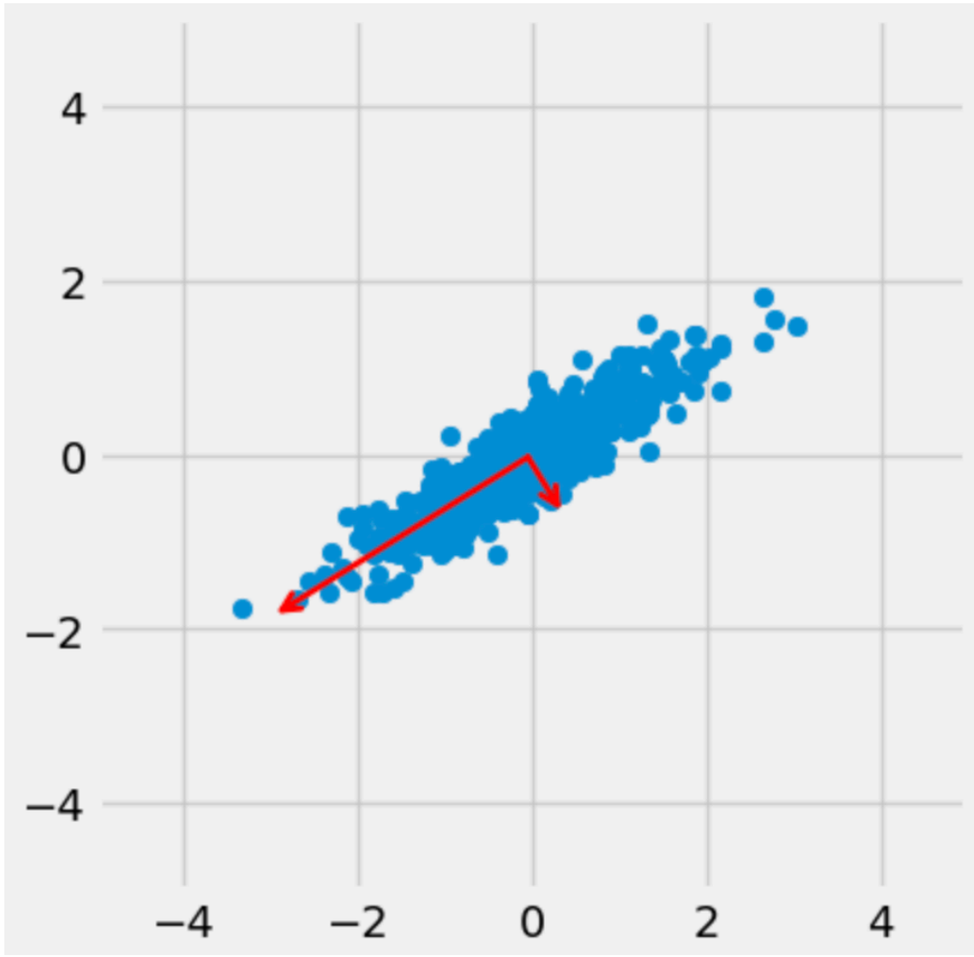


Analyse en composantes principales

En anglais : principal component analysis (**PCA**)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- Trouve le premier axe avec la variance la plus large
- Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- Continue jusqu'à avoir fait tous les axes



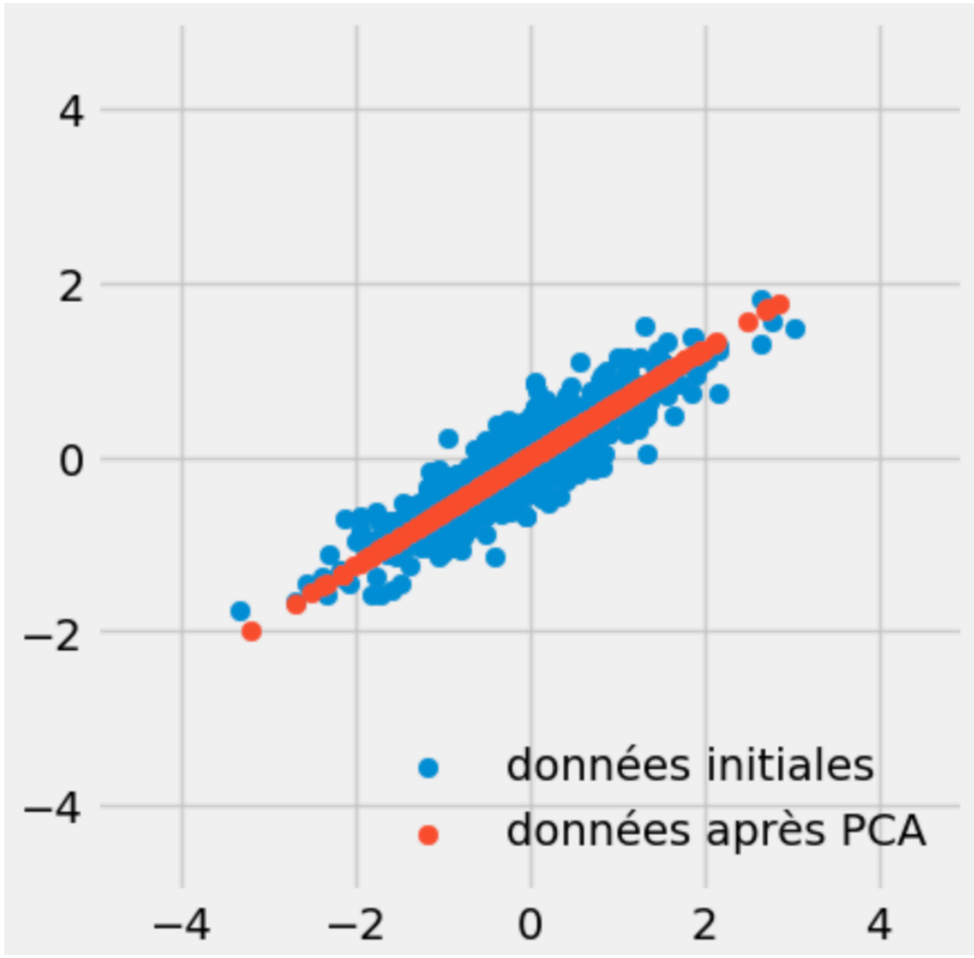
Analyse en composantes principales

En anglais : principal component analysis (**PCA**)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- Trouve le premier axe avec la variance la plus large
- Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- Continue jusqu'à avoir fait tous les axes

On peut ensuite supprimer les axes de faible variance en projetant nos données



Analyse en composantes principales

Mathématiquement :

Calcule la matrice de covariance des données initiale :

$$\begin{pmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \cdots & \mathrm{Cov}(X_1, X_p) \\ \mathrm{Cov}(X_2, X_1) & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_p, X_1) & \cdots & \cdots & \mathrm{Var}(X_p) \end{pmatrix}$$

Les données d'entrée doivent être normalisées

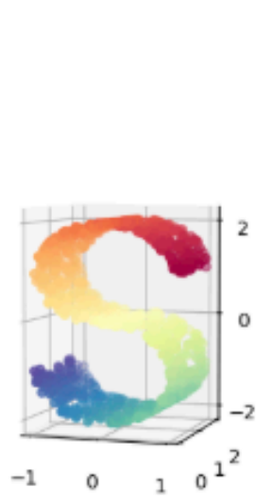
À partir de celles-ci, on trouve les vecteurs et valeurs propres ➔
Correspondent aux directions principales et la variance associée

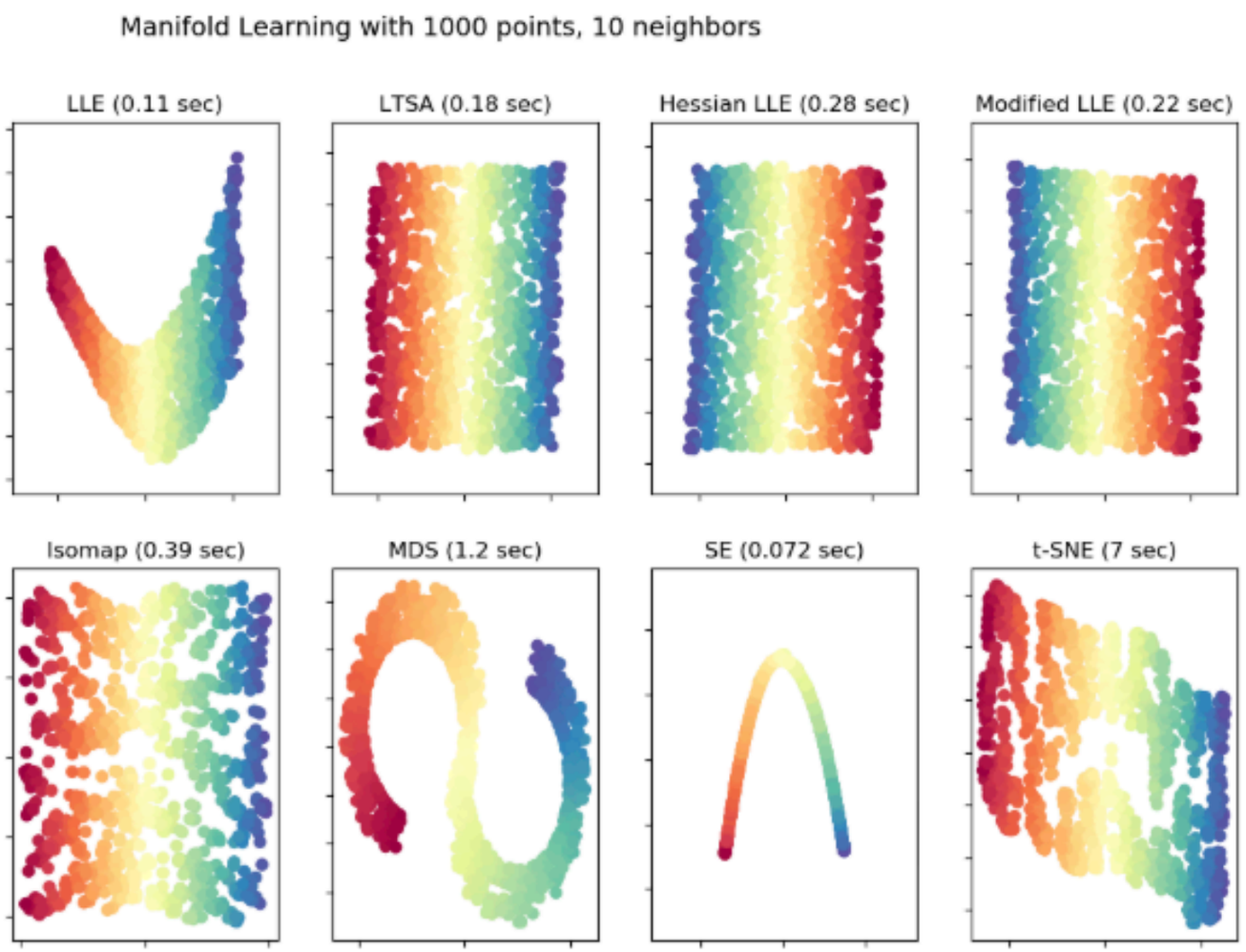
$$\mathrm{Var}(X) = \mathrm{Cov}(X, X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \qquad \mathrm{cov}(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y))$$

Manifold learning

La PCM est très rapide et efficace, mais elle ne fonctionne que les données sont séparables linéairement

Manifold learning : trouve une représentation 2D de donnée de haute dimension





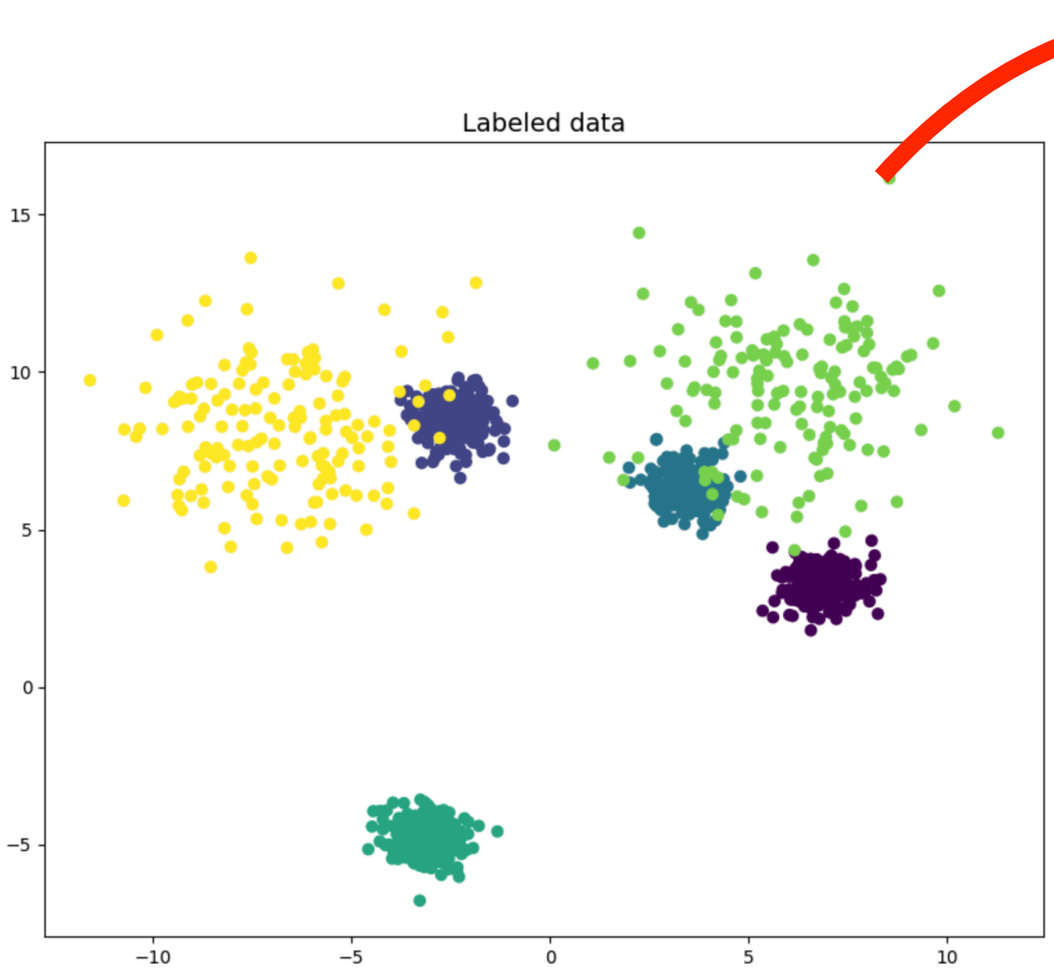
Méthode très utile pour visualiser simplement des données de haute dimension

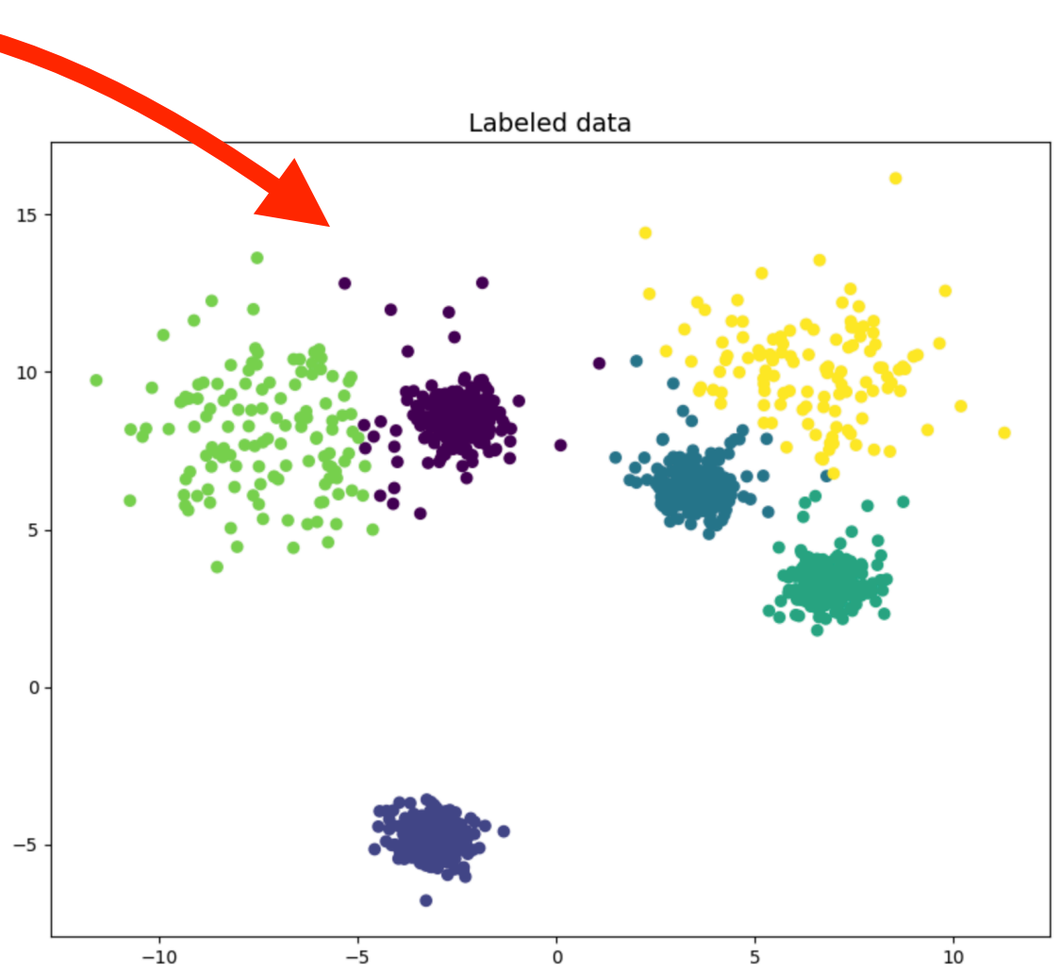
GaussianMixture

Inefficacité du k-mean pour les données in-homogènes

Données générées selon deux types de distribution différentes

Problème à l'interface des deux distributions



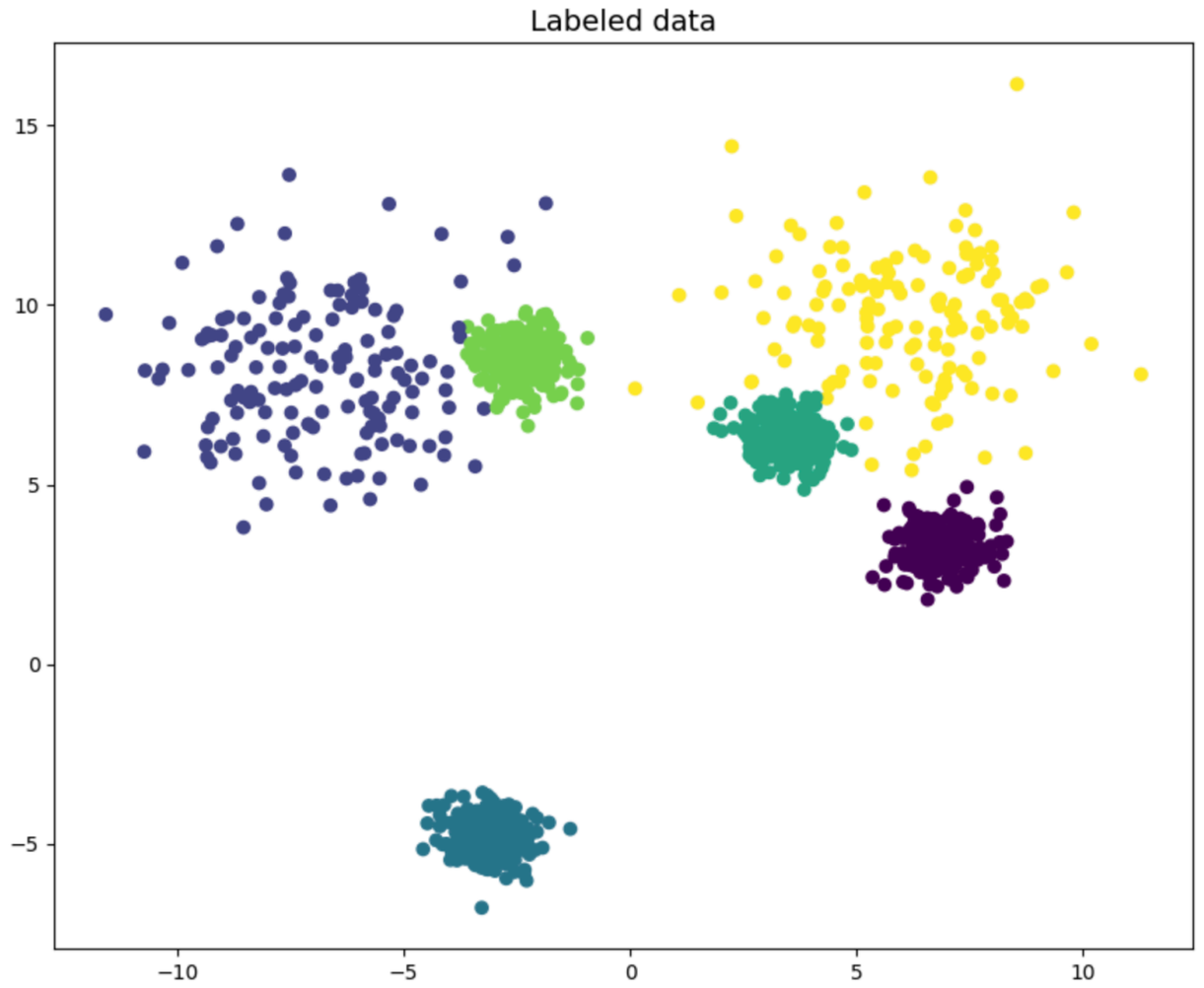


GaussianMixture

Gaussian mixture model

Assume que les données proviennent d'une combinaison de distributions gaussiennes

- Implémentation proche du k-mean : part de K distribution gaussienne
- Calcule la probabilité de chaque point d'avoir été générés par chaque distribution
- Mise à jour des poids des distributions pour maximiser la vraisemblance des données



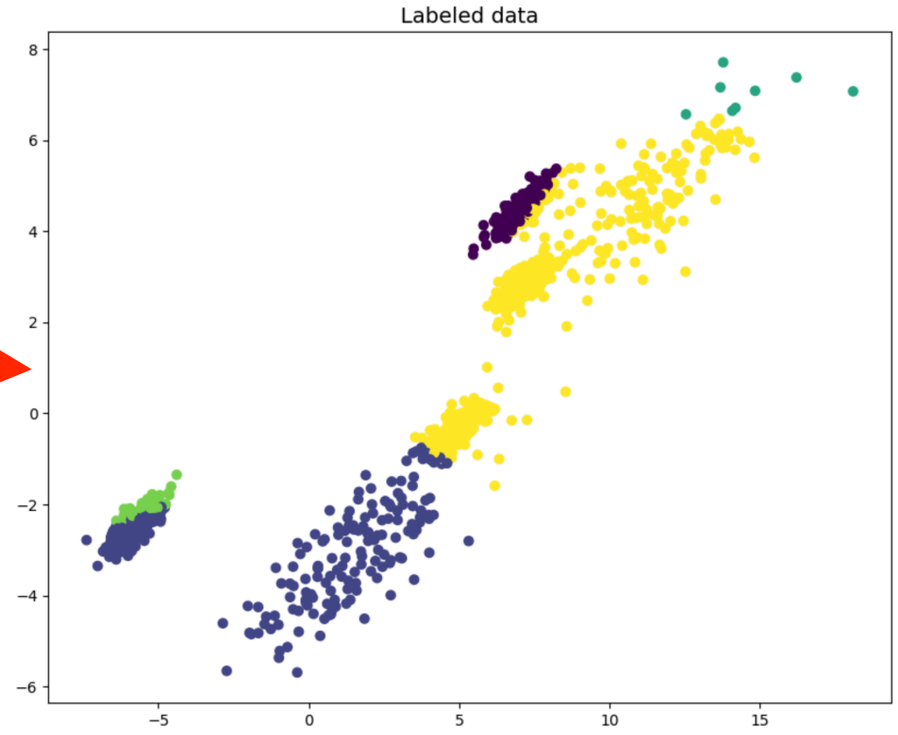
- Chaque point : probabilité d'appartenance aux différents clusters

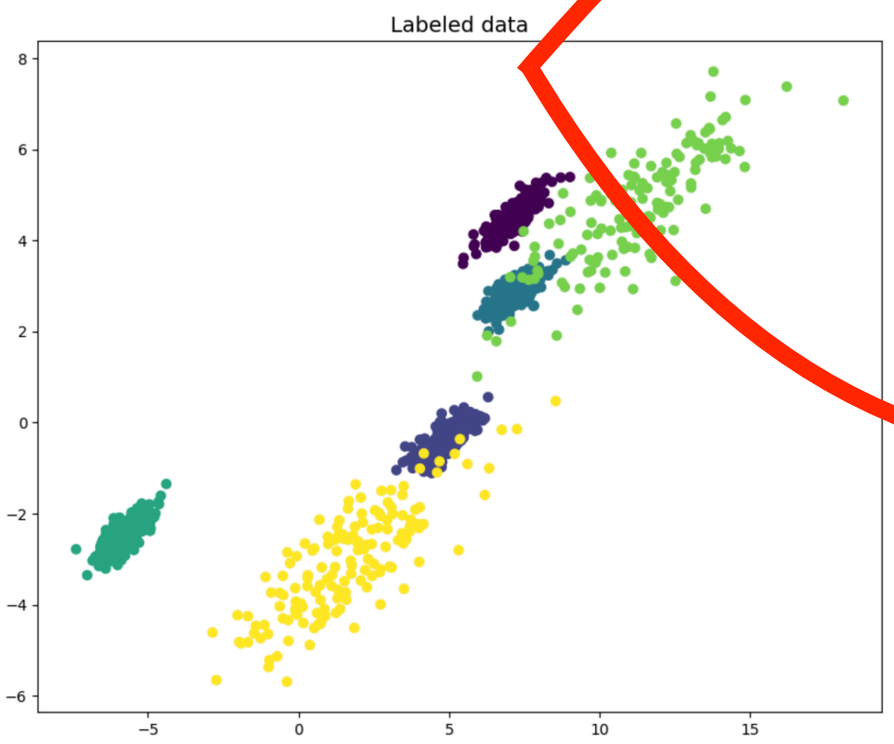
GaussianMixture

Données corrélées

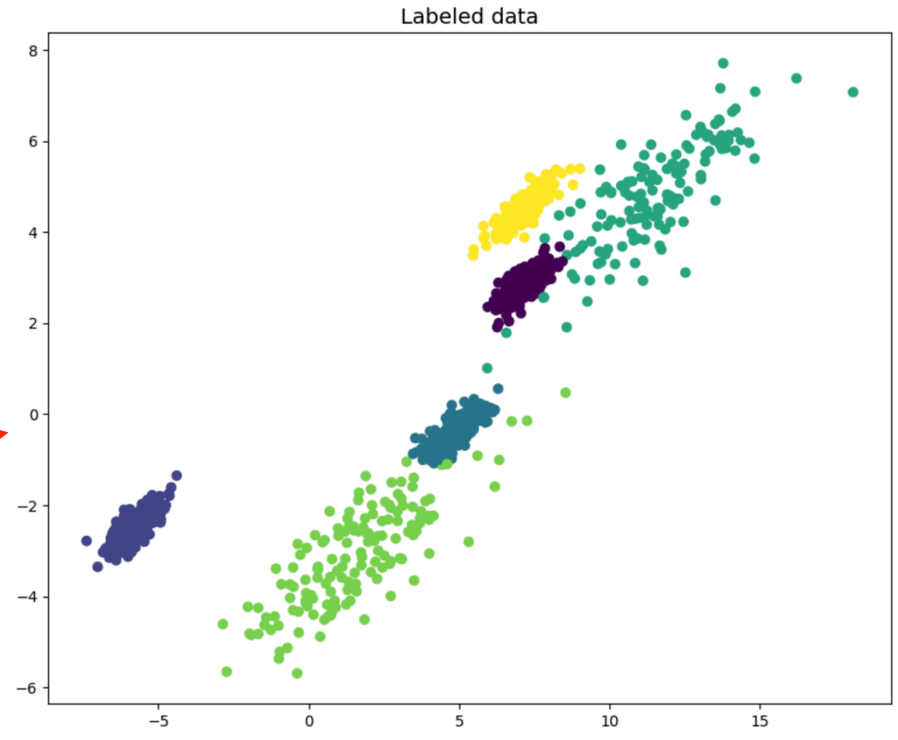
GMM bien meilleur pour les données corrélées (non circulaire)

kMean





GMM

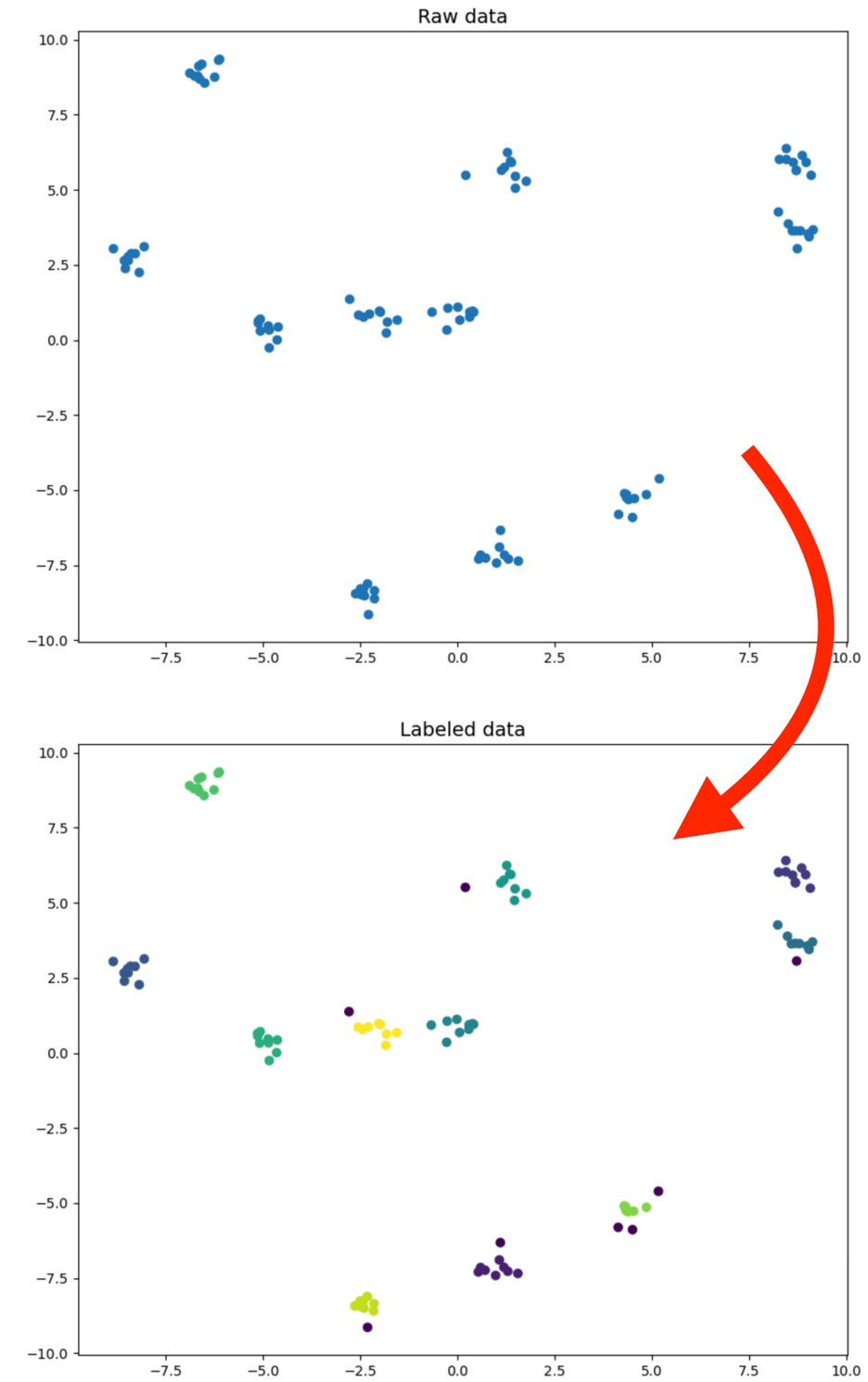


DBSCAN

Nombre de cluster libre

kMean, GMM ➔ On connait le nombre de clusters et on veut leurs propriétés

DBSCAN ➔ On ignore le nombre de clusters, mais on connait / peut approximer leurs propriétés



DBSCAN

Implementation :

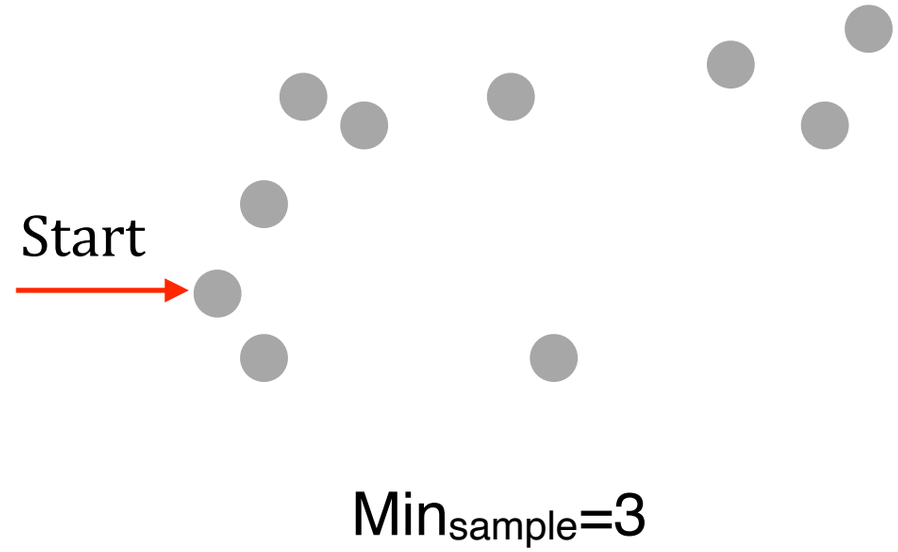
Basé sur la densité de donnée (ne marche pas pour des clusters superposés)

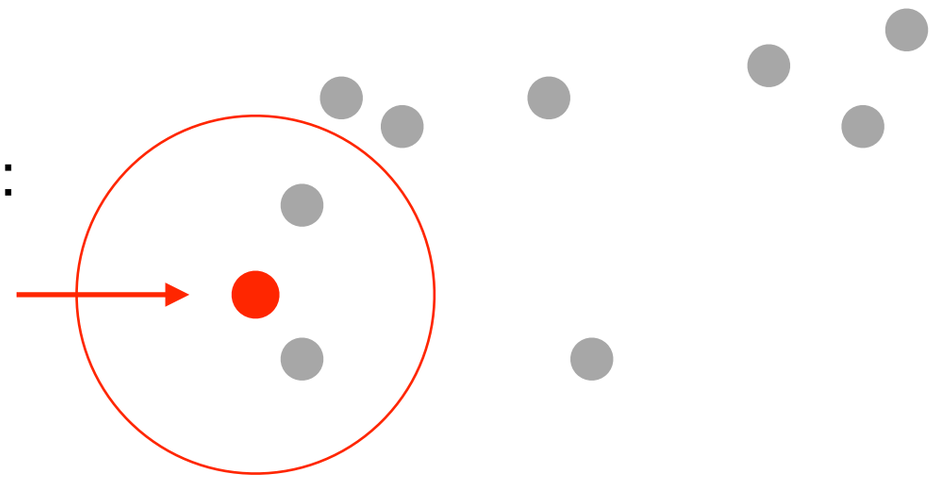
Deux paramètres d'entrée :

- ε : distance max entre 2 voisins
- Min_{sample} : nombre min de voisins

➡ Sélectionne un point pour commencer le clustering

➡ S'il a plus de Min_{sample} voisin à une distance ε : crée un cluster





DBSCAN

Implementation :

Basé sur la densité de donnée (ne marche pas bien pour des clusters superposés)

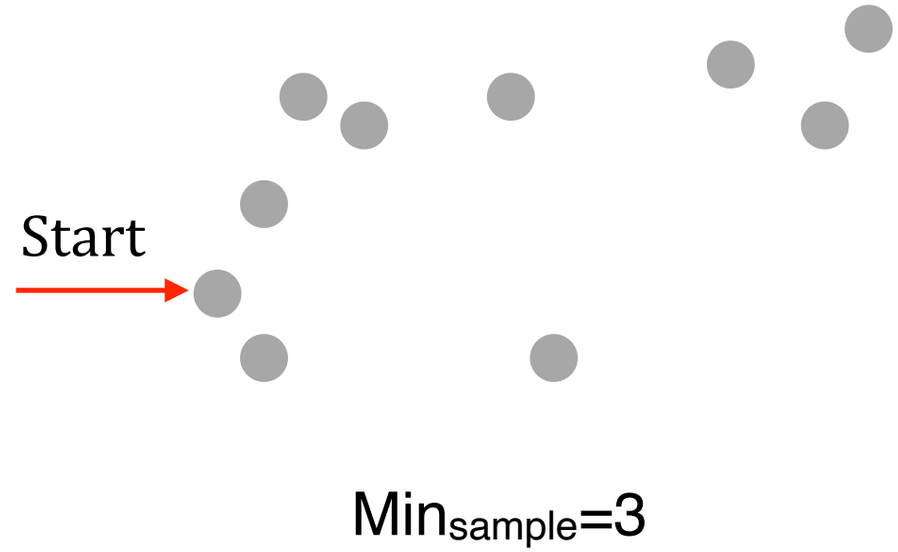
Deux paramètres d'entrée :

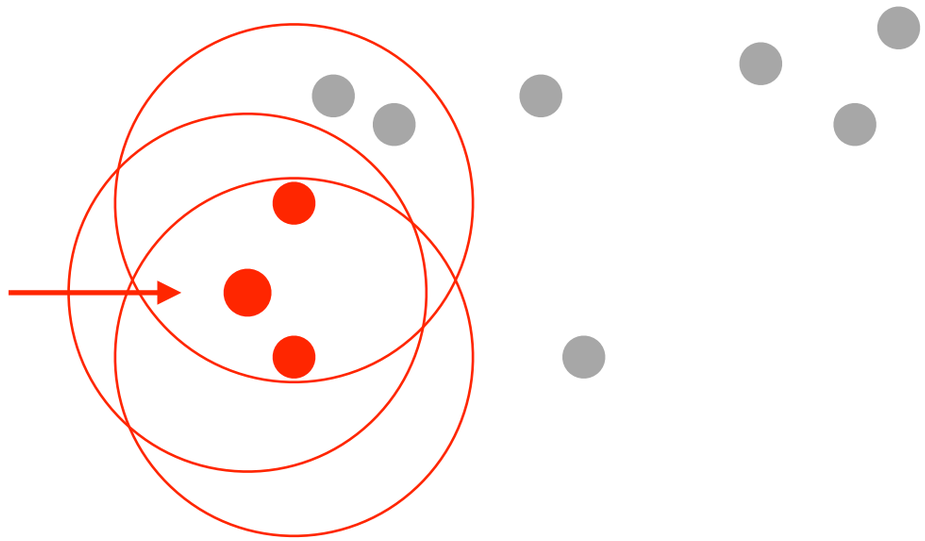
- ε : distance max entre 2 voisins
- Min_{sample} : nombre min de voisins

➡ Sélectionne un point pour commencer le clustering

➡Cluster : point plus voisin (d< ε)

➡Répète le processus pour les voisins





DBSCAN

Implementation :

Basé sur la densité de donnée (ne marche pas bien pour des clusters superposés)

Deux paramètres d'entrée :

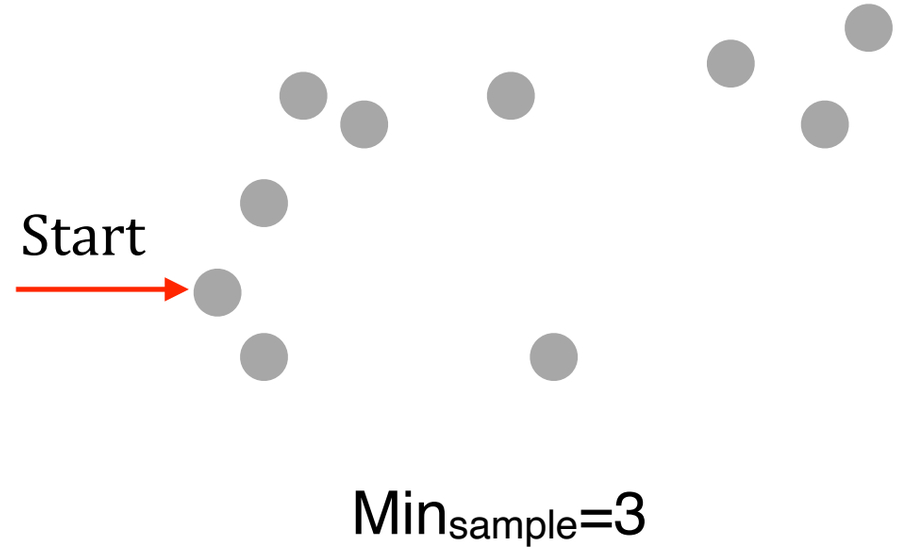
- ε : distance max entre 2 voisins
- Min_{sample} : nombre min de voisins

➡ Sélectionne un point pour commencer le clustering

➡Cluster : point plus voisin (d< ε)

➡Répète le processus pour les voisins

➡Une fois le cluster fini continu avec un autre point



DBSCAN

Implementation :

Basé sur la densité de donnée (ne marche pas bien pour des clusters superposés)

Deux paramètres d'entrée :

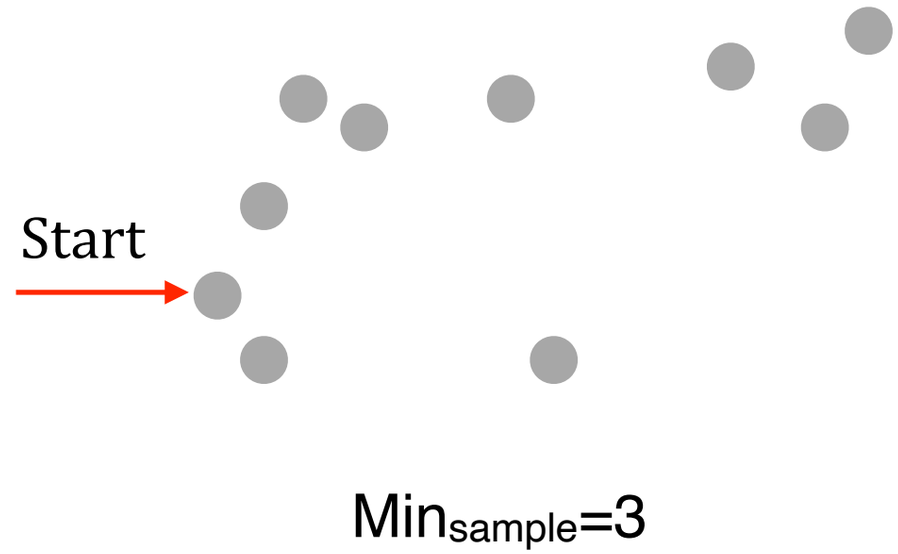
- ε : distance max entre 2 voisins
- Min_{sample} : nombre min de voisins

➡ Sélectionne un point pour commencer le clustering

➡Cluster : point plus voisin (d< ε)

➡Répète le processus pour les voisins

➡Une fois le cluster fini continu avec un autre point



DBSCAN

Nombre de cluster libre

kMean, GMM ➔ On connait le nombre de clusters et on veut leurs propriétés

DBSCAN ➔ On ignore le nombre de clusters, mais on connait / peut approximer leurs propriétés