



Session : Formation Dev IA

Matière : Introduction au Machine Learning

Date : 18.09.2023

Formateur : Balthazar Potet

Auteur : Balthazar Potet

# Machine Learning





## Pourquoi apprendre ?

- On veut un programme capable d'évoluer
  - Les spécifications d'un problème peuvent changer au cours du temps
- On ne sait pas faire autrement
  - Il y a des tâches pour lesquelles on ne connaît pas d'algorithme explicite



## Que signifie “apprendre” ?

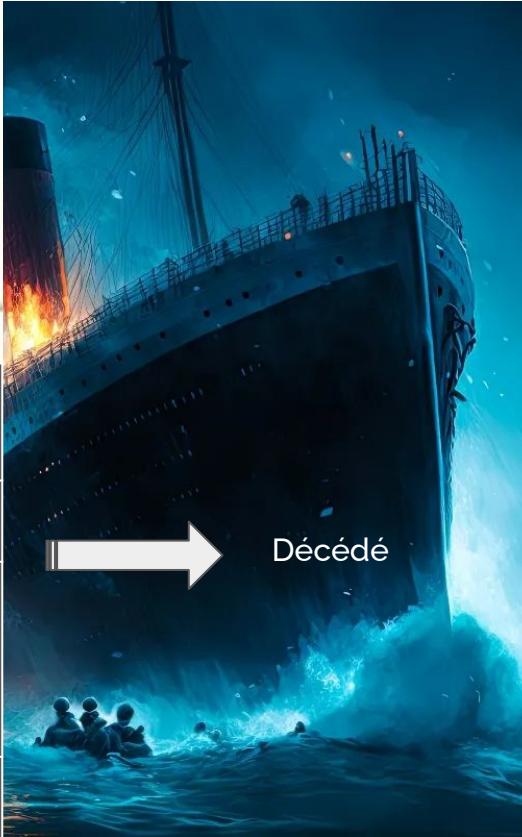


Améliorer ses performances à partir d'observations.

- “Améliorer...”
  - Algorithmes d'apprentissage
- “...ses performances...”
  - Tests de performance
- “...à partir d'observations.”
  - Collecte et traitement des données

# Les grandes étapes d'un projet de Machine Learning

1. Définir le problème
2. Collecter et traiter les données
3. Sélectionner une classe de modèles
4. Entraîner le modèle
5. Évaluer le modèle entraîné
6. Utiliser le modèle !

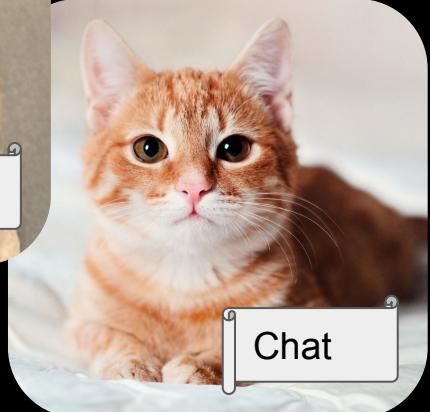


## Comment définir un problème ?

- En entrée, une représentation qui peut contenir des attributs (*features*) de types :
  - Nombres
  - Valeurs booléennes
  - Catégories
- En sortie, soit :
  - Une valeur prise dans un ensemble fini (💡 *Classification*)
  - Un nombre (⌚ *Régression*)

# Quelles techniques pour apprendre ?

- Apprentissage supervisé
  - Les observations sont étiquetées (*labeled*), l'agent apprend la relation entre les observations et leurs étiquettes.
- Apprentissage non supervisé
  - L'agent apprend des motifs présents dans les observations.
- Apprentissage par renforcement
  - L'agent reçoit un retour positif ou négatif sur ce qu'il produit, il apprend à produire des résultats qui apportent des récompenses



Apprentissage supervisé



Apprentissage non supervisé

# Apprentissage supervisé

 Étant donné un *jeu de données d'entraînement* de  $N$  exemples de paires entrée-sortie :

$$(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$$

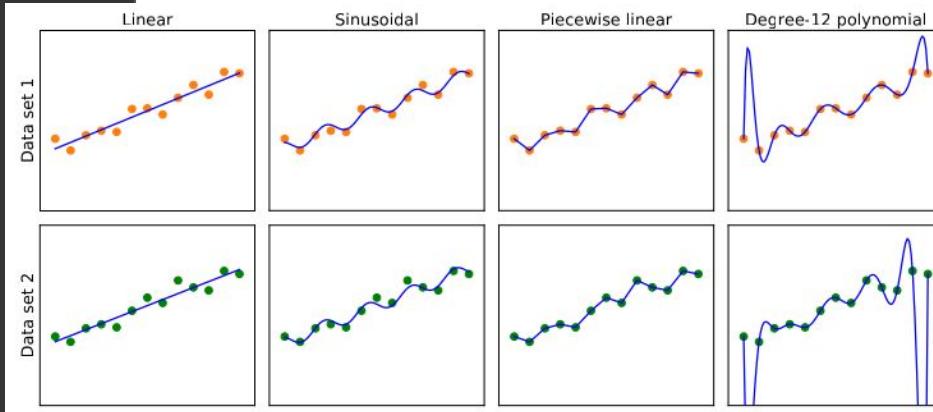
où chaque paire a été générée par une fonction inconnue  $f(x) = y$ , découvrir une fonction  $h$  qui approxime la fonction  $f$ .

- $f$  est la *fonction cible*.
- $h$  est appelée *hypothèse* ou *modèle*.
  - $h$  est choisie dans un *ensemble d'hypothèses* ou dans une *famille de modèles*.

# Séparation des jeux de données

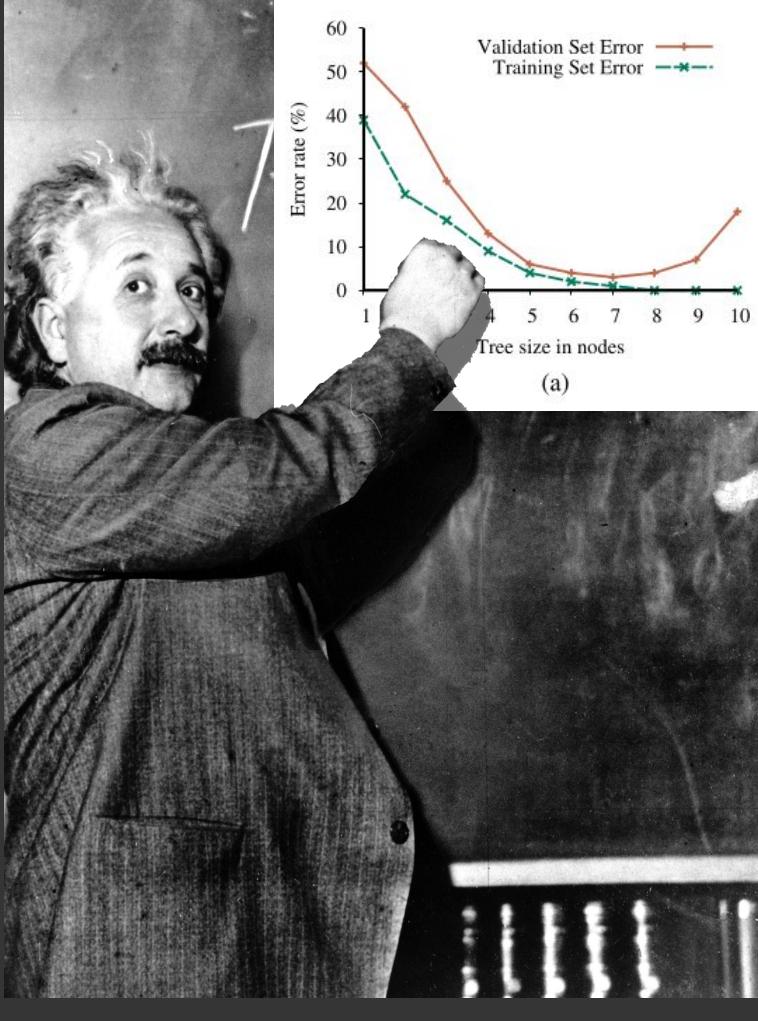
On veut évaluer la capacité d'un modèle à généraliser. On sépare donc notre jeu de données en plusieurs parties :

- *Jeu d'entraînement*
  - Sert à entraîner le modèle.
- *Jeu de validation (optionnel)*
  - Sert à comparer les performances de différents modèles.
- *Jeu de test*
  - Sert à évaluer le modèle final.
  - Il est crucial que les observations de ce jeu ne soient jamais utilisées lors de l'entraînement !



## Biais et variance

- Le *biais* est la tendance d'une hypothèse à dévier de la fonction cible (indépendamment du jeu d'entraînement).
  - Lorsqu'une classe de modèles a un biais élevé, on parle d'*underfitting*.
- La *variance* est une mesure des changements dans l'hypothèse suite à des fluctuations dans les données d'entraînement.
  - Lorsqu'une classe de modèles a une variance élevée, on parle d'*overfitting*.



## Le compromis biais/variance

*Rendez les choses aussi simples que possible, mais pas plus simple.*

- Albert Einstein (paraphrasé)

# Comment sélectionner un modèle ?

On a plusieurs choix à faire :

- La classe de modèles
  - Arbres de décisions, régression linéaire, régression logistique, réseaux de neurone...
- Les hyperparamètres
  - Il s'agit des paramètres du modèle qui ne sont pas modifiés pendant l'entraînement (par exemple, la profondeur maximale d'un arbre de décision).

Méthode : entraîner plusieurs modèles avec différents hyperparamètres puis comparer leurs performances sur un jeu de validation pour sélectionner le meilleur.

# Comment évaluer un modèle ?

 Une *fonction erreur (loss function)* mesure la distance entre une prédiction  $\hat{y}$  et la valeur réelle  $y$ . Quelques exemples :

- Erreur absolue :

$$L_1(y, \hat{y}) = |y - \hat{y}|$$

- Erreur quadratique :

$$L_2(y, \hat{y}) = (y - \hat{y})^2$$

- Erreur discrète :

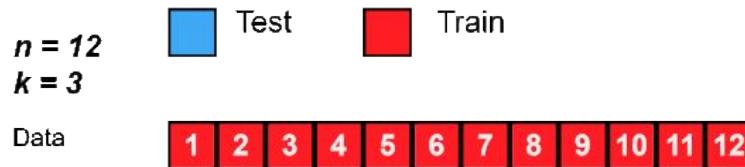
$$L_{0/1}(y, \hat{y}) = 0 \text{ si } y = \hat{y}, 1 \text{ sinon}$$

# Comment favoriser les modèles plus simples ?

 La *régularisation* consiste en l'ajout d'un terme supplémentaire à la fonction objectif. Ce terme mesure la complexité du modèle.

# Pas assez de données pour la validation ?

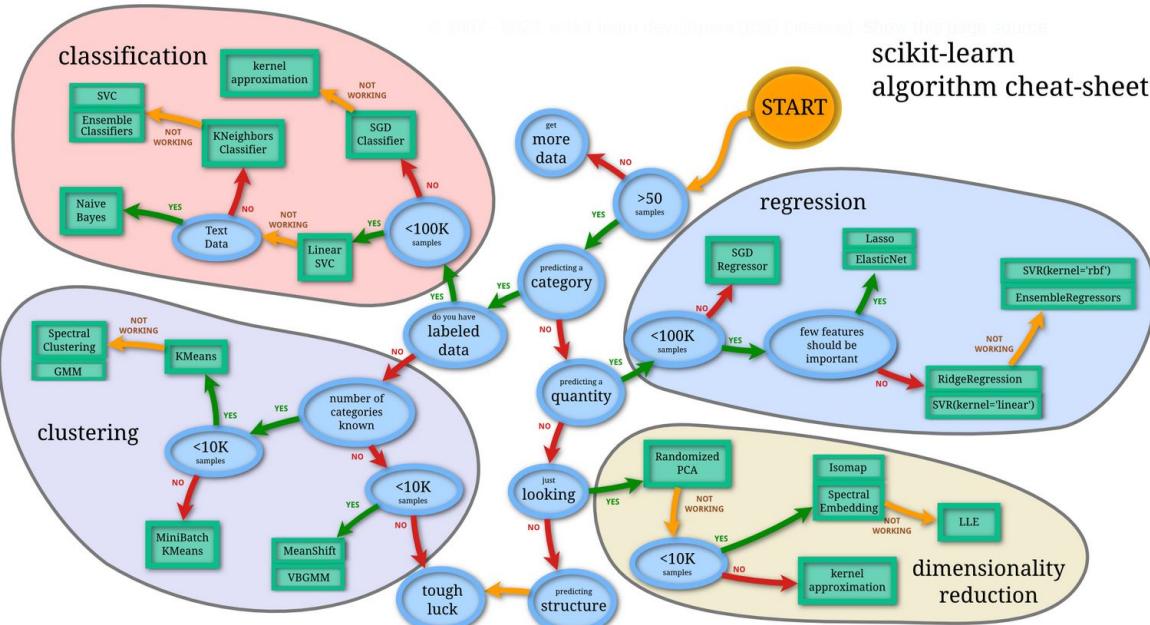
 La *validation croisée* permet d'éviter d'avoir à créer un jeu de validation. Attention, elle ne permet pas d'éviter la création d'un jeu de test !



Source : Wikipédia

1. Partitionner le jeu d'entraînement en  $k$  parties de tailles égales.
2. Pour chaque partie : mettre de côté cette partie et entraîner un modèle sur les données restantes. Puis évaluer ce modèle sur la partie mise de côté.
3. Calculer la moyenne des évaluations pour obtenir une évaluation globale.

# Comment sélectionner un modèle quand on n'a pas le temps ?



Source : [https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)

## Où trouver des ressources ?

- Kaggle
- Hugging face
- La documentation de scikit-learn