



Introduction au Machine Learning

Séance 0 — Le cours, et l'atelier dans lequel il se pratique

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

<https://ebtoure.sn/cours/introduction-machine-learning/chapitres/00-le-ml-et-son-atelier.html>

Version du 11 octobre 2026



Version en ligne

Introduction au ML — Séance 0

Le cours, et l'atelier dans lequel il se pratique

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

Objectifs de la séance

Contenu

Cette séance répond à la question fondatrice — *qu'est-ce que le Machine Learning ?* — puis installe l'**atelier** dans lequel il se pratique. À la fin de la séance, chacun sait situer le ML, nommer ses trois familles, et exécute sa première cellule de code.

- **Définir** le ML : la définition historique (Samuel) et la définition moderne (Mitchell).
- **Situer** le ML entre l'Intelligence Artificielle et le Deep Learning.
- Distinguer les **trois types d'apprentissage** : supervisé, non supervisé, par renforcement.
- Acquérir le **vocabulaire** de base et l'idée de **généralisation** — et connaître les limites du ML.
- Installer la **pile logicielle** : Anaconda → environnement `ml` → Jupyter.

Pourquoi le Machine Learning

Une question pour commencer

Le mystère de la fraude

Chaque jour, des millions de transferts d'argent mobile circulent au Sénégal. Parmi eux, quelques centaines sont frauduleux — et l'opérateur les repère *en temps réel*. Pourtant, **personne n'a écrit la règle** « si le montant dépasse X et que l'heure est Y, alors fraude » : les fraudeurs changent de méthode chaque semaine, et toute règle écrite à la main serait obsolète avant d'être déployée.

Comment une machine peut-elle détecter ce qu'aucun programmeur ne sait décrire ? C'est exactement la question à laquelle répond le Machine Learning — et cette séance.

Le ML est déjà partout autour de nous

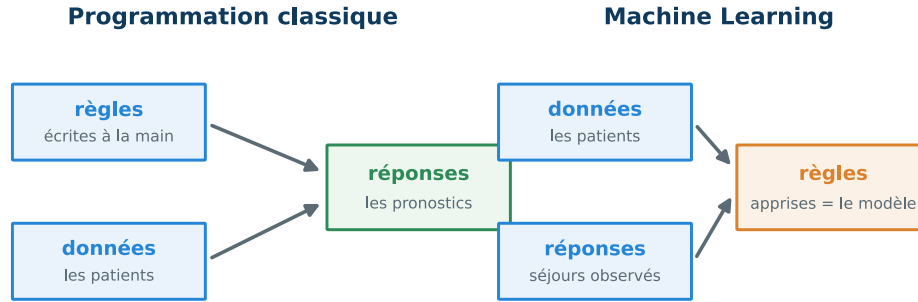
Des décisions apprises des données

Derrière des services quotidiens se cache la même mécanique : une machine qui a **appris une règle de décision** à partir de milliers d'exemples passés, au lieu qu'un programmeur l'écrive.

- **Santé** : à l'hôpital, anticiper les séjours longs pour planifier les lits — le fil rouge de ce cours.
- **Finance mobile** : détecter une transaction frauduleuse parmi des millions de transferts.
- **Agriculture** : prédire un rendement de mil ou d'arachide à partir de la pluviométrie et du sol.
- **Quotidien** : filtrer le spam, recommander une vidéo, déverrouiller un téléphone par le visage.

Quatre domaines, un seul mécanisme — et c'est ce mécanisme que le cours apprend à manier.

Programmer autrement



Le ML inverse le sens de la flèche : on fournit des exemples résolus, la machine en déduit la règle de prédiction.

En programmation classique, les règles sont écrites à la main et appliquées aux données. En Machine Learning, on fournit des couples (données, réponses) et la machine **produit la règle** — qu'on appelle alors un **modèle**.

Qu'ont en commun tous ces exemples ?

- Dans chaque cas, la règle de décision est **trop complexe ou trop mouvante** pour être écrite à la main.
- Dans chaque cas, on dispose d'**exemples résolus** : des transferts étiquetés fraude/légitime, des patients dont le séjour est connu, des saisons dont le rendement est mesuré.
- Dans chaque cas, la machine **extraie la règle des exemples** — et l'applique aux cas futurs.

La transition

Trois ingrédients reviennent toujours : une **tâche**, de l'**expérience** (les exemples), et une mesure de **performance**. Il ne reste qu'à en faire une définition — c'est exactement ce que fit Tom Mitchell.

Définir le Machine Learning

La définition historique — Samuel

Arthur Samuel — 1959

Le Machine Learning est « le champ d'étude qui donne aux ordinateurs la capacité d'**apprendre sans être explicitement programmés** ». (« *Field of study that gives computers the ability to learn without being explicitly programmed.* »)

L'anecdote fondatrice

Samuel écrit un programme de **jeu de dames** qui joue des dizaines de milliers de parties *contre lui-même*. À force de parties, le programme apprend quelles positions mènent à la victoire — et finit par **battre Samuel lui-même**, qui n'était pourtant pas un grand joueur. La machine a dépassé son programmeur : elle n'a pas été programmée pour bien jouer, elle a *appris* à bien jouer.

La définition moderne — Mitchell, formellement

L'idée en une phrase

La définition de Samuel dit *quoi* ; celle de Tom Mitchell (1997) dit *comment le vérifier* — elle rend l'apprentissage **mesurable**.

Définition de Mitchell

Un programme **apprend** de l'expérience E , relativement à une tâche T et une mesure de performance P , si sa performance sur T , mesurée par P , **s'améliore avec** E .

Décryptage : T est ce qu'on veut faire ; E est ce qu'on montre à la machine (les exemples) ; P est le chiffre qui dit si ça marche. Apprendre = voir P monter quand E grandit. Sans P , impossible de distinguer un programme qui apprend d'un programme qui récite.

Exemple — Mitchell instancié deux fois

Sur la fraude mobile money

T : classer chaque transfert en fraude ou légitime. E : l'historique de millions de transferts, étiquetés après enquête. P : la proportion de fraudes correctement détectées sur les transferts du mois suivant.

Sur le fil rouge du cours

T : prédire si le séjour hospitalier d'un patient sera long. E : un registre de 10 000 patients passés, pour lesquels la réponse est connue. P : la proportion de prédictions correctes sur des patients *jamais vus*.

Le triplet (T, E, P) est la **grille de lecture** de tout problème de ML : chaque séance du cours commencera, implicitement, par le remplir.

Exercice — instancier Mitchell

Énoncé

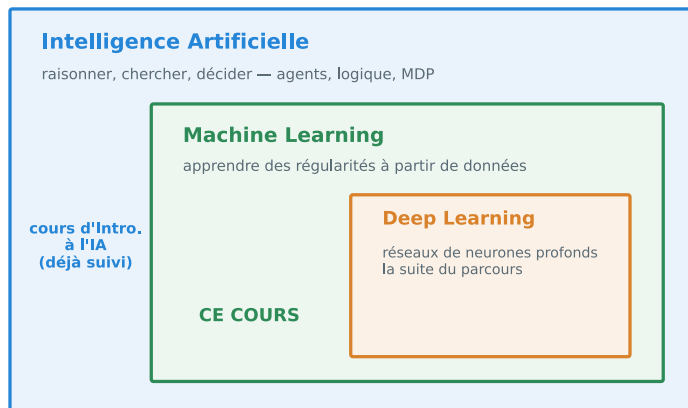
Une coopérative veut prédire le **rendement d'arachide** (en tonnes par hectare) de chaque parcelle, à partir de la pluviométrie, du type de sol et de la date de semis. Elle dispose des relevés complets des 12 dernières campagnes. Identifier T , E et P .

Correction

T : prédire le rendement d'une parcelle (un *nombre*). E : les relevés des 12 campagnes — parcelles *avec* leur rendement observé. P : l'écart moyen entre rendement prédit et rendement réel sur une campagne *non utilisée* pour apprendre. Noter que P n'est pas « un pourcentage de bonnes réponses » : la sortie étant un nombre, on mesure une *erreur* — la distinction arrive dans quelques diapositives.

Situer le ML entre IA et Deep Learning

Trois cercles emboîtés



Le ML est un **sous-domaine** de l'Intelligence Artificielle ; le Deep Learning est un **sous-domaine** du ML. Ce cours occupe le cercle du milieu.

En amont et en aval de ce cours

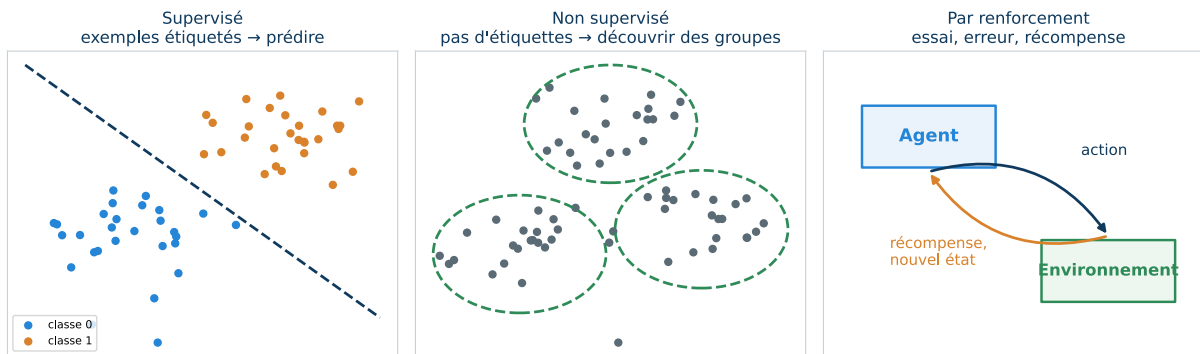
- **En amont — l'IA symbolique** (cours d'Introduction à l'IA) : des agents qui *raisonnent sur des règles données* — recherche de chemins, logique, décision sous incertitude. Les règles y sont **écrites**, pas apprises.
- **Ici — le Machine Learning** : la machine *apprend les régularités* directement des données. C'est le changement de paradigme de la diapositive « Programmer autrement ».
- **En aval — le Deep Learning** : du ML dont les modèles sont des **réseaux de neurones profonds**, rois des images, du son et du texte. Tout ce qui s'apprend ici (données, entraînement, généralisation, évaluation) y reste valable — ce cours pose les fondations.

Le pont avec le cours d'IA

Le raisonnement bayésien et les MDP rencontrés en IA reviendront : le ML en est le prolongement naturel, quand les probabilités cessent d'être données et doivent être *estimées à partir des données*.

Les trois types d'apprentissage

Trois manières d'apprendre



Supervisé : des exemples *étiquetés*, une frontière à apprendre. **Non supervisé** : des données *sans* étiquettes, des structures à découvrir. **Par renforcement** : un agent qui apprend de ses *récompenses*, par essai-erreur.

L'apprentissage supervisé

Définition

En apprentissage **supervisé**, chaque exemple est accompagné de la **bonne réponse** (l'étiquette). La machine apprend l'application entrée → sortie, pour répondre ensuite sur des entrées nouvelles.

- Si la sortie est un **nombre** (rendement, prix, durée), on parle de **régression**.
- Si la sortie est une **catégorie** (fraude/légitime, séjour long/court), on parle de **classification**.

C'est de très loin le type le plus utilisé en pratique — l'essentiel de la valeur créée par le ML aujourd'hui repose sur ce simple schéma entrée → sortie. Les Séances 3 à 8 lui sont consacrées.

Exercice — régression ou classification ?

Énoncé

Pour chaque tâche, dire s'il s'agit de régression ou de classification : **(a)** prédire la durée d'un séjour hospitalier en jours ; **(b)** décider si un mail est un spam ; **(c)** prédire le prix d'un mouton de Tabaski ; **(d)** reconnaître la lettre manuscrite sur une copie (26 possibilités).

Correction

(a) régression — la sortie est un nombre ; **(b)** classification — deux catégories ; **(c)** régression ; **(d)** classification — la sortie est une catégorie parmi 26, pas une quantité qu'on mesure. Le critère est toujours la *nature de la sortie*, jamais celle des entrées.

L'apprentissage non supervisé

Définition

En apprentissage **non supervisé**, les données arrivent **sans étiquettes**. La machine ne prédit pas une réponse : elle *découvre une structure* cachée dans les données.

- **Clustering** : regrouper les exemples qui se ressemblent — segmenter les clients du mobile money en profils d'usage, sans qu'aucun profil n'ait été défini à l'avance.
- **Réduction de dimension** : compresser des centaines de variables en quelques axes essentiels, pour visualiser ou accélérer.

Plus exploratoire, souvent préparatoire au supervisé — la Séance 9 y revient.

L'apprentissage par renforcement

Définition

En apprentissage **par renforcement**, un **agent** agit dans un environnement et reçoit des **récompenses**. Personne ne lui montre la bonne action : il la découvre par *essai-erreur*, en maximisant sa récompense cumulée.

- C'est le cadre des **MDP** rencontrés en Introduction à l'IA — avec une différence décisive : les probabilités de transition et les récompenses ne sont plus *données*, l'agent doit les *apprendre en agissant*.
- Le programme de dames de Samuel en est l'ancêtre : ses parties contre lui-même *sont* l'essai-erreur.

Pour être complet

Il existe des variantes intermédiaires (semi-supervisé, auto-supervisé) — hors du périmètre de ce cours, qui se concentre sur le supervisé et ouvre une porte vers le non supervisé.

Exercice — quel type d'apprentissage ?

Énoncé

Pour chaque situation, identifier le type d'apprentissage : **(a)** regrouper 50 000 abonnés télécom en profils de consommation, sans catégories prédéfinies ; **(b)** prédire si un patient sera réadmis sous 30 jours, à partir de dossiers où la réadmission est connue ; **(c)** faire apprendre à un thermostat à climatiser une salle en le récompensant quand la température est bonne et la consommation basse.

Correction

(a) non supervisé (clustering) — aucune étiquette ; **(b)** supervisé (classification) — l'étiquette réadmis/non réadmis est présente dans les données ; **(c)** par renforcement — pas d'étiquette, mais une récompense et de l'essai-erreur. La question à se poser est toujours : *qu'est-ce que la machine reçoit pour apprendre ?*

Le vocabulaire et le but du jeu

Les cinq mots du métier

Terme	Ce que c'est	Dans le fil rouge
Caractéristiques (<i>features</i> , X)	les variables qui décrivent un exemple	âge, glycémie, hémoglobine, fièvre, saison
Étiquette (<i>label</i> , y)	la réponse à prédire	séjour long ou court
Exemple	un cas complet — caractéristiques et étiquette	un patient du registre
Jeu d'entraînement	les exemples montrés à la machine	les patients passés
Modèle	la règle apprise qui prédit y à partir de X	le produit final du cours

Le but du jeu : généraliser

Définition

Généraliser, c'est prédire juste sur des exemples **jamais vus** pendant l'apprentissage. C'est le *seul* objectif du ML : un modèle qui ne sait répondre que sur ses données d'entraînement n'a rien appris — il a mémorisé.

Analogie

Un étudiant qui récite par cœur les exercices du TD peut échouer à l'examen : réussir le TD ne prouve rien, réussir des exercices *nouveaux* prouve la compréhension. Pour le modèle, c'est pareil — on gardera toujours des données de côté pour lui faire passer un **examen**.

Cette idée — séparer des données pour évaluer honnêtement — structure tout le cours ; elle est mise en pratique dès la Séance 3.

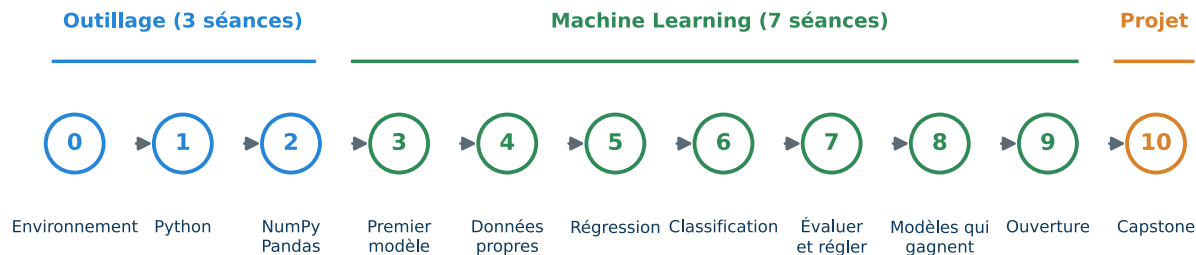
Ce que le ML ne fait pas

- **Pas de miracle sur de mauvaises données** : un modèle nourri de données fausses ou biaisées apprend des règles fausses ou biaisées — *garbage in, garbage out*.
- **Pas de généralisation au-delà de son expérience** : un modèle entraîné sur les patients d'une seule région peut se tromper lourdement sur le reste du pays — l'échantillon doit représenter la cible.
- **Pas de causalité** : le ML capture des *régularités*, pas des causes — corrélation n'est pas causalité.
- **Pas de magie** : un modèle ne remplace ni la compréhension du problème, ni la connaissance du terrain — il les *outille*.

Erreur fréquente

Croire un chiffre de performance sans demander *sur quelles données* il a été mesuré. Un score calculé sur les données d'entraînement est une illusion — toute la suite du cours apprend à mesurer honnêtement.

La route des dix séances



Trois séances d'outillage (environnement, Python, données), sept séances de Machine Learning — presque toutes consacrées au **supervisé**, avec l'ouverture non supervisée en Séance 9 — et un projet final sur données réelles. Tout commence ici, par l'atelier.

Les données du cours

Le fil rouge

DataSANTÉ-221 : un registre hospitalier de 10 000 patients de cinq régions du Sénégal — âge, glycémie, hémoglobine, fièvre, saison — avec la durée de séjour observée. Toutes les notions du cours s’y incarnent d’abord.

- Le fil rouge donne un **terrain familier** : à chaque nouvelle notion, les données sont déjà connues.
- S’y ajoutent des **jeux réels** : Pima (diabète), California Housing (immobilier), Heart Disease — et, pour le capstone, des données de compétitions **Zindi**, ancrées dans le contexte africain.

La pile logicielle

Pourquoi un environnement local

Le choix du cours

Le cours travaille **en local** : tout s'installe sur la machine de chacun. C'est l'outillage du métier — celui qui fonctionne sans connexion, sur ses propres fichiers, et que l'on contrôle de bout en bout.

- **Autonomie** : pas de dépendance à une connexion stable ni à un service distant.
- **Réalisme** : en entreprise comme en recherche, on travaille dans un environnement maîtrisé.
- **Pérennité** : l'installation d'aujourd'hui servira aux cours et projets des années suivantes.

Erreur fréquente

Compter sur un service en ligne « le jour J » : une coupure de connexion pendant une évaluation ou une démonstration, et plus rien ne tourne. L'atelier local, lui, ne dépend que de la machine.

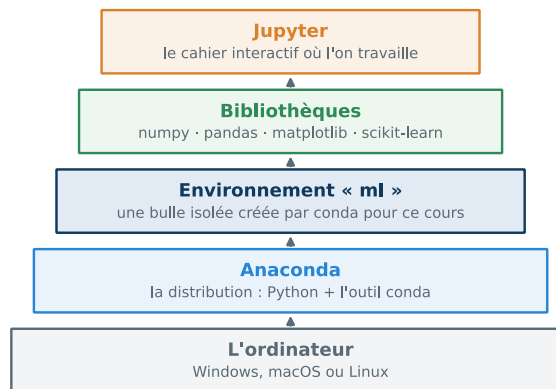
La pile : qui s'appuie sur qui

Quatre étages

Anaconda installe Python et l'outil **conda** ;
conda crée l'**environnement ml** ;
l'environnement contient les **bibliothèques** ;
Jupyter est la salle de travail où on les utilise.

Toute la séance consiste à monter ces étages, un par un, dans l'ordre.

Chaque étage s'appuie sur celui du dessous



Le vocabulaire de l'atelier

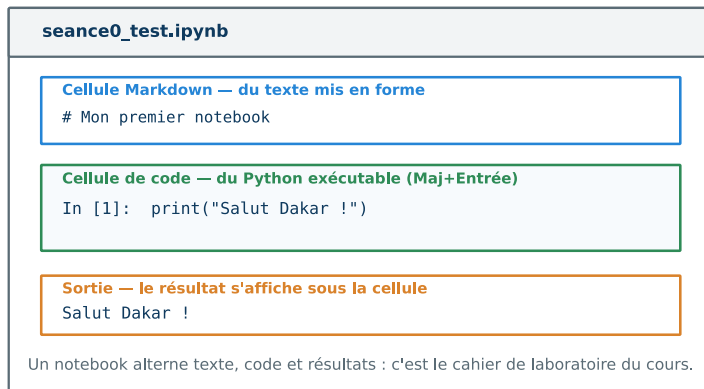
Terme	Ce que c'est
Terminal	La fenêtre où l'on tape des commandes texte (Anaconda Prompt sous Windows ; Terminal sous macOS et Linux).
Distribution	Un paquet « tout-en-un » : Anaconda fournit Python, conda et des centaines de bibliothèques.
Environnement	Une bulle isolée contenant une version de Python et ses bibliothèques. Le nôtre s'appelle ml .
Bibliothèque	Du code prêt à l'emploi qu'on importe : numpy , pandas , matplotlib , scikit-learn .
Notebook	Un document interactif mêlant texte, code et résultats — le format de tous les TP.

Les quatre bibliothèques du cours

- **NumPy** — le calcul sur tableaux de nombres : rapide, vectorisé, à la base de tout le reste.
- **Pandas** — les tableaux de données étiquetés (lignes, colonnes nommées) : lire un fichier, filtrer, agréger.
- **Matplotlib** — les graphiques : voir ses données avant de les modéliser.
- **scikit-learn** — les modèles de Machine Learning, tous derrière la même interface `fit / predict`.

Les Séances 1 et 2 apprennent à manier les trois premières ; `scikit-learn` entre en scène à la Séance 3 — et n'en sort plus.

Jupyter : le cahier de laboratoire



Un notebook est une suite de **cellules**. Les cellules *Markdown* portent le texte ; les cellules de *code* s'exécutent avec **Maj+Entrée** et affichent leur résultat juste en dessous. On expérimente, on corrige, on réexécute — le cycle naturel du ML.

Exercice — qui fait quoi ?

Énoncé

Associer chaque besoin à l'outil de la pile : **(a)** créer une bulle isolée pour le cours ; **(b)** lire un fichier de patients et filtrer les plus de 60 ans ; **(c)** entraîner un arbre de décision ; **(d)** écrire et exécuter le TP cellule par cellule.

Correction

(a) `conda` (commande `conda env create`) ; **(b)** `pandas` ; **(c)** `scikit-learn` ; **(d)** Jupyter.
Chaque outil a un rôle unique — c'est leur empilement qui fait l'atelier.

Installation pas à pas

La feuille de route en six étapes

Le chemin complet

1. Télécharger Anaconda. ;**2.** L'installer. ;**3.** Vérifier `conda` dans un terminal. ;**4.** Créer l'environnement `m1`. ;**5.** L'activer. ;**6.** Lancer Jupyter et exécuter la cellule de vérification.

Chaque étape se vérifie avant de passer à la suivante : une installation qui échoue à l'étape 4 se répare à l'étape 4 — jamais en recommençant tout.

Étape 1 — télécharger Anaconda

- Se rendre sur **anaconda.com/download** et choisir l'installateur de son système.
- **Windows** : installateur graphique `.exe` (64 bits). **macOS** : installateur `.pkg` — attention au choix entre puce *Apple Silicon* (M1 à M4) et *Intel*. **Linux** : script `.sh`.
- Le fichier pèse environ 1 Go : prévoir le téléchargement *avant* la séance si la connexion est lente — il sera aussi distribué sur clé USB en début de séance.

Erreur fréquente

Sur macOS, télécharger la version Intel sur une machine Apple Silicon (ou l'inverse) : l'installation passe, mais les performances s'effondrent. Vérifier sa puce dans « À propos de ce Mac ».

Étape 2 — installer

- **Windows** : lancer l'.exe, accepter les choix par défaut — installation « Just Me », *ne pas* cocher l'ajout manuel au **PATH** (l'*Anaconda Prompt* s'en charge proprement).
- **macOS** : ouvrir le .pkg et suivre l'assistant.
- **Linux** : `bash Anaconda3-*.sh`, accepter la licence, accepter l'initialisation de **conda** en fin d'installation, puis rouvrir le terminal.

Erreur fréquente

Installer dans un dossier dont le chemin contient des **espaces ou des accents** (« `C:\Users\Mamadou Bâ\...` ») : certains outils s'y perdent. En cas de doute, choisir un chemin court et sans accents, par exemple `C:\anaconda3`.

Étape 3 — ouvrir le terminal et vérifier

Windows : menu Démarrer → *Anaconda Prompt*. **macOS / Linux** : application *Terminal*.

```
conda --version
# conda 24.x.y      <- la version exacte importe peu : elle doit s'afficher
```

Si une version s'affiche, l'étape Anaconda est posé. Sinon, voir le dépannage en fin de séance — la cause est presque toujours l'utilisation du mauvais terminal sous Windows.

Réflexe à prendre

Sous Windows, *toutes* les commandes du cours se tapent dans l'**Anaconda Prompt** — jamais dans l'invite de commandes classique, qui ne connaît pas [conda](#).

Le fichier de l'environnement

L'environnement du cours est décrit dans un petit fichier texte, `environnement - ml.yml`, distribué avec la séance :

```
name: ml
channels:
  - defaults
dependencies:
  - python=3.11
  - numpy
  - pandas
  - matplotlib
  - scikit-learn
  - jupyter
  - flake8
```

Une ligne par bibliothèque : le fichier est la liste de courses de l'atelier. Le même fichier recrée le même environnement sur n'importe quelle machine — c'est sa force.

Étape 4 — créer l'environnement ml

Dans le terminal, se placer dans le dossier contenant le fichier (commande `cd`), puis :

```
conda env create -f environnement-ml.yml  
# Telechargement et installation des paquets : quelques minutes...
```

En cas d'absence du fichier, la commande équivalente tient en une ligne :

```
conda create -n ml python=3.11 numpy pandas matplotlib scikit-learn jupyter flake8
```

La création ne se fait qu'**une seule fois** : l'environnement reste installé pour tout le cours.

Étape 5 — activer l'environnement

```
conda activate ml  
# l'invite change : (ml) C:\Users\fatou>
```

Ce que fait l'activation

Activer, c'est **entrer dans la bulle** : à partir de là, `python` et `jupyter` désignent ceux de l'environnement `ml`, avec ses bibliothèques. Le préfixe `(ml)` dans l'invite confirme qu'on est dedans.

Réflexe à prendre

À chaque nouvelle session de travail : ouvrir le terminal, taper `conda activate ml`, puis lancer Jupyter. L'activation ne survit pas à la fermeture du terminal.

Étape 6 — lancer Jupyter

```
(ml) jupyter notebook  
# Le navigateur s'ouvre sur http://localhost:8888 ...
```

- Le navigateur affiche les fichiers du dossier courant ; **New** → **Python 3** crée un notebook.
- Le terminal devient le **moteur** de Jupyter : il affiche des messages tant que Jupyter tourne.
- Pour arrêter proprement : fermer les notebooks, puis **Ctrl+C** dans le terminal.

Erreur fréquente

Fermer la fenêtre du terminal pendant le travail : Jupyter s'éteint avec elle et les cellules cessent de répondre. Le terminal reste ouvert, en arrière-plan, toute la session.

Le premier notebook

Créer un notebook, le renommer `seance0_test`, puis taper dans la première cellule :

```
print("Salut Dakar !")
```

Maj+Entrée exécute la cellule : le texte s'affiche dessous, et une cellule vide apparaît. C'est le geste fondamental — il sera répété des centaines de fois dans ce cours.

Ce qui vient de se passer

La cellule a traversé toute la pile : Jupyter a transmis le code au Python de l'environnement `m1`, qui l'a exécuté et renvoyé le résultat. Quatre étages, un aller-retour — en quelques millisecondes.

La cellule de vérification

Dans une nouvelle cellule, vérifier que les quatre bibliothèques répondent :

```
import numpy, pandas, matplotlib, sklearn
print("numpy      :", numpy.__version__)
print("pandas     :", pandas.__version__)
print("matplotlib  :", matplotlib.__version__)
print("scikit-learn :", sklearn.__version__)
```

Quatre numéros de version s'affichent : l'atelier est complet. Cette cellule est le **certificat d'installation** — elle ouvre tous les notebooks du cours, et son échec désigne immédiatement l'étage en panne.

Exercice — lire une commande

Énoncé

Un étudiant tape les trois commandes suivantes, dans cet ordre : `conda activate ml`, puis `jupyter notebook`, puis il ferme le terminal. **(a)** Que fait chacune des deux commandes ? **(b)** Que se passe-t-il à la fermeture du terminal ?

Correction

(a) La première *entre dans la bulle ml* (le préfixe `(ml)` apparaît) ; la seconde démarre Jupyter, dont le terminal devient le moteur. **(b)** Le moteur s'éteint avec le terminal : le notebook ouvert dans le navigateur ne répond plus. Il fallait laisser le terminal ouvert.

Et l'éditeur du métier : VS Code

Pour aller plus loin

Visual Studio Code (code.visualstudio.com) est l'éditeur le plus répandu du métier. Avec ses extensions *Python* et *Jupyter*, il ouvre les notebooks du cours et sait utiliser l'environnement `ml` (sélecteur de noyau, en haut à droite).

- Son installation est **recommandée mais pas exigée** : Jupyter dans le navigateur suffit pour tout le cours.
- Son vrai intérêt viendra avec les projets : fichiers `.py`, recherche dans le code, `flake8` intégré.

Organiser ses fichiers

Une arborescence pour tout le cours

Créer un dossier `intro-ml` (chemin **sans espaces ni accents**), avec un sous-dossier par séance : `seance00`, `seance01`, ... Chaque sous-dossier reçoit le notebook de TP et les données de la séance. Lancer Jupyter *depuis* `intro-ml`.

Une arborescence stable évite la moitié des pannes : fichiers de données « introuvables », notebooks égarés, doublons. L'autre moitié se traite à la section suivante.

Dépannage

Symptôme et remède

Symptôme	Cause probable et remède
<code>conda~::~commande introuvable</code>	Mauvais terminal (Windows) : ouvrir l' <i>Anaconda Prompt</i> . Sous Linux : rouvrir le terminal après l'installation.
<code>ModuleNotFoundError~::~sklearn</code>	Jupyter lancé <i>hors</i> de la bulle : fermer, <code>conda activate ml</code> , relancer.
L'invite n'affiche pas <code>(ml)</code>	L'activation a été oubliée ou faite dans un autre terminal.
Les cellules ne répondent plus	Le terminal-moteur a été fermé : relancer Jupyter (l'environnement, lui, est intact).
<code>EnvironmentLocationNotFound</code>	L'environnement n'a jamais été créé : refaire l'étape 4.

Exercice — diagnostiquer une panne

Énoncé

Une étudiante lance `jupyter notebook`, ouvre un notebook et exécute `import sklearn: ModuleNotFoundError`. Pourtant, l'étape 4 s'était terminée sans erreur la veille. **(a)** Quelle est la cause la plus probable ? **(b)** Donner la séquence exacte de commandes qui répare.

Correction

(a) Jupyter a été lancé *hors* de l'environnement `ml` : il utilise le Python de base, où `scikit-learn` n'est pas installé. L'environnement, créé la veille, existe toujours — il n'était simplement pas *activé*. **(b)** Fermer Jupyter (`Ctrl+C`), puis : `conda activate ml` et `jupyter notebook`.

La checklist de sortie

Avant de quitter la séance

1. `conda --version` répond. ; **2.** `conda activate ml` affiche le préfixe (ml). ; **3.** Jupyter s'ouvre dans le navigateur. ; **4.** La cellule de vérification affiche les quatre versions. ; **5.** Le dossier `intro-ml` existe, sans espaces ni accents dans son chemin.

Cinq cases cochées : la machine est prête pour les dix séances. Une case manquante : la table « symptôme et remède » donne la réparation — et la séance ne se termine pas avant que tout le monde coche les cinq.

Synthèse

À retenir — Séance 0

L'essentiel

- Le ML **inverse la programmation** : on fournit des exemples résolus, la machine produit la règle — le modèle.
- **Mitchell** rend l'apprentissage mesurable : une tâche T , une expérience E , une performance P qui s'améliore.
- **IA** $\supset$ **ML** $\supset$ **DL** — et trois types d'apprentissage : supervisé (étiquettes), non supervisé (structures), par renforcement (récompenses).
- Le but unique : **généraliser** — prédire juste sur du jamais-vu, pas réciter l'entraînement.
- L'atelier : Anaconda $\rightarrow$ `conda` $\rightarrow$ environnement `ml` $\rightarrow$ bibliothèques $\rightarrow$ Jupyter ; le rituel de chaque session est `conda activate ml` puis `jupyter notebook`.

Pièges fréquents

- Confondre régression et classification : le critère est la *nature de la sortie* (nombre ou catégorie).
- Croire qu'un modèle « comprend » : il capture des régularités — corrélation n'est pas causalité.
- Évaluer un modèle sur ses données d'entraînement : mémoriser n'est pas généraliser.
- Travailler hors de l'environnement `ml` : tout semble marcher — jusqu'au premier `import sklearn`.
- Fermer le terminal qui fait tourner Jupyter : les cellules cessent de répondre.

Prochaine séance

Séance 1 — Python de zéro

L'atelier est monté ; il reste à apprendre la langue qu'on y parle. La Séance 1 enseigne Python en partant de zéro — variables, listes, conditions, boucles, fonctions — entièrement dans Jupyter, sur l'environnement installé aujourd'hui.

Références

- T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 3^e éd., O'Reilly, 2022.
- A. C. Müller, S. Guido, *Introduction to Machine Learning with Python*, O'Reilly, 2016.
- G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, 2^e éd., Springer, 2021.
- Documentation officielle : conda (docs.conda.io) ; Jupyter (docs.jupyter.org).