

Introduction au Machine Learning



Séance 11 — Le capstone : un projet de bout en bout

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/11-capstone.html>

Version du 11 octobre 2026



Version en ligne

Introduction au ML — Séance 11

Le capstone : un projet de bout en bout

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

Une séance différente

Cette dernière séance n'introduit *aucun* modèle nouveau. Elle **rassemble** tout le cours en un **projet complet**, mené de bout en bout comme dans une compétition de science des données (type *Zindi*, la plateforme africaine). L'objectif n'est plus d'apprendre une technique, mais d'orchestrer toutes celles déjà vues, dans le bon ordre, avec méthode.

- Le **cycle de vie** d'un projet ML : cadrer, explorer, préparer, comparer, régler, évaluer, interpréter, présenter.
- **Mobiliser chaque séance** à sa place : préparation et fuite (S4), baseline (S3), modèles (S5–S10), validation croisée et réglage (S7), métriques et seuil (S6).
- Un **fil rouge unique** : prédire le paludisme sur DataSANTÉ-221, du fichier brut à la recommandation finale.
- La **leçon centrale** : sur des données déséquilibrées, un modèle peut afficher 88 % d'accuracy en *n'attrapant aucun* malade. Savoir le voir, et le corriger.
- La **soutenance** : comment présenter un projet ML — ce qui sera évalué.

Le cycle de vie d'un projet

D'une collection de techniques à une démarche

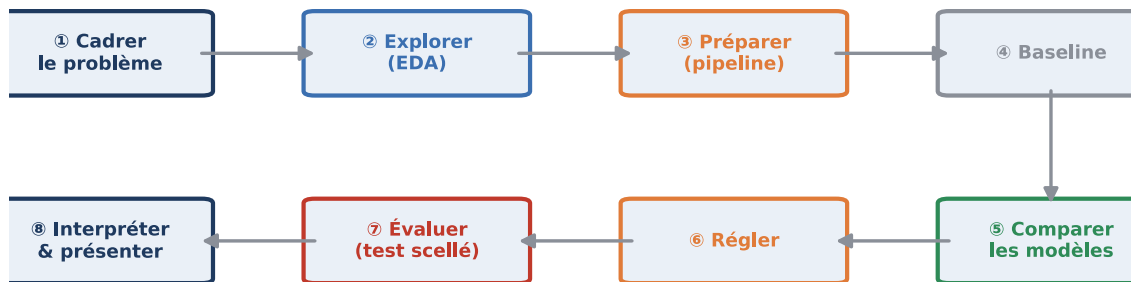
Le cours, vu d'en haut

Dix séances ont construit une boîte à outils : arbres, régressions, ensembles, réseaux, métriques, validation. Mais un praticien n'est pas celui qui connaît le plus d'outils — c'est celui qui sait, face à un problème neuf, *lesquels* utiliser, *dans quel ordre*, et comment *juger* le résultat. C'est cette démarche que le capstone fait pratiquer.

Un projet réel ne commence jamais par « quel modèle ? ». Il commence par une question métier et un fichier de données, et il finit par une décision et une présentation. Entre les deux, une suite d'étapes *toujours les mêmes*, quelle que soit la discipline. Les voici — c'est la carte de toute la séance.

Le cycle de vie, en image

Le cycle de vie d'un projet de Machine Learning



Un fil rouge unique relie toutes les séances : c'est ce que le capstone fait pratiquer.

Les huit étapes d'un projet ML, du cadrage à la présentation. Chaque étape mobilise une ou plusieurs séances du cours. La séance suit cet ordre, étape par étape, sur un projet réel : prédire le paludisme.

Le projet fil rouge

Le cadre, comme sur Zindi

Question : à partir de données de patients (âge, glycémie, hémoglobine, fièvre, saison, durée de séjour), prédire le **paludisme**. **Données** : DataSANTÉ-221, 10 000 patients. **Livrable** : un modèle évalué honnêtement, interprété, et une recommandation.

C'est exactement le format d'une compétition Zindi : un jeu de données, une cible, une métrique, et un classement. Mais une compétition n'évalue qu'un score ; un *projet* exige plus — comprendre les données, éviter les pièges méthodologiques, interpréter, et savoir présenter. Le capstone vise ce projet complet, pas seulement le score. Commençons par la première étape, trop souvent négligée : **cadrer**.

Cadrer et explorer

Étape 1 — cadrer le problème

Quatre questions avant tout code

Avant la moindre ligne de code : **(1)** quelle *tâche* ? (ici : classification binaire) ; **(2)** quelle *métrique* ? ; **(3)** quelle *référence* à battre ? ; **(4)** quelles *contraintes* métier ?

Ces choix orientent tout le reste. La cible (paludisme : oui/non) impose une *classification*. La prévalence — seulement 11,8 % de cas positifs — signale des données **déséquilibrées**, ce qui disqualifie l'accuracy comme métrique (on le verra cruellement) et impose l'**AUC** et le **rappel** (S6). La contrainte métier en santé est décisive : *rater un malade* (faux négatif) est bien plus grave que déclencher un examen inutile (faux positif). Ce cadrage guidera jusqu'au choix du seuil final.

Étape 2 — explorer les données (EDA)

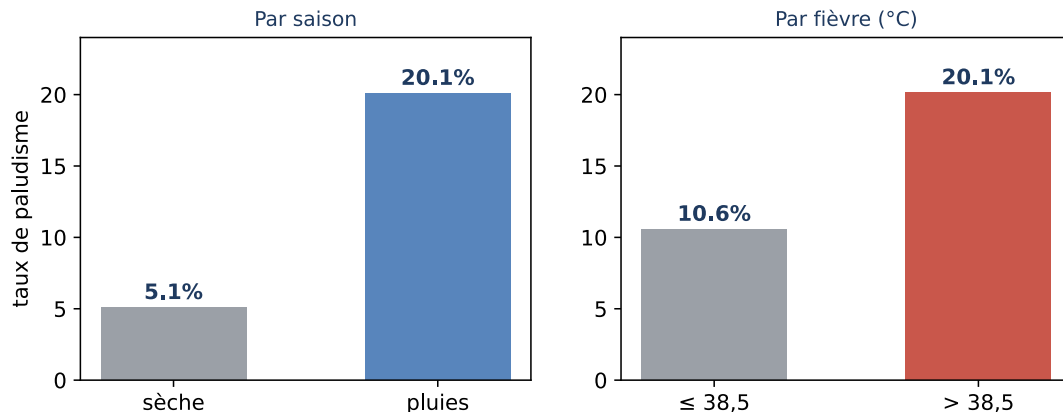
Regarder avant de modéliser

L'**analyse exploratoire** (EDA) consiste à *comprendre* les données avant de les modéliser : distributions des variables, valeurs aberrantes, et surtout liens entre les variables et la cible. C'est gratuit, rapide, et cela oriente tout le projet.

On y repère les facteurs de risque *à l'œil nu*, ce qui donne une intuition de ce qu'un modèle devrait capter — et un moyen de vérifier, plus tard, que ses conclusions sont plausibles. Sur DataSANTÉ, deux variables sautent aux yeux : la *saison* et la *fièvre*. Regardons.

L'EDA, en image

EDA : deux facteurs de risque ressortent avant tout modèle



À gauche, le paludisme est quatre fois plus fréquent en saison des pluies (20,1 %) qu'en saison sèche (5,1 %). À droite, une forte fièvre ($> 38,5^{\circ}\text{C}$) double le taux (20,1 % contre 10,6 %). Ces deux facteurs, visibles *sans* modèle, devront ressortir dans l'interprétation finale — un test de cohérence.

Préparer : le piège de la fuite

Étape 3 — préparer sans tricher

Le piège le plus coûteux du ML

La préparation des données (nettoyage, encodage, standardisation) cache le piège le plus insidieux du métier : la **fuite de données**. Si une information du jeu de test influence l'entraînement — même indirectement, via une moyenne de standardisation calculée sur *toutes* les données — le score devient *mensonger*, et le modèle s'effondre en production. C'est l'objet de la Séance 4.

La parade est une discipline stricte, et un ordre immuable : **sceller le test d'abord**, ne calculer les transformations que sur le *train*, puis les *appliquer* au test. L'outil qui garantit cette hygiène est le **Pipeline** de scikit-learn. Première étape concrète : découper.

Sceller le jeu de test, avant tout

Première décision : sceller un jeu de test, **AVANT** tout

TRAIN — 80 % (8000)
validation croisée, réglage

TEST — 20 %
SCELLÉ
(2000)

Stratifié : prévalence 11,8 % préservée des deux côtés. Le test n'est ouvert qu'à la toute fin.

La toute première opération sur les données : mettre 20 % de côté comme *test scellé*, et ne plus y toucher jusqu'à l'évaluation finale. Le découpage est **stratifié** (S7) : la prévalence 11,8 % est préservée dans le train et le test. Tout le travail — exploration, réglage — se fera sur les 80 % restants.

L'idée : le Pipeline scelle l'hygiène

L'idée en une phrase

Un **Pipeline** enchaîne préparation et modèle en un seul objet. Lors de la validation croisée, il *réapprend* la préparation sur chaque pli d'entraînement — empêchant *structurellement* toute fuite de la standardisation.

Sans pipeline, on standardise souvent une fois sur tout le train, *avant* la validation croisée : la moyenne et l'écart-type « voient » alors les plis de validation — une fuite. Le pipeline corrige cela en traitant la préparation comme une partie du modèle, réapprise à chaque pli. C'est la traduction technique de la règle de la Séance 4 : *rien* de ce qui touche le test (ou la validation) ne doit influencer l'apprentissage.

Le pipeline, en pratique

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# 1) sceller le test AVANT tout (stratifiée)
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y)

# 2) preparation + modele dans UN pipeline (anti-fuite)
modele = Pipeline([
    ("scaler", StandardScaler()),          # reappris a chaque pli
    ("clf", LogisticRegression(max_iter=1000)),
])
```

Le **Pipeline** se manipule comme un modèle ordinaire (**fit**, **predict**), mais garantit que la standardisation est apprise au bon endroit. C'est l'outil qui rend la préparation *reproductible* et *honnête*.

Comparer et régler

Étape 4 — la baseline, juge de paix

Toujours une référence d'abord

Avant tout modèle sophistiqué, on établit une **baseline** : la performance d'une stratégie triviale (ici, prédire toujours la classe majoritaire). Aucun modèle n'a de valeur tant qu'il ne la bat pas *de façon utile*.

Sur DataSANTÉ, prédire « jamais de paludisme » donne déjà 88,2 % d'accuracy — puisque 88,2 % des patients sont effectivement sains. Ce chiffre est un **piège tendu d'avance** : il montre qu'une accuracy élevée peut être totalement vide. La baseline n'est donc pas qu'une référence de score ; c'est un *avertissement* sur la métrique. Tout modèle devra apporter un vrai gain — en AUC, et en capacité à *repérer les malades*.

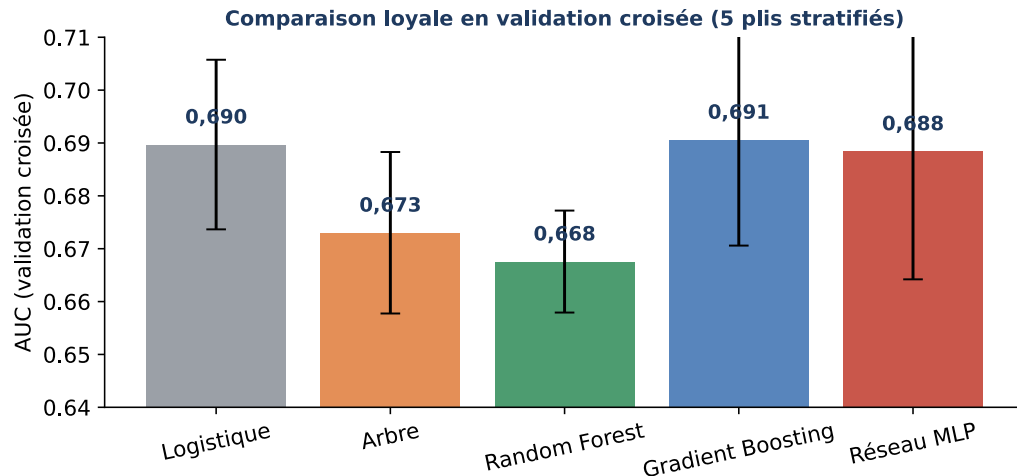
Étape 5 — comparer les modèles loyalement

Une compétition interne, en validation croisée

On confronte les grandes familles du cours — régression logistique (S6), arbre (S3), forêt et boosting (S8), réseau (S10) — *sur un pied d'égalité* : même validation croisée stratifiée (S7), même métrique (AUC). Pas de réglage fin à ce stade : on cherche le *candidat* le plus prometteur.

La validation croisée est cruciale : un seul découpage train/validation donnerait un classement instable (S7). En moyennant sur 5 plis, on obtient un score et sa variabilité — de quoi distinguer un vrai écart d'un bruit de découpage. C'est la version rigoureuse du « lequel est le meilleur ? ».

Le comparatif, en image



AUC en validation croisée (5 plis stratifiés), avec barres d'écart-type. Les modèles se tiennent dans un mouchoir : gradient boosting (0,691) et régression logistique (0,690) en tête, réseau juste derrière (0,688). Les écarts sont de l'ordre du bruit.

Conformément aux Séances 8 et 10, sur ce *tabulaire*, aucun modèle complexe n'écrase le plus simple.

Étape 6 — régler le meilleur candidat

GridSearchCV, sur le train seulement

Une fois le candidat choisi (ici le gradient boosting), on **règle ses hyperparamètres** par recherche systématique en validation croisée ([GridSearchCV](#), S7) : nombre d'arbres, profondeur, taux d'apprentissage. Le test reste *scellé*.

La recherche explore une grille de combinaisons et retient celle de meilleure AUC *en validation croisée* — jamais sur le test. Ici, le réglage fait passer l'AUC croisée de 0,691 à 0,697 : un gain modeste mais réel, obtenu sans toucher au jeu de test. Ce raffinement terminé, et *seulement* maintenant, on peut ouvrir l'enveloppe scellée.

Évaluer : la leçon du capstone

Étape 7 — ouvrir le test scellé

Le moment de vérité

Le test n'a jamais servi : ni à explorer, ni à comparer, ni à régler. Son score est donc une estimation *honnête* de la performance en production. On l'ouvre **une seule fois**, à la fin. L'AUC sur le test scellé est de **0,732** — cohérent avec la validation croisée, signe qu'il n'y a pas eu de fuite.

Jusqu'ici, tout va bien : un modèle réglé, une AUC honnête de 0,732, supérieure au hasard. On pourrait conclure au succès. Mais l'AUC ne dit pas *tout*. Regardons ce que le modèle prédit *réellement*, patient par patient — et la séance prend un tour inattendu.

Le piège se referme

La leçon du capstone : l'accuracy ment sur données déséquilibrées

Seuil 0,5 : accuracy 0,882
mais 0 vrai positif !

réel sain	1764	1
réel palu	235	0
	prédit sain	prédit palu

Seuil 0,20 : rappel 0,285
on rattrape des malades

réel sain	1574	191
réel palu	168	67
	prédit sain	prédit palu

À gauche, au seuil par défaut (0,5) : le modèle atteint 88,2 % d'accuracy... en ne prédisant **jamais** « paludisme » (0 vrai positif, 235 malades manqués). Il a appris à dire « sain » à tout le monde — exactement la baseline. À droite, en abaissant le seuil à 0,20, le modèle repère enfin 67 malades (rappel 0,285), au prix de quelques fausses alertes.

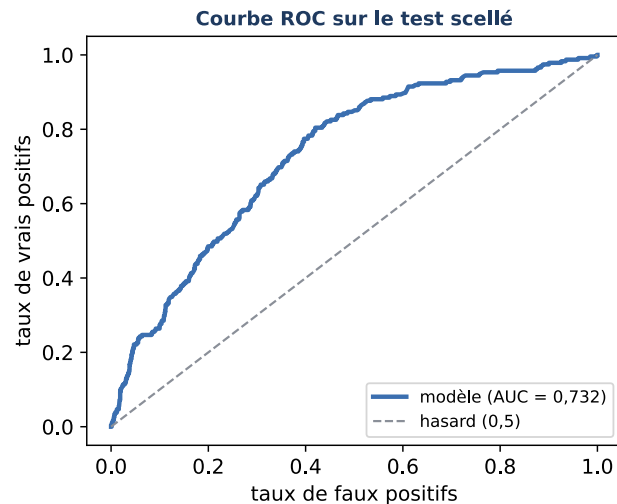
La leçon centrale du cours

L'accuracy ment sur les données déséquilibrées

Le modèle « à 88 % » est *inutile* : il rate 100 % des malades. L'accuracy, gonflée par la classe majoritaire, masquait un échec complet. C'est le piège que la baseline annonçait — et il vient de se refermer pour de vrai. Sur données déséquilibrées, juger un modèle de santé à son accuracy peut coûter des vies.

Deux remèdes, vus au fil du cours. **La métrique** : juger à l'AUC, à la précision et au *rappel* (S6), pas à l'accuracy. **Le seuil** : le seuil de décision n'a pas à valoir 0,5 ; on l'abaisse pour attraper plus de malades (plus de *rappel*), en acceptant plus de fausses alertes (moins de précision). Le bon seuil découle de la *contrainte métier* fixée à l'étape 1 : en santé, mieux vaut un examen inutile qu'un malade manqué.

Choisir le seuil selon le métier



La courbe ROC résume tous les seuils possibles : chaque point est un compromis (faux positifs, vrais positifs). L'aire sous la courbe ($AUC = 0,732$) mesure la qualité *globale*, indépendamment du seuil. Choisir un point sur la courbe — un seuil — est une décision *métier*, pas statistique : en santé, on remonte vers le haut (plus de vrais positifs), quitte à accepter des faux positifs.

Interpréter et présenter

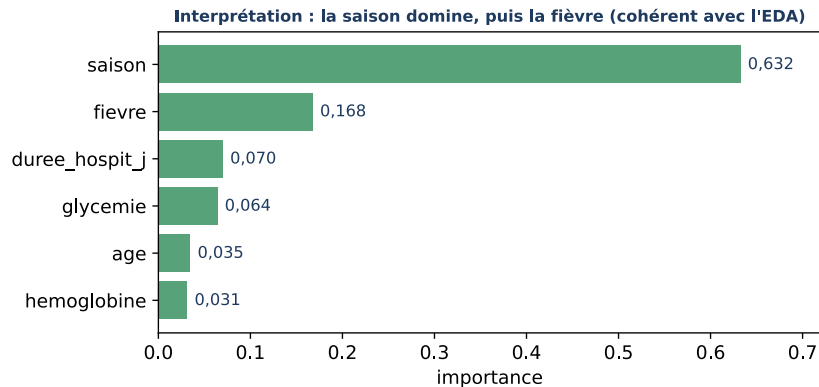
Étape 8 — interpréter le modèle

Un modèle utile est un modèle compris

Au-delà du score, il faut *expliquer* ce que le modèle a appris : quelles variables pèsent le plus (S3, S8) ? Les conclusions sont-elles *plausibles* au regard de l'EDA et du métier ? Un modèle performant mais incohérent est suspect ; un modèle cohérent inspire confiance.

L'importance des variables relie le modèle au monde réel : elle doit recouper ce que l'exploration avait montré. C'est aussi ce qui permet de *communiquer* le résultat à un décideur non technique — un médecin, une autorité de santé. Vérifions la cohérence.

L'interprétation, en image



Le modèle s'appuie surtout sur la *saison* (0,632), puis la *fièvre* (0,168) — précisément les deux facteurs repérés à l'EDA. Cette cohérence est rassurante : le modèle a capté des liens *réels*, pas des artefacts. On peut l'expliquer à un médecin en une phrase : « le risque dépend d'abord de la saison, puis de la fièvre ».

Présenter le projet : la soutenance

Ce qui sera évalué

Le capstone s'évalue sur une **soutenance** — pas sur un score brut. Une bonne présentation suit le fil du projet :

1. le **problème** et son enjeu métier (1 phrase claire) ;
2. les données et ce que l'**EDA** a révélé ;
3. la **démarche** : split scellé, pipeline anti-fuite, comparaison loyale, réglage ;
4. le **résultat honnête** : AUC sur test, et le piège de l'accuracy, et le seuil retenu ;
5. l'**interprétation** et une **recommandation** actionnable.

Le jury valorise la *rigueur méthodologique* et l'*honnêteté* (avoir vu et corrigé le piège du seuil) bien plus qu'un dixième d'AUC. C'est cela, faire du ML pour de vrai.

Les erreurs qui coulent un projet

- **Fuite de données** : standardiser avant de découper, régler sur le test. Le score devient mensonger. → pipeline + test scellé.
- **Juger à l'accuracy** sur données déséquilibrées : on récompense un modèle qui ignore les cas rares. → AUC, rappel, matrice de confusion.
- **Oublier la baseline** : sans référence, on ne sait pas si le modèle apporte quoi que ce soit.
- **Garder le seuil 0,5 par défaut** : il n'a aucune raison d'être optimal pour le métier.
- **Comparer des modèles déloyalement** (réglages inégaux, un seul découpage) : le classement n'a pas de sens.
- **Ne pas interpréter** : un modèle qu'on ne comprend pas ne peut pas être défendu ni déployé en confiance.

Synthèse : tout le cours

La carte complète du cours

Toute la progression du cours, en une carte

Fondations	S1-S2 : Python, NumPy, Pandas
Le socle du ML	S3 premier modèle · S4 données/pipeline · S5 régression+gradient
Évaluer & améliorer	S6 métriques · S7 validation croisée · S8 ensembles
Élargir	S9 non-supervisé · S10 réseaux de neurones
Synthétiser	S11 capstone : tout, de bout en bout

Les douze séances en cinq temps : les fondations (S1-S2), le socle du ML (S3-S5), évaluer et améliorer (S6-S8), élargir (S9-S10), et synthétiser (S11). Le capstone n'est pas une séance de plus : c'est le moment où toutes les autres se rejoignent sur un projet réel.

Le protocole d'un projet, de bout en bout

À garder pour tout projet futur

1. *Cadrer* : tâche, métrique, baseline, contrainte métier → **2.** *Explorer* (EDA) : comprendre avant de modéliser → **3.** *Sceller* le test, préparer dans un *pipeline* (anti-fuite) → **4.** établir la *baseline* → **5.** *comparer* les modèles en validation croisée stratifiée → **6.** *régler* le meilleur (GridSearch), test toujours scellé → **7.** *évaluer* une fois sur le test : AUC et matrice de confusion et seuil métier → **8.** *interpréter* et *présenter*.

À retenir — Séance 11 (1/2)

La démarche

- Un projet ML est une **démarche ordonnée**, pas un choix de modèle : cadrer → explorer → préparer → comparer → régler → évaluer → interpréter → présenter.
- **Sceller le test en premier** et préparer dans un **pipeline** : la seule parade fiable à la fuite de données (S4).
- Toujours une **baseline** ; comparer les modèles *loyalement* en validation croisée stratifiée (S7) ; régler sans toucher au test.
- Sur *tabulaire*, les modèles se tiennent ; le plus simple suffit souvent (S8, S10).

À retenir — Séance 11 (2/2)

Le jugement

- L'**accuracy** ment sur les données déséquilibrées : 88 % en n'attrapant aucun malade. Juger à l'**AUC**, à la **précision** et au **rappel** (S6).
- Le **seuil** de décision est un choix *métier*, pas 0,5 par défaut : l'abaisser pour gagner du rappel quand rater un cas est grave.
- **Interpréter** : les variables importantes doivent recouper l'EDA et le métier — gage de confiance.
- La **soutenance** valorise la rigueur et l'honnêteté méthodologique, pas un dixième d'AUC.

Le mot de la fin

Vous savez faire du Machine Learning

En douze séances, vous êtes passés de « qu'est-ce que le ML ? » à un projet complet, mené avec la rigueur d'un professionnel. Vous savez préparer des données sans tricher, choisir et comparer des modèles, les évaluer honnêtement, en lire les pièges, et les expliquer. Aucune de ces compétences n'est propre à un outil : elles vous serviront quel que soit le langage, la bibliothèque ou le problème de demain.

La suite — apprentissage profond, traitement du langage, vision — s'appuiera sur ces fondations. Mais la démarche restera la même : cadrer, préparer, comparer, évaluer, interpréter. C'est cela, le métier. Bon courage pour vos projets.

Références

- Géron — *Hands-On Machine Learning*, 3^e éd., O'Reilly, 2022 (chap. 2 : un projet de bout en bout).
- Huyen — *Designing Machine Learning Systems*, O'Reilly, 2022 (mise en production).
- Kuhn, Johnson — *Applied Predictive Modeling*, Springer, 2013.
- Müller, Guido — *Introduction to Machine Learning with Python*, O'Reilly, 2016.
- Plateforme Zindi (zindi.africa) : compétitions de science des données pour l'Afrique.

Ressources de la séance

- **Notebook** · [TP 11 — Capstone](#) (114 Ko)
- **TD** · [Fiche d'exercices 11](#) (137 Ko)