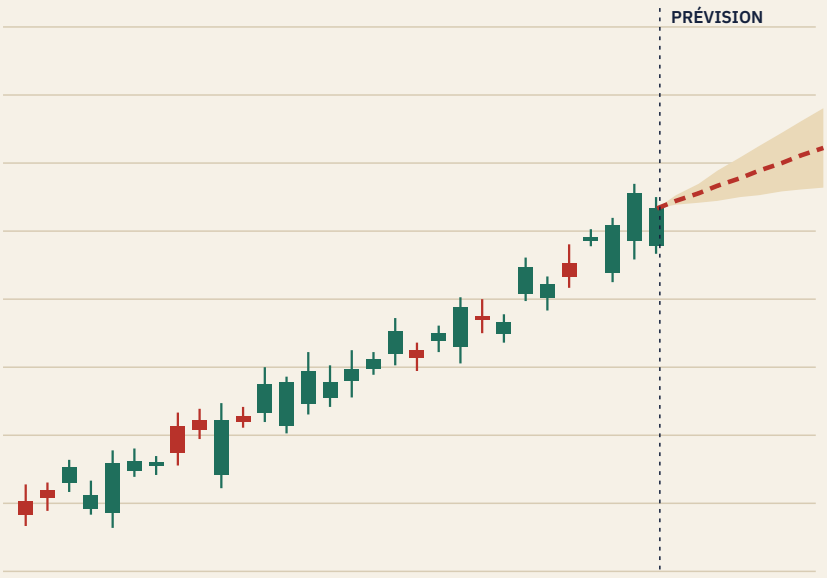
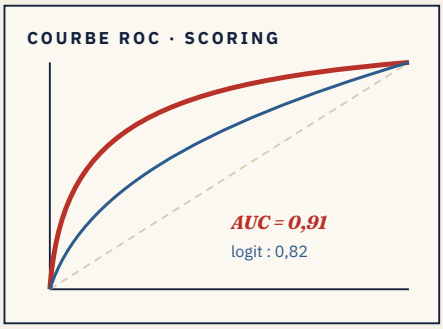
PRÉVISION · RISQUE DE CRÉDIT · FRAUDE

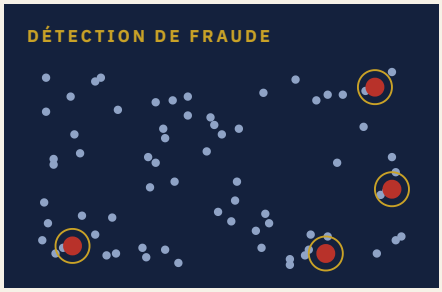
Machine Learning *en Finance*

Des données brutes aux modèles déployés : méthodes, code Python commenté et bonnes pratiques réglementaires pour la banque, l'assurance et les marchés.



Cours simulé et prévision avec intervalle de confiance





7	20+	3	XAI
CHAPITRES	CODES PYTHON	CAS D'USAGE COMPLETS	SHAP & GOUVERNANCE



EL KHLIFI Imad
Doctorant en Économie & Gestion · Data & BI
FSJES Tanger · Université Abdelmalek Essaâdi

ÉDITION 2026

ÉDITORIAL

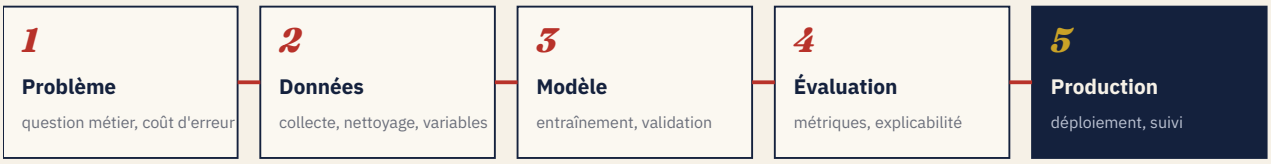
Prédire, *mesurer*, expliquer.

Pourquoi chaque analyste financier doit aujourd'hui comprendre le machine learning — et en connaître les limites.

Les banques notent leurs clients, les assureurs détectent les sinistres suspects, les gérants anticipent la volatilité : derrière chacune de ces décisions, des modèles apprennent désormais à partir des données. Le machine learning n'a pas remplacé l'économétrie ; il l'a complétée, en capturant des relations non linéaires et des interactions que les modèles classiques peinent à saisir.

Mais la finance n'est pas un terrain comme les autres. Les données sont bruitées, les événements rares, les régimes changeants, et chaque décision automatisée engage la responsabilité de l'institution devant ses clients et son régulateur. Un modèle performant mais opaque, ou entraîné sur un passé qui ne se répétera pas, peut coûter très cher.

Ce guide suit donc une ligne claire : **des méthodes éprouvées, du code reproductible et une exigence constante d'explicabilité**. Chaque chapitre associe les concepts, un exemple en Python et les pièges à éviter.



Le cycle de vie d'un projet de machine learning en finance

> *« Un modèle n'a de valeur que s'il est juste, compris et utilisé. »*

Pour qui ?

Étudiants et doctorants en finance, économie et data science ; analystes risques, crédit, conformité et audit.

Prérequis

Bases de statistiques et de Python (pandas). Les notions de machine learning sont introduites pas à pas.

Outils

Python 3, pandas, scikit-learn, XGBoost / LightGBM, statsmodels, SHAP — tous gratuits et open source.

SOMMAIRE

Au *sommaire*

01 **Le machine learning appliqué à la finance**
Concepts, types d'apprentissage, cas d'usage, cadre réglementaire

02 **Préparer les données financières**
Sources, qualité, variables, fuites d'information, déséquilibre

03 **Les modèles essentiels**
Régressions, arbres, forêts, boosting, réseaux de neurones, validation

04 **Prévoir : séries temporelles et volatilité**
ARIMA, GARCH, boosting, LSTM, validation glissante

05 **Mesurer le risque de crédit**
PD, LGD, EAD, scorecard, XGBoost, AUC, Gini, IFRS 9

06 **Détecter la fraude**
Classes rares, précision-rappel, Isolation Forest, coûts, temps réel

07 **Expliquer, gouverner, déployer**
SHAP, gestion du risque de modèle, AI Act, dérive, MLOps

A **Annexes**
Aide-mémoire des métriques · Jeux de données · Feuille de route · Références

CHAPITRE

01

MACHINE LEARNING EN FINANCE

Le machine learning appliqué à la finance

> *Ce que l'apprentissage automatique apporte — et n'apporte pas — à l'analyse financière, les grandes familles de méthodes et le cadre réglementaire dans lequel elles s'inscrivent.*



AU PROGRAMME

1.1 Définitions et vocabulaire

1.2 Trois façons d'apprendre

1.3 Économétrie ou machine learning ?

1.4 Panorama des cas d'usage

1.5 Le cadre réglementaire

1.6 Votre environnement de travail

SECTION 1.1

Définitions et vocabulaire

Le **machine learning** (apprentissage automatique) désigne l'ensemble des méthodes qui permettent à un programme d'apprendre une relation à partir d'exemples, plutôt que de la recevoir sous forme de règles écrites à la main. Un modèle de scoring n'est pas programmé avec « si le revenu est inférieur à X, refuser » : il découvre, à partir de milliers de dossiers passés, les combinaisons de variables associées au défaut.

Intelligence artificielle

Le domaine le plus large : tout système qui accomplit des tâches associées à l'intelligence humaine.

Machine learning

Sous-ensemble de l'IA : des modèles qui apprennent à partir de données (arbres, forêts, boosting…).

Deep learning

Sous-ensemble du ML : réseaux de neurones profonds, puissants sur le texte, l'image et les séquences longues.

TERME	SIGNIFICATION
Observation	Une ligne de données : un client, une transaction, une date
Variables explicatives (features)	Les colonnes utilisées pour prédire : revenu, ancienneté, ratios financiers…
Cible (target)	Ce que l'on veut prédire : défaut (oui / non), montant, rendement
Entraînement	Ajustement des paramètres du modèle sur un échantillon de données
Généralisation	Capacité à bien prédire sur des données jamais vues
Hyperparamètres	Réglages choisis avant l'entraînement : profondeur d'un arbre, taux d'apprentissage

Tableau 1.1 — Le vocabulaire de base

SECTION 1.2

Trois façons d'apprendre

I Supervisé

Les exemples sont étiquetés : on connaît la bonne réponse. **Classification** (défaut, fraude) ou **régression** (montant, rendement).

Scoring de crédit, prévision du chiffre d'affaires.

II Non supervisé

Pas d'étiquette : le modèle découvre une structure. **Segmentation** (clustering), **détection d'anomalies**, réduction de dimension.

Segmentation de clientèle, transactions atypiques.

III Par renforcement

Un agent apprend par essais et erreurs en maximisant une récompense dans un environnement.

Exécution d'ordres, allocation dynamique — usage plus expérimental.

Le dilemme biais-variance

Un modèle trop simple ne capte pas la relation (**sous-apprentissage**) ; un modèle trop complexe apprend le bruit de l'échantillon (**surapprentissage**). Toute la pratique du machine learning consiste à trouver le bon niveau de complexité, en mesurant l'erreur sur des données que le modèle n'a pas vues.

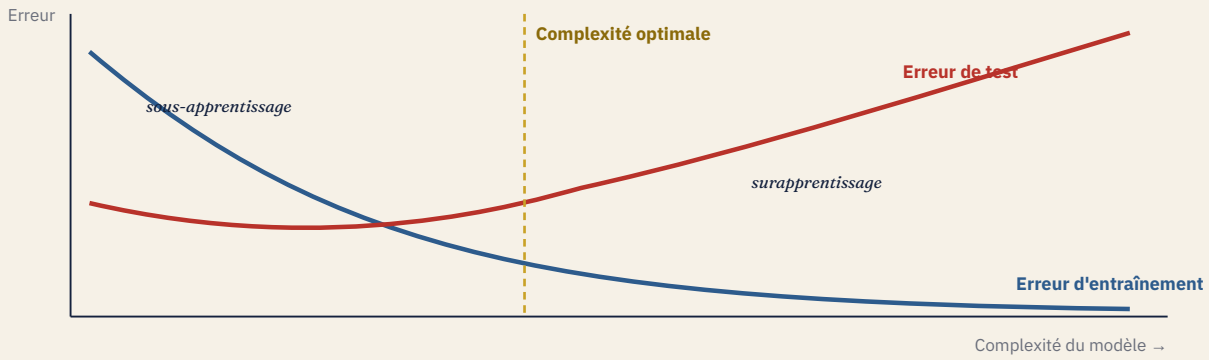


FIGURE 1.1 *L'erreur d'entraînement baisse toujours avec la complexité ; l'erreur de test, elle, remonte au-delà d'un certain point.*

SECTION 1.3

Économétrie ou machine learning ?

Les deux approches ne s'opposent pas : elles répondent à des questions différentes. L'économétrie cherche à **estimer et interpréter** un effet ; le machine learning cherche à **prédire** le mieux possible.

DIMENSION	ÉCONOMÉTRIE	MACHINE LEARNING
Objectif principal	Inférence : estimer β et sa significativité	Prédiction : minimiser l'erreur hors échantillon
Point de départ	Une théorie et un modèle spécifié	Les données ; la forme est apprise
Relations	Souvent linéaires, spécifiées à l'avance	Non linéaires, interactions automatiques
Validation	Tests statistiques, diagnostics des résidus	Validation croisée, échantillon de test
Interprétation	Directe (coefficients)	Indirecte (SHAP, importance des variables)
Dimension	Peu de variables	Des centaines ou milliers de variables

Tableau 1.2 — Deux cultures complémentaires

> **À RETENIR**
> La recherche récente combine les deux : utiliser le ML pour **prédire** (par exemple les rendements d'actifs, comme Gu, Kelly et Xiu, 2020) ou pour contrôler un grand nombre de variables, puis l'économétrie pour **interpréter** et tester des hypothèses causales.

SECTION 1.4

Panorama des cas d'usage

DOMAINE	PROBLÈME	TYPE	MÉTHODES TYPIQUES
Crédit	Probabilité de défaut d'un emprunteur	Classification	Logit, XGBoost, LightGBM
Fraude	Transactions frauduleuses par carte	Classification rare / anomalies	Boosting, Isolation Forest, autoencodeurs
Marchés	Volatilité, rendements, risque de marché	Régression / séries temporelles	GARCH, boosting, LSTM
Trésorerie	Flux de trésorerie, dépôts, liquidité	Prévision	ARIMA, Prophet, boosting
LCB-FT	Blanchiment, financement du terrorisme	Anomalies, graphes	Réseaux de transactions, scoring
Assurance	Tarification, sinistralité, résiliation	Régression / classification	GLM, GBM
Relation client	Attrition, appétence, segmentation	Classification / clustering	Forêts, k-means
Documents	Analyse de rapports, actualités, contrats	Traitement du langage	Modèles de langage, analyse de sentiment

Tableau 1.3 — Où le machine learning crée de la valeur en finance

PD
probabilité de défaut, cœur du risque de crédit (chapitre 5)

0,17 %
part des fraudes dans le jeu de données public de référence sur les cartes (chapitre 6)

t + h
horizon de prévision, à fixer avant tout modèle (chapitre 4)

SHAP
pour expliquer chaque décision au client et au régulateur (chapitre 7)

SECTION 1.5

Le cadre réglementaire

En finance, un modèle n'est pas seulement un outil technique : c'est un objet réglementé, qui doit être documenté, validé et contrôlé.

Cadres prudentiels et comptables

- **Bâle II / III** : modèles internes de notation (PD, LGD, EAD) soumis à l'approbation du superviseur
- **IFRS 9** : pertes de crédit attendues, prospectives
- **Gestion du risque de modèle** : inventaire, validation indépendante, suivi (référence : SR 11-7 aux États-Unis)
- Au Maroc : exigences de **Bank Al-Maghrib** en matière de gestion des risques et de gouvernance

Intelligence artificielle

- **AI Act européen** : l'évaluation de la solvabilité et le scoring de crédit des personnes physiques sont classés « à haut risque »
- Exigences : gestion des risques, qualité des données, documentation, supervision humaine, transparence
- Protection des données personnelles : RGPD en Europe, loi 09-08 au Maroc
- Non-discrimination : interdiction d'utiliser des critères protégés, directement ou par procuration

REPÈRE 2026

L'accord « Digital Omnibus » adopté en 2026 a reporté l'application des obligations de l'AI Act pour les systèmes à haut risque de l'annexe III — dont le scoring de crédit — du 2 août 2026 au **2 décembre 2027**. Le report concerne le calendrier, pas le périmètre : les institutions ont intérêt à documenter leurs modèles dès maintenant.

SECTION 1.6

Votre environnement de travail

Tout le code de ce guide utilise des bibliothèques Python libres et gratuites. Une installation via Anaconda ou un simple environnement virtuel suffit ; Google Colab permet aussi de tout exécuter dans le navigateur.

LISTING 1 · INSTALLATION ET VÉRIFICATION PYTHON

```python
# Créer un environnement dédié et installer les bibliothèques
# python -m venv ml-finance && source ml-finance/bin/activate
# pip install pandas numpy scikit-learn xgboost lightgbm statsmodels arch shap matplotlib

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import xgboost as xgb
import shap

print(pd.__version__, xgb.__version__)
```

BIBLIOTHÈQUE	RÔLE
pandas, NumPy	Manipulation des données et calcul numérique
scikit-learn	Modèles classiques, préparation, validation, métriques
XGBoost, LightGBM, CatBoost	Gradient boosting performant sur données tabulaires
statsmodels, arch	Économétrie, ARIMA, modèles GARCH
SHAP	Explicabilité des prédictions
PyTorch / Keras	Réseaux de neurones et LSTM

EN PRATIQUE

Fixez toujours la graine aléatoire (`random_state=42`) et notez les versions des bibliothèques : vos résultats deviennent reproductibles, ce que tout validateur ou évaluateur exigera.

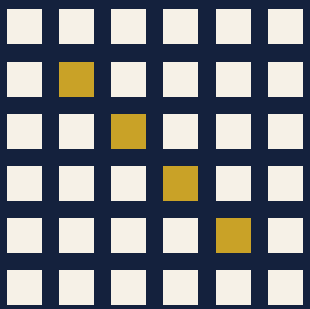
CHAPITRE

02

MACHINE LEARNING EN FINANCE

Préparer les données financières

> *80 % du travail d'un projet de machine learning se joue avant le modèle : collecter, nettoyer, construire des variables pertinentes et, surtout, éviter que l'avenir ne s'invite dans l'entraînement.*



AU PROGRAMME

2.1 Les sources de données

2.2 Qualité et nettoyage

2.3 Construire les variables

2.4 Fuites d'information

2.5 Découper les données

2.6 Classes déséquilibrées

SECTION 2.1

Les sources de données

FAMILLE	EXEMPLES	PARTICULARITÉS
Données internes	Historique des crédits, transactions, comptes, impayés	Riches mais biaisées : on n'observe que les dossiers acceptés
États financiers	Bilans, comptes de résultat, ratios	Fréquence annuelle ou semestrielle, retards de publication
Données de marché	Cours, volumes, taux, indices, volatilité implicite	Haute fréquence, non stationnaires
Macroéconomie	PIB, inflation, chômage, taux directeur	Révisées après publication : attention aux millésimes
Bureaux de crédit	Incidents de paiement, encours, historique	Accès réglementé
Données alternatives	Texte (rapports, presse), données ESG, géolocalisation	Prometteuses mais à valider soigneusement

Tableau 2.1 — Les grandes sources de données financières

ATTENTION

Biais de sélection. Un modèle de crédit n'est entraîné que sur les clients *acceptés* par le passé : il ignore le comportement des refusés. Ce biais, traité par les techniques de « reject inference », doit au minimum être signalé dans la documentation.

SECTION 2.2

Qualité et nettoyage

Valeurs manquantes

- Comprendre **pourquoi** elles manquent : un revenu absent n'est pas aléatoire
- Imputer par la médiane, par groupe ou par modèle
- Ajouter un indicateur binaire « valeur manquante » : souvent très prédictif
- Les modèles de boosting gèrent nativement les manquants

Valeurs extrêmes

- Distinguer erreur de saisie et événement réel (krach, défaut)
- Écrêter (winsoriser) aux 1er et 99e centiles
- Transformations : logarithme des montants
- Ne jamais supprimer les événements rares que l'on cherche à prédire

LISTING 2 · DIAGNOSTIC ET NETTOYAGE D'UNE BASE DE CRÉDITS PYTHON

```python
import pandas as pd
import numpy as np

df = pd.read_csv("credits.csv", parse_dates=["date_octroi"])

# 1. Diagnostic rapide
print(df.shape)
print(df.isna().mean().sort_values(ascending=False).head(10))   # taux de manquants
print(df.describe().T)

# 2. Indicateur de manquant + imputation par la médiane
df["revenu_manquant"] = df["revenu"].isna().astype(int)
df["revenu"] = df["revenu"].fillna(df["revenu"].median())

# 3. Winsorisation des montants aux centiles 1 % et 99 %
bas, haut = df["montant"].quantile([0.01, 0.99])
df["montant"] = df["montant"].clip(bas, haut)
df["log_montant"] = np.log1p(df["montant"])
```

SECTION 2.3

Construire les variables (feature engineering)

Les meilleurs gains de performance viennent rarement d'un algorithme plus sophistiqué, mais de **variables mieux construites**, qui traduisent l'expertise métier.

DOMAINE	VARIABLES CONSTRUITES
Crédit	Taux d'endettement, reste à vivre, ancienneté, nombre d'incidents sur 12 mois, utilisation des lignes
Entreprises	Ratios de liquidité, de levier, de rentabilité, couverture des intérêts, variation annuelle
Marchés	Rendements passés (retards), volatilité glissante, moyennes mobiles, momentum, volumes relatifs
Fraude	Montant / montant moyen du client, nombre de transactions sur 1 h, nouveau commerçant, distance au domicile
Calendrier	Jour de la semaine, fin de mois, jours fériés, saisonnalité

LISTING 3 · VARIABLES RETARDÉES ET GLISSANTES PYTHON

```python
# Variables de séries temporelles sur des prix journaliers
px = pd.read_csv("prix.csv", parse_dates=["date"], index_col="date")
px["ret"] = np.log(px["close"]).diff()                    # rendement logarithmique

for k in [1, 2, 5, 10]:
    px[f"ret_lag{k}"] = px["ret"].shift(k)                # rendements passés
px["vol_20j"] = px["ret"].rolling(20).std().shift(1)      # volatilité glissante
px["mm_50"]   = px["close"].rolling(50).mean().shift(1)   # moyenne mobile
px["ecart_mm"] = px["close"].shift(1) / px["mm_50"] - 1
px["jour"] = px.index.dayofweek

px = px.dropna()
```

> **À RETENIR**
>
> Remarquez le `.shift(1)` systématique : chaque variable ne doit utiliser que l'information **disponible à la date de la décision**. C'est le principe le plus important de la modélisation financière.

SECTION 2.4

Les fuites d'information : l'erreur n° 1

Une **fuite d'information** (data leakage) se produit lorsque le modèle utilise, pendant l'entraînement, une information qui ne serait pas disponible au moment de la prédiction. Le résultat est toujours le même : des performances spectaculaires en test… et décevantes en production.

FUITE	EXEMPLE	CORRECTION
Variable postérieure	« Nombre de relances » pour prédire le défaut	N'utiliser que les données connues à l'octroi
Biais d'anticipation	Volatilité calculée avec la journée à prédire	Décaler toutes les variables (shift)
Prétraitement global	Normaliser avant de découper en train / test	Ajuster le prétraitement sur le seul entraînement (Pipeline)
Doublons	Le même client dans le train et le test	Découper par client (GroupKFold)
Données révisées	PIB révisé utilisé comme s'il était connu à l'époque	Utiliser les millésimes de publication

« Si le résultat semble trop beau pour être vrai, cherchez la fuite. »

SECTION 2.5

Découper les données

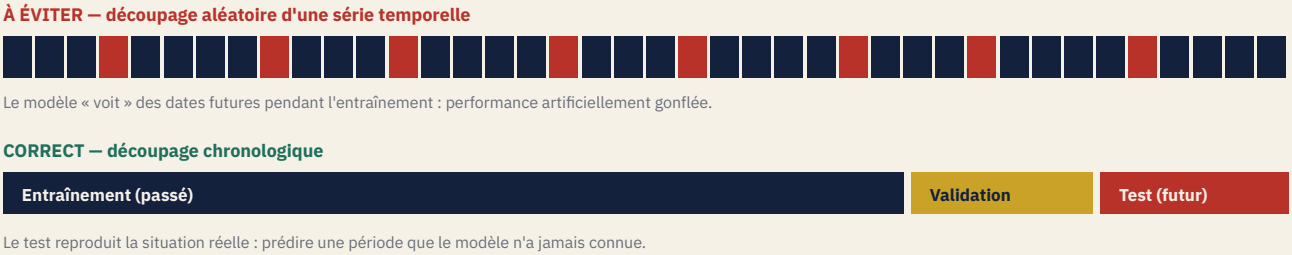


FIGURE 2.1 *Pour toute donnée ordonnée dans le temps, le découpage doit respecter la chronologie.*

LISTING 4 · DÉCOUPAGE CHRONOLOGIQUE ET PIPELINE PYTHON

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

# Découpage chronologique : 70 % passé pour entraîner, 30 % futur pour tester
df = df.sort_values("date_octroi")
coupe = int(len(df) * 0.7)
train, test = df.iloc[:coupe], df.iloc[coupe:]

# Le Pipeline garantit que la normalisation n'apprend que sur le train
modele = Pipeline([("scale", StandardScaler()),
                   ("logit", LogisticRegression(max_iter=1000))])

# Validation croisée respectant le temps
tscv = TimeSeriesSplit(n_splits=5)
```

SECTION 2.6

Les classes déséquilibrées

Défauts et fraudes sont rares : quelques pour cent des crédits, souvent moins de 1 % des transactions. Un modèle qui prédit toujours « pas de défaut » affiche alors 99 % d'exactitude… et ne sert à rien.

Changer de métrique

Abandonner l'exactitude au profit de l'AUC, du rappel, de la précision ou de l'aire sous la courbe précision-rappel.

Pondérer

`class_weight="balanced"` ou `scale_pos_weight` : pénaliser davantage les erreurs sur la classe rare.

Rééchantillonner

Sous-échantillonner la classe majoritaire ou générer des exemples synthétiques (SMOTE, Chawla et al., 2002) — **uniquement sur l'entraînement**.

> **ATTENTION**
>
> Le rééchantillonnage modifie la proportion de positifs : les probabilités prédites ne sont plus calibrées. Si vous avez besoin de vraies probabilités (PD réglementaire, provisions), recalibrez-les ensuite sur des données non rééchantillonnées.

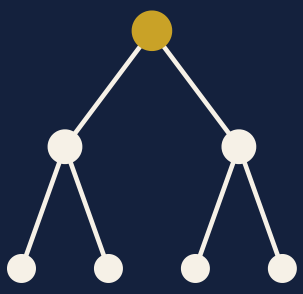
CHAPITRE

03

MACHINE LEARNING EN FINANCE

Les modèles essentiels

> *Des régressions aux réseaux de neurones, les algorithmes qui couvrent l'immense majorité des besoins en finance — et la façon rigoureuse de les comparer.*



AU PROGRAMME

3.1	Régressions linéaire et logistique
3.2	Arbres de décision
3.3	Forêts aléatoires
3.4	Gradient boosting
3.5	Réseaux de neurones
3.6	Validation et réglage
3.7	Quel modèle choisir ?

SECTION 3.1

Régressions linéaire et logistique

La **régression logistique** reste la référence du scoring de crédit : simple, stable, interprétable et acceptée par les régulateurs. Elle modélise la probabilité d'un événement (défaut, fraude) comme une fonction sigmoïde d'une combinaison linéaire des variables.

$$P(y = 1 \mid x) = 1 / \left(1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k)}\right)$$

Chaque coefficient β se lit comme l'effet d'une variable sur le logarithme de la cote (log-odds).

+ Forces

- Interprétation directe des coefficients
- Probabilités bien calibrées
- Peu de données nécessaires, très stable
- Standard réglementaire et comptable

− Limites

- Relations supposées linéaires (en log-odds)
- Interactions à créer à la main
- Sensible à la multicolinéarité
- Performance souvent inférieure au boosting

EN PRATIQUE

Avec la **régularisation** (Ridge L2, Lasso L1, Elastic Net), la régression gère des dizaines de variables corrélées ; le Lasso met à zéro les coefficients inutiles et réalise une sélection de variables automatique.

SECTION 3.2

Arbres de décision

Un arbre découpe successivement l'échantillon selon la variable et le seuil qui séparent le mieux les classes : « taux d'endettement > 40 % ? », puis « ancienneté < 2 ans ? »… Chaque feuille donne une prédiction.

Avantages

- Lisible comme une suite de règles métier
- Capte les non-linéarités et interactions
- Aucune normalisation nécessaire

Inconvénient majeur

Un arbre seul est **instable** : un petit changement de données modifie toute sa structure, et un arbre profond surapprend. D'où les méthodes d'ensemble, qui combinent des centaines d'arbres.

SECTION 3.3

Forêts aléatoires

Une **forêt aléatoire** (Breiman, 2001) entraîne de nombreux arbres sur des échantillons bootstrap et, à chaque division, sur un sous-ensemble aléatoire de variables ; la prédiction finale est la moyenne (ou le vote) des arbres. Le hasard décorrèle les arbres et réduit fortement la variance.

LISTING 5 · FORÊT ALÉATOIRE AVEC SCIKIT-LEARN PYTHON

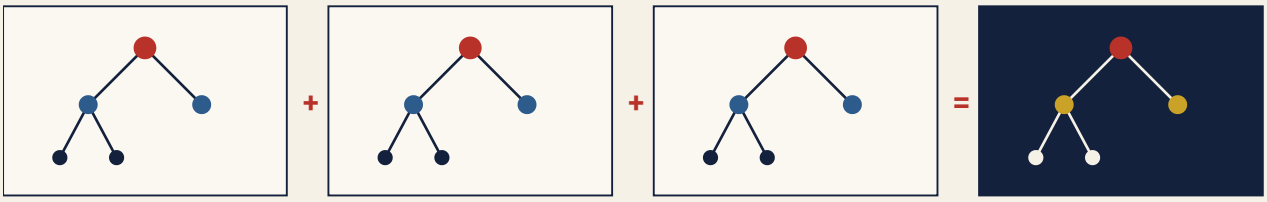
```python
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=500,        # nombre d'arbres
    max_depth=8,             # profondeur maximale : limite le surapprentissage
    min_samples_leaf=50,     # au moins 50 observations par feuille
    class_weight="balanced", # classes déséquilibrées
    n_jobs=-1, random_state=42)
rf.fit(X_train, y_train)
proba = rf.predict_proba(X_test)[:, 1]
```

SECTION 3.4

Gradient boosting : la référence sur données tabulaires

Le **gradient boosting** construit les arbres **séquentiellement** : chaque arbre corrige les erreurs des précédents. Ses implémentations modernes — **XGBoost** (Chen & Guestrin, 2016), **LightGBM** (Ke et al., 2017) et **CatBoost** — dominent les compétitions et les benchmarks sur données tabulaires, y compris en scoring de crédit (Lessmann et al., 2015).



Chaque nouvel arbre apprend les erreurs résiduelles des précédents, avec un petit pas (taux d'apprentissage).

FIGURE 3.1 *Le principe du boosting : une somme d'arbres faibles, chacun ciblant les erreurs restantes.*

LISTING 6 · XGBOOST AVEC ARRÊT PRÉCOCE SUR UN ÉCHANTILLON DE VALIDATION PYTHON

```python
import xgboost as xgb

ratio = (y_train == 0).sum() / (y_train == 1).sum()   # déséquilibre des classes

modele = xgb.XGBClassifier(
    n_estimators=2000,       # beaucoup d'arbres, arrêtés par early stopping
    learning_rate=0.03,      # petit pas : plus robuste
    max_depth=4,             # arbres peu profonds
    subsample=0.8, colsample_bytree=0.8,
    min_child_weight=5, reg_lambda=1.0,
    scale_pos_weight=ratio,
    eval_metric="auc", early_stopping_rounds=100,
    random_state=42)

modele.fit(X_train, y_train, eval_set=[(X_valid, y_valid)], verbose=False)
print("Meilleure itération :", modele.best_iteration)
```

HYPERPARAMÈTRE	RÔLE	PLAGE USUELLE
learning_rate	Taille du pas de chaque arbre	0,01 – 0,1
max_depth / num_leaves	Complexité de chaque arbre	3 – 8 / 15 – 63
n_estimators + early stopping	Nombre d'arbres, arrêté quand la validation ne progresse plus	100 – 5 000
subsample, colsample_bytree	Part des lignes et des colonnes tirées par arbre	0,6 – 1
reg_lambda, reg_alpha	Régularisation L2 et L1	0 – 10

Tableau 3.1 — Les réglages qui comptent vraiment

SECTION 3.5

Réseaux de neurones

Un réseau de neurones enchaîne des couches de transformations non linéaires. Sur des données tabulaires de taille moyenne, il fait rarement mieux que le boosting ; il devient en revanche incontournable pour les **séquences** (LSTM, Hochreiter & Schmidhuber, 1997 ; Transformers), le **texte** (rapports annuels, actualités) et les très grands volumes.

Perceptron multicouche

Données tabulaires ; nécessite normalisation et réglage soigneux.

LSTM / GRU

Séries temporelles et séquences de transactions (chapitre 4).

Autoencodeurs

Détection d'anomalies par erreur de reconstruction (chapitre 6).

SECTION 3.6

Validation et réglage des hyperparamètres

La **validation croisée** estime la performance hors échantillon en entraînant et testant le modèle sur plusieurs découpages. En finance, on respecte la chronologie (`TimeSeriesSplit`) ou les groupes (un client = un seul pli).

LISTING 7 · RECHERCHE ALÉATOIRE D'HYPERPARAMÈTRES AVEC VALIDATION CHRONOLOGIQUE — PYTHON

```python
from sklearn.model_selection import RandomizedSearchCV, TimeSeriesSplit
from scipy.stats import uniform, randint
import lightgbm as lgb

grille = {"num_leaves": randint(15, 64), "learning_rate": uniform(0.01, 0.09),
          "min_child_samples": randint(20, 200), "subsample": uniform(0.6, 0.4),
          "colsample_bytree": uniform(0.6, 0.4)}

recherche = RandomizedSearchCV(
    lgb.LGBMClassifier(n_estimators=500, subsample_freq=1, random_state=42),
    grille, n_iter=40, scoring="roc_auc",
    cv=TimeSeriesSplit(n_splits=5), n_jobs=-1, random_state=42)
recherche.fit(X_train, y_train)
print(recherche.best_params_, round(recherche.best_score_, 3))
```

> **ATTENTION**
> L'échantillon de **test** ne sert qu'une fois, à la toute fin. Si vous le consultez pour choisir vos réglages, il devient un échantillon de validation et votre estimation de performance est biaisée à la hausse.

SECTION 3.7

Quel modèle choisir ?

MODÈLE	PERFORMANCE TABULAIRE	INTERPRÉTABILITÉ	DONNÉES NÉCESSAIRES	USAGE TYPE
Régression logistique	★★☆	★★★	Faible	Scorecard réglementaire, benchmark
Arbre de décision	★☆☆	★★★	Faible	Règles métier, segmentation
Forêt aléatoire	★★☆	★★☆	Moyenne	Modèle robuste de référence
Gradient boosting	★★★	★★☆ (avec SHAP)	Moyenne	Crédit, fraude, prévision tabulaire
Réseau de neurones	★★☆	★☆☆	Élevée	Séquences, texte, grands volumes

Tableau 3.2 — Une grille de choix pragmatique

Commencez toujours par un modèle simple : il fixe la barre que tout modèle complexe doit franchir pour justifier son opacité.

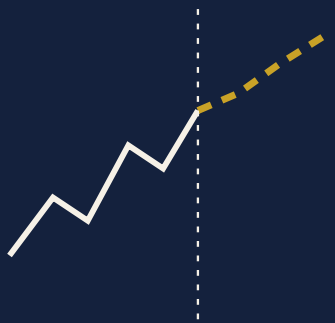
CHAPITRE

04

MACHINE LEARNING EN FINANCE

Prévoir : séries temporelles et volatilité

> *Prévoir une variable financière exige humilité et méthode : des modèles de référence solides, une validation qui respecte le temps et une évaluation honnête face à la marche aléatoire.*



AU PROGRAMME

4.1	Les spécificités des séries financières
4.2	Les modèles de référence
4.3	Prévoir la volatilité
4.4	Le machine learning appliqué
4.5	Réseaux récurrents
4.6	Validation glissante
4.7	Évaluer une prévision
4.8	Combiner les prévisions

SECTION 4.1

Les spécificités des séries financières

Ce qui rend la prévision difficile

- **Faible rapport signal / bruit** : les rendements sont presque imprévisibles
- **Non-stationnarité** : tendances, ruptures, changements de régime
- **Volatilité groupée** : les périodes agitées se suivent
- **Queues épaisses** : les événements extrêmes sont plus fréquents que sous la loi normale

Ce qui est plus prévisible

- La **volatilité** et le risque, bien plus que les rendements
- Les agrégats bancaires : dépôts, encours, liquidité
- Les variables macroéconomiques à court terme
- Les flux saisonniers (trésorerie, paiements)

$$r_t = ln(P_t) - ln(P_{t-1})$$

On modélise les rendements (stationnaires), rarement les prix bruts.

LISTING 8 · TESTER LA STATIONNARITÉ (DICKEY-FULLER AUGMENTÉ) PYTHON

```python
from statsmodels.tsa.stattools import adfuller

stat, pvalue, *_ = adfuller(px["close"])
print(f"Prix : p-value ADF = {pvalue:.3f}")          # souvent > 0,05 : non stationnaire

stat, pvalue, *_ = adfuller(px["ret"].dropna())
print(f"Rendements : p-value ADF = {pvalue:.3f}")    # < 0,05 : stationnaire
```

SECTION 4.2

Les modèles de référence

Avant tout modèle de machine learning, établissez des **références simples**. Un modèle sophistiqué qui ne bat pas la marche aléatoire n'a aucune valeur.

MODÈLE	PRINCIPE	QUAND L'UTILISER
Marche aléatoire (naïf)	Prévision = dernière valeur observée	Toujours, comme référence minimale
Moyenne saisonnière	Valeur de la même période l'an dernier	Séries saisonnières
ARIMA (Box-Jenkins)	Autorégression + moyenne mobile sur la série différenciée	Séries univariées, court terme
Lissage exponentiel (ETS)	Pondération décroissante du passé, tendance et saisonnalité	Prévisions opérationnelles
Prophet	Tendance + saisonnalités + jours fériés (Taylor & Letham, 2018)	Séries d'activité avec saisonnalités multiples

LISTING 9 · MODÈLE ARIMA SUR L'ENCOURS MENSUEL DES DÉPÔTS PYTHON

```python
from statsmodels.tsa.arima.model import ARIMA

serie = depots["encours"].asfreq("MS")              # série mensuelle
train, test = serie[:-12], serie[-12:]

arima = ARIMA(train, order=(1, 1, 1)).fit()
prev = arima.get_forecast(steps=12)
prevision = prev.predicted_mean
intervalle = prev.conf_int(alpha=0.05)              # intervalle à 95 %
```

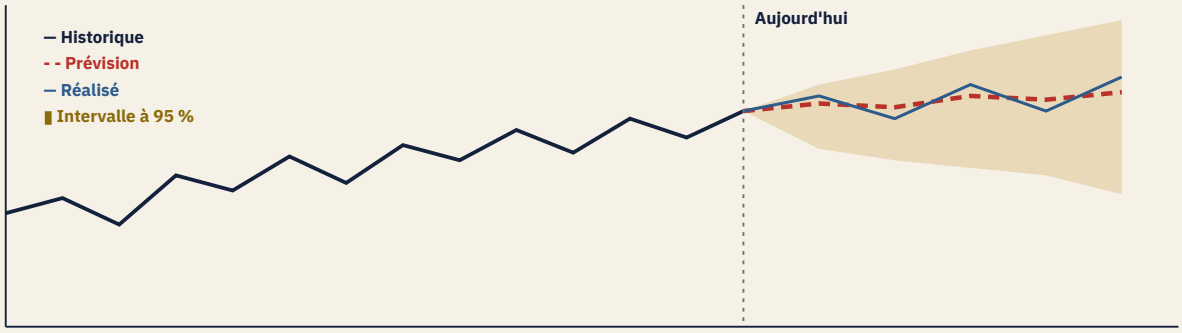


FIGURE 4.1 *Une prévision se présente toujours avec son incertitude : un point seul ne suffit pas à décider.*

SECTION 4.3

Prévoir la volatilité

Le modèle **GARCH(1,1)** (Bollerslev, 1986) reste la référence : la variance d'aujourd'hui dépend du choc d'hier et de la variance d'hier. Il alimente la VaR, l'allocation d'actifs et la tarification des options.

$$\sigma^2_t = \omega + \alpha\,\varepsilon^2_{t-1} + \beta\,\sigma^2_{t-1}$$

α mesure la réaction aux chocs, β la persistance ; α + β proche de 1 = forte persistance.

LISTING 10 · GARCH(1,1) AVEC LOI DE STUDENT PYTHON

```python
from arch import arch_model

r = 100 * px["ret"].dropna()                        # rendements en %
garch = arch_model(r, vol="GARCH", p=1, q=1, dist="t").fit(disp="off")
print(garch.summary())

prev_vol = garch.forecast(horizon=5).variance.iloc[-1] ** 0.5
print("Volatilité prévue à 5 jours (%) :", prev_vol.round(2).tolist())
```

EN PRATIQUE

Une approche hybride efficace : utiliser la prévision GARCH comme **variable explicative** d'un modèle de boosting, aux côtés de la volatilité réalisée, des volumes et de variables macroéconomiques.

SECTION 4.4

Le machine learning appliqué à la prévision

Pour utiliser un modèle tabulaire (XGBoost, LightGBM), on transforme la série en problème supervisé : chaque ligne contient les **valeurs passées** (retards), des statistiques glissantes et des variables exogènes ; la cible est la valeur à l'horizon *h*.

LISTING 11 · PRÉVOIR LA VOLATILITÉ RÉALISÉE À 5 JOURS AVEC LIGHTGBM — PYTHON

```python
import lightgbm as lgb

h = 5                                                  # horizon : 5 jours
px["cible"] = px["ret"].rolling(h).std().shift(-h)    # volatilité réalisée future
X_cols = ["ret_lag1", "ret_lag2", "ret_lag5", "ret_lag10", "vol_20j", "ecart_mm", "jour"]
data = px.dropna(subset=X_cols + ["cible"])

coupe = int(len(data) * 0.8)
X_tr, y_tr = data[X_cols][:coupe], data["cible"][:coupe]
X_te, y_te = data[X_cols][coupe:], data["cible"][coupe:]

gbm = lgb.LGBMRegressor(n_estimators=600, learning_rate=0.02, num_leaves=15,
                        min_child_samples=50, random_state=42)
gbm.fit(X_tr, y_tr)
prevision = gbm.predict(X_te)
```

ATTENTION

La cible utilise `shift(-h)` : elle regarde dans le futur, c'est normal pour une cible. Les variables explicatives, elles, ne doivent jamais le faire. Vérifiez aussi qu'aucune ligne du test ne partage sa fenêtre de cible avec l'entraînement : laissez un écart de *h* jours entre les deux.

SECTION 4.5

Réseaux récurrents : LSTM

Les **LSTM** lisent une séquence pas à pas en conservant une mémoire des informations utiles. Ils sont pertinents pour de longues séquences ou de multiples séries liées, mais exigent beaucoup de données, une normalisation soigneuse et un réglage coûteux.

LISTING 12 · UN LSTM COMPACT AVEC KERAS PYTHON

```python
import keras
from keras import layers

fenetre, n_var = 60, X_seq.shape[2]                  # 60 jours d'historique par exemple
lstm = keras.Sequential([
    layers.Input(shape=(fenetre, n_var)),
    layers.LSTM(32),
    layers.Dropout(0.2),
    layers.Dense(1)])
lstm.compile(optimizer="adam", loss="mse")
lstm.fit(X_seq_tr, y_seq_tr, validation_split=0.2, epochs=50, batch_size=64,
         callbacks=[keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)],
         shuffle=False)                              # ne pas mélanger le temps
```

SECTION 4.6

La validation glissante (walk-forward)

En prévision, la validation de référence consiste à **rejouer l'histoire** : entraîner sur le passé, prévoir la période suivante, avancer d'un pas, recommencer. On obtient une série d'erreurs hors échantillon qui reflète l'usage réel du modèle.

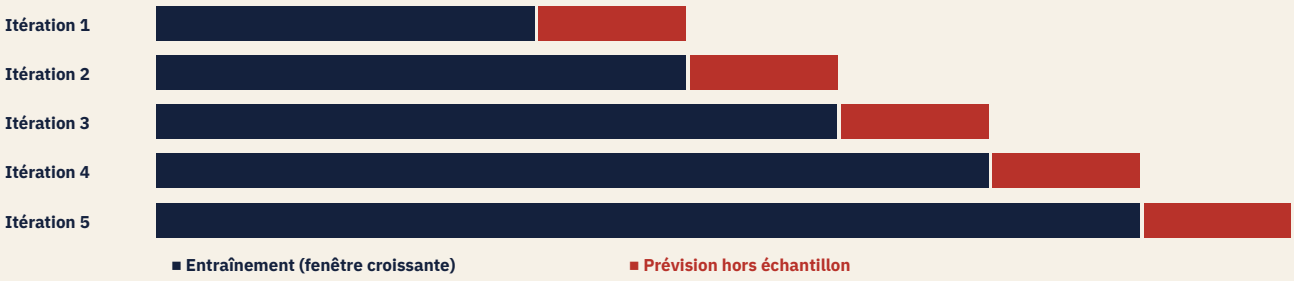


FIGURE 4.2 *La validation glissante avec fenêtre croissante. Une fenêtre fixe (glissante) est préférable si les régimes changent.*

SECTION 4.7

Évaluer une prévision

MÉTRIQUE	FORMULE (IDÉE)	REMARQUE
MAE	Moyenne des |erreurs|	Lisible, dans l'unité de la variable
RMSE	Racine de la moyenne des erreurs²	Pénalise les grosses erreurs
MAPE	Moyenne des |erreurs| / |valeurs réelles|	À éviter si la série approche de zéro
QLIKE	Perte adaptée à la variance	Recommandée pour comparer des prévisions de volatilité
Taux de bonne direction	% de signes correctement prévus	Utile en gestion, insuffisant seul

> **À RETENIR**
> Pour affirmer qu'un modèle prévoit **significativement** mieux qu'un autre, appliquez le test de **Diebold-Mariano** (1995) aux séries d'erreurs hors échantillon. Un écart de RMSE de quelques pour cent peut n'être que du bruit.

En prévision financière, battre la marche aléatoire de façon stable est déjà un résultat remarquable.

SECTION 4.8

Combiner les prévisions

Aucun modèle ne domine en permanence. Une **combinaison de prévisions** — même une simple moyenne — est souvent plus précise et plus stable que le meilleur modèle pris isolément, car les erreurs des différents modèles se compensent en partie.

LISTING 13 · COMPARER PUIS COMBINER TROIS PRÉVISIONS — PYTHON

```python
import numpy as np
from sklearn.metrics import mean_absolute_error

previsions = {"naif": prev_naif, "garch": prev_garch, "lgbm": prev_lgbm}
for nom, p in previsions.items():
    print(f"{nom:6s} MAE = {mean_absolute_error(y_te, p):.4f}")

# Moyenne simple, puis pondération inverse de l'erreur sur la validation
moyenne = np.mean(list(previsions.values()), axis=0)
poids = {k: 1 / mean_absolute_error(y_va, v) for k, v in previsions_validation.items()}
total = sum(poids.values())
ponderee = sum(poids[k] / total * previsions[k] for k in previsions)
print("Moyenne  MAE =", round(mean_absolute_error(y_te, moyenne), 4))
print("Pondérée MAE =", round(mean_absolute_error(y_te, ponderee), 4))
```

ÉTAPE	BONNE PRATIQUE
1. Références	Marche aléatoire, moyenne historique, ARIMA / GARCH
2. Challengers	Boosting sur variables retardées, LSTM si les données le justifient
3. Validation	Walk-forward sur plusieurs années, incluant des périodes de crise
4. Comparaison	MAE / RMSE / QLIKE et test de Diebold-Mariano
5. Décision	Retenir le modèle ou la combinaison la plus stable, pas le meilleur d'une seule période

Tableau 4.2 — Une démarche de prévision en cinq étapes

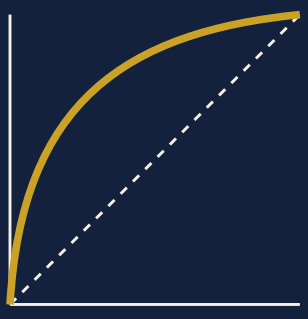
CHAPITRE

05

MACHINE LEARNING EN FINANCE

Mesurer le risque de crédit

> *Le cas d'usage historique du scoring : estimer la probabilité qu'un emprunteur fasse défaut, comparer le modèle traditionnel au gradient boosting, et produire des scores justes, stables et explicables.*



AU PROGRAMME

5.1	Les paramètres du risque
5.2	La scorecard traditionnelle
5.3	Un modèle complet en Python
5.4	Mesurer la discrimination
5.5	Calibrer les probabilités
5.6	Du score à la décision
5.7	IFRS 9 et équité

SECTION 5.1

Les paramètres du risque de crédit

$$EL = PD \times LGD \times EAD$$

Perte attendue = probabilité de défaut × perte en cas de défaut × exposition au moment du défaut

PD

Probabilité de défaut sur un horizon donné (souvent 12 mois). Problème de classification binaire.

LGD

Perte en cas de défaut : part de l'exposition non récupérée. Régression sur [0 ; 1], souvent bimodale.

EAD

Exposition au défaut : encours au moment du défaut, y compris les tirages sur lignes non utilisées.

Ce chapitre se concentre sur la **PD**, où le machine learning apporte le plus. La définition du défaut doit être fixée avant toute modélisation : par exemple, un retard de paiement de plus de 90 jours dans les 12 mois suivant l'octroi.

SECTION 5.2

La scorecard traditionnelle : WoE et IV

La méthode historique (Siddiqi, 2017) découpe chaque variable en classes, remplace chaque classe par son **poids de l'évidence** (Weight of Evidence), puis estime une régression logistique. Le résultat est une grille de points lisible par tous.

$$WoE_i = \ln(\%\,bons_i / \%\,mauvais_i) \cdot IV = \Sigma\,(\%\,bons_i - \%\,mauvais_i) \times WoE_i$$

L'Information Value (IV) mesure le pouvoir prédictif d'une variable.

IV	POUVOIR PRÉDICTIF
< 0,02	Inutile
0,02 – 0,10	Faible
0,10 – 0,30	Moyen
0,30 – 0,50	Fort
> 0,50	Suspect : vérifier la fuite

VARIABLE	CLASSE	POINTS
Ancienneté emploi	< 1 an	12
	1 – 5 ans	28
	> 5 ans	41
Taux d'endettement	> 45 %	5
	30 – 45 %	22
	< 30 %	38

Tableau 5.1 — Lecture de l'IV et extrait d'une grille de score (exemple illustratif)

LISTING 14 · CALCULER LE WOE ET L'IV D'UNE VARIABLE PYTHON

```python
import numpy as np, pandas as pd

def woe_iv(df, var, cible, n_classes=5):
    d = df[[var, cible]].copy()
    d["classe"] = pd.qcut(d[var], n_classes, duplicates="drop")
    t = d.groupby("classe", observed=True)[cible].agg(["count", "sum"])
    t["mauvais"] = t["sum"]; t["bons"] = t["count"] - t["sum"]
    t["%mauvais"] = t["mauvais"] / t["mauvais"].sum()
    t["%bons"] = t["bons"] / t["bons"].sum()
    t["WoE"] = np.log((t["%bons"] + 1e-6) / (t["%mauvais"] + 1e-6))
    iv = ((t["%bons"] - t["%mauvais"]) * t["WoE"]).sum()
    return t[["count", "mauvais", "WoE"]], iv

tableau, iv = woe_iv(df, "taux_endettement", "defaut")
print(tableau); print(f"IV = {iv:.3f}")
```

SECTION 5.3

Un modèle complet en Python

Comparons sur les mêmes données une régression logistique et un modèle XGBoost. L'exemple suppose une base de crédits avec une cible `defaut` (0 / 1) et une date d'octroi.

LISTING 15 · LOGIT CONTRE XGBOOST SUR UN DÉCOUPAGE CHRONOLOGIQUE PYTHON

```python
import pandas as pd, xgboost as xgb
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

df = pd.read_csv("credits.csv", parse_dates=["date_octroi"]).sort_values("date_octroi")
variables = ["revenu", "taux_endettement", "anciennete_emploi", "age",
             "nb_incidents_12m", "montant", "duree", "revenu_manquant"]

# Découpage chronologique : entraînement / validation / test
n = len(df); a, b = int(n * 0.6), int(n * 0.8)
tr, va, te = df.iloc[:a], df.iloc[a:b], df.iloc[b:]
X_tr, y_tr = tr[variables], tr["defaut"]
X_va, y_va = va[variables], va["defaut"]
X_te, y_te = te[variables], te["defaut"]

# 1. Référence : régression logistique
logit = Pipeline([("sc", StandardScaler()),
                  ("lr", LogisticRegression(max_iter=1000, class_weight="balanced"))])
logit.fit(X_tr, y_tr)
p_logit = logit.predict_proba(X_te)[:, 1]

# 2. Challenger : XGBoost avec arrêt précoce sur la validation
xgbm = xgb.XGBClassifier(n_estimators=3000, learning_rate=0.03, max_depth=4,
                         subsample=0.8, colsample_bytree=0.8, min_child_weight=10,
                         eval_metric="auc", early_stopping_rounds=150, random_state=42)
xgbm.fit(X_tr, y_tr, eval_set=[(X_va, y_va)], verbose=False)
p_xgb = xgbm.predict_proba(X_te)[:, 1]

for nom, p in [("Logit", p_logit), ("XGBoost", p_xgb)]:
    auc = roc_auc_score(y_te, p)
    print(f"{nom:8s}  AUC = {auc:.3f}   Gini = {2 * auc - 1:.3f}")
```

SECTION 5.4

Mesurer la discrimination

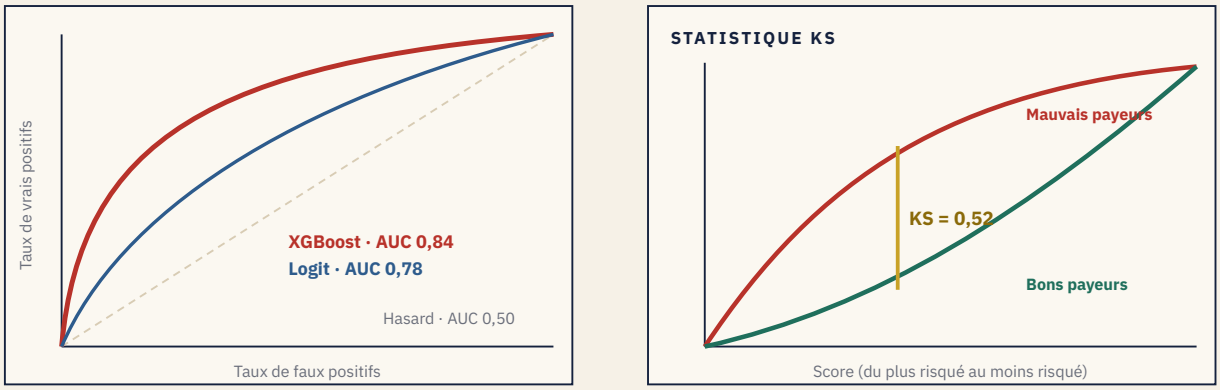


FIGURE 5.1 *À gauche, la courbe ROC compare la capacité des modèles à classer les mauvais payeurs avant les bons ; à droite, la statistique KS mesure l'écart maximal entre les deux distributions cumulées. Valeurs illustratives.*

MÉTRIQUE	DÉFINITION	REPÈRES EN SCORING
AUC	Probabilité qu'un mauvais payeur tiré au hasard ait un score plus risqué qu'un bon	0,70 acceptable · 0,80 bon · > 0,90 vérifier la fuite
Gini	2 × AUC − 1	Indicateur le plus utilisé par les banques
KS	Écart maximal entre distributions cumulées des bons et des mauvais	0,30 – 0,50 usuel
PSI	Stabilité de la distribution des scores dans le temps	< 0,10 stable · > 0,25 instabilité forte

SECTION 5.5

Calibrer les probabilités

Un bon classement ne garantit pas de bonnes probabilités. Or la PD alimente le capital réglementaire, les provisions et la tarification : si le modèle annonce 5 %, il faut observer environ 5 % de défauts dans cette population.

LISTING 16 · VÉRIFIER ET CORRIGER LA CALIBRATION — PYTHON

```python
from sklearn.calibration import calibration_curve, CalibratedClassifierCV
from sklearn.metrics import brier_score_loss

frac_obs, proba_moy = calibration_curve(y_te, p_xgb, n_bins=10, strategy="quantile")
print("Brier :", round(brier_score_loss(y_te, p_xgb), 4))

# Recalibration isotone sur l'échantillon de validation
xgb_ref = xgb.XGBClassifier(n_estimators=xgbm.best_iteration + 1, learning_rate=0.03,
                            max_depth=4, subsample=0.8, colsample_bytree=0.8,
                            min_child_weight=10, random_state=42).fit(X_tr, y_tr)
cal = CalibratedClassifierCV(xgb_ref, method="isotonic", cv="prefit").fit(X_va, y_va)
p_cal = cal.predict_proba(X_te)[:, 1]
```

À RETENIR

Selon votre version de scikit-learn, l'option `cv='prefit'` peut être remplacée par l'encapsulation du modèle dans `FrozenEstimator`. Le principe reste le même : calibrer sur des données non utilisées pour l'entraînement.

SECTION 5.6

Du score à la décision

Classes de risque

Les scores sont regroupés en 7 à 10 classes homogènes, avec une PD croissante et monotone. Chaque classe correspond à une politique : acceptation automatique, étude manuelle, garanties, refus.

Seuil d'acceptation

Le seuil se fixe par un arbitrage économique : marge d'intérêt des bons dossiers acceptés contre pertes des mauvais. Tracez le profit attendu selon le taux d'acceptation plutôt que de retenir un seuil à 0,5.

SECTION 5.7

IFRS 9 et équité



FIGURE 5.2 *Les trois étapes d'IFRS 9 : la PD détermine le passage d'une étape à l'autre et l'horizon de la perte attendue.*

Sous IFRS 9, les PD doivent être **« point-in-time »** et intégrer des scénarios macroéconomiques prospectifs : le machine learning permet de relier plus finement les variables macroéconomiques aux trajectoires de défaut.

ATTENTION

Équité. N'utilisez ni le sexe, ni l'origine, ni la religion — et méfiez-vous des variables qui les reproduisent indirectement (variables de substitution, comme certaines localisations). Mesurez les écarts de taux d'acceptation et d'erreur entre groupes et documentez-les : c'est une exigence éthique, juridique et, pour le scoring de crédit, une exigence explicite de l'AI Act.

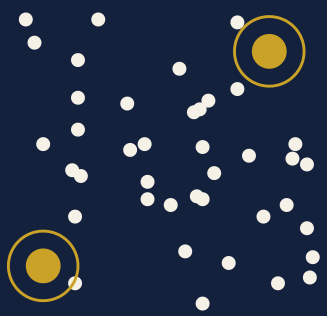
CHAPITRE

06

MACHINE LEARNING EN FINANCE

Détecter la fraude

> *Trouver une aiguille dans une botte de foin, en temps réel, face à un adversaire qui s'adapte : la détection de fraude impose ses propres métriques, ses propres modèles et un raisonnement en coûts.*



AU PROGRAMME

6.1	Un problème à part
6.2	Les bonnes métriques
6.3	Approche supervisée
6.4	Approche non supervisée
6.5	Choisir le seuil par les coûts
6.6	En production

SECTION 6.1

Un problème à part

Rare
souvent bien moins de 1 % des transactions

Adverse
les fraudeurs changent de méthode dès qu'ils sont repérés

Rapide
décision en quelques millisecondes lors du paiement

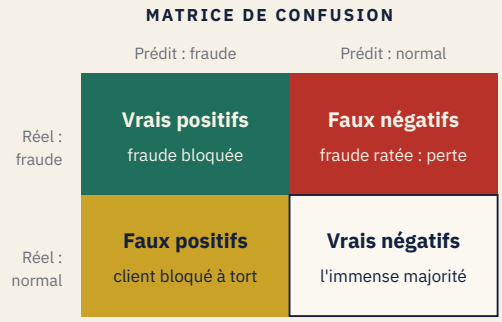
Tardif
l'étiquette arrive après coup : contestation du client, enquête

TYPE DE FRAUDE	EXEMPLES	SIGNAUX TYPIQUES
Paiement par carte	Carte volée ou clonée, achat en ligne frauduleux	Montant inhabituel, nouveau commerçant, pays lointain, rafale de transactions
Prise de contrôle de compte	Identifiants dérobés par hameçonnage	Nouvel appareil, changement de coordonnées puis virement
Fraude à l'ouverture	Identité usurpée ou synthétique	Incohérences documentaires, données partagées entre dossiers
Fraude interne	Détournement par un collaborateur	Opérations hors horaires, contournement des contrôles
Assurance	Sinistre fictif ou gonflé	Déclaration rapide après souscription, réseaux d'intervenants

Le jeu de données public de référence — les transactions par carte de clients européens collectées en partenariat avec l'Université libre de Bruxelles (Dal Pozzolo et al., 2015) — compte 492 fraudes sur 284 807 transactions, soit **0,17 %**. Il est idéal pour s'exercer.

SECTION 6.2

Les bonnes métriques



***Précision* = VP / (VP + FP)**
Parmi les alertes, quelle part est vraiment frauduleuse ?

***Rappel* = VP / (VP + FN)**
Parmi les fraudes, quelle part est détectée ?

***F1* = 2 · P · R / (P + R)**
Compromis entre précision et rappel.

L'exactitude (accuracy) est trompeuse : 99,8 % en ne détectant rien.

FIGURE 6.1 *La matrice de confusion et les métriques qui en découlent. Les deux types d'erreur n'ont pas le même coût.*

> **À RETENIR**
> Sur des classes très rares, préférez l'**aire sous la courbe précision-rappel** (PR-AUC, ou « average precision ») à l'AUC ROC : cette dernière reste élevée même quand la précision est faible, car le nombre énorme de vrais négatifs écrase le taux de faux positifs.

SECTION 6.3

Approche supervisée

Lorsque l'on dispose d'un historique de fraudes étiquetées, le gradient boosting avec pondération des classes est la référence. Les variables d'**agrégation comportementale** font l'essentiel de la performance.

LISTING 17 · VARIABLES COMPORTEMENTALES ET LIGHTGBM PONDÉRÉ — PYTHON

```python
import pandas as pd, lightgbm as lgb
from sklearn.metrics import average_precision_score, roc_auc_score

tx = pd.read_csv("transactions.csv", parse_dates=["horodatage"]).sort_values("horodatage")

# Variables comportementales par client, calculées sur le passé uniquement
g = tx.groupby("client_id")
tx["montant_moyen_client"] = g["montant"].transform(lambda s: s.shift(1).expanding().mean())
tx["ratio_montant"] = tx["montant"] / tx["montant_moyen_client"]
tx["nb_tx_1h"] = (tx.set_index("horodatage").groupby("client_id")["montant"]
                    .transform(lambda s: s.rolling("1h").count()).values - 1)
tx["nouveau_commercant"] = (~tx.duplicated(["client_id", "commercant_id"])).astype(int)

X_cols = ["montant", "ratio_montant", "nb_tx_1h", "nouveau_commercant", "heure", "distance_km"]
coupe = int(len(tx) * 0.8)
tr, te = tx.iloc[:coupe], tx.iloc[coupe:]

ratio = (tr["fraude"] == 0).sum() / (tr["fraude"] == 1).sum()
clf = lgb.LGBMClassifier(n_estimators=800, learning_rate=0.03, num_leaves=31,
                         scale_pos_weight=ratio, random_state=42)
clf.fit(tr[X_cols], tr["fraude"])
p = clf.predict_proba(te[X_cols])[:, 1]

print("PR-AUC :", round(average_precision_score(te["fraude"], p), 3))
print("ROC-AUC:", round(roc_auc_score(te["fraude"], p), 3))
```

EN PRATIQUE

Les meilleures variables comparent la transaction au **comportement habituel** du client : écart au montant moyen, fréquence récente, nouveauté du commerçant, de l'appareil ou du pays. Un modèle global sans ces agrégats voit chaque transaction isolément et passe à côté de l'essentiel.

SECTION 6.4

Approche non supervisée

Les étiquettes manquent ou arrivent tard, et les nouvelles fraudes ne ressemblent pas aux anciennes. Les méthodes non supervisées repèrent ce qui est **anormal**, sans savoir à quoi ressemble la fraude.

Isolation Forest

Isole chaque point par des coupures aléatoires : les anomalies s'isolent en peu de coupures (Liu, Ting & Zhou, 2008).

Autoencodeur

Un réseau apprend à reconstruire les transactions normales ; une erreur de reconstruction élevée signale une anomalie.

Analyse de graphes

Relie clients, cartes, appareils et adresses : les réseaux de fraude organisée apparaissent comme des communautés suspectes.

LISTING 18 · ISOLATION FOREST POUR PRIORISER LES TRANSACTIONS ATYPIQUES — PYTHON

```python
from sklearn.ensemble import IsolationForest

iso = IsolationForest(n_estimators=300, contamination=0.002, random_state=42)
iso.fit(tr[X_cols].fillna(0))

score_anomalie = -iso.score_samples(te[X_cols].fillna(0))   # plus élevé = plus anormal
te = te.assign(score_anomalie=score_anomalie)
alertes = te.nlargest(100, "score_anomalie")                 # 100 cas à examiner
```

> **À RETENIR**
> En pratique, les systèmes performants sont **hybrides** : règles métier pour les schémas connus, modèle supervisé pour les fraudes déjà observées, détection d'anomalies pour les nouvelles, et analystes humains pour trancher.

SECTION 6.5

Choisir le seuil par les coûts

Le bon seuil n'est jamais 0,5. Il résulte d'un arbitrage entre le **coût d'une fraude ratée** (montant perdu) et le **coût d'une fausse alerte** (traitement par un analyste, client mécontent, vente perdue) — et de la capacité de l'équipe d'enquête.

LISTING 19 · SEUIL MINIMISANT LE COÛT TOTAL — PYTHON

```python
import numpy as np

cout_fn = te["montant"].values      # une fraude ratée coûte son montant
cout_fp = 5.0                       # coût moyen d'une fausse alerte (traitement, client)
y = te["fraude"].values

seuils = np.linspace(0.01, 0.99, 99)
couts = [((p < s) & (y == 1)) @ cout_fn + ((p >= s) & (y == 0)).sum() * cout_fp
         for s in seuils]
s_opt = seuils[int(np.argmin(couts))]
print(f"Seuil optimal : {s_opt:.2f}  ·  coût total : {min(couts):,.0f}")
```

SECTION 6.6

En production

Contraintes opérationnelles

- Latence : quelques dizaines de millisecondes
- Variables calculées en flux, identiques à l'entraînement
- File d'alertes dimensionnée pour les analystes
- Retour des enquêtes réinjecté comme nouvelles étiquettes

Pièges

- **Dérive** : les fraudeurs s'adaptent, le modèle vieillit en quelques semaines
- **Étiquettes tardives** : une transaction récente « non frauduleuse » peut encore être contestée
- **Boucle de rétroaction** : on n'étiquette que ce que le modèle signale
- Réentraîner régulièrement et suivre la PR-AUC chaque semaine

CHAPITRE

07

MACHINE LEARNING EN FINANCE

Expliquer, gouverner, déployer

> *Un modèle n'a de valeur qu'utilisé avec confiance : expliquer ses décisions, le valider de façon indépendante, le déployer proprement et surveiller sa dérive dans le temps.*

AU PROGRAMME

7.1	Pourquoi expliquer ?
7.2	SHAP en pratique
7.3	La gestion du risque de modèle
7.4	Déployer
7.5	Surveiller la dérive
7.6	IA générative et finance

SECTION 7.1

Pourquoi expliquer ?

Le client
a le droit de comprendre un refus de crédit et de le contester.

Le régulateur
exige des modèles documentés, compréhensibles et non discriminatoires.

Le métier
n'adopte un modèle que s'il en comprend la logique.

Le data scientist
détecte grâce à l'explication les fuites, biais et erreurs.

MÉTHODE	PORTÉE	PRINCIPE
Coefficients (logit)	Globale	Effet de chaque variable, lisible directement
Importance par permutation	Globale	Perte de performance quand on mélange une variable
Dépendance partielle (PDP)	Globale	Effet moyen d'une variable sur la prédiction
LIME	Locale	Approximation linéaire autour d'une prédiction (Ribeiro et al., 2016)
SHAP	Locale et globale	Contribution de chaque variable selon les valeurs de Shapley (Lundberg & Lee, 2017)

SECTION 7.2

SHAP en pratique

Les valeurs **SHAP** répartissent l'écart entre la prédiction d'un client et la prédiction moyenne entre ses variables, de façon cohérente et additive. Pour les modèles d'arbres, leur calcul est exact et rapide.

EXPLICATION D'UNE DÉCISION — DOSSIER N° 4 127

Score de base (moyenne du portefeuille) : PD = 4,0 % → PD prédite pour ce client : 11,8 %

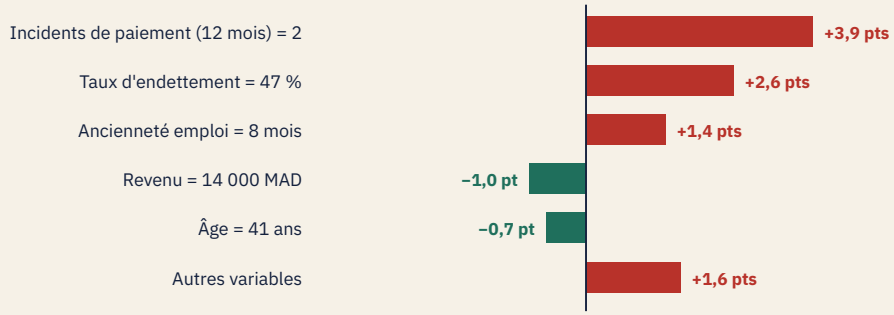


FIGURE 7.1 *Explication locale d'un refus : chaque barre indique la contribution d'une variable à la PD de ce client (exemple illustratif, échelle des probabilités).*

LISTING 20 · EXPLICATIONS GLOBALES ET LOCALES AVEC SHAP PYTHON

```python
import shap

explainer = shap.TreeExplainer(xgbm)
sv = explainer(X_te)                        # valeurs SHAP de chaque client du test

shap.plots.beeswarm(sv, max_display=10)     # vue globale : importance et sens des effets
shap.plots.waterfall(sv[0])                 # vue locale : explication du premier dossier

# Motifs de refus : les 3 variables qui augmentent le plus le risque du client i
i = 0
contrib = pd.Series(sv.values[i], index=X_te.columns).sort_values(ascending=False)
print(contrib.head(3))
```

ATTENTION

SHAP explique **le modèle**, pas la réalité : une contribution n'est pas un effet causal. Avec des variables très corrélées, les contributions se répartissent de façon parfois contre-intuitive. Présentez-les comme « ce qui a pesé dans la décision du modèle ».

SECTION 7.3

La gestion du risque de modèle

Un modèle erroné, mal utilisé ou dégradé fait courir un **risque de modèle**. Les banques le gèrent par un cycle de vie formalisé, inspiré notamment des lignes directrices SR 11-7 de la Réserve fédérale américaine et des attentes des superviseurs européens et nationaux.

Développement

- Objectif et périmètre
- Données et définition de la cible
- Choix justifiés, références comparées
- Documentation complète

Validation indépendante

- Revue conceptuelle
- Réplication des résultats
- Tests de robustesse et de stabilité
- Analyse d'équité et d'explicabilité

Utilisation et suivi

- Inventaire des modèles
- Indicateurs de performance et de stabilité
- Seuils d'alerte, revue annuelle
- Plan de remplacement

REPÈRE 2026

Pour le scoring de crédit des particuliers, l'AI Act européen ajoute des exigences propres aux systèmes à haut risque — gestion des risques, gouvernance des données, documentation technique, journalisation, supervision humaine, exactitude et robustesse — applicables à partir du **2 décembre 2027** après le report décidé en 2026.

SECTION 7.4

Déploier

LISTING 21 · EXPOSER UN MODÈLE DE SCORING VIA UNE API PYTHON

```python
import joblib
from fastapi import FastAPI
from pydantic import BaseModel
import pandas as pd

modele = joblib.load("pd_xgb_v1.joblib")            # modèle entraîné et versionné
app = FastAPI(title="API de scoring PD")

class Dossier(BaseModel):
    revenu: float; taux_endettement: float; anciennete_emploi: float; age: float
    nb_incidents_12m: int; montant: float; duree: int; revenu_manquant: int

@app.post("/score")
def score(d: Dossier):
    X = pd.DataFrame([d.model_dump()])
    pd_ = float(modele.predict_proba(X)[0, 1])
    return {"pd": round(pd_, 4), "version": "pd_xgb_v1"}
```

Bonnes pratiques MLOps

- Versionner code, données et modèles (Git, MLflow)
- Même code de préparation à l'entraînement et en production
- Tests automatiques avant chaque mise en service
- Journaliser chaque prédiction et sa version de modèle

Supervision humaine

- Cas limites orientés vers un analyste
- Possibilité de déroger au modèle, avec justification
- Motifs de décision communiqués au client
- Revue périodique des dérogations

SECTION 7.5

Surveiller la dérive

Les données changent : crise, nouvelle offre, nouvelle clientèle, nouveaux modes de fraude. L'**indice de stabilité de population** (PSI) compare la distribution actuelle d'un score ou d'une variable à celle de l'entraînement.

$$PSI = \Sigma \left(\%actuel_i - \%référence_i \right) \times \ln\left(\%actuel_i / \%référence_i \right)$$

< 0,10 : stable · 0,10 – 0,25 : à surveiller · > 0,25 : dérive importante

LISTING 22 · CALCULER LE PSI D'UN SCORE PYTHON

```python
import numpy as np

def psi(reference, actuel, n_classes=10):
    bornes = np.quantile(reference, np.linspace(0, 1, n_classes + 1))
    bornes[0], bornes[-1] = -np.inf, np.inf
    ref = np.histogram(reference, bornes)[0] / len(reference)
    act = np.histogram(actuel, bornes)[0] / len(actuel)
    ref, act = np.clip(ref, 1e-6, None), np.clip(act, 1e-6, None)
    return float(np.sum((act - ref) * np.log(act / ref)))

print("PSI du score :", round(psi(scores_entrainement, scores_mois_courant), 3))
```

SECTION 7.6

IA générative et finance

Les grands modèles de langage ouvrent de nouveaux usages : synthèse de rapports annuels et de notes de crédit, extraction d'informations dans des contrats, analyse du sentiment des communications de banques centrales, assistance à la conformité et au code. Ils ne remplacent pas les modèles quantitatifs décrits dans ce guide, mais les alimentent en variables textuelles et accélèrent le travail des analystes.

> **ATTENTION**
> Leurs réponses peuvent être inexactes ou inventées, et les données confidentielles des clients ne doivent jamais être envoyées vers un service non autorisé. Toute décision qui engage un client reste soumise aux exigences de validation, d'explicabilité et de supervision humaine présentées dans ce chapitre.

ANNEXE A

Aide-mémoire des *métriques*

Quelle métrique pour quel problème : la référence à garder sous la main.

PROBLÈME	MÉTRIQUE PRINCIPALE	MÉTRIQUES COMPLÉMENTAIRES	À ÉVITER
Scoring de crédit (PD)	AUC / Gini	KS, Brier, calibration par classe, PSI	Exactitude seule
Détection de fraude	PR-AUC (average precision)	Rappel à précision fixée, coût total, alertes par jour	AUC ROC seule, exactitude
Prévision de niveau	MAE, RMSE	Comparaison à la marche aléatoire, test de Diebold-Mariano	R^2 en échantillon
Prévision de volatilité	QLIKE, MSE sur la variance	Couverture des intervalles, backtests de VaR	MAPE
LGD (régression bornée)	MAE, RMSE	Corrélation de rang, calibration par classe	—
Segmentation	Silhouette, stabilité	Interprétabilité métier des segments	Nombre de segments arbitraire

Classification : les formules

MÉTRIQUE	FORMULE
Précision	VP / (VP + FP)
Rappel (sensibilité)	VP / (VP + FN)
Spécificité	VN / (VN + FP)
F1	2PR / (P + R)
Gini	2 × AUC − 1
Brier	moyenne de $(p - y)^2$

Régression et prévision

MÉTRIQUE	FORMULE
MAE	moyenne de $\lvert y - \hat{y} \rvert$
RMSE	racine de la moyenne de $(y - \hat{y})^2$
MAPE	moyenne de $\lvert y - \hat{y} \rvert / \lvert y \rvert$
Theil U	RMSE modèle / RMSE naïf
PSI	$\Sigma\ (a - r)\ \ln(a / r)$
EL	PD × LGD × EAD

EN PRATIQUE

Rapportez toujours vos métriques **hors échantillon**, sur une période postérieure à l'entraînement, avec l'intervalle de confiance ou la variabilité entre plis de validation.

ANNEXE B

Jeux de données pour *s'exercer*

Des données publiques pour reproduire chaque chapitre de ce guide.

JEU DE DONNÉES	CONTENU	CHAPITRE
Credit Card Fraud Detection (ULB)	284 807 transactions par carte, dont 492 fraudes ; variables anonymisées par ACP	6
Give Me Some Credit	Données de crédit à la consommation, défaut à deux ans	5
Home Credit Default Risk	Dossiers de crédit avec historiques et bureaux de crédit, plusieurs tables	2 et 5
German Credit (UCI)	1 000 dossiers, 20 variables : idéal pour la scorecard WoE	5
Lending Club	Prêts entre particuliers, montants, taux, statut de remboursement	5
Cours boursiers (Yahoo Finance, Stooq)	Prix et volumes journaliers d'actions et d'indices	4
FRED (Réserve fédérale de Saint-Louis)	Séries macroéconomiques et financières longues	4
Bank Al-Maghrib, HCP	Statistiques monétaires, bancaires et macroéconomiques marocaines	4

Tableau B.1 — Vérifiez les conditions d'utilisation de chaque source avant usage

ANNEXE C

Feuille de route : *8 semaines*

S1–2 Bases

pandas, visualisation, statistiques descriptives ; chapitres 1 et 2 sur German Credit.

S3–4 Modèles

Logit, forêt, XGBoost, validation croisée ; chapitre 3 et scorecard du chapitre 5.

S5–6 Cas d'usage

Prévision de volatilité (chapitre 4) et fraude ULB (chapitre 6) en projet complet.

S7–8 Portfolio

SHAP, API, documentation ; publier un projet commenté sur GitHub et LinkedIn.

ANNEXE D

Références

Les travaux fondateurs cités dans ce guide (style APA 7).

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*(3), 307–327.

Box, G. E. P., & Jenkins, G. M. (1970). *Time series analysis: Forecasting and control*. Holden-Day.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence* (pp. 159–166).

Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics, 13*(3), 253–263.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies, 33*(5), 2223–2273.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2e éd.). Springer.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Ke, G., Meng, Q., Finley, T., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems, 30*.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422).

López de Prado, M. (2018). *Advances in financial machine learning*. Wiley.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems, 30*.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144).

Siddiqi, N. (2017). *Intelligent credit scoring: Building and implementing better credit risk scorecards* (2e éd.). Wiley.

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician, 72*(1), 37–45.

CONCLUSION

La donnée propose, *l'analyste* dispose.

Le machine learning a profondément transformé la mesure des risques et la prévision en finance. Mais ce guide l'a montré à chaque chapitre : la performance d'un modèle se joue d'abord dans la qualité des données, la rigueur de la validation et la clarté de l'explication. Les algorithmes changeront ; ces exigences resteront.

1 Le temps d'abord

Toujours découper, valider et construire les variables dans le respect de la chronologie.

2 Simple avant complexe

Un modèle complexe doit battre une référence simple pour justifier son opacité.

3 Expliquer pour décider

Un modèle que personne ne comprend ne sera ni accepté, ni corrigé, ni utilisé.

> *« Tous les modèles sont faux, mais certains sont utiles. »*
>
> GEORGE E. P. BOX

NOTE DE L'AUTEUR

Les bibliothèques, leurs syntaxes et le cadre réglementaire évoluent : ce guide reflète leur état fin 2026. Les exemples de code sont pédagogiques et doivent être adaptés, testés et validés avant tout usage professionnel. Les chiffres présentés dans les figures sont illustratifs.

MERCI POUR VOTRE LECTURE

Ce guide vous a été *utile* ?

Aidez-le à atteindre les analystes, étudiants et chercheurs qui veulent mettre la donnée au service de la décision financière.

01
Réagissez
pour que le guide atteigne plus de personnes de votre réseau.

02
Commentez
votre cas d'usage préféré : crédit, fraude ou prévision ?

03
Partagez
avec un collègue analyste, risk manager ou data scientist.

04
Suivez-moi
pour les prochains guides de la série Data & Finance.



EL KHLIFI Imad
Doctorant en Économie & Gestion · FSJES Tanger · Université Abdelmalek Essaâdi