Cours Machine Learning

Chapitre 02

Métriques d'Évaluation

Objectifs d'apprentissage :

— Maîtriser les métriques de classification (matrice de confusion, accuracy, precision, recall, F1-score, ROC, AUC)
— Comprendre les métriques de régression (MSE, RMSE, MAE, Rš, MAPE)
— Savoir choisir la bonne métrique selon le problème
— Maîtriser les techniques de validation (train/test split, K-fold, stratified K-fold)
— Identifier les pièges courants et éviter l'overfitting

Prérequis : Chapitre 01 - Fondamentaux Mathématiques

Durée estimée : 6-8 heures

Notebooks : $02_demo_*.ipynb$

Cours ML - Sandbox-ML
Version 1.0 - 2026

Table des matières

1 Motivation

L'évaluation des modèles de Machine Learning est une étape cruciale qui détermine si un modèle est prêt pour la production. Contrairement à la programmation traditionnelle où on peut vérifier la correction d'un programme par des tests unitaires, en ML, un modèle n'est jamais parfait à 100%.

Exemple : Le dilemme du diagnostic médical

Imaginez un système de détection automatique du cancer à partir de radiographies :
Scénario A : Le modèle détecte 95% des cancers (excellent !) mais donne 40% de faux positifs (mauvais !).

- **Conséquence :** 40% des patients sains passent des examens invasifs inutiles
- **Coût :** Anxiété des patients + surcharge du système de santé

Scénario B : Le modèle ne donne que 2% de faux positifs (excellent !) mais ne détecte que 60% des cancers (catastrophique !).

- **Conséquence :** 40% des malades ne sont pas diagnostiqués
- **Coût :** Vies humaines

Question clé : Quelle métrique unique utiliser pour comparer ces deux modèles ? L'accuracy ? La précision ? Le recall ? Le F1-score ?
Réponse : Cela dépend du contexte médical et des coûts associés. Ce chapitre vous apprendra à faire ce choix de manière éclairée.

Attention

Une métrique unique ne suffit jamais ! Un modèle avec 99% d'accuracy peut être complètement inutile si :

- Les classes sont déséquilibrées (ex : 1% de fraudes bancaires)
- Les coûts d'erreur sont asymétriques (faux négatif en médecine » faux positif)
- Le modèle ne généralise pas (overfitting)

2 Métriques de Classification

2.1 Matrice de Confusion

Définition : Matrice de Confusion

La **matrice de confusion** est un tableau qui décrit les performances d'un modèle de classification en comparant les prédictions aux vraies valeurs. Pour un problème de classification binaire, elle contient 4 valeurs :

TABLE 1 – Matrice de Confusion (Classification Binaire)

		Prédiction	
		Positive	Negative
Réalité	Positive	TP (Vrai Positif)	FN (Faux Négatif)
	Negative	FP (Faux Positif)	TN (Vrai Négatif)

— **TP (True Positive)** : Cas positifs correctement classés comme positifs
— **TN (True Negative)** : Cas négatifs correctement classés comme négatifs
— **FP (False Positive)** : Cas négatifs incorrectement classés comme positifs (Erreur de Type I)
— **FN (False Negative)** : Cas positifs incorrectement classés comme négatifs (Erreur de Type II)

Exemple : Détection de spam

Sur 1000 emails analysés :
— 150 vrais spams, 850 emails légitimes
— TP = 130 : spams détectés correctement
— FN = 20 : spams non détectés (passent en boîte de réception)
— FP = 40 : emails légitimes classés comme spam (perdus !)
— TN = 810 : emails légitimes correctement classés

2.2 Accuracy (Exactitude)

Définition : Accuracy

L'**accuracy** (exactitude) est la proportion de prédictions correctes sur l'ensemble total :

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{Correct}}{\text{Total}} \tag{1}$$

Valeur : entre 0 et 1 (ou 0% et 100%).

Pour l'exemple spam :

$$\text{Accuracy} = \frac{130 + 810}{1000} = \frac{940}{1000} = 0.94 = 94\% \tag{2}$$

Attention

L'accuracy peut être trompeuse avec des classes déséquilibrées !
Exemple : Détection de fraude bancaire (0.1% de fraudes)
— Sur 10 000 transactions : 10 fraudes, 9 990 légitimes
— Modèle stupide : prédire toujours "légitime"
— Accuracy = $\frac{9990}{10000} = 99.9\%$ (excellent en apparence !)
— Mais 0% de fraudes détectées (catastrophique !)
Conclusion : L'accuracy seule est insuffisante pour des classes déséquilibrées.

2.3 Precision (Précision)

Définition : Precision

La **precision** (précision) mesure la proportion de prédictions positives qui sont correctes :

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{Total Prédictions Positives}} \tag{3}$$

Interprétation : "Quand le modèle prédit 'positif', à quelle fréquence a-t-il raison ?"

Pour l'exemple spam :

$$\text{Precision} = \frac{130}{130 + 40} = \frac{130}{170} \approx 0.765 = 76.5\% \tag{4}$$

Cela signifie que 76.5% des emails classés comme spam sont réellement des spams. Les 23.5% restants sont des faux positifs (emails légitimes perdus).

Astuce

Utilisez la **precision** quand les **faux positifs sont coûteux** :

- Filtrage spam : éviter de perdre des emails importants
- Recommandation de produits : éviter de recommander des produits non pertinents
- Publicité ciblée : éviter de dépenser du budget sur de mauvaises cibles

2.4 Recall (Rappel / Sensibilité)

Définition : Recall (Sensibilité)

Le **recall** (rappel ou sensibilité) mesure la proportion de cas positifs réels qui sont détectés :

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{Total Cas Positifs Réels}} \tag{5}$$

Interprétation : "Parmi tous les cas positifs réels, combien le modèle en détecte-t-il ?"

Pour l'exemple spam :

$$\text{Recall} = \frac{130}{130 + 20} = \frac{130}{150} \approx 0.867 = 86.7\% \tag{6}$$

Cela signifie que 86.7% des spams sont détectés. Les 13.3% restants passent en boîte de réception (faux négatifs).

Astuce

Utilisez le **recall** quand les **faux négatifs sont coûteux** :

- Détection de cancer : manquer un cancer peut être fatal
- Détection de fraude : manquer une fraude peut coûter cher
- Détection d'intrusion réseau : manquer une attaque peut compromettre le système

2.5 Spécificité

Définition : Spécificité

La **spécificité** mesure la proportion de cas négatifs réels qui sont correctement classés :

$$\text{Spécificité} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{\text{TN}}{\text{Total Cas Négatifs Réels}} \tag{7}$$

Pour l'exemple spam :

$$\text{Spécificité} = \frac{810}{810 + 40} = \frac{810}{850} \approx 0.953 = 95.3\% \tag{8}$$

La spécificité est le complément du taux de faux positifs (FPR) :

$$\text{FPR} = 1 - \text{Spécificité} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{9}$$

2.6 F1-Score (Moyenne Harmonique)

Définition : F1-Score

Le **F1-score** est la moyenne harmonique entre la precision et le recall :

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{10}$$

Pour l'exemple spam :

$$F_1 = 2 \times \frac{0.765 \times 0.867}{0.765 + 0.867} = 2 \times \frac{0.663}{1.632} \approx 0.813 = 81.3\% \tag{11}$$

Pourquoi la moyenne harmonique ?

La moyenne harmonique pénalise les déséquilibres :

- Si Precision = 100% et Recall = 10% : $F_1 \approx 18\%$ (faible !)
- Si Precision = 90% et Recall = 90% : $F_1 = 90\%$ (bon équilibre)

En comparaison, la moyenne arithmétique donnerait :

- $(100\% + 10\%)/2 = 55\%$ (trop optimiste !)
- $(90\% + 90\%)/2 = 90\%$ (identique)

Astuce

Le F1-score est la métrique par défaut quand :

- Les classes sont déséquilibrées
- Precision et Recall sont tous deux importants
- Vous cherchez un bon compromis entre FP et FN

2.6.1 Variante : F-Beta Score

Pour donner plus de poids au recall ou à la precision, on utilise le **F-beta score** :

$$F_\beta = (1 + \beta^2) \times \frac{\text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}} \tag{12}$$

- $\beta = 1$: F1-score (poids égal)
- $\beta = 0.5$: F0.5-score (favorise la precision)
- $\beta = 2$: F2-score (favorise le recall)

2.7 Courbe ROC et AUC

2.7.1 Courbe ROC

Définition : Courbe ROC

La **courbe ROC** (Receiver Operating Characteristic) représente le compromis entre le taux de vrais positifs (TPR = Recall) et le taux de faux positifs (FPR) pour différents seuils de classification.

Axes :

- Axe X : FPR (False Positive Rate) = $\frac{\text{FP}}{\text{FP+TN}} = 1 - \text{Spécificité}$
- Axe Y : TPR (True Positive Rate) = $\frac{\text{TP}}{\text{TP+FN}} = \text{Recall}$

La plupart des classifieurs produisent des probabilités (ex : `predict_proba()` en scikit-learn). La classification finale dépend d'un seuil (généralement 0.5) :

- Si $P(\text{classe positive}) \geq 0.5$: prédire "positif"
- Si $P(\text{classe positive}) < 0.5$: prédire "négatif"

La courbe ROC trace TPR vs FPR pour tous les seuils possibles (0 à 1).

Interprétation visuelle :

- **Classifieur aléatoire** : ligne diagonale (TPR = FPR)
- **Classifieur parfait** : passe par le point (0, 1) (TPR = 100%, FPR = 0%)
- **Meilleur modèle** : courbe la plus proche du coin supérieur gauche

2.7.2 AUC (Area Under Curve)

Définition : AUC

L'**AUC** (Area Under the ROC Curve) mesure l'aire sous la courbe ROC. Elle représente la probabilité qu'un classifieur classe un exemple positif aléatoire plus haut qu'un exemple négatif aléatoire.

$$\text{AUC} \in [0, 1] \tag{13}$$

Interprétation :

- AUC = 0.5 : Classifieur aléatoire (inutile)
- AUC = 0.7 : Acceptable
- AUC = 0.8 : Bon
- AUC = 0.9 : Excellent
- AUC = 1.0 : Parfait (rare en pratique, souvent signe d'overfitting)

Astuce

L'AUC est une excellente métrique car :
- Insensible au seuil de classification
- Robuste aux classes déséquilibrées (dans une certaine mesure)
- Permet de comparer différents modèles indépendamment du seuil choisi

2.8 Courbe Precision-Recall

Définition : Courbe Precision-Recall

La **courbe Precision-Recall** trace la precision en fonction du recall pour différents seuils.

Axes :
- Axe X : Recall (TPR)
- Axe Y : Precision

Différence avec ROC :
- La courbe ROC utilise FPR (sensible aux TN)
- La courbe PR utilise Precision (ignore les TN)

Astuce

Préférez la courbe **Precision-Recall** plutôt que ROC quand :
- Les classes sont très déséquilibrées (ex : 1% de positifs)
- Vous vous souciez plus des positifs que des négatifs
- Exemples : détection de fraude, détection d'anomalies, recherche d'information

Raison : Avec 99% de négatifs, FPR reste proche de 0 même avec beaucoup de FP, ce qui rend la courbe ROC trop optimiste.

2.9 Métriques Multi-Classes

Pour les problèmes de classification à plus de 2 classes, on peut calculer les métriques de trois façons :

Définition : Moyennes pour Multi-Classes

- **Macro Average** : moyenne simple des métriques de chaque classe

$$\text{Precision}_{\text{macro}} = \frac{1}{C}\sum_{i=1}^{C}\text{Precision}_i \tag{14}$$

  Traite toutes les classes de manière égale (utile si toutes les classes sont aussi importantes).
- **Micro Average** : calcule les métriques globalement (agrège TP, FP, FN)

$$\text{Precision}_{\text{micro}} = \frac{\sum_{i=1}^{C}\text{TP}_i}{\sum_{i=1}^{C}(\text{TP}_i + \text{FP}_i)} \tag{15}$$

Favorise les classes majoritaires (utile si les classes déséquilibrées reflètent la réalité).

— **Weighted Average** : moyenne pondérée par le nombre d'exemples de chaque classe

$$\text{Precision}_{\text{weighted}} = \frac{1}{N}\sum_{i=1}^{C} n_i \times \text{Precision}_i \tag{16}$$

où n_i est le nombre d'exemples de la classe i et $N = \sum n_i$.

Exemple : Classification d'images (3 classes)

Analyse :

— Macro average = $(0.90 + 0.88 + 0.70)/3 = 0.83$ (classe "Oiseau" pèse autant que les autres)

— Weighted average plus élevée car elle donne plus de poids aux classes majoritaires (Chat, Chien)

3 Métriques de Régression

Pour les problèmes de régression, on cherche à mesurer l'écart entre les prédictions continues et les vraies valeurs.

Notations :

— y_i : vraie valeur pour l'exemple i

— $\hat{y}_i$: prédiction pour l'exemple i

— n : nombre d'exemples

— $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$: moyenne des vraies valeurs

3.1 MSE (Mean Squared Error)

Définition : MSE

Le **MSE** (Mean Squared Error) est la moyenne des carrés des erreurs :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{17}$$

Propriétés :

— Toujours ≥ 0 (0 = prédictions parfaites)

— Pénalise fortement les grandes erreurs (à cause du carré)

— Unité : carré de l'unité de y (difficile à interpréter)

— Très sensible aux outliers

Exemple : Prédiction de prix immobiliers

Prix réels : $[200, 250, 300, 350, 400]$ (en k)
Prédictions : $[210, 240, 310, 340, 500]$ (en k)

Erreurs : $[10, -10, 10, -10, 100]$

$$\text{MSE} = \frac{10^2 + 10^2 + 10^2 + 10^2 + 100^2}{5} = \frac{10300}{5} = 2060 \text{ (k)}^2 \tag{18}$$

L'erreur de 100 k domine complètement le MSE.

3.2 RMSE (Root Mean Squared Error)

Définition : RMSE

Le **RMSE** est la racine carrée du MSE :

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{19}$$

Avantage : Même unité que y (plus facile à interpréter).

Pour l'exemple précédent :

$$\text{RMSE} = \sqrt{2060} \approx 45.4 \text{ k} \tag{20}$$

Interprétation : en moyenne, les prédictions s'écartent de 45.4 k de la réalité.

3.3 MAE (Mean Absolute Error)

Définition : MAE

Le **MAE** (Mean Absolute Error) est la moyenne des valeurs absolues des erreurs :

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{21}$$

Propriétés :

- Toujours ≥ 0
- Moins sensible aux outliers que MSE/RMSE
- Même unité que y
- Pénalise toutes les erreurs de manière proportionnelle

Pour l'exemple précédent :

$$\text{MAE} = \frac{|10| + |10| + |10| + |10| + |100|}{5} = \frac{140}{5} = 28 \text{ k} \tag{22}$$

Comparaison MSE vs MAE :

- RMSE = 45.4 k (fortement influencé par l'outlier de 100 k)
- MAE = 28 k (moins influencé par l'outlier)

Astuce

Choisir entre RMSE et MAE :

- Utilisez **RMSE** si les grandes erreurs sont très pénalisantes (finance, sécurité)
- Utilisez **MAE** si toutes les erreurs ont un coût similaire (météo, estimation générale)
- En cas de doute, reportez les deux !

3.4 Rš (Coefficient de Détermination)

Définition : Rš

Le **Rš** (coefficient de détermination) mesure la proportion de variance expliquée par le modèle :

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{23}$$

où :

- $\text{SS}_{\text{res}} = \sum(y_i - \hat{y}_i)^2$: somme des carrés des résidus (variance non expliquée)
- $\text{SS}_{\text{tot}} = \sum(y_i - \bar{y})^2$: variance totale

Interprétation :

- $R^2 = 1$: modèle parfait (prédictions exactes)
- $R^2 = 0$: modèle équivalent à prédire la moyenne $\bar{y}$
- $R^2 < 0$: modèle pire que prédire la moyenne (très mauvais !)

Exemple :

- Si $R^2 = 0.85$: le modèle explique 85% de la variance des données
- Les 15% restants sont dus au bruit ou à des variables non prises en compte

Attention

Limites du Rš :

- Augmente automatiquement quand on ajoute des features (même inutiles)
- Ne détecte pas l'overfitting
- Solution : utiliser le **Rš ajusté** qui pénalise le nombre de features

$$R^2_{\text{ajusté}} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{24}$$

où p est le nombre de features.

3.5 MAPE (Mean Absolute Percentage Error)

Définition : MAPE

Le **MAPE** mesure l'erreur moyenne en pourcentage :

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{25}$$

Avantage : Indépendant de l'échelle (permet de comparer des modèles sur différents datasets).

Interprétation : MAPE = 10% signifie que les prédictions s'écartent en moyenne de 10% des vraies valeurs.

Attention

Problème du MAPE :
- Non défini si $y_i = 0$
- Asymétrique : pénalise plus les surestimations que les sous-estimations
- Exemple : erreur de +50% vs -50% ne donnent pas le même MAPE

3.6 Comparaison des Métriques de Régression

TABLE 3 – Comparaison des métriques de régression

Métrique	Sensibilité outliers	Interprétation	Unité
MSE	Très élevée	Difficile	$(y)^2$
RMSE	Élevée	Facile	y
MAE	Modérée	Facile	y
Rš	N/A	Facile	Sans unité
MAPE	Modérée	Facile	%

4 Techniques de Validation

L'évaluation d'un modèle ne peut pas se faire sur les données d'entraînement ! Un modèle peut mémoriser les données (overfitting) et obtenir 100% d'accuracy en entraînement mais échouer en production.

Attention

Règle d'or : Ne jamais évaluer un modèle sur les données utilisées pour l'entraîner !

4.1 Train/Test Split

Définition : Train/Test Split

La technique la plus simple : diviser aléatoirement le dataset en deux ensembles :
- **Training set** (70-80%) : pour entraîner le modèle
- **Test set** (20-30%) : pour évaluer le modèle

```python
from sklearn.model_selection import train_test_split

# Split 80/20
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,      # 20% pour le test
    random_state=42,    # reproductibilité
    stratify=y          # préserve distribution des classes
)

```

```
# Entraînement
model.fit(X_train, y_train)

# Évaluation sur données JAMAIS vues
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Listing 1 – Train/Test Split avec scikit-learn

Avantages :

- Simple et rapide
- Suffisant pour les grands datasets

Limites :

- Résultat dépend du split aléatoire (variance élevée)
- Perte de données (20-30% non utilisés pour l'entraînement)
- Risque de split non représentatif sur petits datasets

4.2 Validation Set (Train/Validation/Test)

Pour optimiser les hyperparamètres, il faut un 3ème ensemble :

Définition : Train/Validation/Test Split

- **Training set** (60%) : entraînement du modèle
- **Validation set** (20%) : tuning des hyperparamètres
- **Test set** (20%) : évaluation finale (jamais touché avant)

Workflow :

1. Entraîner plusieurs modèles avec différents hyperparamètres sur le training set
2. Évaluer chaque modèle sur le validation set
3. Choisir le meilleur modèle
4. Évaluation finale sur le test set (une seule fois !)

Attention

Si vous évaluez plusieurs fois sur le test set et ajustez le modèle en conséquence, le test set devient un validation set déguisé ! Vous risquez l'overfitting sur le test set.

4.3 K-Fold Cross-Validation

Définition : K-Fold Cross-Validation

La **validation croisée K-fold** divise le dataset en K sous-ensembles (folds) de taille égale. Le modèle est entraîné et évalué K fois, chaque fold servant une fois de test set.

Algorithme :

Algorithm 1 K-Fold Cross-Validation

Require : Dataset D, Modèle M, Nombre de folds K

Ensure : Score moyen et variance

Diviser D en K folds : $D_1, D_2, \ldots, D_K$
Initialiser liste de scores : scores = []
for $i = 1$ **to** K **do**
    test_set = D_i
    train_set = $D \setminus D_i$ (tous les folds sauf D_i)
    Entraîner M sur train_set
    score$_i$ = évaluer M sur test_set
    Ajouter score$_i$ à scores
end for
return mean(scores), std(scores)

```python
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

# 5-fold cross-validation
scores = cross_val_score(
    model, X, y,
    cv=5,                    # 5 folds
    scoring='accuracy'       # métrique
)

print(f"Scores: {scores}")
print(f"Moyenne: {scores.mean():.3f} (+/- {scores.std():.3f})")

# Exemple de sortie :
# Scores: [0.82, 0.85, 0.81, 0.84, 0.83]
# Moyenne: 0.830 (+/- 0.015)
```

Listing 2 – K-Fold Cross-Validation avec scikit-learn

Avantages :

- Utilise toutes les données pour l'entraînement et le test
- Réduit la variance de l'estimation (score moyen sur K runs)
- Fournit un intervalle de confiance (écart-type)

Limites :

- Coût computationnel : K fois plus long qu'un simple train/test
- Valeur typique : $K = 5$ ou $K = 10$

4.4 Stratified K-Fold

Définition : Stratified K-Fold

Le **Stratified K-Fold** est une variante de K-Fold qui préserve la proportion de chaque classe dans chaque fold.

Pourquoi c'est important :

Exemple : Dataset de 100 exemples (90 classe A, 10 classe B)

K-Fold classique :

- Un fold pourrait contenir 0 exemple de classe B (problème !)
- Entraînement sur données non représentatives

Stratified K-Fold :

- Chaque fold contient environ 90% classe A et 10% classe B
- Distribution préservée

```python
from sklearn.model_selection import StratifiedKFold, cross_val_score

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scores = cross_val_score(
    model, X, y,
    cv=skf,                    # utiliser stratified K-fold
    scoring='f1_weighted'
)
```

Listing 3 – Stratified K-Fold

Astuce

Utilisez **Stratified K-Fold** par défaut pour la classification, surtout si :

- Les classes sont déséquilibrées
- Le dataset est petit

Pour la régression, utilisez K-Fold classique.

4.5 Leave-One-Out Cross-Validation (LOOCV)

Définition : LOOCV

Le **LOOCV** est un cas extrême de K-Fold où $K = n$ (nombre d'exemples). Chaque exemple sert une fois de test set.

Avantages :

- Estimation quasi non biaisée
- Utile pour très petits datasets (< 100 exemples)

Limites :

- Extrêmement coûteux : n entraînements complets
- Variance élevée de l'estimation
- Rarement utilisé en pratique (préférer 5 ou 10-fold)

4.6 Time Series Split

Définition : Time Series Split

Pour les séries temporelles, il faut respecter l'ordre chronologique. Le **Time Series Split** crée des folds où le training set contient toujours des données antérieures au test set.

Exemple avec 5 splits :

```
Fold 1: train [1:100], test [101:120]
Fold 2: train [1:120], test [121:140]
Fold 3: train [1:140], test [141:160]
Fold 4: train [1:160], test [161:180]
Fold 5: train [1:180], test [181:200]
```

```python
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)

for train_index, test_index in tscv.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    print(f"Score: {score:.3f}")
```

Listing 4 – Time Series Split

Attention

Pour les séries temporelles, **JAMAIS de shuffle** ni de K-Fold classique ! Utiliser uniquement Time Series Split ou un simple train/test split chronologique.

4.7 Comparaison des Techniques de Validation

Table 4 – Comparaison des techniques de validation

Technique	Données utilisées	Variance	Coût	Cas d'usage
Train/Test	80%	Élevée	Faible	Grands datasets, première itération
Train/Val/Test	60%	Élevée	Faible	Tuning hyperparamètres
K-Fold (K=5)	100%	Modérée	Moyen	Usage général, petits/moyens datasets
K-Fold (K=10)	100%	Faible	Élevé	Datasets moyens, besoin précision
LOOCV	100%	Très faible	Très élevé	Très petits datasets (< 100)
Stratified K-Fold	100%	Modérée	Moyen	Classes déséquilibrées
Time Series Split	100%	Modérée	Moyen	Séries temporelles uniquement

5 Pièges Courants et Bonnes Pratiques

5.1 Data Leakage (Fuite de données)

Attention

Le **data leakage** est la principale cause de modèles qui performent bien en développement mais échouent en production.

Définition : Data Leakage

Il y a **data leakage** quand des informations du test set influencent l'entraînement du modèle. Cela conduit à des performances surestimées.

Types de leakage :

5.1.1 Leakage via le preprocessing

MAUVAIS :

```
from sklearn.preprocessing import StandardScaler

# ERREUR: normalisation AVANT le split
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)  # utilise mean/std de TOUT le
    dataset

# Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y)

# Le test set a influencé la normalisation !
```

Listing 5 – Leakage via normalisation (INCORRECT)

BON :

```
# Split AVANT preprocessing
X_train, X_test, y_train, y_test = train_test_split(X, y)

# Normalisation basée UNIQUEMENT sur train set
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)  # fit sur train
X_test_scaled = scaler.transform(X_test)        # transform sur test (
    sans fit!)
```

Listing 6 – Normalisation correcte (CORRECT)

Pourquoi c'est important :

- Le scaler calculé sur tout le dataset "voit" les statistiques du test set
- En production, vous n'aurez pas accès aux données futures
- Impact : performances surestimées de 5-10% ou plus

5.1.2 Leakage via les features

Exemple : Prédiction de fraude bancaire (leakage)

Feature problématique : `fraud_score` (score de fraude calculé manuellement après investigation)
Problème : Ce score n'existe qu'après l'enquête, donc inutilisable en production (au moment de la transaction).
Symptôme : Modèle avec 99% d'accuracy en validation, mais 0% en production.

5.1.3 Leakage temporel

Exemple : Prédiction de prix d'actions

Erreur : Utiliser K-Fold classique sur des données temporelles.
Problème : Le modèle s'entraîne sur des données futures pour prédire le passé (impossible en production).
Solution : Utiliser Time Series Split.

5.2 Classes Déséquilibrées

Attention

Avec des classes très déséquilibrées (99% / 1%), l'accuracy devient inutile.

Stratégies :

1. **Changer la métrique :**
   — Utiliser Precision, Recall, F1-score, AUC
   — Ignorer l'accuracy
2. **Rééchantillonnage :**
   — **Oversampling :** dupliquer la classe minoritaire (risque d'overfitting)
   — **Undersampling :** réduire la classe majoritaire (perte de données)
   — **SMOTE :** générer des exemples synthétiques (meilleure approche)
3. **Class weights :**

```
from sklearn.linear_model import LogisticRegression

# Pénaliser plus les erreurs sur la classe minoritaire
model = LogisticRegression(class_weight='balanced')

```

4. **Algorithmes robustes :**
   — Random Forest, XGBoost (gèrent bien le déséquilibre)

5.3 Overfitting et Underfitting

Définition : Overfitting et Underfitting

— **Overfitting :** Le modèle mémorise les données d'entraînement (bruit inclus). Performance excellente en train, mauvaise en test.
— **Underfitting :** Le modèle est trop simple pour capturer les patterns. Performance médiocre en train ET en test.

Détection :

Situation	Train Score	Test Score	Diagnostic
Bon modèle	85%	83%	Équilibre
Overfitting	99%	70%	Trop complexe
Underfitting	65%	63%	Trop simple

Solutions :

Contre l'overfitting :
— Augmenter les données d'entraînement
— Régularisation (L1, L2, Dropout)
— Réduire la complexité du modèle
— Early stopping
— Data augmentation

Contre l'underfitting :
— Augmenter la complexité du modèle
— Ajouter des features
— Réduire la régularisation
— Entraîner plus longtemps

5.4 Sélection de Métriques : Guide Pratique

Table 5 – Quelle métrique utiliser ?

Contexte	Métrique recommandée	Raison
Classification équilibrée	Accuracy, F1-score	Classes également importantes
Classification déséquilibrée	Precision, Recall, F1, AUC	Accuracy trompeuse
Spam detection	Precision (+ Recall)	Éviter de perdre emails légitimes
Détection cancer	Recall (+ Precision)	Ne pas manquer de malades
Détection fraude	AUC, F1-score	Compromis FP/FN + robustesse seuil
Multi-classes équilibrées	Macro average	Toutes classes égales
Multi-classes déséquilibrées	Weighted average	Classes réalistes
Régression générale	RMSE, MAE, Rš	Standard, facile à interpréter
Régression avec outliers	MAE	Moins sensible aux outliers
Régression sans outliers	RMSE	Pénalise les grandes erreurs
Comparaison multi-datasets	MAPE, Rš	Indépendant de l'échelle
Séries temporelles	Time Series Split + RMSE/MAE	Respect de la temporalité

6 Résumé du Chapitre

6.1 Points Clés

- **Métriques de classification :**
  - Matrice de confusion : TP, TN, FP, FN
  - Accuracy : proportion de prédictions correctes (attention aux classes déséquilibrées)
  - Precision : "Quand je prédis positif, ai-je raison ?" (important si FP coûteux)
  - Recall : "Parmi les vrais positifs, combien sont détectés ?" (important si FN coûteux)
  - F1-score : compromis entre Precision et Recall
  - AUC-ROC : performance indépendante du seuil (bon pour comparer des modèles)
  - Precision-Recall curve : meilleure que ROC pour classes déséquilibrées
- **Métriques de régression :**
  - MSE : pénalise fortement les grandes erreurs (sensible aux outliers)
  - RMSE : version interprétable du MSE (même unité que y)
  - MAE : robuste aux outliers, pénalise toutes les erreurs également
  - Rš : proportion de variance expliquée (0 = modèle inutile, 1 = parfait)
  - MAPE : erreur en pourcentage (indépendant de l'échelle, mais problèmes si $y = 0$)
- **Validation :**
  - Ne JAMAIS évaluer sur les données d'entraînement
  - Train/Test split : simple, mais variance élevée
  - K-Fold CV : utilise toutes les données, réduit la variance
  - Stratified K-Fold : préserve la distribution des classes (recommandé)
  - Time Series Split : obligatoire pour les séries temporelles
- **Pièges à éviter :**
  - Data leakage (preprocessing APRÈS split)
  - Choisir la mauvaise métrique (accuracy avec classes déséquilibrées)
  - Overfitting (train score » test score)
  - Évaluer plusieurs fois sur le test set

6.2 Formules Essentielles

Formules à retenir

Classification :

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{26}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{27}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{28}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{29}$$

Régression :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{30}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{31}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{32}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{33}$$

7 Exercices

7.1 Questions de compréhension

1. Expliquez pourquoi l'accuracy peut être trompeuse avec des classes déséquilibrées (99% / 1%). Donnez un exemple concret.
2. Un modèle de détection de cancer a Precision = 80% et Recall = 95%. Que signifient ces chiffres concrètement ? Quelle métrique est la plus importante dans ce contexte ?
3. Pourquoi utilise-t-on la moyenne harmonique (F1) plutôt que la moyenne arithmétique entre Precision et Recall ?
4. Vous entraînez un modèle et obtenez : Train accuracy = 98%, Test accuracy = 72%. Quel est le problème ? Quelles solutions proposez-vous ?
5. Expliquez la différence entre RMSE et MAE. Dans quel contexte préférer l'un à l'autre ?
6. Pourquoi ne peut-on pas utiliser K-Fold classique pour des séries temporelles ? Quelle technique utiliser ?
7. Qu'est-ce que le data leakage ? Donnez un exemple de leakage via le preprocessing.
8. Un dataset contient 1% de fraudes. Vous voulez optimiser un seuil de classification. Quelle courbe utiliser : ROC ou Precision-Recall ? Pourquoi ?

7.2 Exercices pratiques

1. **Calcul manuel de métriques :**
   — Soit une matrice de confusion : TP = 45, FP = 10, FN = 5, TN = 140

- Calculez : Accuracy, Precision, Recall, F1-score, Spécificité
- Interprétez les résultats

2. **Comparaison de modèles (régression) :**
   - Modèle A : MSE = 25, MAE = 4, Rš = 0.85
   - Modèle B : MSE = 30, MAE = 3.5, Rš = 0.82
   - Quel modèle choisir ? Justifiez selon le contexte.
3. **Implémentation from scratch :**
   - Implémenter Precision, Recall, F1-score en NumPy pur (sans scikit-learn)
   - Tester sur des données synthétiques
   - Comparer avec `sklearn.metrics`
4. **Validation croisée :**
   - Dataset : Breast Cancer (sklearn)
   - Comparer : simple train/test, 5-fold CV, 10-fold CV, Stratified 5-fold
   - Analyser moyenne et écart-type des scores
5. **Classes déséquilibrées :**
   - Créer un dataset déséquilibré (95% / 5%)
   - Entraîner un modèle baseline
   - Tester : SMOTE, class weights, undersampling
   - Comparer les métriques (Accuracy, F1, AUC)
6. **Visualisations :**
   - Tracer une matrice de confusion avec seaborn
   - Tracer courbes ROC et Precision-Recall
   - Comparer 3 modèles sur le même graphique

8 Pour Aller Plus Loin

8.1 Lectures Recommandées

- **Articles :**
  - "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets" (Saito & Rehmsmeier, 2015)
  - "A Survey on Deep Learning for Imbalanced Classification" (Zhang et al., 2023)
- **Livres :**
  - "Hands-On Machine Learning" (Aurélien Géron) - Chapitre 3
  - "The Elements of Statistical Learning" (Hastie et al.) - Chapitre 7
- **Blogs :**
  - https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-pyt
  - https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

8.2 Ressources en Ligne

- Documentation scikit-learn : https://scikit-learn.org/stable/modules/model_evaluation.html
- Imbalanced-learn (SMOTE) : https://imbalanced-learn.org/
- Interactive ROC/PR curves : https://roc.mlvis.com/

8.3 Notebooks Associés

1. $02_demo_metriques\_classification.ipynb : Calculmanuel, visualisationsROC/PR$

1. $02_demo_metriques_regression.ipynb : ComparaisonMSE/MAE/R$š

8.4 Prochaines Étapes

Chapitre suivant recommandé : **Chapitre 03 - Régression Linéaire**

Vous savez maintenant évaluer un modèle. Le chapitre suivant vous apprendra à construire votre premier modèle de régression, en appliquant toutes les métriques et techniques de validation vues ici.

Références

1. Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.
2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.
3. Saito, T., & Rehmsmeier, M. (2015). "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". *PLOS ONE*, 10(3).
4. Pedregosa, F. et al. (2011). "Scikit-learn : Machine Learning in Python". *Journal of Machine Learning Research*, 12, 2825-2830.
5. Provost, F., & Fawcett, T. (2013). *Data Science for Business*. O'Reilly Media.