

# Introduction au Machine Learning et Préparation des Données

Concepts fondamentaux et bonnes pratiques

**Joseph Azar**

Maître de conférences  
Université Marie Louis Pasteur  
Chercheur à Femto-ST  
[joseph.azar@univ-fcomte.fr](mailto:joseph.azar@univ-fcomte.fr)

# Plan de la présentation

---

## Partie 1: Fondamentaux

1. Qu'est-ce que le Machine Learning ?
2. Types d'apprentissage
3. Variables et types de données
4. Le processus ML complet

## Partie 2: Préparation des données

5. Feature Engineering
6. Encodage et transformation
7. Normalisation et standardisation
8. Train-Test Split
9. Validation et métriques

# Qu'est-ce que le Machine Learning ?

---

**Définition :** Le Machine Learning est le processus de résolution d'un problème pratique en :

1. Collectant un dataset
2. Construisant algorithmiquement un modèle statistique basé sur ce dataset

**Exemple concret :** Prédire si un email est un spam

- Dataset : 10,000 emails étiquetés (spam/non-spam)
- Modèle : Algorithme qui apprend les patterns des spams
- Résultat : Prédiction automatique pour de nouveaux emails

# Programmation Traditionnelle vs Machine Learning



## Programmation Traditionnelle

Règles

+

Données



Réponses

On code explicitement les règles



## Machine Learning

Données

+

Réponses



Règles

La machine apprend les règles

# Les Types d'Apprentissage

---

## Supervisé

Données étiquetées  $\{(x, y)\}$

Ex: Email  $\rightarrow$  Spam/Non-spam

## Non-supervisé

Données non étiquetées  $\{x\}$

Ex: Grouper des clients similaires

## Semi-supervisé

Mix de données étiquetées et non-étiquetées

Ex: Peu d'exemples annotés

## Par renforcement

Apprentissage par récompenses

Ex: Jeux, robotique

# Apprentissage Supervisé en Détail

Dataset = Collection d'exemples étiquetés  $\{(x_i, y_i)\}_{i=1}^N$

## Classification

- Label  $y \in \{\text{classe}_1, \text{classe}_2, \dots\}$
- Prédire une catégorie

### **Exemple :** Diagnostic médical

- $x = [\text{âge}, \text{poids}, \text{tension}, \dots]$
- $y = \{\text{sain}, \text{malade}\}$

## Régression

- Label  $y \in \mathbb{R}$  (nombre réel)
- Prédire une valeur continue

### **Exemple :** Prix immobilier

- $x = [\text{surface}, \text{quartier}, \text{année}, \dots]$
- $y = 250,000\text{€}$

# Le Vecteur de Features (Caractéristiques)

Un **feature vector**  $x = [x^{(1)}, x^{(2)}, \dots, x^{(D)}]$  est une représentation numérique d'un exemple

**Exemple : Représenter une personne pour prédire un risque de diabète**

Feature	Description	Valeur	Type
$x^{(1)}$	Âge (années)	45	Numérique
$x^{(2)}$	IMC (kg/m <sup>2</sup> )	28.3	Numérique
$x^{(3)}$	Sexe	1 (H) ou 0 (F)	Binaire
$x^{(4)}$	Fumeur	1 (oui) ou 0 (non)	Binaire
$x^{(5)}$	Glycémie (mg/dL)	126	Numérique

Vecteur final :  $x = [45, 28.3, 1, 1, 126]$

# Variables Indépendantes vs Dépendantes

## Variables Indépendantes (X)

### Autres noms :

- Features / Caractéristiques
- Prédicteurs
- Variables explicatives
- Entrées

**Rôle :** Ce qu'on utilise pour prédire

## Variable Dépendante (Y)

### Autres noms :

- Label / Étiquette
- Target / Cible
- Variable réponse
- Sortie

**Rôle :** Ce qu'on veut prédire

### Exemple concret : Prédire le salaire d'un employé

- **X** = [années\_expérience, niveau\_études, secteur, ville] → Variables indépendantes
- **Y** = salaire\_annuel → Variable dépendante



# Types de Données en ML

---



## Données Numériques

### Continues :

- Température : 23.5°C
- Salaire : 45,750.50€
- Distance : 12.7 km

### Discrètes :

- Nombre d'enfants : 2
- Âge en années : 35
- Nombre de chambres : 4



## Données Catégorielles

### Nominales (sans ordre) :

- Couleur : {rouge, bleu, vert}
- Ville : {Paris, Lyon, Marseille}
- Type : {A, B, C}

### Ordinales (avec ordre) :

- Taille : {S, M, L, XL}
- Satisfaction : {faible, moyen, élevé}
- Note : {A, B, C, D, F}

# Pourquoi le Type de Données est Important ?

## ⚠ Attention !

Traiter incorrectement les types de données peut ruiner votre modèle

### Exemple problématique : Code postal

#### ❌ MAUVAISE APPROCHE

Traiter comme numérique :

- $75001 < 75002 < 75003$
- Le modèle pense qu'il y a un ordre
- Calcule des moyennes absurdes

#### ✅ BONNE APPROCHE

Traiter comme catégoriel :

- 75001, 75002, 75003 sont des catégories
- Pas d'ordre implicite
- Encodage approprié (one-hot)

# Le Processus ML Complet

---

- 1 Collecte des données** - Rassembler les données brutes
- 2 Préparation des données** - Nettoyer, transformer, encoder
- 3 Division des données** - Train, validation, test sets
- 4 Entraînement du modèle** - Apprentissage sur train set
- 5 Évaluation et optimisation** - Tester et améliorer

# Feature Engineering

**Définition :** L'art de transformer les données brutes en features informatives pour l'apprentissage

## Exemple : Prédire l'abandon de clients (churn)

### DONNÉES BRUTES (LOGS)

```
2024-01-15 09:30:45 UserID:123 Login
2024-01-15 09:35:12 UserID:123 PageView
2024-01-15 09:40:33 UserID:123 Purchase
2024-01-15 09:45:21 UserID:123 Logout
```

### FEATURES CRÉÉES

- Durée moyenne de session : 15 min
- Nombre de connexions/semaine : 12
- Montant total d'achats : 450€
- Jours depuis dernier achat : 7
- Taux de conversion : 0.23

# One-Hot Encoding

Transformer une variable catégorielle en plusieurs variables binaires

**Exemple : Encoder la couleur d'une voiture**

**AVANT (CATÉGORIEL)**

Voiture	Couleur
1	Rouge
2	Bleu
3	Vert
4	Rouge

**APRÈS (ONE-HOT)**

Voiture	Rouge	Bleu	Vert
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0

**Pourquoi ?** Évite que le modèle pense que Rouge=1, Bleu=2, Vert=3 implique un ordre

# Quand Utiliser One-Hot Encoding ?

---

## Utiliser One-Hot pour :

- **Catégories nominales**
  - Marque de voiture
  - Département
  - Type de produit
- **Peu de catégories** (< 10-20)
- **Modèles linéaires**

## Éviter One-Hot pour :

- **Catégories ordinales**
  - Taille : S, M, L, XL
  - Note : A, B, C, D
- **Beaucoup de catégories** (> 50)
- **Arbres de décision** (peuvent gérer directement)

# Label Encoding pour Variables Ordinales

Remplacer les catégories ordinales par des nombres qui respectent l'ordre

## Exemple : Niveau d'éducation

Niveau Original	Valeur Encodée
Primaire	1
Collège	2
Lycée	3
Licence	4
Master	5
Doctorat	6

✓ L'ordre est préservé : Primaire < Collège < Lycée < ...

# Binning (Discrétisation)

Transformer une variable continue en catégories (intervalles)

## Exemple : Grouper les âges

### DONNÉES CONTINUES

- Personne 1 : 23 ans
- Personne 2 : 67 ans
- Personne 3 : 45 ans
- Personne 4 : 18 ans
- Personne 5 : 35 ans

### APRÈS BINNING

- Personne 1 : Jeune (18-30)
- Personne 2 : Senior (60+)
- Personne 3 : Adulte (31-59)
- Personne 4 : Jeune (18-30)
- Personne 5 : Adulte (31-59)

**Avantage :** Capture des patterns non-linéaires



# Gestion des Valeurs Manquantes

⚠ Les valeurs manquantes peuvent casser votre modèle !



## Suppression

- **Ligne complète** : Si peu de lignes concernées
- **Colonne complète** : Si >50% manquant



## Imputation par moyenne/médiane

- **Moyenne** : Pour distributions normales
- **Médiane** : Pour données avec outliers



## Imputation par mode

- Pour variables catégorielles
- Remplacer par la valeur la plus fréquente



## Imputation avancée

- K-NN : Utiliser les voisins proches
- Régression : Prédire la valeur manquante

# Normalisation (Min-Max Scaling)

Transformer les valeurs dans l'intervalle [0, 1] ou [-1, 1]

$$\bar{x} = (x - \min) / (\max - \min)$$

## Exemple : Normaliser une feature

### Données salaire :

- Min : 20,000€
- Max : 120,000€
- Valeur : 45,000€

### Calcul :

$$\bar{x} = (45,000 - 20,000) / 100,000$$

$$\bar{x} = 0.25$$

# Normalisation avec Plusieurs Features

Problème : Échelles très différentes

Personne	Âge (années)	Salaire (€)
Alice	25	30,000
Bob	45	60,000
Charlie	35	100,000

## ✗ SANS NORMALISATION

Vecteur Alice : [25, 30000]  
Distance euclidienne dominée par le salaire!  
L'âge n'a quasi aucun impact

## ✓ AVEC NORMALISATION

Min-Max : [0, 1]  
Âge :  $(25-25)/(45-25) = 0/20 = 0$   
Salaire :  $(30-30)/(100-30) = 0/70 = 0$   
**Vecteur Alice : [0, 0]**

Résultat final (tous normalisés) :

Alice : Âge= $(25-25)/20=0$ , Sal= $(30-30)/70=0$  → [0, 0]  
Bob : Âge= $(45-25)/20=1$ , Sal= $(60-30)/70=0.43$  → [1, 0.43]  
Charlie : Âge= $(35-25)/20=0.5$ , Sal= $(100-30)/70=1$  → [0.5, 1]

# Rappel : Concepts Statistiques de Base

Données exemple : Notes d'étudiants

Notes : [12, 14, 15, 13, 16] sur 20

## Moyenne ( $\mu$ )

La valeur "typique" ou centre des données

$$\mu = (12 + 14 + 15 + 13 + 16) / 5$$

$$\mu = 70 / 5 = \mathbf{14}$$

## Variance ( $\sigma^2$ )

Mesure la dispersion des données

Écarts :  $(12-14)^2=4$ ,  $(14-14)^2=0$ ,  $(15-14)^2=1$ ,  $(13-14)^2=1$ ,  $(16-14)^2=4$

$$\sigma^2 = (4 + 0 + 1 + 1 + 4) / 5 = 10 / 5 = \mathbf{2}$$

## Écart-type ( $\sigma$ )

Racine carrée de la variance (même unité que les données)

$$\sigma = \sqrt{\text{variance}} = \sqrt{2} \approx \mathbf{1.41}$$

→ Les notes varient de  $\pm 1.41$  points autour de 14

**Interprétation :** 68% des valeurs sont à  $\pm 1\sigma$  de la moyenne

# Standardisation (Z-Score)

Transformer pour avoir moyenne = 0 et écart-type = 1

$$\hat{z} = (x - \mu) / \sigma$$

où  $\mu$  = moyenne,  $\sigma$  = écart-type

## Exemple : Standardiser la taille

### DONNÉES

- Moyenne ( $\mu$ ) : 170 cm
- Écart-type ( $\sigma$ ) : 10 cm
- Valeur : 185 cm

### CALCUL

$$\hat{z} = (185 - 170) / 10$$

$$\hat{z} = 15 / 10$$

$$\hat{z} = \mathbf{1.5}$$

→ 1.5 écart-types au-dessus de la moyenne

# Normalisation vs Standardisation : Quand utiliser ?

## Normalisation

### Utiliser quand :

- Données bornées connues
- Distribution uniforme
- Réseaux de neurones (activation sigmoid)
- Algorithmes basés sur distance (K-NN)

 Sensible aux outliers

## Standardisation

### Utiliser quand :

- Distribution normale/gaussienne
- Présence d'outliers
- SVM, régression logistique
- PCA, clustering

 Robuste aux outliers

# Impact de la Mise à l'Échelle

## Problème : Features à échelles différentes

Personne	Âge (années)	Salaire (€)	Distance domicile (km)
A	25	30,000	5
B	40	80,000	15

### Sans mise à l'échelle :

Le salaire (30,000 - 80,000) domine complètement l'âge (25 - 40) et la distance (5 - 15)

→ Le modèle ignore presque l'âge et la distance !

### Avec mise à l'échelle :

Toutes les features dans  $[0, 1]$  ou moyenne=0, std=1

# Détection et Gestion des Outliers

**Outlier** : Valeur anormalement éloignée des autres observations

## Exemple : Salaires dans une entreprise

Employés : 30k, 35k, 32k, 38k, 33k, **450k** (PDG)

### AVEC OUTLIER

- Moyenne : 103k €
- Médiane : 34k €

### SANS OUTLIER

- Moyenne : 33.6k €
- Médiane : 33k €

## Méthodes de détection

- **IQR (Interquartile Range)** : Outlier si  $x < Q1 - 1.5 \times IQR$  ou  $x > Q3 + 1.5 \times IQR$
- **Z-Score** : Outlier si  $|z| > 3$  (à 3 écart-types de la moyenne)



# Skewness (Asymétrie)

Mesure de l'asymétrie de la distribution des données

## ← Négative (gauche)

```
  *
 ***
*****
*****
*****
*****
*****
```

Queue à gauche

Ex: Âge de retraite

## ⚖ Symétrique

```
  *
 ***
*****
*****
*****
*****
***
 *
```

Skewness  $\approx 0$

Ex: Taille adultes

## → Positive (droite)

```
*****
*****
*****
*****
***
 *
```

Queue à droite

Ex: Revenus

# Corriger la Skewness

Une forte asymétrie peut affecter les performances de certains modèles (régression linéaire, etc.)

## Transformations courantes

### POUR SKEWNESS POSITIVE ➡

- **Log** :  $\log(x)$
- **Racine carrée** :  $\sqrt{x}$
- **Box-Cox** : Transformation optimale

**Exemple** : Revenus

1k, 2k, 3k, 5k, 100k

→ log: 3, 3.3, 3.5, 3.7, 5

### POUR SKEWNESS NÉGATIVE ⬅

- **Carré** :  $x^2$
- **Exponentielle** :  $\exp(x)$
- **Puissance** :  $x^n$

**Exemple** : Notes d'examen

Beaucoup de 18-20/20

→ Transformation inverse

# Division Train-Test Split

---

Séparer les données en ensembles d'entraînement et de test

**Dataset Complet (100%)**



**Train Set (70-80%)**

Pour entraîner le modèle

**Test Set (20-30%)**

Pour évaluer

**! Règle d'or**

Ne JAMAIS utiliser le test set pendant l'entraînement !

# Pourquoi Diviser les Données ?

## Analogie : Apprendre pour un examen



### TRAIN SET = EXERCICES

- On s'entraîne dessus
- On connaît les réponses
- On apprend les patterns
- On peut les refaire plusieurs fois



### TEST SET = EXAMEN FINAL

- Questions jamais vues
- Test réel des connaissances
- Une seule chance
- Évalue la vraie performance

**Problème sans division :** Le modèle "mémorise" les données au lieu d'apprendre

→ Excellent sur les données connues, nul sur les nouvelles (overfitting)

# Random Split vs Stratified Split



## Random Split

Division aléatoire simple

**Dataset :** 100 emails

- 90 non-spam
- 10 spam

⚠ **Risque : Test set sans spam !**



## Stratified Split

Préserve les proportions

**Résultat garanti :**

- Train : 72 non-spam, 8 spam
- Test : 18 non-spam, 2 spam

✅ **Proportions préservées !**

**Règle :** Toujours utiliser Stratified Split pour les datasets déséquilibrés

# Le Validation Set

Un 3ème ensemble pour ajuster les hyperparamètres sans toucher au test set

**Train (60%)**

Entraînement

**Validation (20%)**

Ajustement

**Test (20%)**

Évaluation finale

## Utilisation typique

1. **Train** : Entraîner différents modèles
2. **Validation** : Choisir le meilleur modèle et hyperparamètres
3. **Test** : Évaluation finale (une seule fois !)

# ⚠ Piège Critique : Scaling sur Train/Test

## ✗ ERREUR COMMUNE (Data Leakage)

```
# MAUVAIS - Ne faites JAMAIS ça !
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X) # Fit sur TOUT
X_train, X_test = train_test_split(X_scaled, y)
```

Le scaler a "vu" les données de test → Fuite d'information !

## ✓ BONNE PRATIQUE

```
# CORRECT - Toujours faire ça !
X_train, X_test = train_test_split(X, y)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train) # Fit sur train
X_test_scaled = scaler.transform(X_test) # Transform seulement
```

# Pourquoi le Scaling Correct est Crucial ?

## Exemple concret : Prédire les revenus

### SI ON SCALE SUR TOUT ❌

- Min dataset : 15k€
- Max dataset : 200k€
- Test contient le max (200k€)
- → Le modèle "sait" qu'il existe un 200k€

### SI ON SCALE CORRECTEMENT ✅

- Min train : 15k€
- Max train : 150k€
- Test avec 200k€ → valeur  $> 1$
- → Situation réaliste

**Règle d'or :** Le test set représente le futur. On ne peut pas utiliser le futur pour préparer le présent !



# Cross-Validation (Validation Croisée)

Technique pour mieux évaluer le modèle quand on a peu de données

## K-Fold Cross-Validation (K=5)

Fold 1:	Test	Train	Train	Train	Train
Fold 2:	Train	Test	Train	Train	Train
Fold 3:	Train	Train	Test	Train	Train
Fold 4:	Train	Train	Train	Test	Train
Fold 5:	Train	Train	Train	Train	Test

**Score final :** Moyenne des 5 scores = estimation robuste

# Overfitting vs Underfitting

---

## Underfitting

Modèle trop simple

**Train** : Mauvais ❌

**Test** : Mauvais ❌

**Analogie** : Étudiant qui n'a pas assez révisé

## Good Fit

Modèle équilibré

**Train** : Bon ✅

**Test** : Bon ✅

**Analogie** : Étudiant qui comprend les concepts

## Overfitting

Modèle trop complexe

**Train** : Excellent ✅

**Test** : Mauvais ❌

**Analogie** : Étudiant qui mémorise sans comprendre

# Matrice de Confusion

Tableau résumant les prédictions correctes et incorrectes

## Exemple : Détection de spam

	Prédit: Spam	Prédit: Non-Spam
Réel: Spam	23 (TP)	1 (FN)
Réel: Non-Spam	12 (FP)	556 (TN)

- **TP (True Positive)** : Spam correctement détecté
- **TN (True Negative)** : Non-spam correctement identifié
- **FP (False Positive)** : Non-spam classé comme spam (erreur)
- **FN (False Negative)** : Spam non détecté (erreur)

# Précision et Rappel

## Précision

$$TP / (TP + FP)$$

"Parmi ce que j'ai prédit positif, combien sont vraiment positifs ?"

$$23 / (23 + 12) = 65.7\%$$

Sur 35 "spam" détectés, 23 vrais

## Rappel

$$TP / (TP + FN)$$

"Parmi tous les positifs réels, combien ai-je trouvé ?"

$$23 / (23 + 1) = 95.8\%$$

Sur 24 vrais spam, 23 détectés

# Le Compromis Précision-Rappel

## Scénario : Diagnostic médical pour cancer

### HAUTE PRÉCISION

"Si je dis cancer, c'est sûr"

- Peu de faux positifs
- Évite les traitements inutiles
- MAIS : peut rater des cas

### HAUT RAPPEL

"Je ne rate aucun cancer"

- Peu de faux négatifs
- Détecte tous les malades
- MAIS : beaucoup de fausses alertes

## Question clé : Qu'est-ce qui est pire pour votre problème ?

- Rater un cas positif (privilégier le rappel)
- Avoir des faux positifs (privilégier la précision)

# Accuracy (Exactitude)

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total}$$

Proportion de prédictions correctes

## Calcul avec notre exemple spam

$$\text{Accuracy} = (23 + 556) / 592 = \mathbf{97.8\%}$$

Semble excellent ! Mais...

## ⚠ Piège : Classes déséquilibrées

**Dataset** : 990 non-frauduleux, 10 frauduleux

**Modèle stupide** : Toujours prédire "non-frauduleux"

**Accuracy** : 99% ! ... mais 0% de fraudes détectées 🤖

→ L'accuracy est trompeuse avec des classes déséquilibrées

# F1-Score

Moyenne harmonique de la précision et du rappel

$$F1 = 2 \times (\text{Précision} \times \text{Rappel}) / (\text{Précision} + \text{Rappel})$$

## Pourquoi utiliser le F1-Score ?

- Balance entre précision et rappel
- Un seul nombre pour comparer les modèles
- Pénalise les valeurs extrêmes

## Exemple :

- Précision = 65.7%, Rappel = 95.8%

# Métriques pour la Régression

---

## MSE (Mean Squared Error)

$$\text{MSE} = \Sigma(y_{\text{pred}} - y_{\text{real}})^2 / n$$

- Pénalise fortement les grosses erreurs
- Unité au carré (€² si on prédit des €)

## RMSE (Root MSE)

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

- Même unité que y
- Plus interprétable

## MAE (Mean Absolute Error)

$$\text{MAE} = \Sigma|y_{\text{pred}} - y_{\text{real}}| / n$$

- Moins sensible aux outliers
- Erreur moyenne absolue

## R² (Coefficient de détermination)

$$R^2 \in [0, 1]$$

- % de variance expliquée
- 1 = parfait, 0 = nul



# Pipeline de Préparation Complet

---

- 1** **Exploration des données** - Types, distributions, valeurs manquantes
- 2** **Nettoyage** - Gérer les valeurs manquantes et outliers
- 3** **Feature Engineering** - Créer de nouvelles features informatives
- 4** **Encodage** - One-hot, label encoding
- 5** **Split** - Train/Validation/Test
- 6** **Scaling** - Fit sur train, transform sur test

# Conseils Pratiques

---

## ✓ Bonnes Pratiques

- Toujours explorer les données d'abord
- Documenter toutes les transformations
- Garder une version des données brutes
- Utiliser des pipelines reproductibles
- Valider sur plusieurs métriques

## ✗ Erreurs Communes

- Oublier de gérer les valeurs manquantes
- Scaler avant le split
- Utiliser le test set pour choisir
- Ignorer les classes déséquilibrées
- Se fier uniquement à l'accuracy

**Règle d'or :** 80% du travail en ML est dans la préparation des données !

# Points Clés à Retenir

---

1. **Les données sont cruciales** - "Garbage in, garbage out"
2. **Comprendre vos variables** - Types, distributions, relations
3. **Feature engineering** - Créativité + connaissance domaine
4. **Encodage approprié** - One-hot vs label encoding
5. **Train-Test split sacré** - Ne jamais contaminer le test
6. **Scaling après split** - Éviter le data leakage
7. **Métriques multiples** - Pas seulement l'accuracy
8. **Itération constante** - Le ML est un processus itératif

# Questions ?

**Joseph Azar**

joseph.azar@univ-fcomte.fr

Université Marie Louis Pasteur

Laboratoire Femto-ST

"Les données sont le nouveau pétrole, mais comme le pétrole, elles doivent être raffinées pour être utiles."