Cours Machine Learning

Chapitre 00

Introduction au Machine Learning

Objectifs d'apprentissage :

— Comprendre ce qu'est le Machine Learning et son évolution historique
— Maîtriser les différents types d'apprentissage automatique
— Connaître le vocabulaire fondamental et le pipeline ML typique
— Distinguer ML classique et Deep Learning
— Identifier les applications concrètes et les enjeux éthiques

Prérequis : Aucun (chapitre d'introduction)

Durée estimée : 4-6 heures (incluant formation Python)

Notebooks : `00_prerequis_python.ipynb`, $00_demo_ml\_pipeline.ipynb$,

`Cours ML - Sandbox-ML`
`Version 1.0 - 2026`

Table des matières

1 Qu'est-ce que le Machine Learning ?

1.1 Définition

Définition : Machine Learning (Apprentissage Automatique)

Le Machine Learning est un sous-domaine de l'intelligence artificielle qui permet aux ordinateurs d'apprendre à partir de données sans être explicitement programmés. Un programme apprend d'une expérience E par rapport à une tâche T et une mesure de performance P, si sa performance sur T, mesurée par P, s'améliore avec l'expérience E.

Cette définition classique de Tom Mitchell (1997) met en évidence trois composantes essentielles :

- **Tâche (T)** : Ce que le système doit accomplir (classification, prédiction, recommandation, etc.)
- **Expérience (E)** : Les données utilisées pour l'apprentissage
- **Performance (P)** : La métrique utilisée pour évaluer la qualité du système

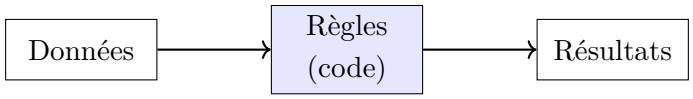
Exemple : Reconnaissance de spam

- **Tâche :** Classifier un email comme spam ou non-spam
- **Expérience :** Base de données d'emails déjà étiquetés (spam/non-spam)
- **Performance :** Pourcentage d'emails correctement classifiés

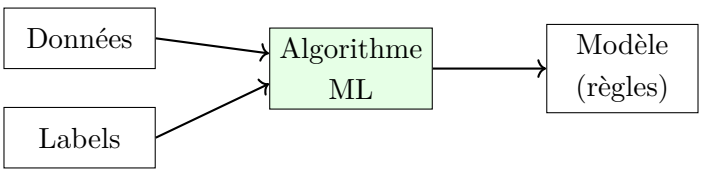
1.2 Programmation Traditionnelle vs Machine Learning

Le Machine Learning inverse le paradigme de la programmation traditionnelle :

Programmation Traditionnelle



Machine Learning



Programmation traditionnelle : On écrit des règles explicites pour transformer les données en résultats.

Machine Learning : On fournit des données et des résultats attendus, et l'algorithme apprend les règles automatiquement.

1.3 Quand utiliser le Machine Learning ?

Astuce

Le Machine Learning est particulièrement adapté quand :

- Le problème est trop complexe pour être résolu par des règles explicites
- Les règles changent fréquemment (adaptation dynamique nécessaire)
- Il existe de grandes quantités de données disponibles
- Le problème nécessite de la personnalisation (recommandations, publicité ciblée)
- On cherche à découvrir des patterns cachés dans les données

Attention

Éviter le Machine Learning quand :

- Des règles simples et explicites suffisent
- Les données sont insuffisantes ou de mauvaise qualité
- L'explicabilité est critique et le modèle doit être interprétable
- Le coût de calcul est prohibitif pour le bénéfice attendu

2 Histoire du Machine Learning (1950-2026)

2.1 Chronologie des Événements Majeurs

2.2 Les Trois Vagues du Machine Learning

Première vague (1950-1980) : L'ère symbolique

- Règles logiques et systèmes experts
- Limitations : incapacité à gérer l'incertitude et la complexité

Deuxième vague (1980-2010) : L'apprentissage statistique

- Approches probabilistes et statistiques
- SVMs, Random Forests, Gradient Boosting
- Méthodes basées sur des features engineered manuellement

Troisième vague (2010-aujourd'hui) : Le Deep Learning

- Réseaux de neurones profonds
- Apprentissage automatique de représentations hiérarchiques
- Performances surhumaines sur certaines tâches

3 Types d'Apprentissage Automatique

3.1 Apprentissage Supervisé

Définition : Apprentissage Supervisé

L'apprentissage supervisé consiste à apprendre une fonction de mapping $f : X \to Y$ à partir d'un ensemble de données étiquetées $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ où $\mathbf{x}_i$ sont les features (entrées) et y_i sont les labels (sorties attendues).

Formulation mathématique :

Table 1 – Principales étapes historiques du Machine Learning

Période	Événements Marquants
1950s	— 1950 : Test de Turing (Alan Turing) — 1957 : Perceptron (Frank Rosenblatt) — 1959 : Terme "Machine Learning" (Arthur Samuel)
1960-70s	— 1969 : Limitations du perceptron (Minsky & Papert) — Premier "hiver de l'IA" (1974-1980) — 1979 : Backpropagation (Werbos)
1980s	— 1986 : Popularisation du backpropagation (Rumelhart et al.) — Développement des réseaux de neurones multicouches — Systèmes experts et IA symbolique
1990s	— 1995 : Support Vector Machines (Vapnik) — 1997 : Random Forests (Ho), LSTM (Hochreiter & Schmidhuber) — 1997 : Deep Blue bat Kasparov aux échecs
2000s	— 2001 : Gradient Boosting Machines — 2006 : Deep Learning renaissance (Hinton et al.) — 2009 : ImageNet dataset créé
2010s	— 2012 : AlexNet révolutionne la vision par ordinateur — 2014 : GANs (Generative Adversarial Networks - Goodfellow) — 2016 : AlphaGo bat Lee Sedol au Go — 2017 : Transformers (Attention is All You Need) — 2018 : BERT pour le NLP
2020s	— 2020 : GPT-3 (175B paramètres) — 2021 : AlphaFold résout le repliement des protéines — 2022 : ChatGPT, Stable Diffusion, DALL-E 2 — 2023 : GPT-4, LLaMA, modèles multimodaux — 2024-2026 : Démocratisation de l'IA générative, LLMs open-source

On cherche à minimiser le risque empirique :

$$\hat{f} = \text{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) \tag{1}$$

où L est une fonction de perte et $\mathcal{F}$ est l'espace des fonctions considérées.

Deux grandes familles :

— **Régression :** Y est continu (ex : prédire le prix d'une maison)
  — Régression linéaire, Ridge, Lasso
  — Arbres de décision, Random Forests
  — SVR (Support Vector Regression)
  — Réseaux de neurones

- **Classification :** Y est discret (ex : spam/non-spam)
  - Régression logistique
  - K-Nearest Neighbors (KNN)
  - Arbres de décision, Random Forests
  - SVM (Support Vector Machines)
  - Réseaux de neurones

Exemple : Applications de l'apprentissage supervisé

- **Vision :** Détection de visages, classification d'images médicales
- **NLP :** Analyse de sentiment, traduction automatique
- **Finance :** Prédiction de cours boursiers, scoring crédit
- **Santé :** Diagnostic médical, prédiction de maladies

3.2 Apprentissage Non-Supervisé

Définition : Apprentissage Non-Supervisé

L'apprentissage non-supervisé consiste à découvrir des structures cachées dans des données non étiquetées $\{\mathbf{x}_i\}_{i=1}^n$. Le but est d'apprendre la distribution sous-jacente des données ou de les organiser de manière significative.

Principales tâches :

- **Clustering :** Regrouper des données similaires
  - K-Means, DBSCAN, Hierarchical Clustering
  - Gaussian Mixture Models
- **Réduction de dimensionnalité :** Projeter les données dans un espace de dimension réduite
  - PCA (Principal Component Analysis)
  - t-SNE, UMAP
  - Autoencoders
- **Détection d'anomalies :** Identifier les points atypiques
  - Isolation Forest
  - One-Class SVM
  - Autoencoders variationnels

Exemple : Applications de l'apprentissage non-supervisé

- **Marketing :** Segmentation de clients
- **Sécurité :** Détection de fraudes, intrusions réseau
- **Biologie :** Découverte de nouveaux types cellulaires
- **Recommandation :** Systèmes de recommandation (collaborative filtering)

3.3 Apprentissage Semi-Supervisé

Définition : Apprentissage Semi-Supervisé

L'apprentissage semi-supervisé combine un petit ensemble de données étiquetées $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_l}$ avec un grand ensemble de données non étiquetées $\{\mathbf{x}_j\}_{j=1}^{n_u}$ où généralement $n_u \gg n_l$.

Motivation : L'étiquetage des données est souvent coûteux et chronophage, tandis que les données non étiquetées sont abondantes.

Approches principales :

- Self-training et co-training
- Modèles génératifs semi-supervisés
- Graph-based methods
- Consistency regularization (modèles modernes)

3.4 Apprentissage par Renforcement

Définition : Apprentissage par Renforcement

L'apprentissage par renforcement consiste à apprendre une politique π qui maximise la récompense cumulative d'un agent interagissant avec un environnement. L'agent apprend par essai-erreur en recevant des récompenses (positives ou négatives) pour ses actions.

Composantes principales :

- **Agent :** Entité qui prend des décisions
- **Environnement :** Monde avec lequel l'agent interagit
- **État :** Configuration actuelle de l'environnement
- **Action :** Choix effectué par l'agent
- **Récompense :** Signal de feedback (positif/négatif)
- **Politique :** Stratégie de l'agent ($\pi : S \to A$)

Objectif : Maximiser la récompense cumulative espérée

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi\right] \tag{2}$$

où $\gamma \in [0, 1]$ est le facteur de discount et r_t la récompense au temps t.

Exemple : Applications de l'apprentissage par renforcement

- **Jeux :** AlphaGo, AlphaZero (Go, échecs, shogi)
- **Robotique :** Contrôle de robots, manipulation d'objets
- **Finance :** Trading automatique
- **Véhicules autonomes :** Conduite autonome
- **Optimisation :** Gestion de datacenters, allocation de ressources

4 Vocabulaire Fondamental

4.1 Terminologie des Données

TABLE 2 – Vocabulaire clé du Machine Learning

Terme	Définition
Dataset	Ensemble complet de données utilisées pour le ML
Sample/Instance	Une observation individuelle (une ligne du dataset)
Feature/Variable	Attribut ou caractéristique mesurée (colonne)
Label/Target	Valeur à prédire (variable dépendante)
Training Set	Données utilisées pour entraîner le modèle (70-80%)
Validation Set	Données pour ajuster les hyperparamètres (10-15%)
Test Set	Données pour évaluer la performance finale (10-15%)
Feature Vector	Vecteur $\mathbf{x} \in \mathbb{R}^d$ représentant une instance
Dimensionnalité	Nombre de features (d)

4.2 Terminologie des Modèles

Terme	Définition
Modèle	Fonction mathématique apprise $f : X \to Y$
Paramètres	Variables internes apprises par le modèle (poids $\mathbf{w}$)
Hyperparamètres	Paramètres fixés avant l'entraînement (learning rate, etc.)
Entraînement/Fit	Processus d'apprentissage des paramètres
Inférence/Prédiction	Utilisation du modèle entraîné sur de nouvelles données
Fonction de perte	Mesure de l'erreur du modèle $L(y, \hat{y})$
Optimisation	Processus de minimisation de la fonction de perte
Epoch	Une passe complète sur tout le dataset d'entraînement
Batch	Sous-ensemble de données traité simultanément

4.3 Problématiques Clés

Définition : Overfitting (Surapprentissage)

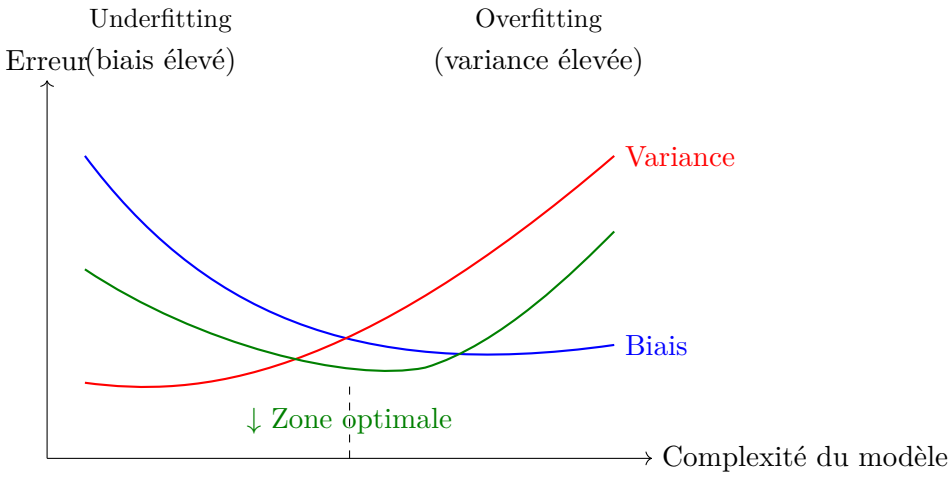
Un modèle en overfitting apprend trop bien les données d'entraînement, incluant le bruit, et généralise mal sur de nouvelles données. Performance élevée sur le train set, faible sur le test set.

Définition : Underfitting (Sous-apprentissage)

Un modèle en underfitting est trop simple pour capturer les patterns des données. Performance faible sur le train set et le test set.

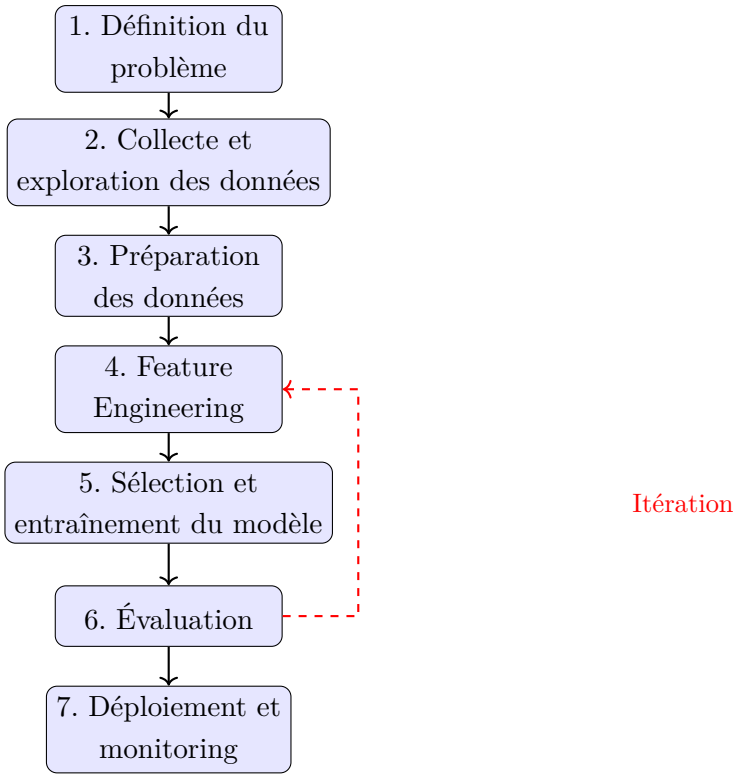
Définition : Compromis Biais-Variance

- **Biais élevé :** Le modèle fait des hypothèses trop fortes (underfitting)
- **Variance élevée :** Le modèle est trop sensible aux variations des données (overfitting)
- **Objectif :** Trouver le bon équilibre entre biais et variance



5 Pipeline Typique d'un Projet Machine Learning

Un projet ML suit généralement les étapes suivantes :



5.1 Définition du Problème

Questions à se poser :

- Quel est le problème business/scientifique à résoudre ?
- Est-ce un problème de classification, régression, clustering, etc. ?
- Quelles métriques de succès utiliser ?
- Quelles sont les contraintes (temps, ressources, explicabilité) ?

5.2 Collecte et Exploration des Données

Analyse exploratoire des données (EDA) :

- Taille du dataset (n instances, d features)
- Types de variables (numériques, catégorielles, textuelles)
- Distribution des variables
- Valeurs manquantes
- Outliers (valeurs aberrantes)
- Corrélations entre features
- Déséquilibre des classes (pour la classification)

5.3 Préparation des Données

Nettoyage des données :

- **Gestion des valeurs manquantes :**
  - Suppression des lignes/colonnes
  - Imputation (moyenne, médiane, mode, KNN, etc.)
- **Gestion des outliers :**
  - Détection (IQR, Z-score)

  - Suppression ou transformation
- **Encodage des variables catégorielles :**
  - One-Hot Encoding
  - Label Encoding
  - Target Encoding
- **Normalisation/Standardisation :**
  - Min-Max scaling : $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$
  - Z-score standardization : $x' = \frac{x - \mu}{\sigma}$

5.4 Feature Engineering

Création de nouvelles features pertinentes à partir des données brutes :

- Combinaisons de features existantes
- Transformations mathématiques (log, sqrt, polynomiales)
- Extraction de features temporelles (jour, mois, jour de la semaine)
- Agrégations et statistiques
- Sélection de features (élimination des redondantes)

5.5 Entraînement du Modèle

```python
# 1. Split train/validation/test (80% / 10% / 10%)
from sklearn.model_selection import train_test_split

X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.2, random_state=42  # 80% train, 20% temp
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42  # 10% val, 10% test
)

# 2. Choix et instanciation du modèle
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=42)

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

# 4. Prédictions
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)
```

Listing 1 – Schéma général d'entraînement

5.6 Évaluation

Métriques pour la classification :

- Accuracy, Precision, Recall, F1-score
- Confusion Matrix

— ROC-AUC, PR-AUC

Métriques pour la régression :

— MAE (Mean Absolute Error)
— MSE (Mean Squared Error)
— RMSE (Root Mean Squared Error)
— Rš (Coefficient de détermination)

Validation croisée :

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
print(f"Accuracy: {scores.mean():.3f} (+/- {scores.std():.3f})")
```

5.7 Déploiement et Monitoring

— Sauvegarde du modèle (pickle, joblib, ONNX)
— Mise en production (API REST, batch processing)
— Monitoring des performances en production
— Réentraînement périodique (drift des données)

6 Machine Learning Classique vs Deep Learning

6.1 Machine Learning Classique

Caractéristiques :

— Features engineered manuellement
— Modèles relativement simples (arbres, SVMs, régression)
— Nécessite expertise domaine pour feature engineering
— Moins de données nécessaires
— Plus rapide à entraîner
— Plus interprétable

Algorithmes typiques :

— Régression linéaire/logistique
— Arbres de décision, Random Forests, Gradient Boosting (XGBoost, LightGBM)
— Support Vector Machines
— K-Nearest Neighbors
— Naive Bayes

6.2 Deep Learning

Caractéristiques :

— Apprentissage automatique de features hiérarchiques
— Réseaux de neurones multicouches (profonds)
— Nécessite beaucoup de données
— Gourmand en ressources de calcul (GPU/TPU)
— Performances supérieures sur données non structurées (images, texte, audio)
— Moins interprétable ("boîte noire")

Architectures typiques :

- **MLP (Multi-Layer Perceptron)** : Réseaux fully-connected
- **CNN (Convolutional Neural Networks)** : Vision par ordinateur
- **RNN/LSTM/GRU** : Séquences temporelles
- **Transformers** : NLP moderne, vision, multimodal
- **Autoencoders, GANs** : Génération et représentation

6.3 Comparaison

Table 3 – ML Classique vs Deep Learning

Critère	ML Classique	Deep Learning
Données	Fonctionne avec peu de données	Nécessite beaucoup de données
Features	Engineering manuel	Apprises automatiquement
Performance	Plateaux rapidement	S'améliore avec plus de données
Temps d'entraînement	Rapide (minutes-heures)	Lent (heures-jours)
Hardware	CPU suffit	GPU/TPU recommandé
Interprétabilité	Bonne	Faible
Données structurées	Excellent	Bon
Données non structurées	Limité	Excellent

Astuce

Quand utiliser quoi ?

- **ML Classique :** Données tabulaires, peu de données, besoin d'interprétabilité, ressources limitées
- **Deep Learning :** Images, texte, audio, grandes quantités de données, performance maximale

7 Applications Concrètes du Machine Learning

7.1 Vision par Ordinateur

Tâches :

- **Classification d'images :** Reconnaître des objets, animaux, scènes
- **Détection d'objets :** Localiser et identifier plusieurs objets (YOLO, R-CNN)
- **Segmentation :** Délimiter précisément les objets pixel par pixel
- **Reconnaissance faciale :** Identification biométrique
- **Génération d'images :** GANs, Diffusion Models (Stable Diffusion, DALL-E)

Applications :

- Véhicules autonomes (détection piétons, panneaux, véhicules)
- Diagnostic médical (radiologie, dermatologie)
- Surveillance et sécurité
- Contrôle qualité industriel

— Réalité augmentée

7.2 Natural Language Processing (NLP)

Tâches :

— **Classification de texte :** Analyse de sentiment, détection de spam
— **Named Entity Recognition :** Extraction d'entités (personnes, lieux, organisations)
— **Traduction automatique :** Google Translate, DeepL
— **Question-Answering :** Systèmes de Q&A
— **Génération de texte :** GPT-4, LLaMA, chatbots
— **Résumé automatique :** Synthèse de documents

Applications :

— Assistants virtuels (Alexa, Siri, Google Assistant)
— Chatbots customer service
— Traduction en temps réel
— Analyse de documents légaux/médicaux
— Génération de code (GitHub Copilot)

7.3 Systèmes de Recommandation

Approches :

— **Filtrage collaboratif :** Basé sur les préférences d'utilisateurs similaires
— **Filtrage basé contenu :** Basé sur les caractéristiques des items
— **Approches hybrides :** Combinaison des deux

Applications :

— Netflix, YouTube (recommandations de vidéos)
— Amazon, Alibaba (produits)
— Spotify, Apple Music (musique)
— Publicité ciblée

7.4 Jeux et Stratégie

Succès remarquables :

— 1997 : Deep Blue bat Kasparov aux échecs
— 2016 : AlphaGo bat Lee Sedol au Go
— 2017 : AlphaZero maîtrise échecs, shogi, Go sans connaissance humaine
— 2019 : AlphaStar atteint niveau grand-maître à StarCraft II
— 2021 : AlphaCode résout des problèmes de programmation compétitive

7.5 Autres Domaines

— **Finance :** Trading algorithmique, détection de fraude, scoring crédit
— **Santé :** Découverte de médicaments (AlphaFold), diagnostic, médecine personnalisée
— **Climatologie :** Prévisions météorologiques, modélisation climatique
— **Agriculture :** Optimisation de rendements, détection de maladies
— **Énergie :** Optimisation de réseaux électriques, prévision de consommation
— **Cybersécurité :** Détection d'intrusions, malware analysis

8 Éthique et Biais en Machine Learning

8.1 Problématiques Éthiques

Attention

Le Machine Learning soulève des questions éthiques importantes :

- **Biais algorithmiques :** Discrimination basée sur race, genre, âge
- **Vie privée :** Collecte et utilisation de données personnelles
- **Transparence :** Modèles opaques ("boîtes noires")
- **Responsabilité :** Qui est responsable des erreurs ?
- **Sécurité :** Attaques adverses, deepfakes
- **Impact social :** Automatisation et emploi

8.2 Biais dans les Données et les Modèles

Sources de biais :

1. **Biais de collecte :** Données non représentatives de la population
2. **Biais de labeling :** Étiquetage subjectif ou inconsistant
3. **Biais historique :** Données reflétant des discriminations passées
4. **Biais de sélection :** Échantillonnage non aléatoire
5. **Biais de confirmation :** Chercher des patterns qui confirment des préjugés

Exemple : Cas concrets de biais

- **COMPAS :** Système de prédiction de récidive aux USA biaisé contre les minorités
- **Recrutement :** Amazon a dû abandonner un outil de tri de CV biaisé contre les femmes
- **Reconnaissance faciale :** Taux d'erreur plus élevés pour les personnes de couleur
- **Traduction :** Associations stéréotypées (ex : "doctor" $\rightarrow$ "il", "nurse" $\rightarrow$ "elle")

8.3 Fairness et Équité

Définitions de fairness :

- **Demographic parity :** Prédictions indépendantes des attributs sensibles
- **Equalized odds :** Taux de vrais/faux positifs égaux entre groupes
- **Individual fairness :** Individus similaires traités de manière similaire

Techniques de mitigation :

- Pré-traitement : Correction des biais dans les données
- In-processing : Contraintes de fairness pendant l'entraînement
- Post-processing : Ajustement des prédictions pour assurer l'équité

8.4 Bonnes Pratiques

Astuce

Développement responsable de ML :

- **Auditer les données :** Vérifier la représentativité et les biais
- **Tester la fairness :** Évaluer les performances par sous-groupes
- **Documenter :** Model cards, datasheets pour les datasets
- **Explicabilité :** Utiliser LIME, SHAP pour comprendre les prédictions
- **Human-in-the-loop :** Validation humaine pour décisions critiques
- **Monitoring continu :** Surveiller les performances en production
- **Diversité :** Équipes diverses pour identifier les angles morts

8.5 Réglementation

Cadres légaux émergents :

- **RGPD (Europe)** : Protection des données personnelles, droit à l'explication
- **AI Act (UE)** : Classification des systèmes IA par niveau de risque
- **Algorithmic Accountability Act (USA)** : Transparence des algorithmes
- **IEEE, ISO :** Standards techniques pour l'IA éthique

9 Écosystème et Outils

9.1 Bibliothèques Python Essentielles

Manipulation de données :

- `NumPy` : Calcul numérique, arrays multidimensionnels
- `Pandas` : DataFrames, manipulation de données tabulaires
- `Polars` : Alternative rapide à Pandas

Machine Learning classique :

- `scikit-learn` : ML général (classification, régression, clustering)
- `XGBoost, LightGBM, CatBoost` : Gradient boosting optimisé
- `statsmodels` : Modèles statistiques

Deep Learning :

- `PyTorch` : Framework DL flexible et pythonic
- `TensorFlow/Keras` : Framework DL production-ready
- `JAX` : Calcul numérique haute performance
- `Hugging Face Transformers` : NLP state-of-the-art

Visualisation :

- `Matplotlib` : Graphiques standard
- `Seaborn` : Visualisations statistiques
- `Plotly` : Graphiques interactifs

9.2 Plateformes et Environnements

- **Jupyter Notebooks** : Développement interactif
- **Google Colab** : Jupyter gratuit avec GPU/TPU

— **Kaggle** : Compétitions, datasets, notebooks
— **Weights & Biases, MLflow** : Tracking d'expériences
— **TensorBoard** : Visualisation d'entraînement

10 Prérequis Python pour le Machine Learning

Avant de vous lancer dans le Machine Learning, il est essentiel de maîtriser les outils Python de base. Cette section vous guide à travers les bibliothèques fondamentales que vous utiliserez tout au long du cours.

Notebook Pratique

Le notebook `00_prerequis_python.ipynb` couvre en détail tous ces prérequis avec des exemples exécutables et des exercices pratiques.
Si vous connaissez déjà Python, NumPy et Pandas, vous pouvez passer cette section et ce notebook.

10.1 Python : Syntaxe Essentielle

Types de base et structures :
— Types : `int`, `float`, `str`, `bool`
— Listes : collections ordonnées modifiables
— Dictionnaires : paires clé-valeur
— Tuples : collections immuables

List comprehensions : Construction compacte de listes, très utilisée en ML.

```
# Exemple : normaliser des scores
scores = [0.85, 0.92, 0.78, 0.95, 0.88]
scores_percent = [score * 100 for score in scores]
# [85.0, 92.0, 78.0, 95.0, 88.0]

# Filtrage
bons_scores = [s for s in scores if s >= 0.90]
# [0.92, 0.95]
```

Fonctions : Encapsulation de logique réutilisable.

```
def normaliser(valeur, min_val, max_val):
    """Normalise une valeur entre 0 et 1"""
    return (valeur - min_val) / (max_val - min_val)
```

10.2 NumPy : Calcul Numérique

NumPy est **la bibliothèque fondamentale** pour le calcul scientifique en Python. Tous les frameworks ML (scikit-learn, PyTorch, TensorFlow) sont construits sur NumPy.

Arrays multidimensionnels :

```
import numpy as np

```

```python
# Vecteur (1D)
v = np.array([1, 2, 3, 4, 5])

# Matrice (2D) : dataset avec 3 personnes, 4 features
X = np.array([
    [25, 50000, 1, 1],  # age, salaire, exp, diplome
    [30, 60000, 3, 2],
    [35, 75000, 5, 3]
])

print(X.shape)  # (3, 4) : 3 lignes, 4 colonnes
```

Opérations vectorisées (rapides) :

```python
# Element-wise operations
a = np.array([1, 2, 3, 4])
b = np.array([10, 20, 30, 40])

print(a + b)   # [11 22 33 44]
print(a * b)   # [10 40 90 160]
print(a ** 2)  # [1 4 9 16]
```

Indexation et slicing :

```python
X = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Selection
X[0, :]     # Premiere ligne : [1, 2, 3]
X[:, 1]     # Deuxieme colonne : [2, 5, 8]
X[0, 1]     # Element : 2

# Masquage booleen (tres utilise en ML)
X[X > 5]    # [6, 7, 8, 9]
```

Broadcasting : Opérations entre arrays de shapes différentes.

```python
matrice = np.array([[1, 2, 3], [4, 5, 6]])
vecteur = np.array([10, 20, 30])

# Le vecteur est "diffuse" sur chaque ligne
resultat = matrice + vecteur
# [[11 22 33]
#  [14 25 36]]
```

Fonctions d'agrégation :

```python
scores = np.array([0.75, 0.82, 0.91, 0.88, 0.79])

print(scores.mean())     # 0.83
print(scores.std())      # 0.059
print(scores.min())      # 0.75
print(scores.max())      # 0.91
```

```
7
8  # Agregation par axe
9  X.mean(axis=0)  # Moyenne par colonne
10 X.mean(axis=1)  # Moyenne par ligne
```

10.3 Pandas : Manipulation de Données

Pandas permet de manipuler des données tabulaires (comme Excel ou CSV). C'est l'outil principal pour l'exploration et le preprocessing de données.

DataFrames : Tableaux avec labels de colonnes et lignes.

```
1  import pandas as pd
2
3  # Creation
4  data = {
5      'nom': ['Alice', 'Bob', 'Charlie'],
6      'age': [25, 30, 35],
7      'salaire': [45000, 55000, 65000]
8  }
9  df = pd.DataFrame(data)
10
11 #       nom  age  salaire
12 # 0   Alice   25    45000
13 # 1     Bob   30    55000
14 # 2 Charlie   35    65000
```

Sélection et filtrage :

```
1  # Selection de colonnes
2  df['age']              # Serie (1 colonne)
3  df[['nom', 'age']]     # DataFrame (plusieurs colonnes)
4
5  # Filtrage
6  seniors = df[df['age'] >= 30]
7  riches = df[(df['age'] >= 30) & (df['salaire'] > 50000)]
```

Statistiques descriptives :

```
1  df.describe()          # Statistiques sur colonnes numeriques
2  df['salaire'].mean()   # Salaire moyen
3  df['age'].median()     # Age median
4  df['nom'].value_counts()  # Comptage des valeurs
```

Groupby (SQL GROUP BY) :

```
1  # Salaire moyen par diplome
2  df.groupby('diplome')['salaire'].mean()
```

Lecture de fichiers :

```python
# CSV (le plus courant)
df = pd.read_csv('data.csv')

# Excel
df = pd.read_excel('data.xlsx')

# JSON
df = pd.read_json('data.json')
```

10.4 Matplotlib : Visualisation

Matplotlib est la bibliothèque de base pour créer des graphiques. Elle est utilisée pour visualiser les données et les résultats des modèles.

Graphiques de base :

```python
import matplotlib.pyplot as plt

# Courbe
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Mon graphique')
plt.show()

# Nuage de points
plt.scatter(X, y)
plt.show()

# Histogramme
plt.hist(scores, bins=20)
plt.show()
```

Subplots (plusieurs graphiques) :

```python
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes[0, 0].plot(x, y)        # Haut gauche
axes[0, 1].scatter(x, y)     # Haut droite
axes[1, 0].hist(scores)      # Bas gauche
axes[1, 1].bar(['A','B'], [1, 2])  # Bas droite

plt.tight_layout()
plt.show()
```

10.5 Installation et Vérification

Si vous utilisez ce cours avec Docker, toutes les bibliothèques sont déjà installées. Sinon, installez-les avec :

```
pip install numpy pandas matplotlib scikit-learn jupyter
```

Vérification :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print(f"NumPy: {np.__version__}")
print(f"Pandas: {pd.__version__}")
print("Tout est pret !")
```

Attention

Ordre d'apprentissage recommandé Pour bien démarrer ce cours :

1. Commencez par le notebook `00_prerequis_python.ipynb`
2. Faites tous les exercices (normalisation, analyse de dataset)
3. Une fois à l'aise, passez au chapitre 01 (Fondamentaux Mathématiques)

Temps estimé : 1-2 heures pour le notebook Python si vous êtes débutant, 15-30 minutes si vous connaissez déjà les bases.

11 Résumé du Chapitre

11.1 Points Clés

- **Machine Learning :** Apprendre à partir de données sans programmation explicite
- **Types d'apprentissage :** Supervisé (labels), non-supervisé (patterns), semi-supervisé (mix), renforcement (récompenses)
- **Pipeline ML :** Problème → Données → Préparation → Features → Modèle → Évaluation → Déploiement
- **ML vs DL :** ML classique (peu de données, interprétable) vs Deep Learning (beaucoup de données, performance)
- **Applications :** Vision, NLP, recommandation, jeux, santé, finance...
- **Éthique :** Attention aux biais, fairness, transparence, responsabilité

11.2 Formules Essentielles

Concepts à retenir

Apprentissage supervisé :

$$\hat{f} = \text{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) \tag{3}$$

Compromis biais-variance :

$$\text{Erreur totale} = \text{Biais}^2 + \text{Variance} + \text{Bruit irréductible} \tag{4}$$

Train/Validation/Test : Typiquement 70% / 15% / 15%

12 Exercices

12.1 Questions de Compréhension

1. Expliquez la différence entre un paramètre et un hyperparamètre. Donnez des exemples.
2. Pourquoi divise-t-on les données en ensembles d'entraînement, validation et test ?
3. Qu'est-ce que l'overfitting ? Comment peut-on le détecter et le prévenir ?
4. Comparez l'apprentissage supervisé et non-supervisé. Donnez un exemple d'application pour chaque.
5. Dans quelles situations privilégieriez-vous le Deep Learning par rapport au ML classique ?
6. Expliquez comment les biais dans les données peuvent mener à des modèles discriminatoires.
7. Décrivez les étapes du pipeline ML pour un problème de prédiction de prix immobiliers.
8. Pourquoi la standardisation des features est-elle importante pour certains algorithmes (ex : SVM, KNN) ?

12.2 Exercices Pratiques

1. **Pipeline ML complet sur Iris :**
   — Charger le dataset Iris depuis scikit-learn
   — Effectuer une analyse exploratoire (EDA)
   — Séparer train/test
   — Entraîner un classifieur (KNN, Decision Tree, ou Random Forest)
   — Évaluer avec accuracy, confusion matrix, et classification report
   Voir le notebook $00_demo_ml\_pipeline.ipynb$
2. **Comparaison de modèles :**
   — Sur le dataset Iris, comparer 3 algorithmes différents
   — Utiliser la validation croisée
   — Analyser les résultats et expliquer les différences
3. **Détection d'overfitting :**
   — Créer un dataset synthétique avec peu d'échantillons
   — Entraîner un modèle complexe (ex : Decision Tree sans limitation de profondeur)

— Comparer les performances train vs test
— Appliquer des techniques de régularisation et observer l'impact

Solutions détaillées disponibles dans $00_exercices\_solutions.ipynb$

13 Pour Aller Plus Loin

13.1 Lectures Recommandées

Livres :

- *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3e éd., 2022) - Aurélien Géron
- *Pattern Recognition and Machine Learning* (2006) - Christopher Bishop
- *The Elements of Statistical Learning* (2009) - Hastie, Tibshirani, Friedman
- *Deep Learning* (2016) - Goodfellow, Bengio, Courville

Cours en ligne :

- Andrew Ng - Machine Learning (Coursera)
- Fast.ai - Practical Deep Learning for Coders
- Stanford CS229 - Machine Learning
- MIT 6.S191 - Introduction to Deep Learning

13.2 Ressources en Ligne

- Documentation scikit-learn : https://scikit-learn.org/
- PyTorch Tutorials : https://pytorch.org/tutorials/
- Papers With Code : https://paperswithcode.com/
- Kaggle Datasets & Competitions : https://www.kaggle.com/
- Hugging Face : https://huggingface.co/
- Distill.pub (visualisations ML) : https://distill.pub/

13.3 Datasets pour Pratiquer

Débutants :

- Iris, Wine, Digits (scikit-learn)
- Titanic, House Prices (Kaggle)
- MNIST (images de chiffres manuscrits)

Intermédiaires :

- CIFAR-10/100 (images couleur)
- IMDB Reviews (analyse de sentiment)
- UCI Machine Learning Repository

13.4 Prochaines Étapes

Chapitre suivant recommandé : **Chapitre 01 - Fondamentaux Mathématiques**

Prérequis pour approfondir : algèbre linéaire, calcul différentiel, probabilités, statistiques

Références

1. Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
2. Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3e éd.). O'Reilly Media.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
4. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2e éd.). Springer.
5. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
6. Samuel, A. L. (1959). "Some Studies in Machine Learning Using the Game of Checkers". *IBM Journal of Research and Development*, 3(3), 210-229.
7. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). "Learning representations by back-propagating errors". *Nature*, 323(6088), 533-536.
8. Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press.