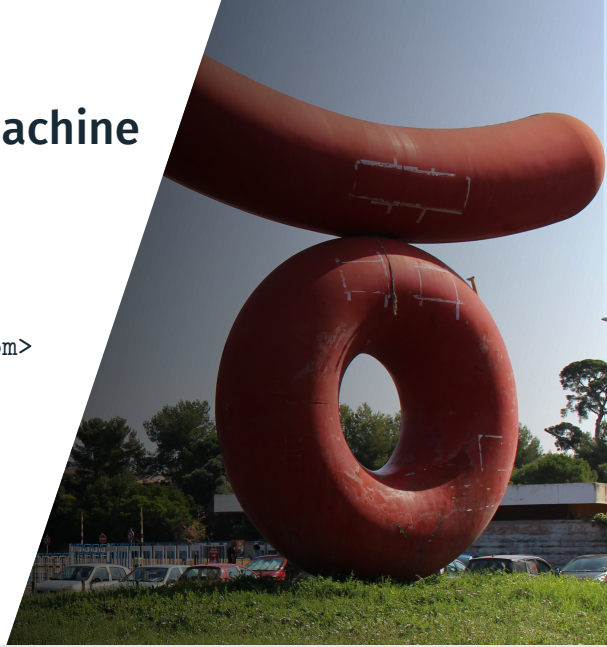
Course: Data Science & Machine Learning

Ali ZAINOUL <ali.zainoul.az@gmail.com>

CrystalClearCode
December 1, 2025



1 Introduction

- Présentation Générale
  - Définition et contexte historique
  - Évolution des besoins industriels
  - Place de la Data Science
  - Rôle de l'ingénieur et du data scientist
  - Applications modernes et enjeux
- Panorama des Compétences Nécessaires
  - Compétences techniques
  - Compétences mathématiques
  - Compétences en programmation
  - Compétences en communication et visualisation
  - Compétences en gestion de données
  - Compétences en modélisation
  - Compétences en déploiement et MLOps
- La Data Science
  - Définition et périmètre
  - Cycle de vie d'un projet Data Science
  - Collecte et prétraitement des données
  - Analyse exploratoire

- Visualisation et interprétation
- Modélisation et expérimentation
- Déploiement et suivi des modèles

- Le Machine Learning
  - Définition générale
  - Catégories d'apprentissage
  - Supervisé vs non supervisé
  - Apprentissage par renforcement
  - Pipeline d'un modèle ML
  - Évaluation et métriques de performance
  - Problèmes courants et biais des modèles
- Le Big Data
  - Définition et caractéristiques (3V, 5V, etc.)
  - Écosystèmes technologiques du Big Data
  - Stockage distribué et systèmes massifs
  - Traitement distribué (batch et streaming)
  - Formats de données massives
  - Applications industrielles du Big Data
- Le Deep Learning
  - Définition et lien avec le Machine Learning

- Lien avec l'approche statistique

- Les Variables Prédictives
  - Définition des variables explicatives
  - Types de variables prédictives
  - Codage et représentation
  - Sélection et importance des variables
  - Traitement et transformation des variables
- Les Variables à Prédire
  - Définition de la variable cible
  - Variable continue vs variable catégorielle
  - Structure des espaces de sortie
  - Problèmes de régression
  - Problèmes de classification
  - Cas particuliers : multilabel, multiclasse, ordinal
- Les Fonctions Hypothèses
  - Définition et rôle
  - Paramétrisation d'un modèle
  - Fonctions hypothèses linéaires
  - Fonctions hypothèses non linéaires
  - Capacité du modèle et approximation

- Hypothèses et espace des hypothèses

- Les Estimateurs : Biais et Variance
  - Notion d'estimateur en statistique
  - Définition du biais
  - Définition de la variance
  - Erreur quadratique moyenne (MSE) d'un estimateur
  - Comportement asymptotique des estimateurs
  - Impact du bruit et de la distribution des données
- Le Compromis Biais–Variance
  - Origine du compromis
  - Compréhension du sur-apprentissage
  - Compréhension du sous-apprentissage
  - Visualisation et interprétation
  - Effet de la complexité du modèle
  - Impact de la taille de l'échantillon
  - Stratégies pour équilibrer le compromis
- Les Fonctions de Perte
  - Définition et rôle dans l'apprentissage
  - Pertes pour la régression
  - Pertes pour la classification

- Propriétés mathématiques importantes
- Convexité et différentiabilité
- Gradients et rétropropagation
- Comparaison entre familles de pertes

La Régularisation des Paramètres

- Motivation et intuition
- Régularisation L1
- Régularisation L2
- Régularisation élastique (Elastic Net)
- Contraintes sur les paramètres
- Impact sur le biais et la variance
- Régularisation avancée : dropout, early stopping

Optimisation des Paramètres

- Problème d'optimisation
- Descente de gradient
- Variants de la descente de gradient
- Optimiseurs modernes (Adam, RMSProp, etc.)
- Conditions de convergence
- Paysages de perte et minima locaux
- Techniques d'accélération

3 La Classification

- Introduction à la Classification
  - Définition du problème de classification
  - Structure des données pour la classification
  - Types de classification (binaire, multiclasse, multilabel)
  - Mesures de performance spécifiques
  - Pipeline général d'un classificateur
  - Lien avec la théorie décisionnelle
- La Régression Logistique
  - Motivation et cadre mathématique
  - Fonction logistique et transformation sigmoïde
  - Fonction de coût (log-loss)
  - Optimisation de la log-vraisemblance
  - Classification binaire : formulation complète
  - Extension multiclasse : softmax et régression logistique multinomiale
  - Interprétabilité du modèle
  - Régularisation dans la régression logistique
- Machines à Vecteurs de Support (SVM)
  - Principe du séparateur optimal
  - Marge maximale et hyperplans

- SVM linéaire
- SVM non linéaire
- Truc des noyaux (kernel trick)
- Pertes hinge et variantes
- Problèmes d'optimisation (primal et dual)
- SVM pour la classification multiclasse

Arbres de Décision

- Structure et terminologie d'un arbre
- Critères de séparation (gini, entropie)
- Processus de construction d'un arbre
- Pruning : pré-élagage et post-élagage
- Avantages et limites des arbres simples
- Extensions : forêts aléatoires, Boosted Trees
- Interprétabilité et visualisation

k Plus Proches Voisins (kNN)

- Principe de proximité
- Choix de la distance (euclidienne, Manhattan, Minkowski, etc.)
- Rôle du paramètre k
- Processus de classification par vote
- Complexité algorithmique et stockage
- Normalisation et métriques adaptées

- Applications, avantages et limites

4 Les Pratiques Essentielles en Machine Learning

- Prétraitement
  - Nettoyage des données
  - Gestion des valeurs manquantes
  - Détection et traitement des outliers
  - Encodage des variables catégorielles
  - Normalisation et standardisation
  - Réduction de dimension
  - Sélection de variables
  - Augmentation de données (data augmentation)
- Compression des Données
  - Motivations et enjeux de la compression
  - Méthodes de compression sans perte
  - Méthodes de compression avec perte
  - Réduction de dimension (ACP, ICA, etc.)
  - Réduction non linéaire (t-SNE, UMAP)
  - Compression via autoencodeurs
  - Impact de la compression sur la performance des modèles
- Réglages des Modèles

- Choix des hyperparamètres clés
- Validation croisée (cross-validation)
- Recherche exhaustive (grid search)
- Recherche aléatoire (random search)
- Optimisation bayésienne
- Réglages spécifiques aux modèles
- Évaluation après réglages
- Gestion du sur-apprentissage via tuning

5 L'Apprentissage d'Ensembles

- Principes de l'Apprentissage d'Ensemble
  - Définition et motivation
  - Réduction de la variance
  - Réduction du biais
  - Combinaison de plusieurs modèles faibles
  - Méthodes de vote et d'agrégation
  - Bagging vs Boosting
  - Contraintes, limites et considérations pratiques
- Les Forêts Aléatoires
  - Concepts clés du bagging
  - Construction d'arbres aléatoires

- Hypothèses du modèle linéaire
- Erreurs, bruit et distribution des résidus
- Biais, variance et capacité du modèle

- Exploration des Données Régressives
  - Analyse exploratoire des variables
  - Relations linéaires et non linéaires
  - Corrélations et multicolinéarité
  - Visualisations pour la régression
  - Détection des points influents
  - Analyse des résidus
  - Transformations des variables
- Performance des Modèles de Régression
  - Fonctions de perte pour la régression
  - Mesures d'évaluation : MSE, RMSE, MAE
  - Coefficient de détermination (R^2)
  - Validation croisée pour la régression
  - Diagnostic des erreurs
  - Sur-apprentissage et sous-apprentissage
  - Comparaison entre modèles

7 Le Clustering

- Principes du Clustering
  - Définition et motivations
  - Apprentissage non supervisé
  - Notion de similarité et de distance
  - Types de structures de clusters
  - Prétraitements indispensables
  - Évaluation du clustering
  - Applications pratiques
- Les k-moyens (k-Means)
  - Principe général
  - Initialisation des centroïdes
  - Assignation des points aux clusters
  - Mise à jour des centroïdes
  - Critères d'arrêt
  - Choix du nombre de clusters (méthode du coude, silhouette)
  - Variantes du k-Means (k-Means++, MiniBatch, etc.)
  - Avantages, limites et cas d'usage
- Le Clustering Hiérarchique
  - Approche agglomérative
  - Approche divisive
  - Métriques de distance inter-clusters

- Dendrogrammes et interprétation
- Choix du nombre de partitions
- Complexité et limites
- Applications spécifiques du clustering hiérarchique

L'Approche DBSCAN

- Motivation : clusters denses vs bruit
- Définition des concepts (eps, minPts)
- Points coeur, bordure et bruit
- Algorithme détaillé
- Choix des paramètres et sensibilité
- Avantages et limites
- Comparaison avec K-means et clustering hiérarchique

8 Le Deep Learning

Principes Généraux du Deep Learning

- Définition et distinction avec le Machine Learning
- Réseaux de neurones comme modèles paramétriques
- Apprentissage par descente de gradient
- Représentations hiérarchiques
- Notion de profondeur
- Applications modernes du Deep Learning

- Le Perceptron
  - Modèle et intuition
  - Fonction d'activation
  - Règle d'apprentissage du perceptron
  - Cas linéairement séparables
  - Limites du perceptron simple
  - Extension au perceptron multicouche
- Réseaux Neuronaux Multicouches (MLP)
  - Architecture d'un MLP
  - Propagation avant (forward pass)
  - Fonctions d'activation modernes
  - Rétropropagation du gradient
  - Coût et optimisation
  - Sur-apprentissage et régularisation
  - Applications du MLP
- Réseaux Neuronaux Convolutifs (CNN)
  - Motivation : données spatiales et images
  - Opération de convolution
  - Couches de pooling
  - Architecture typique (Conv -> Pool -> Dense)
  - CNN modernes : VGG, ResNet, Inception

- Normalisation et régularisation dans les CNN
- Applications : vision, traitement vidéo

- Réseaux Neuronaux Récurrifs (RNN)
  - Principe des dépendances temporelles
  - Structure récurrente : états cachés
  - Problèmes de gradients explosifs/qui disparaissent
  - LSTM et GRU : architectures avancées
  - Séquences, séries temporelles et NLP
  - Applications : texte, audio, signaux
  - Comparaison RNN vs CNN vs MLP

** Introduction **

Présentation Générale

Définition et contexte historique

- La **Data Science** est une discipline intégrant statistiques, informatique, mathématiques et expertise métier afin d'extraire de la valeur à partir des données.
- L'Intelligence Artificielle (IA) a émergé dans les années 1950, notamment avec Alan Turing et les premiers algorithmes de résolution symbolique.
- L'explosion du volume des données dans les années 2000 a mené à la naissance formelle du **Big Data** et de la **Data Science moderne**.
- L'essor du Deep Learning après 2012 (victoire d'AlexNet) a transformé les capacités de traitement images/sons/texte.
- Aujourd'hui, la Data Science et l'IA constituent des briques essentielles dans les systèmes numériques contemporains.

Évolution des besoins industriels

- Les entreprises génèrent aujourd'hui des quantités massives de données (IoT, web, mobile, réseaux sociaux).
- Les besoins sont passés du simple reporting vers des systèmes **prédictifs** et **automatisés**.
- Les organisations recherchent désormais des profils capables de :
  - structurer et manipuler les données,
  - développer des modèles de Machine Learning,
  - déployer des solutions scalables (cloud, Big Data),
  - interpréter les résultats pour guider la stratégie.
- L'automatisation et les systèmes autonomes (véhicules, robots, agents conversationnels) renforcent ces besoins.

Place de la Data Science

- La Data Science constitue un pont entre :
  - les infrastructures Big Data,
  - les systèmes applicatifs,
  - les algorithmes d'IA.
- Elle joue un rôle clé dans :
  - l'optimisation des processus industriels,
  - la personnalisation de l'expérience utilisateur,
  - la cybersécurité,
  - la maintenance prédictive,
  - l'aide à la décision.
- La Data Science fournit les fondations des produits numériques modernes :
  - moteurs de recommandation,
  - systèmes de prévision,
  - plateformes d'analyse en temps réel.

Rôle de l'ingénieur et du data scientist

- Le rôle de l'ingénieur IA / data scientist combine :
  - l'analyse des besoins métiers,
  - la préparation et la transformation des données,
  - la construction d'algorithmes prédictifs,
  - la mise en production des modèles.
- Son travail implique une compréhension avancée des mathématiques : probabilités, optimisation, statistiques.
- L'ingénieur IA doit aussi maîtriser les outils modernes :
  - Python, Pandas, NumPy, scikit-learn,
  - TensorFlow / PyTorch,
  - Spark, SQL, infrastructures cloud.

Applications modernes et enjeux

- Les applications modernes de la Data Science et de l'IA couvrent :
  - la santé (diagnostic assisté),
  - la finance (détection de fraude),
  - les transports (véhicules autonomes),
  - la logistique (optimisation des flux),
  - la traduction automatique,
  - les systèmes conversationnels.
- Les enjeux actuels incluent :
  - l'éthique et la transparence des modèles,
  - la gestion de la vie privée (RGPD),
  - l'explicabilité des décisions algorithmiques,
  - la robustesse et les attaques adversariales,
  - l'impact environnemental des modèles géants.
- L'IA doit être maîtrisée, auditée et intégrée de manière responsable.

Compétences techniques en Data Science

- Les compétences techniques regroupent l'ensemble des outils, technologies et environnements nécessaires pour mener à bien un projet data.
- Elles incluent la maîtrise des systèmes, des environnements cloud, des outils de versioning, et des pipelines de traitement.
- Une bonne compréhension des architectures distribuées est également essentielle à l'ère du Big Data.

Compétences mathématiques essentielles

- La Data Science repose sur des fondements mathématiques solides : algèbre linéaire, calcul matriciel, optimisation.
- La statistique et les probabilités sont indispensables pour construire, valider et interpréter des modèles prédictifs.
- La compréhension des gradients, dérivées partielles et fonctions convexes est essentielle pour maîtriser l'apprentissage automatique.

```python
# Example: Simple computation with matrices
import numpy as np

A = np.array([[1, 2], [3, 4]])
v = np.array([1, 1])
print("Matrix-vector product:", A @ v)
```

Compétences en programmation

- La programmation est au coeur du métier : Python est la norme industrielle grâce à ses bibliothèques scientifiques.
- Le data scientist doit être capable d'écrire un code propre, modulaire, testé, et optimisé.
- La maîtrise des structures de données, de la complexité algorithmique et de la POO permet de créer des pipelines robustes.

Communication et visualisation des données

- Le data scientist doit présenter des résultats de manière claire, synthétique et convaincante.
- Les compétences en dataviz permettent de transformer des données brutes en insights directement exploitables.
- Les outils comme Matplotlib, Seaborn, Plotly, et Power BI sont incontournables pour illustrer les résultats.

Compétences en gestion de données

- Manipulation de données structurées (SQL) et non structurées (JSON, logs, images, textes).
- Nettoyage, validation, transformation et normalisation des datasets.
- Connaissance des ETL, des data warehouses et des systèmes distribués comme Hadoop ou Spark.

Compétences en modélisation

- Construction, entraînement et validation de modèles de Machine Learning.
- Compréhension des notions de biais, variance, surapprentissage et régularisation.
- Capacité à construire des modèles explicables (XAI) et à interpréter les résultats.

Compétences en déploiement et MLOps

- Maitrise des pipelines CI/CD pour déployer des modèles en production.
- Connaissance des outils : Docker, Kubernetes, MLflow, DVC.
- Automatisation du monitoring, versionning des données et des modèles.

Définition et périmètre de la Data Science

- La Data Science est un domaine multidisciplinaire qui combine statistiques, informatique et connaissances métier pour extraire de la valeur à partir des données.
- Son périmètre englobe la collecte, le stockage, la transformation, l'analyse et l'exploitation des données.
- Elle vise à produire des insights, des modèles prédictifs et des outils décisionnels adaptés à des contextes réels.
- Elle constitue un pilier essentiel de la transformation numérique des entreprises.

Cycle de vie d'un projet Data Science

- Un projet suit un cycle composé de plusieurs étapes : définition du problème, collecte des données, exploration, modélisation et déploiement.
- Ce cycle est itératif : les modèles sont ajustés en fonction des résultats, et les données peuvent nécessiter d'être rééchantillonnées.
- La documentation et la reproductibilité du processus sont essentielles pour l'industrialisation.

Collecte et prétraitement des données

- La collecte consiste à récupérer des données issues de différentes sources : bases SQL, APIs, fichiers plats, logs, streaming.
- Le prétraitement inclut le nettoyage, la détection des anomalies, la gestion des valeurs manquantes et la normalisation.
- Cette étape représente souvent 60 à 80% du temps d'un projet Data Science.

Analyse exploratoire (EDA)

- L'Analyse Exploratoire des Données (EDA) permet de comprendre la structure, la distribution et les relations entre variables.
- Elle révèle les corrélations, tendances, valeurs aberrantes et motifs cachés susceptibles d'influencer la modélisation.
- Techniques courantes : statistiques descriptives, corrélations, histogrammes, PCA, clustering exploratoire.

Visualisation et interprétation

- La visualisation facilite l'identification des patterns et la communication des résultats aux parties prenantes.
- Des outils comme Matplotlib, Seaborn ou Plotly permettent d'explorer rapidement les relations entre variables.
- L'interprétation est indispensable pour contextualiser les résultats et guider la prise de décision.

Modélisation et expérimentation

- La modélisation consiste à utiliser des algorithmes de Machine Learning pour prédire, classifier ou détecter des patterns.
- L'expérimentation implique l'ajustement des hyperparamètres, la validation croisée et l'étude de la performance.
- Les modèles doivent être évalués selon des métriques adaptées : précision, RMSE, AUC, F1-score, etc.

Déploiement et suivi des modèles

- Après validation, les modèles doivent être déployés dans un environnement réel : API, microservices, batch, ou edge computing.
- Le suivi inclut la surveillance des performances, la détection du drift et la réentraînement automatique.
- Les solutions modernes utilisent Docker, Kubernetes, MLflow, DVC ou SageMaker pour industrialiser tout le pipeline.

Définition générale du Machine Learning

- Le **Machine Learning** (ML) est une branche de l'IA qui permet aux systèmes d'apprendre à partir de données sans être explicitement programmés.
- Les modèles ML construisent des fonctions prédictives ou descriptives basées sur des patterns extraits des données.
- L'objectif est d'automatiser la prise de décision, la classification, la prédiction ou la détection d'anomalies.

Catégories d'apprentissage

- **Apprentissage supervisé** : les données d'entrée sont associées à des étiquettes. Objectif : prédire la sortie à partir de nouvelles entrées.
- **Apprentissage non supervisé** : seules les données d'entrée sont disponibles. Objectif : identifier des structures ou regroupements (clusters) dans les données.
- **Apprentissage semi-supervisé** : mélange de données étiquetées et non étiquetées.
- **Apprentissage par renforcement** : apprentissage basé sur des récompenses et punitions en fonction des actions d'un agent dans un environnement.

Supervisé vs non supervisé

- **Supervisé :** classification (labels discrets) ou régression (valeurs continues).
- **Non supervisé :** clustering, réduction de dimensionnalité (PCA, t-SNE), détection d'anomalies.
- **Exemple pratique :**

Apprentissage par renforcement (RL)

- Un agent interagit avec un environnement et apprend à maximiser une récompense cumulée.
- Composants clés : état, action, récompense, politique, fonction de valeur.
- Utilisé dans les jeux, la robotique, l'optimisation de processus.

Pipeline d’un modèle ML

- Les étapes classiques d’un pipeline ML :
  1. Collecte et prétraitement des données
  2. Séparation train/test
  3. Sélection de caractéristiques
  4. Entraînement du modèle
  5. Évaluation et validation
  6. Optimisation et réglage des hyperparamètres
  7. Déploiement et suivi

Évaluation et métriques de performance

- Les métriques dépendent du type de problème :
  - Classification : accuracy, precision, recall, F1-score, ROC-AUC
  - Régression : RMSE, MAE, R^2
  - Clustering : silhouette score, Davies-Bouldin
- La validation croisée permet d'estimer la performance sur des données non vues.

Problèmes courants et biais des modèles

- Surapprentissage (overfitting) : le modèle apprend le bruit au lieu des patterns.
- Sous-apprentissage (underfitting) : le modèle est trop simple pour capturer la complexité des données.
- Biais dans les données : déséquilibre des classes, données manquantes ou bruitées.
- Biais algorithmique : choix de modèle inadéquat ou hyperparamètres mal ajustés.

Définition et caractéristiques du Big Data

- Le **Big Data** désigne des ensembles de données trop volumineux, variés ou rapides pour être traités par des systèmes traditionnels.
- Les caractéristiques sont souvent décrites par les **3V** :
  - **Volume** : grande quantité de données
  - **Vélocité** : rapidité de génération et traitement
  - **Variété** : données structurées, semi-structurées, non structurées
- Extensions possibles : **5V** ou plus, incluant **Véracité** (qualité des données) et **Valeur** (utilité des données).

Écosystèmes technologiques du Big Data

- Hadoop : HDFS pour le stockage distribué, MapReduce pour le traitement.
- Spark : traitement distribué en mémoire, support batch et streaming.
- Bases NoSQL : MongoDB, Cassandra, HBase pour les données non relationnelles.
- Outils d'orchestration et ETL : Airflow, NiFi.

Stockage distribué et systèmes massifs

- Les données sont stockées sur plusieurs machines pour assurer scalabilité et tolérance aux pannes.
- HDFS : fichiers fragmentés et répliqués sur plusieurs noeuds.
- Systèmes massifs nécessitent des techniques de partitionnement, sharding et réplication.

Traitement distribué : batch et streaming

- **Batch** : traitement de grandes quantités de données accumulées (ex. MapReduce, Spark batch).
- **Streaming** : traitement en temps réel des données générées continuellement (ex. Spark Streaming, Kafka).
- L'approche distribuée permet de paralléliser les calculs et d'optimiser les performances.

Formats de données massives

- Données **structurées** : CSV, Parquet, ORC
- Données **semi-structurées** : JSON, XML, Avro
- Données **non structurées** : images, vidéos, logs, textes libres
- Le choix du format impacte la compression, l'accès et la vitesse de traitement.

Applications industrielles du Big Data

- **Finance** : détection de fraude, scoring de crédit
- **Santé** : analyse d'images médicales, suivi des patients
- **Marketing** : recommandation de produits, ciblage publicitaire
- **Transport** : optimisation des itinéraires, maintenance prédictive
- **Industrie 4.0** : surveillance des machines, analyse prédictive de la production

Définition et lien avec le Machine Learning

- Le **Deep Learning (DL)** est une sous-branche du Machine Learning qui utilise des réseaux de neurones profonds pour apprendre des représentations hiérarchiques des données.
- Contrairement au ML classique, le DL peut automatiquement extraire des features complexes à partir de données brutes.
- Il est particulièrement efficace pour les données non structurées : images, texte, audio, vidéo.

Réseaux de neurones artificiels (ANN)

- Les ANN sont composés de couches de neurones : couche d'entrée, couches cachées, couche de sortie.
- Chaque neurone effectue une combinaison linéaire suivie d'une fonction d'activation non-linéaire.
- Les ANN sont utilisés pour la classification, la régression et la détection de patterns.

Architectures profondes

- **MLP (Multi-Layer Perceptron)** : couches entièrement connectées.
- **CNN (Convolutional Neural Networks)** : extraction de features locales pour les images.
- **RNN (Recurrent Neural Networks)** : traitement des séquences et séries temporelles.
- **Transformers** : attention et auto-attention pour NLP et multimodalité.

Entraînement et optimisation

- L'entraînement consiste à ajuster les poids via **backpropagation** et **descente de gradient**.
- Hyperparamètres importants : learning rate, batch size, nombre d'époques, fonctions d'activation.
- Techniques pour améliorer la performance : régularisation, dropout, normalisation, early stopping.

Applications en vision par ordinateur

- Classification d'images, détection d'objets, segmentation d'images.
- Reconnaissance faciale et analyse biométrique.
- Conduite autonome : détection et suivi des objets sur route.

Applications en NLP (Natural Language Processing)

- Classification de textes, analyse de sentiments, résumé automatique.
- Traduction automatique et génération de texte.
- Chatbots et assistants intelligents.

Applications audio, séries temporelles et multimodalité

- Audio : reconnaissance vocale, classification de sons.
- Séries temporelles : prévision financière, analyse de capteurs IoT.
- Multimodalité : fusion de texte, image, audio pour applications avancées.

Concept d'apprentissage à partir de données

- L'apprentissage machine consiste à construire des modèles capables de **prédire ou décrire** un phénomène à partir de données historiques.
- Les modèles identifient des **patterns** et relations dans les données.
- Plus de données pertinentes conduisent généralement à de meilleures performances.

Notion de généralisation

- La généralisation mesure la capacité d'un modèle à **effectuer de bonnes prédictions sur de nouvelles données non vues**.
- **Overfitting** : excellent sur les données d'entraînement mais mauvais sur données nouvelles.
- **Underfitting** : modèle trop simple, performance faible même sur l'entraînement.

Rôle des données et de leurs distributions

- La qualité et la quantité de données influencent directement les performances des modèles.
- La distribution des données doit être représentative de l'environnement réel.
- Données biaisées ou déséquilibrées peuvent induire des modèles injustes ou inefficaces.

Paradigmes modernes d'entraînement

- **Batch learning** : modèle entraîné sur l'ensemble des données d'un coup.
- **Online learning** : modèle mis à jour progressivement avec de nouvelles données.
- **Self-supervised learning** : apprentissage avec supervision implicite à partir de grandes quantités de données non étiquetées.
- **Few-shot / zero-shot** : modèles capables de généraliser à partir de très peu ou aucune donnée annotée.

Importance de la validation et du test

- Validation : permet d'ajuster les hyperparamètres et prévenir l'overfitting.
- Test : évaluer la performance finale sur des données totalement nouvelles.
- Techniques : hold-out, k-fold cross-validation, stratified sampling.

Vers le self-supervised learning

- Technique qui crée automatiquement des tâches de supervision à partir des données brutes.
- Permet d'utiliser de grandes quantités de données non annotées pour préformer des modèles.
- Exemples : masquage de mots en NLP (BERT), prédiction de parties manquantes en vision (MAE).

Tendances actuelles et perspectives

- Modèles géants pré-entraînés (GPT, BERT, PaLM) et fine-tuning spécifique.
- IA multimodale : fusion de texte, image, audio, vidéo.
- AutoML : automatisation de la construction et du réglage des modèles.
- Éthique et biais : fairness, explicabilité et régulation de l'IA.

** Les fondamentaux **

Problématique de prédiction

- Le Machine Learning vise à résoudre des problèmes où l'on cherche à **prédire une valeur future ou inconnue** à partir d'observations.
- Deux grandes familles :
  - **Prédiction de classes** (classification)
  - **Prédiction de valeurs continues** (régression)
- La problématique se formule comme la recherche d'une fonction qui relie des entrées à une sortie.
- L'apprentissage repose sur l'analyse de données historiques pour construire cette fonction.

Notion d'entrée et de sortie

- Chaque observation est décrite par :
  - des **entrées** ou **features** (variables explicatives)
  - une **sortie** ou **label** (variable cible)
- Les entrées peuvent être :
  - numériques (valeurs continues)
  - catégorielles (valeurs discrètes)
  - textuelles, visuelles, temporelles, etc.
- La sortie dépend du type de tâche :
  - classe : chat / chien / oiseau
  - valeur : prix, température, durée
  - structure : séquence, segmentation, traduction

Représentation mathématique d'un modèle

- Un modèle de Machine Learning est une fonction mathématique :

$$\hat{y} = f_\theta(x)$$

où :

  - x représente les entrées,
  - $\hat{y}$ est la prédiction,
  - θ est l'ensemble des paramètres du modèle (poids, biais, etc.).

- L'apprentissage correspond à la recherche des paramètres θ qui minimisent une **fonction de perte**.
- L'objectif est d'obtenir la meilleure approximation possible de la fonction réelle qui relie x à y.

Structure générale d'un système d'apprentissage

- Un système de Machine Learning suit généralement la structure suivante :
  - **Acquisition et préparation des données** : nettoyage, transformation, normalisation.
  - **Séparation des données** : entraînement, validation, test.
  - **Choix du modèle** : linéaire, arborescent, probabiliste, profond, etc.
  - **Entraînement** : optimisation des paramètres.
  - **Évaluation** : métriques adaptées au problème.
  - **Déploiement et monitoring**.
- Le système est cyclique : il se réévalue et s'améliore continuellement en fonction des données.

Lien avec l'approche statistique

- Le Machine Learning et la statistique partagent un objectif commun : **modéliser les relations entre variables**.
- La statistique s'intéresse davantage à :
  - l'interprétation,
  - l'inférence,
  - la significativité des paramètres.
- Le Machine Learning met l'accent sur :
  - la performance prédictive,
  - la capacité de généralisation,
  - la scalabilité et l'adaptation à des données massives.
- Les deux disciplines sont complémentaires et se rejoignent dans l'estimation de modèles probabilistes.

Définition des variables explicatives

- Les variables prédictives, aussi appelées **variables explicatives** ou **features**, sont les informations utilisées par un modèle pour effectuer une prédiction.
- Elles représentent les caractéristiques observées d'un phénomène ou d'un individu.
- Elles constituent l'entrée du modèle dans le processus d'apprentissage supervisé.
- Leur qualité, pertinence et représentativité influencent fortement la performance du modèle.
- Une bonne définition des variables explicatives permet d'améliorer la robustesse et la généralisation des modèles.

Types de variables prédictives

- On distingue plusieurs types de variables :
  - **Variables numériques** : continues (taille, prix) ou discrètes (nombre d'enfants).
  - **Variables catégorielles** : nominales (couleur), ordinales (niveau d'étude).
  - **Variables textuelles** : documents, messages, descriptions.
  - **Variables temporelles** : séries chronologiques, horodatages.
  - **Variables géospatiales** : coordonnées, zones géographiques.
  - **Variables multimodales** : images, sons, signaux.
- Chaque type de variable nécessite un traitement et une représentation adaptés.

Codage et représentation

- Le codage transforme les variables brutes en une forme exploitable par un modèle.
- Méthodes courantes :
  - **Normalisation / standardisation** pour les variables numériques.
  - **Encodage catégoriel** : one-hot, ordinal, target encoding.
  - **Vectorisation du texte** : bag-of-words, TF-IDF, embeddings.
  - **Transformation temporelle** : extraction de tendances, saisonnalité.
  - **Représentations d'images** : pixels, filtres, embeddings de CNN.
- Une bonne représentation améliore l'efficacité des modèles et réduit la complexité du problème.

Sélection et importance des variables

- Tous les attributs ne contribuent pas de manière égale à la prédiction.
- Objectifs de la sélection des variables :
  - réduire la dimensionnalité,
  - éliminer le bruit et les redondances,
  - améliorer la performance et la vitesse d'entraînement,
  - faciliter l'interprétation.
- Méthodes principales :
  - **Sélection par filtres** : corrélations, tests statistiques.
  - **Sélection par wrapper** : recherche séquentielle, élimination récursive.
  - **Méthodes embarquées** : régularisation, arbres de décision, modèle linéaire.
- Importance des variables : mesure quantitative de la contribution de chaque feature.

Traitement et transformation des variables

- Le traitement des variables vise à préparer les données avant la modélisation.
- Opérations courantes :
  - **Gestion des valeurs manquantes** : imputation, suppression.
  - **Détection et traitement des outliers**.
  - **Transformation** : logarithme, racine carrée, puissance.
  - **Binarisation ou discrétisation** des variables continues.
  - **Création de nouvelles variables** (feature engineering).
- Une transformation pertinente peut révéler des relations non linéaires utiles à l'apprentissage.

Définition de la variable cible

- La **variable à prédire**, aussi appelée *variable cible* ou *output*, représente ce que le modèle cherche à estimer.
- Elle peut être :
  - une **valeur numérique**,
  - une **classe**,
  - une **structure complexe** (séquence, vecteur, label multiple).
- La nature de cette variable détermine :
  - le type de tâche (régression, classification, etc.),
  - la fonction de perte,
  - les métriques d'évaluation,
  - les modèles adaptés.
- Comprendre la variable cible est essentiel pour bien formuler un problème d'apprentissage supervisé.

Variable continue vs variable catégorielle

- On distingue deux grandes familles de variables à prédire :
- **Variables continues**
  - valeurs numériques sur un espace continu,
  - associées aux tâches de régression,
  - sensibles aux outliers et à la distribution.
- **Variables catégorielles**
  - valeurs discrètes représentant des classes,
  - associées aux tâches de classification,
  - nécessitent parfois un encodage adapté ou un calibrage.
- Le type de variable conditionne profondément le modèle à utiliser.

Structure des espaces de sortie

- L'espace de sortie correspond à l'ensemble des valeurs possibles de la variable cible.
- Différentes structures possibles :
  - **Espace continu** : réels ou vecteurs continus.
  - **Espace discret fini** : classes en classification.
  - **Espace ordonné** : classification ordinale.
  - **Espace multidimensionnel** : plusieurs sorties simultanées.
- La structure de l'espace de sortie influence :
  - le modèle,
  - l'architecture (dans les réseaux neurones),
  - le choix de la fonction de perte.

Problèmes de régression

- Utilisés lorsque la variable cible est **continue**.
- Objectif : apprendre une fonction approchant une grandeur réelle.
- Exemples de cibles :
  - prix, température, durée, taux de conversion.
- Spécificités :
  - sensibilité aux valeurs extrêmes,
  - importance de l'homoscédasticité,
  - métriques dédiées (MSE, RMSE, MAE).

Problèmes de classification

- Utilisés lorsque la variable cible est **catégorielle**.
- Objectif : assigner une observation à une classe parmi un ensemble fini.
- Exemples de cibles :
  - spam / non spam,
  - type d'objet sur une image,
  - diagnostic médical.
- Particularités :
  - gestion du déséquilibre des classes,
  - choix de métriques adaptées (F1-score, accuracy, AUC),
  - interprétation probabiliste des prédictions.

Cas particuliers : multilabel, multiclasse, ordinal

- Plusieurs extensions de la classification existent :
- **Classification multiclasse**
  - plus de deux classes,
  - une seule classe possible par observation.
- **Classification multilabel**
  - plusieurs labels possibles simultanément,
  - structure de sortie vectorielle binaire.
- **Classification ordinale**
  - les classes sont ordonnées,
  - nécessite des modèles ou pertes adaptés à l'ordre.
- Ces cas particuliers imposent des architectures et méthodes d'entraînement spécifiques.

Définition et rôle

- Une **fonction hypothèse** (ou *hypothesis function*) est la fonction utilisée par le modèle pour faire des prédictions.
- Notée généralement $h_\theta(x)$, elle associe :
  - une observation x,
  - à une prédiction $h_\theta(x)$.
- Rôle central :
  - définir la famille de modèles possibles,
  - encadrer la capacité d'apprentissage,
  - influencer la complexité et la performance.
- La qualité de la fonction hypothèse conditionne la performance finale.

Paramétrisation d'un modèle

- Une fonction hypothèse dépend de paramètres notés θ :
  - coefficients d'un modèle linéaire,
  - poids et biais d'un réseau de neurones,
  - paramètres d'un noyau ou d'une transformation.
- L'objectif de l'apprentissage est de **trouver les paramètres optimaux** $\theta^\star$.
- Ces paramètres sont ajustés pour minimiser une fonction de coût.
- La paramétrisation définit :
  - la structure du modèle,
  - son expressivité,
  - son comportement en généralisation.

Fonctions hypothèses linéaires

- Une fonction hypothèse est dite **linéaire** lorsqu'elle combine les variables prédictives de manière linéaire :

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

- Caractéristiques :
  - simple à entraîner,
  - interprétable,
  - bien adaptée lorsque les relations sont linéaires,
  - sensible au sur-apprentissage en haute dimension.
- Exemples :
  - régression linéaire,
  - SVM linéaire,
  - régression logistique (fonction linéaire transformée par une sigmoïde).

Fonctions hypothèses non linéaires

- Les fonctions hypothèses non linéaires permettent de modéliser des relations complexes.
- Exemples de non-linéarités :
  - polynômes,
  - fonctions noyau (kernel trick),
  - réseaux de neurones (ReLU, tanh, etc.),
  - arbres de décision.
- Avantages :
  - grande expressivité,
  - capacité d'approximation élevée.
- Inconvénients :
  - risque de sur-apprentissage,
  - optimisation plus complexe,
  - interprétabilité souvent réduite.

Capacité du modèle et approximation

- La capacité d'un modèle désigne sa **capacité à représenter des fonctions complexes**.
- Une faible capacité entraîne :
  - un sous-apprentissage (*underfitting*),
  - une mauvaise approximation des données.
- Une capacité trop élevée entraîne :
  - un sur-apprentissage (*overfitting*),
  - une mauvaise généralisation.
- L'objectif est de trouver un **juste équilibre** :
  - modèle suffisamment expressif,
  - mais pas trop complexe.
- Lié à la notion de biais-variance (développée dans une autre section).

Hypothèses et espace des hypothèses

- L'**espace des hypothèses** (noté $\mathcal{H}$) représente l'ensemble des fonctions possibles que le modèle peut apprendre.
- Exemples d'espaces d'hypothèses :
  - modèles linéaires,
  - arbres de décision de profondeur $\leq d$,
  - réseaux de neurones d'une architecture donnée.
- Plus $\mathcal{H}$ est large, plus le modèle peut apprendre des structures complexes.
- Mais un espace trop vaste augmente le risque de sur-apprentissage.
- L'apprentissage consiste à choisir, dans $\mathcal{H}$, l'hypothèse qui minimise la fonction de perte sur les données.

Notion d'estimateur en statistique

- Un **estimateur** est une fonction qui, à partir d'un échantillon de données, fournit une estimation d'un paramètre inconnu.
- Exemple de paramètres :
  - moyenne réelle μ,
  - variance réelle σ^2,
  - paramètres d'un modèle θ.
- Un estimateur est une **variable aléatoire** car il dépend des données observées.
- Ses propriétés théoriques déterminent sa qualité : efficacité, cohérence, biais, variance.
- En apprentissage automatique, chaque modèle appris constitue un **estimateur** des relations dans les données.

Définition du biais

- Le **biais** d'un estimateur mesure son **écart systématique** par rapport à la valeur réelle du paramètre.

$$\text{Biais}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$$

- Un estimateur est :
  - **sans biais** si son espérance est égale au paramètre réel,
  - **biaisé** sinon.
- Un biais élevé indique une erreur systématique due à :
  - un modèle trop simple,
  - des hypothèses trop restrictives,
  - une erreur de spécification.
- En machine learning : sous-apprentissage = biais élevé.

Définition de la variance

- La **variance** d'un estimateur mesure sa **sensibilité aux fluctuations des données**.

$$\text{Var}(\hat{\theta}) = \mathbb{V}[\hat{\theta}]$$

- Un estimateur à forte variance :
  - varie beaucoup d'un échantillon à l'autre,
  - a du mal à généraliser,
  - reflète un modèle trop complexe.
- En machine learning : sur-apprentissage = variance élevée.
- L'enjeu est de trouver un compromis entre **biais faible** et **variance faible**.

Erreur quadratique moyenne (MSE)

- La MSE résume l'exactitude globale d'un estimateur :

$$\text{MSE}(\hat{\theta}) = \mathbb{E}\left[(\hat{\theta} - \theta)^2\right]$$

- Décomposition fondamentale :

$$\text{MSE} = \text{Biais}^2 + \text{Variance} + \text{Bruit irréductible}$$

- Cette décomposition explique :
  - l'équilibre entre biais et variance,
  - les erreurs inévitables liées au bruit,
  - pourquoi complexifier un modèle peut augmenter la MSE.
- C'est une mesure essentielle pour évaluer la qualité d'un modèle.

Comportement asymptotique des estimateurs

- En statistique, on étudie le comportement des estimateurs quand la taille de l'échantillon n augmente.
- Propriété clé : la **consistance**.
  - Un estimateur est **consistent** si $\hat{\theta} \to \theta$ quand $n \to \infty$.
  - Cela implique : biais $\to 0$ et variance $\to 0$.
- D'autres propriétés asymptotiques :
  - normalité asymptotique,
  - efficacité asymptotique,
  - convergence en probabilité ou en loi.
- En pratique, les modèles ML se comportent mieux avec davantage de données — mais pas toujours linéairement.

Impact du bruit et des données

- Le bruit représente la part d'imprévisibilité ou de variabilité non modélisable dans les données.
- Même un estimateur parfait ne peut pas prédire ce bruit : c'est le **bruit irréductible**.
- Impacts majeurs :
  - augmente la MSE minimale atteignable,
  - rend l'apprentissage plus difficile,
  - nécessite des modèles robustes.
- La distribution des données influence :
  - la variance de l'estimateur,
  - la difficulté de la tâche,
  - l'adéquation du modèle choisi.
- Les modèles sont souvent très sensibles aux données rares, déséquilibrées ou bruitées.

Origine du compromis biais–variance

- L'erreur totale d'un modèle supervisé se décompose en trois parties :

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

- Le **biais** augmente lorsque le modèle est trop simple pour capturer les relations réelles.
- La **variance** augmente lorsque le modèle est trop flexible et apprend trop les fluctuations du dataset.
- Une amélioration d'un terme entraîne souvent la dégradation de l'autre.
- Le compromis biais–variance est donc une **tension structurelle** entre simplicité et complexité.

Compréhension du sur-apprentissage

- Le **sur-apprentissage (overfitting)** survient lorsque le modèle apprend trop fidèlement l'échantillon d'entraînement.
- Symptômes typiques :
  - faible erreur sur le training set,
  - forte erreur sur le test set,
  - forte sensibilité aux données,
  - variance élevée.
- Causes principales :
  - modèle trop complexe,
  - manque de données,
  - bruit important dans les observations.
- Un modèle en sur-apprentissage généralise mal et nécessite une régularisation ou davantage de données.

Compréhension du sous-apprentissage

- Le **sous-apprentissage (underfitting)** survient lorsque le modèle est trop simple pour capturer la structure réelle.
- Symptômes :
  - forte erreur en entraînement,
  - forte erreur en test,
  - biais élevé.
- Causes principales :
  - modèle trop rigide (linéaire pour un problème non linéaire),
  - absence de features pertinentes,
  - entraînement insuffisant,
  - prétraitement inadéquat.
- Le remède implique généralement d'augmenter la complexité ou d'améliorer les données.

Visualisation et interprétation

- Le compromis biais–variance est souvent représenté graphiquement :
- Interprétation du graphique :
  - augmenter la complexité diminue le biais mais augmente la variance,
  - réduire la complexité augmente le biais mais diminue la variance.
- L'objectif est de trouver la zone optimale, souvent appelée **sweet spot**.
- Cette visualisation aide à comprendre pourquoi un modèle simple ou trop complexe donne de mauvaises performances.

Effet de la complexité du modèle

- La complexité d'un modèle correspond à sa capacité d'approximation :
  - modèles simples : régression linéaire, k-NN avec k élevé,
  - modèles complexes : réseaux profonds, forêts aléatoires, SVM non linéaires.
- Effets observés :
  - **faible complexité** $\Rightarrow$ biais élevé, variance faible,
  - **forte complexité** $\Rightarrow$ biais faible, variance élevée.
- La recherche du modèle optimal nécessite un ajustement progressif de la complexité.
- La validation croisée permet de détecter le niveau optimal.

Impact de la taille de l'échantillon

- La quantité de données influence directement biais et variance :
  - petites données $\Rightarrow$ forte variance,
  - grandes données $\Rightarrow$ variance réduite.
- Un modèle complexe nécessite davantage d'exemples pour être correctement estimé.
- La loi des grands nombres améliore :
  - la stabilité des estimateurs,
  - la généralisation en test.
- Taille insuffisante = risque accru de sur-apprentissage.
- D'où l'importance du data augmentation ou de l'acquisition de données.

Stratégies pour équilibrer le compromis

- Plusieurs techniques permettent de trouver un compromis optimal :
  - **Régularisation** : L1, L2, Elastic Net.
  - **Validation croisée** pour ajuster le niveau de complexité.
  - **Early stopping** dans les modèles entraînés par gradient.
  - **Pruning** dans les arbres et réseaux de neurones.
  - **Ensembles de modèles** :
    - bagging (réduction de variance),
    - boosting (réduction de biais).
  - **Augmentation du dataset** : collecte, génération, augmentation artificielle.
  - **Feature engineering** pour réduire le biais structurel.
- L'objectif final est une **généralisation optimale** sur des données jamais vues.

Définition et rôle dans l'apprentissage

- Une **fonction de perte** $L(y, \hat{y})$ mesure l'écart entre la vérité terrain y et la prédiction $\hat{y}$.
- Elle fournit le **signal d'apprentissage** utilisé par l'optimiseur pour ajuster les paramètres θ.
- Dans un cadre empirique, on minimise la **perte empirique** :

$$\hat{\mathcal{L}}(\theta) \;=\; \frac{1}{n}\sum_{i=1}^{n} L\big(y_i,\, h_\theta(x_i)\big).$$

- Le choix de la perte conditionne :
  - la sensibilité aux outliers,
  - la convexité du problème d'optimisation,
  - la signification probabiliste (MLE, calibration).

Pertes pour la régression

- **MSE (Mean Squared Error)** :

$$L_{\mathrm{MSE}}(y, \hat{y}) = (y - \hat{y})^2.$$

  Avantages : convexe, différentiable, lien avec bruit gaussien.
  Inconvénient : sensible aux grosses erreurs.
- **MAE (Mean Absolute Error)** :

$$L_{\mathrm{MAE}}(y, \hat{y}) = |y - \hat{y}|.$$

  Avantages : robuste aux outliers. Inconvénient : non différentiable en 0 (sous-gradient nécessaire).
- **Huber loss** (seuil δ) : combinaison robuste

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{si } |y - \hat{y}| \le \delta, \\ \delta\,|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{sinon.} \end{cases}$$

Pertes pour la classification

- **Entropie croisée (Cross-Entropy)** — binaire :

$$L_{\mathrm{CE}}(y, \hat{p}) = -\big[y \log \hat{p} + (1-y) \log(1-\hat{p})\big],$$

où $\hat{p} = P(y = 1 \mid x)$.

- **Entropie croisée multiclasse** (softmax + log-loss) :

$$L_{\mathrm{CE}}(y, \hat{p}) = -\sum_{k=1}^{K} \mathbf{1}_{\{y=k\}} \log \hat{p}_k.$$

- **Hinge loss** (SVM) :

$$L_{\mathrm{hinge}}(y, f(x)) = \max\big(0,\, 1 - yf(x)\big),$$

avec $y \in \{-1, 1\}$ et $f(x)$ la fonction de décision.

- **Pertes calibrées** : focal loss (déséquilibre sévère), categorical focal pour atténuer l'effet des classes majoritaires.

Propriétés mathématiques importantes

- **Borne inférieure et positivité** : généralement $L \geq 0$ et $L = 0$ si et seulement si prédiction parfaite (selon la perte).
- **Convexité** : facilite la recherche d'un minimum global (ex. MSE, log-loss sont convexes pour modèles linéaires).
- **Différentiabilité** : nécessaire pour les méthodes basées sur les gradients ; sinon utiliser sous-gradients.
- **Robustesse** : résistance aux outliers (MAE > MSE).
- **Calibration probabiliste** : certaines pertes (log-loss) correspondent directement à la vraisemblance et produisent des probabilités calibrées.

Convexité et différentiabilité

- **Convexité locale/globale** : si L est convexe par rapport aux paramètres θ, alors tout minimum local est un minimum global.
- Dans les modèles linéaires, MSE et log-loss sont convexes ; dans les réseaux profonds, la perte empirique est généralement non convexe.
- **Différentiabilité** : permet le calcul explicite du gradient $\nabla_\theta \hat{\mathcal{L}}(\theta)$.
- En absence de différentiabilité (ex. MAE en 0), on utilise des **sous-gradients** ou des approximations lisses (Huber).

Gradients et rétropropagation

- L'optimisation par gradient nécessite le calcul du gradient de la perte par rapport aux paramètres :

$$\nabla_\theta \hat{\mathcal{L}}(\theta) \;=\; \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L\big(y_i, h_\theta(x_i)\big).$$

- Exemples de dérivées usuelles :
  - pour MSE : $\dfrac{\partial}{\partial \hat{y}}(y-\hat{y})^2 = -2(y-\hat{y})$,
  - pour log-loss (binaire) : $\dfrac{\partial}{\partial \hat{p}}[-y\log\hat{p} - (1-y)\log(1-\hat{p})] = -\dfrac{y}{\hat{p}} + \dfrac{1-y}{1-\hat{p}}$.
- La **rétropropagation** applique la chaîne de dérivation pour propager ces gradients vers les paramètres des couches précédentes.
- Stabilité numérique et choix d'échelles (normalisation) impactent fortement le calcul des gradients.

Comparaison entre familles de pertes

- **MSE vs MAE :**
  - MSE pénalise fortement les grosses erreurs (sensible aux outliers).
  - MAE est plus robuste mais a des gradients constants (moins informatifs pour petits écarts).
- **Log-loss (cross-entropy) vs Hinge :**
  - Log-loss fournit des probabilités calibrées ; hinge se concentre sur la marge de décision.
  - Hinge est bien adapté aux marges maximales (SVM), log-loss mieux pour probabilités.
- **Critères pratiques de choix :**
  - nature de la cible (continue vs discrète),
  - sensibilité aux outliers,
  - nécessité de probabilités calibrées,
  - contraintes d'optimisation (convexité, différentiabilité).
- En pratique, tester plusieurs pertes et valider sur jeu de validation.

Motivation et intuition

- La régularisation vise à **contrôler la complexité** du modèle pour éviter le sur-apprentissage.
- Idée générale : pénaliser les modèles aux paramètres trop grands ou trop complexes.
- Elle s'écrit comme une pénalisation ajoutée à la perte :

$$\mathcal{L}_{\text{reg}}(\theta) = \hat{\mathcal{L}}(\theta) + \lambda\,\Omega(\theta),$$

où :

- $\hat{\mathcal{L}}(\theta)$ = perte empirique,
- $\Omega(\theta)$ = terme de pénalité,
- λ = hyperparamètre contrôlant la force de la régularisation.

- Effets généraux :
  - réduction de la variance,
  - amélioration de la généralisation,
  - stabilisation de l'optimisation.

Régularisation L1

- La régularisation L1 applique une pénalité proportionnelle à la norme ℓ_1 :

$$\Omega_{\mathrm{L1}}(\theta) = \|\theta\|_1 = \sum_i |\theta_i|.$$

- Effets principaux :
  - favorise les modèles **parcimonieux** (sparse),
  - pousse certains paramètres exactement à zéro,
  - permet une **sélection automatique de variables**.
- Inconvénients :
  - optimisation plus difficile (non différentiable en 0),
  - peut conduire à des modèles instables si features corrélés.

Régularisation L2

- La régularisation L2 pénalise la norme quadratique :

$$\Omega_{\mathrm{L2}}(\theta) = \|\theta\|_2^2 = \sum_i \theta_i^2.$$

- Effets principaux :
  - réduit la magnitude des paramètres sans les annuler,
  - stabilise les modèles en présence de multicolinéarité,
  - favorise les solutions **lisses** (smooth).
- Avantages :
  - différentiable partout,
  - convexité facilitant l'optimisation,
  - largement utilisée : ridge regression, weight decay en deep learning.

Régularisation élastique (Elastic Net)

- L'Elastic Net combine L1 et L2 :

$$\Omega_{\mathrm{EN}}(\theta) = \alpha\|\theta\|_1 + (1-\alpha)\|\theta\|_2^2.$$

- Effets :
  - sélection de variables (via L1),
  - stabilité accrue pour variables corrélées (via L2),
  - contrôle flexible de la parcimonie via α.
- Applications :
  - modèles linéaires complexes,
  - datasets haute dimension $p \gg n$,
  - pré-traitement pour modèles non linéaires.

Contraintes sur les paramètres

- La régularisation peut être vue comme une **contrainte explicite** :

$$\min_{\theta} \hat{\mathcal{L}}(\theta) \quad \text{sujet à} \quad \|\theta\|_p \leq c.$$

- Exemples :
  - contraintes de norme : $\|\theta\|_2 \leq c$ (balles euclidiennes),
  - contraintes de positivité (modèles interprétables),
  - contraintes d'orthogonalité (PCA, factorisation).
- Interprétation géométrique :
  - L1 : polytope en diamant, induit parcimonie,
  - L2 : sphère, favorise solutions distribuées.

Impact sur le biais et la variance

- La régularisation modifie le compromis biais-variance :
  - **L1/L2 augmentent le biais,**
  - **mais réduisent fortement la variance.**
- Effet global :

$$\text{Erreur totale} = \text{biais}^2 + \text{variance} + \text{bruit}.$$

- Utilisation idéale lorsque :
  - dataset petit ou bruité,
  - modèle trop flexible,
  - nombreuses variables non pertinentes.
- L'objectif : **minimiser l'erreur de généralisation.**

Régularisation avancée : dropout, early stopping

- Outre L1/L2, les modèles complexes (notamment réseaux profonds) utilisent :
  - **Dropout** :
    - désactive aléatoirement une proportion de neurones,
    - évite la co-adaptation des unités,
    - agit comme un ensemble de modèles implicites.
  - **Early stopping** :
    - stoppe l'apprentissage avant sur-apprentissage,
    - suit la perte sur un **jeu de validation**,
    - agit comme une régularisation implicite similaire à L2.
  - Autres formes avancées :
    - **data augmentation**,
    - **normalisation** (BatchNorm, LayerNorm),
    - **stochastic depth** dans les modèles très profonds,
    - **weight sharing** dans les architectures convolutionnelles et transformers.
- Objectif : améliorer la **généralisation** tout en permettant un modèle expressif.

Problème d'optimisation

- L'apprentissage consiste à **minimiser une fonction de perte** $\mathcal{L}(\theta)$ par rapport aux paramètres θ.
- Formulation générale :

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta)$$

- Challenges principaux :
  - espace de paramètres de grande dimension,
  - fonctions de perte non convexes,
  - bruit dans les gradients dû aux échantillons,
  - contraintes de temps et ressources.
- Objectif : atteindre un minimum global ou un minimum local satisfaisant.

Descente de gradient

- Méthode itérative pour optimiser θ en suivant le **gradient de la perte** :

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

- η = taux d'apprentissage (learning rate).
- Avantages :
  - simple à implémenter,
  - efficace pour les problèmes différentiables.
- Limites :
  - peut converger lentement,
  - sensible aux minima locaux et plateaux,
  - choix critique du taux d'apprentissage.

Variants de la descente de gradient

- **Batch Gradient Descent** : utilise l'ensemble du dataset à chaque mise à jour.
- **Stochastic Gradient Descent (SGD)** : utilise un seul échantillon par mise à jour, plus bruité mais plus rapide.
- **Mini-batch Gradient Descent** : compromis entre batch et SGD, mise à jour sur sous-ensembles.
- Techniques additionnelles :
  - **Momentum** : accélère convergence sur pentes plates,
  - **Nesterov Accelerated Gradient** : anticipe la direction de la mise à jour.
- Choix du variant dépend de :
  - taille du dataset,
  - stabilité souhaitée,
  - capacité de parallélisation.

Optimiseurs modernes (Adam, RMSProp, etc.)

- Les optimiseurs modernes combinent :
  - **adaptation du learning rate par paramètre** (RMSProp, Adagrad),
  - **momentum** pour accélérer la convergence (Adam, AdamW).
- Avantages :
  - meilleure stabilité,
  - convergence plus rapide sur fonctions non convexes,
  - robustesse au bruit dans les gradients.
- Inconvénients :
  - hyperparamètres supplémentaires,
  - parfois moins généralisant que SGD pur.

Conditions de convergence

- Pour que l'optimisation converge :
  - taux d'apprentissage η correctement choisi,
  - gradients cohérents et estimés correctement,
  - fonction de perte différentiable (de préférence lisse),
  - absence de plateaux trop longs ou minima locaux très profonds.
- Techniques pour favoriser convergence :
  - normalisation des features,
  - annealing du learning rate,
  - early stopping basé sur validation.
- Convergence ne garantit pas nécessairement un minimum global.

Paysages de perte et minima locaux

- La fonction de perte définit un **paysage dans l'espace des paramètres**.
- Minima locaux :
  - points où le gradient est nul mais la perte n'est pas minimale globalement,
  - fréquents dans les réseaux profonds.
- Plateaux et saddle points ralentissent convergence.
- Stratégies pour éviter pièges :
  - random initialization,
  - momentum et accélérateurs,
  - batch/stochastic updates.

Techniques d'accélération

- Améliorer vitesse et stabilité de convergence :
  - **Momentum** et **Nesterov** pour dépasser plateaux,
  - **Learning rate scheduling** : step decay, cosine annealing,
  - **Batch normalization** pour gradients stables,
  - **Gradient clipping** pour éviter explosions,
  - **Optimiseurs adaptatifs** : Adam, RMSProp.
- Objectif : convergence rapide vers minima acceptables tout en maintenant la généralisation.

** La classification **

Définition du problème de classification

- La classification est une tâche de Machine Learning supervisé où l'objectif est d'**attribuer une catégorie ou classe à une observation**.
- Formulation : pour une entrée x, le modèle prédit une classe $y \in \{C_1, C_2, ..., C_K\}$.
- Applications courantes :
  - détection de spam (email spam / non spam),
  - reconnaissance d'images (chien / chat / oiseau),
  - diagnostic médical (maladie présente / absente).
- Le choix du modèle dépend du nombre de classes, de la distribution et du type de données.

Structure des données pour la classification

- Chaque observation est décrite par :
  - **Variables prédictives** (X) : features numériques, catégorielles, textuelles, etc.
  - **Variable cible** (y) : classe associée.
- Les données sont souvent organisées sous forme de tableau :
  - lignes = observations,
  - colonnes = features et variable cible.
- Prétraitements spécifiques :
  - encodage des variables catégorielles,
  - normalisation ou standardisation,
  - gestion des classes déséquilibrées.

Types de classification

- **Classification binaire** : deux classes possibles (ex : spam / non spam).
- **Classification multiclasse** : plus de deux classes exclusives (ex : reconnaissance d'images).
- **Classification multilabel** : une observation peut appartenir à plusieurs classes simultanément.
- Choix des métriques et modèles dépend du type de classification.

Mesures de performance spécifiques à la classification

- Mesures standard :
  - **Exactitude (accuracy)** : proportion de prédictions correctes,
  - **Précision (precision)** : proportion de vrais positifs parmi les prédits positifs,
  - **Rappel (recall)** : proportion de vrais positifs parmi les réels positifs,
  - **F1-score** : moyenne harmonique de précision et rappel.
- Courbes ROC et AUC pour évaluer les classificateurs binaires.
- Gestion de classes déséquilibrées : métriques pondérées et matrices de confusion.

Pipeline général d'un classificateur

- Étapes typiques :
  - collecte et nettoyage des données,
  - sélection et transformation des features,
  - division en ensembles d'entraînement, validation et test,
  - choix et entraînement du modèle,
  - évaluation sur validation et ajustement des hyperparamètres,
  - déploiement et suivi en production.
- Importance des étapes de prétraitement et de validation pour la robustesse.

Lien avec la théorie décisionnelle

- La classification peut être vue comme un problème de **prise de décision sous incertitude**.
- La théorie décisionnelle associe :
  - des probabilités aux classes ($P(y|x)$),
  - des coûts ou utilités aux décisions.
- Objectif : minimiser le **risque attendu** ou maximiser l'utilité.
- Permet de justifier le choix des seuils et des stratégies de décision pour des classes déséquilibrées.

Motivation et cadre mathématique

- La régression logistique est utilisée pour résoudre des problèmes de **classification binaire**.
- Elle permet de modéliser la probabilité qu'une observation appartienne à une classe donnée :

$$P(y = 1|x) = f_\theta(x)$$

- Objectif : passer de la prédiction d'une valeur continue (comme dans la régression linéaire) à la prédiction d'une probabilité dans [0,1].
- Permet un cadre probabiliste pour la prise de décision et la gestion des seuils.

Fonction logistique et transformation sigmoïde

- La fonction logistique (ou sigmoïde) transforme toute valeur réelle en probabilité :

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Propriétés :
  - sortie comprise entre 0 et 1,
  - croissance monotone,
  - dérivable et continue.
- Application dans le modèle :

$$P(y = 1|x) = \sigma(\theta^T x)$$

Fonction de coût (log-loss)

- La fonction de coût mesure l'écart entre les probabilités prédites et les labels réels.
- Pour la classification binaire :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

- Minimiser le log-loss équivaut à maximiser la log-vraisemblance.
- Convexe pour la régression logistique, garantissant un minimum global.

Optimisation de la log-vraisemblance

- La log-vraisemblance s'écrit :

$$\ell(\theta) = \sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

- L'objectif est de trouver θ qui maximise $\ell(\theta)$ ou minimise le log-loss.
- Méthodes d'optimisation :
  - descente de gradient classique,
  - variants : SGD, mini-batch,
  - algorithmes de type Newton-Raphson pour convergence rapide.

Classification binaire : formulation complète

- Modèle complet :

$$P(y = 1|x) = \sigma(\theta^T x), \quad P(y = 0|x) = 1 - \sigma(\theta^T x)$$

- Prédiction finale avec un seuil t :

$$\hat{y} = \begin{cases} 1 & \text{si } \sigma(\theta^T x) \geq t \\ 0 & \text{sinon} \end{cases}$$

- Souvent $t = 0.5$, mais ajustable selon le compromis précision/rappel.

Extension multiclasse : softmax

- Pour K classes, utiliser la régression logistique multinomiale :

$$P(y = k|x) = \frac{e^{\theta_k^T x}}{\sum_{j=1}^{K} e^{\theta_j^T x}}$$

- Fonction softmax : généralisation de la sigmoïde à plusieurs classes.
- Log-loss généralisé pour multinomiale :

$$J(\theta) = -\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)}$$

- Permet la classification exclusive multi-classe.

Interprétabilité du modèle

- Chaque coefficient θ_j mesure l'impact de la feature x_j sur la probabilité logit :

$$\log \frac{P(y=1|x)}{P(y=0|x)} = \theta^T x$$

- Signes et valeurs des coefficients indiquent direction et force de l'influence.
- Permet l'analyse des facteurs de risque ou des caractéristiques discriminantes.
- Facile à expliquer aux non-spécialistes, avantage pour la médecine, finance, etc.

Régularisation dans la régression logistique

- Pour éviter le sur-apprentissage, on ajoute un terme de régularisation :
  - **Régularisation L1** : favorise la sparsité des coefficients,
  - **Régularisation L2** : limite la magnitude des coefficients.
- Forme générale du coût régularisé :

$$J(\theta) = \text{log-loss} + \lambda R(\theta)$$

- Permet d'améliorer la généralisation, surtout avec peu de données ou de nombreuses features.

Principe du séparateur optimal

- Les SVM visent à trouver un **hyperplan** qui sépare les classes dans l'espace des features.
- L'objectif est de choisir le séparateur qui maximise la **marge** entre les classes.
- Avantages :
  - robustesse aux outliers proches du bord,
  - bonne généralisation sur des espaces de grande dimension.
- Applicable pour des données linéairement séparables ou non, via transformation.

Marge maximale et hyperplans

- La **marge** est la distance entre l'hyperplan et les points les plus proches de chaque classe (vecteurs de support).
- Hyperplan optimal : maximise cette marge.
- Définition mathématique :

$$w^T x + b = 0 \quad \text{avec } \max \text{ marge}$$

- Les vecteurs de support sont cruciaux : seuls ces points influencent la position de l'hyperplan.

SVM linéaire

- Cas simple : classes séparables par un hyperplan linéaire.
- Fonction de décision :

$$f(x) = \text{sign}(w^T x + b)$$

- Optimisation :
  - minimiser $\frac{1}{2}\|w\|^2$
  - contraintes : $y_i(w^T x_i + b) \geq 1$
- Permet un apprentissage rapide et interprétable.

SVM non linéaire

- Certaines données ne sont pas linéairement séparables.
- Solution : transformer l'espace des features vers un espace de dimension plus élevée.
- L'hyperplan devient alors linéaire dans cet espace transformé.
- Transformation implicite possible grâce aux **noyaux**.

Truc des noyaux (kernel trick)

- L'idée : calculer les produits scalaires dans l'espace transformé sans passer par la transformation explicite.
- Noyaux courants :
  - linéaire : $K(x_i, x_j) = x_i^T x_j$
  - polynomial : $K(x_i, x_j) = (x_i^T x_j + c)^d$
  - RBF / gaussien : $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$
- Permet de gérer efficacement des problèmes non linéaires.

Pertes hinge et variantes

- Fonction de perte standard SVM : **hinge loss** :

$$L(y, f(x)) = \max(0, 1 - yf(x))$$

- Pénalise les points mal classés ou trop proches de l'hyperplan.
- Variantes :
  - hinge squared, smooth hinge,
  - combinaisons avec régularisation L1/L2.
- Optimisation basée sur la minimisation de la perte + régularisation.

Problèmes d'optimisation : primal et dual

- Problème primal :

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \quad \text{sous contraintes } y_i(w^T x_i + b) \geq 1 - \xi_i$$

- Formulation dual :
  - optimise les multiplicateurs de Lagrange α_i
  - permet d'utiliser le kernel trick
  - réduit la complexité pour les SVM linéaires dans des espaces à grande dimension
- Dualité primale-duale : avantage computationnel et théorique.

SVM pour la classification multiclasse

- SVM intrinsèquement binaire, extension aux multiples classes :
  - **One-vs-All (OvA)** : un SVM par classe contre toutes les autres
  - **One-vs-One (OvO)** : SVM entre chaque paire de classes
- Prédiction finale : choix de la classe la plus confiante selon la marge.
- Performance dépend du nombre de classes et de la complexité des frontières.

Structure et terminologie d'un arbre

- Un arbre de décision est un modèle prédictif basé sur une hiérarchie de décisions.
- Éléments clés :
  - **Noeud racine** : premier point de décision
  - **Noeuds internes** : conditions ou tests sur les variables
  - **Feuilles** : sorties ou prédictions finales
  - **Branches** : relient les noeuds et représentent le chemin décisionnel
- L'arbre partitionne l'espace des données en régions homogènes pour la variable cible.

Critères de séparation (Gini, Entropie)

- L'objectif est de choisir la meilleure variable et le meilleur seuil à chaque noeud.
- Critères courants :
  - **Indice de Gini** : mesure l'impureté d'un noeud
  - **Entropie (information gain)** : quantifie le gain d'information
  - **Erreur de classification** : proportion d'erreurs dans un noeud
- La variable et le seuil qui maximisent la pureté des sous-noeuds sont sélectionnés.

Processus de construction d'un arbre

- Construction récursive de l'arbre :
  - évaluer toutes les variables et tous les seuils
  - sélectionner le meilleur split selon un critère
  - répéter pour chaque sous-noeud jusqu'à :
    - noeud pur
    - profondeur maximale atteinte
    - nombre minimal d'exemples dans un noeud
- Résultat : un arbre capable de prédire en suivant les chemins des noeuds.

Pruning : pré-élagage et post-élagage

- Les arbres complets peuvent sur-apprendre (overfitting).
- Techniques de réduction :
  - **Pré-élagage** : arrêter la croissance si le gain est trop faible ou si le noeud est trop petit
  - **Post-élagage** : construire l'arbre complet puis supprimer les branches inutiles
- Objectif : améliorer la généralisation sur des données non vues.

Avantages et limites des arbres simples

- **Avantages :**
  - Interprétables et intuitifs
  - Peu de prétraitement nécessaire
  - Gestion des données mixtes (numériques, catégorielles)
- **Limites :**
  - Sensibles au sur-apprentissage
  - Hautement instables vis-à-vis des variations des données
  - Performances limitées sur des problèmes complexes

Extensions : forêts aléatoires, Boosted Trees

- Pour améliorer la performance et réduire la variance :
  - **Random Forests** : ensemble d'arbres construits sur des sous-échantillons et features aléatoires
  - **Boosted Trees** : construction séquentielle d'arbres corrigeant les erreurs précédentes (ex : Gradient Boosting)
- Ces méthodes offrent robustesse et performances accrues par rapport à un arbre simple.

Interprétabilité et visualisation

- Les arbres de décision permettent :
  - visualisation graphique des splits
  - extraction de règles décisionnelles
  - compréhension de l'importance des variables
- Pour les forêts et modèles boostés :
  - importance moyenne des features
  - partial dependence plots
  - SHAP values pour interprétation locale
- L'interprétabilité reste un atout majeur dans le contexte réglementaire et métier.

Principe de proximité

- Le kNN est une méthode basée sur les instances : la prédiction pour un nouvel exemple se fait en fonction des **k points les plus proches** dans l'ensemble d'apprentissage.
- L'hypothèse clé : des points similaires (proches dans l'espace des features) ont des labels similaires.
- Approche non paramétrique : pas de fonction paramétrique explicite, l'ensemble des données est directement utilisé pour la prédiction.

Choix de la distance

- La notion de proximité dépend de la métrique choisie :
  - **Distance euclidienne** : distance géométrique classique
  - **Distance Manhattan** : somme des distances absolues
  - **Distance Minkowski** : généralisation paramétrable
  - Autres distances : cosine, Mahalanobis, hamming (pour variables catégorielles)
- Le choix de la distance impacte fortement la performance et la pertinence des voisins.

Rôle du paramètre k

- k détermine le nombre de voisins pris en compte pour la prédiction.
- Choix de k :
  - petit k : plus sensible au bruit, risque de sur-apprentissage
  - grand k : lissage, réduction du bruit mais risque de sous-apprentissage
- k optimal souvent trouvé par validation croisée.

Processus de classification par vote

- Étapes de prédiction pour un nouvel exemple :
  - calculer les distances entre le nouvel exemple et tous les points de l'ensemble d'apprentissage
  - sélectionner les k plus proches voisins
  - **vote majoritaire** : la classe la plus fréquente parmi les voisins est prédite
  - possibilité de pondérer les votes selon la distance
- Approche similaire pour la régression : moyenne ou médiane des valeurs des k voisins.

Complexité algorithmique et stockage

- kNN est coûteux en calcul et mémoire :
  - **Entraînement** : simple, stockage des données
  - **Prédiction** : calcul des distances pour chaque nouvel exemple
- Complexité :
  - naïve : $O(n \cdot d)$ par prédiction (n = nb d'exemples, d = dimension)
  - optimisations : arbres KD, Ball Trees, Approximate Nearest Neighbors

Normalisation et métriques adaptées

- Importance de la normalisation des features :
  - évite que les variables à grande échelle dominent la distance
  - méthodes : min-max scaling, standardisation
- Choix de métriques adaptées selon le type de données :
  - numériques : euclidienne, Minkowski
  - catégorielles : Hamming
  - mixtes : Gower

Applications, avantages et limites

- **Applications :**
  - reconnaissance de formes, détection d'anomalies
  - recommandation, systèmes de filtrage collaboratif
  - classification médicale ou textuelle simple
- **Avantages :**
  - facile à comprendre et à implémenter
  - non paramétrique : flexible pour différentes formes de données
- **Limites :**
  - coûteux pour grandes bases de données
  - sensibilité au bruit et aux variables non pertinentes
  - nécessite normalisation et choix judicieux de k et de la distance

** Les pratiques **

Nettoyage des données

- Le nettoyage consiste à préparer les données brutes pour l'analyse et la modélisation.
- Actions courantes :
  - suppression ou correction des doublons
  - correction des incohérences et erreurs typographiques
  - harmonisation des formats et unités
  - filtrage des valeurs aberrantes évidentes
- Objectif : obtenir un jeu de données fiable et cohérent pour l'apprentissage.

Gestion des valeurs manquantes

- Les valeurs manquantes peuvent dégrader la performance des modèles.
- Méthodes de traitement :
  - suppression des lignes ou colonnes concernées
  - imputation par moyenne, médiane ou mode
  - imputation par modèles prédictifs (KNN, régression)
  - encodage spécifique pour conserver l'information manquante
- Choix de la méthode dépend de la proportion et du mécanisme de missingness.

Détection et traitement des outliers

- Les outliers sont des valeurs extrêmes pouvant perturber l'apprentissage.
- Détection :
  - méthodes statistiques : Z-score, IQR
  - méthodes visuelles : boxplots, scatter plots
- Traitement :
  - suppression
  - transformation (log, racine carrée)
  - winsorisation (limitation des valeurs extrêmes)
- Objectif : réduire l'effet des valeurs aberrantes sans perdre d'information pertinente.

Encodage des variables catégorielles

- Les modèles numériques nécessitent que les variables catégorielles soient converties en format numérique.
- Méthodes courantes :
  - **One-hot encoding** : une colonne par modalité
  - **Ordinal encoding** : transformation en entiers selon un ordre
  - **Target encoding** : moyenne de la cible par catégorie
  - **Embeddings** pour variables à forte cardinalité
- Choix de la méthode en fonction du modèle et de la nature des données.

Normalisation et standardisation

- Les features doivent souvent être mises à la même échelle pour certains modèles (kNN, SVM, réseaux de neurones).
- Techniques principales :
  - **Normalisation** : transformation des valeurs entre 0 et 1
  - **Standardisation** : centrer et réduire (moyenne 0, écart-type 1)
  - **Scaling robuste** : basé sur médiane et IQR pour données avec outliers
- Objectif : éviter que certaines variables dominent la distance ou la mise à jour des poids.

Réduction de dimension

- La réduction de dimension permet de simplifier les données tout en conservant l'information essentielle.
- Techniques courantes :
  - PCA (Analyse en Composantes Principales)
  - t-SNE, UMAP pour visualisation
  - Sélection de variables via corrélation ou importance
- Avantages :
  - réduction du bruit
  - amélioration des performances
  - meilleure visualisation

Sélection de variables

- Identifier les variables les plus pertinentes pour la prédiction.
- Méthodes principales :
  - filtres : corrélations, tests statistiques
  - wrappers : forward/backward selection, recursive feature elimination
  - méthodes embarquées : importance des features dans arbres ou modèles linéaires régularisés
- Objectifs :
  - réduire la dimensionnalité
  - améliorer la généralisation
  - diminuer le temps d'entraînement

Augmentation de données (data augmentation)

- Technique consistant à générer des données supplémentaires à partir du dataset existant.
- Applications principales :
  - images : rotations, translations, flips
  - texte : synonymes, back-translation
  - audio : bruit, pitch, time-stretch
- Objectifs :
  - améliorer la robustesse et généralisation du modèle
  - réduire le sur-apprentissage
  - compenser un dataset de petite taille

Motivations et enjeux de la compression

- La compression vise à réduire la taille des données tout en conservant l'information essentielle.
- Enjeux principaux :
  - accélérer les traitements et l'apprentissage
  - diminuer les coûts de stockage et de transmission
  - réduire le bruit et la redondance
  - faciliter la visualisation et l'interprétation
- La compression peut être **sans perte** ou **avec perte**, selon le compromis taille/qualité.

Méthodes de compression sans perte

- Les méthodes sans perte permettent de récupérer exactement les données originales.
- Techniques courantes :
  - codage Huffman
  - codage LZW
  - compression par dictionnaire ou run-length encoding
- Utilisation : fichiers texte, bases de données, données sensibles.
- Avantage : aucune information perdue, fiable pour tous types de données.

Méthodes de compression avec perte

- Les méthodes avec perte réduisent la taille en sacrifiant certaines informations jugées non essentielles.
- Techniques courantes :
  - compression JPEG pour images
  - compression MP3 pour audio
  - quantification et codage de données numériques
- Objectif : compromis entre taille et qualité perçue.
- Utilisation : images, vidéos, audio, signaux où une perte partielle est tolérable.

Réduction de dimension (ACP, ICA, etc.)

- Techniques linéaires permettant de projeter les données dans un espace de dimension inférieure.
- Principales méthodes :
  - **ACP (PCA)** : maximise la variance conservée dans les composantes principales
  - **ICA** : extraction de composantes indépendantes
  - **LDA** : maximisation de la séparation entre classes
- Avantages :
  - réduction du bruit
  - accélération des algorithmes
  - visualisation en 2D ou 3D

Réduction non linéaire (t-SNE, UMAP)

- Méthodes non linéaires pour projeter des données complexes dans un espace de faible dimension.
- Techniques :
  - **t-SNE** : préserve les distances locales, idéal pour visualisation
  - **UMAP** : préserve la structure globale et locale, plus rapide que t-SNE
- Utilisation : exploration, clustering et visualisation de données à haute dimension.
- Objectif : conserver les relations complexes entre points.

Compression via autoencodeurs

- Les autoencodeurs sont des réseaux de neurones entraînés à reproduire leurs entrées en sortie.
- Structure :
  - **Encodeur** : réduit la dimension (latent space)
  - **Decodeur** : reconstruit l'entrée
- Avantages :
  - compression non linéaire
  - extraction de caractéristiques pertinentes
  - applicable à images, audio et données tabulaires

Impact de la compression sur la performance des modèles

- La compression peut affecter la qualité des prédictions.
- Points clés :
  - réduction de dimension ou perte d'information peut diminuer la précision
  - réduction du bruit peut améliorer la généralisation
  - choix de la méthode de compression dépend du type de modèle et de la tolérance à la perte
- Stratégie : trouver un compromis entre compacité, efficacité et performance prédictive.

Choix des hyperparamètres clés

- Les hyperparamètres contrôlent le comportement et la capacité du modèle.
- Exemples courants :
  - nombre de couches et de neurones pour les réseaux de neurones
  - profondeur et nombre d'arbres pour les forêts aléatoires
  - valeur de k pour kNN
  - coefficient de régularisation pour la régression logistique ou SVM
- Le choix adéquat influence directement la performance et la généralisation du modèle.

Validation croisée (cross-validation)

- Technique pour estimer la performance d'un modèle sur des données non vues.
- Principe :
  - partitionner le jeu de données en k sous-ensembles (folds)
  - entraîner sur k-1 folds et tester sur le fold restant
  - répéter k fois pour obtenir une performance moyenne
- Avantages : réduit le risque d'overfitting et fournit une estimation robuste.

Recherche exhaustive (grid search)

- Méthode systématique pour trouver la meilleure combinaison d'hyperparamètres.
- Principe :
  - définir une grille de valeurs possibles pour chaque hyperparamètre
  - entraîner et évaluer le modèle pour chaque combinaison
  - sélectionner celle qui maximise la performance
- Limite : coûteux en temps de calcul si la grille est large.

Recherche aléatoire (random search)

- Alternative à la recherche exhaustive.
- Principe :
  - échantillonner aléatoirement des combinaisons d'hyperparamètres
  - évaluer les performances et retenir la meilleure combinaison
- Avantage : exploration plus large et efficace sur des espaces de grande dimension.

Optimisation bayésienne

- Méthode intelligente pour rechercher les hyperparamètres optimaux.
- Principe :
  - modéliser la fonction de performance comme une distribution probabiliste
  - choisir les points à tester pour maximiser l'amélioration attendue
  - mettre à jour la distribution à chaque itération
- Avantages : moins d'évaluations nécessaires, adaptée aux modèles coûteux à entraîner.

Réglages spécifiques aux modèles

- Chaque type de modèle possède des hyperparamètres particuliers :
  - **Réseaux de neurones** : taux d'apprentissage, dropout, batch size
  - **SVM** : coefficient C, type de noyau, gamma
  - **Arbres de décision** : profondeur max, min samples split, critère de division
  - **kNN** : nombre de voisins k, type de distance
- Adapter ces hyperparamètres permet de tirer le meilleur parti de chaque modèle.

Évaluation après réglages

- Après avoir sélectionné les hyperparamètres, il est crucial de réévaluer le modèle.
- Utiliser un jeu de test indépendant pour mesurer la performance réelle.
- Comparer les métriques avant et après réglages pour valider l'amélioration.
- Permet de détecter d'éventuels sur-ajustements sur le jeu de validation.

Gestion du sur-apprentissage via tuning

- Les techniques de tuning aident à réduire le sur-apprentissage :
  - régularisation (L1, L2)
  - réduction de la complexité du modèle
  - early stopping pour les réseaux de neurones
  - cross-validation pour éviter l'optimisation sur un jeu particulier
- Objectif : maximiser la performance sur des données nouvelles tout en évitant le sur-apprentissage.

** L’apprentissage d’ensembles **

Définition et motivation

- L'apprentissage d'ensemble consiste à combiner plusieurs modèles pour améliorer la performance globale.
- Motivation :
  - réduire les erreurs aléatoires,
  - augmenter la robustesse du modèle,
  - exploiter la complémentarité de différents modèles.
- Concept clé : un ensemble de modèles faibles peut surpasser un modèle unique plus complexe.

Réduction de la variance

- La variance correspond aux fluctuations de la prédiction sur différents échantillons.
- L'ensemble permet de lisser ces fluctuations :
  - plusieurs modèles indépendants sont combinés,
  - les erreurs individuelles tendent à se compenser,
  - la performance moyenne est plus stable.
- Exemples : forêts aléatoires (Random Forest).

Réduction du biais

- Le biais correspond à l'écart systématique entre les prédictions et la réalité.
- Certaines méthodes d'ensemble (boosting) permettent de corriger les erreurs successives des modèles faibles.
- L'ensemble augmente la capacité du modèle final à mieux approximier la fonction réelle.

Combinaison de plusieurs modèles faibles

- Un modèle faible est légèrement meilleur que le hasard.
- Plusieurs modèles faibles peuvent être combinés pour former un modèle fort.
- Techniques principales :
  - vote majoritaire pour classification,
  - moyenne pondérée pour régression.
- Avantage : robustesse et meilleure généralisation.

Méthodes de vote et d'agrégation

- **Vote majoritaire** : chaque modèle vote pour une classe, la classe la plus votée est retenue.
- **Moyenne ou moyenne pondérée** : combine les prédictions numériques pour la régression.
- **Stacking** : apprentissage d'un modèle méta sur les prédictions des modèles de base.
- Ces méthodes permettent d'exploiter la diversité des modèles.

Bagging vs Boosting

- **Bagging** (Bootstrap Aggregating) :
  - échantillonne aléatoirement les données,
  - entraîne des modèles indépendants,
  - réduit la variance (ex : Random Forest).
- **Boosting** :
  - construit les modèles séquentiellement,
  - chaque modèle corrige les erreurs du précédent,
  - réduit le biais (ex : AdaBoost, XGBoost, LightGBM).
- Choix de la méthode selon l'objectif : réduction de variance ou de biais.

Contraintes, limites et considérations pratiques

- Ensembles complexes = coûts de calcul et mémoire plus élevés.
- Risque de sur-apprentissage si les modèles de base sont trop corrélés.
- Nécessité de diversifier les modèles pour bénéficier de l'effet d'ensemble.
- Analyse et interprétabilité plus difficiles que pour un modèle simple.
- Importance de l'évaluation sur des données indépendantes pour valider la généralisation.

Concepts clés du bagging

- Le bagging (Bootstrap Aggregating) consiste à entraîner plusieurs modèles indépendants sur des échantillons aléatoires des données.
- Objectif : réduire la variance des prédictions et améliorer la robustesse.
- Chaque modèle contribue à la décision finale par agrégation (vote pour classification, moyenne pour régression).
- Les Forêts Aléatoires utilisent ce principe appliqué aux arbres de décision.

Construction d'arbres aléatoires

- Chaque arbre est entraîné sur un sous-échantillon bootstrap des données.
- L'arbre est construit en utilisant des critères classiques (Gini, entropie) pour les splits.
- Les arbres sont profondément construits pour capturer les relations complexes.
- Les corrélations entre arbres sont réduites grâce à la sélection aléatoire des variables à chaque split.

Sélection aléatoire des variables

- À chaque noeud, un sous-ensemble aléatoire de variables est considéré pour la séparation.
- Avantages :
  - réduction de la corrélation entre arbres,
  - amélioration de la généralisation,
  - robustesse face aux variables fortement dominantes.
- Paramètre clé : nombre de variables candidates à chaque split (`max_features`).

Variance, robustesse et stabilité

- La combinaison d'arbres indépendants réduit la variance du modèle final.
- La prédiction devient plus stable et moins sensible aux fluctuations des données.
- Les Forêts Aléatoires sont moins sujettes au sur-apprentissage qu'un arbre unique.
- Robustesse accrue face au bruit et aux outliers.

Importance des variables

- Les Forêts Aléatoires permettent d'évaluer l'influence de chaque variable sur la prédiction.
- Méthodes courantes :
  - diminution de l'impureté moyenne (Gini ou entropie),
  - permutation des variables et mesure de l'impact sur la performance.
- Utilité : sélection de variables et interprétation du modèle.

Problèmes et limites des Random Forests

- Moins interprétables qu'un arbre unique.
- Sensibles à la présence de nombreuses variables bruitées ou fortement corrélées.
- Coût mémoire et calcul plus élevé en raison de nombreux arbres.
- Moins performantes sur des données très clairsemées ou lorsque des extrapolations sont nécessaires.

Applications typiques

- Classification d'images et reconnaissance d'objets.
- Détection de fraudes et analyse financière.
- Prévision de la demande ou du comportement des clients.
- Analyse biomédicale et génomique.
- Recommandation et scoring de risque.

Principe général du boosting

- Le boosting est une technique d'apprentissage d'ensemble qui combine plusieurs modèles faibles pour obtenir un modèle fort.
- Chaque modèle successif se concentre sur les erreurs des modèles précédents.
- Objectif : réduire le biais et améliorer la performance prédictive.
- Différent du bagging : les modèles sont entraînés **séquentiellement**, pas indépendamment.

Minimisation séquentielle du résidu

- Chaque modèle tente de corriger les erreurs résiduelles du modèle précédent.
- La prédiction finale est la somme pondérée des prédictions de tous les modèles faibles.
- Permet de capturer progressivement les relations complexes dans les données.
- Le poids de chaque modèle peut être ajusté via un **learning rate**.

Arbres faibles successifs

- Les modèles faibles sont souvent de petits arbres de décision (stumps) avec faible profondeur.
- Ces arbres capturent uniquement des relations simples et sont combinés pour former un modèle puissant.
- Chaque arbre est entraîné sur les résidus pondérés des arbres précédents.
- Permet d'éviter le sur-apprentissage par simplicité des arbres individuels.

Fonction de perte et gradients

- Le Gradient Boosting se base sur la minimisation d'une fonction de perte différentiable.
- À chaque itération, le nouvel arbre est entraîné pour approximer le **gradient de la perte** par rapport aux prédictions actuelles.
- Permet une optimisation flexible : MSE pour régression, log-loss pour classification, etc.
- C'est la raison du nom **Gradient Boosting**.

Méthodes modernes (XGBoost, LightGBM, CatBoost)

- **XGBoost** : régularisation, parallélisation et gestion efficace des valeurs manquantes.
- **LightGBM** : histogram-based, rapide sur grands datasets et faible consommation mémoire.
- **CatBoost** : gère efficacement les variables catégorielles et réduit le biais des ordres de features.
- Ces méthodes sont largement utilisées en compétition et en production.

Hyperparamètres essentiels

- **Nombre d'arbres** (`n_estimators`) : plus d'arbres = meilleure approximation mais risque de sur-apprentissage.
- **Profondeur des arbres** (`max_depth`) : contrôle la complexité des arbres faibles.
- **Learning rate** : réduit l'impact de chaque arbre pour stabiliser l'apprentissage.
- **Subsample / colsample** : fraction des données ou features utilisées à chaque arbre.
- **Regularisation** : L1/L2 pour éviter le sur-apprentissage.

Avantages, limites et bonnes pratiques

- **Avantages :**
  - Très performant sur données tabulaires,
  - Capacité à gérer variables numériques et catégorielles,
  - Robustesse face aux valeurs manquantes.
- **Limites :**
  - Temps d'entraînement plus long que les méthodes simples,
  - Sensible aux hyperparamètres,
  - Moins interprétable qu'un arbre unique.
- **Bonnes pratiques :**
  - Utiliser validation croisée pour le tuning,
  - Appliquer régularisation et learning rate adapté,
  - Surveiller overfitting via early stopping.

** La régression **

Définition et objectifs

- La régression vise à prédire une **variable continue** à partir de variables explicatives.
- Objectifs principaux :
  - Estimer la relation entre les variables d'entrée et la sortie,
  - Quantifier l'effet de chaque variable explicative,
  - Fournir des prédictions pour de nouvelles observations.
- Utilisée dans de nombreux domaines : finance, économie, sciences, ingénierie.

Formulation mathématique générale

- Un modèle de régression peut être représenté par :

$$y = f(x) + \epsilon$$

où :

- y : variable cible,
- x : vecteur des variables explicatives,
- $f(x)$: fonction reliant x à y,
- ϵ : terme d'erreur aléatoire.

- L'objectif est d'estimer $f(x)$ à partir de données observées.

Types de variables en régression

- Variables **continues** : taille, prix, poids.
- Variables **discrètes** : nombre d'enfants, nombre d'achats.
- Variables explicatives peuvent être numériques ou catégorielles.
- Les transformations et encodages adaptés sont essentiels pour la performance du modèle.

Régression linéaire simple

- Relie une variable cible *y* à une seule variable explicative *x* :

$$y = \beta_0 + \beta_1 x + \epsilon$$

- β_0 : intercept, β_1 : pente.
- Hypothèses :
  - linéarité,
  - erreurs indépendantes et identiquement distribuées,
  - homoscédasticité (variance constante),
  - distribution normale des résidus.

Régression linéaire multiple

- Relie la variable cible à plusieurs variables explicatives :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

- Permet de modéliser des relations plus complexes.
- Hypothèses similaires à la régression simple : linéarité, indépendance des erreurs, homoscédasticité.
- Évaluation via R^2, RMSE et tests de significativité des coefficients.

Hypothèses du modèle linéaire

- **Linéarité** : la relation entre chaque variable explicative et la cible est linéaire.
- **Indépendance des résidus** : les erreurs sont indépendantes.
- **Homoscédasticité** : variance constante des erreurs.
- **Normalité des résidus** : erreurs distribuées selon une loi normale.
- Respect des hypothèses = validité des tests statistiques et confiance dans les prédictions.

Erreurs, bruit et distribution des résidus

- Les résidus ($\hat{\epsilon} = y - \hat{y}$) représentent la différence entre observations et prédictions.
- Le bruit est inhérent aux données et limite la précision du modèle.
- Analyse des résidus :
  - détection de tendances non modélisées,
  - identification d'outliers,
  - vérification des hypothèses.

Biais, variance et capacité du modèle

- Le biais : erreur due aux hypothèses restrictives du modèle (ex. linéarité).
- La variance : sensibilité du modèle aux fluctuations des données d'entraînement.
- La capacité du modèle : complexité maximale qu'il peut apprendre.
- Objectif : équilibrer biais et variance pour une meilleure généralisation.

Analyse exploratoire des variables

- Examiner les statistiques descriptives des variables :
  - moyennes, médianes, écarts-types,
  - minimum et maximum, quantiles.
- Identifier les distributions, asymétries et outliers.
- Comprendre la nature des variables : continue, discrète, catégorielle.
- Détection précoce de problèmes de qualité de données.

Relations linéaires et non linéaires

- Explorer les relations entre chaque variable explicative et la cible.
- Détection de dépendances linéaires via corrélation.
- Identifier les relations non linéaires possibles nécessitant transformation ou modèle complexe.
- Importance pour le choix du modèle et des transformations.

Corrélations et multicolinéarité

- Calculer les corrélations entre variables explicatives.
- La **multicolinéarité** peut dégrader la stabilité des coefficients en régression multiple.
- Méthodes pour détecter :
  - matrice de corrélation,
  - facteur d'inflation de variance (VIF).
- Conséquences : coefficients peu fiables, difficultés d'interprétation.

Visualisations pour la régression

- Diagrammes de dispersion (scatter plots) pour visualiser relations.
- Boxplots et histogrammes pour distributions.
- Heatmaps pour visualiser corrélations.
- Graphiques résidu vs prédiction pour détecter tendances ou hétéroscédasticité.

Détection des points influents

- Identifier les observations atypiques qui impactent fortement le modèle.
- Méthodes classiques :
  - Distance de Cook,
  - Leverage (hiérarchie des points dans l'espace des X),
  - Analyse des résidus studentisés.
- Importance : améliorer la robustesse et fiabilité du modèle.

Analyse des résidus

- Les résidus représentent la différence entre valeur observée et prédite.
- Vérifier :
  - distribution normale des résidus,
  - homoscédasticité,
  - absence de structure non modélisée.
- Résidus anormaux = indication de problèmes dans le modèle ou les données.

Transformations des variables

- Objectif : améliorer linéarité et distribution.
- Techniques courantes :
  - logarithme, racine carrée, puissance,
  - standardisation et normalisation,
  - encodage catégoriel approprié.
- Permet de réduire hétéroscédasticité et améliorer performance du modèle.

Fonctions de perte pour la régression

- La fonction de perte quantifie l'erreur entre valeurs observées et prédites.
- Fonctions de perte courantes :
  - **Erreur quadratique (MSE)** : met l'accent sur les grandes erreurs,
  - **Erreur absolue (MAE)** : robuste aux outliers,
  - **Huber loss** : compromis entre MSE et MAE.
- Le choix impacte l'optimisation et la sensibilité du modèle aux valeurs extrêmes.

Mesures d'évaluation : MSE, RMSE, MAE

- **MSE (Mean Squared Error)** : moyenne des carrés des erreurs.
- **RMSE (Root Mean Squared Error)** : racine carrée de MSE, dans les mêmes unités que la cible.
- **MAE (Mean Absolute Error)** : moyenne des erreurs absolues, robuste aux outliers.
- Permettent de comparer les performances de différents modèles.

Coefficient de détermination (R^2)

- Indique la proportion de variance de la variable cible expliquée par le modèle.
- Valeurs :
  - $R^2 = 1$: modèle parfait,
  - $R^2 = 0$: le modèle n'explique rien,
  - $R^2 < 0$: modèle moins performant qu'une moyenne simple.
- Utile pour juger de l'adéquation globale du modèle.

Validation croisée pour la régression

- Permet d'estimer la performance du modèle sur des données non vues.
- Techniques :
  - K-fold cross-validation,
  - Leave-One-Out (LOO),
  - Shuffle & split.
- Objectif : réduire le sur-apprentissage et obtenir une mesure fiable de généralisation.

Diagnostic des erreurs

- Analyse des résidus pour vérifier :
  - distribution centrée autour de 0,
  - homoscédasticité (variance constante),
  - absence de corrélation avec les variables explicatives.
- Identification des patterns non modélisés.
- Détection de points influents et outliers.

Sur-apprentissage et sous-apprentissage

- **Sur-apprentissage (overfitting)** : modèle trop complexe, bonne performance en entraînement mais mauvaise en test.
- **Sous-apprentissage (underfitting)** : modèle trop simple, mauvaise performance en entraînement et en test.
- L'équilibre dépend du choix du modèle, des features et du volume de données.

Comparaison entre modèles

- Comparer plusieurs modèles à l'aide de :
  - métriques d'erreur (MSE, RMSE, MAE),
  - R^2,
  - validation croisée.
- Prendre en compte :
  - complexité du modèle,
  - temps d'entraînement,
  - robustesse aux outliers.
- Objectif : choisir le modèle avec meilleure généralisation sur des données non vues.

** Le clustering **

Définition et motivations

- Le clustering consiste à regrouper des observations similaires en **clusters** ou groupes.
- Objectifs :
  - Identifier des structures cachées dans les données,
  - Segmentation de clients, documents, images, etc.,
  - Réduction de complexité pour l'analyse exploratoire.
- Approche exploratoire non supervisée : pas de variable cible.

Apprentissage non supervisé

- Contrairement à l'apprentissage supervisé, il n'existe pas de labels.
- Le modèle cherche des **motifs intrinsèques** dans les données.
- Le clustering est une technique clé de l'apprentissage non supervisé.
- Applications typiques : segmentation, détection d'anomalies, compression de données.

Notion de similarité et de distance

- La définition des clusters repose sur une mesure de **proximité**.
- Mesures courantes :
  - **Distance Euclidienne** : adaptée aux données continues,
  - **Distance de Manhattan / Minkowski**,
  - **Distances cosinus ou corrélation** : pour vecteurs normalisés,
  - **Mesures spécifiques** : Jaccard, Hamming, etc.
- Le choix de la distance influence fortement la forme et la structure des clusters.

Types de structures de clusters

- Différentes structures peuvent apparaître selon les données :
  - **Clusters sphériques** : k-means, k-medoids,
  - **Clusters hiérarchiques** : dendrogrammes, agglomératif,
  - **Clusters de densité** : DBSCAN, OPTICS,
  - **Structures non linéaires ou manifolds** : t-SNE, UMAP pour visualisation.
- Comprendre la structure permet de choisir l'algorithme adapté.

Prétraitements indispensables

- Prétraitements essentiels pour un clustering efficace :
  - **Normalisation / standardisation** des variables,
  - **Gestion des valeurs manquantes** et outliers,
  - **Réduction de dimension** si données haute dimension (PCA, t-SNE),
  - **Encodage des variables catégorielles** si nécessaire.
- Objectif : homogénéiser l'échelle et améliorer la qualité des clusters.

Évaluation du clustering

- Le clustering étant non supervisé, l'évaluation repose sur des mesures internes ou externes :
  - **Indices internes** : silhouette, Davies-Bouldin, Dunn index,
  - **Indices externes** : ajustés si labels connus (ARI, NMI),
  - **Stabilité des clusters** : répétitions avec différentes initialisations.
- Permet de déterminer le nombre optimal de clusters et la qualité de la segmentation.

Applications pratiques

- **Segmentation client** : marketing et personnalisation,
- **Regroupement de documents** : moteurs de recherche et recommandation,
- **Détection d'anomalies** : fraude, maintenance prédictive,
- **Exploration de données** : visualisation et réduction de dimension,
- **Bioinformatique** : clustering de gènes ou de protéines.

Principe général

- k-Means est un algorithme de clustering partitionnel.
- Objectif : regrouper n observations en k clusters afin de minimiser la variance intra-cluster.
- Chaque cluster est représenté par un **centroïde**, calculé comme la moyenne des points assignés.
- Processus itératif : assignation → recalcul des centroïdes → convergence.

Initialisation des centroïdes

- L'initialisation influence fortement la convergence et la qualité finale.
- Méthodes courantes :
  - Choix aléatoire de k points parmi les observations,
  - k-Means++ : initialise les centroïdes de manière espacée pour améliorer la convergence.
- Mauvaise initialisation peut entraîner des minima locaux suboptimaux.

Assignation des points aux clusters

- Chaque point est assigné au cluster dont le centroïde est le plus proche.
- Distance typique : Euclidienne, mais d'autres distances peuvent être utilisées.
- L'assignation définit la composition des clusters à l'itération courante.

Mise à jour des centroïdes

- Une fois tous les points assignés, chaque centroïde est recalculé :

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

où C_j est le cluster j et μ_j son centroïde.

- Répétition itérative : assignation → mise à jour jusqu'à convergence.

Critères d'arrêt

- L'algorithme s'arrête lorsque :
  - Les centroïdes ne changent plus ou changent très peu,
  - La variation de la somme des distances intra-cluster est minimale,
  - Un nombre maximal d'itérations est atteint.
- Assure la stabilité des clusters et la fin du processus itératif.

Choix du nombre de clusters

- k est un hyperparamètre clé du clustering.
- Méthodes pour déterminer k :
  - **Méthode du coude** : observer la courbe de variance intra-cluster en fonction de k,
  - **Coefficient de silhouette** : mesure la cohésion intra-cluster et la séparation inter-cluster,
  - **Indices statistiques** : Calinski-Harabasz, Davies-Bouldin.

Variantes du k-Means

- **k-Means++** : améliore l'initialisation des centroïdes.
- **MiniBatch k-Means** : traitement de grands volumes de données par mini-lots.
- **k-Medoids** : remplace les centroïdes par des points représentatifs pour plus de robustesse aux outliers.
- Objectif : réduire le temps de calcul et améliorer la stabilité des résultats.

Avantages, limites et cas d'usage

- **Avantages** :
  - Simplicité et rapidité pour les petits et moyens datasets,
  - Interprétabilité des clusters par centroïdes.
- **Limites** :
  - Sensible aux outliers et aux valeurs aberrantes,
  - Nécessite de fixer k à l'avance,
  - Forme sphérique implicite des clusters.
- **Cas d'usage** : segmentation client, compression de données, clustering d'images ou de textes.

Approche agglomérative

- Méthode bottom-up : chaque point commence comme un cluster individuel.
- À chaque étape, les clusters les plus proches sont fusionnés.
- Le processus se répète jusqu'à obtenir un seul cluster ou un nombre souhaité de clusters.
- Avantage : produit une hiérarchie complète des clusters.

Approche divisive

- Méthode top-down : commence avec un unique cluster englobant toutes les données.
- À chaque étape, un cluster est divisé en sous-clusters plus homogènes.
- Répétition jusqu'à atteindre des clusters atomiques ou un nombre souhaité.
- Moins utilisée que l'approche agglomérative en raison de sa complexité computationnelle.

Métriques de distance inter-clusters

- La fusion/division dépend de la mesure de distance entre clusters.
- Méthodes courantes :
  - **Distance simple (single linkage)** : distance minimale entre points de deux clusters.
  - **Distance complète (complete linkage)** : distance maximale.
  - **Distance moyenne (average linkage)** : moyenne des distances inter-points.
  - **Centroid linkage** : distance entre centroïdes des clusters.
- Le choix impacte la forme et la structure finale des clusters.

Dendrogrammes et interprétation

- Le dendrogramme est une représentation graphique de la hiérarchie des clusters.
- Chaque feuille représente un point, chaque noeud une fusion/division.
- La hauteur des noeuds indique la distance entre clusters.
- Permet de visualiser la structure et de décider du nombre de clusters à retenir.

Choix du nombre de partitions

- Le nombre de clusters peut être choisi en « coupant » le dendrogramme à une certaine hauteur.
- Méthodes quantitatives :
  - Analyse de la distance entre fusions successives,
  - Indices de cohésion et de séparation (silhouette, Davies-Bouldin),
  - Critères statistiques ou heuristiques.
- Objectif : obtenir des clusters homogènes et bien séparés.

Complexité et limites

- Complexité temporelle : $O(n^2 \log n)$ pour les implémentations optimisées, $O(n^3)$ dans le pire cas.
- Complexité mémoire : stockage complet des distances inter-points.
- Limites :
  - Sensible au bruit et aux outliers,
  - Moins adapté aux grands jeux de données,
  - Résultats dépendants de la métrique et du linkage choisi.

Applications spécifiques du clustering hiérarchique

- Biologie : classification de gènes et taxonomie des espèces.
- Text Mining : regroupement de documents ou articles similaires.
- Marketing : segmentation clients hiérarchique.
- Analyse d'images : détection de structures ou motifs hiérarchiques.
- Exploration de données complexes où la hiérarchie est informative.

Motivation : clusters denses vs bruit

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) vise à identifier des clusters **denses** dans les données.
- Permet de distinguer **points bruit** (outliers) des points appartenant à un cluster.
- Ne nécessite pas de connaître le nombre de clusters à l'avance.
- Adapté à des formes de clusters irrégulières et non sphériques.

Définition des concepts : eps et minPts

- **eps** : rayon de voisinage autour d'un point.
- **minPts** : nombre minimum de points pour qu'un point soit considéré comme dense.
- Un point est **core** si au moins minPts points sont dans son voisinage eps.
- Ces deux paramètres contrôlent la densité minimale requise pour former un cluster.

Points coeur, bordure et bruit

- **Point coeur** : possède au moins minPts dans son voisinage eps.
- **Point bordure** : voisin d'un point coeur mais ne satisfait pas minPts.
- **Bruit (outlier)** : ne fait partie d'aucun cluster et n'est pas un point bordure.
- Ces distinctions permettent à DBSCAN de gérer les outliers efficacement.

Algorithme DBSCAN

- Étape 1 : marquer tous les points comme non visités.
- Étape 2 : pour chaque point non visité, vérifier si c'est un point coeur.
- Étape 3 : si oui, créer un nouveau cluster et ajouter tous les points densément connectés.
- Étape 4 : répéter jusqu'à ce que tous les points aient été visités.
- Résultat : ensemble de clusters et points bruit identifiés.

Choix des paramètres et sensibilité

- eps trop petit $\Rightarrow$ trop de petits clusters ou points isolés.
- eps trop grand $\Rightarrow$ fusion de clusters distincts.
- minPts faible $\Rightarrow$ clusters très sensibles au bruit.
- minPts élevé $\Rightarrow$ points denses uniquement considérés, certains clusters peuvent être ignorés.
- Recommandation : visualisation préliminaire, diagramme k-distances pour eps.

Avantages et limites de DBSCAN

- **Avantages :**
  - Identifie clusters de forme arbitraire,
  - Détecte le bruit,
  - Pas besoin de connaître le nombre de clusters.
- **Limites :**
  - Sensible aux paramètres eps et minPts,
  - Difficultés avec densités très variées,
  - Complexité : $O(n \log n)$ avec structures de données adaptées, sinon $O(n^2)$.

Comparaison avec K-means et clustering hiérarchique

- **K-means** : clusters sphériques, sensible aux outliers, nécessite K.
- **Clustering hiérarchique** : construit une hiérarchie, sensible aux choix de distance et linkage.
- **DBSCAN** : clusters arbitraires, robuste au bruit, pas besoin de nombre de clusters.
- Choix dépend de :
  - forme des clusters,
  - présence de bruit,
  - taille et densité des données.

** Le deep learning **

Définition et distinction avec le Machine Learning

- Le **Deep Learning** est une sous-catégorie du Machine Learning utilisant des **réseaux de neurones profonds**.
- Différence clé :
  - Machine Learning classique : souvent basé sur des features manuelles et modèles simples.
  - Deep Learning : apprend automatiquement des représentations hiérarchiques des données.
- Idéal pour les données volumineuses et complexes : images, audio, texte, vidéo.

Réseaux de neurones comme modèles paramétriques

- Un réseau de neurones est une fonction paramétrique :

$$\hat{y} = f_\theta(x)$$

  avec θ représentant tous les poids et biais.
- Les neurones appliquent des transformations non linéaires sur les entrées.
- La composition de plusieurs couches permet de modéliser des relations complexes.
- L'apprentissage consiste à ajuster θ pour minimiser une fonction de perte.

Apprentissage par descente de gradient

- Objectif : minimiser une fonction de perte $L(\theta)$ par rapport aux paramètres θ.
- Descente de gradient classique :

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$$

- Variantes courantes : SGD, mini-batch, Adam, RMSProp.
- La rétropropagation calcule efficacement les gradients couche par couche.

Représentations hiérarchiques

- Chaque couche apprend des **représentations de plus en plus abstraites**.
- Exemples :
  - Vision : pixels $\rightarrow$ bords $\rightarrow$ motifs $\rightarrow$ objets.
  - NLP : caractères $\rightarrow$ mots $\rightarrow$ phrases $\rightarrow$ concepts.
- Permet de capturer des structures complexes et invariances dans les données.

Notion de profondeur

- La profondeur correspond au nombre de couches dans un réseau.
- Plus de couches = plus de capacité à représenter des fonctions complexes.
- Risques associés : sur-apprentissage, vanishing/exploding gradients.
- Techniques pour contrer ces problèmes : normalisation, résidual connections, activation appropriée.

Applications modernes du Deep Learning

- Vision par ordinateur : reconnaissance d'images, détection d'objets, segmentation.
- Traitement du langage naturel : traduction automatique, génération de texte, analyse de sentiments.
- Audio et signal : reconnaissance vocale, synthèse sonore.
- Séries temporelles : prévisions financières, maintenance prédictive.
- Applications multimodales : IA générative, agents conversationnels, systèmes autonomes.

Modèle et intuition

- Le perceptron est le modèle de neurone artificiel le plus simple.
- Il combine les entrées pondérées et produit une sortie binaire :

$$\hat{y} = f\Big(\sum_i w_i x_i + b\Big)$$

- Intuition : un neurone prend des signaux d'entrée et "décide" d'activer ou non.
- Idéal pour apprendre des frontières linéaires entre classes.

Fonction d'activation

- La fonction d'activation transforme la sortie linéaire en signal exploitable.
- Types courants pour le perceptron :
  - Fonction **Heaviside** (seuil) : binaire 0 ou 1.
  - Fonction sigmoïde : valeurs continues entre 0 et 1.
- Permet de modéliser la décision non linéaire lorsque combiné dans des architectures profondes.

Règle d'apprentissage du perceptron

- Ajustement des poids basé sur l'erreur de prédiction :

$$w_i \leftarrow w_i + \eta(y - \hat{y})x_i$$

- η : taux d'apprentissage.
- La règle converge si les données sont linéairement séparables.
- Processus itératif sur toutes les observations de l'ensemble d'entraînement.

Cas linéairement séparables

- Si les classes peuvent être séparées par un hyperplan, le perceptron converge vers une solution correcte.
- Exemple : deux classes distinctes dans un plan 2D.
- Limitation : impossible de résoudre des problèmes non linéaires comme le XOR.

Limites du perceptron simple

- Ne peut pas modéliser des relations non linéaires complexes.
- Sensible à l'ordre des données et au taux d'apprentissage.
- Ne fournit pas de probabilités directes pour les classes.
- Nécessité d'extensions pour des tâches modernes.

Extension au perceptron multicouche

- Plusieurs couches de neurones permettent de capturer des frontières non linéaires.
- Introduction de fonctions d'activation non linéaires : sigmoïde, ReLU, tanh.
- Apprentissage via **rétropropagation** pour ajuster tous les poids simultanément.
- Base de tous les réseaux de neurones modernes.

Architecture d'un MLP

- Un MLP est composé de plusieurs couches de neurones :
  - **Couche d'entrée** : reçoit les features.
  - **Couches cachées** : traitement non linéaire des informations.
  - **Couche de sortie** : génère la prédiction.
- Chaque neurone effectue une combinaison linéaire suivie d'une activation non linéaire.
- Les MLP peuvent approximer toute fonction continue sous certaines conditions.

Propagation avant (forward pass)

- Les entrées sont propagées couche par couche jusqu'à la sortie.
- Calcul dans chaque neurone :

$$z = \sum_i w_i x_i + b, \quad a = f(z)$$

- Les activations a de chaque couche deviennent les entrées de la suivante.
- Le forward pass permet de calculer la sortie et la perte associée.

Fonctions d'activation modernes

- Les fonctions d'activation introduisent la non-linéarité :
  - Sigmoïde : $\sigma(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
  - ReLU : $\mathrm{ReLU}(x) = \max(0, x)$
  - Variantes : Leaky ReLU, GELU, ELU
- Choix critique pour convergence et performance.

Rétropropagation du gradient

- Méthode pour calculer les gradients de la perte par rapport aux poids.
- Applique la règle de la chaîne en partant de la sortie vers l'entrée.
- Permet la mise à jour des paramètres via la descente de gradient :

$$w \leftarrow w - \eta \frac{\partial \mathcal{L}}{\partial w}$$

- Fondement de l'entraînement des MLP et réseaux profonds.

Coût et optimisation

- Fonction de perte $\mathcal{L}$: mesure l'écart entre prédiction et vérité.
- Objectif : minimiser $\mathcal{L}$ sur tout l'ensemble d'entraînement.
- Optimisation par descente de gradient et variantes :
  - SGD, mini-batch,
  - Adam, RMSProp, AdaGrad.
- Paramètres ajustés pour réduire l'erreur et améliorer la généralisation.

Sur-apprentissage et régularisation

- Les MLP très profonds peuvent sur-apprendre aux données d'entraînement.
- Techniques de régularisation :
  - Dropout : suppression aléatoire de neurones pendant l'entraînement.
  - Early stopping : arrêt de l'entraînement avant convergence complète.
  - Régularisation L1/L2 sur les poids.
  - Data augmentation pour augmenter la diversité des données.
- Objectif : améliorer la capacité de généralisation sur données nouvelles.

Applications du MLP

- Classification binaire et multiclasse.
- Régression et approximation de fonctions.
- Reconnaissance d'images simples et prétraitées.
- Séries temporelles et prévisions financières.
- Précurseur des architectures profondes modernes (CNN, RNN, Transformer).

Motivation : données spatiales et images

- Les CNN sont conçus pour traiter des données structurées spatialement, notamment :
  - images,
  - vidéos,
  - cartes 2D/3D,
  - signaux multivariés.
- Ils exploitent la proximité locale et la hiérarchie des motifs.
- Réduction drastique du nombre de paramètres par rapport aux MLP classiques pour les mêmes entrées.

Opération de convolution

- La convolution applique un **filtre/kernel** sur l'entrée pour extraire des caractéristiques.
- Formule :

$$(I * K)(x, y) = \sum_m \sum_n I(x - m, y - n) K(m, n)$$

- Permet :
  - détection de motifs locaux (bords, textures),
  - translation invariance partielle.
- Chaque couche apprend des filtres adaptés aux données.

Couches de pooling

- Réduisent la dimension spatiale tout en conservant l'information importante.
- Types de pooling :
  - Max pooling : sélection du maximum,
  - Average pooling : moyenne locale,
  - Global pooling : réduction à une seule valeur par canal.
- Avantages :
  - réduction des paramètres et du coût computationnel,
  - meilleure invariance aux translations.

Architecture typique d'un CNN

- Organisation classique :
  1. Couches convolutionnelles pour extraire des features,
  2. Couches de pooling pour réduire la dimension,
  3. Couches entièrement connectées (dense) pour classifier ou régresser.
- Ajout possible de normalisation (BatchNorm) et de régularisation (Dropout).
- Empilement de blocs convolutionnels pour apprendre des représentations hiérarchiques.

CNN modernes : VGG, ResNet, Inception

- VGG : empilement simple de convolutions 3x3 avec pooling.
- ResNet : introduction de **skip connections** pour faciliter l'apprentissage de réseaux profonds.
- Inception : blocs parallèles avec convolutions de tailles différentes pour capturer diverses échelles.
- Chaque architecture répond à des besoins spécifiques en profondeur, complexité et capacité d'extraction de features.

Normalisation et régularisation dans les CNN

- Normalisation :
  - Batch Normalization,
  - Layer Normalization.
- Régularisation :
  - Dropout dans les couches fully-connected,
  - Data augmentation pour images (rotation, flipping, bruit),
  - Early stopping pendant l'entraînement.
- Objectif : stabiliser l'entraînement et éviter le sur-apprentissage.

Applications des CNN

- Vision par ordinateur :
  - classification d'images,
  - détection d'objets,
  - segmentation sémantique.
- Traitement vidéo :
  - suivi d'objets,
  - reconnaissance d'actions.
- Autres domaines :
  - reconnaissance faciale,
  - imagerie médicale,
  - véhicules autonomes.

Principe des dépendances temporelles

- Les RNN sont conçus pour traiter des **données séquentielles**.
- Ils exploitent la dépendance entre les éléments de la séquence.
- Chaque prédiction peut dépendre des états précédents :

$$h_t = f(h_{t-1}, x_t)$$

- Permet de modéliser :
  - séries temporelles,
  - texte et langage,
  - signaux audio.

Structure récurrente : états cachés

- Chaque unité RNN maintient un **état caché** qui mémorise l'information de la séquence :

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b)$$

- Les états cachés transmettent l'information au pas suivant.
- Permet de capturer des patterns temporels et contextuels.

Problèmes de gradients explosifs ou qui disparaissent

- Pendant la rétropropagation dans le temps (BPTT) :
  - les gradients peuvent devenir très petits $\Rightarrow$ disparition,
  - les gradients peuvent devenir très grands $\Rightarrow$ explosion.
- Conséquences :
  - apprentissage instable,
  - incapacité à capturer de longues dépendances.
- Solutions : gradient clipping, architectures LSTM/GRU.

LSTM et GRU : architectures avancées

- LSTM (Long Short-Term Memory) : cellules avec portes d'entrée, sortie et oubli.
- GRU (Gated Recurrent Unit) : version simplifiée avec moins de paramètres.
- Objectif : capturer de longues dépendances sans disparaître les gradients.
- Amélioration significative des performances sur les séquences longues.

Séquences, séries temporelles et NLP

- RNNs adaptés à :
  - prévision de séries temporelles (finance, météo),
  - analyse de texte et langage naturel (traduction, génération),
  - reconnaissance vocale et traitement audio.
- Permettent de prédire un pas suivant, une séquence entière ou un label global.

Applications des RNN

- Traitement du langage naturel :
  - génération de texte, traduction automatique.
- Traitement audio :
  - reconnaissance vocale, détection de sons.
- Séries temporelles et signaux :
  - prévision financière, analyse médicale, capteurs IoT.

Comparaison RNN vs CNN vs MLP

- **MLP** : convient aux données tabulaires et non structurées.
- **CNN** : exploite la structure spatiale, idéal pour images et grilles.
- **RNN** : exploite les dépendances temporelles et séquentielles.
- Choix du modèle selon :
  - nature des données,
  - longueur des séquences,
  - besoin en mémoire contextuelle.

Sources bibliographiques

- **Joel Grus**, *Data Science from Scratch: First Principles with Python*, O Reilly.
- **Éric Biernat and Michel Lutz**, *Data science : fondamentaux et études de cas – Machine Learning avec Python et R*, Eyrolles, october 2015, ISBN 978-2-212-14243-3.
- **Dirk P.Kroese, Zdravko I.Botev, Thomas Taimre and Radislav Vaisman**, *Data Science and Machine Learning: Mathematical and Statistical Methods*, Chapman & Hall/CRC, 2019.

** Merci de votre attention ! **