



Module de spécialisation Data Business Intelligence

19 Janvier 2026
Ines Krissaane

Bordeaux - Lyon - Paris

Data & Business Intelligence

Type d'intervention	Description	Date
Module 1	Module de spécialisation	29/09/2025
Module 2	Module de spécialisation	30/09/2025
Module 3	Module de spécialisation	01/10/2025
Module 4	Module de spécialisation	19/01/2025
Projet 1	Kick-Off + Workshop	21/11/2025
Projet 1	Workshop & Follow-up	12/12/2025
Projet 1	Workshop & soutenance	19/12/2025
Projet 2	Kick-Off + Workshop	20/01/2026
Projet 2	Workshop & Follow-up	22/01/2026
Projet 2	Workshop & Follow-up	26/01/2026
Projet 2	Workshop & Follow-up	27/01/2026
Projet 2	Workshop & soutenance	29/01/2026
Projet 3	Kick-Off + Workshop	20/03/2026
Projet 3	Workshop & Follow-up	03/04/2026
Projet 3	Workshop & soutenance	17/04/2026

Module 4 - Deep Learning et IA responsable

Séquence	Durée	Contenu / Activité
Matin		
Séquence 1	3h	Cours théorique
Après-midi		
Séquence 2	1h30	Atelier pratique 4
Séquence 3	2h	Restitution & QCM

Objectifs pédagogiques

Partie 1 – Apprentissage automatique et réseaux de neurones

- Comprendre l'**architecture des réseaux de neurones** et leurs principes de fonctionnement.
- Explorer les **différents types de réseaux** :
 - **CNN (Convolutional Neural Networks)**
 - **RNN (Recurrent Neural Networks)**
 - **LSTM (Long Short-Term Memory)**
 - **Transformers**

Partie 2 - IA responsable, KPI & reporting stratégique

- Comprendre et appliquer les principes d'**équité, transparence et régulation** en IA
- Utiliser des outils d'**interprétabilité** : SHAP, LIME
- Mettre en place une stratégie de **monitoring**
- Traduire les performances techniques en **indicateurs business** (KPI, ROI)

Partie 1 – Apprentissage automatique et réseaux de neurones

Rappel : Modèles étudiés lors du Module 3

Nous avons exploré plusieurs grandes familles de modèles de Machine Learning supervisé:

- **Arbres de décision** : divisent l'espace des variables explicatives en régions homogènes afin de produire des règles de décision interprétables.
- **Forêts aléatoires** : combinent un grand nombre d'arbres de décision entraînés sur des sous-échantillons aléatoires pour améliorer la robustesse et la précision des prédictions.
- **Méthodes de Gradient Boosting** (XGBoost, LightGBM, CatBoost) : construisent séquentiellement des arbres faibles, chaque nouvel arbre cherchant à corriger les erreurs des précédents.

1. Validation croisée : éviter le surapprentissage

Principe : diviser les données en plusieurs sous-ensembles (k-fold) et tester le modèle sur des parties différentes.

2. Métriques d'évaluation

a) Régression :

- **RMSE (Root Mean Squared Error)** → évalue l'erreur quadratique moyenne.
- **MAE (Mean Absolute Error)** → mesure l'écart moyen absolu.
- **R² (Coefficient de détermination)** → proportion de la variance expliquée.

b) Classification :

- **Précision** → proportion de bonnes prédictions.
- **Rappel** → capacité à détecter les cas positifs.
- **F1-score** → équilibre entre précision et rappel.
- **AUC / ROC** → performance globale du classifieur.

Rappel : Hyperparameter tuning

Les **hyperparamètres** sont des choix de conception **meaningful**, faits avant l'entraînement, qui conditionnent la performance, la stabilité et la généralisabilité du modèle.

Exemples d'hyperparamètres :

- taux d'apprentissage (*learning rate*),
- profondeur maximale d'un arbre,
- nombre d'arbres ou de couches,
- taille des lots (*batch size*),

L'optimisation peut être réalisée via des méthodes telles que la **recherche sur grille (Grid Search)**, la **recherche aléatoire (Random Search)** ou l'**optimisation bayésienne**, en s'appuyant sur des métriques de performance adaptées.

Pourquoi les réseaux de neurones ?

- Limites des modèles linéaires : Cette hypothèse est souvent insuffisante pour modéliser des phénomènes complexes.
- Besoin de modèles capables d'approximer des fonctions complexes.

Théorème d'approximation universelle

Un réseau à une couche cachée peut approximer toute fonction continue (sous conditions).

Principe : empilement de couches de *neurones artificiels* reliés par des poids

Unité de base : neurone = combinaison linéaire + fonction d'activation non linéaire (ReLU, sigmoid, tanh)

$$z = \sum_{i=1}^n w_i x_i + b \quad , \quad a = \sigma(z)$$

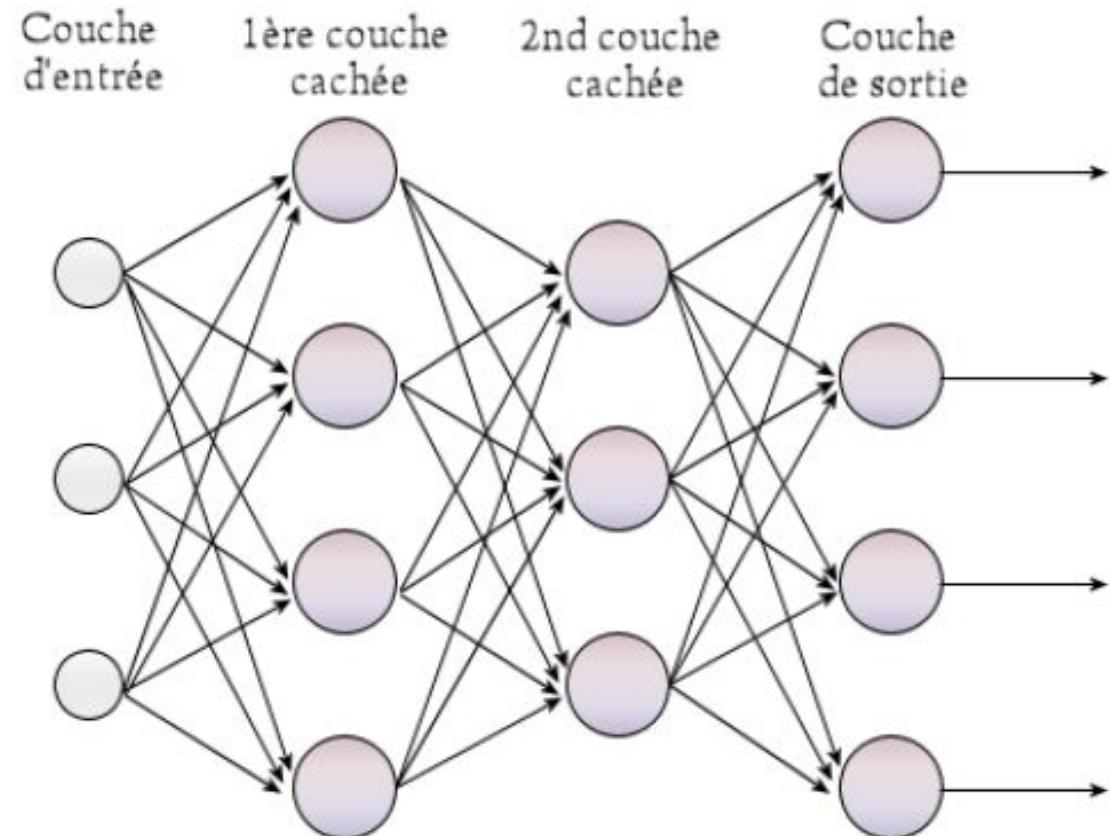
où w_i = poids, b = biais, σ = fonction d'activation

Architecture multi-couches

Le perceptron multicouche (*multilayer perceptron MLP* en anglais) est un type de réseau neuronal artificiel organisé en plusieurs couches.

Un perceptron multicouche possède au moins trois couches : une couche d'entrée, au moins une couche cachée, et une couche de sortie.

Chaque couche est constituée d'un nombre (potentiellement différent) de neurones.



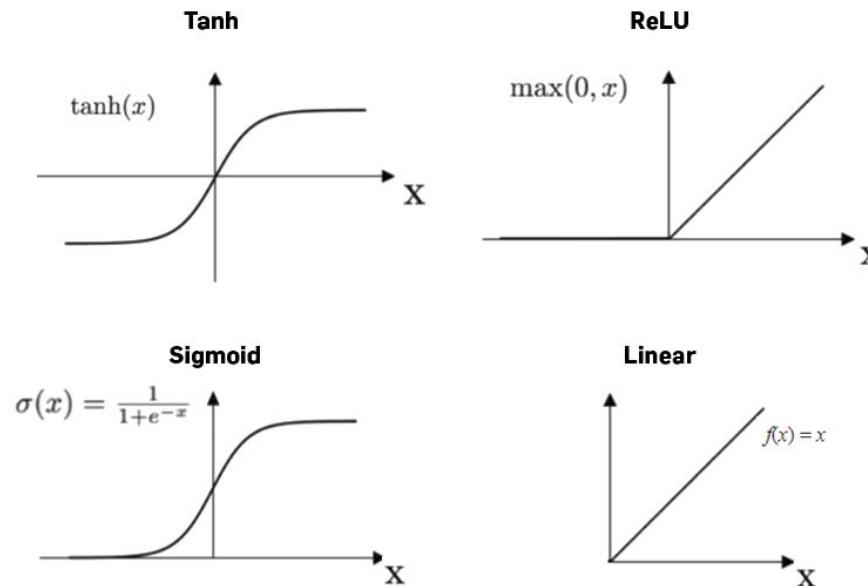
Pourquoi les fonctions d'activation sont cruciales ?

Sans fonction d'activation, un empilement de couches reviendrait à un seul modèle linéaire.

Les fonctions d'activation comme ReLU, sigmoid ou tanh introduisent des **seuils**, des **saturations** et des **zones de sensibilité**.

Cela permet au réseau de découper l'espace des données en régions complexes.

Les fonctions d'activation usuelles sont :



Architecture multi-couches

Couche d'entrée (l=0, n = nombre de neurones pour la couche l)

La première couche prend $x=[x_1, \dots, x_d]$ et calcule pour chaque neurone :

$$z_j^{(1)} = \sum_{i=1}^d W_{ji}^{(1)} x_i + b_j^{(1)}$$

$$a_j^{(1)} = \phi(z_j^{(1)})$$

- $j=1,2,\dots,n$ correspond aux **neurones de la couche l**
- $i=1,2,\dots,m$ correspond aux **neurones de la couche précédente** (ou d si c'est la couche d'entrée)

Chaque neurone de la couche cachée fait une **combinaison linéaire pondérée** de tous les éléments du vecteur d'entrée.

Ensuite, il applique une fonction d'activation non linéaire ϕ .

Le vecteur $a^{(1)}$ de dimension n devient la **nouvelle représentation des données**, transformée par cette couche.

Architecture multi-couches

Couches cachées ($l=1,2,\dots,L-1$)

Chaque neurone combine les sorties de la couche précédente et applique une fonction d'activation :

$$z_j^{(l)} = \sum_i W_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}$$

$$a_j^{(l)} = \phi(z_j^{(l)})$$

- W^l = matrice de poids (importance des entrées)
- b^l = biais (décalage du neurone)
- z^l = entrée du neurone avant activation

Chaque couche transforme progressivement les données pour **extraire des motifs plus abstraits**.

Couche de sortie ($I=L$)

Enfin, le vecteur d'activations de la dernière couche cachée $a(L-1)$ est transformé pour produire la prédiction finale :

$$\hat{y} = \mathbf{a}^{(L)}$$

Comment les données circulent dans le réseau ?

- Chaque neurone de la couche précédente envoie sa sortie à tous les neurones de la couche suivante (connexion complète)
- La couche suivante pèse et combine ces signaux pour produire ses propres sorties
- Les premières couches apprennent des caractéristiques simples, par exemple :
 - Dans une image : bords, lignes
 - Dans des données tabulaires : relations simples entre variables
- Les couches profondes combinent ces caractéristiques simples pour former des motifs plus complexes :
 - Dans une image : formes, textures, objets
 - Dans des données tabulaires : interactions complexes entre variables

Réseaux Convolutionnels

Principe des CNN

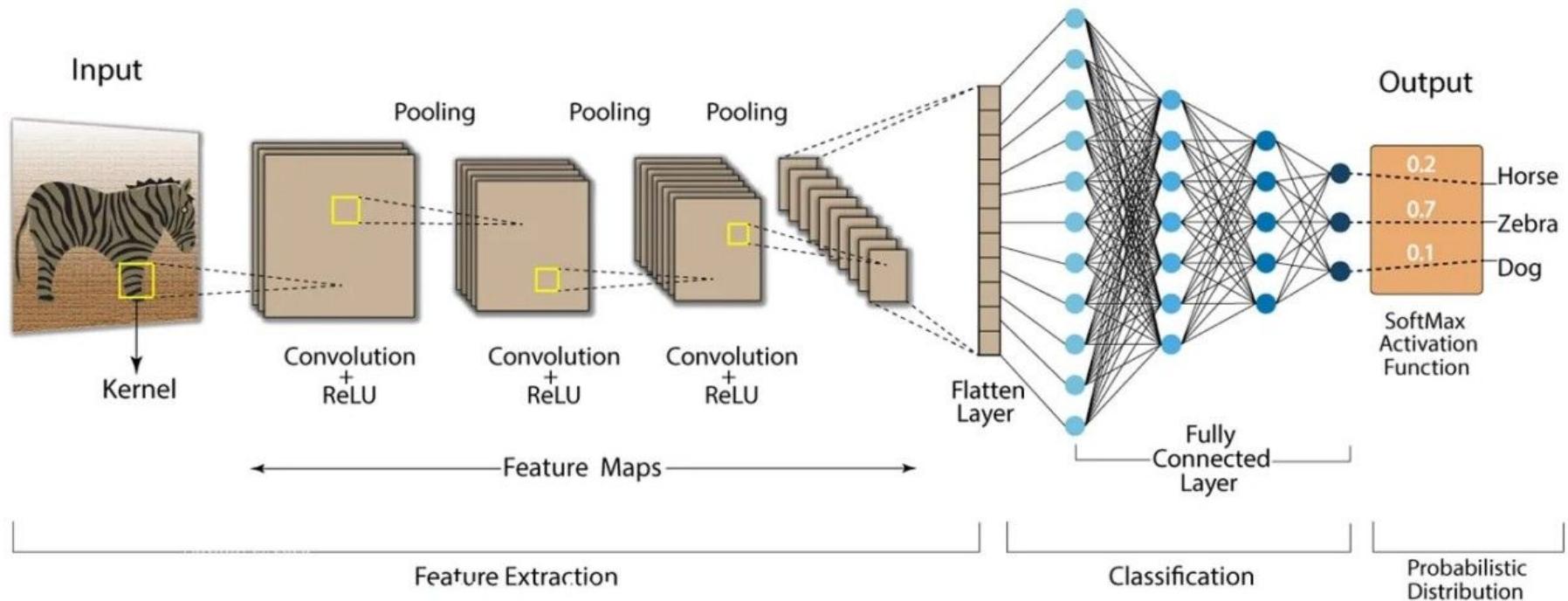
- Les CNN sont conçus pour traiter des données structurées spatialement, comme les images.
- L'idée : détecter des motifs locaux (bords, textures, formes) et les combiner pour construire des caractéristiques complexes.
- Contrairement aux MLP, ils partagent les poids et réduisent le nombre de paramètres.

Composants principaux

- **Couche de convolution** : Un filtre (kernel) glisse sur l'image (opération de convolution) et calcule une combinaison pondérée des pixels voisins.
- **Fonction d'activation** (Généralement ReLU)
- **Pooling / sous-échantillonnage**
Objectif : réduire la taille de la feature map, diminuer le nombre de paramètres et rendre la détection des motifs plus robuste aux déplacements dans l'image.
Exemple classique : **max pooling 2x2**
 - On divise la feature map en carrés 2x2
 - On conserve uniquement la **valeur maximale** dans chaque carré
- **Couche fully connected (dense)**
À la fin, les features extraites sont aplatis et envoyées dans des couches denses pour produire la prédiction finale.

Convolutional Neural Networks

Convolution Neural Network (CNN)



Principe des RNN

- Les réseaux de neurones classiques (MLP, CNN) traitent chaque entrée indépendamment des autres.
- Les RNN sont conçus pour traiter des données séquentielles : texte, séries temporelles, signaux audio...
- Les RNN ajoutent une mémoire interne pour prendre en compte le passé.

Composants principaux

Le RNN reçoit **l'entrée actuelle xt** et **l'état caché précédent $ht-1$** , qui contient la mémoire du passé. Il combine ces informations pour produire :

- **Un nouvel état caché ht** → mémoire mise à jour
- **Une sortie yt** → prédiction à cet instant

Chaque étape prend en compte **tout ce qui a été lu avant**, grâce à l'état caché ht .

Long Short Term Memory

Problème de disparition du gradient (*vanishing gradient*)

Les RNN traitent des séquences et ajustent leurs poids via backpropagation dans le temps. Chaque mise à jour dépend d'un produit de plusieurs dérivées (chaîne de dépendances à travers le temps).

- Ce produit de dérivées diminue très rapidement lorsqu'on remonte loin dans le temps.
- Résultat : les événements très anciens n'ont presque plus d'impact sur l'ajustement des poids.
- Autrement dit : le réseau oublie les informations lointaines.

Solution

- Utiliser des architectures conçues pour contourner ce problème :
 - LSTM (Long Short-Term Memory)
 - GRU (Gated Recurrent Unit)
- Ces modèles ont des mécanismes de portes qui permettent de conserver l'information plus longtemps.

Long Short Term Memory

Les LSTM introduisent une nouvelle unité appelée cellule mémoire, qui permet au réseau de stocker et d'accéder à des informations sur une période étendue.

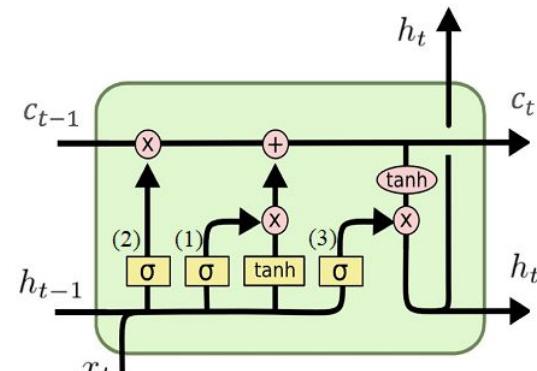
La cellule mémoire d'un LSTM est composée de plusieurs portes : une porte d'entrée, une porte de sortie et une porte d'oubli. Ces portes régulent le flux d'informations à l'intérieur de la cellule mémoire, permettant ainsi de contrôler les informations à retenir et celles à oublier.

h_t est l'état caché habituel des RNN mais dans les réseaux LSTM, il y a un deuxième état appelé c_t .

Ici, h_t représente la mémoire courte du neurone et c_t représente la mémoire à long terme.

Les LSTM sont conçus pour traiter des données séquentielles où le contexte passé est important.

1. Traduction automatique (Google Translate)
2. Génération de texte
3. Reconnaissance de la parole (Assistants vocaux de première génération)



LSTM
(Long-Short Term Memory)

Limites des LSTM

- Traitement séquentiel → entraînement lent
- Mémoire limitée → contexte court
- Difficulté à gérer de très longues séquences

Idée clé des Transformers

- Le Transformer lit toute la phrase d'un coup
Il regarde quels mots sont importants entre eux
- Il n'a pas besoin de mémoire étape par étape comme les LSTM
- Base des modèles modernes : ChatGPT, Gemini, Claude

Intuition

- Chaque mot “regarde” les autres mots.
- Il décide à quels mots faire attention.
- Plus un mot est important pour comprendre un autre, plus son poids est élevé. C'est la **self-attention**

CNN – Convolutional Neural Networks

```
import torch.nn as nn

model = nn.Sequential(
    nn.Conv2d(3, 32, kernel_size=3),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Flatten(),
    nn.Linear(32*15*15, 10))
```

RNN – Recurrent Neural Networks

```
model = nn.RNN(
    input_size=100,
    hidden_size=128,
    batch_first=True)
```

LSTM

```
model = nn.LSTM(
    input_size=100,
    hidden_size=128,
    batch_first=True)
```

Transformer

```
model = nn.Transformer(
    d_model=512,
    nhead=8,
    num_encoder_layers=6)
```

Partie 2 - IA responsable, KPI & reporting stratégique

1. **Équité (Fairness)** : éviter que le modèle ne favorise ou défavorise certains groupes.
2. **Transparence (Explainability)** : comprendre et expliquer les décisions du modèle.
3. **Régulation et conformité**
 - a. General Data Protection Regulation – UE - Règlement européen sur la protection des données personnelles, entré en vigueur en 2018.
 - b. AI Act - Projet de réglementation de l'UE pour encadrer les systèmes d'IA (en cours de mise en place, prévu pour 2024-2025).
 - c. Certaines juridictions hors UE ont leurs propres règles :
 - USA : loi sur l'égalité d'accès au crédit, directives FTC sur l'IA et la transparence
 - Singapour : Model AI Governance Framework pour une IA responsable
 - Canada : Directive sur la responsabilité algorithmique pour le secteur public

Équité et transparence en IA

Contexte	Problème	Impact	Solution / Correction	Référence / URL
COMPAS – Système de prédition de récidive (justice pénale USA)	Algorithme de risque évaluant la récidive donne des scores plus élevés aux défendeurs noirs malgré l'absence de race dans les variables	Risque accru d'incarcération et biais racial dans les décisions judiciaires	Audit, calibrage des scores, transparence du modèle, suivi d'équité	<i>How We Analyzed the COMPAS Recidivism Algorithm —</i> ProPublica (analyse des biais) http://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm
Amazon Hiring Algorithm (historique)	Système de recrutement pénalisait les CV contenant des mots associés aux femmes	Système abandonné après détection du biais, mais met en lumière le risque	Supprimer biais historique, diversifier données, mesurer fairness	Cas historique de biais d'IA dans les recrutements https://www.reddit.com/user/Secuodsoftpvtltd/comments/1p3l0m9/what_is_ai_ethics_and_bias_examples_how_to_build/?
IA en diagnostic médical / soins de santé	Algorithmes biaisés favorisent certains groupes (par ex. sous-représentation d'ethnies dans les données d'entraînement)	Soins inéquitables, erreurs de diagnostic pour certains patients	Rééquilibrage des données, suivi par groupe démographique, audit de biais	<i>Addressing AI Algorithmic Bias in Health Care (JAMA)</i> https://jamanetwork.com/journals/jama/article-abstract/2823006?

Équité (Fairness) en IA

- L'équité mesure si le **modèle prend des décisions impartiales** entre différents groupes, souvent définis par des attributs sensibles comme le genre, l'âge, l'origine ethnique, ou le revenu.
- En pratique, un modèle équitable doit **réduire ou éliminer les biais** qui pourraient causer une discrimination systématique.

Formes d'équité techniques :

1. **Demographic parity (parité démographique)** : La proportion de prédictions positives est **la même pour tous les groupes**.
 - Exemple : si 30 % des hommes obtiennent un prêt, 30 % des femmes doivent aussi l'obtenir, indépendamment des autres variables.
2. **Equalized odds (égalité des chances)** : Le modèle doit avoir **la même performance (Taux de vrais positifs / Taux de faux positifs) pour tous les groupes**.
 - Exemple : la précision pour prédire la récidive est la même pour les noirs et les blancs.
3. **Predictive parity (parité prédictive)**
 - La probabilité qu'une prédition positive soit correcte doit être **la même pour tous les groupes**.
 - Exemple : un score de risque de 0.8 correspond à 80 % de chance réelle d'événement, pour tous les groupes.

Équité (Fairness) en IA - Techniques pour corriger les biais

1. Pré-processing : rééquilibrer les données avant l'entraînement
 - a. **Reweighting** / pondération des exemples : donner plus de poids aux groupes minoritaires
 - b. Sur-échantillonnage (**oversampling**) : dupliquer ou générer artificiellement des données pour les groupes minoritaires (génération de données synthétiques via SMOTE Synthetic Minority Oversampling Technique).
 - c. Sous-échantillonnage (**undersampling**) : réduire la taille des groupes sur-représentés pour équilibrer les classes.
 - d. Suppression ou transformation de features sensibles
2. In-processing : introduire des contraintes de fairness directement dans l'algorithme
 - a. **Contrainte de fairness** dans la fonction de perte : ajouter un terme de pénalité pour réduire la différence de taux de prédiction entre groupes sensibles.
 - b. **Adversarial debiasing** : créer un “adversaire” qui essaie de prédire le groupe sensible à partir des prédictions du modèle. L’objectif est que le modèle principal ne contienne aucune information sur le groupe sensible.
 - c. **Fair regularization** / constraints : contraindre le modèle à respecter la parité des chances ou la parité prédictive pour tous les groupes.
3. Post-processing : ajuster les prédictions après l'entraînement pour corriger les biais
 - a. **Thresholding / calibration par groupe** : ajuster le seuil de décision pour chaque groupe afin d'aligner les taux de prédiction.
 - b. **Equalized odds post-processing** : modifier les prédictions pour que les TPR/FPR soient similaires pour tous les groupes.
 - c. **Reject option classification** : Prendre des décisions “neutres” (ex. prédiction aléatoire ou non-classée) pour les cas proches du seuil, surtout pour les groupes désavantagés.

Transparence / Explicabilité (Explainability)

Définition technique :

- La transparence (explainability) désigne la capacité à **comprendre et interpréter les décisions du modèle**, surtout pour des modèles complexes comme les réseaux neuronaux ou les ensembles (XGBoost, Random Forest).

Outils et méthodes techniques :

SHAP (SHapley Additive exPlanations)

- ❑ Basé sur la théorie des jeux : attribue une valeur à chaque feature pour expliquer son impact sur la prédiction.
- ❑ Fonctionne localement (une prédiction spécifique) ou globalement (importance moyenne des features).

Pour une observation x , la prédiction est décomposée comme :

$$f(x) = \mathbb{E}[f(x)] + \sum_{i=1}^d \phi_i$$

où :

- $\mathbb{E}[f(x)]$ = prédiction moyenne du modèle (baseline)
- ϕ_i = valeur SHAP de la feature i
- d = nombre de features

Transparence / Explicabilité (Explainability)

❑ LIME (Local Interpretable Model-agnostic Explanations)

- ❑ LIME vise à expliquer une prédiction individuelle d'un modèle complexe en l'approximant localement par un modèle interprétable (régression linéaire, arbre peu profond).
- ❑ Permet d'identifier quelles features influencent le plus une décision particulière.

Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin, “Why Should I Trust You?: Explaining the Predictions of Any Classifier”

❑ Feature importance / Partial Dependence Plots

- ❑ Mesure directement l'impact des variables sur la sortie du modèle.
 - ❑ **Importance basée sur l'arbre (Tree-based models)** : mesure la réduction de l'impureté (Gini, MSE) apportée par chaque feature dans les splits des arbres. (Exemple : Random Forest, XGBoost, LightGBM).
 - ❑ **Permutation Feature Importance** : Dégrade la performance du modèle en permuttant les valeurs d'une feature. La chute de performance indique l'importance de la feature.

Objectif

Assurer que les modèles restent fiables, équitables et performants après leur déploiement.

Pourquoi le drift est critique

Un modèle suppose que les données en production suivent la même distribution que les données d'entraînement.

Quand cette hypothèse est violée → perte de performance, biais émergents, décisions erronées.

1. **Feature Drift (Covariate Drift)** : la distribution des variables d'entrée X change dans le temps.
2. **Target drift** : la distribution de la variable cible y change.
3. **Concept Drift (le plus critique)** : la relation entre les features et la cible change. Le modèle devient conceptuellement faux.

- **ROI = Retour sur Investissement :**

Mesure financière de la valeur générée par le modèle par rapport aux coûts de développement et déploiement.

- **KPI = Key Performance Indicator**

Mesure quantitative permettant de suivre la performance d'un modèle IA par rapport à des objectifs business.

- Nombre d'erreurs évitées
- Temps moyen de réponse amélioré

Métrique technique	Interprétation business	Exemple concret
Précision / Rappel	Réduction d'erreurs ou alertes inutiles	Moins de faux positifs = économies sur détection fraude
F1-score	Équilibre entre erreurs manquées et alertes inutiles	Optimisation du diagnostic médical → plus de patients correctement identifiés
AUC	Capacité du modèle à distinguer classes	Prévention de défauts clients → baisse du risque financier
Drift détecté	Changement du marché ou du comportement utilisateur	Alerte pour réentraîner modèle de prévision des ventes

Atelier pratique 4

Pour chaque concept étudié pendant le cours (Drift au choix, SHAP, LIME, SMOTE), **par groupe de 3/4 élèves:**

- Utiliser un jeu de données réel de votre choix (de préférence simple)
- Mettre en œuvre la méthode associée avec une explication
- Présenter un code simple et reproductible
- Interpréter les résultats

Livrables

- Notebooks sous format pdf ou html (max 4 pages)
- Présentation orale : 5 min

Neural Networks

- Neural networks survey & comparison (CNN, RNN, LSTM, GRU, etc.) - <https://arxiv.org/abs/2305.17473>
- Intro aux réseaux de neurones (PDF cours) - <https://www.infor.uva.es/~teodoro/neuro-intro.pdf>

Tutorial PyTorch

- CNN - https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- LSTM - https://docs.pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html
- Transformer - <https://nlp.seas.harvard.edu/2018/04/03/attention.html>

Fairness

- Bias & fairness in machine learning -
<https://timkimutai.medium.com/bias-and-fairness-in-machine-learning-a-beginners-guide-to-building-models-that-don-t-play-c9a503c3c78b>
- Fairness explainable ML paper - <https://www.kdd.org/kdd2016/papers/files/rfp0573-ribeiroA.pdf>