Rapport : Intelligence Artificielle et Machine Learning

Table des matières

Introduction

Le Machine Learning (ML) s'impose désormais comme un levier du développement logiciel et des métiers de la data. Dans les produits digitaux, il ne s'agit plus seulement d'écrire du code déterministe : on conçoit, entraîne et opère des modèles statistiques qui apprennent des données pour prendre des décisions (prédire, classer, recommander). Cette évolution transforme l'architecture des systèmes, les pratiques d'ingénierie, la gouvernance de l'information et les responsabilités des équipes.

Ce rapport analysera, dans cette perspective, les principes et enjeux du ML pour les développeurs et les professionnels de la data. D'abord, il clarifie les notions souvent confondues d'Intelligence Artificielle (IA), de Machine Learning (ML) et de Deep Learning (DL), afin de poser un vocabulaire partagé et des frontières utiles selon les besoins métier, les volumes de données et les contraintes de calcul.

Nous présenterons ensuite les trois grands types d'apprentissage supervisé, non supervisé et par renforcement à travers des exemples concrets orientés "dev & data", pour relier objectifs, algorithmes et métriques d'évaluation à la valeur produit.

La suite détaille les étapes essentielles du cycle de vie d'un projet ML : collecte et qualité des données, préparation et création de variables, entraînement et validation, évaluation orientée métier, puis déploiement et exploitation en production, avec les pratiques d'industrialisation (MLOps) nécessaires à la fiabilité, à la reproductibilité et à l'observabilité des modèles.

Pour terminer, nous discuterons des limites et risques actuels, biais et équité, surapprentissage, consommation énergétique, interprétabilité et sécurité des données, avant d'ouvrir sur une réflexion personnelle quant à l'impact du ML sur notre futur métier : compétences à acquérir, collaboration, arbitrages entre performance et coûts.

1) Clarification entre IA, Machine Learning et Deep Learning

Intelligence Artificielle (IA)

L'intelligence artificielle désigne un champ très large qui cherche à concevoir des systèmes capables d'accomplir des tâches normalement associées à l'intelligence humaine : raisonner, planifier, percevoir, comprendre le langage ou encore agir dans un environnement. Elle recouvre à la fois des approches dites « symboliques », basées sur des règles, de la logique ou des moteurs d'inférence, et des approches plus numériques, fondées sur l'apprentissage statistique. C'est dans cette dernière famille que s'inscrit le Machine Learning.

Machine Learning (ML)

Le Machine Learning est un sous-domaine de l'IA où l'on ne programme pas toutes les règles à la main : le système apprend à partir des données, en construisant des modèles capables de prédire ou de décider. On distingue trois grands types d'apprentissage en commençant par le supervisé, non supervisé et par renforcement. Ces méthodes s'avèrent particulièrement efficaces pour exploiter des données tabulaires, des séries temporelles ou pour résoudre des problèmes prédictifs classiques comme la régression et la classification.

Deep Learning (DL)

Le Deep Learning est lui-même un sous-domaine du Machine Learning, qui repose sur des réseaux de neurones profonds composés de plusieurs couches. Ces architectures apprennent automatiquement des représentations complexes à partir des données et excellent dans le traitement de données non structurées, comme les images, l'audio ou le texte. Le Deep Learning est aujourd'hui dominant dans les domaines de la perception et du langage, mais il demande généralement beaucoup de données, une puissance de calcul importante et une attention particulière lors du déploiement opérationnel.

2) Les principaux types d'apprentissage

a) Apprentissage supervisé

Le principe de l'apprentissage supervisé repose sur des exemples annotés : on dispose de données d'entrée décrites par leurs caractéristiques et d'une cible connue (par exemple un prix, une catégorie, un label). L'objectif est d'apprendre une fonction f(x) capable de généraliser, c'est-à-dire de prédire correctement la valeur ou la classe sur de nouvelles données jamais vues.

Parmi les algorithmes courants, on retrouve la régression linéaire ou logistique, les SVM, les arbres de décision, les forêts aléatoires, les techniques de gradient boosting (XGBoost, LightGBM, CatBoost), les k plus proches voisins (k-NN), ou encore les réseaux de neurones peu profonds adaptés aux données tabulaires.

Au niveau des métriques clés, on a :

- **Classification** : Accuracy, précision/Recall/F1, ROC-AUC, PR-AUC et une matrice de confusion
- **Régression** : RMSE, MAE, MAPE, R^2.
- **Métier** : coût/recette attendus, taux de faux positifs toléré, SLA de latence.

Exemples concrets :

- **Churn d'abonnés** (télécom, SaaS) : prédire le risque de résiliation à 30j ; actions ciblées (offre de rétention).
- **Prévision de la demande** (retail, e-commerce, grande distribution) : estimer les ventes par magasin/jour pour ajuster les stocks et approvisionnement.
- **Scoring de risque** (banque/assurance) : probabilité de défaut de paiement ; calibrage des seuils vs contraintes réglementaires.
- **Détection de fraude** (paiement en ligne) : classification binaire temps réel ; gestion fort déséquilibre (0,1-1% de fraudes). (Exemple : BPN Paribas Finance –> Annexe lien n°1)

- **Estimation des délais de livraisons** (logistique/3PL) : régression sur délais prévus selon origine, destination, trafic, météo.

b) Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, il n'y a ici pas de cible étiquetée. L'objectif est de laisser l'algorithme découvrir par lui-même des structures cachées dans les données : des regroupements naturels, des facteurs communs, des proximités ou encore des anomalies.

Il y a plusieurs familles d'algorithmes qui sont utilisées. Par exemple, le clustering où on retrouve le k-means, le k-medoids, ou encore DBSCAN et HDBSCAN qui gèrent des formes de clusters non convexes. Les modèles de mélange gaussien (GMM) permettent une approche probabiliste des regroupements. Pour réduire la dimensionnalité et faciliter l'exploration ou la visualisation, on utilise des techniques comme l'ACP (PCA), UMAP ou t-SNE. Les règles d'association (Apriori, FP-Growth) permettent d'identifier des co-occurrences fréquentes, utiles en e-commerce. Enfin, pour la détection d'anomalies, on peut recourir à Isolation Forest, One-Class SVM ou à des auto-encodeurs, parfois en lien avec le Deep Learning.

Au niveau des métriques clés d'évaluations c'est plus complexe qu'en apprentissage supervisé puisqu'il n'existe pas de « vérité terrain » explicite. On s'appuie sur des métriques internes comme le coefficient de silhouette, l'indice de Davies-Bouldin ou l'inertie pour k-means. Du point de vue métier, la cohérence des segments se mesure aussi à travers des indicateurs concrets : taux de conversion, panier moyen ou encore coût d'investigation lié aux anomalies détectées.

Exemples concrets :

- **Segmentation client** (retail, banque, assurance, énergie) : regrouper les clients par comportements d'achat/usage pour adapter offres et messages.
- **Détection d'anomalies IoT** (industrie/énergie) : repérer des séries capteurs atypiques pour maintenance préventive.
- **Assortiment & recommandation "à froid"** (e-commerce) : rapprocher des produits par caractéristiques ou co-achats pour organiser le catalogue.
- **Analyse de texte francophone** (SAV, réseaux sociaux) : réduction de dimension (PCA/UMAP) sur embeddings pour cartographier thèmes/intentions.

c) Apprentissage par renforcement (RL)

Le principe de l'apprentissage par renforcement s'inspire du comportement d'un agent qui apprend en interagissant avec son environnement. À chaque étape, l'agent se trouve dans un état « ***s*** », choisit une action « ***a*** », reçoit une récompense « ***r*** » en retour et observe un nouvel état. Le but est d'apprendre une stratégie qui maximise la

récompense cumulée à long terme ou alors, vu autrement, qui minimise le regret lié aux mauvaises décisions.

Comme pour les apprentissages précédents, celui-là aussi possèdent plusieurs variantes. Les bandits multi-bras et leurs versions contextuelles permettent de gérer des choix séquentiels simples, comme tester plusieurs actions et garder celles qui rapportent le plus. Le RL tabulaire, avec des méthodes comme le Q-learning, fonctionne bien pour des environnements de taille modeste. Enfin, le reinforcement learning profond (DQN, PPO, SAC) s'appuie sur des réseaux de neurones et peut traiter de vastes espaces d'états et d'actions, comme ceux rencontrés dans la robotique ou la gestion de ressources.

Les performances d'un agent sont évaluées via la récompense moyenne obtenue, le regret accumulé, le respect de la politique hors du contexte d'entraînement, ou encore sa capacité à respecter des contraintes de coût, de risque et stabilité durant l'apprentissage.

Il y a une chaîne YouTube très connue sur le sujet, qui utilise le personnage d'Albert dont son but est d'apprendre à s'échapper, dans une autre vidéo à marcher, avec différents niveaux. (Annexe lien n°2)

Exemple concret :

- **Budget publicitaire et personnalisation** : des bandits contextuels peuvent allouer dynamiquement les meilleures variantes de messages ou d'offres par segment, ce qui offre plus de flexibilité qu'un A/B test classique.
- **Orchestration de ressources cloud** : des politiques d'auto-scaling ajustent automatiquement le nombre de serveurs en fonction de la charge, tout en équilibrant coût et latence.
- **Pricing dynamique "light"** (transport, billetterie, e-commerce) : des modèles ajustent progressivement les prix, dans le respect des contraintes réglementaires et d'équité.
- **Priorisation de tickets de support client** : un agent peut décider de l'ordre de traitement des demandes, afin de réduire le temps d'attente global en tenant compte de la criticité de chaque cas.

3) Cycle de vie d'un projet ML & MLOps

a) Cadrage et collecte de données

Pour le cadrage métier :

- Il faut définir l'unité de prédiction (client, commande…), l'horizon (J+7, M+1), la cible (churn oui/non, montant), les contraintes (latence, budget, conformité), les KPI de succès (ex. uplift de rétention, MAE sur ventes).
- Formaliser la matrice de coûts (FP/FN) pour guider les seuils et l'évaluation.

Pour les sources et la gouvernance :

- Inventorier les sources (logs, IoT, etc…), les droits d'accès, la traçabilité (lignage).
- Qualité de données : complétude, exactitude, fraîcheur, unicité. Mettre en place des tests de validation de schéma et des règles d'alerte.
- Conformité & éthique : minimisation des données, consentement, anonymisation/pseudonymisation, conservation. (RGPD)

b) Préparation et feature engineering

La première étape consiste à fiabiliser les données en passant par le nettoyage : éliminer les doublons, traiter les valeurs manquantes en les imputant de manière appropriée, gérer les valeurs aberrantes (par exemple avec la winsorisation), et appliquer une normalisation.

Ensuite, il y a les variables catégorielles qui peuvent être transformées par des méthodes comme le one-hot encoding ou le target encoding, en veillant à éviter toute fuite d'information. Pour les dates et séries temporelles, on crée souvent des variables liées aux saisonnalités, des décalages (lags) ou encore des fenêtres mobiles. Côté agrégations, il est utile de générer des indicateurs multi-niveaux (par client, produit ou magasin), des comptages ou des ratios. Si du texte ou des images sont présents, on peut extraire des représentations vectorielles (embeddings NLP ou vision), mais aussi des caractéristiques plus simples comme des longueurs de texte ou alors des scores de base.

A savoir aussi que lorsque les classes sont déséquilibrées (fraude vs non fraude, churn vs non churn), ils existent différentes techniques : par exemple, ajuster les poids de classes, appliquer du sur- ou sous-échantillonnage, choisir des métriques adaptées comme la PR-AUC ou le F1-score. Il est aussi possible d'adapter les seuils de décision en fonction des besoins métier.

c) Entraînement et validation

Baselines & sélection d'algorithmes :

- Toujours comparer aux baselines (moyenne/majoritaire, modèle simple).
- Commencer par des modèles simples/interprétables (régression/arbres) avant d'escalader (GBDT, NN).

Validation & tuning:

- Cross-validation (k-fold, stratified, time series split).

- Recherche d'hyperparamètres (grid/random/bayesian), early stopping, régularisation.
- Reproductibilité : graine aléatoire, versions de données/code/librairies, suivi des expériences (params, métriques, artefacts).

Fiabilité & équité :

- Stress tests hors distribution, perturbations réalistes.
- Évaluer par sous-populations (équité), calibration des probabilités.

d) Evaluation (orientée métier)

Comme dit précédemment, voici les métriques d'évaluations, en commençant par les :

Métriques techniques :

- Classification : ROC-AUC, PR-AUC (déséquilibre), précision/recall/F1, calibration (Brier, reliability plot).
- Régression : MAE/RMSE, MAPE, pinball loss (quantiles).
- Séries temporelles : sMAPE, MASE, backtesting glissant.

Métriques business et seuils :

- Traduire en gains/pertes attendus (coût FP/FN, valeur par décision).
- Optimiser le seuil pour la valeur.
- Évaluer l'impact par expérimentations quand c'est possible.

Explicabilité :

- Importance globale (gain/impureté), SHAP/LIME pour les cas individuels.
- Documentation des limites et domaines d'inapplicabilité.

e) Déploiement et intégration

Patterns de serving :

- Batch (scores nocturnes, enrichissement DWH).
- Temps réel (API REST/gRPC) avec SLA de latence.
    - Streaming (par exemple : file d'événements) si décisions séquentielles.

Chaîne d'artefacts :

- Registry de modèles (version, signature d'entrées/sorties), empaquetage (container), vérifications d'intégrité.
- Parité features offline/online (mêmes calculs, mêmes fenêtres).

Mise en production sûre :

- Shadow mode (lecture seule), canary/blue-green, rollback.
- Tests de contrat (schémas d'entrée), tests de performance (p95/p99), tests de sécurité.

Intégration produit :

- Définir la boucle d'action (qui consomme la prédiction, quels écrans/API, quelles décisions).
- Journaliser features, prédictions, décisions et outcomes pour le monitoring et le ré-entraînement.

Le schéma conseillé dans la logique serait : Data → Features → Entraînement/Validation → Evaluation → Déploiement.

5) Limites et risques actuels

a) Biais

Un modèle apprend ce qu'on lui montre. Si l'historique est incomplet, déséquilibré ou reflète des pratiques inéquitables, il reproduira ces biais parfois en silence. En gros, un score défavorise certains profils, une recommandation ignore des minorités. L'antidotes pour prévenir : échantillons plus représentatifs, audits par sous-populations, métriques d'équité (écart de précision/recall), revue humaine sur les cas sensibles, et surtout des boucles de feedback pour corriger le tir.

b) Surapprentissage (overfitting)

Le modèle "connait son cours par cœur" mais rate le contrôle. Signes typiques : score parfait en entraînement, décevant en test ou en production. Les causes : trop de complexité, fuite de données, split mal fait. Les remèdes : validation temporelle, régularisation, early stopping, simplifier quand c'est possible, et surveiller en prod (drift) avec des tests hors distribution. Un modèle un peu moins "brillant" mais plus stable a plus de valeur.

c) Consommation énergétique

Entraîner et servir des modèles coûte de l'argent et du carbone. Les gros réseaux (et itérations multiples) pèsent très vite lourd. Le bon réflexe côté équipes : sobriété par design. Commencer simple, mesurer le coût par 1 000 prédictions, préférer des modèles compacts sur données tabulaires (GBDT, régressions), puis optimiser, mutualiser les features, et adapter l'architecture (batch vs temps réel) au besoin réel de fraîcheur.

d) Interprétabilité

Pour décider, corriger, et se conformer (clients, métier, régulateur), il faut comprendre

pourquoi le modèle répond comme ça. Les modèles simples sont lisibles, les plus puissants demandent des explications post-hoc (SHAP/LIME), à manier avec prudence. En utilisant une approche pragmatique, documenter ce que le modèle sait ou ne sait pas faire, afficher les facteurs clés au moment de la décision, prévoir une voie d'escalade humaine, et accepter qu'"un point de F1" n'achète pas toujours la confiance.

Conclusion

Je retiens que mon futur métier ne consistera plus seulement à écrire du code, mais à prendre des décisions guidées par les données, en étant responsable de leurs effets sur les utilisateurs et sur l'entreprise. Le Machine Learning m'oblige à penser au-delà du résultat immédiat : la qualité des données, l'équité des décisions, l'impact environnemental, la clarté de ce que je livre.

A savoir que dès ma deuxième année d'alternance au sein du groupe Bouygues Telecom Business Solution, je vais commencer à mettre tout ça en pratique. Au début, on me mettra sur des petits cas concrets, bien cadrés, pour apprendre vite. Je travaillerai auprès des équipes produit et data. L'objectif est de prendre les bonnes habitudes dès maintenant, afin d'avoir une progression sereine.

Annexe

BNP Paribas PF limite la fraude grâce à l'IA

https://www.republik-it.fr/decideurs-it/cas-usage/pratiques/bnp-paribas-pf-limite-la-fraude-grace-a-l-ia.html

AI Agent Learns to Escape (deep reinforcement learning)

https://www.youtube.com/watch?v=v3UBlEJDXR0

AI Learns to Walk (deep reinforcement learning)

https://www.youtube.com/watch?v=L_4BPjLBF4E