

Intelligence artificielle pour product owner

Redha Moulla

Bordeaux, 20-22 octobre 2025

Plan de la formation

- Qu'est-ce que l'intelligence artificielle ?
- Machine learning
- Deep learning
- Vision par ordinateur
- Traitement automatique du langage
- Intelligence artificielle générative
- Industrialisation des modèles de machine learning
- Enjeux éthiques de l'IA

Qu'est-ce que l'intelligence artificielle ?

Définition littérale de l'intelligence artificielle

1. Intelligence

Ensemble des fonctions mentales ayant pour objet la connaissance conceptuelle et relationnelle.

- Larousse

2. Artificielle

Qui est produit de l'activité humaine (opposé à la nature).

- Larousse

Qu'est-ce que l'intelligence ?

La notion d'intelligence recouvre plusieurs facultés cognitives :

- ① **Raisonnement** : La capacité à résoudre des problèmes et à faire des déductions logiques.
- ② **Apprentissage** : L'aptitude à acquérir de nouvelles connaissances et à s'améliorer grâce à l'expérience.
- ③ **Perception** : La compétence pour reconnaître et interpréter les stimuli sensoriels.
- ④ **Compréhension** : L'habileté à saisir le sens et l'importance de divers concepts et situations.
- ⑤ **Mémorisation** : La faculté de stocker et de rappeler des informations.
- ⑥ **Créativité** : Le pouvoir d'inventer ou de produire de nouvelles idées, de l'originalité dans la pensée.

Mais est-ce que l'intelligence est réductible à des facultés mesurables ?

Le mythe des servantes d'or

“Si chaque instrument était capable, sur une simple injonction, ou même pressentant ce qu'on va lui demander, d'accomplir le travail qui lui est propre, comme on le raconte des statues de Dédale ou des trépieds d'Héphaïstos, lesquels, dit le poète, : “Se rendaient d'eux-mêmes à l'assemblée des dieux”, si, de la même manière, les navettes tissaient d'elles-mêmes, et les plectres pinçaient tout seuls la cithare, alors, ni les chefs d'artisans n'auraient besoin d'ouvriers, ni les maîtres d'esclaves.”

— Aristote

Conférence de Dartmouth ?

Articles

AI Magazine Volume 27 Number 4 (2006) © AAAI

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

August 31, 1955

John McCarthy, Marvin L. Minsky,
Nathaniel Rochester,
and Claude E. Shannon

The 1956 Dartmouth summer research project on artificial intelligence was organized by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The report contains 17 papers, plus a title page. Copies of the report can be found at the Dartmouth College library. The first 5 pages state the purpose of the study and give the names and interests of the four who proposed the study. In the interest of brevity, this article summarizes the main points of the report, and gives bibliographical statements of the papers.

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College, Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use lan-

guage, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

1. Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. It may be speculated that present computers may be insufficient to simulate many of the higher functions of the mind. This is not because of any lack of machine capacity, but our inability to write programs taking full advantage of what we have.

2. How Can a Computer be Programmed to Use a Language

It may be speculated that a large part of human language consists of sequences of words according to rules of reasoning and rules of conjecture. From this point of view, learning a generalization consists of admitting a new

"We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."

L'intelligence artificielle selon John McCarthy

"It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."

— John McCarthy

Le Test de Turing

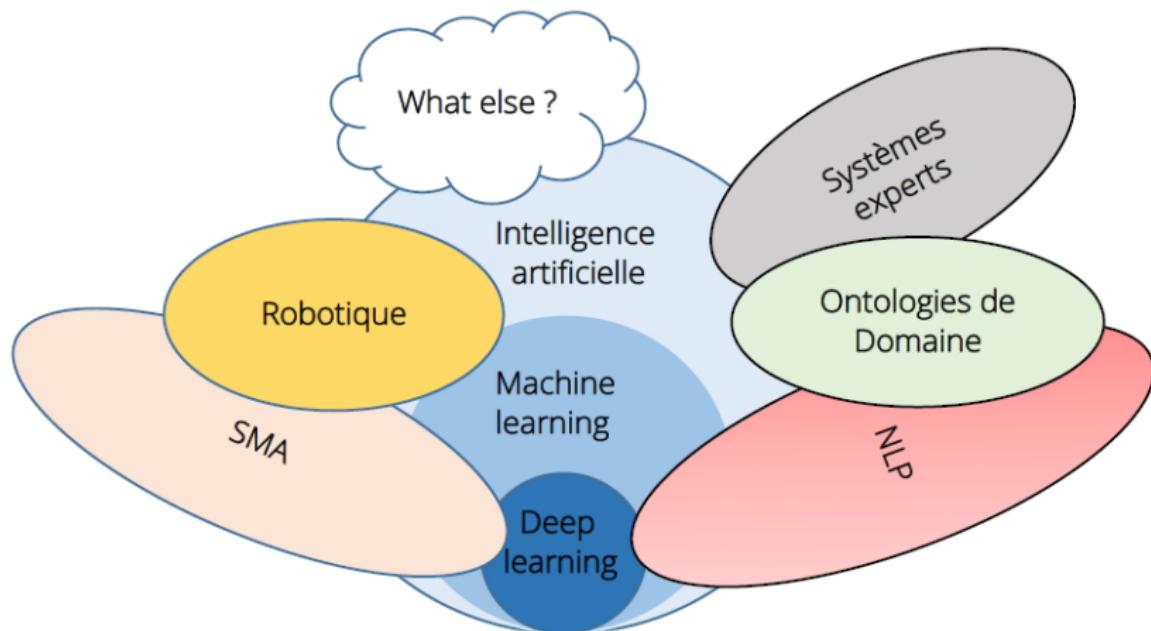
Le Test de Turing, développé par Alan Turing en 1950, est une tentative de mesurer l'intelligence d'une machine, plus précisément de la faculté d'une machine à penser. Cette dernière n'étant pas si évidente à mesurer, le test substitute finalement à la faculté de penser celle de traiter le langage naturel comme un humain.

Les points clés du Test de Turing sont :

- Un interrogateur humain engage une conversation avec un humain et une machine, chacun étant caché de la vue de l'interrogateur.
- Si l'interrogateur ne peut pas déterminer systématiquement quelle est la machine, celle-ci est considérée comme ayant passé le test.
- Le test ne mesure pas la connaissance ou la capacité à être vérifiable, mais plutôt la capacité de reproduire le comportement humain.

Définition pragmatique de l'intelligence artificielle

Il s'agit d'un ensemble de techniques qui permettent à la machine d'accomplir des tâches qui requièrent traditionnellement une intelligence humaine.



IA forte vs IA faible

La distinction entre IA forte et IA faible se réfère à deux approches conceptuelles différentes dans le domaine de l'intelligence artificielle.

IA faible :

- Aussi connue sous le nom d'IA "étroite", elle est conçue pour effectuer des tâches spécifiques et ne possède pas de conscience.
- Les systèmes d'IA faible agissent et réagissent uniquement en fonction des instructions programmées et des algorithmes spécifiques.
- Exemples : assistants virtuels, systèmes de recommandation, reconnaissance vocale.

IA forte :

- Vise à créer des machines dotées de conscience, de compréhension et d'esprit, similaires à l'intelligence humaine.
- L'IA forte serait capable d'apprendre, de raisonner, de résoudre des problèmes et de prendre des décisions indépendamment.
- À ce jour, l'IA forte reste un objectif à atteindre, qui fait l'objet de recherches intensives.

IA connexionniste vs IA symbolique

Intelligence artificielle symbolique :

Systèmes basés sur des règles et des symboles pour imiter le raisonnement humain.

- Logique
- Ensemble de règles
- Orientée connaissance

Intelligence artificielle connexionniste

: Modèles inspirés du cerveau humain pour apprendre des tâches à partir de données.

- Probabiliste
- Apprentissage machine
- Orientée données

Machine learning

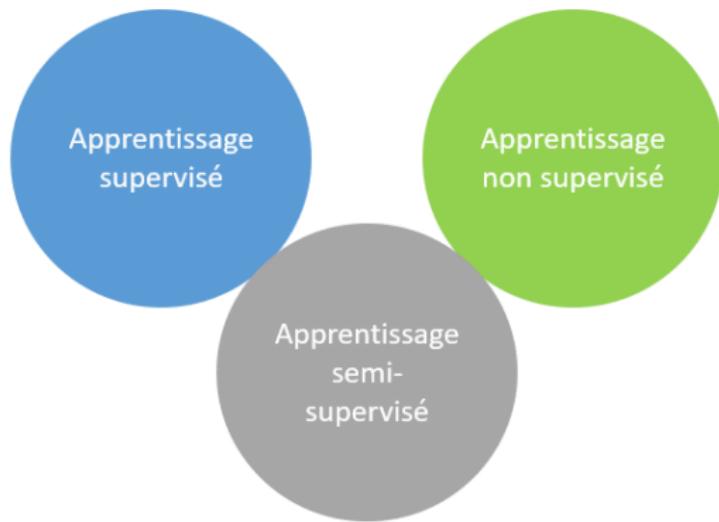
Définition de l'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui consiste à doter les machines de la capacité d'apprendre à partir de données sans que celles-ci ne soient explicitement programmées pour exécuter des tâches spécifiques.

Le machine learning englobe plusieurs types d'apprentissage :

- **Supervisé** : Les algorithmes apprennent à partir de données étiquetées pour faire des prédictions ou classifications.
- **Non supervisé** : L'apprentissage est effectué sur des données non étiquetées pour trouver des structures cachées.
- **Semi-supervisé** : Combine des éléments des deux premiers types en utilisant une petite quantité de données étiquetées et une grande quantité de données non étiquetées.
- **Par renforcement** : Les modèles apprennent à prendre des décisions en maximisant une récompense à travers des interactions.

Typologies d'apprentissage automatique



L'apprentissage supervisé

L'apprentissage supervisé consiste à apprendre un modèle qui associe une étiquette (*label*) à un ensemble de caractéristiques (*features*).

- **Inputs** : un jeu de données *annotées* pour entraîner le modèle.
 - Exemple : des textes (tweets, etc.) avec les *sentiment* associés, positifs ou négatifs.
- **Output** : une étiquette pour un point de donnée inconnu par le modèle.

L'apprentissage supervisé se décline lui-même en deux grandes familles :

- **La classification** : prédire une catégorie ou une classe.
 - Exemple : prédire l'étiquette d'une image (chat, chien, etc.), le sentiment associé à un texte, le centre d'intérêt d'un client à partir de ses commentaires, etc.
- **La régression** : prédire une valeur continue (un nombre réel typiquement).
 - Exemple : prédire le prix d'un appartement, la lifetime value d'un client, etc.

Classification

Exemple de classification : credit scoring

Âge	Revenu annuel (k€)	Historique de crédit	Nombre de cartes de crédit	Niveau d'éducation	Propriétaire immobilier (Oui/Non)	Label (y)
30	50	Bon	2	Licence	Oui	Accepté
45	80	Moyen	3	Master	Oui	Accepté
22	20	Mauvais	1	Bac	Non	Refusé
35	60	Bon	4	Licence	Oui	Accepté
40	70	Moyen	2	Bac+2	Non	Refusé

Régression

Exemple de régression : prédition des prix des logements

Surface (m ²)	Nombre de chambres	Distance du centre-ville (km)	Année de construction	Quartier (Score 1-10)	Prix (k€) (y)
80	3	5	2010	8	300
120	4	10	2005	7	450
60	2	2	2020	9	200
150	5	15	1995	6	600
100	3	8	2015	7	400

L'apprentissage non supervisé

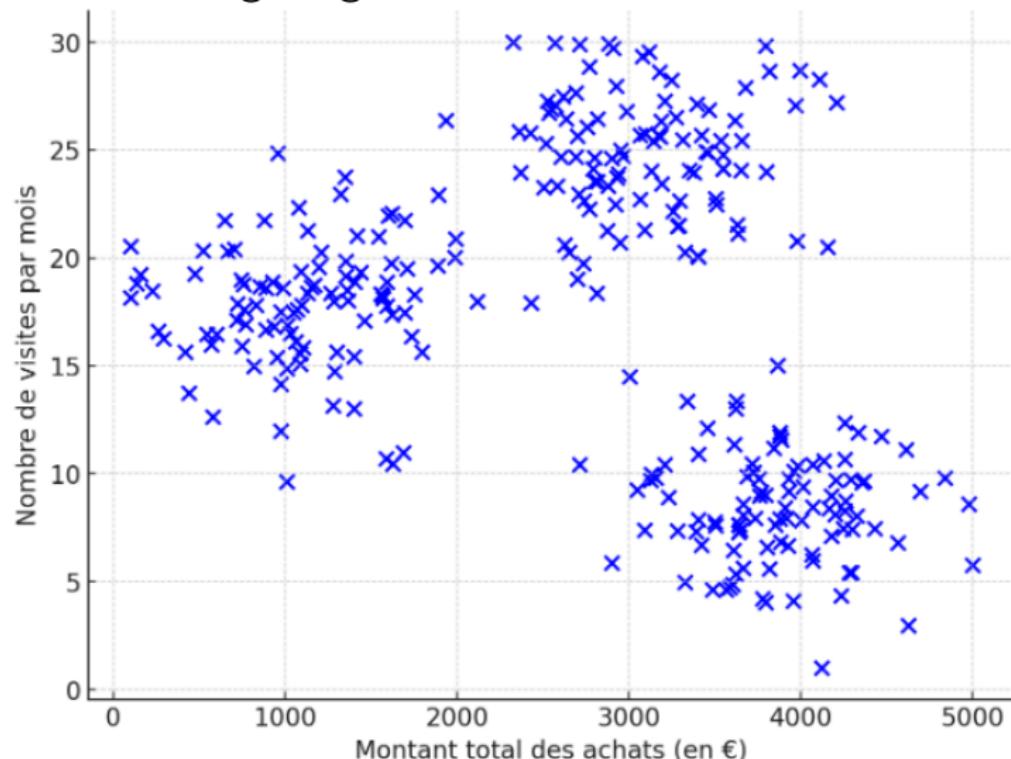
L'apprentissage non supervisé se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

Principales typologies de l'apprentissage non supervisé :

- **Clustering** : Regroupement de points de données similaires ensemble.
Exemple : segmentation de marché, regroupement social.
- **Détection d'anomalies** : Détecter des observations dont les caractéristiques sont inhabituelles par rapport à la majorité.

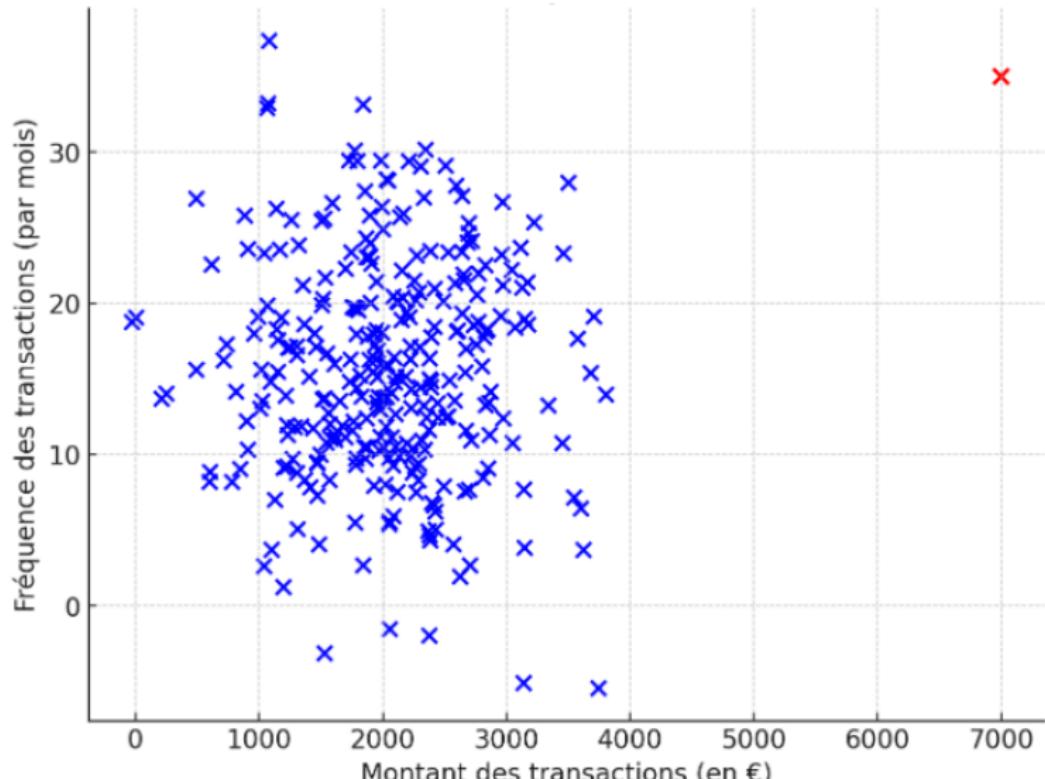
Clustering

Exemple de clustering : segmentation clients



Détection d'anomalies

Exemple de détection d'anomalies : fraude bancaire



L'apprentissage semi-supervisé

L'apprentissage semi-supervisé combine des éléments des approches supervisées et non supervisées. Il utilise un petit ensemble de données étiquetées et un plus grand ensemble de données non étiquetées pour former des modèles.

Cette méthode est particulièrement utile quand :

- Les données étiquetées nécessitent des ressources coûteuses pour les obtenir, mais les données non étiquetées sont abondantes.
- L'ajout d'un peu d'information étiquetée peut améliorer significativement la performance de modèles entraînés avec des données non étiquetées.

Les applications typiques incluent :

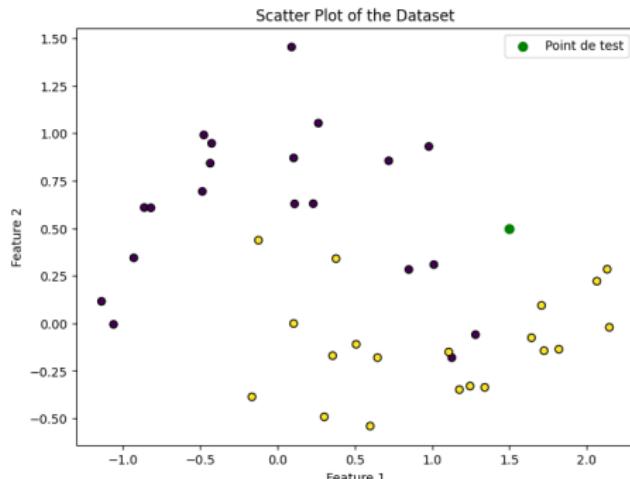
- Développement de systèmes de recommandation plus performants.
- Traitement de langage naturel et analyse de sentiment lorsque les annotations complètes ne sont pas disponibles.

L'apprentissage tente d'exploiter "le meilleur des deux mondes" de l'étiquetage et de la découverte de structure.

Apprentissage supervisé

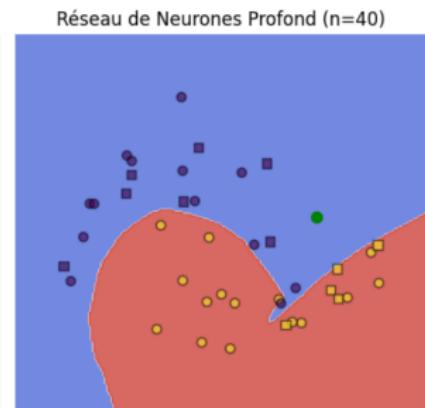
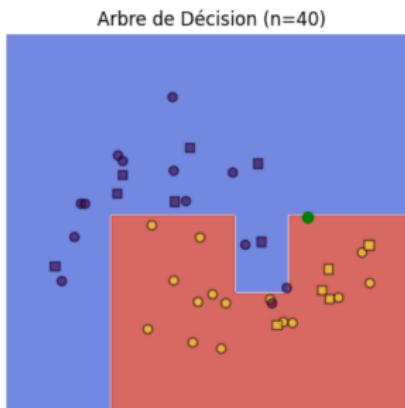
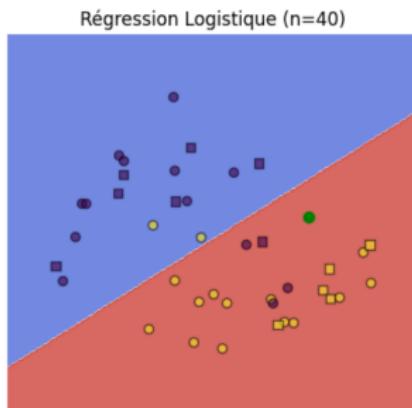
L'apprentissage supervisé comme un problème d'induction

- **Définition** : L'apprentissage supervisé consiste à apprendre une fonction f qui mappe les entrées X aux sorties y , à partir d'un ensemble d'exemples d'entraînement (X, y) .
- **Induction** : Le modèle induit une règle générale à partir de données particulières, dans le but de généraliser à de nouvelles instances.
- **Problème de généralisation** : Comment garantir que le modèle apprend une règle qui s'applique à de nouvelles données ?



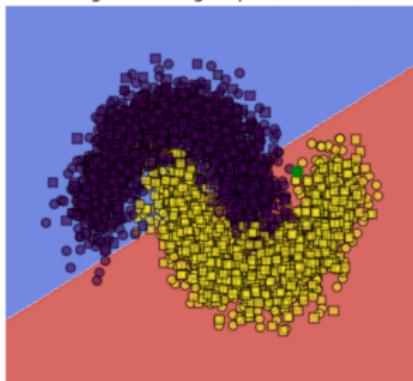
Une indétermination intrinsèque pour le choix du modèle

Il y a une infinité de manières d'induire un modèle à partir d'un échantillon de données d'entraînement.

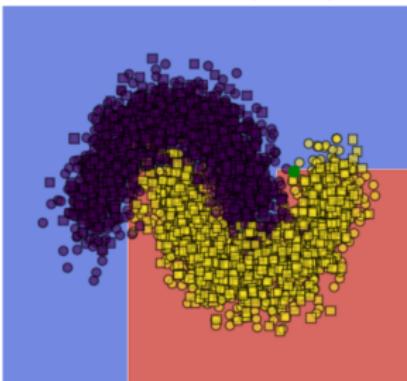


Sous-apprentissage et surapprentissage sur les données d'entraînement

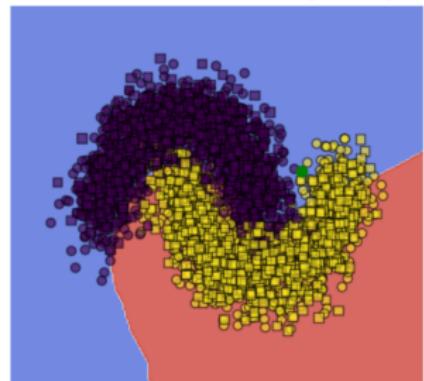
Régression Logistique (n=3000)



Arbre de Décision (n=3000)



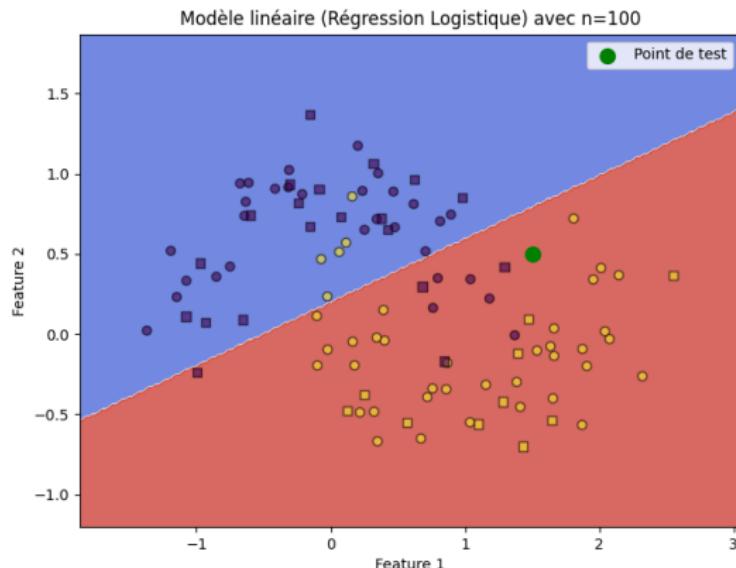
Réseau de Neurones Profond (n=3000)



Sous-apprentissage

Definition

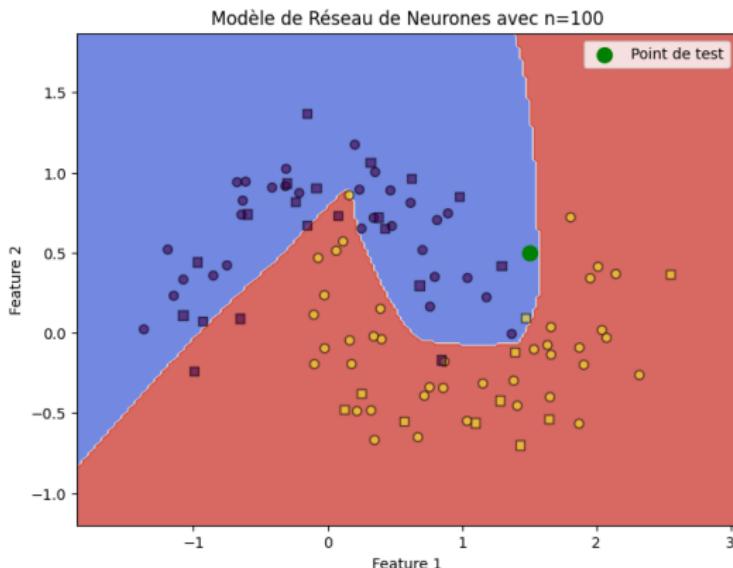
On dit qu'un modèle de machine learning est en régime de sous-apprentissage (underfitting) lorsqu'il n'arrive pas à capturer la complexité (l'information) présente dans le jeu de données d'entraînement.



Sur-apprentissage

Definition

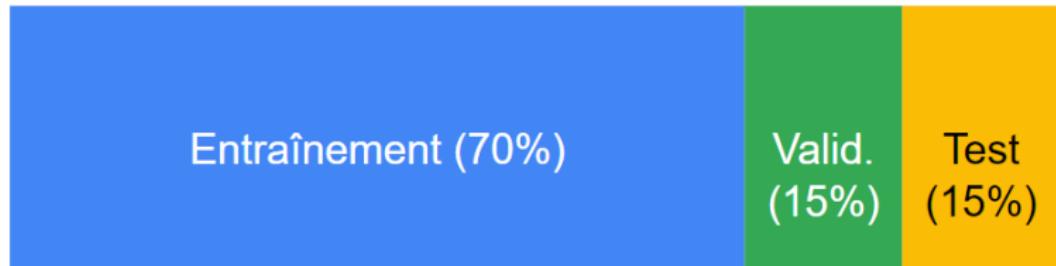
On dit qu'un modèle de machine learning est en régime de sur-apprentissage (overfitting) lorsqu'il n'arrive pas à généraliser à des données non encore observées, i.e. lorsqu'il est trop adapté aux données d'entraînement.



Sélection de modèle

Pour sélectionner le modèle le plus pertinent par rapport à une métrique donnée, on applique la méthodologie suivante :

- On partitionne le jeu de données disponible en trois parties : un jeu d'entraînement, un jeu de validation et un jeu de test.
- On entraîne M modèles sur le jeu d'entraînement.
- On évalue les performances respectives des M modèles sur le jeu de validation et on sélectionne le meilleur.
- Le modèle sélectionné est ensuite évalué sur le jeu de test. Idéalement, le jeu de test est ainsi utilisé une seule fois.



Exemple : sélection de modèle

Modèle	Précision Entraînement	Précision Validation
Régression Logistique	0.90	0.88
Arbre de Décision	0.95	0.92
Réseau de Neurones Profond	1	0.90

Table: Comparaison des précisions des modèles sur les jeux d'entraînement et de validation

Métriques de performance : régression

On dispose d'un certain nombre de métriques pour évaluer les performances des modèles de machine learning. Celles-ci peuvent être divisées en deux catégories.

Régression

- L'erreur quadratique moyenne (MSE) : elle est définie comme la moyenne des carrés des écarts entre les prédictions et les valeurs observées.

$$MSE = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

- La racine carrée de l'erreur quadratique moyenne (RMSE) :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2}$$

Métriques de performance : classification 1/2

Accuracy : L'accuracy est la métrique de base qui permet d'évaluer les performances d'un modèle de classification. Elle est définie comme :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Matrice de confusion : La matrice de confusion est une représentation permettant d'offrir plus de finesse par rapport à l'accuracy, notamment quand le jeu de données est déséquilibré (présence de classes majoritaires). Elle compare les prédictions du modèle avec les valeurs réelles et est structurée comme suit :

		Valeur Prédite	
		Positif	Négatif
Valeur Réelle	Positif	Vrai Positif (VP)	Faux Négatif (FN)
	Négatif	Faux Positif (FP)	Vrai Négatif (VN)

Métriques de performance : classification 2/2

A partir de la matrice de confusion, on peut dériver d'autres métriques :

- Précision : elle est définie comme la proportion des prédictions correctes parmi toutes les prédictions positives :

$$\text{Précision} = \frac{VP}{VP + FP}$$

- Rappel (recall) : il représente la proportion des vrais positifs correctement prédits par le modèle.

$$\text{Rappel} = \frac{VP}{VP + FN}$$

- Score F1 (F1-score) : Le score F1 est défini comme la moyenne harmonique de la précision et du rappel.

$$\text{Score F1} = 2 \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

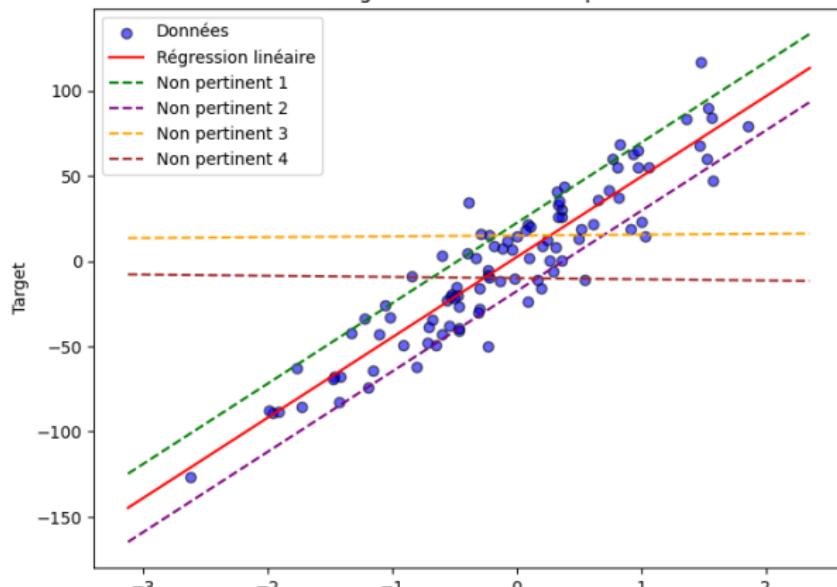
Modèles classiques de machine learning

Régression linéaire simple

Soit un ensemble de n observations x_1, x_2, \dots, x_n avec les labels correspondants y_1, y_2, \dots, y_n , on cherche le modèle linéaire qui ajuste le mieux ces données.

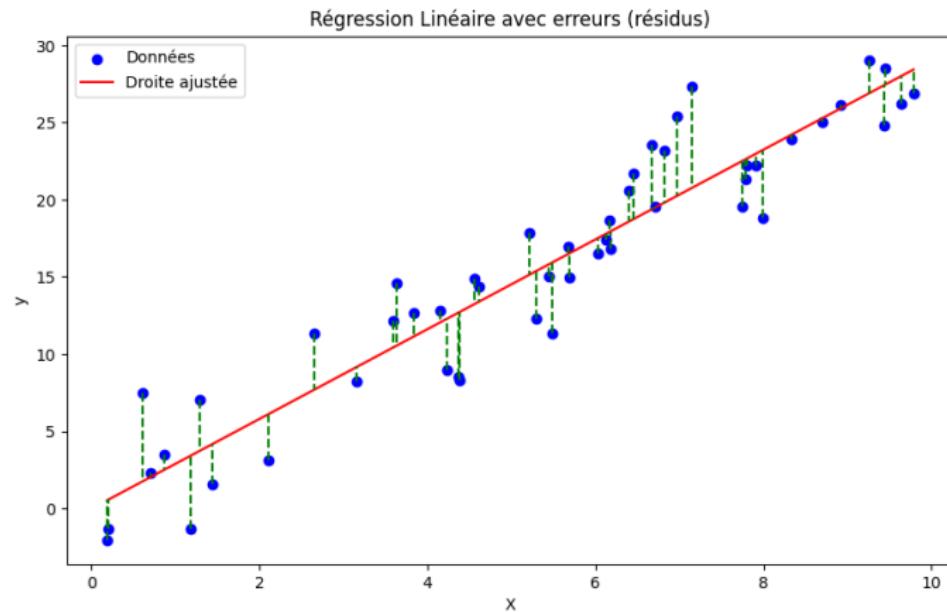
$$\hat{y} = \beta_0 + \beta_1 x$$

Illustration de la régression linéaire avec plusieurs modèles



Minimisation du risque empirique 1/2

L'erreur de prédiction pour la i ième observation est : $e_i = y_i - \hat{y}_i$. où $\hat{y}_i = \beta_0 + \beta_1 x_i$.



Minimisation du risque empirique 2/2

Déterminer un modèle de régression linéaire simple revient à minimiser les erreurs de prédiction (risque empirique) :

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Autrement dit, chercher les coefficients β_j qui minimisent le risque empirique :

$$\arg \min_{\beta_0, \beta_1} \left(\sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2 \right)$$

Les expressions des β sont obtenues en résolvant les équations normales :

$$\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

où \bar{x} et \bar{y} sont les moyennes des x_i et y_i , respectivement.

Exemple : prédire le prix d'un logement en fonction de la surface 1/2

Soit un dataset de 100 points où x représente la surface (en m²) et y le prix des logements (en €). Les 4 premières lignes sont :

x (Surface m ²)	y (Prix en €)
40	120 000
65	200 000
80	250 000
55	160 000
:	:

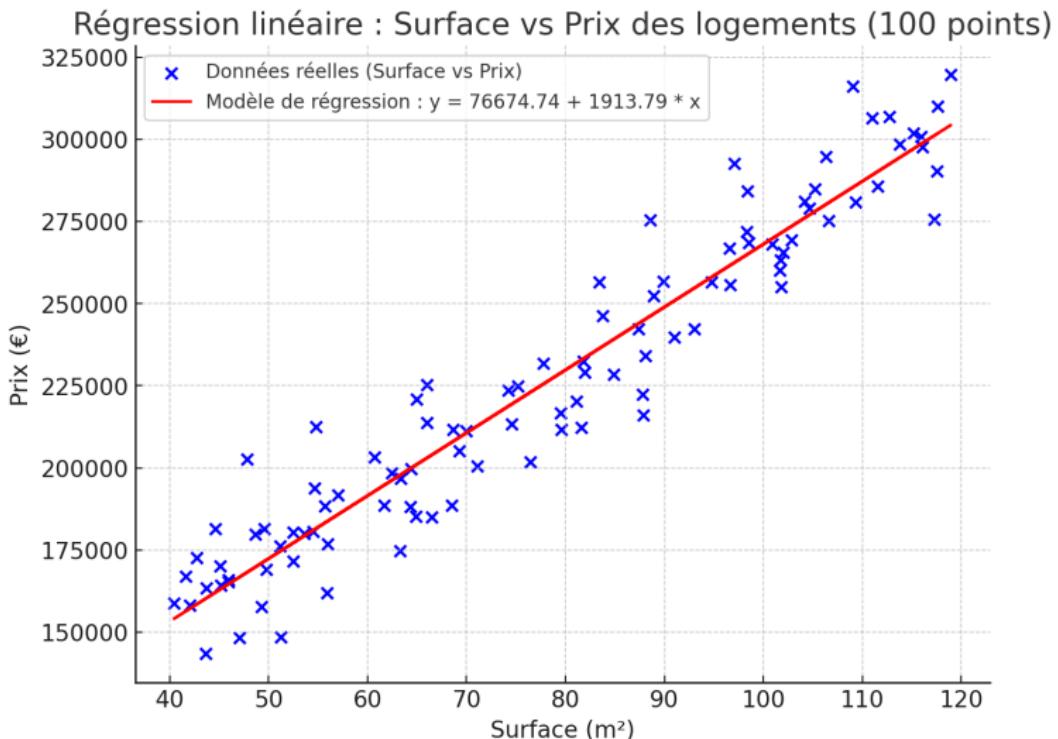
Les coefficients β_0 et β_1 sont calculés à partir des formules suivantes :

$$\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{1234567}{56789} = 2173.15$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} = 200000 - 2173.15 \times 60 = 69410.5$$

L'équation de la droite de régression obtenue est donc :

Exemple : prédire le prix d'un logement en fonction de la surface 2/2



Régression linéaire multiple

On considère n observations X^1, X^2, \dots, X^n où chaque observation X^i est désormais un vecteur ayant p composantes (p variables explicatives).

$$X^i = \begin{pmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_p^i \end{pmatrix}$$

La régression linéaire s'écrit alors :

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Les coefficients $\beta_0, \beta_1, \dots, \beta_p$ sont déterminés par la méthode des moindres carrés :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \sum_{i=1}^n \left(y^i - (\beta_0 + \sum_{j=1}^p \beta_j x_j^i) \right)^2$$

Choix de modèle

Il y a plusieurs manières d'évaluer la pertinence d'un modèle de régression linéaire. Le coefficient coefficient de détermination R^2 mesure l'ajustement du modèle. Il est donné par :

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Pour une régression linéaire multiple, on préférera cependant le coefficient de détermination ajusté R_a^2 .

$$R_a^2 = 1 - \frac{n - 1}{n - k - 1} * (1 - R^2)$$

où n est le nombre d'abosrvations et k le nombre de variables.

Des critères comme le AIC et le BIC sont également utilisés pour sélectionner un modèle.

Régression polynomiale

La régression linéaire peut prendre en compte les dépendances non linéaires entre les variables explicatives x_1, x_2, \dots, x_p et la variable expliquée y . Lorsque cette dépendance prend la forme d'un polynôme de degré d , la régression linéaire s'écrit alors :

$$\hat{y} = \beta_{00} + \sum_{j=1}^p \beta_{ij} x_j + \sum_{j=1}^p \beta_{ij} x_j^2 + \dots + \sum_{j=1}^p \beta_{ij} x_j^d$$

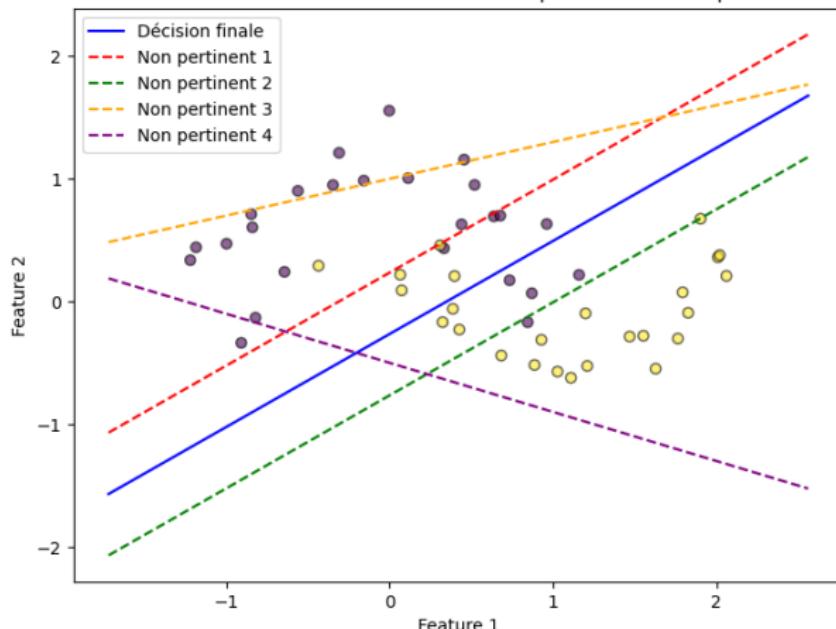
Les coefficients β_{ij} peuvent être de la même manière, avec la méthode des moindres carrés.

Remarque : On peut utiliser n'importe quelle fonction non linéaire pour transformer les variables explicatives (\cos , \ln , etc.). Le modèle reste tout de même linéaire (linéarité par rapport aux coefficients).

Principe de la régression logistique

- **Définition :** La régression logistique est un algorithme de classification linéaire utilisé pour prédire la probabilité qu'une observation appartienne à une classe donnée.

Illustration de la surface de décision avec plusieurs droites possibles



Régression logistique : introduction

La régression logistique est une technique d'analyse statistique utilisée pour modéliser la probabilité d'une variable dépendante binaire. C'est un cas particulier de modèle linéaire généralisé qui est utilisé pour des problèmes de classification.

Principes de la régression logistique :

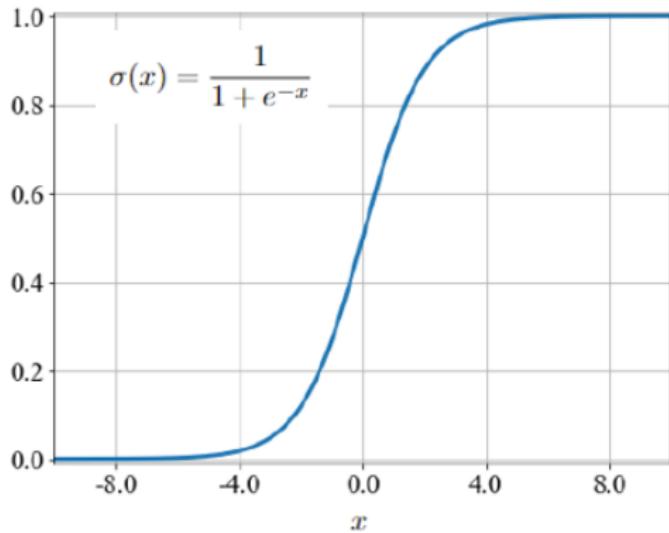
- **Variable dépendante** : On cherche la probabilité que la variable dépendante (y) appartienne à une classe (0 ou 1, vrai ou faux, succès ou échec). Autrement dit, on cherche à modéliser $P(y = 1)$ en fonction des variables dépendantes (explicatives) x .
- **Odds ratio** : Plus concrètement, on cherche à exprimer la côte anglaise (odd ratio) en fonction des variables dépendantes (x).

$$\ln \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Régression logistique : fonction sigmoïde

Après quelques simplifications, on peut écrire la probabilité $p(x)$ (la probabilité pour que y soit un succès par exemple) :

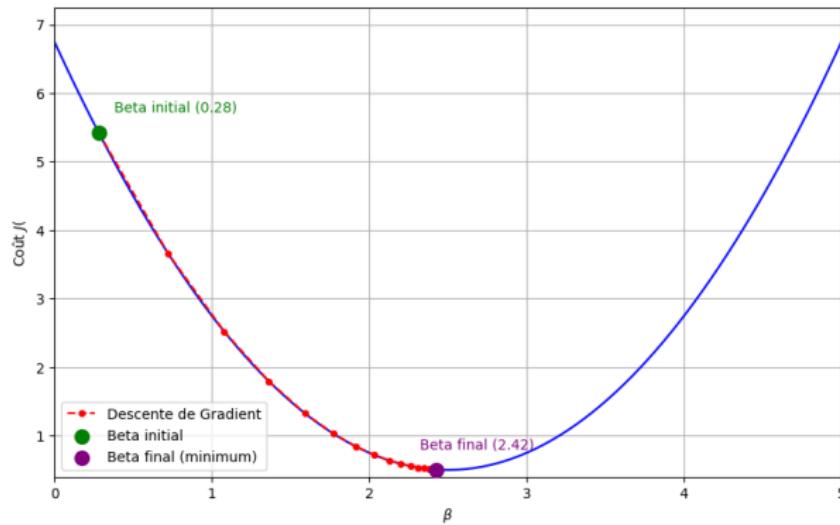
$$p(\mathbf{x}) = \frac{1}{1 + e^{-(\beta^T \mathbf{x})}}$$



Calcul des coefficients de la régression logistique

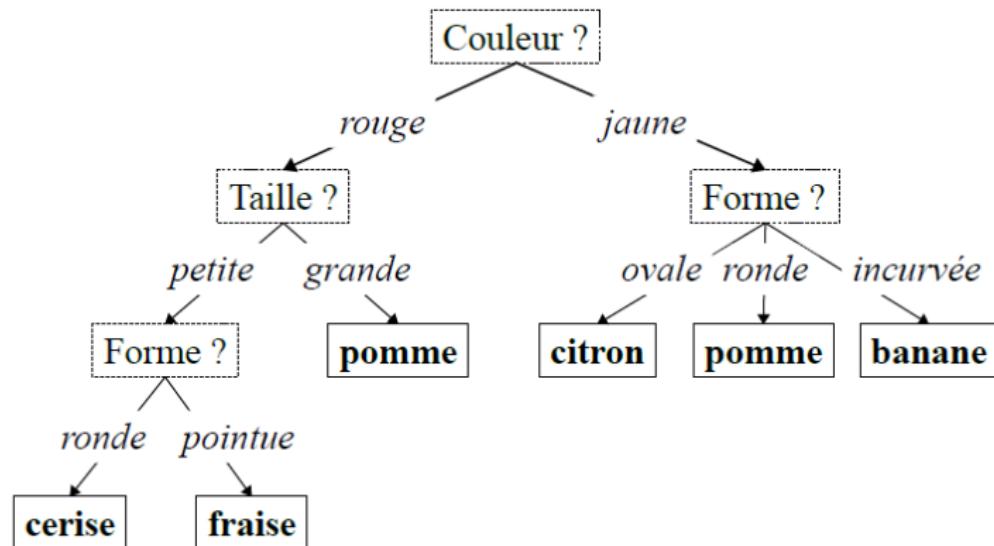
Les coefficients de la régression logistique peuvent être calculés en minimisant le risque empirique par rapport à une fonction de coût sous forme d'entropie croisée :

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left(- \sum_{i=1}^n y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i) \right)$$



Arbres de décision

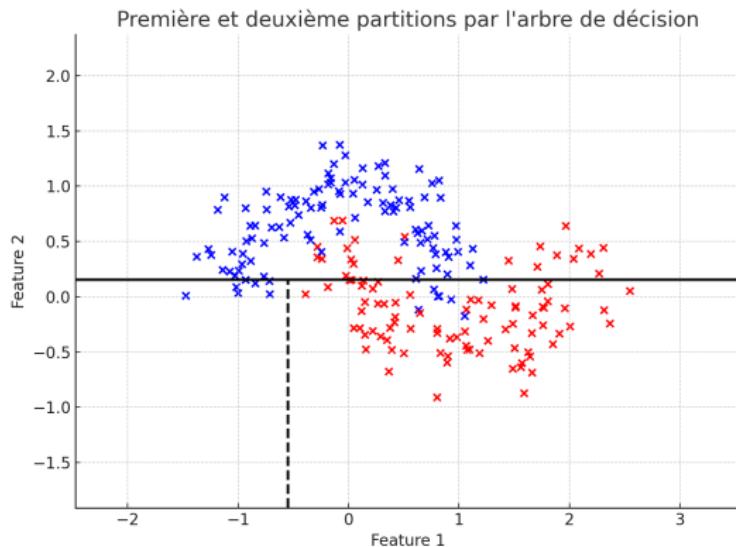
Les arbres de décisions sont des modèles dont le processus de décision est hiérarchique et prend la forme d'un arbre.



Entraînement des arbres de décision

Les arbres de décisions sont généralement entraînés à l'aide de la technique CART (Classification And Regression Trees).

Etant donné un ensemble d'observations $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, les arbres de décision partitionnent cet espace en plusieurs régions R_1, R_2, \dots, R_m .



Critères d'optimisation

Le critère d'optimisation dépend de la tâche en question.

- **Classification**

- Indice de Gini simplifié : $\sum_{k=1}^K p_{mk}(1 - p_{mk})$
- Entropie croisée : $-\sum_{k=1}^K p_{mk} \ln(p_{mk})$

- **Régression**

$$\sum_{i=n}^n (y_i - f(x_i))^2$$

Forêts aléatoires (random forests)

La technique des forêts aléatoires (random forests) consiste à appliquer une approche de type bagging sur les arbres de décision.

L'algorithme random forests suit cette procédure :

- Tirer par bootstrap B échantillons de tailles n à partir de l'ensemble D .
- Pour chaque échantillon tiré, construire un arbre en répétant les étapes suivantes jusqu'à atteindre n_{min} .
 - Tirer d'une manière aléatoire m variables parmi les p variables.
 - Sélectionner la meilleure variable avec le meilleur point de partitionnement.
 - Partitionner le noeud en deux sous-branches.
- Agréger les arbres construits.

Les prédictions sont agrégées selon qu'il s'agisse de régression ou de classification :

- Régression : moyenne $f^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$
- Classification : vote majoritaire.

Remarques

- L'algorithme de random forests intègre nativement une forme de validation croisée. Les performances mesurées sur $\bigcup_{b_i \neq b_k}$ (out of bag ou OOB) sont souvent proches de celles que l'on pourrait mesurer avec une validation croisée.
- Le nombre de variable tirées pour chaque noeud est généralement donné par \sqrt{p} pour la classification et $\frac{p}{3}$ pour la régression. Cet hyperparamètre dépend cependant du problème considéré.
- Lorsque le nombre de variable est élevé alors que le nombre de variables réellement partinente est faible, la probabilité que les p variables sélectionnées pour chaque partitionnement incluent des variables pertinentes devient faible, et les performances du modèle en termes de généralisation peuvent se détériorer considérablement.
- L'algorithme random forests permet de restituer des informations sur l'importance des variable (feature importance).

Apprentissage non supervisé

Apprentissage non supervisé

Dans l'apprentissage non supervisé, on considère n observations sans labels. On s'intéresse fondamentalement à la probabilité jointe de ces observations.

On peut distinguer deux grandes catégories d'apprentissage non supervisé :

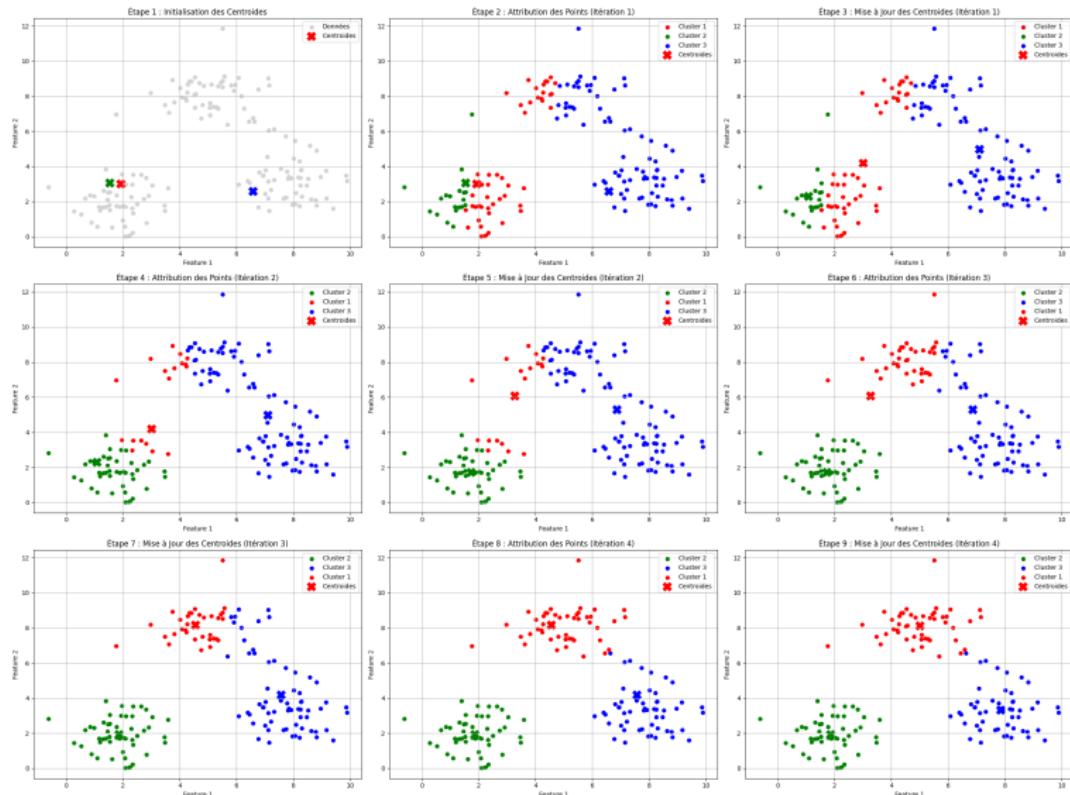
- **Clustering (partitionnement)** : cela consiste à partitionner les n observations en K groupes pertinents (généralement le critère de pertinence a une signification d'un point de vue métier).
- **Détection d'anomalie** : il s'agit de détecter des observations qui présentent un profil (des features) inhabituels par rapport au profil moyen de la majorité des observations.

K-means

L'algorithme de Lloyd tente de regrouper les données en clusters en minimisant les distances entre les points d'un même cluster tout en maximisant les distances entre points appartenant à différents clusters.



Algorithme de Lloyd

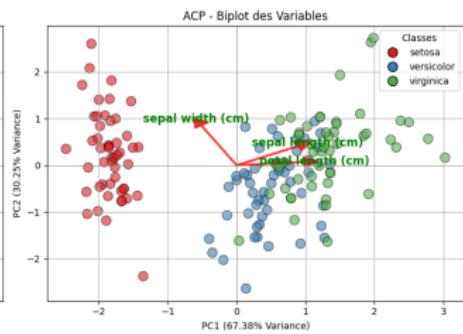
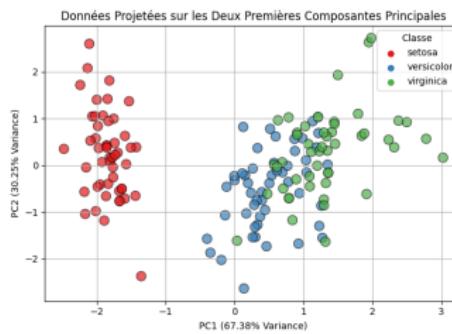
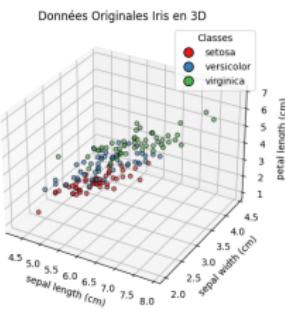


Remarques

- L'algorithme des k-means étant basé sur une distance euclidienne, il est nécessaire de normaliser les données avant de l'exécuter.
- L'algorithme des k-means est très sensible aux données aberrantes (outliers). Il faut donc considérer les données d'une manière attentive. Cependant, cela permet également d'utiliser l'algorithme des k-means pour la détection automatique des outliers.
- Les centroïdes étant initialisés d'une manière aléatoire, les clusters obtenus ne sont pas stables ; les clusters peuvent changer d'une exécution à l'autre. Il existe cependant une variante plus stable, appelée k-means++, qui permet de sélectionner les centroïdes d'une manière semi-aléatoire.
- Il est possible de partitionner les données avec une métrique plus générale que la distance euclidienne. On peut définir un algorithme k-means à noyau sur un espace de Hilbert pour aller au-delà de la métrique euclidienne.
- **K-means n'est pas adapté aux données en grande dimension.**

Analyse en Composantes Principales (ACP)

L'Analyse en Composantes Principales (ACP) est une technique statistique de réduction de dimensionnalité. Elle transforme les données en un nouveau système de coordonnées où la plus grande variance est capturée sur les premiers axes, appelés composantes principales.



Formulation de l'ACP

Soit X une matrice de données de dimension $n \times p$ (n observations, p variables), centrée (moyenne nulle). L'ACP cherche à trouver les vecteurs propres et les valeurs propres de la matrice de covariance $C = \frac{1}{n-1} X^T X$.

La matrice de covariance C peut être décomposée comme suit :

$$C = VLV^T$$

où $V = [u_1, u_2, \dots, u_p]$ est la matrice des vecteurs propres et L est une matrice diagonale des valeurs propres λ_k .

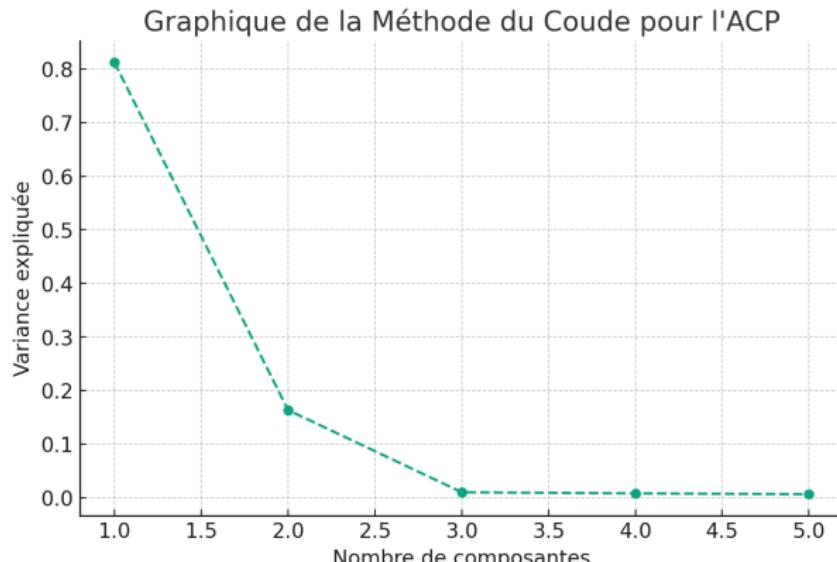
La contribution de chaque composante principale à la variance totale est donnée par :

$$\frac{\lambda_k}{\sum_{i=1}^p \lambda_i}$$

Choix du nombre de composantes principales

Le nombre de composantes à retenir est déterminé en fonction du pourcentage de variance totale que l'on souhaite expliquer.

On utilise généralement la méthode du coude (Scree plot). Il s'agit d'un graphique montrant la proportion de la variance expliquée en fonction du nombre de composantes.



Isolation Forest : Principe

L'Isolation Forest est une technique de détection d'anomalies basée sur l'isolement des observations. Son efficacité repose sur l'hypothèse que les anomalies sont "faciles à isoler" par rapport aux observations normales.

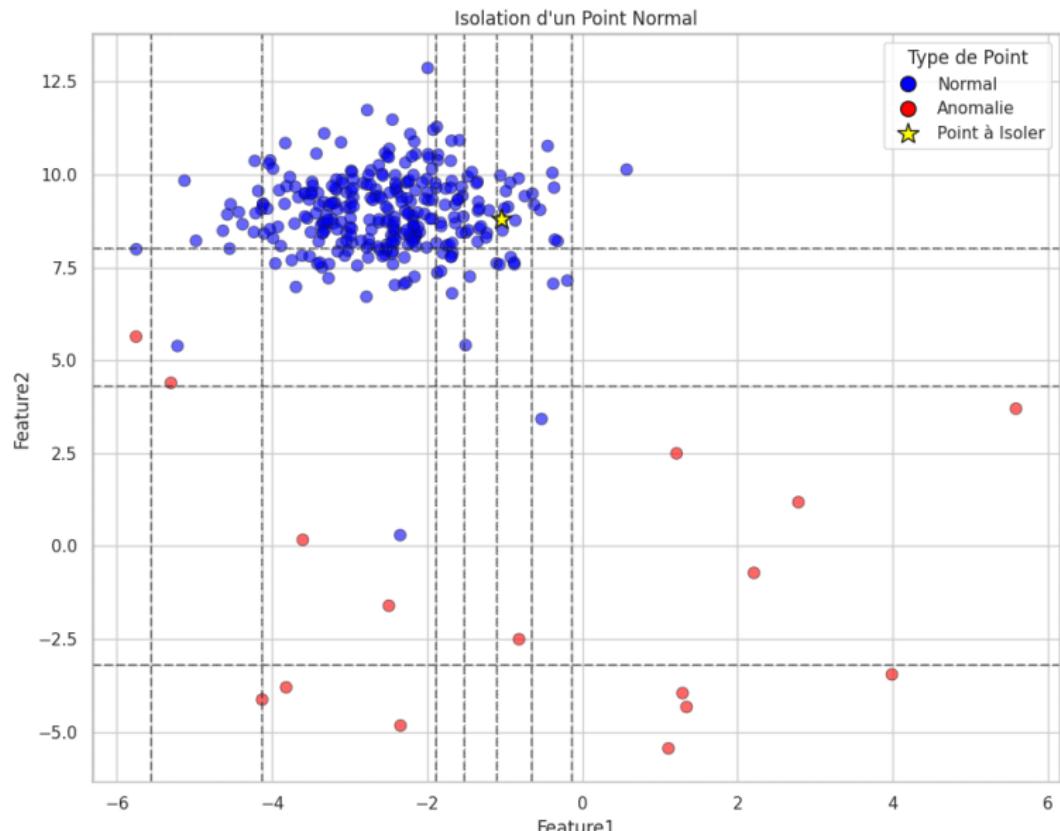
- Fonctionne en construisant des arbres d'isolement à partir de sous-ensembles de données.
- Isoler une observation signifie la séparer des autres par des divisions aléatoires de l'espace des caractéristiques.
- Moins de divisions sont nécessaires pour isoler une anomalie, ce qui constitue le fondement du score d'anomalie.

Les arbres d'isolement sont construits de manière récursive :

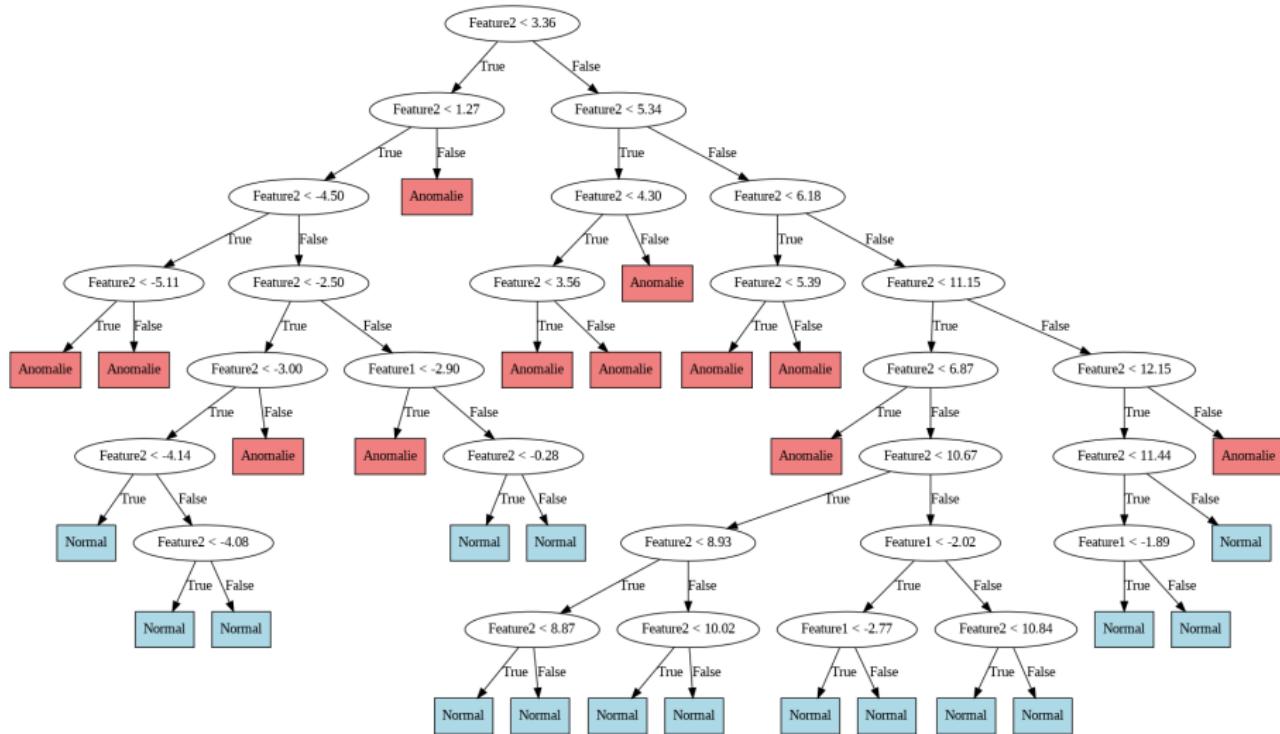
- ➊ Sélection aléatoire d'un sous-ensemble de données.
- ➋ Choix aléatoire d'une caractéristique et d'une valeur de seuil pour diviser le sous-ensemble.
- ➌ Répétition des divisions jusqu'à l'isolement des observations ou atteinte d'une limite de profondeur prédéfinie.

Chaque arbre est ainsi unique, offrant une perspective différente sur les données.

Isolation tree : fonctionnement 1/2



Isolation tree : fonctionnement 2/2



Deep learning

Introduction aux réseaux de neurones

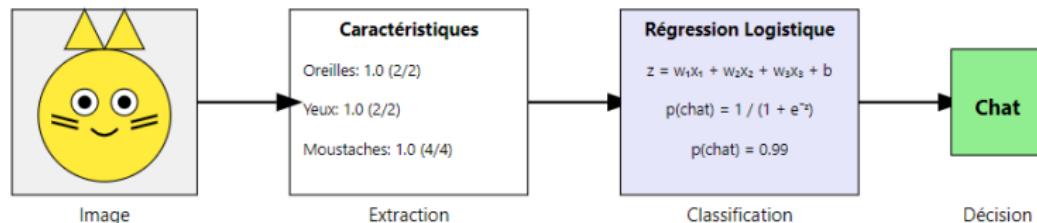
Définition : Les réseaux de neurones sont des modèles computationnels inspirés par le fonctionnement des neurones dans le cerveau humain. Ils sont capables d'apprendre des tâches complexes en modélisant des relations non linéaires entre les entrées et les sorties.

Caractéristiques :

- **Extraction automatique des features** : Capacité d'adaptation et d'extraction des features à partir des données sans programmation explicite.
- **Modélisation non linéaire** : Aptitude à capturer des relations complexes dans les données.
- **Modélisation en grande dimension** : Les modèles de deep learning sont particulièrement adaptés pour les données en grande dimension (images, texte, etc.).
- **Flexibilité** : Applicables à un large éventail de tâches et de typologies de données (images, langage naturel, données graphiques, etc.).

Extraction de features

Extraction de features en machine learning "classique"



Extraction de features en deep learning

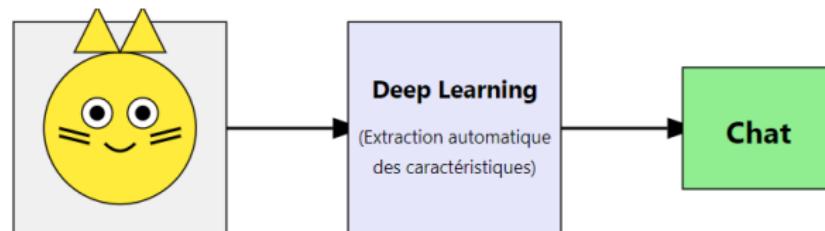
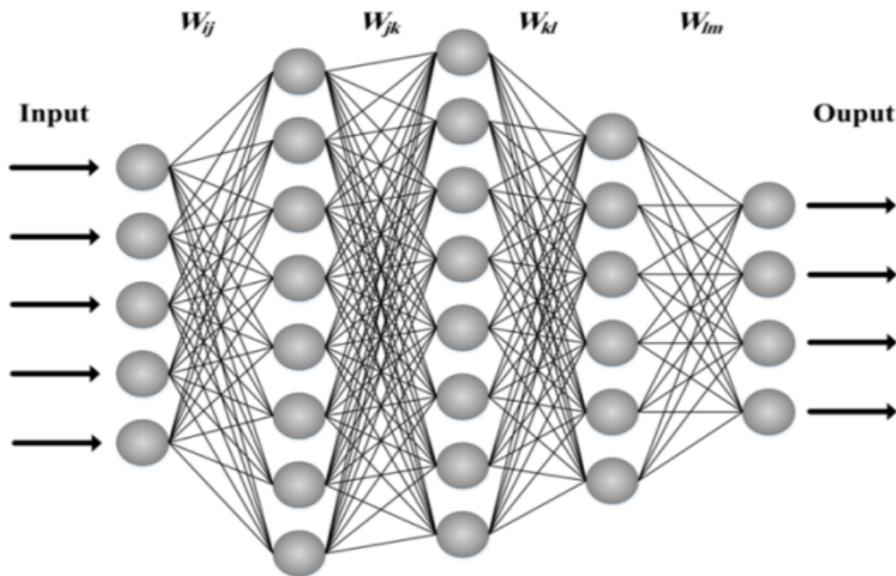
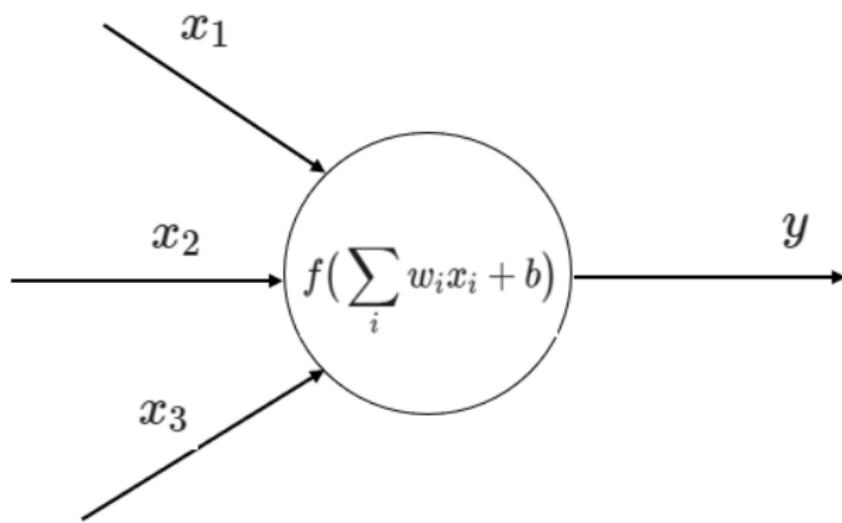


Illustration d'un réseau de neurones classique



Neurone aritificial



Fonctions d'activation

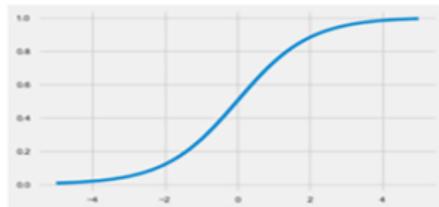
Les fonctions d'activation permettent aux modèles d'apprendre des relations plus complexes en capturant les non-linéarités dans les données.

- **Sigmoïde** : $\sigma(z) = \frac{1}{1+e^{-z}}$, plage de sortie (0, 1), utilisée pour la probabilité dans la classification binaire.
- **Tangente Hyperbolique (tanh)** : $\tanh(z)$, plage de sortie (-1, 1), version centrée et normalisée de la sigmoïde.
- **ReLU (Unité Linéaire Rectifiée)** : $f(z) = \max(0, z)$, non saturante, favorise la convergence rapide et permet d'éviter le problème de disparition des gradients.
- **Leaky ReLU** : $f(z) = \max(\alpha z, z)$, variante de ReLU qui permet un petit gradient lorsque $z < 0$.
- **Softmax** : Utilisée pour la couche de sortie des problèmes de classification multi-classes.

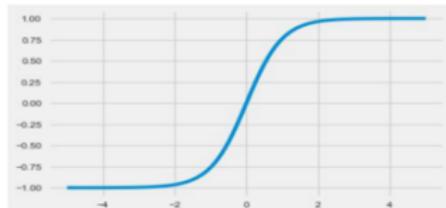
$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Illustration des fonctions d'activation

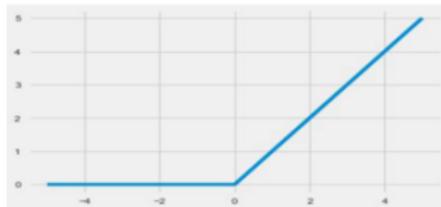
Sigmoïde



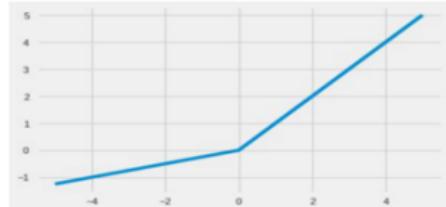
Tanh



ReLU



ReLU paramétrique



Entraînement d'un réseau de neurones artificiel

Principe d'Entraînement : L'entraînement d'un réseau de neurones consiste à ajuster ses poids pour minimiser une fonction de coût qui mesure l'erreur entre les prédictions et les vraies valeurs.

Descente de gradient stochastique (SGD) :

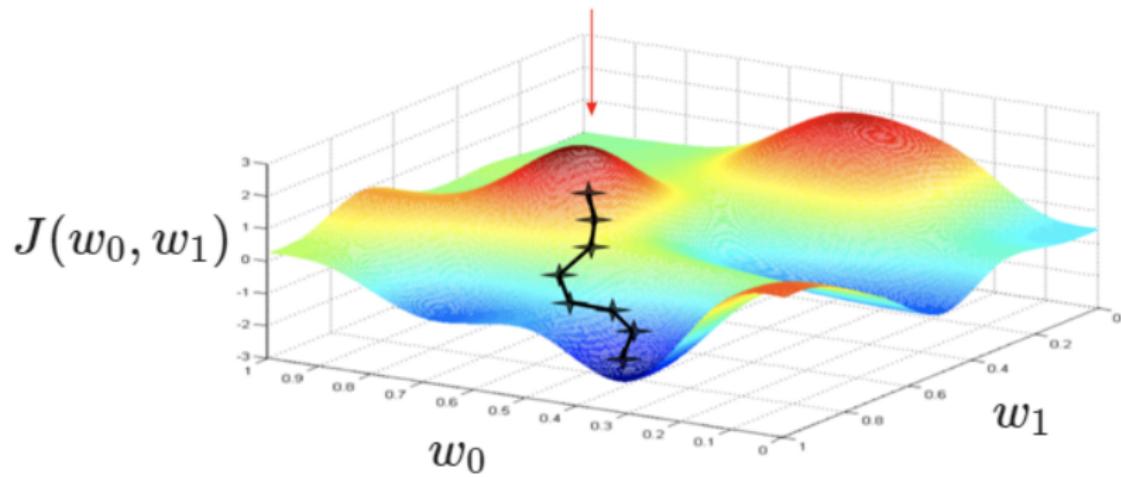
- Méthode d'optimisation utilisée pour mettre à jour les poids du réseau de manière itérative.
- À chaque itération, un sous-ensemble (batch) de données est utilisé pour calculer le gradient de la fonction de coût.
- Les poids sont mis à jour dans la direction opposée du gradient pour réduire l'erreur.

$$w_{new} = w_{old} - \eta \cdot \nabla_w J(w)$$

où :

- w_{old} et w_{new} sont les valeurs des poids avant et après la mise à jour,
- η est le taux d'apprentissage,
- $\nabla_w J(w)$ est le gradient de la fonction de coût par rapport aux poids.

Illustration graphique de la SGD



Rétropropagation du gradient

Il est facile de calculer les gradients correspondant à la dernière couche $\frac{\partial J}{\partial W^{(n)}}$ car l'erreur J dépend immédiatement de $W^{(n)}$.



Examinons maintenant le cas de la couche $n - 1$:

$$\frac{\partial J}{\partial W^{(n-1)}} = \frac{\partial J}{\partial W^{(n)}} \frac{\partial W^{(n)}}{\partial O_n} \frac{\partial O_n}{\partial W^{(n-1)}}$$

Or

$$\frac{\partial O_n}{\partial W^{(n-1)}} = \frac{\partial O_n}{\partial O_{n-1}} \frac{\partial O_{n-1}}{\partial W^{(n-1)}}$$

Surapprentissage dans les réseaux de neurones

Le surapprentissage (overfitting) se produit lorsqu'un réseau de neurones apprend trop bien les détails et le bruit des données d'entraînement, au détriment de sa capacité à généraliser sur de nouvelles données.

Le surapprentissage dans les réseaux de neurones peut se produire pour différentes raisons :

- Complexité excessive du modèle
- Manque de diversité dans les données d'entraînement
- Entraînement prolongé

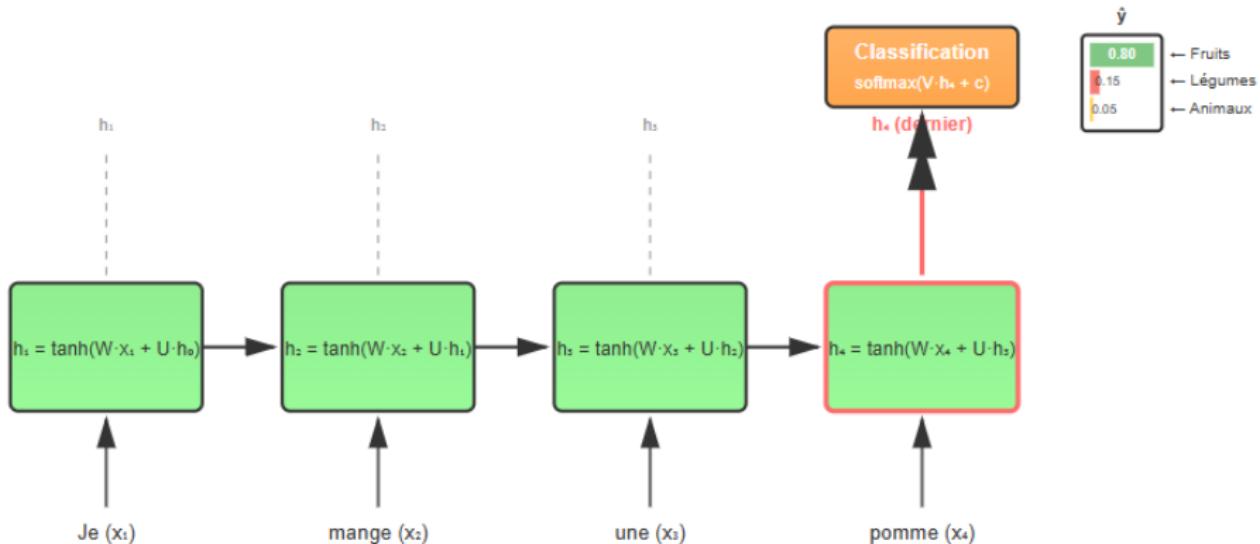
Stratégies pour atténuer le surapprentissage :

- **Régularisation** : Techniques de régularisation telles que L1/L2, Dropout, pour limiter la complexité du modèle.
- **Dropout** : "Eteindre" un ensemble de neurones choisis aléatoirement pendant l'entraînement.
- **Early Stopping** : Arrêt de l'entraînement lorsque la performance sur un ensemble de validation cesse de s'améliorer.
- **Augmentation de données** : Augmenter la variété des données d'entraînement pour améliorer la robustesse du modèle.

Traitement automatique du langage (NLP)

Anciens modèles pour le traitement du langage : RNN

- **Mémoire à court terme** : Les réseaux de neurones récurrents (RNN) ont la capacité de se "souvenir" d'informations passées grâce à leurs connexions récurrentes.
 - **Traitement de séquences** : Ils sont particulièrement adaptés pour des tâches où les données sont séquentielles et où le contexte est important.



Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

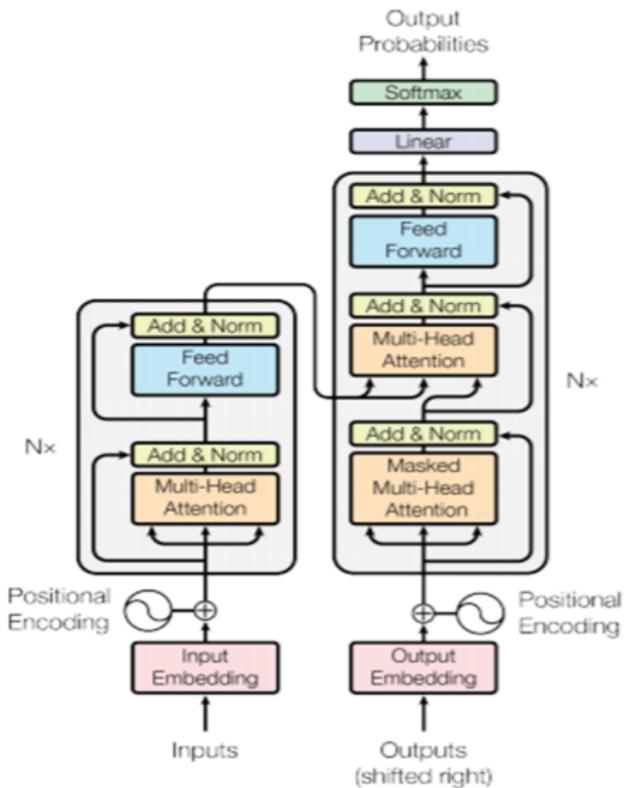
Lukasz Kaiser*
Google Brain
lukasz.kaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

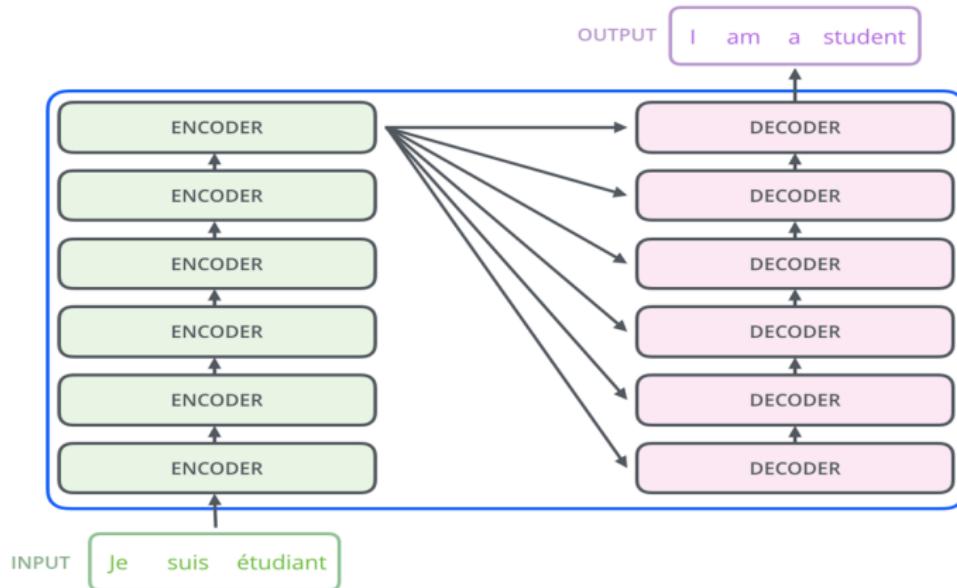
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to

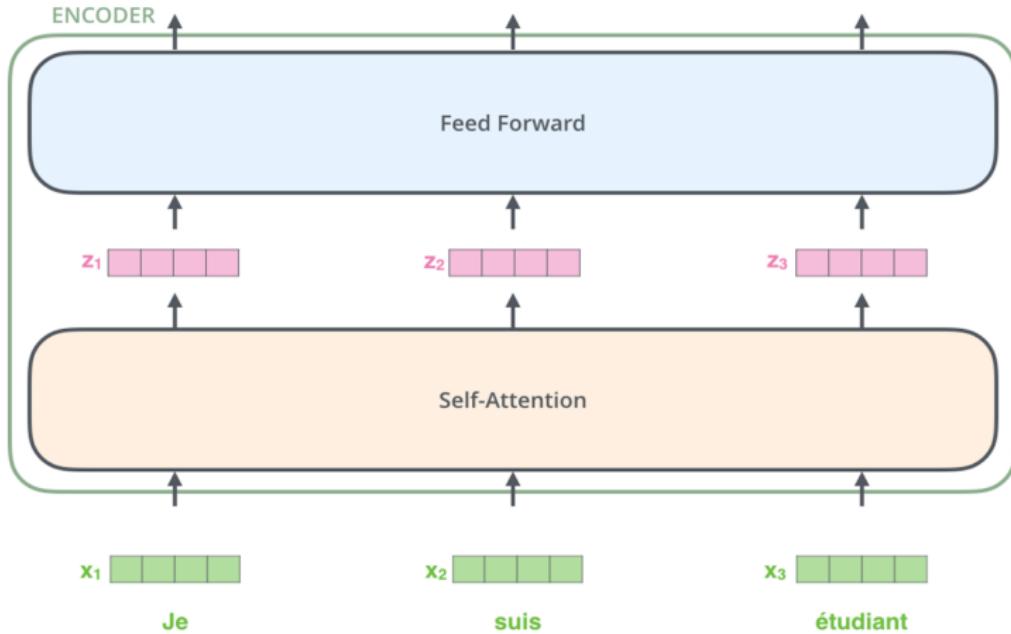
Transformer originel



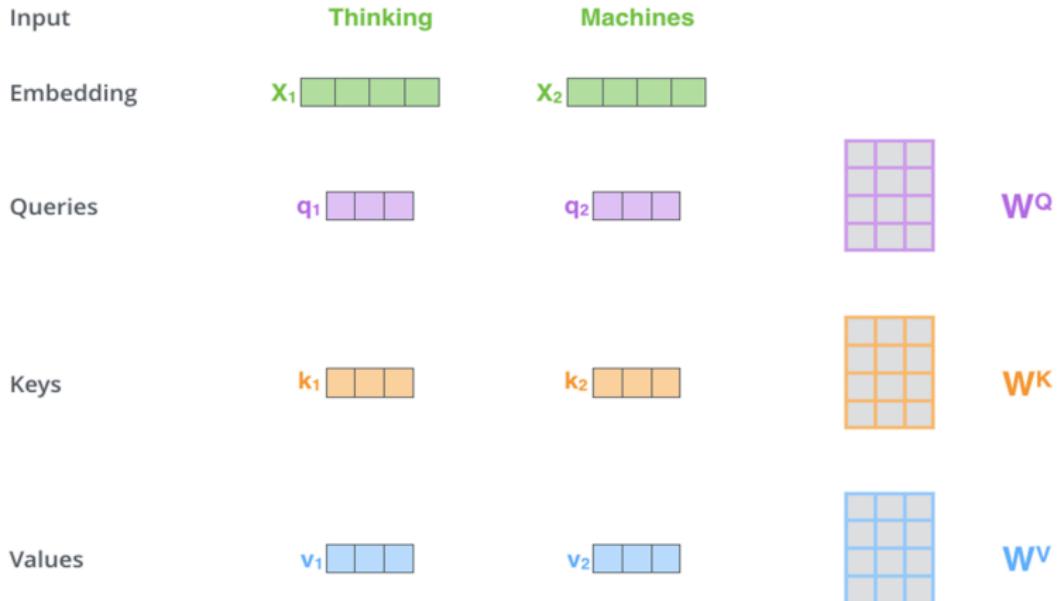
Mécanisme de cross-attention



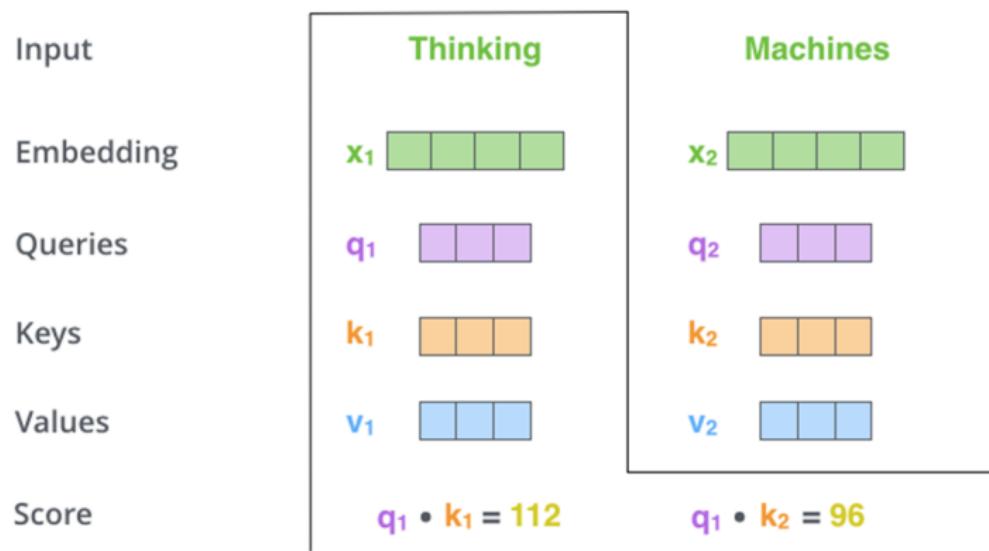
Mécanisme de self-attention



Fonctionnement du mécanisme de self-attention



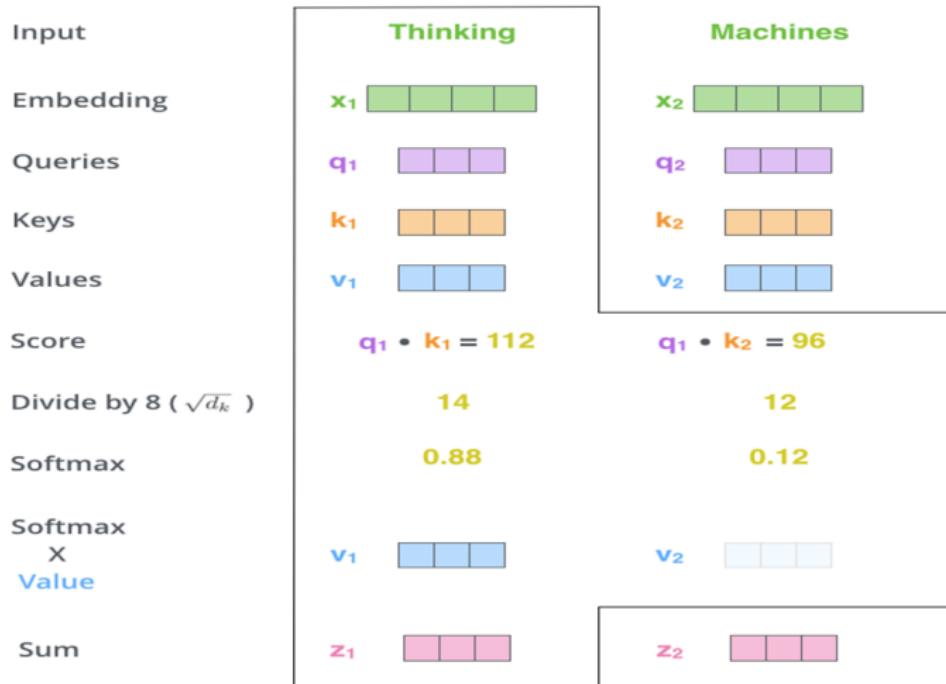
Calcul des scores de self-attention



Calcul des scores de self-attention

Input	Thinking		Machines	
Embedding	x_1	[4 green boxes]	x_2	[4 green boxes]
Queries	q_1	[3 purple boxes]	q_2	[3 purple boxes]
Keys	k_1	[3 orange boxes]	k_2	[3 orange boxes]
Values	v_1	[3 blue boxes]	v_2	[3 blue boxes]
Score	$q_1 \bullet k_1 = 112$		$q_1 \bullet k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14		12	
Softmax	0.88		0.12	

Calcul de la nouvelle représentation



Performances du tranformer originel

Entraîné sur WMT 2014 English-German dataset, comprenant près de 4.5 millions de phrases sentence, le WMT 2014 English-French dataset, comprenant près de 36 millions de phrases.

- 8 NVIDIA P10 GPUs
- **Base** : 12 heures
- **Large** : 3.5 jours

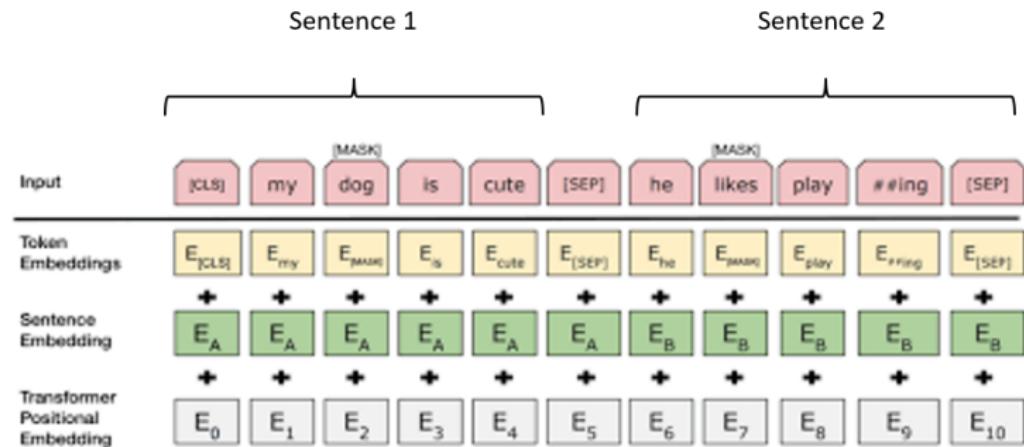
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

BERT

BERT (Bidirectional Encoder Representations from Transformers) représente une avancée majeure dans le NLP, introduisant une approche novatrice pour la modélisation de langage. Il utilise la bidirectionnalité pour comprendre le contexte des mots, permettant une compréhension plus fine du langage.

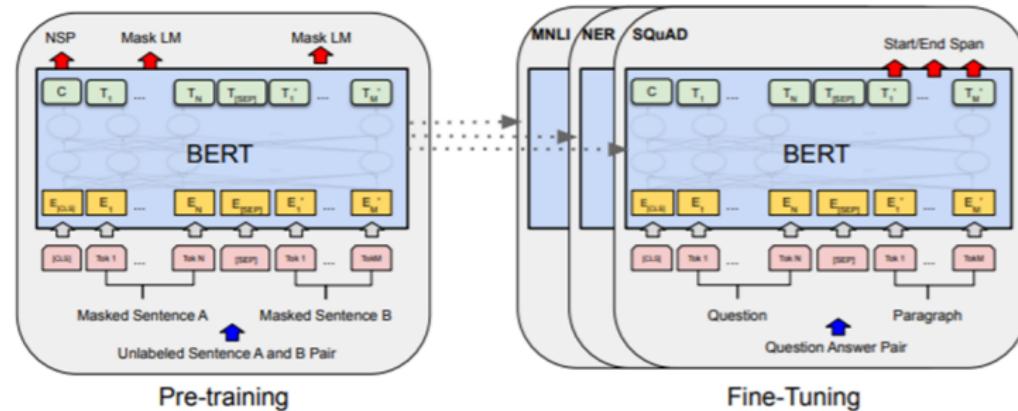
Pré-entraînement



Fine-tuning de BERT pour des tâches spécifiques

Après pré-entraînement, BERT est affiné pour des tâches spécifiques:

- Ajout d'une couche de sortie spécifique à la tâche (classification, NER, QA).
 - Fine-tuning de tous les paramètres du modèle pré-entraîné sur le corpus de la tâche.



Performances de BERT

BERT a été pré-entraîné sur le BookCorpus (800 millions de mots) et English Wikipedia (2,500 millions de mots).

Deux modèles :

- **BERT Base** : 12 couches avec 110 millions de paramètres.
- **BERT Large** : 24 couches avec 340 millions de paramètres.

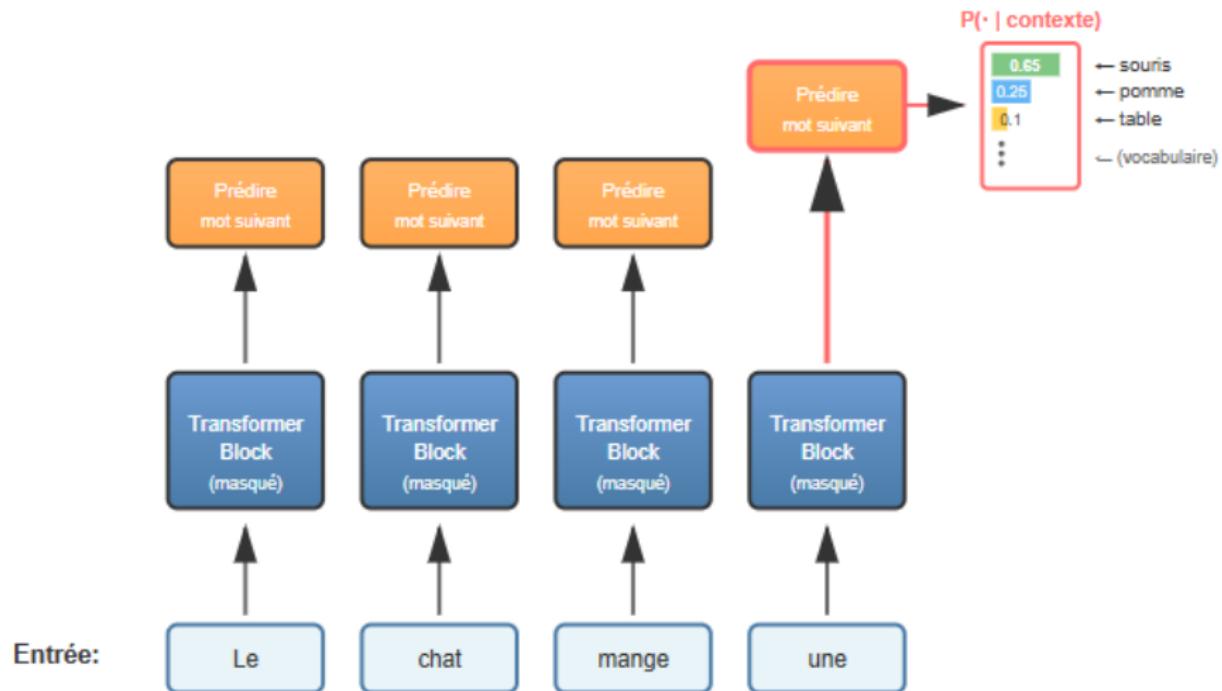
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Modèles génératifs (GPT)

*Ce que je ne peux pas créer,
je ne le comprends pas.*

– Richard Feynman

Fonctionnement des modèles génératifs



Le paramètre de température

Définition La température (T) contrôle le degré d'aléa dans les réponses générées. Elle agit sur la distribution de probabilité des mots avant l'échantillonnage.

Formule mathématique :

$$P(x_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

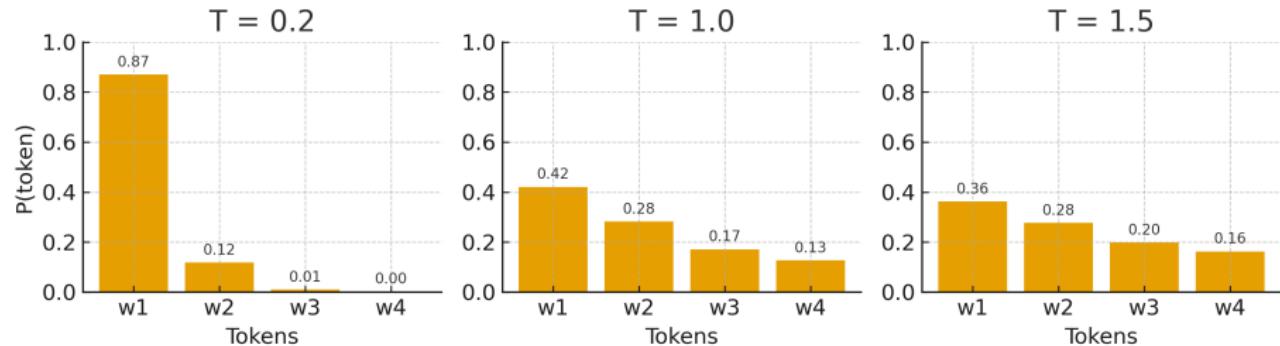
- $T < 1 \rightarrow$ distribution plus concentrée, réponses plus prévisibles
- $T > 1 \rightarrow$ distribution plus plate, réponses plus variées et créatives

Autrement dit :

- Basse température \rightarrow précision et cohérence.
- Haute température \rightarrow imagination et diversité.

Illustration graphique de l'effet de la température

Effet de la température sur la distribution des probabilités



La tokénisation

Problème de départ : Les modèles de langage ne peuvent pas traiter directement du texte brut : les mots, les accents, la ponctuation et les variations sont trop nombreux (plusieurs millions de combinaisons possibles).

La tokénisation : Elle consiste à découper le texte en **unités réutilisables et limitées en nombre**, appelées *tokens*. Chaque token est ensuite représenté par un identifiant numérique que le modèle peut apprendre à prédire.

Intérêt de la tokénisation :

- Réduire la complexité du vocabulaire : on passe de millions de mots à quelques dizaines de milliers de tokens.
- Gérer les mots inconnus ou rares en les décomposant en sous-unités connues.
- Offrir une représentation numérique uniforme du langage, compatible avec les réseaux de neurones.

La composition des données d'entraînement des modèles

Principe général Les modèles de langage sont entraînés sur d'immenses corpus de texte — souvent plusieurs milliers de milliards de tokens — issus de sources très diverses pour couvrir la langue, les faits et les styles.

Sources typiques de données :

- **Web public** : pages web, articles, forums, Wikipédia (ex. Common Crawl).
- **Livres et documents** : œuvres littéraires, articles scientifiques, rapports techniques.
- **Données de code** : dépôts GitHub, notebooks, documentation technique.
- **Conversations et dialogues** : jeux de données de chat, discussions de forums, données annotées par des humains.
- **Données synthétiques** : textes générés par d'autres modèles pour enrichir certaines thématiques.

Équilibre recherché :

- Diversité linguistique et thématique (langues, registres, domaines).
- Qualité et cohérence (filtrage du spam, des doublons, des contenus toxiques).
- Représentation équilibrée entre langage courant, technique et conversationnel.

GPT-3

- Développé par OpenAI, publié en 2020.
- Modèle auto-régressif basé sur l'architecture Transformer.

Caractéristique	Valeur / Détail
Date	2020
Paramètres	175 milliards
Corpus d'entraînement	Environ 300 milliards de tokens
Contexte maximum	2048 tokens
Ressources GPU	Plusieurs centaines de GPU (type V100)
Coût d'entraînement (estim.)	4,6 M\$ à 12 M\$ (selon les sources)

- **Applications** : génération de texte, Q&R, résumé, traduction, assistance à la programmation.

Le few-shot learning

- Les grands modèles (comme GPT-3) peuvent réaliser des tâches **sans être explicitement réentraînés** dessus.
- Le principe repose sur **l'utilisation de quelques exemples** (exemples étiquetés) directement dans le prompt ou l'entrée.
- **Zero-shot** : aucune démonstration fournie, le modèle doit comprendre la tâche à partir de sa connaissance générale.
- **One-shot / Few-shot** : on inclut un nombre très réduit d'exemples (un ou quelques-uns) pour guider le modèle dans la résolution de la tâche.
- Exemple :

English: "cat" → French: "chat"

English: "house" → French: "maison"

English: "car" → French: "voiture"

English: "tree" → French:

ChatGPT

ChatGPT, développé par OpenAI, est un modèle de langage basé sur l'architecture GPT (Generative Pre-trained Transformer) optimisé pour comprendre et générer des dialogues naturels. L'entraînement de ChatGPT se décline selon les étapes suivantes :

- ① Pré-entraînement sur un corpus volumineux :** Comme GPT-3, ChatGPT est d'abord pré-entraîné sur un vaste ensemble de données textuelles, englobant un large éventail de la littérature disponible sur Internet, pour apprendre une compréhension générale du langage.
- ② Fine-tuning supervisé :** Ensuite, ChatGPT est affiné sur des dialogues spécifiques pour améliorer ses compétences conversationnelles. Cette étape utilise des paires question-réponse et des conversations pour enseigner au modèle des structures de dialogue et des réponses contextuellement appropriées.
- ③ Reinforcement Learning from Human Feedback (RLHF):** Utilisation de techniques de renforcement pour ajuster les réponses du modèle basées sur les préférences et les corrections fournies par des évaluateurs humains, raffinant davantage la pertinence et la naturalité des réponses.

LLMs propriétaires vs Open Source

- **Modèles propriétaires**

- ChatGPT, Claude, Gemini, Grok...
- Poids non disponibles au public.
- Performances élevées, accès limité par API payante
- Protection intellectuelle stricte (code non accessible)
- Support technique assuré, documentation avancée
- Dépendance aux fournisseurs et coûts potentiellement élevés

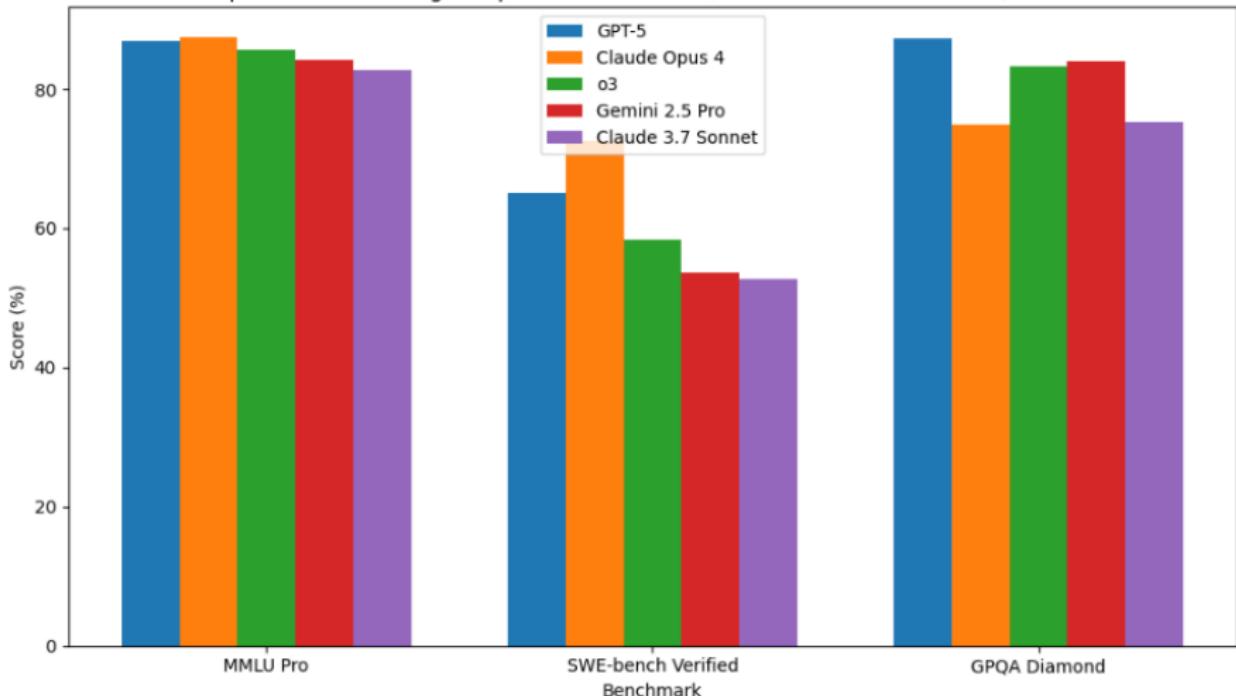
- **Modèles open source**

- Llama, Mistral, Gemma, DeepSeek, Qwen...
- Poids disponibles au public
- Transparence du code, flexibilité d'adaptation
- Performances parfois inférieures, mais amélioration continue
- Gratuit, mais coût de l'infrastructure à gérer
- Exposition aux failles potentielles de sécurité, support communautaire

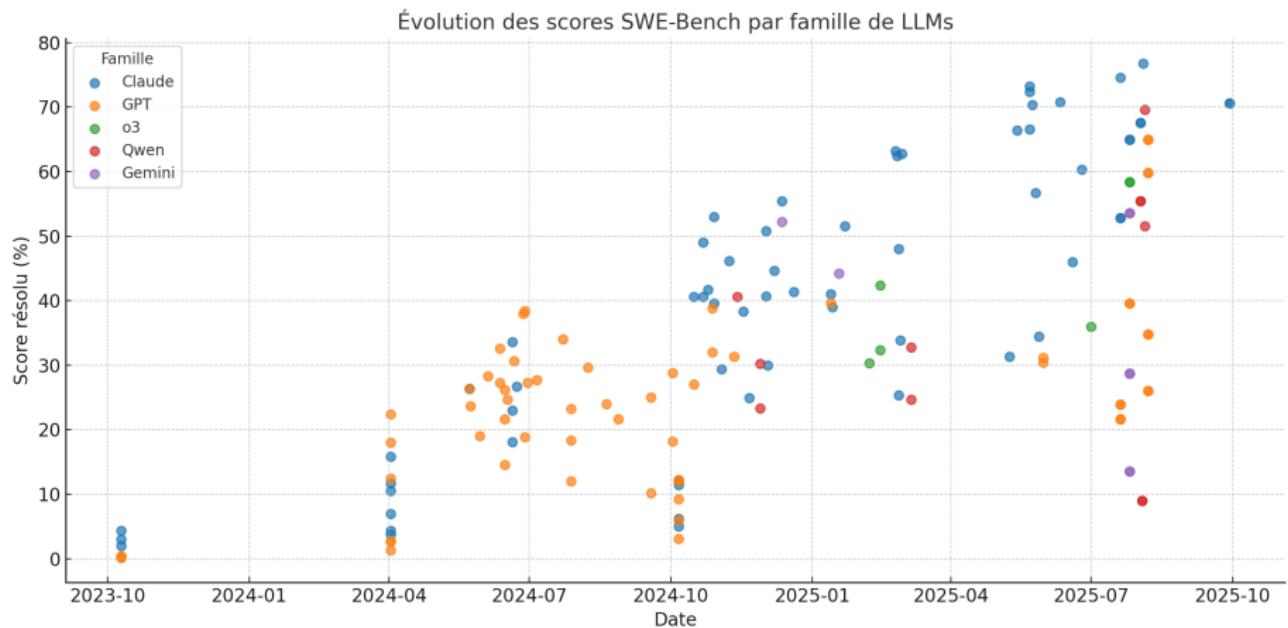
Conclusion : Choisir entre contrôle, flexibilité et performances optimales.

Comparaison des performances des LLMs : benchmarks classiques

Comparaison des LLMs grand public sur MMLU Pro, SWE-bench Verified et GPQA Diamond



Evolution des performances des LLMs sur SWE-Bench



Introduction au prompting

Définition : Le prompting est l'art de formuler des instructions à un modèle de langage afin d'obtenir une réponse pertinente et de qualité.

Idée clé : La qualité de la sortie dépend fortement de la manière dont l'entrée est rédigée.

Enjeux :

- Obtenir des résultats précis et cohérents.
- Réduire les risques d'ambiguités et d'hallucinations.
- Adapter le style et le niveau de détail de la réponse.

Techniques de base du prompting

- **Prompt clair et explicite** : éviter les formulations vagues.
- **Contexte** : fournir des informations supplémentaires pour orienter le modèle.
- **Format attendu** : indiquer la structure de sortie souhaitée (liste, tableau, texte court).
- **Exemples** : montrer un ou plusieurs cas pour guider la génération.

Exemple : Au lieu de "C'est quoi le machine learning ?", écrire "Donne une explication détaillée du fonctionnement du machine learning, illustrée avec un exemple".

1. Le ton change la structure du raisonnement

Principe

Le ton du prompt influence non seulement le style, mais aussi la structure logique de la réponse.

Effets observés

- Ton calme et analytique : produit une écriture déductive et ordonnée, où le modèle explicite les causes et développe les idées avant de conclure.
- Ton urgent ou tendu : génère une écriture télégraphique, orientée vers l'action, avec phrases courtes et listes de décisions.

Exemples

Explique posément, en suivant un raisonnement logique. Vite : résume-moi ce que je dois dire au client dans deux minutes.

À retenir

Le ton est un levier cognitif : il modifie la manière dont le modèle raisonne avant même d'écrire

2. Les métaphores encadrent la pensée

Principe

La métaphore impose un cadre de raisonnement : elle force le modèle à réencoder le concept dans une structure familière.

Effets observés

- Elle permet d'aborder des notions abstraites par transfert analogique.
- Elle structure naturellement la réponse selon les dimensions de la métaphore (par exemple : ingrédients, recettes, cuisson).

Exemples

Explique le machine learning comme si c'était une cuisine. Explique l'optimisation comme une randonnée avec des étapes et des sommets.

À retenir

La métaphore agit comme un cadre cognitif qui oriente la logique et le vocabulaire du texte.

3. Les rôles activent des corpus latents

Principe

Attribuer un rôle au modèle sélectionne une manière de parler et de raisonner issue de son corpus.

Effets observés

- Historien : met l'accent sur la chronologie et les causes profondes.
- Ingénieur sceptique : insiste sur les hypothèses, les tests et les limites.
- Poète critique : privilégie les images, les tensions et le rythme.

Exemples

Agis comme un historien en expliquant les causes profondes. Réponds comme un ingénieur qui doute de ses propres hypothèses.

À retenir

Le rôle agit comme un filtre stylistique et culturel qui oriente la réponse sans en changer le fond.

4. La précision du contexte vaut plus que la longueur

Principe

Un contexte précis et incarné produit de meilleures réponses qu'une longue liste d'instructions abstraites.

Effets observés

- Le modèle comprend mieux la situation lorsqu'elle est ancrée dans un cadre concret.
- Les réponses deviennent plus pertinentes et plus naturelles.

Exemples

Tu écris à ton supérieur, ton ferme mais respectueux, objectif : obtenir une validation avant 18 h. En 200 mots, identifie trois risques et formule une demande de décision.

À retenir

Un cadre incarné aide le modèle à raisonner comme dans une situation réelle.

5. Les verbes d'action changent la dynamique

Principe

Le verbe choisi détermine le type de raisonnement attendu.

Effets observés

- Explique : logique de cause à effet.
- Montre : logique d'exemple et d'illustration.
- Révèle : logique d'enquête ou de découverte.

Exemples

Explique en trois causes principales. Montre deux exemples concrets et un contre-exemple. Révèle l'élément caché qui change la conclusion.

À retenir

Le verbe oriente la structure du texte et la nature du raisonnement.

6. Le silence comme outil

Principe

Ne pas tout préciser laisse au modèle une marge d'interprétation, propice à la créativité.

Effets observés

- Le modèle complète avec des régularités contextuelles, souvent pertinentes.
- Trop de contraintes bloque la pensée et conduit à des réponses formatées.

Exemples

Donne trois options, dont une plus audacieuse, sans les justifier. Rédige un brouillon brut, sans titres, avec les idées clés uniquement.

À retenir

Le vide agit comme un espace de respiration qui stimule la créativité.

7. Les émotions modulent l'énergie du contenu

Principe

Le ton émotionnel du prompt influence le rythme et la profondeur de la réponse.

Effets observés

- Colère : texte direct, tranchant, orienté vers la critique.
- Tristesse : rythme lent, ton grave, introspection.
- Joie : rythme rapide, style fluide et enthousiaste.
- Admiration : langage figuré et ample.

Exemples

Sans détour, dis-moi ce qui ne va pas dans cette stratégie. Parle avec gravité des risques systémiques en trois points.

À retenir

L'émotion structure le raisonnement en modulant l'énergie et la focalisation.

8. Le paradoxe pousse la nuance

Principe

Introduire une contradiction dans le prompt oblige le modèle à produire une réponse nuancée.

Effets observés

- Le modèle cherche un équilibre entre deux pôles logiques.
- Il produit une pensée dialectique plutôt qu'un simple commentaire.

Exemples

Explique pourquoi l'IA est à la fois une menace et une promesse. Montre en quoi le progrès peut être à la fois un risque et une chance.

À retenir

Le paradoxe favorise la nuance et la réflexion véritable.

1. Le prompt comme dispositif de situation

- Les modèles réagissent mieux à une situation incarnée qu'à une instruction abstraite. → Exemple : Vous êtes face à un comité sceptique, vous devez défendre votre position.
- L'efficacité vient d'un processus itératif : cadrer, ajuster, affiner le ton et la forme.
- Les consignes les plus performantes précisent l'objectif plutôt que les moyens.
→ Exemple : Rédige un texte qui donne envie de s'inscrire à la formation.

Un bon prompt crée un contexte de pensée, pas une suite d'ordres.

2. Créer un cadre ouvert mais dirigé

- Laisser une marge d'interprétation encourage la créativité sans perdre le contrôle. → Exemple : Propose trois options, dont une plus audacieuse.
- La reformulation est un outil d'ancrage : elle stabilise le style et le ton. → Exemple : Conserve ce ton, mais rends le propos plus concret.
- Les prompts dialectiques favorisent la nuance et la profondeur. → Exemple : Expose les arguments pour et contre l'usage de l'IA générative.

L'ouverture maîtrisée est plus féconde que la prescription exhaustive.

3. Soigner l'entrée du modèle

- La première phrase fixe le registre cognitif du texte. → Agis comme... crée un ton professionnel. → Imagine que... stimule la créativité. → Explique-moi... installe un registre analytique.
- Certains mots volontairement flous activent des régularités de style. → Exemple : Écris un texte juste conduit souvent à une écriture sobre et sincère.

Les premiers mots d'un prompt déterminent souvent la posture du modèle.

4. Guider la génération au fil du dialogue

- Les relances directionnelles (Va plus loin, Simplifie sans appauvrir) orientent la production sans l'alourdir.
- Demander au modèle de se corriger déclenche une forme de métacognition.
→ Exemple : Relis ta réponse comme un relecteur exigeant. Que modifierais-tu ?
- Le feedback positif est plus structurant que la contrainte négative. → Il permet au modèle d'ajuster progressivement sa cohérence stylistique.

Le dialogue, bien conduit, devient un véritable apprentissage contextuel.

5. Dialoguer avec une voix, pas avec une machine

- Les modèles ne obéissent pas : ils estiment la suite la plus probable. → Il s'agit donc d'orienter plutôt que d'imposer.
- Tu peux être incisif produit souvent de meilleurs résultats que Sois incisif.
- Initier un ton puis le prolonger (Continue dans le même style) garantit la cohérence d'ensemble.
- Les LLMs sont des condensés de mémoire culturelle : plus le langage est humain, plus la réponse le sera.

Un prompt efficace s'adresse à une voix, pas à un programme.

6. Structurer le prompt : du zero-shot au few-shot

- **Zero-shot** : le modèle exécute la tâche uniquement à partir d'une consigne.
→ Adapté aux tâches simples ou familières. → Exemple : Rédige un e-mail de relance professionnel pour un client inactif.
- **Few-shot** : on fournit au modèle quelques exemples du comportement attendu avant la demande réelle. → Le modèle en déduit la structure, le ton et le registre à reproduire.
- **Chain-of-thought** : on invite le modèle à expliciter son raisonnement avant la conclusion. → Exemple : Décris ton raisonnement étape par étape avant de proposer la recommandation finale.

La structuration du prompt agit comme un guide cognitif : elle oriente la logique, la précision et la cohérence du résultat.

Modèles de raisonnement

Objectif : améliorer la capacité des LLMs à résoudre des problèmes complexes en rendant explicites les étapes de raisonnement.

Principe des Chain-of-Thoughts (CoT) :

- Décomposer une question en **étapes intermédiaires logiques**.
- Forcer le modèle à “**penser à voix haute**” avant de donner la réponse finale.
- Exemple : résoudre une équation, planifier une suite d’actions, raisonner sur des données tabulaires.

Méthodes d’entraînement :

- **Fine-tuning supervisé** sur des datasets annotés avec étapes de raisonnement.
- **Self-consistency** : générer plusieurs chaînes et agréger la réponse la plus fréquente/cohérente.
- **Distillation** : transférer les capacités de raisonnement d'un grand modèle vers un plus petit.

Agents basés sur les LLMs

Définition : un agent est un système qui exploite un LLM comme moteur de raisonnement et de décision, capable d'interagir avec des outils et un environnement externe.

Caractéristiques principales :

- **Boucle perception-action** : observe, raisonne, agit, et réévalue.
- **Accès à des outils** : API, bases de données, navigateurs, scripts.
- **Mémoire** : conserve l'historique et les connaissances pertinentes.
- **Planification** : décompose un objectif en sous-tâches exécutables.

Limites actuelles des LLMs : deux axes majeurs

1. Hallucinations

- Génération de contenus inexacts ou inventés, parfois avec une forte confiance.
- Difficulté à distinguer le vrai du plausible, surtout hors du domaine d'entraînement.
- Conséquences : risque de désinformation, perte de fiabilité, nécessité de vérification humaine.

2. Non-déterminisme

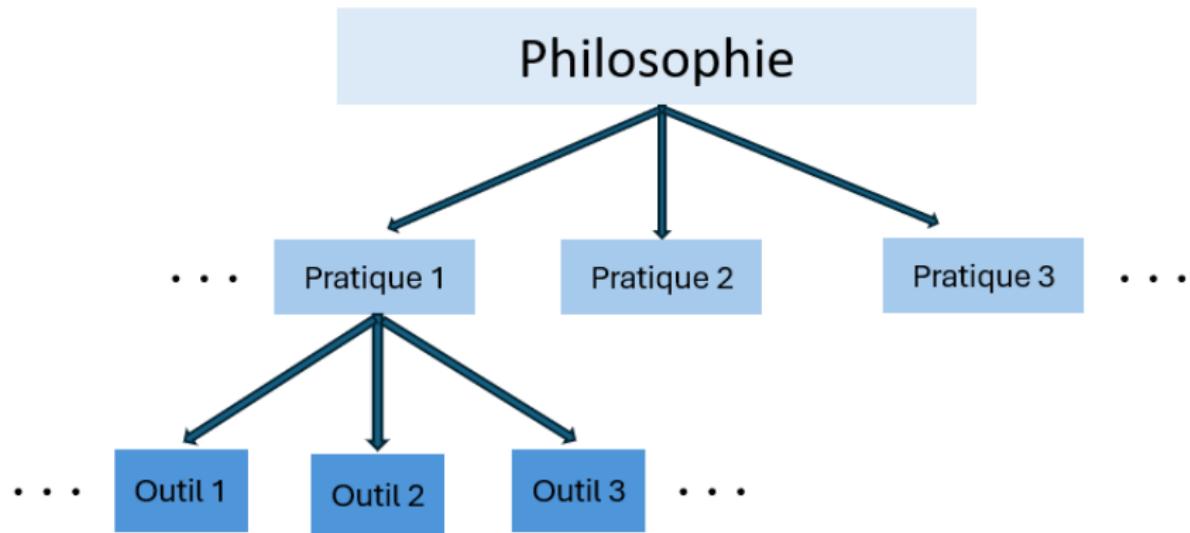
- Un même prompt peut produire plusieurs réponses différentes.
- Résultats sensibles à la température, au contexte ou à de légères variations de formulation.
- Enjeu pour la reproductibilité et le contrôle qualité des systèmes fondés sur les LLMs.

Industrialisation du machine learning et MLOPS ?

Définition

Le MLOps est un ensemble de pratiques et d'outils mis en oeuvre pour automatiser et optimiser le déploiement, la gestion et la maintenance des modèles de machine learning en production.

Qu'est-ce que le MLOps



Cycle de vie d'un projet de machine learning



Mais ce processus est en réalité très itératif.

Pourquoi le MLOps ?

Les différents problèmes que l'on peut rencontrer à l'interface entre la phase machine learning et la phase opérations.

- Problèmes de communication
- Problèmes techniques
- Problèmes de surveillance et de maintenance des modèles
- Problèmes de gestion des données
- Problèmes liés au déploiement et à l'automatisation
- Problèmes de collaboration organisationnelle
- Problèmes liés à la sécurité et à la conformité
- Problèmes de compétences et de formation

Différence de nature entre les projets Data et Dev

Projets data

- Nature exploratoire, résultats incertains
- Estimation des délais difficile
- "Produit fini" flou, amélioration continue
- Attentes parfois irréalistes
- Métriques de succès statistiques
- Cycle de vie : mises à jour fréquentes
- Confort avec l'ambiguïté, exploration

Projets dev

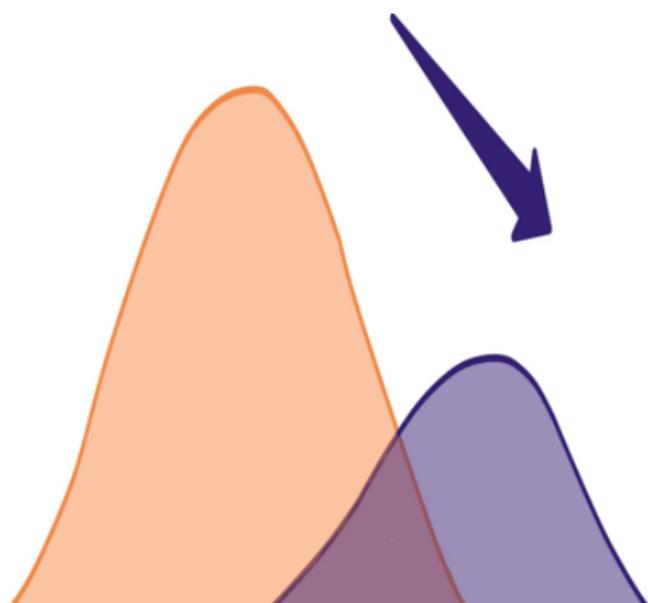
- Nature déterministe, spécifications claires
- Estimations plus précises
- "Produit fini" défini par les spécifications
- Attentes mieux alignées techniquement
- Métriques liées aux fonctionnalités
- Cycle de vie plus traditionnel
- Tendance à réduire l'incertitude

0. Problème de data drift

Le data drift est le phénomène de changement dans la distribution des données.

Il apparaît d'une manière fréquente dans les modèles mis en production. Il peut être dû à plusieurs facteurs :

- Différence entre les données de test et les données en production.
 - Changement de l'environnement (avant et après le Covid par exemple).
 - Changement dans le comportement des utilisateurs ou clients.
 - Effets de l'algorithme lui-même sur les utilisateurs.
 - Etc.



1. Problèmes de communication

- **Terminologie et langage différents :**

Les data scientists et les développeurs parlent souvent des "mêmes" concepts avec des terminologies différentes. Par exemple, un développeur peut parler d'API ou de scalabilité, tandis qu'un data scientist peut parler de modèles prédictifs et de dérive des données. Cette différence peut créer des incompréhensions.

- **Objectifs mal alignés :**

Les priorités des deux équipes peuvent différer, avec les data scientists axés sur la précision des modèles et les développeurs concentrés sur la stabilité et l'optimisation du code en production. Cette divergence d'objectifs peut entraîner des tensions si les équipes ne communiquent pas clairement leurs attentes respectives.

- **Manque de documentation :**

Les data scientists peuvent fournir des modèles ou des scripts sans documentation complète, ce qui rend difficile pour les développeurs de comprendre le contexte ou les paramètres du modèle lors du déploiement en production.

2. Problèmes techniques

- **Prototypage vs production :**

Le code écrit dans les notebooks n'est pas directement utilisable en production. Il manque souvent de structure, de gestion des erreurs et de tests nécessaires pour être fiable en environnement de production.

- **Scalabilité et performance :**

Les modèles et pipelines développés par les data scientists ne sont pas toujours adaptés pour une exécution à grande échelle en production. Les problèmes de performance (latence, mémoire) peuvent survenir lors du déploiement à grande échelle.

- **Compatibilité des outils :**

Les outils utilisés en exploration (Pandas, scikit-learn) ne sont pas toujours compatibles avec les environnements de production (Kubernetes, Docker, TensorFlow Serving), nécessitant des ajustements ou réécritures du code.

3. Problèmes de gestion des données

- **Qualité et gouvernance des données :**

Les data scientists ont besoin de données propres et de haute qualité, mais si la gouvernance des données n'est pas solide, cela peut entraîner des modèles biaisés ou des erreurs.

- **Accès et gestion des données :**

Il peut y avoir des problèmes de gestion des droits d'accès aux données entre les équipes, ce qui ralentit les projets ou complique les échanges de données entre les équipes data et dev.

- **Versioning des données :**

Les modèles peuvent être entraînés sur une version des données et mis en production avec une autre, ce qui entraîne des incohérences. La gestion des versions des données n'est pas toujours bien synchronisée entre les équipes.

4. Problèmes liés au déploiement et à l'automatisation

- **Déploiement manuel :**

Les modèles sont souvent développés dans des environnements isolés et il peut être difficile de les déployer manuellement en production sans automatisation.

- **MLOps immature :**

Si les pratiques MLOps (intégration et déploiement continus pour les modèles) ne sont pas mises en place, il peut être difficile de maintenir les modèles et de les réentraîner régulièrement en production.

- **Maintenance des modèles :**

Une fois en production, les modèles doivent être surveillés et mis à jour. Sans une automatisation du monitoring et de la reformation, les modèles peuvent devenir obsolètes et imprécis.

5. Problèmes de collaboration organisationnelle

- **Silos organisationnels :**

Les équipes data et dev travaillent souvent dans des silos, ce qui entraîne une mauvaise communication et une collaboration limitée sur les projets communs.

- **Problèmes de priorisation :**

Les équipes data peuvent prioriser des améliorations de modèles, tandis que les équipes dev se concentrent sur la stabilité du système, entraînant des désaccords sur la priorisation des projets.

- **Propriété et responsabilité :**

Il peut être difficile de définir qui est responsable de la performance continue des modèles en production. Les développeurs s'occupent souvent de l'infrastructure, mais les data scientists sont responsables de la précision des modèles.

6. Problèmes liés à la sécurité et à la conformité

- **Protection des données :**

Les data scientists et les développeurs doivent gérer des données sensibles, ce qui impose des contraintes de sécurité pour éviter les violations de la confidentialité (ex : RGPD).

- **Gestion des accès :**

Il est important de s'assurer que seuls les membres autorisés peuvent accéder aux données critiques. Une mauvaise gestion des accès peut entraîner des fuites de données ou des violations de conformité.

Les différentes organisations des équipes Data et Dev

Dans le développement et le déploiement des projets de Machine Learning, plusieurs types d'organisation sont possibles pour maximiser la collaboration et l'efficacité entre les équipes de data science, de développement et d'ingénierie.

- ① **Approche en silos (traditionnelle)** : Les équipes travaillent de manière indépendante, avec transfert de responsabilités des data scientists aux devs.
- ② **Approche collaborative (cross-fonctionnelle)** : Les équipes collaborent dès le début du projet et tout au long du cycle de vie du modèle.
- ③ **Approche DevOps/MLOps** : Automatisation des processus d'entraînement, de déploiement et de surveillance des modèles à travers des pipelines CI/CD.
- ④ **Équipe centralisée (Center of Excellence)** : Une équipe dédiée centralise les compétences et les processus pour tous les projets de ML de l'entreprise.
- ⑤ **Squads dédiés ou équipes produits (Product Teams)** : Des équipes autonomes et multidisciplinaires sont responsables du développement de produits spécifiques intégrant des modèles ML.
- ⑥ **Approche par chapitre et guilde (Spotify Model)** : Une combinaison d'équipes autonomes (squads) et de groupes d'experts (chapitres) partageant des pratiques et des standards communs.

Approche en silos (traditionnelle)

Dans cette organisation, les équipes de data science et de développement fonctionnent de manière indépendante. Les data scientists développent leurs modèles en isolation, puis passent le modèle aux développeurs pour le déploiement.

Avantages :

- Chaque équipe se concentre sur son expertise principale.
- Les processus peuvent être optimisés indépendamment pour chaque équipe.

Inconvénients :

- Manque de communication, ce qui peut entraîner des problèmes d'intégration en production (incompatibilité de l'infrastructure, erreurs dans le pipeline de données).
- Les retours d'expérience des développeurs arrivent trop tard, après la phase de conception du modèle.
- Problèmes de duplication d'efforts et de perte de temps lors des itérations.

Quand l'utiliser :

- Pour des projets simples et bien définis où les étapes sont clairement séparées.

Approche collaborative (cross-fonctionnelle)

Les équipes de data science et de développement travaillent ensemble dès le début du projet. Les développeurs participent aux étapes de conception des modèles, et les data scientists collaborent pendant le développement des systèmes de production.

Avantages :

- Meilleure communication entre les équipes.
- Les problèmes d'intégration sont détectés plus tôt.
- Les modèles sont conçus pour être déployés plus rapidement, avec moins de risques d'incompatibilités techniques.

Inconvénients :

- Plus de coordination et de gestion de projet nécessaire.
- Peut ralentir le développement initial en raison des échanges constants.

Quand l'utiliser :

- Pour des projets complexes où l'intégration du modèle dans le système de production est cruciale et nécessite une optimisation fine.
- Lorsque des itérations fréquentes entre le développement du modèle et le déploiement sont nécessaires.

Équipe centralisée (Center of Excellence)

Dans cette organisation, une équipe centralisée est responsable à la fois du développement des modèles de machine learning et de leur déploiement. Elle regroupe des compétences en data science, en développement et en MLOps.

Avantages :

- Unité dans la gestion des processus et des technologies.
- Facilite la collaboration entre les disciplines.
- Meilleure standardisation des pratiques et des outils utilisés dans toute l'entreprise.

Inconvénients :

- Peut manquer de flexibilité si les besoins spécifiques des équipes sont très variés.
- Risque d'une surcharge de travail pour l'équipe centralisée si l'organisation est trop large.

Quand l'utiliser :

- Pour des entreprises qui veulent centraliser l'expertise en machine learning et l'appliquer à plusieurs départements.

Squads dédiés ou équipes produits (Product Teams)

Les équipes de data science et de développement sont intégrées au sein de petites équipes produits, appelées "squads". Chaque squad est autonome et responsable d'un produit spécifique ou d'une fonctionnalité (ex. : un modèle de recommandation, un algorithme de détection de fraude).

Avantages :

- Les équipes sont alignées sur un objectif commun : le produit.
- Collaboration directe et quotidienne entre data scientists, développeurs, et autres membres du produit.
- Grande réactivité et agilité pour itérer rapidement sur le produit.

Inconvénients :

- Risque de duplication d'efforts si chaque squad développe des solutions similaires sans coordination.
- Peut être difficile à mettre en place dans des grandes organisations avec des équipes réparties.

Quand l'utiliser :

- Dans des organisations avec une forte culture d'agilité ou des entreprises tech où la personnalisation de chaque produit est clé.

Approche par chapitre et guilde

Inspirée du modèle d'organisation de Spotify, cette approche combine des squads (équipes produits) avec des chapitres (groupes d'experts dans un domaine, comme la data science ou le développement) et des guildes (groupes transversaux autour de pratiques communes).

Avantages :

- Maintient une expertise forte au sein des équipes tout en encourageant la collaboration inter-équipes.
- Favorise l'innovation et l'échange de bonnes pratiques.
- Flexibilité pour adapter les équipes aux besoins du projet.

Inconvénients :

- Complexité organisationnelle si mal gérée.
- Peut entraîner des conflits de priorités entre les squads et les chapitres.

Quand l'utiliser :

- Dans des entreprises de taille moyenne à grande qui veulent favoriser l'innovation tout en maintenant une cohérence entre les équipes.

Quels problèmes le MLOps résout-il ?

- **Déploiement** : Les modèles développés en laboratoire ne sont souvent pas directement utilisables en production en raison de différences d'environnement, d'outils, et de processus.
- **Répétabilité** : Les processus manuels dans le développement et le déploiement des modèles sont souvent source d'erreurs, de lenteurs, et de difficultés à reproduire les résultats.
- **Cycle de vie** : Une fois déployés, les modèles ML peuvent perdre en performance au fil du temps (en raison de la dérive des données, par exemple). Sans une gestion appropriée, cela peut entraîner des décisions erronées ou des inefficacités.
- **Collaboration** : Les équipes de data science, d'ingénierie, et d'opérations travaillent souvent en silos, ce qui complique la collaboration et entraîne des frictions lors du déploiement et de la gestion des modèles en production.
- **Scalabilité** : Le déploiement de modèles ML à grande échelle peut être complexe, surtout lorsque plusieurs modèles doivent être gérés, surveillés, et mis à jour simultanément.

Collaboration

Data scientist



Opérations



Passage du code



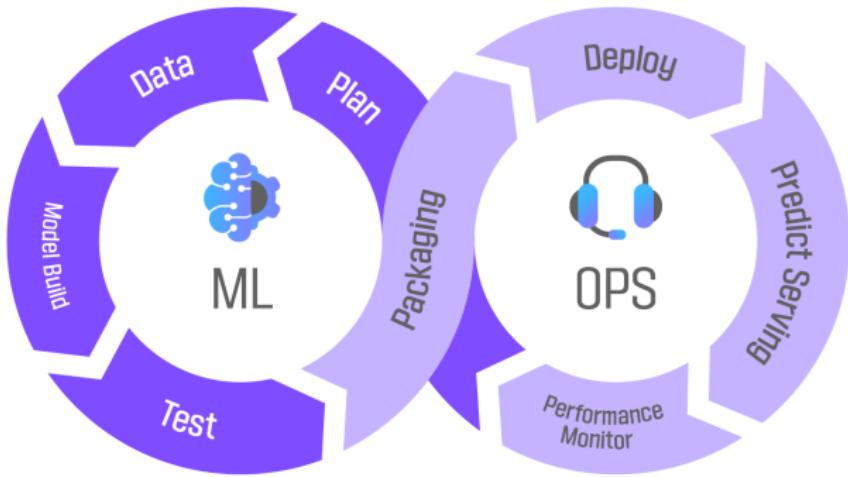
Exposition via API



MLOps



Machine learning + Opérations



Principes du MLOps

- Automatisation du cycle de vie des modèles.
- Intégration continue et déploiement continu (CI/CD).
- Collaboration entre les équipes ML (data scientists) et les équipes opérations (Data engineer, ML engineers, Développeurs).
- Monitoring des modèles et des données en production.
- Versioning et traçabilité du code et des données.
- Optimisation et gestion des ressources, notamment sur le Cloud.

Dans la pratique...

Les approches MLOps peuvent être différentes d'une entreprise à une autre et dépendent de plusieurs facteurs :

- Infrastructure de l'entreprise.
- Organisation de l'entreprise.
- Ressources disponibles en termes de compétences.
- Projets mis en oeuvre.
- Maturité de l'entreprise du point de vue du ML.
- Etc. etc.

Outils



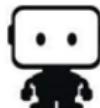
Google Cloud Platform



vertex.ai



Amazon
SageMaker



DataRobot



databricks



data
iku

Plate-formes end-to-end



databricks



dataiku



Kubeflow



DataRobot



Google Cloud



Microsoft Azure

Organisation : principaux métiers de l'IA

Les métiers de l'IA sont en constante évolution et ont subi une évolution assez importantes ces dernières années. On peut distinguer cependant les métiers suivants :

- **Data scientist** : il est au cœur du projet IA. Son principal rôle consiste à traduire le besoin métier en une problématique technique pour élaborer un modèle de ML ou autre, qu'il doit tester et valider avec les métiers. Il est donc normalement amené à travailler étroitement avec les métiers.
- **Machine Learning Engineer** : son rôle consiste à adapter le code développé par le data scientist pour le déployer, souvent sur une infrastructure Cloud (AWS, GCP, Azure, etc.). Il doit ensuite veiller au bon fonctionnement de l'application.
- **Data Engineer** : il est responsable de l'infrastructure liée au projet (pipelines de données, etc.).
- **Data analyst** : son rôle est d'analyser les données pour répondre à des questions métier.
- **Data architecte** : il conçoit l'architecture de la solution qui intègre éventuellement de l'IA (bases de données, back-end, front-end, services Cloud, etc.), notamment quand celle-ci est complexe.

Enjeux éthiques, sécurité et conformité de l'IAG

Pourquoi parler d'éthique et de sécurité ?

L'intelligence artificielle générative ne pose pas seulement des défis techniques : elle soulève des enjeux critiques pour l'entreprise.

- **Sécurité** : risques de fuite de secrets ou d'attaques via les prompts.
- **Confidentialité** : conformité au RGPD, respect des données sensibles.
- **Propriété intellectuelle** : propriété des données d'entraînement et du contenu généré par l'IA.
- **Éthique** : biais, équité, responsabilité dans l'usage.

Message clé : l'adoption de l'IAG doit être accompagnée d'une gouvernance adaptée.

Risques de sécurité

Exemples de menaces :

- **Fuite de secrets** : code ou identifiants envoyés à une API externe.
- **Prompt injection** : manipulation du modèle par un utilisateur malveillant.
- **Dépendance externe** : indisponibilité ou vulnérabilité du fournisseur de LLM.

Bonne pratique : utiliser des proxys, filtrer les prompts et éviter d'envoyer des données sensibles.

Enjeux de confidentialité

Problématique centrale : l'IAG traite souvent des données sensibles ou personnelles.

- **RGPD** : droit à l'oubli, consentement, minimisation des données.
- **Localisation des données** : datacenters UE vs hors UE.
- **Solutions** :
 - anonymisation avant envoi,
 - hébergement de modèles open source en interne,
 - passage par un proxy d'entreprise.

Un enjeu majeur souvent sous-estimé dans l'usage de l'IAG.

- **Origine des données d'entraînement** : risque que le modèle régénère des contenus protégés (texte, code, images).
- **Droits sur les contenus générés** :
 - Dans l'UE, pas de droit d'auteur automatique sur une production 100% IA.
 - L'entreprise doit définir une politique claire (contrats, licences internes).
- **Licences et open source** : vérifier la licence des modèles (Apache 2.0, MIT, restrictions) et des datasets utilisés.
- **Bonnes pratiques** :
 - tracer les sources de données et les outputs,
 - sensibiliser les équipes aux risques de réutilisation non conforme,
 - éviter d'intégrer sans vérification un contenu généré dans des livrables contractuels.

Biais et équité

Constat : les modèles reflètent les biais de leurs données d'entraînement.

- Risque de stéréotypes et de discriminations dans les réponses.
- Impact fort dans les domaines sensibles (recrutement, médical, juridique).
- Besoin d'auditer les modèles dans leur contexte métier.

Message clé : l'évaluation humaine reste indispensable pour corriger les biais.

Questions clés :

- Qui est responsable si une IA génère du code erroné ou trompeur ?
- Comment tracer et auditer les décisions de l'IA ?

Bonnes pratiques :

- Supervision humaine obligatoire pour les tâches critiques.
- Mise en place d'une charte d'usage de l'IAG.
- Définition de rôles et responsabilités clairs.

Conclusion et bonnes pratiques

Synthèse :

- Les enjeux de sécurité, confidentialité et éthique sont incontournables.
- La gouvernance doit accompagner toute intégration de l'IAG.
- Commencer petit, superviser, documenter et auditer.

Bonne pratique : intégrer les aspects éthiques dès la conception des projets IAG.

Merci de votre attention

redha.moulla@axia-conseil.com