Régression

PR.IDRISSI ZOUGGARI NADIA

Introduction

- La **régression** est une technique fondamentale du **Machine Learning supervisé**.
  Son objectif est de **modéliser la relation entre une ou plusieurs variables explicatives (X)** et **une variable cible (Y)** continue, afin de **prédire** cette dernière à partir de nouvelles observations.

Régression linéaire simple

- **X** : le nombre d'années d'expérience.
- **Y** : le salaire observé.
- On observe une **tendance croissante** :

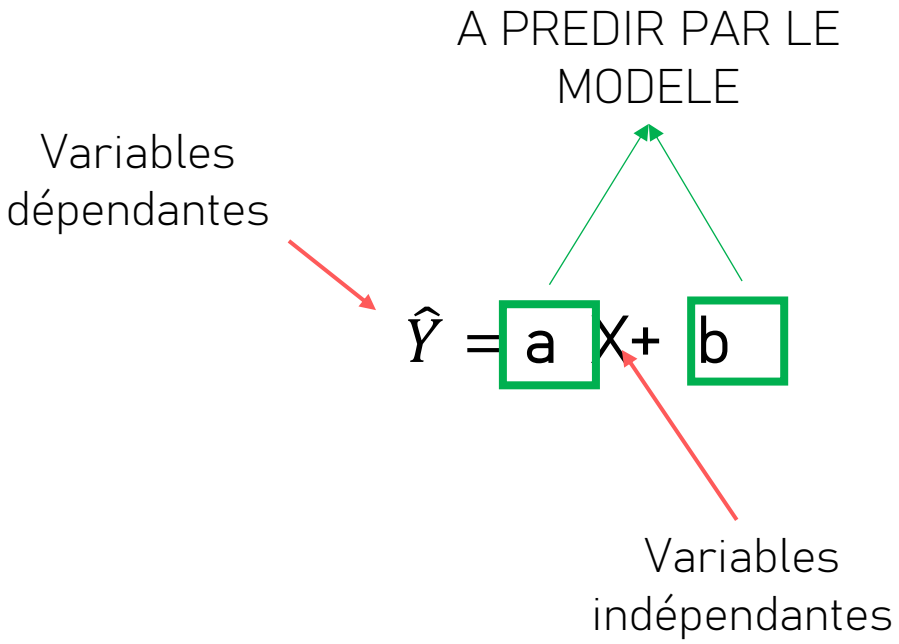
plus l'expérience augmente, plus le salaire est élevé.

- Le **but de la régression linéaire** est de trouver la droite la plus proche possible de ces points, exprimée sous la forme :

$$\hat{Y} = a\ X + b$$

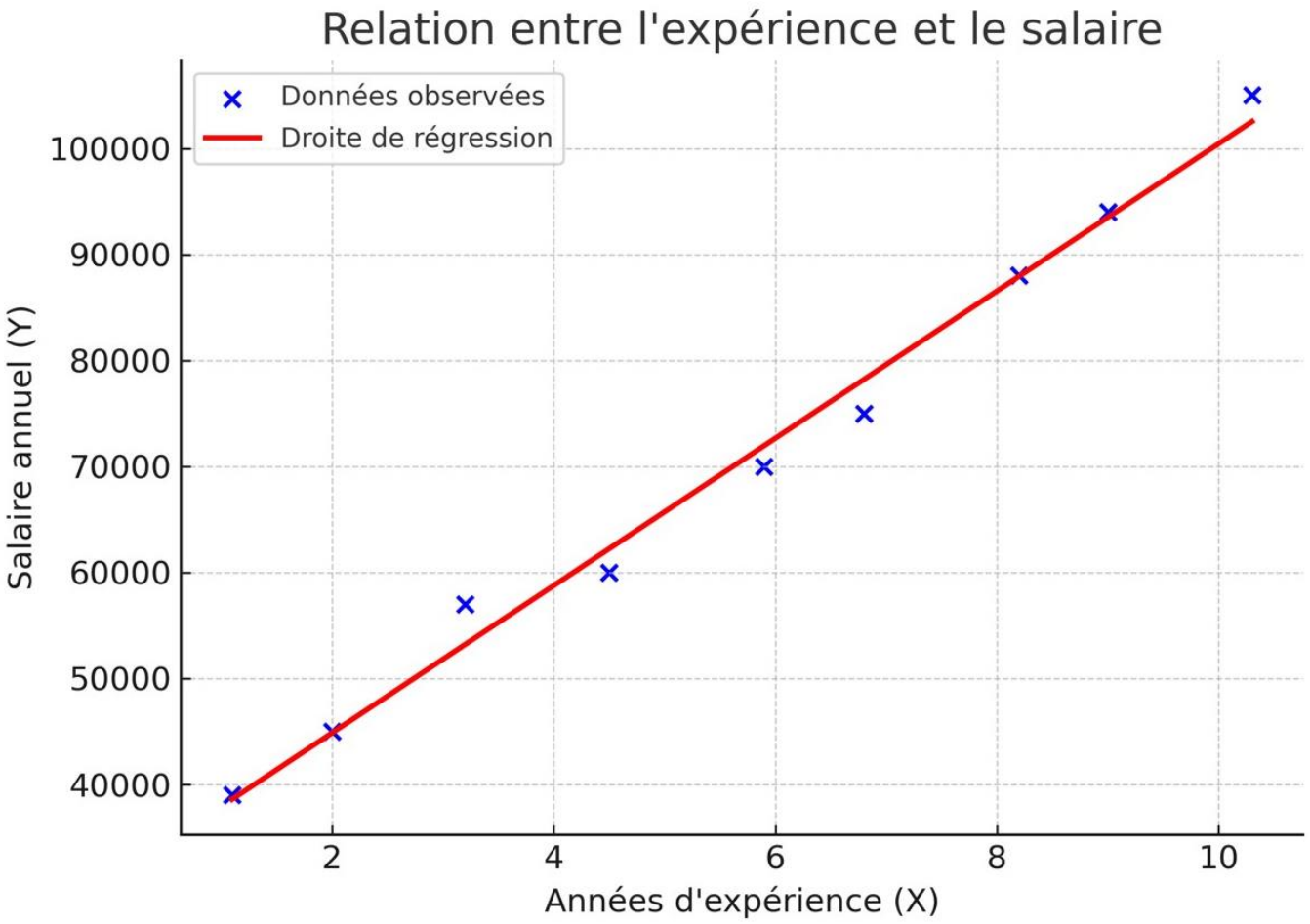
Variables indépendantes X	Variables dépendantes Y
Années d'expérience	Salaire annuel
1.1	39 000
2.0	45 000
3.2	57 000
4.5	60 000
5.9	70 000
6.8	75 000
8.2	88 000
9.0	94 000
10.3	105 000

Régression linéaire simple



Variables indépendantes X	Variables dépendantes Y
Années d'expérience	Salaire annuel
1.1	39 000
2.0	45 000
3.2	57 000
4.5	60 000
5.9	70 000
6.8	75 000
8.2	88 000
9.0	94 000
10.3	105 000

Régression linéaire simple



Variables indépendantes
X

Variables dépendantes
Y

Années d'expérience	Salaire annuel
1.1	39 000
2.0	45 000
3.2	57 000
4.5	60 000
5.9	70 000
6.8	75 000
8.2	88 000
9.0	94 000
10.3	105 000

Influence des coefficients

- la **régression linéaire simple** cherche à ajuster une droite qui relie une variable explicative X(ex. années d'expérience) à une variable cible Y(ex. salaire).

$$\hat{Y} = aX + b$$

- Que se passe-t-il si on **change la valeur de a ou de b** ?

En Machine Learning, a et **b** ne sont pas de simples chiffres. Ce sont les **paramètres** de notre modèle. Ce sont les "boutons" que la machine va apprendre à régler. Notre travail est de comprendre ce que chaque bouton fait.

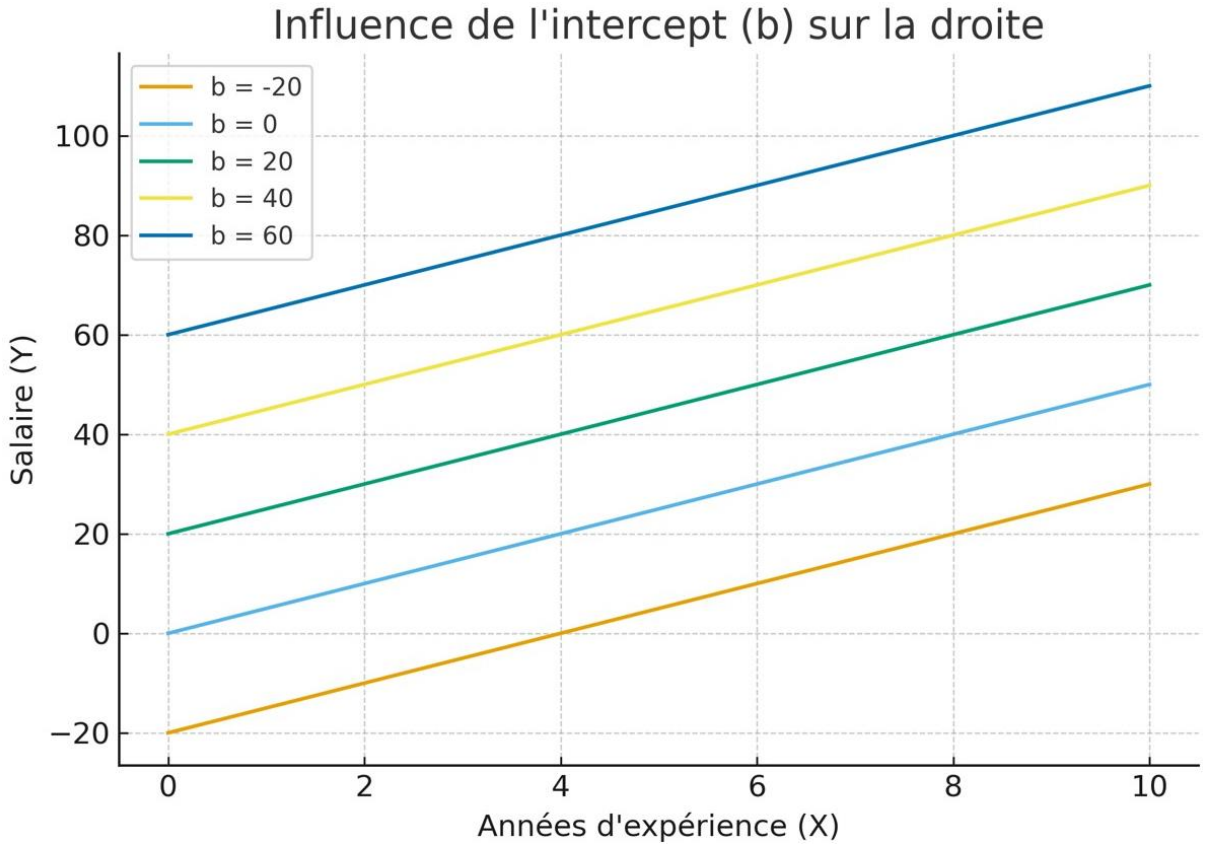
Influence du coefficient b

Le paramètre b est le plus simple à comprendre. C'est la valeur de y lorsque x = 0.

- Son Rôle : Il contrôle la position verticale de la droite.
- Son Influence Visuelle : Changer b fait "monter" ou "descendre" la droite sur le graphique, mais sans changer son inclinaison. C'est une translation verticale.

Exemple Concret :

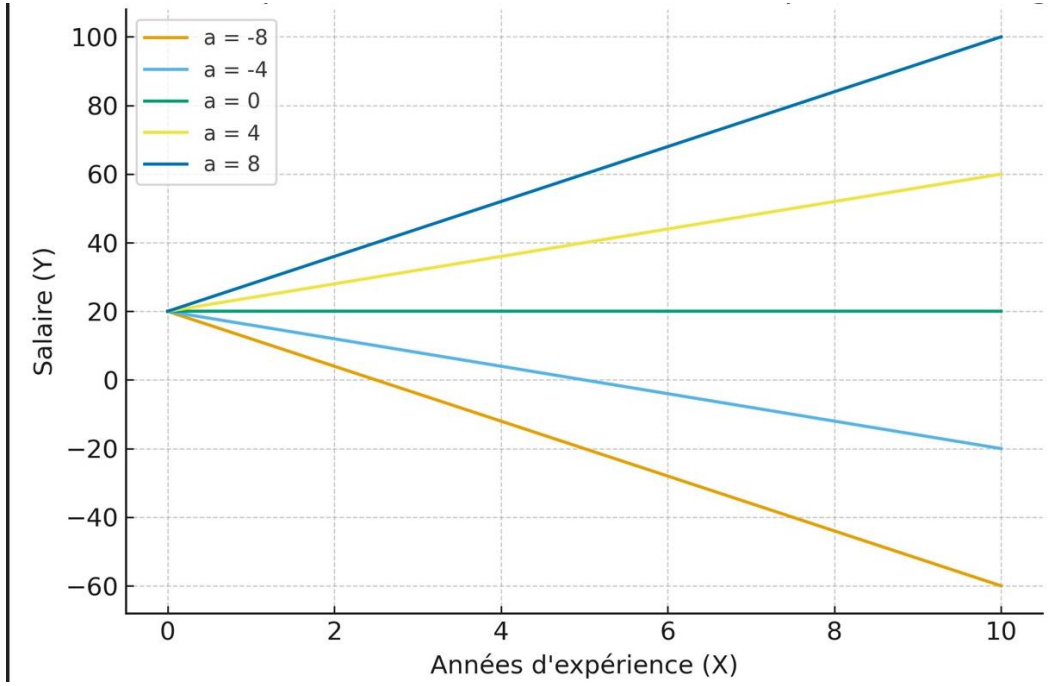
- Équation : Salaire = a × (Années Expérience) + b
- Interprétation : Si Années Expérience= 0, alors Salaire = b.
- B est donc le salaire de base (ou d'embauche) d'une personne n'ayant aucune année d'expérience



Influence du coefficient a

Le paramètre a mesure de combien y change lorsque x augmente d'une seule unité.

- Son Rôle : Il contrôle l'inclinaison et la direction de la droite.
- Son Influence Visuelle :
  - Si a > 0 : La droite "monte". Y augmente quand x augmente.
  - Si a < 0 : La droite "descend". Y diminue quand x augmente.
  - Si a = 0 : La droite est horizontale. x n'a aucune influence sur y.



Exemple Concret :

- Équation : Salaire = a × (Années Expérience) + b
- Interprétation : C'est l'augmentation de Salaire pour une année d'expérience en plus.
- Si a = 5000, cela signifie que pour chaque année d'expérience supplémentaire, le modèle prédit une augmentation de salaire de 5000 DHS .

Fonction coût

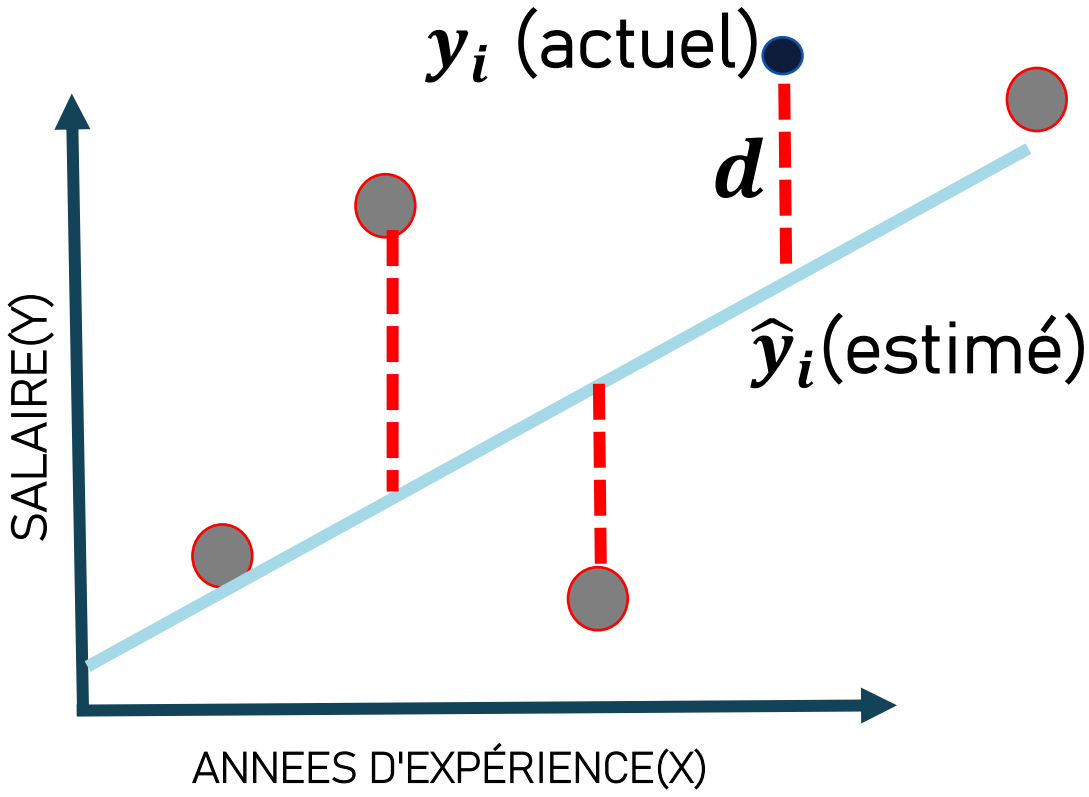
•Pour chaque point de donnée (chaque *vrai* salaire), notre droite propose une *prédiction*.

•La plupart du temps, la prédiction est fausse.

•L'écart vertical entre le point réel (la donnée) et la prédiction (la droite) s'appelle une **erreur**, ou plus techniquement, un **résidu**.

$$résidu = \hat{y}_i - \mathrm{y}_i$$

Une "bonne" droite est une droite qui minimise la somme de toutes ces erreurs. On veut que les écarts verticaux soient, en moyenne, les plus petits possibles



Fonction coût

On ne peut pas juste *additionner* les résidus.

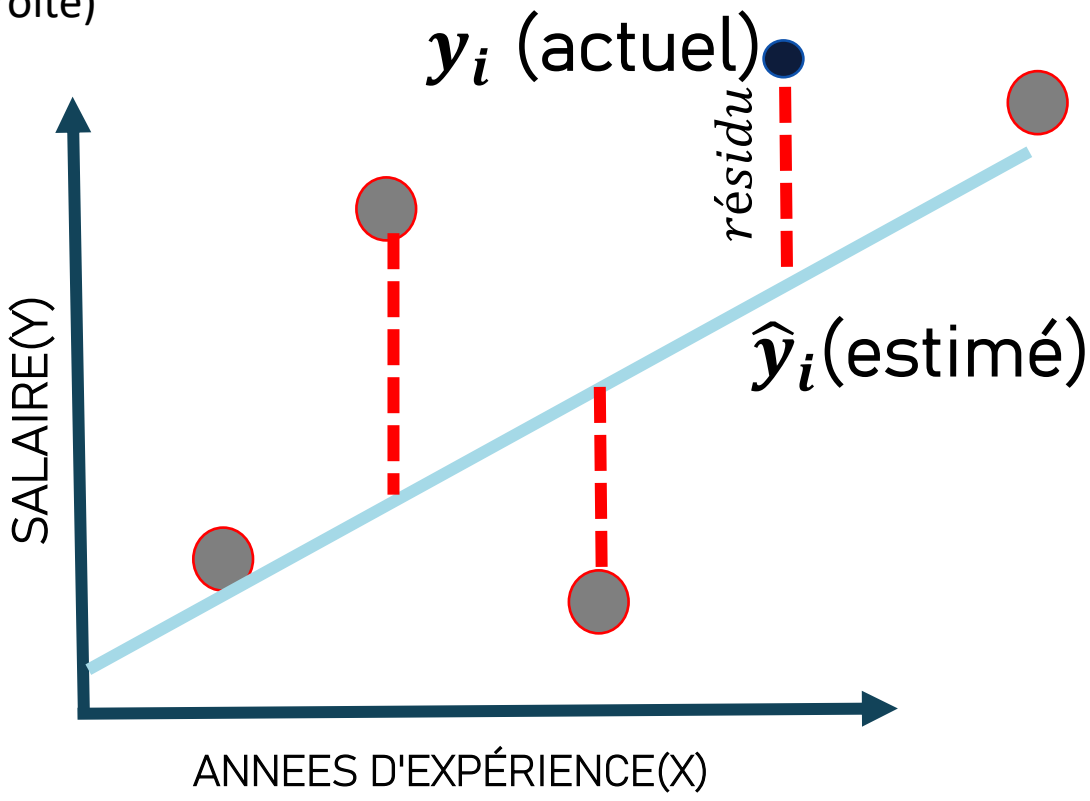
- **Pourquoi ?** Parce que les erreurs positives (au-dessus de la droite) et les erreurs négatives (en dessous) vont **s'annuler** !
- **La solution ?** On met chaque erreur **au carré** avant de les additionner.

•On calcule le **carré** (Squared) de chaque erreur (résidu).

•On calcule la **moyenne** (Mean) de tous ces carrés.

C'est la Moyenne des Erreurs au Carré (MSE)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{réel}} - \hat{y}_{\text{prédit}})^2$$



Fonction coût : MSE

Le MSE (*Mean Squared Error*) ou erreur quadratique moyenne mesure l'écart moyen au carré entre les valeurs réelles y_i et les valeurs prédites $\widehat{y_i}$ par le modèle.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{réel}} - \hat{y}_{\text{prédit}})^2$$

- Plus le MSE est petit, plus les prédictions du modèle sont proches des valeurs réelles.
- Comme les erreurs sont mises au carré, le MSE pénalise fortement les grandes erreurs.

Fonction coût :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{réel}} - \hat{y}_{\text{prédit}})^2$$

$$\text{J(a,b)} = \frac{1}{2n}\sum_{i=1}^{n}(y - (ax + b))^2$$

Nous spécialisons notre 'Métrique' (le MSE) pour un modèle précis : la Régression Linéaire. La prédiction $\hat{Y}$ devient la droite a x + b

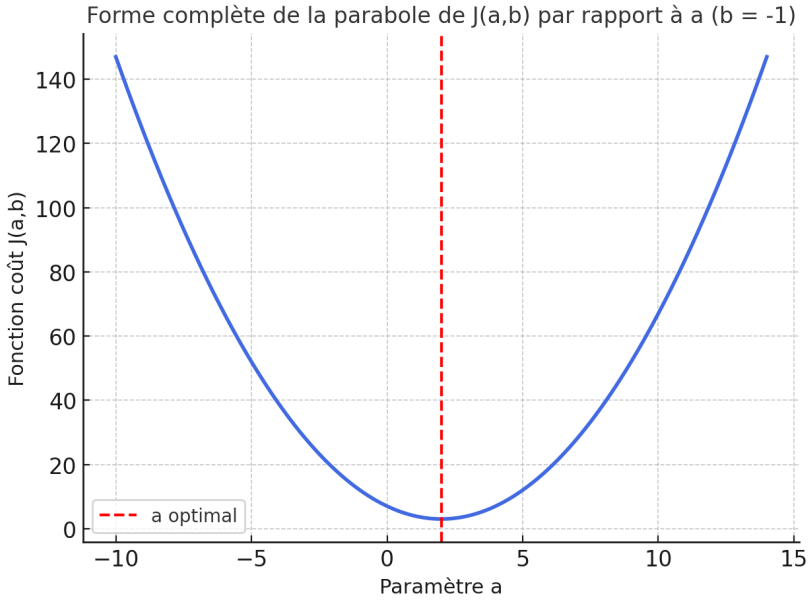
La "Fonction Coût" J dépend donc des paramètres a (la pente) et b (l'ordonnée à l'origine) que l'on cherche à optimiser.

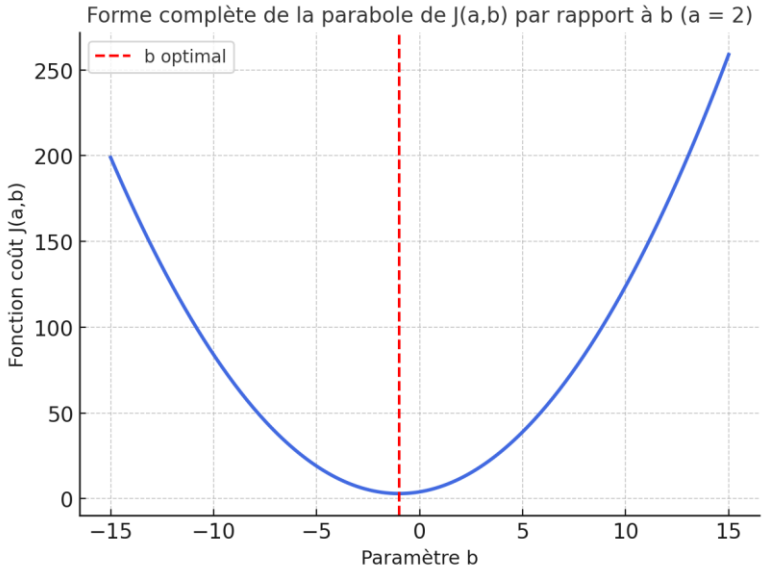
Fonction coût :

$$J(a,b)= \frac{1}{2n}\sum_{i=1}^{n}(y-(ax+b))^2$$

Lorsqu'on fait varier les paramètres a et b de notre modèle, on constate que la fonction coût prend la forme d'une parabole.

Ces graphiques illustrent ce qui se produit lorsqu'on calcule la fonction coût pour différents modèles en faisant varier la valeur de a et de b



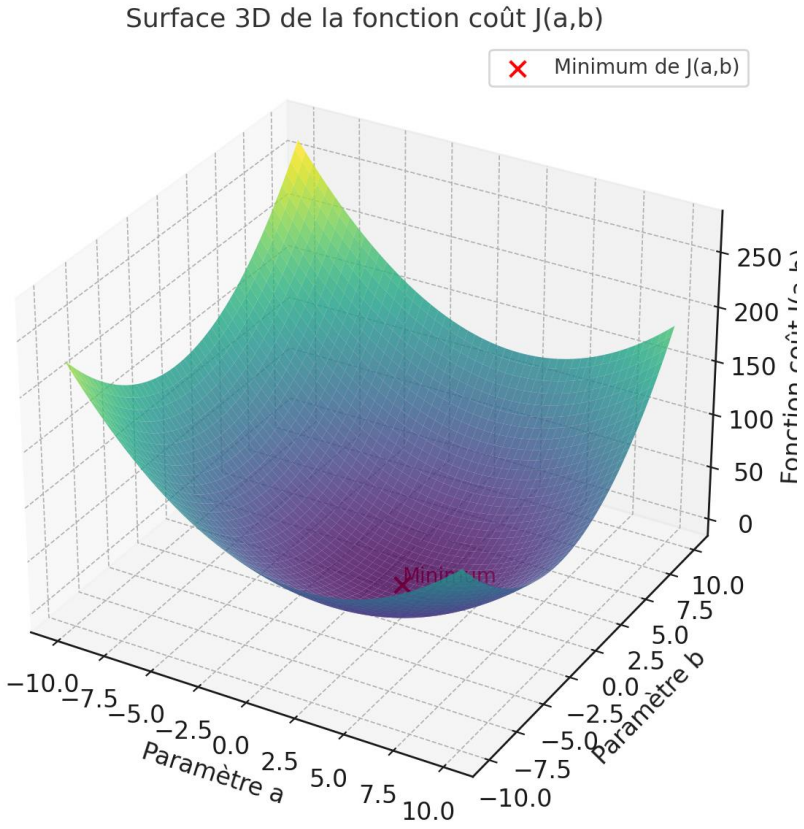


les deux courbes montrent des paraboles , avec un minimum clair pour $a = 2$ et $b = -1$. donc le modèle fait les plus petites erreurs lorsque a=2 et b= -1

Fonction coût :

$$J(a,b) = \frac{1}{2n}\sum_{i=1}^{n}(y-(ax+b))^2$$

Il est possible de visualiser ces deux graphiques en les combinant dans un graphique 3D. On obtient alors une surface en forme de bol, au fond de laquelle se trouve le minimum de la fonction coût, ainsi que les coordonnées (a; b) permettant d'y accéder.



Fonction coût : Optimisation

•La fonction coût permet de mesurer la qualité d'un modèle pour différentes valeurs des paramètres a et b.

•Le minimum de cette fonction représente les paramètres qui entraînent l'erreur la plus faible.

•Dans les exemples graphiques, ce minimum est visible au point le plus bas de la parabole ou de la surface.

•Cependant, dans un problème réel, il est impossible d'évaluer toutes les combinaisons possibles de a et b pour tracer la fonction coût.

•La machine doit donc trouver ce minimum sans connaître la forme complète de la fonction.

•Pour cela, on utilise un algorithme itératif d'optimisation, appelé **descente de gradient**, qui permet d'ajuster progressivement les paramètres jusqu'à atteindre le minimum du coût.

Fonction coût : Optimisation

•L'objectif de la **descente de gradient** est de **trouver automatiquement les valeurs optimales** des paramètres a et b qui minimisent la fonction coût J(A,B).

•L'idée est d'ajuster progressivement les paramètres dans la **direction opposée au gradient** (c'est-à-dire la pente la plus forte de la fonction).

•À chaque itération, les paramètres sont mis à jour selon les formules suivantes :

$$A_i = A_{i-1} - \alpha \frac{\partial J}{\partial A_{i-1}}$$

$$B_i = B_{i-1} - \alpha \frac{\partial J}{\partial B_{i-1}}$$

Fonction coût : Optimisation

- α: est le **taux d'apprentissage (learning rate)**, un coefficient positif qui contrôle la taille du pas ;
- $\frac{\partial J}{\partial A}, \frac{\partial J}{\partial B}$: représentent les **dérivées partielles** du coût par rapport aux paramètres.
- Le processus est **itératif** : à chaque étape, la fonction coût diminue jusqu'à atteindre (ou approcher) le minimum.
- Si le taux d'apprentissage est **trop grand**, l'algorithme risque de **dépasser le minimum**.
- S'il est **trop petit**, la convergence sera **trop lente**.

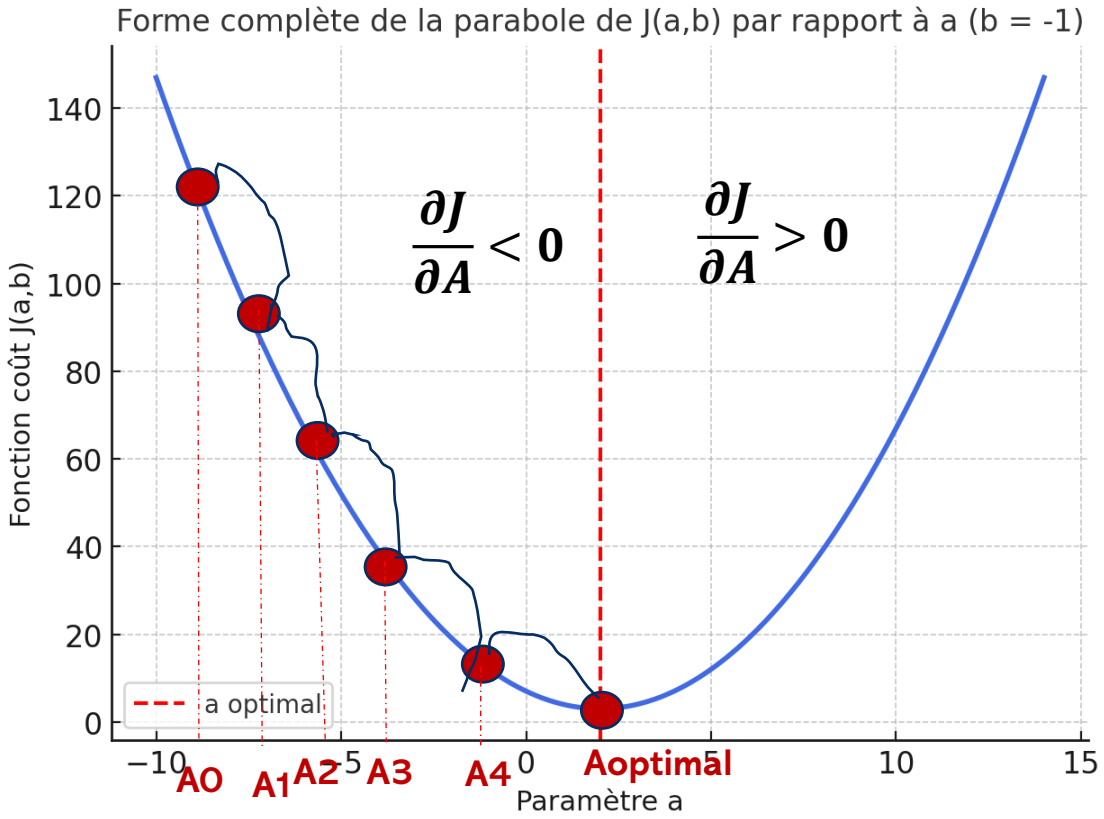
$$A_i = A_{i-1} - \alpha \frac{\partial J}{\partial A_{i-1}}$$

$$B_i = B_{i-1} - \alpha \frac{\partial J}{\partial B_{i-1}}$$

Fonction coût : Optimisation

$$A_i = A_{i-1} - \alpha \frac{\partial J}{\partial A_{i-1}}$$

$$B_i = B_{i-1} - \alpha \frac{\partial J}{\partial B_{i-1}}$$



Métriques d'évaluations: R^2

Le R^2, ou *coefficient de détermination*, mesure la proportion de la variance de Y qui est expliquée par le modèle de régression.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

où :

- $SS_{res} = \sum(y_i - \hat{y}_i)^2$ somme des carrés des résidus (erreurs du modèle)
- $SS_{tot} = \sum(y_i - \bar{y})^2$ somme des carrés des écarts à la moyenne (variance totale)

où :

- y_i :les vraies valeurs observées
- $\bar{y}$:la moyenne de Y
- $\left(y_i - \bar{y}\right)^2$:la distance au carré de chaque point à la moyenne

Métriques d'évaluations: R^2

Quand on fait une régression, le modèle essaie d'expliquer une partie de cette variation.

•Ce qu'il explique bien → c'est la partie "capturée" par la droite de régression.

•Ce qu'il explique mal → ce sont les erreurs (résidus).

Donc :

$$R^2 = 1 - \frac{\textit{erreur du modèle (SSres)}}{\textit{variation totale (SStot)}}$$

Si le modèle explique toute la variation → $R^2 = 1$

S'il n'explique rien → $R^2 = 0$

Métriques d'évaluations: R^2 (exemple)

Imaginons que nos données (notre "problème") sont les suivantes :

Années d'Expérience (X)	Vrai Salaire (Y)
1 an	40 000
2 ans	50 000
3 ans	90 000

Le Salaire Moyen est de (40k + 50k + 90k) / 3 = 60 000

C'est notre point de départ. Si on ne connaît rien à la régression, on prédit 60 000 pour tout le monde.

Vrai Salaire	Prédiction (Moyenne)	Erreur
40 000	60 000	-20 000
50 000	60 000	-10 000
90 000	60 000	+30 000
Erreur Totale (SST)		(On met au carré) ➔ 1 400 000 000

C'est l'erreur totale que nous devons battre. C'est notre 100% du problème.

Métriques d'évaluations: R^2 (exemple)

Maintenant, nous entraînons un modèle de régression. La machine trouve la meilleure droite, disons :
Salaire = 25 000 * Expérience + 15 000.
Calculons ses prédictions et ses erreurs :

Vrai Salaire	**Prédiction (Modèle)**	**Erreur (Résidu)**
40 000	25k*1 + 15k = **40 000**	0
50 000	25k*2 + 15k = **65 000**	-15 000
90 000	25k*3 + 15k = **90 000**	0
Erreur du Modèle (SSR)		(On met au carré) ➔ 225 000 000

Métriques d'évaluations: R^2 (exemple)

Calcul du R^2:

Comparons les deux situations :

1.Erreur Totale : 1 400 000 000

2.Erreur de Notre Modèle : 225 000 000

Notre modèle est bien meilleur ! Son erreur est beaucoup plus petite.

Le R^2 nous dit simplement quel pourcentage de l'erreur totale a été *éliminé* par notre modèle.

•Erreur éliminée = Erreur Totale - Erreur du Modèle

•Erreur éliminée = 1 400 000 000 - 225 000 000 = 1 175 000 000

Pour le mettre en pourcentage : (Erreur Éliminée / Erreur Totale) = 1 175 000 000 / 1 400 000 000 = 0.84

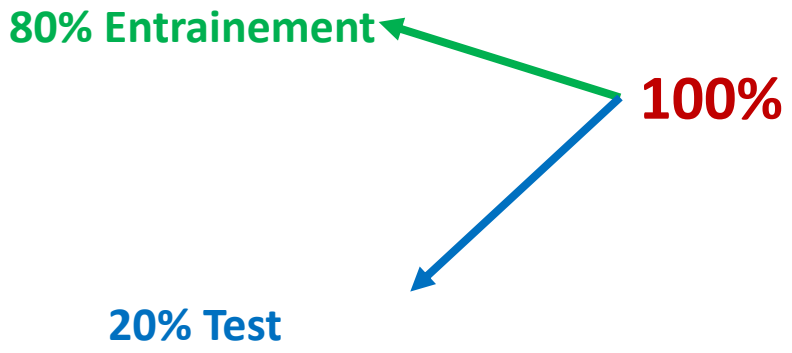
Notre R^2 est de 84%

Apprentissage et Test du modèle

Séparer les données en deux parties pour :

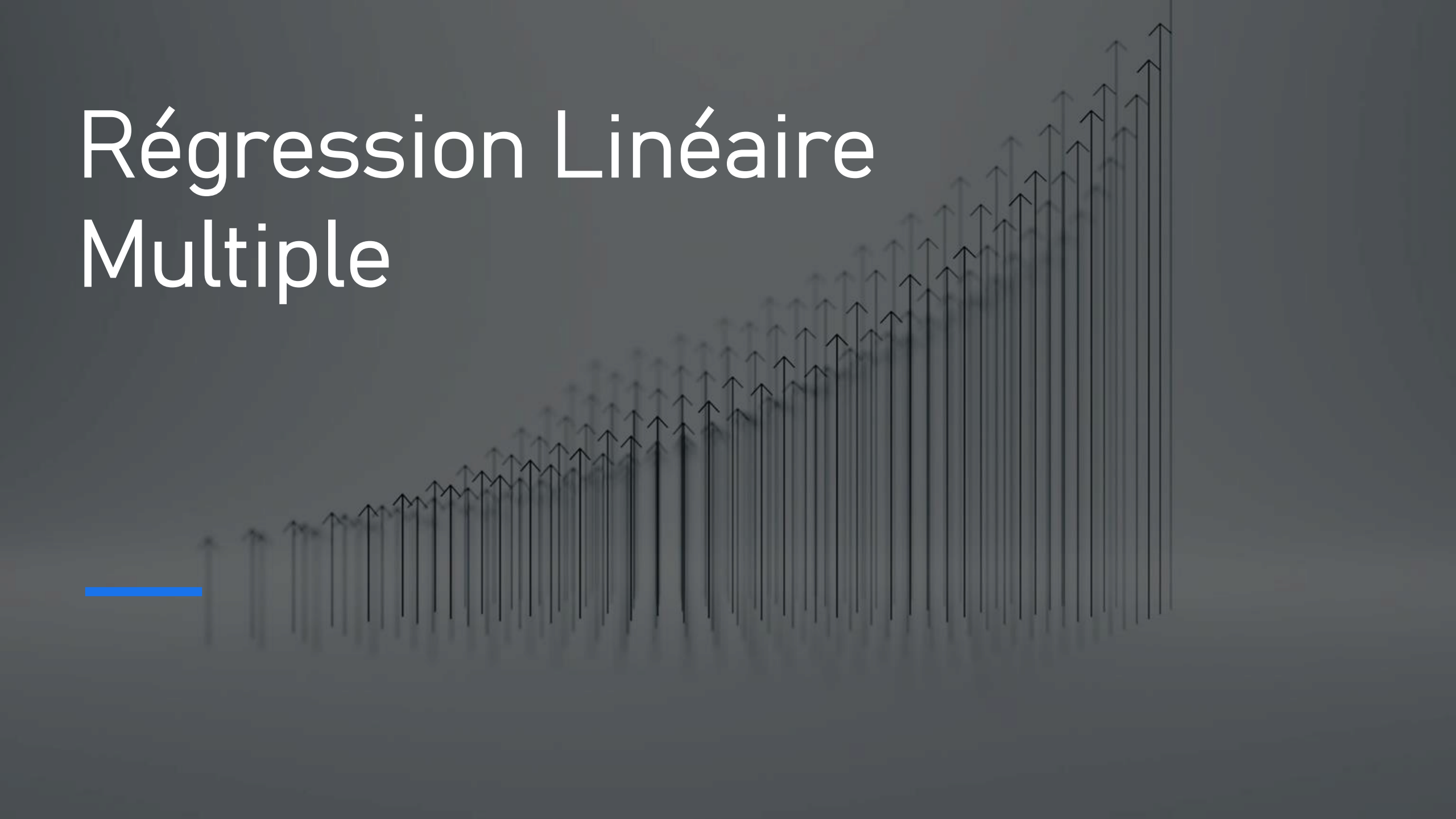
1. Apprendre le modèle sur une partie des données (train),
2. Évaluer sa performance sur une autre partie jamais vue (test).

Cela permet de vérifier la capacité de généralisation du modèle.



Années d'expérience	Salaire annuel
1.1	39 000
2.0	45 000
3.2	57 000
4.5	60 000
5.9	70 000
6.8	75 000
8.2	88 000
9.0	94 000
10.3	105 000

Régression Linéaire Multiple



Régression linéaire Multiple

$$\hat{Y} = b + a_1 X_1 + a_2 X_2 + a_3 X_3$$

$$\text{Salaire} = b + a_1 \times (\acute{E}xp\acute{e}rience) + a_2 \times (\hat{A}ge) + a_3 \times (Niveau\ d'\acute{e}tudes)$$

- Le salaire dépend maintenant de plusieurs facteurs.
- Le modèle devient un plan en 3D ou un hyperplan (si plus de 3 variables)

Régression linéaire Multiple

$$\hat{Y} = b + a_1 X_1 + a_2 X_2 + a_3 X_3$$

Variables indépendantes
Xi

Variables dépendantes
Y: Cible

	Expérience (années)	Âge	Niveau d'études (1=Licence, 2=Master, 3=Doctorat)	Salaire
	1	22	1	28 000
X1	3	25	1	35 000
	5	28	2	48 000
	7	32	2	58 000
	9	35	2	65 000
	11	38	3	74 000
X^7	13	42	3	82 000
	15	45	3	91 000

X_i = i –ème feature

i=1……n , n=3

X_i^j = la ligne j du i –ème colonne

X^7 = [13 42 3]

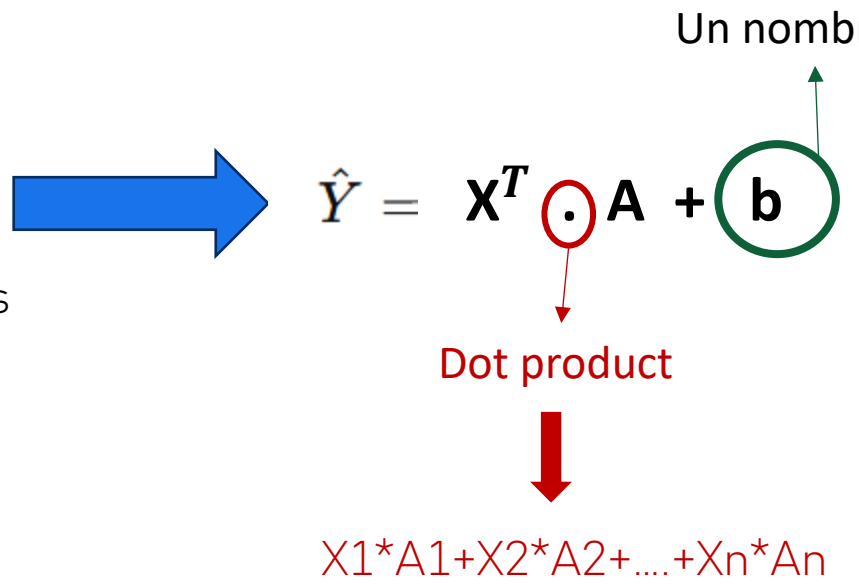
X_1^7 = 13
La valeur de la 1er colonne du 7eme cas

Régression linéaire Multiple

$$\hat{Y} = b + a_1 X_1 + a_2 X_2 + a_3 X_3$$

X=[X_1,X_2,…….X_N] Feature vecteur

A=[a_1,a_2,…….a_N] vecteur de poids



Y=AX+B → **DROITE**

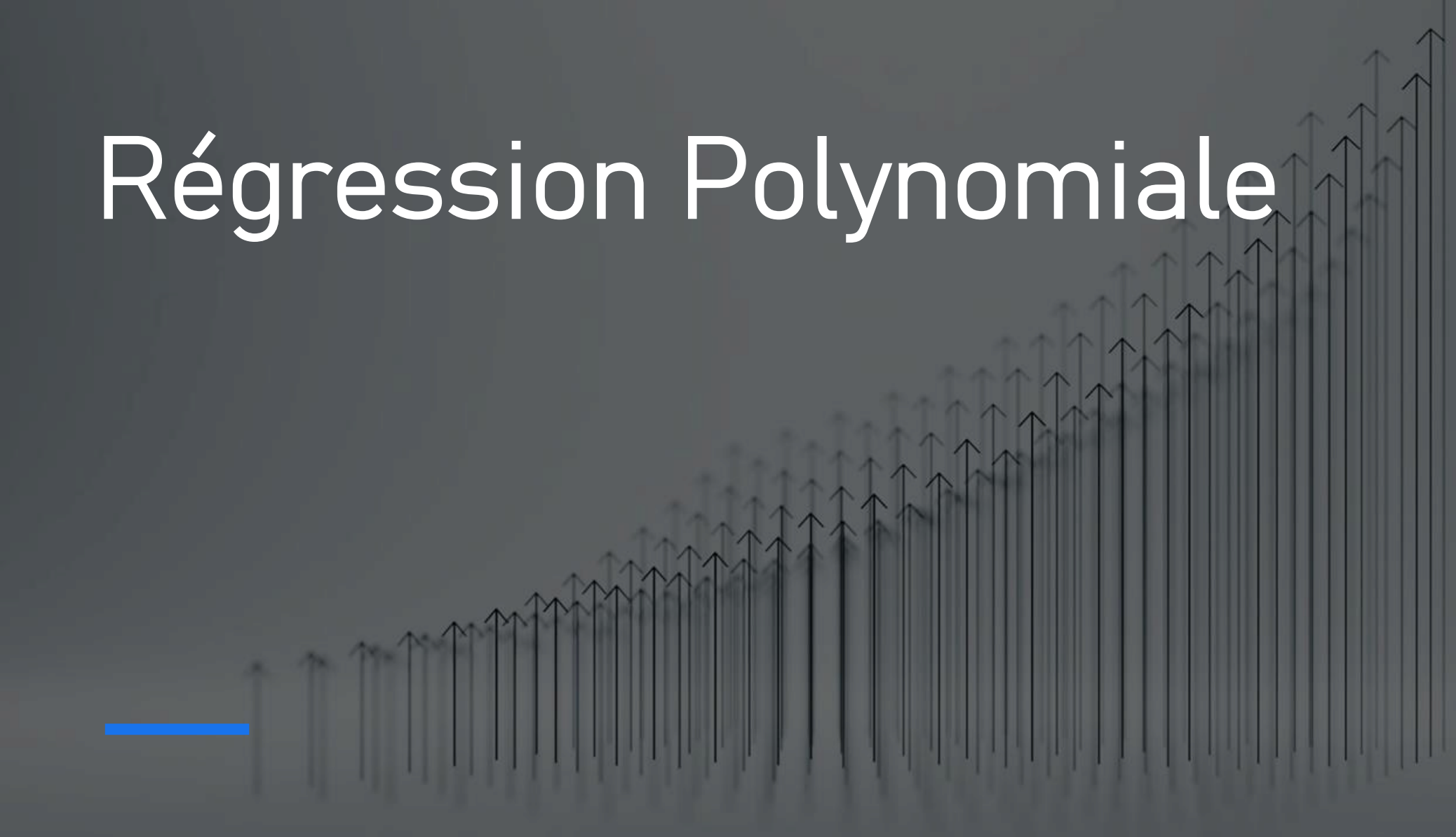
Y=B+A_1X_1+…….A_NX_N → **PLAN**

Régression linéaire Multiple

$$\hat{Y} = b + a_1 X_1 + a_2 X_2 + a_3 X_3$$

$\hat{Y} =$ Salaire de base+ 2000 * Années d'expériences – 1000 * ÂGE + 3000* Niveau d'études

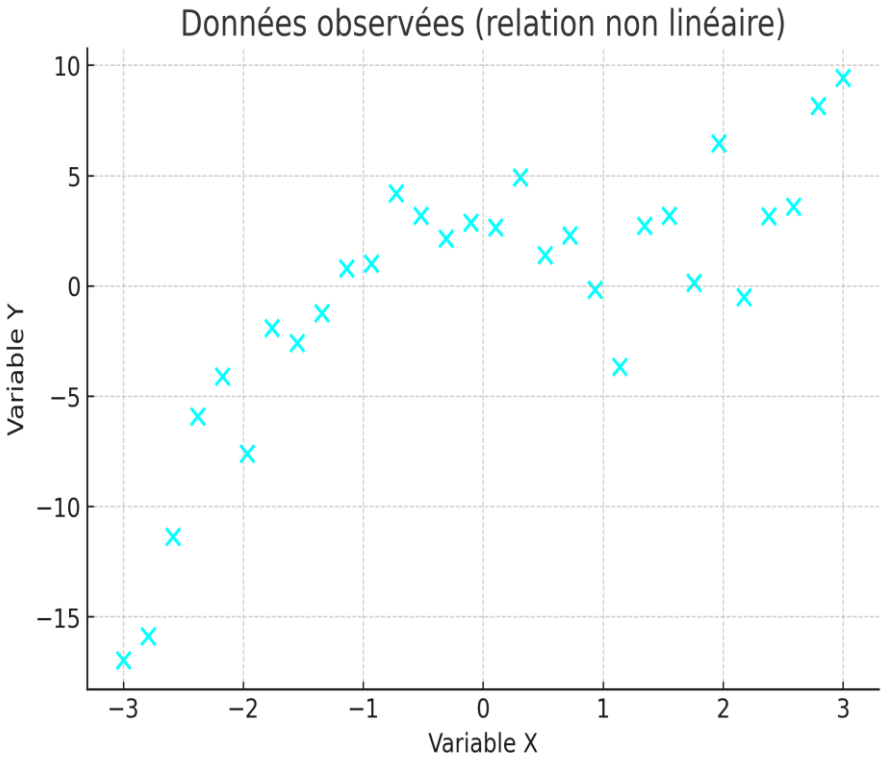
Régression Polynomiale

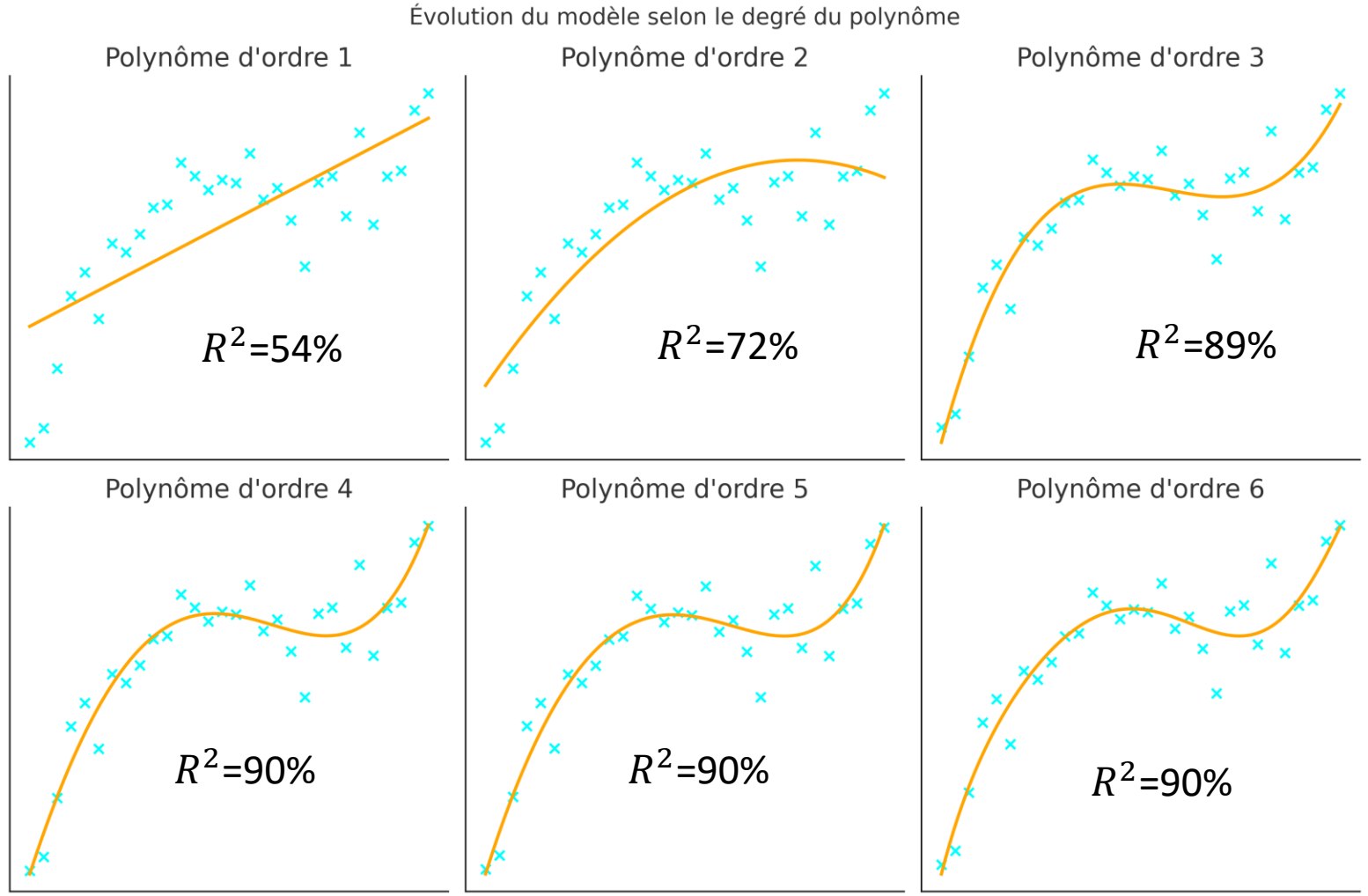


Régression Polynomiale

- *Rappel Equation polynomiale*
- $\hat{Y} = B + a_1 X$ Equation polynomiale degré 1
- $\hat{Y} = B + a_1 X + a_2 X^2$ Equation polynomiale degré 2
- $\hat{Y} = B + a_1 X + a_2 X^2 + \ldots\ldots\ldots + a_n X^n$ Equation polynomiale degré n

Régression Polynomiale:





BIC(Bayesian Information Criterion)

- Le BIC est, comme son nom l'indique, un "critère d'information". Son objectif est de répondre à la question : "Quel est le meilleur modèle pour *décrire* ces données ?"

$$BIC = \mathrm{n}\ln(MSE) + K\ln(n)$$

n: nombre d'observation
K: Degré du polynôme

BIC(Bayesian Information Criterion)

$n \ln(MSE)$:

MSE : C'est l'erreur quadratique moyenne

$ln(MSE)$: •**réduit la dispersion,**
•**rend la comparaison entre modèles plus stable et lisible.**

Evaluation de la qualité du modèle plus stable et proportionnelle.

Exemple:

Modèle	MSE	ln(MSE)
A	10	2.30
B	100	4.61
C	1000	6.91

le MSE passe de 10 → 100 → 1000 (**x100**)

ln(MSE) passe juste de 2.3 → 4.6 → 6.9 (**+4.6 seulement**).

Le **logarithme rend les différences plus faciles à comparer**

BIC(Bayesian Information Criterion)

$\mathrm{n}\ \ln(MSE)$: **pourquoi on multiplie par n**

n est le nombre d'observations , ln(MSE): la qualité du Modèle

Sans le “n”	**Avec le “n”**
On mesurerait l’erreur par point	**On mesure l’erreur totale sur tout le jeu de données**
On comparerait juste les modèles localement	**On tient compte du nombre d’observations (plus le jeu est grand, plus la pénalité est forte)**
Le résultat serait une moyenne	**Le résultat devient global et comparable entre modèles**

BIC(Bayesian Information Criterion)

$\mathrm{n}\ \ln(MSE)$: pourquoi on multiplie par n

Imaginons deux modèles avec :

- **Modèle A** : 10 observations
- **Modèle B** : 100 observations
- Tous deux ont le même MSE = 4

Si on ne multiplie pas par n → BIC = ln(4) = 1.386 pour les deux .

Mais avec le n :

$$BIC_A = 10 \times 1.386 = 13.86$$

$$BIC_B = 100 \times 1.386 = 138.6$$

On voit que le modèle sur **plus de données** a une **pénalisation plus forte**

BIC(Bayesian Information Criterion)

$K \ln(n)$: k le degré du polynôme

Plus on ajoute de paramètres, plus le modèle devient complexe, plus il risque de sur-apprendre.

Le rôle de est donc de pénaliser la complexité.

Si tu ajoutes des paramètres inutiles, le BIC augmente → ton modèle devient moins bon.

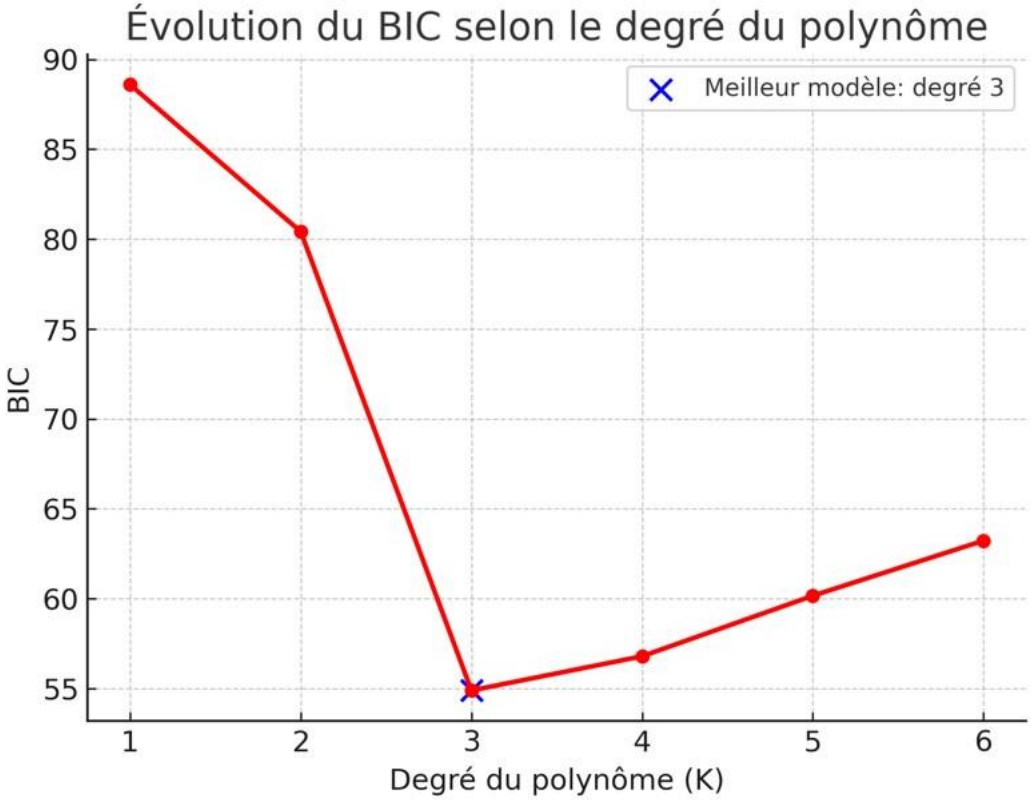
Élément	Rôle
(k)	Degré du polynôme → plus il est grand, plus le modèle est complexe
(ln(n))	Ajuste la pénalité selon la taille des données
(k ln(n))	**Pénalité adaptative** : plus de données → modèle plus puni s'il est trop complexe

BIC(Bayesian Information Criterion)

$$BIC = \mathrm{n}\ln(MSE) + K\ln(n)$$

Objectif	Élément du BIC	Effet
Minimiser l'erreur	(n ln(MSE))	Récompense la qualité d'ajustement
Limiter la complexité	(K ln(n))	Pénalise les modèles trop riches

BIC(Bayesian Information Criterion)

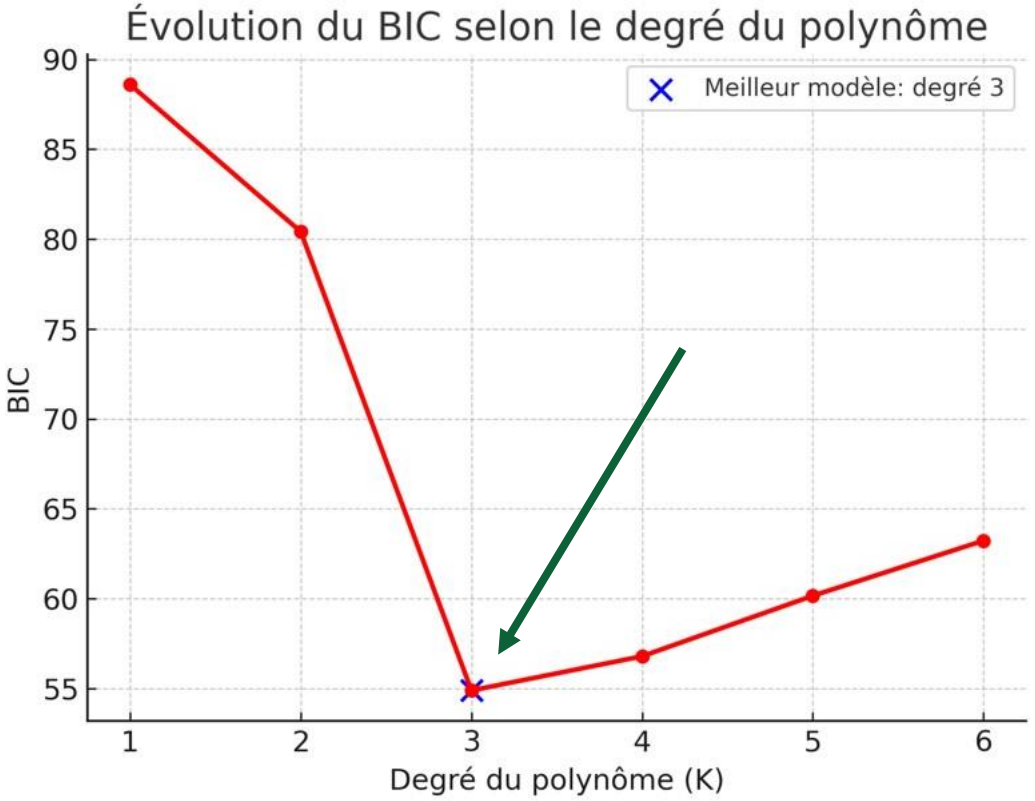


$$BIC = \mathrm{n}\ln(MSE) + K\ln(n)$$

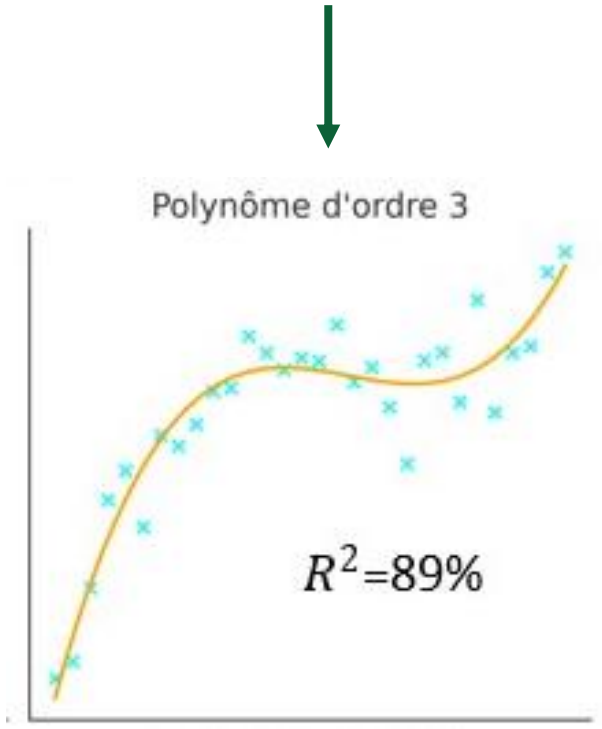
- **Le BIC diminue d'abord car le modèle s'ajuste mieux (le MSE baisse).**
- **Puis il remonte quand le modèle devient trop complexe (la pénalité K ln(n) augmente .**
- **Le minimum du BIC indique le degré optimal : c'est le meilleur compromis entre erreur faible et simplicité du modèle.**

BIC(Bayesian Information Criterion)

$$BIC = \mathrm{n}\ln(MSE) + K\ln(n)$$



Selon la valeur du BIC le modèle avec un degré de polynôme = 3 est le plus optimal.



Processus du M L

Les Données : Définir le Problème
- **Identifier la cible (Y)**
- **Identifier les variables (X)**
- **Séparer les données:**
  - **Un Train Set (80%) pour "cuire" le modèle.**
  - **Un Test Set (20%) pour "goûter" le résultat final.**

Le Modèle :
- **On choisit une "forme" mathématique qui *pourrait* lier X à Y.**
- **Exemple (Régression)**
- **Notre but est de trouver les meilleurs paramètres (a et b).**

La Fonction Coût : Mesurer l'Erreur
- On définit une formule qui "note" à quel point notre modèle se trompe.
- **Exemple (MSE) :** $J(a,b) = \frac{1}{2n}\sum_{i=1}^{n}(y-(ax+b))^2$
- **Objectif :** Trouver les paramètres (a, b) qui rendent ce score **le plus bas possible**

L'Entraînement : L'Optimisation
- **On utilise un algorithme d'optimisation pour trouver le minimum de la fonction coût.**
- **Exemple (Descente de Gradient)**

L'Évaluation : Vérifier la Performance
- **L'entraînement est fini ! On a trouvé nos a et b optimaux.**
- **On utilise maintenant le Test Set (les données mises de côté à l'étape 1) pour voir si le modèle "généralise" bien.**
- **Exemple (Métriques) : On calcule le R^2 ou le MSE sur ces nouvelles données pour obtenir une note finale honnête.**