# Méthodologie en IA

*Bon Courage les gars pour ce cours tarpin longggg, essayer de comprendre et n'hésitez pas à m'envoyer un message si jamais vous avez des questions !!*

## I. Définitions

- ★ **L'intelligence artificielle** : l'ensemble des **théories** et des **techniques** mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine. (Larousse)
- ★ **Automatisation :** l'exécution de tâches techniques par des machines qui fonctionneraient sans l'intervention humaine.
- ★ **« machine learning »** : l'automatisation de l'apprentissage par la machine.

## II. C'est quoi apprendre pour une machine (machine learning) ?

Au cours du temps, il y a différentes définitions qui ont été proposées.

Une 1ère : par **Arthur Samuel** qui explique que le « machine learning c'est le domaine d'études qui permet aux machines d'apprendre sans être explicitement programmé par un humain. »

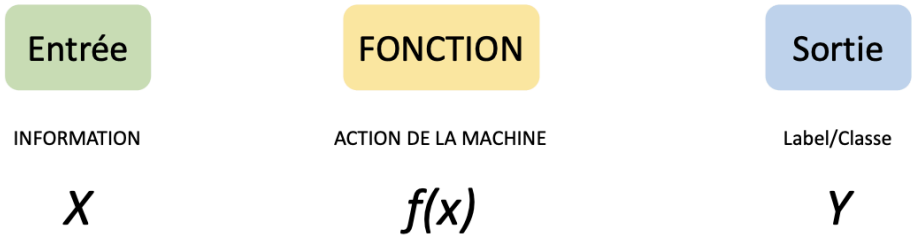
Ensuite une 2ème : par **Tom Mitchell** qui est plus technique et qui explique que si on considère une tache T, dont la performance est évaluée par P. Alors une machine apprend, quant au fur à mesure d'une expérience E, ses performances pour la tache T augmentent.


Donc c'est une des choses qui va être **importante** en machine learning.
Dans le cours du Dr. Imbert, vous avez vu qu'en intelligence artificielle, on va parler d'intelligence artificielle **model driven**, ce qui va au final plus se rapprocher de **l'automatisation**, où l'homme connaît le modèle qu'il veut appliquer, et d'intelligence artificielle **data driven**, dont le **machine learning** va plus se rapprocher, car c'est bien **l'expérience** et donc les **data** qui permettent **d'améliorer la tâche T.**

Cependant ce n'est pas quelque chose qui est restreint, on peut en fait combiner un peu des deux, on peut **démarrer** une tâche sur un **modèle driven** et faire en sorte que les **data améliorent** cette tâche sur une idée du **datatrium**.

*Rappel : Un système **model-driven** (piloté par le modèle) repose sur des **lois, équations ou connaissances théoriques** déjà connues pour décrire le fonctionnement d'un phénomène : on part du savoir humain (par exemple les lois de la physique) pour faire des prédictions. À l'inverse, un système **data-driven** (piloté par les données) apprend **directement à partir des données**, sans modèle théorique préalable.*

Une autre façon de représenter ceci et d'expliquer l'objectif qu'on a en machine learning, c'est de parler d'entrée, de fonction ou de tâche et de sortie :




Objectif : trouver *f(x)* telle que *f(X) = Y*

- En **entrée**, on va avoir de **l'information** ou de l'expérience → **X**.
- La **tâche**, ça va être **l'ac0on** de la machine, *en général,* c'est une **fonc0on** → **f(x)**
- Et la **sortie**, ça va être par exemple un **label** ou une **classe** → **Y**

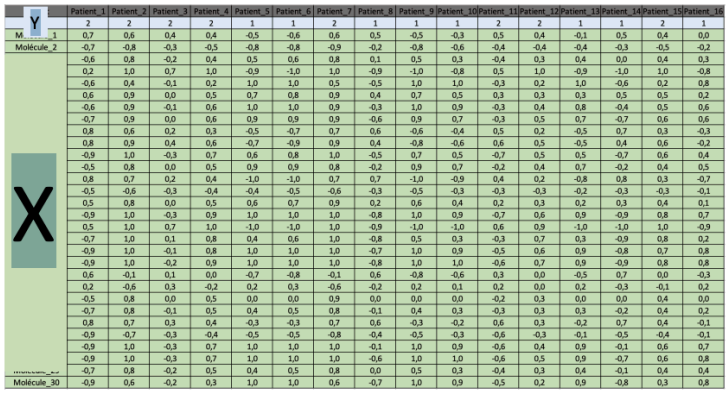
Notre **objectif** : **créer un modèle ou une fonction f qui permette d'obtenir Y quand on l'applique à X → trouver f(x) tel que f(x) = y**

*Et ça c'est une notion importante de comprendre déjà si on vous donne des données qu'est-ce qui correspond à une entrée X qu'est-ce qui correspond à une sortie Y un objectif et ensuite comprendre que le but c'est de créer une fonction f qui au final sera plus ou moins complexe ensemble on va voir des fonctions simples mais ça peut devenir beaucoup plus compliqué, mais on ne rentrera pas trop dans les détails par rapport à ça.*

Exemple : un tableau de données qui représente des quantités de molécules chez différents patients qui présente des caractéristiques différentes.
1ère ligne = numéro de patient (osf) on ne cherche pas à le prédire
2ème ligne = label soit égale à 2 soit égale à 1 → traduit une condition pour chaque patient
Exemple ici : label 1 → tumeur bénigne / label 2 → tumeur maligne

Reste des lignes différentes molécules et pour chaque mol on a des valeurs qui correspondent à la quantité de molécules pour chaque parent.

| Patient | Patient_1 | Patient_2 | Patient_3 | Patient_4 | Patient_5 | Patient_6 | Patient_7 | Patient_8 | Patient_9 | Patient_10 | Patient_11 | Patient_12 | Patient_13 | Patient_14 | Patient_15 | Patient_16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Label | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 |
| Molécule_1 | 0,7 | 0,6 | 0,4 | 0,4 | -0,5 | -0,6 | 0,6 | 0,5 | -0,5 | -0,3 | 0,5 | 0,4 | -0,1 | 0,5 | 0,4 | 0,0 |
| Molécule_2 | -0,7 | -0,8 | -0,3 | -0,5 | -0,8 | -0,8 | -0,9 | -0,2 | -0,8 | -0,6 | -0,4 | -0,4 | -0,4 | -0,3 | -0,5 | -0,2 |
| Molécule_3 | -0,6 | 0,8 | -0,2 | 0,4 | 0,5 | 0,6 | 0,8 | 0,1 | 0,5 | 0,3 | -0,4 | 0,3 | 0,4 | 0,0 | 0,4 | 0,3 |
| Molécule_4 | 0,2 | 1,0 | 0,7 | 1,0 | -0,9 | -1,0 | 1,0 | -0,9 | -1,0 | -0,8 | 0,5 | 1,0 | -0,9 | -1,0 | 1,0 | -0,8 |
| Molécule_5 | -0,6 | 0,4 | -0,1 | 0,2 | 1,0 | 1,0 | 0,5 | -0,5 | 1,0 | 1,0 | -0,3 | 0,2 | 1,0 | -0,6 | 0,2 | 0,8 |
| Molécule_6 | 0,6 | 0,9 | 0,0 | 0,5 | 0,7 | 0,8 | 0,9 | 0,4 | 0,7 | 0,5 | 0,3 | 0,3 | 0,3 | 0,5 | 0,5 | 0,2 |
| Molécule_7 | -0,6 | 0,9 | -0,1 | 0,6 | 1,0 | 1,0 | 0,9 | -0,3 | 1,0 | 0,9 | -0,3 | 0,4 | 0,8 | -0,4 | 0,5 | 0,6 |
| Molécule_8 | -0,7 | 0,9 | 0,0 | 0,6 | 0,9 | 0,9 | 0,9 | -0,6 | 0,9 | 0,7 | -0,3 | 0,5 | 0,7 | -0,7 | 0,6 | 0,6 |
| Molécule_9 | 0,8 | 0,6 | 0,2 | 0,3 | -0,5 | -0,7 | 0,7 | 0,6 | -0,6 | -0,4 | 0,5 | 0,2 | -0,5 | 0,7 | 0,3 | -0,3 |
| Molécule_10 | 0,8 | 0,9 | 0,4 | 0,6 | -0,7 | -0,9 | 0,9 | 0,4 | -0,8 | -0,6 | 0,6 | 0,5 | -0,5 | 0,4 | 0,6 | -0,2 |
| Molécule_11 | -0,9 | 1,0 | -0,3 | 0,7 | 0,6 | 0,8 | 1,0 | -0,5 | 0,7 | 0,5 | -0,7 | 0,5 | 0,5 | -0,7 | 0,6 | 0,4 |
| Molécule_12 | -0,5 | 0,8 | 0,0 | 0,5 | 0,9 | 0,9 | 0,8 | -0,2 | 0,9 | 0,7 | -0,2 | 0,4 | 0,7 | -0,2 | 0,4 | 0,5 |
| Molécule_13 | 0,8 | 0,7 | 0,2 | 0,4 | -1,0 | -1,0 | 0,7 | 0,7 | -1,0 | -0,9 | 0,4 | 0,2 | -0,8 | 0,8 | 0,3 | -0,7 |
| Molécule_14 | -0,5 | -0,6 | -0,3 | -0,4 | -0,4 | -0,5 | -0,6 | -0,3 | -0,5 | -0,3 | -0,3 | -0,3 | -0,2 | -0,3 | -0,3 | -0,1 |
| Molécule_15 | 0,5 | 0,8 | 0,0 | 0,5 | 0,6 | 0,7 | 0,9 | 0,2 | 0,6 | 0,4 | 0,2 | 0,3 | 0,2 | 0,3 | 0,4 | 0,1 |
| Molécule_16 | -0,9 | 1,0 | -0,3 | 0,9 | 1,0 | 1,0 | 1,0 | -0,8 | 1,0 | 0,9 | -0,7 | 0,6 | 0,9 | -0,9 | 0,8 | 0,7 |
| Molécule_17 | 0,5 | 1,0 | 0,7 | 1,0 | -1,0 | -1,0 | 1,0 | -0,9 | -1,0 | -1,0 | 0,6 | 0,9 | -1,0 | -1,0 | 1,0 | -0,9 |
| Molécule_18 | -0,7 | 1,0 | 0,1 | 0,8 | 0,4 | 0,6 | 1,0 | -0,8 | 0,5 | 0,3 | -0,3 | 0,7 | 0,3 | -0,9 | 0,8 | 0,2 |
| Molécule_19 | -0,9 | 1,0 | -0,1 | 0,8 | 1,0 | 1,0 | 1,0 | -0,7 | 1,0 | 0,9 | -0,5 | 0,6 | 0,9 | -0,8 | 0,7 | 0,8 |
| Molécule_20 | -0,9 | 1,0 | -0,2 | 0,9 | 1,0 | 1,0 | 1,0 | -0,8 | 1,0 | 1,0 | -0,6 | 0,7 | 0,9 | -0,9 | 0,8 | 0,8 |
| Molécule_21 | 0,6 | -0,1 | 0,1 | 0,0 | -0,7 | -0,8 | -0,1 | 0,6 | -0,8 | -0,6 | 0,3 | 0,0 | -0,5 | 0,7 | 0,0 | -0,3 |
| Molécule_22 | 0,2 | -0,6 | 0,3 | -0,2 | 0,2 | 0,3 | -0,6 | -0,2 | 0,2 | 0,1 | 0,2 | 0,0 | 0,2 | -0,3 | -0,1 | 0,2 |
| Molécule_23 | -0,5 | 0,8 | 0,0 | 0,5 | 0,0 | 0,0 | 0,9 | 0,0 | 0,0 | 0,0 | -0,2 | 0,3 | 0,0 | 0,0 | 0,4 | 0,0 |
| Molécule_24 | -0,7 | 0,8 | -0,1 | 0,5 | 0,4 | 0,5 | 0,8 | -0,1 | 0,4 | 0,3 | -0,3 | 0,3 | 0,3 | -0,2 | 0,4 | 0,2 |
| Molécule_25 | 0,8 | 0,7 | 0,3 | 0,4 | -0,3 | -0,3 | 0,7 | 0,6 | -0,3 | -0,2 | 0,6 | 0,3 | -0,2 | 0,7 | 0,4 | -0,1 |
| Molécule_26 | -0,9 | -0,7 | -0,3 | -0,4 | -0,5 | -0,5 | -0,8 | -0,4 | -0,5 | -0,3 | -0,6 | -0,3 | -0,1 | -0,5 | -0,4 | -0,1 |
| Molécule_27 | -0,9 | 1,0 | -0,3 | 0,7 | 1,0 | 1,0 | 1,0 | -0,1 | 1,0 | 0,9 | -0,6 | 0,4 | 0,9 | -0,1 | 0,6 | 0,7 |
| Molécule_28 | -0,9 | 1,0 | -0,3 | 0,7 | 1,0 | 1,0 | 1,0 | -0,6 | 1,0 | 1,0 | -0,6 | 0,5 | 0,9 | -0,7 | 0,6 | 0,8 |
| Molécule_29 | -0,7 | 0,8 | -0,2 | 0,5 | 0,4 | 0,5 | 0,8 | 0,0 | 0,5 | 0,3 | -0,4 | 0,3 | 0,4 | -0,1 | 0,4 | 0,4 |
| Molécule_30 | -0,9 | 0,6 | -0,2 | 0,3 | 1,0 | 1,0 | 0,6 | -0,7 | 1,0 | 0,9 | -0,5 | 0,2 | 0,9 | -0,8 | 0,3 | 0,8 |




Dans cet exemple si on veut entrainer une machine à déterminer si un patient présente une tumeur maligne ou bénigne en fonction des quantités de chaque molécule, on va désigner par X toutes les quantités de molécules et donc Y sera notre label
1 = bénigne ou 2 = maligne → on a bien une entrée X et un objectif Y et ce qu'on veut créer *(absent dans notre tableau)* → fonction qui permet d'obtenir Y à partir de X.

En fonction du label (Y) on a 2 grands types de problèmes.

★ Si Y appartient à une valeur **quantitative** (Y appartient au réel **R** ) → logik mais bon : la fonction qu'on recherche pourra donner comme résultat **n'importe quel nombre réel**

→ Ex : prédire le nombre de patients atteints du COVID-19 après 1 mois de pandémie.
Ex : prédire le prix de vente d'un bien immobilier.

★ Si Y est une valeur **qualitative** : c'est donc une catégorie qui pourra être transformée en 1,2... entrer → sortie peut prendre un nombre **limité** de valeurs.

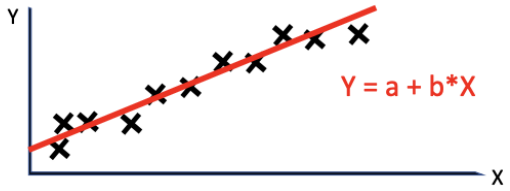
→ Ex : prédire si un parent malade aura une forme grave ou non de maladie.
*(On a bien deux catégories qu'on sépare en sévère ou non. On peut si on veut rajouter une troisième catégorie en disant ça c'est la forme asymptomatique. Dans TOUS les cas, il n'y aura que deux ou trois sorties possibles.)*

**Régréssion** *:* On veut prédire y à partir d'une seule variable x

→ Ex : prédire le prix d'une maison en fonction de la surface en $m^2$

Les prix (X) en fonction de la surface (Y) et inversement si on veut définir le prix à partir de la surface au $m^2$ il nous faut une fonction f qui relie surface au prix.

- Régression : Prédire une valeur quantitative ($Y \in \mathbb{R}$)



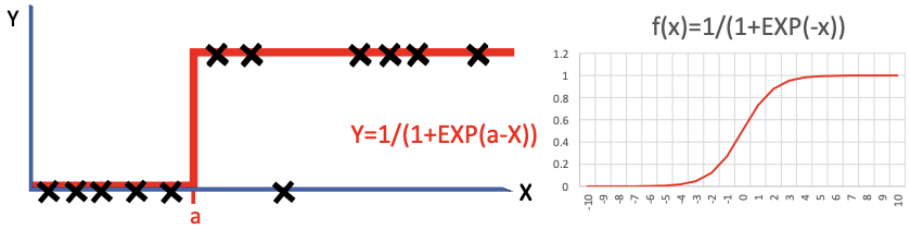

Y= a+b*X régression linéaire

**Classification** : on peut prédire le risque de survenue d'un cancer en fonction de la durée d'exposition aux radiations. Donc là la sortie Y : il y a deux types de possibilités.

Le parent a eu un cancer = 1. / Le parent n'a pas eu de cancer = 0.

*( fait un aparté sur Tchernobyl et dit qu'on dira qu'il y a un seuil à partir duquel les parents ont eu un cancer. on veut savoir maintenant, on a un patient, on sait combien de temps il a été exposé et on veut savoir si oui ou non il a un cancer, on va utiliser une fonction donc la fonction de l'exemple c'est une fonction en S : sigmoïde c'est une fonction logistique, donc elle sera bornée, elle ne pourra que donner des valeurs entre 0 et 1 . et donc on pourra décider que toutes les valeurs en dessous de 0,5 seront passés à zéro et au-dessus 0,5 à un .)*

- Classification : Prédire une valeur qualitative ($Y \in \mathbb{N}$)



| | |
|---|---|
| **Apprentissage supervisé**<br><br>*(Le plus fréquent en médecine)* | On **connait** les **labels** / les valeurs qu'on veut prédire avec notre modèle. = quand on connaît les objectifs qu'on veut atteindre.<br><br>Exemples :<br><br>- Dépistage de cancer : Cancer vs Pas de cancer<br>- Prédiction du prix de l'immobilier,<br>- Triage de mails : SPAM ou non SPAM. |
| **Apprentissage non supervisé** | On veut trouver des tendances / des groupes, mais on ne sait pas à l'avance lesquels Dans la majeure partie des cas ce sera des problèmes de classifications non supervisé<br><br>*Exemples :*<br><br>- Regroupement de sites web similaires pour faire des suggestions,<br>- Recherche de sous-groupes de tumeurs. |
| **Apprentissage semi-supervisé** | Qu'on ne va pas développer dans ce cours qui concerne des bases de données, pour lesquelles seul une partie des labels à prédire sont connus. |

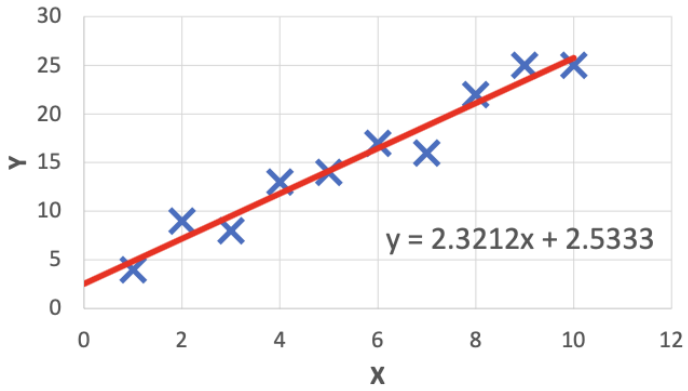
## III. Comment apprendre ?

En principe, on rappelle que le but = apprendre à effectuer une tâche et que pour savoir si on effectue correctement cette tâche on va **mesurer** une performance et ce qu'il faut c'est qu'au fur et à mesure qu'on apprenne on devienne de plus en plus performant.

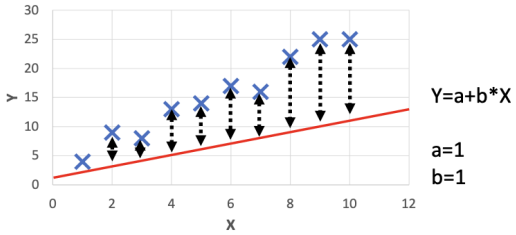
C'est une notion **très générale** pas spécifique aux machines, même pour nous au fur et à mesure qu'on acquiert de l'expérience on devient de plus en plus performant pour certaines tâches.

**Comment est-ce qu'on sait qu'on devient plus performant ?**
Au fur et à mesure qu'on apprend on se rapproche d'un objectif final ou autrement dit on fait de moins en moins d'erreur et si pour nous c'est quelque chose d'assez intuitif.
En machine learning il va falloir programmer la machine pour qu'elle minimise l'erreur au fur et à mesure de l'expérience et il va falloir définir ce qu'est l'erreur.
Pour définir cette erreur on va utiliser ce qu'on appelle la fonction de coût ou fonction de perte qui va représenter globalement la différence entre ce qu'on a obtenu et ce que l'on souhaite obtenir comme résultat.
On va voir ensemble qu'il existe des algorithmes qui permettent de minimiser peu à peu cette erreur.

Pour illustrer cette fonction de coût on va reprendre notre régression linéaire de l'exemple précédent et on va voir comment on arrive à cette droite qui représente donc la régression linéaire avec cette fonction Y = a + b*X.
Donc on va repartir des données avec Y en fonction de X et on sait que la forme de la droite qu'on a envie de fabriquer est de type Y = a + b*X et l'objectif ça va être de trouver a et b.
Au départ on connaît ni a ni b donc on va faire un premier test par exemple en prenant a=1et b=1 ça va nous donner cette courbe :



Ce qui on le voit n'est pas la courbe qu'on attend une fois qu'on a cette courbe il va falloir mesurer l'erreur qu'on a engendré en utilisant cette courbe donc pour ce faire on va utiliser la formule ci-dessous :




$$\text{Erreur: } J(a,b)=\frac{1}{2m}\sum_{i=1}^{m}(h(x_i)-Y)^2$$

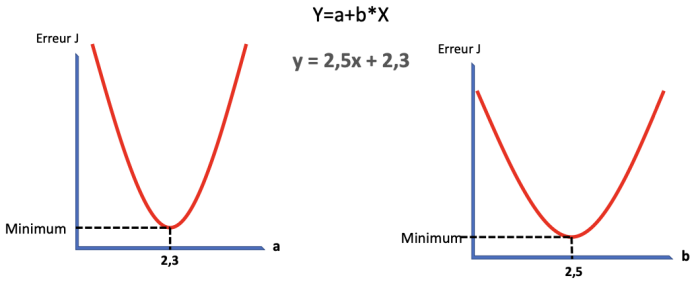
Y predit Y objectif

La formule que vous n'avez pas besoin de connaître par cœur et pour laquelle il faut comprendre que ça représente la moyenne de la différence absolue entre le résultat de notre courbe et le résultat qu'on attend .
Alors si on s'intéresse à la formule en réalité *(mais mdr elle était pas censé être inintéressante pour nous)* c'est pas strictement une moyenne mais la moyenne des carrés des différences pour chaque point divisé par deux mais on utilise cette formule uniquement pour faciliter les calculs et la programmation qu'il y a derrière et vous n'avez pas besoin de la connaître par cœur.

Ce qu'il faut comprendre par contre c'est que si la courbe s'éloigne du résultat qu'on attend, par exemple ici, si on diminue a en le passant à 0 cette erreur elle va augmenter alors que si la courbe se rapproche du résultat attendu par exemple en passant a à 2 l'erreur va diminuer ce qui est donc notre objectif et quand on va écrire notre algorithme on va pouvoir faire en sorte que l'erreur continue à diminuer jusqu'à ce qu'elle ne puisse plus diminuer jusqu'à ce qu'on arrive à un minimum.

*Tut'l'explique : si tu augmentes ton erreur tu t'éloignes de la courbe et si tu diminues l'erreur tu t'en rapproche notre objectif c'est de diminuer un max l'erreur pour arriver à un minimum d'erreur.*

Ainsi petit à petit on va pouvoir trouver les meilleures paramètres et à partir de là la fonction de coût ne pourra plus diminuer elle sera au minimum possible et on va pouvoir tracer l'erreur associé au paramètre a en fonction du paramètre a et l'erreur associé au paramètre b en fonction du paramètre b et pour arriver à une erreur minimale pour a et b on va utiliser un **algorithme de base** en **machine learning** qui s'appelle la **descente de gradients**

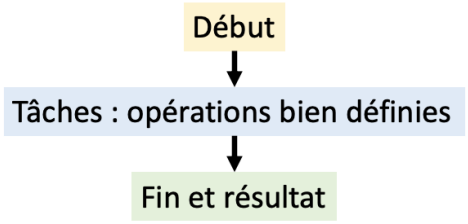


## IV. Qu'est-ce que la descente de gradients *(spoiler alert : c'est pas la mienne même si ce cours me pousse à bout)*

Def : La descente de gradients c'est un algorithme

## V. Qu'est-ce que c'est un algorithme ?

**Un algorithme** = un **ensemble de règles opératoires** dont l'application permet de **résoudre un problème** énoncé au moyen d'un nombre fini d'opérations.
C'est quelque chose qui a un début qui consiste en l'ensemble d'opérations bien définies et qui aboutit à un résultat donc au final ça peut s'appliquer à beaucoup de situations.




*Par exemple* : Si on cherche **un nom dans une liste mal classée**, on va sûrement appliquer **un algorithme** qui sera très simple.
On va **lire le premier nom**, ce sera **la première tâche élémentaire**. Ensuite, on va suivre **une règle** :

- si c'est **le nom qu'on cherche**, **on s'arrête**,
- sinon **on passe à la ligne suivante**, et **on continue** comme ça jusqu'à ce qu'on s'arrête.

On lit **le premier nom** → ce n'est pas le bon.
On lit **le deuxième nom** → ce n'est pas bon.
On lit **le troisième nom** → **c'est celui qu'on cherche**, **on s'arrête**.

En **médecine**, on va aussi suivre des **algorithmes**, notamment des choses qu'on appelle **des algorithmes décisionnels** ou **arbres décisionnels**, pour lesquels on sera dans la même situation :
On va **partir d'une situation initiale** et **suivre un ensemble de règles** jusqu'à **aboutir à un résultat**.

*Exemple* : une décision thérapeutique ou un diagnostic
La **descente de gradient** = **algorithme** donc il doit aussi être composé d'opérations bien définies.
Mais quelles sont ces opérations ?
En réalité ça peut se résumer à ce qu'on a ici si on prend une **fonction avec plusieurs paramètres**, par exemple la fonction simple de tout à l'heure : **Y = a + b*X**
La descente de gradient va **démarrer en prenant des paramètres arbitraires**
(ex : a = 0 et b = 0) puis elle va les mettre à jour suivant la formule et ceci jusqu'à ce qu'on atteigne la convergence ce qui veut dire que les paramètres ne bougeront plus au fur et à mesure qu'on les met à jour.

$$\theta := \theta - \alpha * \frac{\partial}{\partial \theta} J(\theta_0, \theta_1 ...)$$

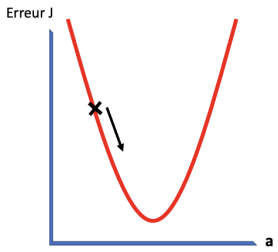
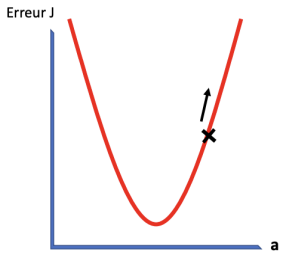
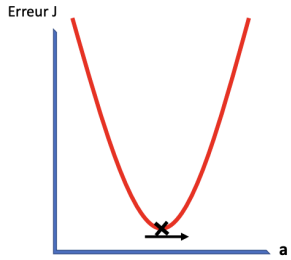
On va s'intéresser à cette formule vous n'avez pas besoin de la connaître par cœur *(mdr c'est la deuxième à pas connaitre par cœur il s'agirait de doser).*
Par contre on va essayer de comprendre ce que ça veut dire donc déjà ce symbole:

- deux points égale (: =) met à jour le paramètre au fur et à mesure donc le nouveau paramètre à gauche va devenir ce qu'on a à droite.
- Les paramètres thêta (θ) qui sont une notation générale pour les paramètres d'une fonction on utilise en général le signe thêta.
- Sur la droite le $\theta_0$ et $\theta_1$ etc correspond à différents paramètres de la fonction. Dans notre exemple : on pourrait remplacer si on travaille sur le parametre "a", on pourrait écrire la même chose en remplaçant **θ** par **a**, et à droite on s'intéresse à la **fonction de cout** qui prend en compte à la fois **a et b**

$$a := a - \alpha * \frac{\partial}{\partial a} J(a, b)$$

- Le paramètre alpha (α) correspond au learning rate ou taux d'apprentissage (on le revoie après)
- A droite on a la dérivé de **la fonction de coût** en fonction du **paramètre qu'on est en train de mettre à jour.**

*Rappel : la dérivée d'une fonction c'est la pente de la courbe à un endroit donné.*

| | |
|---|---|
|  $\frac{\partial}{\partial\theta}J(\theta_0, \theta_1...)$ Valeur < 0 | Si on se place sur la première partie de notre fonction de perte ici : si “a” augmente la valeur de “j” diminue et donc la courbe descend donc notre dérivé va nous donner une **valeur négative.** |
|  $\frac{\partial}{\partial\theta}J(\theta_0, \theta_1...)$ Valeur > 0 | Par contre si on se place plus loin sur la courbe alors quand “a” augmente “j” augmente également donc la pente est croissante donc notre dérivé va nous donner une **valeur positive.** |
|  $\frac{\partial}{\partial\theta}J(\theta_0, \theta_1...)$ Valeur = 0 | Enfin, si on est au plus bas de notre courbe à l'endroit où elle passe d'une pente descendante à une pente croissante, on va être sur un point où la **dérivée est égale à 0** et ce sera notre minimum local. |

Donc si on reprend notre formule on voit qu'avant notre dérivé on a un signe moins et donc on comprend bien que si augmenter la valeur de notre paramètre diminue l'erreur notre dérivé va être négative et donc appliquer notre formule reviendra à augmenter un petit peu la valeur de notre paramètre :

$$\theta := \theta - \alpha * \frac{\partial}{\partial \theta} J(\theta_0, \theta_1 ...)$$

Si au contraire augmenter le paramètre augmente l'erreur alors dérivé va être positive et donc appliquer la formule va entraîner de diminuer la valeur du paramètre.
Enfin si on est arrivé sur un minimum local notre dérivé va être nul et donc notre paramètre ne changera pas de valeur et une fois que ce sera le cas pour tous nos paramètres les opérations sont terminées et l'algorithme nous aura donné les paramètres qui permettent d'avoir une erreur minimale.

*Si je résume tout :*

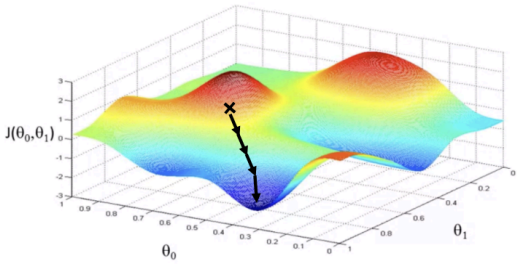
*Dans la **descente de gradient**, le **signe de la dérivée** indique **dans quelle direction** il faut modifier les paramètres pour **réduire l'erreur** :*

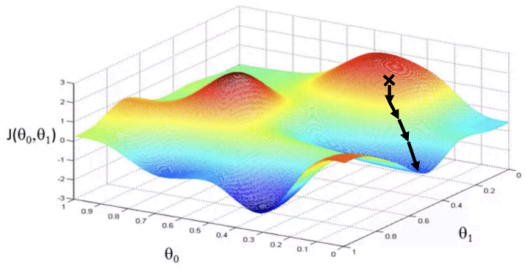
- *Si la **dérivée est négative**, ça veut dire que **augmenter le paramètre diminue l'erreur**, donc **l'algorithme augmente légèrement sa valeur**.*
- *Si la **dérivée est positive**, ça veut dire que **augmenter le paramètre augmente l'erreur**, donc **l'algorithme diminue sa valeur**.*
- *Et quand la **dérivée devient nulle**, cela signifie qu'on est **arrivé à un minimum local** : les paramètres **ne changent plus**, l'erreur est **minimale**, et **l'algorithme s'arrête**.*

*Si vous avez compris ça vous avez compris le principe général de l'algorithme de descente de gradients et ce qu'il faut bien comprendre c'est que cet algorithme on peut l'utiliser pour un grand nombre de fonctions et pas seulement pour la régression linéaire ça va même au-delà du machine learning parce que ça peut s'appliquer à d'autres types de problèmes. Notamment quand on fait de la reconstruction d'image.*

Ce qu'il faut comprendre aussi c'est que si dans notre exemple de régression linéaire la fonction de coût présentait un minimum unique pour des valeurs uniques de a et b et donc prenez la forme qu'on a ici.Dans d'autres situations **il peut exister plusieurs minimums locaux** possibles et donc en fonction des paramètres qu'on va utiliser au départ le résultat ne sera pas forcément le même *(regardez les images de la pages suivantes pour mieux comprendre)*

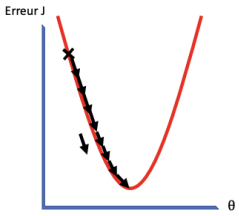
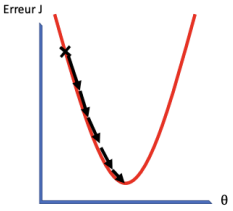
En effet si on démarre notre algorithme à un endroit il va nous mener à ce minimum ici alors que si on démarre un autre endroit notre algorithme va nous mener à un autre minimum local et donc il n'existe pas toujours une solution unique à nos problèmes.

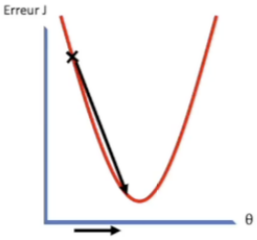



## VI. Quelques subtilités

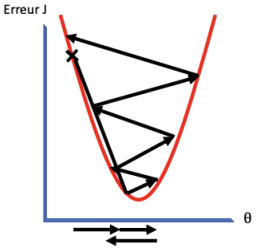
D'abord, revenons sur le **paramètre alpha (α)** de la formule.
C'est ce qu'on appelle le **taux d'apprentissage** : c'est un facteur qui **multiplie la dérivée** dans la formule.

| | |
|---|---|
|  • $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 ...)$<br>$\alpha$ trop petit : petites variations de θ<br>Beaucoup de calculs pour arriver au mini | Si **α est petit**, les mises à jour de **θ** seront **faibles** : le paramètre bouge très peu à chaque étape.<br>Du coup, l'algorithme **avance lentement** vers le minimum et il faut **beaucoup de mise à jour et donc beaucoup de calcule** pour y arriver. |
|  • $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 ...)$<br>• $\alpha$ = valeur fixe.<br>• $\frac{\partial}{\partial\theta} J$ diminue au fur et à mesure que l'on s'approche du minimum | Avec un **α fixe**, la vitesse et la stabilité de la convergence dépendent entièrement de la **valeur choisie** de α, il est **plus stable** (moins de risque de "sauter" le minimum).<br><br>Si α est fixe, il est possible de **converger efficacement** vers le minimum sans modifier α au cours de l'algorithme. En effet, à mesure que l'on se rapproche du minimum, la **pente de la fonction de coût** (la dérivée de J) diminue progressivement. Comme les mises à jour de θ dépendent à la fois de α et de cette pente, les modifications apportées à θ deviennent naturellement **de plus en plus petites** au fur et à mesure que l'on approche du minimum, même avec un α constant. |




• $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 ...)$

$\alpha$ trop grand: grandes variations de




• $\theta := \theta - \alpha * \frac{\partial}{\partial\theta} J(\theta_0, \theta_1 ...)$

$\alpha$ trop grand: grandes variations de

Risque : pas de convergence voir mê divergence

Maintenant, si on choisit un **alpha trop grand**, les mises à jour deviennent **très importantes**.
À chaque étape, **θ** change beaucoup, donc on **risque de dépasser le minimum** au lieu de s'en approcher progressivement.

Par exemple, si on est presque au minimum et que la pente est encore descendante, une mise à jour trop forte peut nous faire **sauter de l'autre côté de la courbe**, là où l'erreur recommence à augmenter.
On va alors corriger dans l'autre sens… mais trop fort encore, ce qui fait qu'on **oscille autour du minimum** sans jamais se stabiliser.

Et donc le risque d'avoir cet alpha trop élevé c'est que notre algorithme ne converge pas voire pire il peut diverger et s'éloigner de notre objectif.

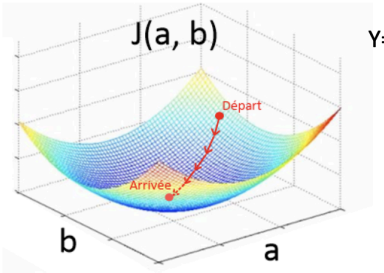
## VII. Le scaling des données ou la mise à l'échelle

C'est quelque chose qui concerne les **données d'entrée** qu'on va utiliser donc **x**. Je rappelle que l'on peut utiliser la **descente de gradient** pour minimiser la **fonction de perte** associée à tout un tas de **fonctions** qui peuvent être plus ou moins **complexes**.

Donc maintenant, on va se mettre dans la situation où l'on essaye de minimiser la **fonction de coût** associée à la fonction :

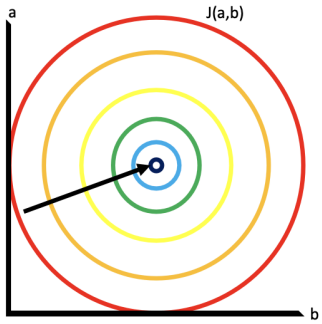
**y = k (constante) + a * $x_1$** (**a** étant le **paramètre**, $x_1$ étant notre première **donnée**) + **b * $x_2$** (**b** étant un deuxième **paramètre** et $x_2$ une deuxième **donnée**).

Si $x_1$ et $x_2$ ont la **même dimension** (même échelle ou de l'unité des données), par exemple s'il s'agit pour les deux cas de dosage biologique pour lesquels les valeurs sont comprises entre 10 et 100 millimoles par litre, alors la **fonction de perte** associée aux **paramètres a** et **b** va prendre l'aspect qu'on avait tout à l'heure (comme dans le **graphique** en bas) et la **descente de gradient** va fonctionner correctement.




$Y = k + a \cdot x_1 + b \cdot x_2$

Si on représente cette fonction de perte sur **une vue 2d** donc ici vue du haut pour le graphique précédent ça va nous donner cette forme :

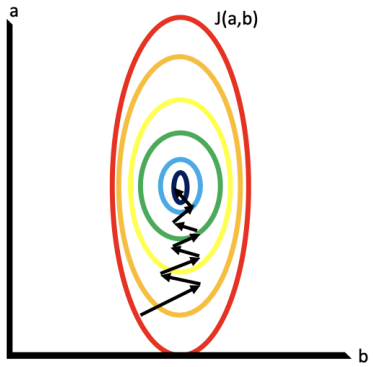



$$Y = k + a \cdot x_1 + b \cdot x_2$$

$x_1$ et $x_2$ ont des dimensions similaires :
Le minimum est trouvé rapidement

Dans ce cas-là, l'algorithme va rapidement trouver le minimum local. *On peut imaginer que c'est comme si on laissait tomber une bille depuis le rebord d'un saladier, celle-ci irait alors rapidement vers le point le plus bas du saladier.*

Maintenant si nos données $x_1$ et $x_2$ présentent des **dimensions très différentes** par exemple si $x_2$ est beaucoup plus grand que $x_1$ , on a :



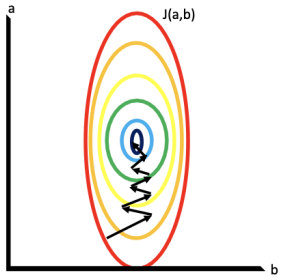

$$Y = k + a \cdot x_1 + b \cdot x_2$$

Ex. $x_1$ : glycémie à jeun (0,8 – 1,8)
$x_2$ : âge (30 – 85)

$x_2$ a une dimension beaucoup plus grande que $x_1$ :
Le minimum est beaucoup plus long à trouver

Par exemple, faire un modèle pour prédire un événement cardiovasculaire chez des parents en fonction de leur glycémie a jeun et de leur âge alors nos **données d'âge vont avoir une dimension beaucoup plus importante que nos données glycémie à jeun**. Car nos données la glycémie à jeun varie entre 0,8 et 1,8 gramme par litre, et nos parents ont entre 30 et 85 ans.

Si on **représente la fonction de coût** associée aux **paramètres A et B** en fonction de ces paramètres, on va voir que celle-ci va prendre une **forme complètement différente** comme représenté.
On voit que dans la **zone verticale centrale** de la figure la **pente menant au minimum** sera beaucoup plus faible et donc si l'algorithme se retrouve dans cette zone à un moment, il va mettre beaucoup plus de temps pour **trouver notre minimum**.

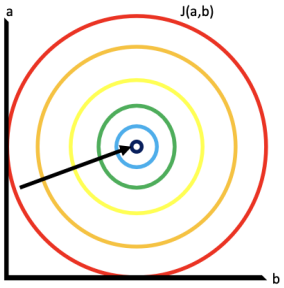



Pour éviter ce problème, on cherche à **modifier les données d'entrée** (**x1** et **x2**) afin qu'elles aient :

- une **même dimension**,
- une **même échelle**, et donc une **fonction de coût plus régulière** (plus "ronde").

Et pour ça, ce qu'on va faire, c'est qu'on va faire un **scaling des données** (**mise à l'échelle)**, on peut aussi dire en français : on va **centrer et réduire les données**.

On aura donc :




Donc ça, qu'est-ce que ça veut dire ?
Si on reprend nos données $\mathbf{x_1}$ et $\mathbf{x_2}$, le **centrage** va avoir pour but que la **moyenne de $\mathbf{x_1}$** et la **moyenne de $\mathbf{x_2}$** soient la même, et pour ça, on va mettre les moyennes de $\mathbf{x_1}$ et $\mathbf{x_2}$ **à 0**.
Pour ça, c'est simple, on va prendre les données de $\mathbf{x_1}$ et on va **soustraire la moyenne de $\mathbf{x_1}$** à chacune des données.
Donc, notre **nouvelle moyenne** sera égale à l'**ancienne moyenne moins celle-ci**, donc **zéro**.\

**Centrage** : $x - \mu \rightarrow$ on met la moyenne à 0.

Ensuite, la **réduction** va consister à faire en sorte que les valeurs de $\mathbf{x_1}$ et $\mathbf{x_2}$ soient réparties sur la **même fourchette**, et donc la **même dimension**.
Et pour ça, on va **diviser chacune des valeurs** pour nos deux variables, par une valeur qui représente la **dispersion** de nos variables.

**Réduction** : xμ\D→ on met les variables sur une même échelle.

En **mathématique**, il existe différentes **mesures de dispersion**, et donc on peut réduire en utilisant ces différentes mesures : on peut **diviser par la variance**, par l'**écart-type** ou par l'**intervalle de distribution** (appelé en anglais : **range**).

Et donc, la **combinaison du centrage et de la réduction** va nous donner la formule qu'on a :

$$x_i' = \frac{x_i - \mu_i}{S_i}$$

$x_i$ = une valeur pour une variable,

$\mu i$ = représente la moyenne de cette variable et

$S_i$ = représente l'écartype, la variance ou le range de cette variable.

$x_i$'= la nouvelle valeur après centrage et réduction

Le **centrage** et la **réduction**, c'est quelque chose qui est utilisé en **statistiques** dans de nombreuses situations, pas seulement dans le cadre de la **descente de gradient**, et il faut savoir en quoi ça consiste.

Si on reprend nos **exemples** de données :
D'abord la **glycémie à jeun**, donc initalement on a des données qui varient entre **0,8 et 1,8**, avec une **moyenne égale a 1**.
Donc si on fait **un centrage et une réduction,** en utilisant le range qui sera donc ici de 1 (1,8-0,8), on obtiendra des **nouvelles valeurs** pour nos données qui varieront entre **-0,2 et 0,8.**

On part des valeurs de la glycémie :

- minimum : $0{,}8$
- maximum : $1{,}8$
- moyenne : $1$
- range : $1{,}8 - 0{,}8 = 1$

On applique la formule du centrage-réduction :

$$x' = \frac{x - \mu}{D} \quad \text{avec } \mu = 1 \text{ et } D = 1$$

On regarde ce que deviennent les valeurs extrêmes :

- Pour la plus petite valeur $x = 0{,}8$ :

$$x' = \frac{0{,}8 - 1}{1} = -0{,}2$$

- Pour la plus grande valeur $x = 1{,}8$ :

$$x' = \frac{1{,}8 - 1}{1} = 0{,}8$$

C'est de là que viennent les bornes **–0,2** et **0,8** :

Pour la glycémie, on constate donc que la **dimension des données** ne change pas énormément après centrage-réduction.

En revanche, si on prend **l'âge** qui varie ici de **30 à 85 ans**, avec une **moyenne qui est de 60 ans**, si on refait la même chose, cette fois **notre range est de 55 (85 - 30)**, et donc après centrage et réduction, les nouvelles valeurs que prendront la variable âge vont aller de **-0,54 à 0,45.**

- Pour l'âge minimum $x = 30$ :

$$x' = \frac{30 - 60}{55} = \frac{-30}{55} \approx -0{,}54$$

- Pour l'âge maximum $x = 85$ :

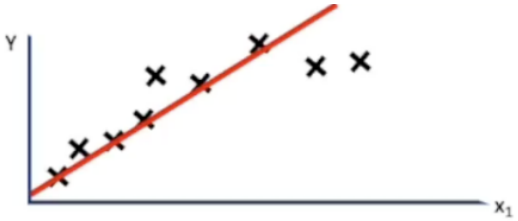
$$x' = \frac{85 - 60}{55} = \frac{25}{55} \approx 0{,}45$$

Et donc pour la variable **âge**, le **scaling** a vraiment diminué la dimension et a fait en sorte que maintenant nos deux variables, **la glycémie à jeun et l'âge**, ont des **dimensions similaires**. Cela permet de **faciliter la réalisation de la descente de gradient.**

## VIII. Pourquoi il y a un intérêt à créer des modèles qui prennent en compte plusieurs variables explicatives ? (Point important du machine learning)

Donc pour ça on va reprendre notre distribution de données avec une variable Y exprimée en fonction d'une variable explicative $x_1$. On peut reprendre l'exemple de la prédiction du prix de maison fonction de leur surface en mètre carré.

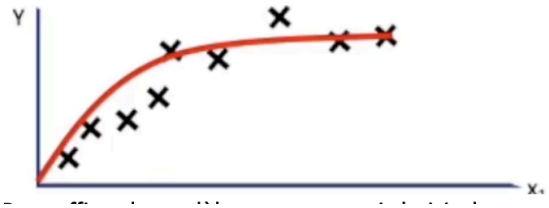
1. *Donc on a vu tout à l'heure* **La régression linéaire** à une seule variable et ça nous a servi à expliquer un peu comment fonctionne la descente de gradient.
   On a donc vu comment on peut utiliser cette **descente de gradient** pour trouver les paramètres $\theta_0$ et $\theta_1$ tel qu'on les a ici pour tracer **la droite de régression** qu'on a en rouge ici qui permettrait de prédire le prix de la maison en fonction de la surface en mètre carré.




$$Y=\theta_0 + \theta_1 x_1$$

Cependant, on voit que ce **modèle est imparfait** et en effet, on sait que le prix d'une maison ne va pas simplement se résumer à cette surface. Donc il va falloir **affiner** notre modèle.

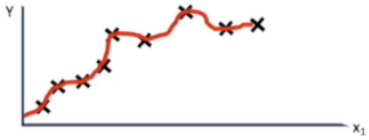
2. Pour ce faire, on va pouvoir par exemple **introduire de nouvelles variables explicatives.**
   Par exemple, on peut **intégrer une variable $x_2$** qui pourrait par exemple représenter la distance du bien immobilier par rapport au centre-ville ou la surface d'une éventuelle terrasse et pour cette nouvelle variable $x_2$, il va falloir trouver un nouveau paramètre $\theta_2$




$$Y=\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Pour **affiner** le modèle, on peut aussi choisir de prendre en compte **des variables supplémentaires** qui correspondraient par exemple à $x_1$ porté au carré ou porté au cube ou des racines carrées de $x_1$ et on ferait à ce moment-là plus de la régression linéaire mais de la **régression polynomiale** *qu'on n'abordera pas en détail.*

3. Mais ce qu'il faut comprendre c'est qu'intégrer de **nouvelles variables** à notre modèle, ça peut permettre de **l'affiner** et donc d'en **augmenter** la permanence.




$$Y=\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1 \cdot x_2 + \dots$$

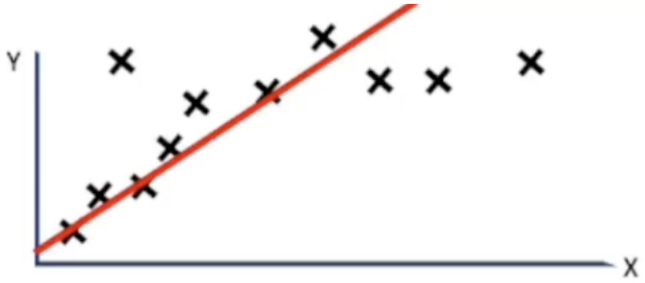
Modèle basé sur un grand nombre de données $x_1, x_2, x_3 \dots$

On peut créer de nouvelles données en combinant certaines variables :

Ex : $x_3 = x_1 \cdot x_2$

Et ainsi, si on est **encore plus loin qu'on intègre d'autant plus de données** que ce soit de nouvelles variables explicatives ou des dérivés des variables explicatives qu'on a déjà, on peut arriver à un **modèle tellement complexe** qu'il passera par l'ensemble des points de nos données existantes.
Cependant, on va voir qu'avoir un modèle **trop complexe, c'est parfois délétère.**

Si on reprend un jeu de données avec Y le prix de nos maisons et X la surface en mètre carré, avec cette fois-ci des variations un peu plus importantes :

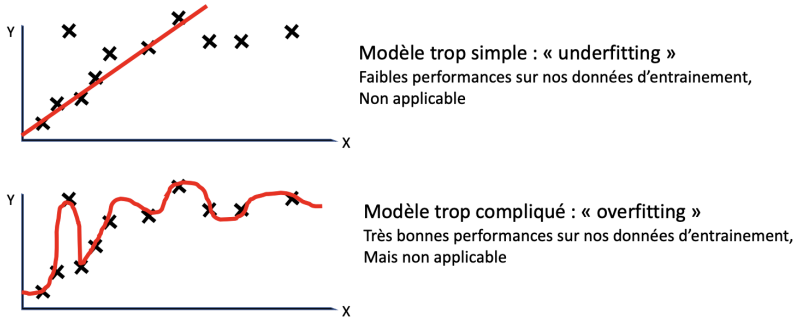



On peut encore une fois utiliser la régression linéaire à une seule variable, ce qui va nous donner cette courbe qui comme on l'a vu est imparfaite et ne prend pas en compte d'autres paramètres, comme par exemple la présence d'une terrasse et donc même si la descente de dernière nous a permis de trouver le prix de la maison en fonction de la surface en mètre carré, l'erreur du modèle va rester significative.
Plusieurs données vont être assez éloignées de notre modèle.

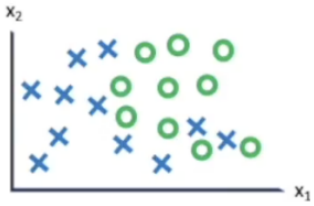
**Le modèle est trop simple** pour représenter la distribution de nos données, donc il ne colle pas assez à nos données et on va dire qu'il y a **underfitting**.

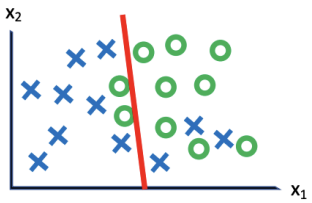
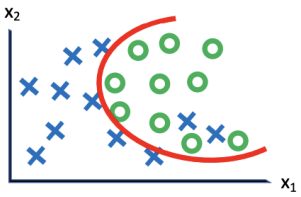
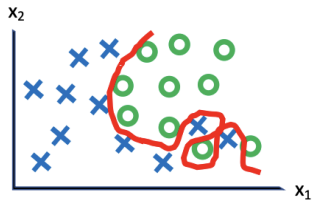
Et donc si on veut réutiliser ce modèle, ce qui reste quand même l'objectif en machine learning pour l'appliquer à de nouvelles données, par exemple si on a une nouvelle maison qui a une surface assez importante, l'application de notre modèle va nous donner un prix qui sera beaucoup plus élevé que la réalité et donc on n'arrivera pas à vendre cette maison. Donc on ne va pas pouvoir utiliser correctement ce modèle

Maintenant si au contraire on utilise un **modèle très complexe**, on peut arriver à une erreur qui est quasi nulle et qu'un modèle qui au final colle parfaitement à nos données, même celle qui correspond au final à des exceptions. Et donc encore une fois si on veut réaliser ce modèle, par exemple pour une maison qui aurait une surface comme ici, on risque de prédire un prix de vente qui encore une fois est trop éloigné de la réalité et on ne pourra donc pas utiliser ce modèle correctement. Et donc dans cette situation où le modèle colle trop bien à nos données d'entraînement, même à toutes les petites exceptions, ce qui le rend **inutilisable** pour de nouvelles données, eh bien on dirait qu'il y a **overfitting**.



Ensuite si on prend un **problème** de **classification**, donc ici un problème avec deux classes représentées par des ronds et des croix, dont on a représenté la distribution en fonction de deux variables $x_1$ et $x_2$, on pourra avoir le même type de problème :




| | |
|---|---|
|  Underfitting | En effet, comme on l'a vu, le but va être de séparer ces deux classes et pour ce faire, on peut utiliser une simple droite comme ici, où on considérera que tout ce qui est à gauche de la droite appartiendra à une première classe, donc ici les croix, et tout ce qui est à droite de la droite appartiendra à la classe des ronds. Sauf qu'avec ce modèle simple, plusieurs de nos individus vont être mal classés. |
|  Bon compromis | Donc on peut avoir envie d'utiliser un modèle plus complexe qui prend en compte d'autres variables. Par exemple, un modèle comme celui-ci qui va mieux séparer nos deux classes, mais qui restera imparfait. |
|  Overfitting | Et si on complique encore plus le modèle, on peut arriver à quelque chose comme ça, où tous nos individus font partie de la bonne classe. Cependant, encore une fois, si on veut réaliser ce modèle sur de nouvelles données, il risque de moins bien marcher que notre modèle intermédiaire. |

Donc comme pour la **régression**, on peut avoir de **l'underfiting** si notre modèle est trop simple ; de l'**overfiting** si il est trop complexe, et ce qu'il va falloir trouver, c'est un **bon intermédiaire** qui va suffisamment bien séparer nos données d'entraînement tout en étant applicable pour de nouvelles données.

Donc **l'underfiting**, c'est le fait de créer des modèles qui sont trop simples pour expliquer nos données.
Et la **solution** à ça, ça va être de **prendre en compte plus de variables explicatives.**

Si on reprend notre exemple où on veut prédire le prix des maisons, si on se rend compte que l'utilisation de la surface en mètres carrés ne suffit pas à avoir un modèle pour correctement prédire le prix de la maison, on peut rajouter des variables comme la présence d'une terrasse ou la présence d'une piscine.

D'autres choses qu'on sait peuvent influencer le prix de notre bien.

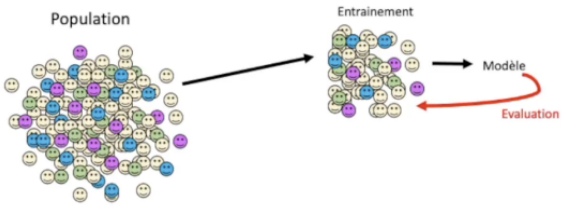
Mais il faudra garder en tête que **rajouter des données**, ça **rajoute aussi un risque d'overfiting**.
Et pour **limiter ce risque d'overfiting**, il faudra tout de même **limiter le nombre de variables explicatives** qu'on utilise.
Ceci peut se faire à la main, donc par l'utilisateur.
Donc si on reprend l'exemple de la vente immobilière, l'utilisateur pourra choisir de ne pas prendre par exemple la couleur des tuiles comme variable explicative car a priori ce n'est pas quelque chose qui devrait influencer le prix de la maison.
Et l'**utilisateur ne gardera que des variables qui sont pertinentes.**

Cependant dans d'autres applications, on ne sait pas forcément à l'avance qu'est-ce qui est pertinent.

Notamment en biologie ou en médecine, on peut travailler avec des **données type omiques** comme des données de génétique ou de génomique où **on a énormément de variables** et à l'avance **on ne sait pas forcément lesquelles vont être présentes.** On pourra dans ce cas-là utiliser des méthodes qui permettent de réduire le nombre de données qu'on prend en compte en essayant d'éliminer les variables qui apportent une information redondante. On ne verra pas en détail comment ça fonctionne mais cela existe.
**Une alternative si on veut garder toutes nos variables c'est d'utiliser une méthode qui s'appelle la régularisation.** Cette méthode fonctionne en **limitant la magnitude des valeurs** que peuvent prendre les paramètres $\theta$ ce qui va permettre de limiter la complexité des modèles qu'on entraînera mais c'est un peu plus compliqué et on n'entrera pas là-dedans en détail.
Enfin ce qu'il faut comprendre c'est que si l'underfiting et l'overfiting peuvent limiter la qualité des modèles qu'on crée et leur utilisation une autre chose qui est très importante c'est de travailler avec des données qui sont de **bonne** qualité.
En effet quand on entraîne un modèle celui-ci se base sur les données qu'on a entrées et donc si ces données contiennent des erreurs notre modèle va être basé sur des erreurs et donc si on veut l'appliquer à des nouvelles données il pourra se tromper.

Les algorithmes d'apprentissage comme la descente des gradients ne prennent pas en compte cette possibilité qu'il existe des erreurs ils prennent les données qu'on leur donne comme des **vérités absolues**.

Une autre chose qui va être importante quand on fait de l'apprentissage supervisé et qui se recoupe avec cette notion de qualité de données, ça va être notre échantillonnage pour entraîner le modèle.
Donc notre objectif en machine learning ça va être de créer un modèle à partir d'un échantillon d'une population pour ensuite pouvoir utiliser ce modèle pour l'appliquer à l'ensemble de la population et donc on va retrouver les mêmes problèmes d'échantillonnage que pour des tests statistiques descriptifs.



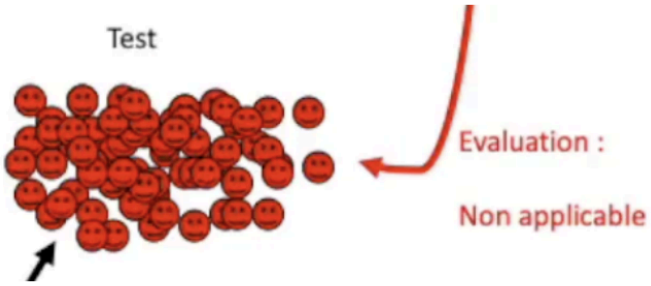

Donc **à partir d'une population** on va **extraire une cohorte ou un échantillon** qui va nous servir pour entraîner le modèle et on va faire une première évaluation du modèle sur cette cohorte on reverra ensuite comment on va appeler cette cohorte notre **cohorte d'entraînement.**


Ensuite on va **extraire une deuxième partie** de la population un deuxième échantillon qu'on va utiliser pour tester uniquement notre modèle pour l'évaluer on appellera cette cohorte notre **cohorte de test.**

Et c'est très important que la cohorte d'entraînement comme la cohorte de test soit bien représentative de notre population générale.

En effet si on prend une **cohorte d'entraînement qui est non représentative** par exemple si on entraîne notre modèle sur un sous-échantillon de notre population ici les patients bleus alors : On va créer un modèle qui pourra être applicable à ce sous-échantillon au bleu mais qui ne sera **pas forcément applicable pour l'ensemble de la population.**

C'est aussi un peu ce qui se passe quand on est confronté à un **problème d'overfiting** dans ce cas-là même si on est sur **un échantillon qui est représentatif de la population générale** pour nous le **modèle est tellement complexe** qu'il prend en compte des **variables qui sont en réalité très spécifiques** à nos données d'entraînement et dont la distribution ne sera pas forcément la même sur le reste de la population.




De même notre **population test** sur laquelle on va évaluer notre modèle elle doit être **représentative de notre population**, si on prend un échantillon de patients qui est extrait d'une autre population peut- être que notre modèle serait efficace sur notre population initiale sauf que sur cette nouvelle population il ne va pas forcément être applicable.
C'est aussi pour ça que si on entraîne un modèle sur des parents venant d'un pays occidental on ne va pas forcément appliquer ce modèle sur des parents d'un pays du tiers monde.

→ Donc il faudra bien **faire attention que nos cohortes d'entraînement** et de **test** soient toutes les **deux représentatives de la population générale** sur laquelle on veut travailler.

Donc comme on l'a vu on va avoir des données d'entraînement, celle-ci vont nous permettre d'entraîner le modèle et on va faire une première évaluation du modèle sur ces données pour vérifier qu'il est suffisamment performant et ensuite pour vérifier que ces performances sont **reproductibles et applicables à l'ensemble de notre population** on va réévaluer notre modèle sur une population test.

# IX. Crossvalidation

= méthode qui est fréquemment utilisée pour l'**évaluation des modèles** si on constate une corde d'entraînement avec 24 patients .


Sur ces **patients** on va vouloir donc **entraîner le modèle** et **évaluer ce modèle** pour se faire ce qu'on pourrait faire c'est utiliser l'**ensemble de la population** pour faire **l'entraînement** et ensuite **appliquer le modèle pour l'évaluation.** Cependant, comme ce sont les mêmes patients qui ont servi à l'entraînement, on risque de **surestimer les performances du modèle**..

Donc ce qu'on va faire c'est qu'on va séparer notre population

- **Grande partie** (environ les trois quarts) servira à **l'entraînement** du modèle,
- **Petite partie** (environ un quart) sera mise de côté uniquement pour **l'évaluation**.

Ces patients réservés à **l'évaluation n'auront pas été utilisés lors de l'entraînement,** ce qui permet une estimation plus réaliste des performances.

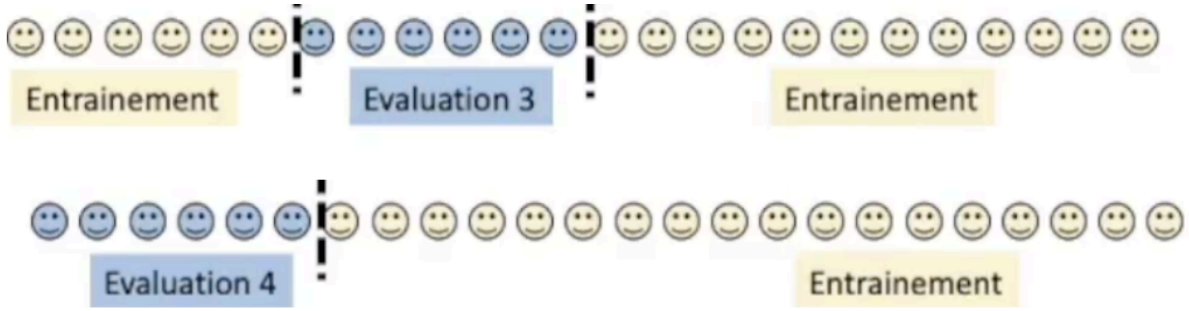

Maintenant ce qu'il faut savoir c'est qu'**un modèle robuste** doit :

- Avoir été entraîné sur un **nombre suffisamment important de patients**
- Et surtout être testé sur un nombre suffisant de patients.

  Ici notre évaluation de la performance basée sur **six parents** ce n'est **pas très robuste.** On va utiliser une astuce avec une méthode qui s'appelle la **cross-validation**, donc on va faire notre première étape avec un entraînement sur 18 patients et un test sur six patients.

2.


Ensuite on va recommencer mais cette fois-ci en prenant sur nos **24 patients**, 18 nouveaux patients pour l'entraînement, et six nouveaux patients pour le test.
Pour le test, les six patients qui avaient été utilisés pour l'évaluation précédemment sont utilisés pour l'entraînement, et on prend dans les 18 autres, six nouveaux patients pour l'entraînement.




On va recommencer **une troisième fois** puis une **quatrième fois,** et donc au final à la place d'évaluer notre modèle **seulement sur un quart de notre population** on a fini par évaluer notre modèle sur **4 quarts** de la population donc **l'ensemble** de la population et ça ça va nous donner une première **information** sur la **qualité de notre modèle** sur cette population d'entraînement.

Dans ce cas précis comme on a coupé notre population en quatre on dit qu'on a fait la cross-validation à quatre **fold** mais on aurait aussi pu le faire en six en trois peu importe on peut faire ceKe cross-valida=on en u=lisant un nombre de fold plus ou moins important.
Ce qui aura une petite influence sur le résultat final et dans tous les cas on aura évalué notre modèle sur l'ensemble des données de notre échantillon.

Ensuite il faudra tout de même évaluer ce modèle sur une deuxième population : une population test celle-ci étant indépendante de la population d'entraînement pour affirmer que notre modèle est bien reproductible.

• Tableau de contingence :

| | | Vérité | |
|---|---|---|---|
| | | Y vrai = négatif | Y vrai = positif |
| Prédiction | Y prédit = négatif | Vrais Négatifs (VN) | Faux Négatifs (FN) |
| | Y prédit = positif | Faux Positifs (FP) | Vrais Positifs (VP) |

Sensibilité (Se) = $\frac{VP}{VP+FN}$

Spécificité (Sp) = $\frac{VN}{VN+FP}$

Valeur Prédictive Positive (VPP) = $\frac{VP}{VP+FP}$

Valeur Prédictive Négative (VPN) = $\frac{VN}{VN+FN}$

Accuracy (Précision) Acc = $\frac{VP+VN}{VP+FP+VN+FN}$

Maintenant on va voir comment on fait pour cette **évaluation**.
Tout d'abord on va parler du cas de la **classification** dans ce cas-là l'évaluation, on va être proche des tests qu'on utilise pour évaluer la performance d'un **examen diagnostique.**
On utilise alors un tableau de contingence.
Dans les cohortes que l'on étudie, on connaît le label réel que le modèle devrait prédire : on le note **Y vrai.**
On fait ensuite tourner le modèle, ce qui nous fournit des valeurs **Y prédites** par le modèle.

On compare alors le **Y prédit au Y vrai** afin d'évaluer si le modèle fonctionne correctement.
Si l'on travaille avec **deux classes**, on obtient un **tableau de contingence** dans lequel :

- les **Y prédits** sont placés à gauche,
- les **Y vrais** sont placés en haut.
- Lorsque le **Y vrai est négatif** et qu'il est correctement prédit comme **négatif**, on obtient un **vrai négatif.**
- Lorsque le **Y vrai est positif** et qu'il est correctement prédit comme **positif**, on obtient un **vrai positif**.

**En revanche :**

- Si le **Y vrai est positif** mais qu'il est prédit comme **négatif**, on parle de **faux négatif** : le modèle donne un résultat négatif alors qu'il aurait dû être positif.
- Si le modèle donne un résultat **positif** alors que le **Y vrai est négatif**, on parle de **faux positif.**

À partir de ces quatre cas (**vrais positifs, vrais négatifs, faux positifs, faux négatifs**), on peut calculer différentes **métriques** qui fournissent des informations sur la **qualité du modèle.**

À partir de là on va pouvoir mesurer **différentes métriques** qui vont nous donner une information sur la qualité du modèle.

Tout d'abord on aura :
- La **sensibilité** = la capacité du modèle à prédire un résultat **positif** quand il doit prédire ce résultat positif
- la **spécificité** = la capacité du modèle à prédire un résultat **négatif** quand il doit être négatif

Ensuite on va pouvoir mesurer des valeurs prédictives avec la **valeur prédictive positive (VPP)** qui correspondra à la **probabilité qu'un résultat soit effectivement positif quand le modèle la prédit** en tant que tel.

Puis la **valeur prédictive négative (VPN)** qui correspondra à la **probabilité qu'un résultat soit effectivement négatif quand le modèle la prédit** en tant que tel.

Enfin en machine learning on va utiliser une autre métrique qui s'appelle **l'accuracy** = **pourcentage de fois où le modèle a correctement fait la prédiction** et donc au nombre de patients qui ont été correctement prédits sur l'ensemble de la cohorte évaluée .

Cette **métrique l'accuracy** est assez **pratique** car elle peut aussi assez facilement être utilisée, quand on utilise **plus de deux classes** par exemple on peut avoir une **table de contingence** comme ici :

| | Y vrai = 1 | Y vrai = 2 | Y vrai = 3 |
|---|---|---|---|
| Y predit = 1 | 50 | 5 | 5 |
| Y predit = 2 | 10 | 52 | 3 |
| Y predit = 3 | 5 | 0 | 60 |

Ici, avec **trois classes** dans ce cas-là, si on veut parler de **sensibilité** ou de **spécificité** il faudra expliquer pour **quelle classe** on parle.

On pourra par exemple parler de la **sensibilité** pour la **classe 1** qui est égale donc au nombre de patients effectivement de classe 1 qui sont prédits comme étant de classe 1 divisé par la totalité des patients qui sont effectivement de classe 1:

$$\text{Sensibilité classe 1} = \frac{\text{nombre de patients effectivement de classe 1 prédits comme classe 1}}{\text{total des patients effectivement de classe 1}}$$

ce qui va donc nous donner une information **uniquement** sur la **qualité du modèle** pour cette **classe 1**, alors que **l'accuracy** à ce moment-là correspondra au **nombre de patients dont la classe prédite est correctement divisée par la totalité des patients.**

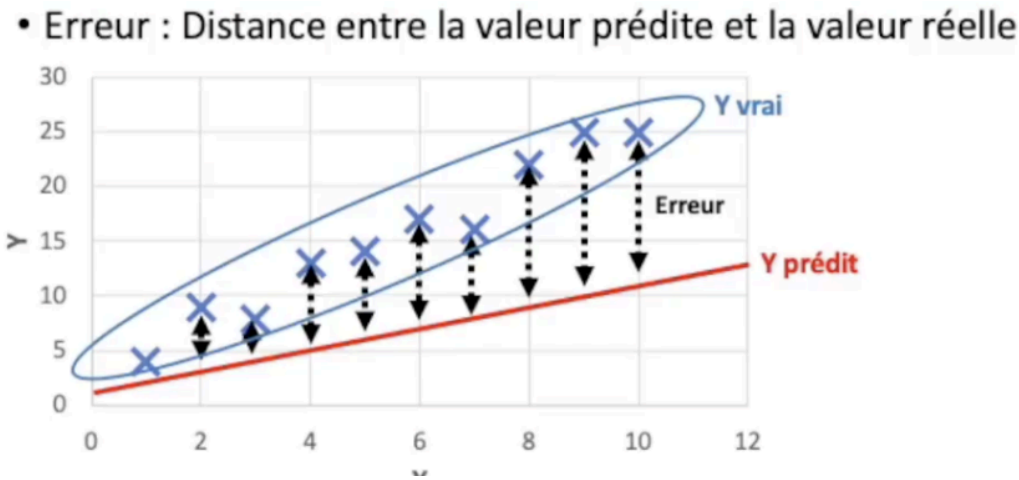
Ce qui est donc une métrique qui représente l'ensemble du modèle.

| | Y vrai = 1 | Y vrai = 2 | Y vrai = 3 |
|---|---|---|---|
| Y predit = 1 | 200 | 20 | 20 |
| Y predit = 2 | 0 | 0 | 0 |
| Y predit = 3 | 0 | 0 | 0 |

$$\text{Accuracy} = \frac{\textit{bonnes prédictions (200)}}{\textit{total (240)}} = 83{,}3\%$$

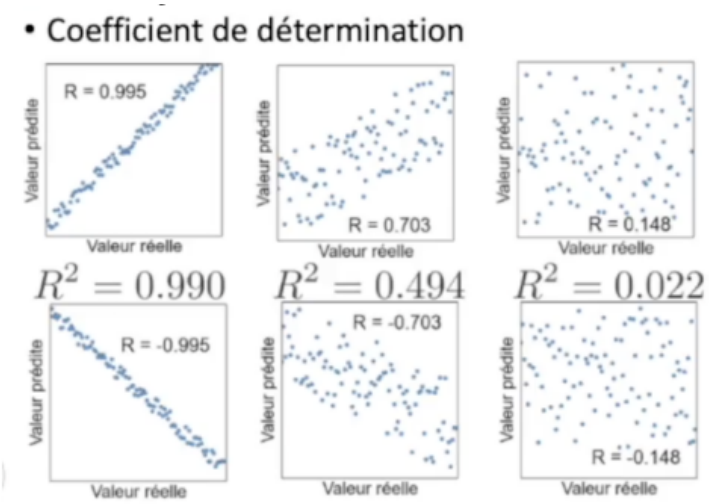
Cependant avec cette métrique il faudra faire attention entre la répartition des classes dans nos populations n'est pas équilibrée.
En effet, si on prend une population avec 200 patients dans la classe 20 patients dans la classe 2 et 20 patients dans la classe 3 même si le modèle met tout le monde dans la classe 1 et est donc inutile.

L'accuracy sera de 200 sur 240 donc de 83% ce qui pourra donner l'impression que le modèle fonctionne correctement alors qu'il fait quelque chose qui est complètement inutile à savoir mettre tout le monde dans le même groupe.



Pour ce qui est de l'évaluation d'un modèle de régression comme notre label Y peut prendre tout un tas de valeurs, on ne va pas utiliser une table de contingence, on va prendre notre cohorte de test et on va y appliquer notre modèle.
Ainsi encore une fois pour tous les individus de notre échantillon on va avoir un Y prédit et un Y vrai et on va comparer ces deux Y pour obtenir l'erreur réalisée par le modèle qui va correspondre à la distance entre le Y prédit et le Y vrai.

Il existera différentes mesures de l'erreur on peut utiliser la somme des différences portée au carré ou la racine carrée de cette somme de différence portée au carré quoi qu'il en soit il se sera toujours une mesure de l'erreur.




On pourra également utiliser une métrique qui s'appelle le R carré ou R2 ou coefficient de détermination. Ce coefficient de détermination correspond au carré du coefficient de corrélation entre les Y prédits et les Y vrais.
Ainsi il faut faire attention car si les Y prédits et les Y vrais sont inversement corrélés le R carré pourra tout de même être très enlevé et on pourra donc avoir faussement l'impression que notre modèle fonctionne très bien alors qu'il renvoie une valeur inverse à celle qui est attendue.

*Mais on ne rentrera pas dans les détails concernant le coefficient de détermina8on et il ne fera pas l'objet d'une question pour l'examen*

**Résumé :**
Pour l'évaluation des modèles en apprentissage supervisé on va faire une première évaluation par cross validation sur notre cohorte d'entraînement ce qui va nous donner une information sur le fit du modèle si les performances sont trop basses on pourra dire que le modèle est **underfit** et si les performances sont très élevées avec une **accuracy** proche de **100%** cela peut vouloir dire que le modèle est très bon mais cela peut aussi être un signe **d'overfiting**.

Ensuite faire une deuxième évaluation sur notre cohorte de test en appliquant directement notre modèle ce qui va nous permettre d'évaluer la reproducbilité de ce modèle pour le reste de notre population.
Si les performances sont équivalentes à celle mesurée sur la cohorte d'entraînement alors que cela veut dire que nos résultats sont **reproductibles**.

Cependant, si les **performances** sont bien moins bonnes que celles mesurées sur la **cohorte d'entraînement**, alors cela peut vouloir dire que notre **modèle** est **overfitting**, ou sinon il y a un **problème d'échantillonnage** et que notre **cohorte de test** est trop différente de notre **cohorte d'entraînement**.
Donc au final, pour appliquer nos **méthodes de machine learning**, que ce soit en **médecine**, en **biologie** ou dans d'autres domaines, il faudra tout d'abord bien **définir notre problème**.

1. Bien définir son problème :
   - Apprentissage Supervisé / Non supervisé.
   - Régression / Classification

2. Bien choisir ses données :
   - Bon échantillonage
   - Bien définir le label Y
   - Bien définir les variables explicatives x

On va **définir la question** à laquelle on va répondre et à partir de ça on saura si on a besoin d'un **apprentissage supervisé** ou **non-supervisé** et s'il faudrait utiliser une **méthode de classification** ou de **régression**.
En parallèle, la **définition de notre problème** va nous permettre de choisir les **données** qui permettront d'y répondre. Pour ce faire, il faudra faire un **bon échantillonnage** dans notre **population cible**, bien définir notre **label** Y dont la **méthode de mesure** devra être **fiable**, et de même il faudra bien définir nos **variables explicatives** X qu'on pourra **sélectionner** manuellement, à l'aide d'un **algorithme dédié**, ou un peu des deux, et pour lesquelles les **méthodes de mesure** devront être également **le plus fiables possible**.

3. Adapter son algorithme d'apprentissage
   - Comprendre le principe d'un algorithme comme la descente de gradient
   - Savoir si un scaling est nécessaire
   - Savoir adapter le taux d'apprentissage

4. Evaluer son modèle
   - Comprendre l'underfitting et l'overfitting
   - Comprendre la cross validation
   - Comprendre les mesures d'évaluation

On utilisera ensuite un **algorithme d'apprentissage** par exemple la **descente de gradient** même s'il en existe d'autres et pour ce faire il faudra comprendre le **principe de fonctionnement** de cet **algorithme** pour pouvoir **ajuster les paramètres**.

Comme par exemple pour la **descente de gradient** savoir **ajuster le taux d'apprentissage**. De même, au moment où on aura analysé les **données d'entrée** il faudra savoir si un **scaling** est nécessaire.

L'application de notre **méthode d'apprentissage** va nous donner un **modèle**, et il va falloir **évaluer les performances** de ce **modèle**. Pour ce faire il faudra comprendre et connaître ce qu'est l'**underfitting** et l'**overfitting**, il faudra comprendre ce qu'est la **cross-validation** et il faudra comprendre quelles sont les **mesures** qui permettent de faire cette **évaluation**.

En particulier pour la **classification supervisée** avec la **sensibilité**, la **spécificité**, l'**accuracy**…

**FIN**

Dédi a Lilalala delcaillou, trop mv cette go

*Les gars (Si vous être brun, +180cm, sportif, DRÔLE et gênant), si vous voulez son snap, n'hésitez pas à m'envoyer un message sur messenger*