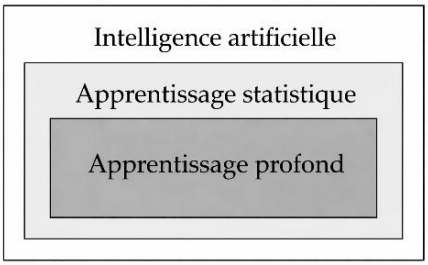
COURS : ALGORITHMES POUR L'INTELLIGENCE ARTIFICIELLE
== NOTIONS GÉNÉRALES ==

I) INTRODUCTION, NOTIONS GÉNÉRALES 2
I.1. Données et représentation vectorielle 3
I.2. Qu'est-ce que l'apprentissage statistique 5
I.3. Prédiction et interprétation 6
I.3.1. Prédiction et erreur irréductible associée 7
I.3.2. Interprétation 8
I.4. Comment est estimée la fonction de prédiction 8
I.4.1. Méthodes paramétriques 9
I.4.2. Méthodes non paramétriques 10
I.5. Apprentissage supervisé et non-supervisé 12
I.5.1. Apprentissage supervisé 12
I.5.2. Apprentissage non supervisé 12
I.6. Régression et classification 13
II) PERFORMANCES D'UN MODÈLE 14
II.1. Fonction de perte et risque empirique 14
II.2. Phénomène de sur-apprentissage 15
II.3. Performances d'un modèle en régression 16
II.4. Performances d'un modèle en classification 18
II.4.1. Taux de réussite 18
II.4.2. Taux de réussite équilibré en classification binaire 18
II.4.3. Taux de réussite équilibré en classification multi-classes 19
II.4.4. Matrice de confusion en classification binaire 19
II.4.5. Matrice de confusion en classification multi-classes 21
II.5. Jeu de données pour les tests 22
II.6. Normalisation des données 23

I) INTRODUCTION, NOTIONS GÉNÉRALES

L'intelligence artificielle (IA) recouvre un ensemble très large de problématiques et de techniques, dont les contours varient selon que l'on insiste sur les objectifs (imiter, reproduire, assister, comprendre) ou sur les méthodes mises en œuvre. Historiquement, l'IA s'est structurée autour de questions fondatrices (dans un article de Turing publié en 1950, puis dans un colloque organisé au Dartmouth College en 1956), et s'est progressivement outillée pour traiter des tâches de décision, de perception ou d'analyse.

L'apprentissage peut être défini comme l'acquisition de connaissances et de compétences par la pratique et l'expérience. Dans le cas d'un ordinateur, les compétences peuvent être matérialisées par des algorithmes et la pratique correspond à l'utilisation des données fournies à la machine.



On distingue trois ensembles inclus les uns dans les autres : l'intelligence artificielle, très générale, qui inclut l'apprentissage statistique (apprentissage automatique, machine learning), qui lui-même inclut l'apprentissage profond (deep learning).

L'intelligence artificielle trouve des applications dans tout type de domaines : traduction automatique de textes, pilotage de véhicules autonomes, de robots, reconnaissance et assistance vocales, génération d'images, reconnaissance de formes, de visages, imagerie, jeux de stratégie, etc.

L'apprentissage statistique intervient dans des systèmes de repérage de comportements d'une clientèle dans un but de publicité ciblée, dans la programmation de stratégies automatisées en finance de marché, dans l'assistance au diagnostic médical...

Dans ce chapitre, on se place dans le cadre de l'apprentissage statistique. En apprentissage statistique, le principe est d'utiliser un grand nombre de données pour que l'algorithme apprenne une méthode qu'il saura utiliser sur des données nouvelles : alors qu'un algorithme classique retourne un résultat à partir des données d'entrée, un algorithme d'apprentissage retourne un algorithme classique qui sera appliqué à des données nouvelles. L'enjeu n'est donc pas seulement de bien « répondre » sur les exemples déjà vus, mais de généraliser à des entrées inédites, ce qui distingue un apprentissage utile d'une simple mémorisation.

D'un point de vue algorithmique, cette démarche s'inscrit dans un pipeline assez standard. On collecte ou constitue un jeu de données, on choisit une représentation adaptée des objets étudiés (les « caractéristiques » ou *features*), on entraîne une méthode sur une partie des données, puis on valide la qualité des résultats sur des données distinctes. Cette structuration a une conséquence pratique importante : la nature et la diversité des données d'entraînement conditionnent fortement la capacité de généralisation, et une base trop petite ou peu représentative conduit souvent à des décisions peu fiables dès que l'on sort des cas déjà rencontrés.

I.1. Données et représentation vectorielle

Supposons que nous voulions étudier l'association entre la publicité et les ventes d'un produit particulier. Le jeu de données `Advertising` contient les ventes de ce produit dans 200 marchés différents, ainsi que les budgets publicitaires du produit dans chacun de ces marchés pour trois médias distincts : la télévision, la radio et la presse écrite. Les données sont présentées à la Figure 1. Si nous déterminons qu'il existe une association entre la publicité et les ventes, nous pourrons recommander au client d'ajuster les budgets publicitaires, augmentant ainsi indirectement les ventes. Autrement dit, notre objectif est de développer un modèle précis pouvant être utilisé pour prédire les ventes à partir des budgets alloués aux trois médias.

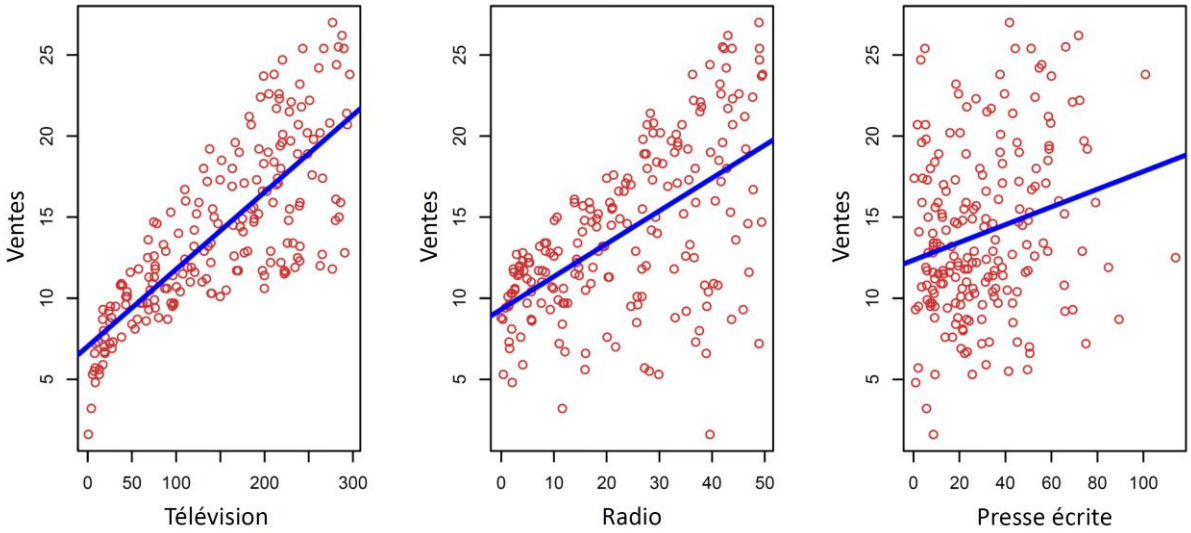


Figure 1 : Ventes en fonction des budgets TV, radio et presse écrite, en milliers de dollars

Dans ce contexte, les budgets publicitaires sont des *variables d'entrée*, tandis que les ventes constituent une *variable de sortie*. Les variables d'entrée sont généralement notées à l'aide du symbole X, avec un indice pour les distinguer. Ainsi, X_1 peut correspondre au budget TV, X_2 au budget radio et X_3 au budget presse écrite. Les variables d'entrée portent différents noms, comme *prédicteurs*, *caractéristiques (features)*, *variables indépendantes* ou, parfois, simplement *variables*. La variable de sortie — ici, les ventes — est souvent appelée la *réponse*, *cible*, *label, étiquette* ou *variable dépendante*, et est généralement notée Y.

Ici, le jeu de données est un ensemble de couples $\{(x_i, y_i)\}_{i=1..n}$, avec n = 200, où chaque $x_i = ((x_i)_1, (x_i)_2, (x_i)_3)^T \in \mathbb{R}^3$ décrit les 3 caractéristiques (budget TV, budget radio et budget presse écrite) du vecteur des caractéristiques x_i.

On utilise généralement une majuscule pour désigner le vecteur des caractéristiques X et la variable de sortie Y « en général » : ce sont les entrées / sorties abstraites du problème (souvent vues comme des variables aléatoires ou, plus simplement, comme des variables génériques). Les minuscules désignent une valeur concrète observée de X ou de Y, c'est-à-dire un exemple mesuré (une réalisation).
Le choix de représenter une observation par un vecteur $x \in \mathbb{R}^p$ n'est pas anodin : il permet de ramener des objets très différents (images, textes, mesures physiques, etc.) à un format commun manipulable par des algorithmes. Sans entrer ici dans l'ingénierie de caractéristiques, on retiendra l'idée suivante : une fois la représentation fixée, l'algorithme

travaille essentiellement dans un espace vectoriel, où des opérations de comparaison, de regroupement et de décision deviennent possibles.

Imaginons par exemple que l'on souhaite mettre en place un algorithme d'apprentissage automatique afin d'apprendre à un ordinateur à distinguer des images de chats de celles contenant des chiens (Figure 2). Ce genre de tâche, où l'on cherche à apprendre à un ordinateur à distinguer différents types ou classes d'objets (ici, des chats et des chiens), est appelé, dans le jargon de l'apprentissage automatique, un problème de classification.

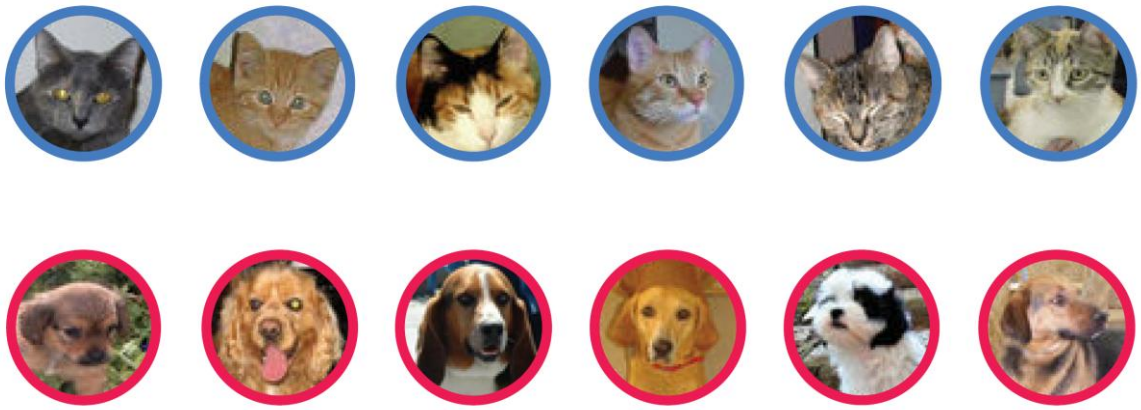


Figure 2 : ensemble d'entraînement composé de six images de chats et de six images de chiens

Pour faire la différence entre des images contenant des chats et celles contenant des chiens, nous (les humains) utilisons la couleur, la taille, la forme des oreilles ou du museau, et/ou une combinaison de ces caractéristiques pour distinguer les deux. Autrement dit, nous ne regardons pas une image comme une simple collection de nombreux petits pixels carrés. Nous en extrayons des détails plus grossiers, ou des caractéristiques, afin d'identifier ce que nous observons. Il en va de même pour les ordinateurs. Pour entraîner avec succès un ordinateur à réaliser cette tâche (et, plus généralement, toute tâche d'apprentissage automatique) nous devons lui fournir des caractéristiques correctement conçues.

Concevoir des caractéristiques de qualité n'est généralement pas une tâche triviale, car elle peut fortement dépendre de l'application. Par exemple, une caractéristique comme la couleur serait moins utile pour discriminer des chats et des chiens (puisque de nombreux chats et chiens ont des couleurs de pelage similaires) qu'elle ne le serait pour distinguer des grizzlis et des ours polaires. De plus, extraire ces caractéristiques à partir d'un jeu de données d'entraînement peut également être difficile. Par exemple, si certaines de nos images d'entraînement étaient floues ou prises sous un angle qui ne permettait pas de voir correctement l'animal, les caractéristiques que nous avons conçues pourraient ne pas être correctement extraites.

Pour notre exemple, on peut cependant imaginer extraire facilement deux caractéristiques de chaque image : la taille du museau relative à la taille de la tête, allant de petite à grande, et la forme des oreilles, allant de ronde à pointue. Avec ce choix de caractéristiques, chaque image peut désormais être représentée par seulement deux nombres : un nombre exprimant la taille relative du museau, et un autre capturant le caractère pointu ou rond des oreilles. Autrement dit, nous pouvons représenter chaque image de notre ensemble d'entraînement dans un espace bidimensionnel à l'aide de vecteurs de caractéristiques, où la

taille du museau et la forme des oreilles constituent respectivement les axes de coordonnées horizontal et vertical (Figure 3) :

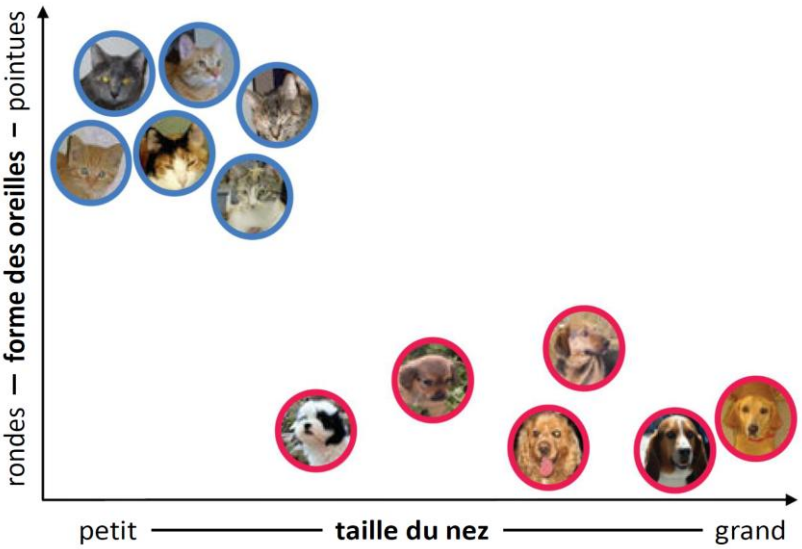


Figure 3 : Représentation dans l'espace des caractéristiques du jeu de données

I.2. Qu'est-ce que l'apprentissage statistique

Plus généralement, supposons que nous observions une réponse quantitative y et p caractéristiques différentes : x_1, x_2, …, x_p. Nous supposons qu'il existe une relation entre Y et $X = (X_1, X_2, \ldots, X_p)$, relation qui peut s'écrire sous la forme très générale suivante :

$$Y = f(X) + \varepsilon$$

Ici, f est une fonction fixe mais inconnue de X_1, …, X_p, et ε est un terme d'erreur aléatoire, indépendant de X et de moyenne nulle. Dans cette formulation, f représente l'information systématique que X fournit à propos de Y.

Comme autre exemple, considérons le panneau de gauche de la Figure 4, qui représente le revenu en fonction du nombre d'années d'études pour 30 individus du jeu de données Revenu.

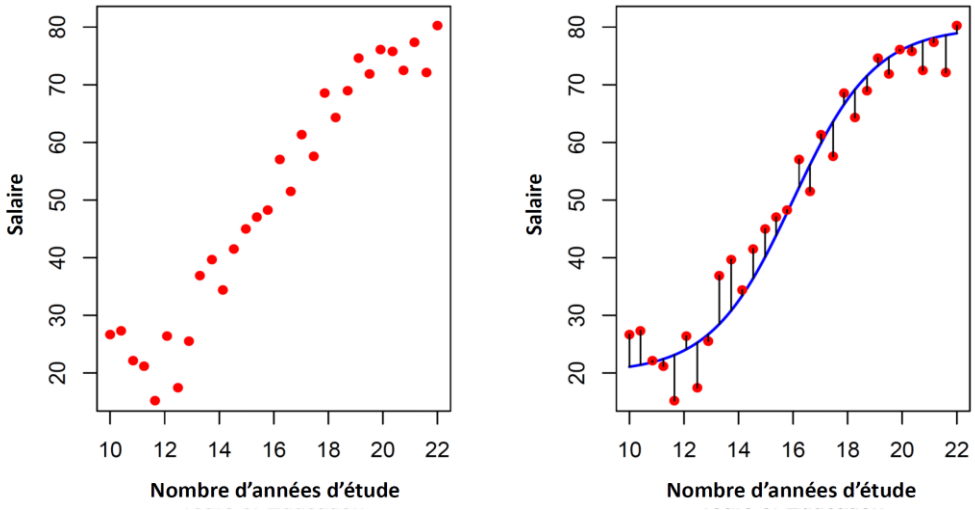


Figure 4 : Jeu de données Revenu

Le graphique suggère qu'il pourrait être possible de prédire le revenu à partir du nombre d'années d'études. Cependant, la fonction f qui relie la variable d'entrée à la variable de sortie est, en général, inconnue. Dans cette situation, il faut estimer f à partir des points observés. Étant donné que Revenu est un jeu de données simulé, f est connue et elle est représentée par la courbe bleue dans le panneau de droite de la Figure 4. Les segments verticaux représentent les termes d'erreur ε. Nous remarquons que certaines des 30 observations se situent au-dessus de la courbe bleue et d'autres en dessous ; globalement, les erreurs ont une moyenne approximativement nulle.

En général, la *fonction de prédiction* (ou *prédicteur*) f peut dépendre de plus d'une variable d'entrée. Dans la Figure 5, nous représentons le revenu en fonction du nombre d'années d'études et de l'ancienneté. Ici, f est une surface bidimensionnelle qui doit être estimée à partir des données observées.

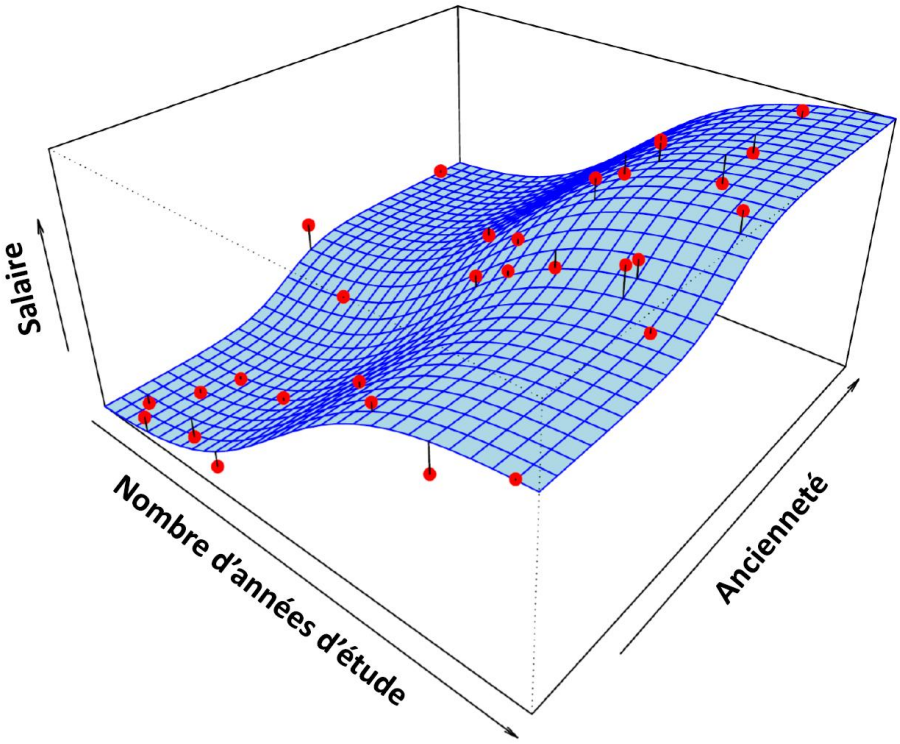


Figure 5 : Revenu en fonction du nombre d'années d'étude et de l'ancienneté

En substance, l'apprentissage statistique désigne un ensemble d'approches visant à estimer f. Dans ce chapitre d'introduction, nous allons voir certains des principaux concepts théoriques qui interviennent dans l'estimation de f, ainsi que des outils permettant d'évaluer les estimations obtenues.

I.3. Prédiction et interprétation

Il existe deux raisons principales pour lesquelles nous pouvons souhaiter estimer f : la *prédiction* et *l'interprétation (*aussi appelée *inférence)*. Dans certains contextes, l'enjeu principal est de prédire correctement la sortie pour de nouvelles entrées ; dans d'autres, on cherche aussi à comprendre quels facteurs expliquent la sortie, ce qui suppose un modèle plus interprétable. Les méthodes abordées dans ce cours sont surtout motivées par l'objectif prédictif.

I.3.1. Prédiction et erreur irréductible associée

Dans de nombreuses situations, un ensemble d'entrées X est facilement disponible, mais la sortie Y ne peut pas être obtenue aisément. Dans ce contexte, puisque le terme d'erreur a une moyenne nulle, on peut prédire Y à l'aide de :

$$\hat{Y} = \hat{f}(X)$$

… où $\hat{f}$ représente notre estimation de f (fonction de prédiction), et $\hat{Y}$ la prédiction correspondante de Y. Dans ce contexte, $\hat{f}$ est souvent considéré comme une « boîte noire », au sens où l'on ne s'intéresse généralement pas à la forme exacte de $\hat{f}$, pourvu qu'elle fournisse des prédictions précises de Y.

À titre d'exemple, supposons que X_1, …, X_p soient des caractéristiques d'un échantillon de sang d'un patient, facilement mesurables en laboratoire, et que Y soit une variable codant le risque, pour ce patient, de subir une réaction indésirable grave à un médicament particulier. Il est naturel de chercher à prédire Y à partir de X, car nous pouvons ainsi éviter d'administrer le médicament en question aux patients qui présentent un risque élevé de réaction indésirable, c'est-à-dire aux patients pour lesquels l'estimation de Y est élevée.

La précision de $\hat{Y}$ en tant que prédiction de Y dépend de deux quantités, que nous appellerons *l'erreur réductible* et *l'erreur irréductible*. En général, $\hat{f}$ ne sera pas une estimation parfaite de f, et cette imperfection introduira une certaine erreur. Cette erreur est dite réductible, car il est potentiellement possible d'améliorer la précision de $\hat{f}$ en utilisant la technique d'apprentissage statistique la plus appropriée pour estimer f. Toutefois, même s'il était possible d'obtenir une estimation parfaite de f, de sorte que notre réponse estimée prenne la forme $\hat{Y} = f(X)$, notre prédiction comporterait malgré tout une part d'erreur. En effet, Y dépend également de ε, qui, par définition, ne peut pas être prédit à partir de X. Par conséquent, la variabilité associée à ε affecte aussi la précision de nos prédictions. On parle alors d'erreur irréductible, car, quelle que soit la qualité avec laquelle nous estimons f, nous ne pouvons pas réduire l'erreur introduite par ε.

La quantité ε peut contenir des variables non mesurées qui seraient utiles pour prédire Y : puisque nous ne les mesurons pas, f ne peut pas les utiliser pour la prédiction. La quantité ε peut aussi contenir une variabilité impossible à mesurer. Par exemple, le risque d'une réaction indésirable peut varier, pour un patient donné et un jour donné, en fonction de variations de fabrication du médicament lui-même ou de l'état général de bien-être du patient ce jour-là.

Considérons une estimation donnée $\hat{f}$ et un ensemble de prédicteurs X, ce qui conduit à la prédiction $\hat{Y} = \hat{f}(X)$. Supposons un instant que $\hat{f}$ et X soient tous deux fixés, de sorte que la seule variabilité provienne de ε. On a alors l'erreur quadratique moyenne de prédiction :

$$\mathbb{E}\left[\left(Y-\hat{Y}\right)^2\right] = \mathbb{E}\left[\left(f(X)+\varepsilon-\hat{f}(X)\right)^2\right] = \mathbb{E}\left[\left(f(X)-\hat{f}(X)\right)^2 + 2\left(f(X)-\hat{f}(X)\right)\varepsilon + \varepsilon^2\right]$$

$$= \underbrace{\mathbb{E}\left[\left(f(X)-\hat{f}(X)\right)^2\right]}_{Réductible} + \underbrace{Var(\varepsilon)}_{Irréductible}$$

L'erreur irréductible fournit toujours une limite supérieure à la précision de notre prédiction de Y. Cette limite est presque toujours inconnue en pratique.

I.3.2. Interprétation

Souvent, on cherche à comprendre l'association entre Y et les caractéristiques X_1, …, X_p. Dans ce cas, nous cherchons à estimer f, mais notre objectif n'est pas nécessairement de prédire Y. Dès lors, $\hat{f}$ ne peut pas être considérée comme une « boîte noire », car nous devons en connaître la forme exacte.

Par exemple, considérons les données `Advertising` illustrées à la Figure 1. On peut chercher à répondre à des questions telles que :

- Quels médias sont associés aux ventes ?
- Quels médias génèrent la plus forte augmentation des ventes ?
- De quelle ampleur est l'augmentation des ventes associée à une augmentation donnée de la publicité à la télévision ?

Cette situation relève du problème de l'interprétation. Un autre exemple consiste à modéliser la marque d'un produit qu'un client pourrait acheter à partir de variables telles que le prix, l'emplacement du magasin, les niveaux de remise, le prix des concurrents, etc. Dans ce contexte, on s'intéresse surtout à l'association entre chaque variable et la probabilité d'achat. Par exemple, dans quelle mesure le prix du produit est-il associé aux ventes ? Il s'agit d'un exemple de modélisation orientée vers l'interprétation.

Selon que l'objectif final est la prédiction ou l'interprétation, ou même une combinaison des deux, différentes méthodes d'estimation de f peuvent être appropriées. Par exemple, les modèles linéaires permettent une interprétation relativement simple et interprétable, mais peuvent ne pas fournir des prédictions aussi précises que certaines autres approches. À l'inverse, certaines approches fortement non linéaires, peuvent potentiellement offrir des prédictions très précises de Y, mais au prix d'un modèle moins interprétable, pour lequel l'interprétation est plus difficile.

Nous allons étudier dans ce cours deux méthodes, parmi lesquelles k-NN (k plus proches voisins). En régression ou classification, cette méthode correspond à une estimation non paramétrique et fortement non linéaire de f : on prédit Y à partir de la moyenne (ou majorité) des voisins proches. Cette méthode peut donner de très bonnes performances de prédiction si la structure est locale et si k et la métrique sont bien choisis, mais le « modèle » est peu interprétable, et l'interprétation est plus difficile.

I.4. Comment est estimée la fonction de prédiction

Il existe de nombreuses approches linéaires et non linéaires pour estimer la fonction de prédiction. Dans ce cours, nous allons étudier la méthode k-NN qui est une méthode non paramétrique. Vous voyez également l'utilisation de réseaux de neurones si vous suivez l'option SI, qui sont des modèles paramétriques.

Ces méthodes partagent généralement certaines caractéristiques et nous allons ici établir un aperçu de ces caractéristiques communes. Nous supposerons toujours que nous avons observé un ensemble de n points de données distincts. Par exemple, dans la Figure 4, nous avons observé n = 30 points de données. Ces observations sont appelées des données d'apprentissage, car nous les utiliserons pour entraîner, ou « enseigner », à notre méthode comment estimer f.

Soit $(x_i)_j$ la valeur du j-ième vecteur des caractéristiques (ou variable d'entrée) pour l'observation i, où i = 1, 2, …, n et j = 1, 2, …, p. De même, soit $y_i \in F$ la variable réponse pour la i-ième observation (F peut être un ensemble fini ou un espace vectoriel). Nos données d'apprentissage sont alors les couples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, où $x_i = \left((x_i)_1, \ldots, (x_i)_p\right)^T$.

Notre objectif est d'appliquer une méthode d'apprentissage statistique aux données d'apprentissage afin d'estimer la fonction inconnue f. Autrement dit, nous voulons trouver une fonction $\hat{f}$ telle que $Y \approx \hat{f}(X)$ pour toute observation (X, Y). De manière générale, la plupart des méthodes d'apprentissage statistique visant cet objectif peuvent être classées comme *paramétriques* ou *non paramétriques*.

I.4.1. Méthodes paramétriques

Les méthodes paramétriques reposent sur une approche fondée sur la création d'un modèle en deux étapes.

1. Tout d'abord, nous formulons une hypothèse concernant la « structure » de f. Par exemple, une hypothèse très simple consiste à supposer que f est linéaire en X (on suppose ici que $F = \mathbb{R}$) :

   $$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

   Une fois que l'on a supposé que f est linéaire, le problème d'estimation de f est grandement simplifié. Au lieu d'avoir à estimer une fonction p-dimensionnelle entièrement arbitraire f(X), il suffit d'estimer les p + 1 coefficients β_0, β_1, …, β_p.

2. Après avoir choisi un modèle, il nous faut une procédure qui utilise les données d'apprentissage pour ajuster (*fit*) ou entraîner le modèle. Dans le cas de l'ajustement du modèle linéaire précédent, nous devons estimer les paramètres β_0, β_1, …, β_p. Autrement dit, nous voulons trouver des valeurs de ces paramètres telles que :

   $$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

   L'approche la plus courante pour ajuster le modèle est la *méthode des moindres carrés*. Elle consiste à choisir les paramètres du modèle en minimisant la somme des carrés des écarts entre les valeurs observées y_i et les valeurs prédites par le modèle $\hat{y}_i$. Autrement dit, on ajuste la droite (ou l'hyperplan) qui rend ces erreurs globalement aussi petites que possible au sens quadratique. Une autre méthode, utilisée pour ajuster les paramètres des réseaux de neurones, est la *méthode par descente du gradient*. C'est une méthode d'optimisation itérative qui ajuste progressivement les paramètres d'un modèle pour diminuer une fonction de perte.

Ces méthodes sont dites paramétriques, car elle ramènent l'estimation de f à l'estimation d'un nombre fini de paramètres. En supposant une forme paramétrique pour f, on simplifie le problème : il est en général plus facile d'estimer des paramètres que d'ajuster une fonction f totalement arbitraire.

La Figure 6 montre un exemple de l'approche paramétrique appliquée aux données Revenus de la Figure 5Figure 4. Le modèle linéaire est de la forme :

$$revenu = \beta_0 + \beta_1 \cdot niveau\_etude + \beta_2 \cdot ancienneté$$

Les observations sont représentées par les points rouges et le plan en jaune est l'hyperplan résultant de l'ajustement des paramètres du modèle :

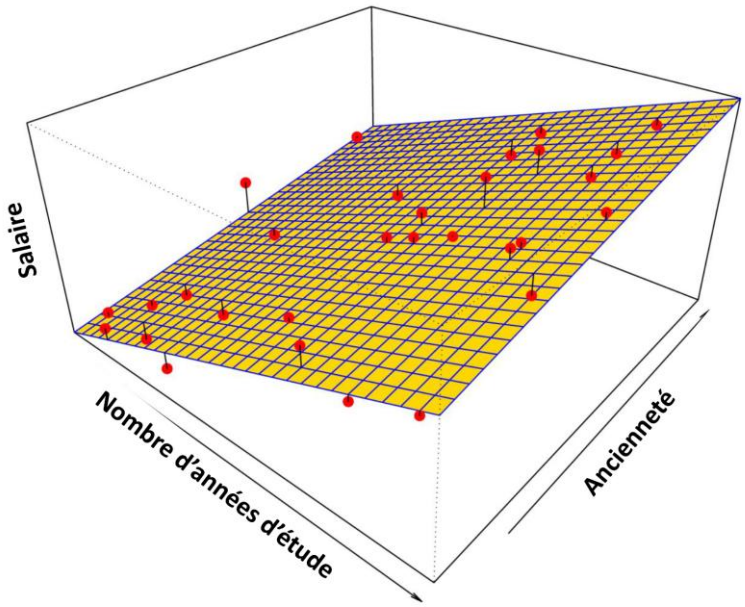


Figure 6 : Un modèle linéaire ajusté par la méthode des moindres carrés.

En comparant la Figure 5 à la Figure 6, on voit que l'ajustement linéaire présenté dans la Figure 6 n'est pas tout à fait correct : la vraie fonction f présente une certaine courbure qui n'est pas reproduite par l'ajustement linéaire. Cependant, l'ajustement linéaire semble tout de même faire un travail raisonnable pour mettre en évidence la relation positive entre le nombre d'années d'études et le revenu, ainsi que la relation un peu moins positive entre l'ancienneté et le revenu. Il est possible qu'avec un nombre d'observations aussi réduit, ce soit le mieux que l'on puisse faire.

L'inconvénient potentiel d'une approche paramétrique est que le modèle que nous choisissons ne correspond généralement pas à la forme réelle, inconnue, de f. Si le modèle retenu est trop éloigné de la vraie fonction f, alors notre estimation sera médiocre. On peut essayer de limiter ce problème en choisissant des modèles plus flexibles, capables d'ajuster de nombreuses formes fonctionnelles possibles de f (tels que les réseaux de neurones). Mais, en général, ajuster un modèle plus flexible impose d'estimer un plus grand nombre de paramètres. Ces modèles plus complexes peuvent alors conduire à un phénomène appelé *surapprentissage* (*overfitting*) : ils épousent trop fidèlement les erreurs, ou le bruit, présents dans les données.

I.4.2. Méthodes non paramétriques

Les méthodes non paramétriques ne font pas d'hypothèses explicites sur la forme fonctionnelle de f. À la place, elles cherchent une estimation de f qui se rapproche autant que possible des points de données.

Ces approches peuvent présenter un avantage majeur par rapport aux approches paramétriques : en évitant de supposer une forme particulière pour f, elles ont le potentiel d'ajuster avec précision une plus grande variété de formes possibles pour f. Toute approche paramétrique comporte le risque que la structure choisie pour estimer f soit très différente de la vraie f, auquel cas le modèle obtenu n'ajustera pas bien les données. À l'inverse, les approches non paramétriques évitent complètement ce danger, puisqu'elles ne font essentiellement aucune hypothèse sur la forme de f.

Mais les approches non paramétriques souffrent aussi d'un inconvénient important : comme elles ne ramènent pas l'estimation de f à un petit nombre de paramètres, il faut un très grand nombre d'observations (bien plus que ce qui est généralement nécessaire avec une approche paramétrique) pour obtenir une estimation précise de f.

Un exemple d'approche non paramétrique pour ajuster les données Revenus est présenté dans la Figure 7 (à gauche). Cette approche n'impose aucun modèle prédéfini à f et cherche plutôt à produire une estimation de f aussi proche que possible des données observées, tout en imposant que l'ajustement — c'est-à-dire la surface jaune de la Figure 7 — soit lisse :

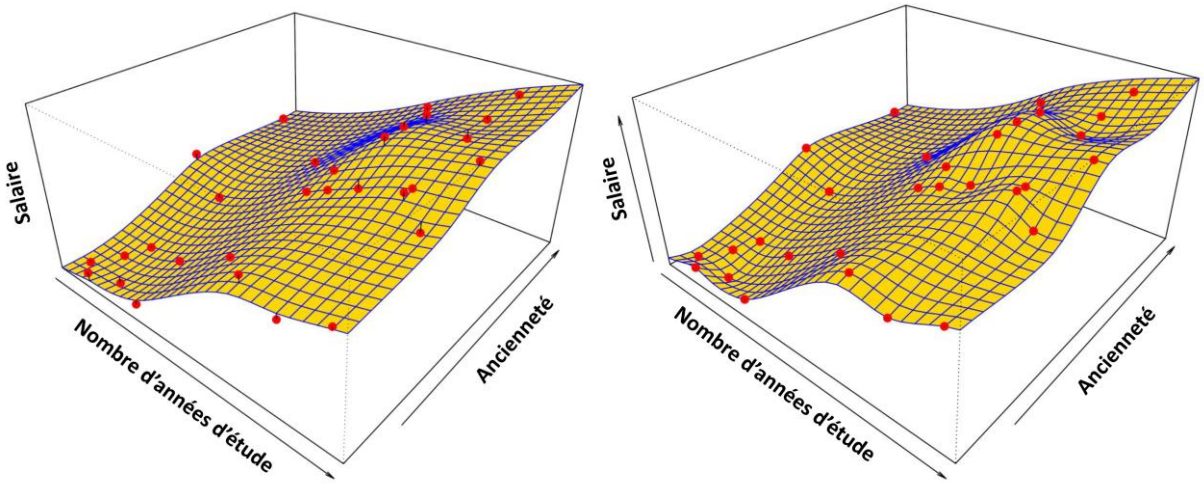


Figure 7 : Exemples de modèles non paramétriques utilisés pour estimer les données Revenus

Dans ce cas, l'ajustement non paramétrique fournit une estimation remarquablement précise de la vraie fonction f présentée dans la Figure 5.

Pour ajuster le modèle, on doit choisir un niveau de lissage. La Figure 7 montre à droite le même ajustement avec un niveau de lissage plus faible que celle de gauche, autorisant une courbe plus « rugueuse ». L'estimation obtenue ajuste alors parfaitement les données observées. Toutefois, la « surface » de la Figure 7 à droite est beaucoup plus variable que la vraie fonction f (Figure 5). C'est un exemple de surapprentissage (overfitting) des données, évoqué précédemment.

Le surapprentissage est une situation indésirable, car l'ajustement obtenu ne fournira pas des estimations fiables de la réponse sur de nouvelles observations qui ne faisaient pas partie du jeu de données d'entraînement initial.

I.5. Apprentissage supervisé et non-supervisé

On distingue classiquement deux cadres majeurs, qui structurent l'ensemble de ce cours : *l'apprentissage supervisé* et *l'apprentissage non supervisé*.

I.5.1. Apprentissage supervisé

En apprentissage supervisé, chaque observation x_i est accompagnée d'une information-cible y_i (un label, une classe, une valeur numérique). Nous cherchons à ajuster un modèle reliant la réponse aux vecteurs des caractéristiques, soit dans le but de prédire le plus précisément possible la réponse pour de nouvelles observations (prédiction), soit pour mieux comprendre la relation entre la réponse et les prédicteurs (inférence/interprétation).

Ainsi, un jeu de données supervisé est un ensemble fini de couples $\{(x_i, y_i)\}_{i=1..n}$, où chaque $x_i = \left((x_i)_1, (x_i)_2, .., (x_i)_p\right) \in \mathbb{R}^p$ décrit p caractéristiques et $y_i \in F$ décrit la réponse associée (F peut être un ensemble fini ou un espace vectoriel). Les exemples que nous avons évoqués jusqu'ici relèvent tous de l'apprentissage supervisé.

Nous aborderons dans ce cours l'apprentissage supervisé à travers la méthode k-NN.

I.5.2. Apprentissage non supervisé

L'apprentissage non supervisé correspond à une situation un peu plus délicate : pour chaque observation i = 1, …, n, on observe un vecteur de mesures x_i, mais aucune réponse associée y_i. Il n'est donc pas possible d'ajuster un modèle de régression linéaire, puisqu'il n'y a pas de variable réponse à prédire. Dans ce cadre, on travaille en quelque sorte « à l'aveugle » : on parle d'apprentissage non supervisé parce qu'il manque une variable réponse qui pourrait guider (superviser) l'analyse.

Ce qu'on cherche à comprendre ici, ce sont les relations entre les variables, ou bien entre les observations elles-mêmes. Un outil classique dans ce contexte est l'analyse de *clusters* (*clustering*). L'objectif du clustering est de déterminer, à partir de x_1, …, x_n, si les observations se répartissent en groupes relativement distincts. Par exemple, dans une étude de segmentation de marché, on peut mesurer plusieurs caractéristiques (variables) de clients potentiels, telles que le code postal, le revenu du foyer et les habitudes d'achat. On peut penser que ces clients se répartissent en groupes, par exemple de « gros » et de « petits » acheteurs. Si l'information sur les dépenses de chaque client était disponible, une analyse supervisée serait possible. Mais cette information ne l'est pas : on ne sait pas, pour chaque client, s'il appartient au groupe des gros acheteurs. Dans ce cas, on peut tenter de regrouper les clients à partir des variables mesurées afin d'identifier des groupes distincts. Mettre en évidence de tels groupes peut être intéressant, car ils peuvent différer selon une propriété d'intérêt, comme les habitudes de dépense.

La Figure 8 fournit une illustration simple du problème de clustering. On y a représenté 150 observations mesurées selon deux variables, X_1 et X_2. Chaque observation appartient à l'un des trois groupes distincts. À titre illustratif, les membres de chaque groupe ont été tracés avec des couleurs et des symboles différents. Cependant, en pratique, l'appartenance à un groupe est inconnue, et l'objectif est de déterminer à quel groupe appartient chaque observation.

Dans le panneau de gauche de la Figure 8, c'est une tâche relativement facile, car les groupes sont bien séparés. À l'inverse, le panneau de droite illustre une situation plus difficile, dans laquelle il existe un certain recouvrement entre les groupes. On ne peut alors pas attendre d'une méthode de clustering qu'elle affecte tous les points situés dans la zone de recouvrement à leur « bon » groupe (bleu, vert ou orange).

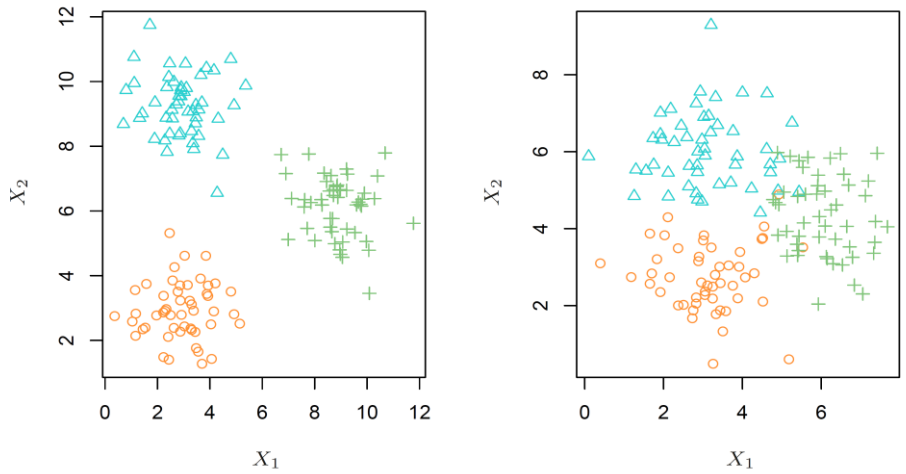


Figure 8 : Deux exemples de jeux de données de clustering comportant trois groupes

Dans les exemples présentés à la Figure 8, il n'y a que deux variables. On peut donc simplement examiner visuellement les nuages de points pour repérer des clusters. Cependant, en pratique, on rencontre souvent des jeux de données comportant bien plus que deux variables. Dans ce cas, il n'est pas facile de représenter les observations graphiquement.

Par exemple, si le jeu de données comporte p variables, on peut tracer p(p−1)/2 nuages de points distincts, et une inspection visuelle n'est tout simplement pas une méthode viable pour identifier des clusters. C'est pourquoi des méthodes de clustering automatisées sont importantes. Nous abordons dans ce cours l'apprentissage non supervisé et le clustering avec l'algorithme des k-moyennes (*k-means*).

I.6. Régression et classification

Les variables peuvent être de nature quantitative ou qualitative (aussi appelées catégorielles). Les variables quantitatives prennent des valeurs numériques. Par exemple : l'âge, la taille ou le revenu d'une personne, la valeur d'une maison, ou le prix d'une action.

À l'inverse, les variables qualitatives prennent leurs valeurs dans l'une des K classes (ou catégories) possibles. Par exemple : l'état matrimonial d'une personne (marié ou non), la marque d'un produit acheté (marque A, B ou C), le fait de faire défaut sur une dette (oui ou non), ou encore un diagnostic médical (rhume, grippe ou autre infection respiratoire).

On appelle généralement problèmes de régression ceux dont la réponse est quantitative, et problèmes de classification ceux dont la réponse est qualitative. Certaines méthodes statistiques, comme les K plus proches voisins que nous allons étudier dans ce cours, peuvent être utilisées aussi bien lorsque la réponse est quantitative que lorsqu'elle est qualitative.

Plus formellement, considérons un jeu de données supervisé constitué de couples $\{(x_i, y_i)\}_{i=1..n}$, où chaque $x_i = \big((x_i)_1, (x_i)_2, .., (x_i)_p\big) \in \mathbb{R}^p$ décrit p caractéristiques et $y_i \in F$ décrit la réponse associée. Deux types de problèmes se posent :

- Si F est un ensemble fini, on parle de problème de classification. Dans ce cas, on peut supposer que F est de la forme $F = \{0, 1, \ldots, \ c - 1\}$, où c est le nombre de classes souhaitées. Dans le cas particulier où c = 2, on a F = {0, 1}, et l'on parle de *classification binaire*. La reconnaissance optique de caractères (ROC) (*optical character recognition, OCR*), la reconnaissance de fleurs sont des problèmes de classification. Le filtrage des spams est quant à lui un problème de classification binaire.
- Si F est un espace vectoriel, on parle de problème de régression. La prédiction des recettes d'un film, de l'âge d'un internaute ou du nombre de clics sur un lien de site internet sont des problèmes de régression.

II) PERFORMANCES D'UN MODÈLE

Pour évaluer la performance d'un modèle sur un jeu de données, il faut disposer d'un moyen de mesurer à quel point ses prédictions correspondent réellement aux données observées. Autrement dit, il faut quantifier dans quelle mesure la réponse prédite pour une observation est proche de la réponse vraie associée à cette observation.

II.1. Fonction de perte et risque empirique

Pour savoir si un modèle est concluant, on définit une *fonction de perte* $\ell: F \times F \to \mathbb{R}^+$.

Pour un problème de classification, on peut poser :

$$\ell(y, \hat{y}_i) = \mathbb{I}(y \neq \hat{y}_i) \ = \ 1 - \delta_{y,\hat{y}_i} \qquad , \hat{y}_i = \hat{f}(x_i)$$

Pour un problème de régression :

$$\ell(y, \hat{y}_i) = \|y - \hat{y}_i\|^2$$

Le prédicteur f doit alors minimiser le risque théorique :

$$R\big(\hat{f}\big) = E\big(\ell(y, \hat{y}_i)\big)$$

… l'espérance étant prise selon la loi (inconnue) du couple (x, y).

Comme cette loi est inconnue, sous l'hypothèse que les données sont des tirages indépendants et identiquement distribués (i.i.d.) et que la perte admet une moyenne finie, la moyenne empirique de la perte est une bonne approximation de son espérance quand n est grand. On l'approche donc par le *risque empirique* (moyenne sur un échantillon de n données) :

$$\hat{R}_n\big(\hat{f}\big) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y}_i)$$

L'apprentissage par minimisation du risque empirique consiste alors à choisir $\hat{f}$ parmi une famille de fonctions $\mathcal{F}$ (une « classe de modèles ») de manière à rendre $\hat{R}_n\big(\hat{f}\big)$ aussi petit que possible.

II.2. Phénomène de sur-apprentissage

La fonction $\hat{f}$ n'est pas fixée à l'avance : elle est choisie dans une famille de fonctions $\mathcal{F}$ en regardant les données au fur et à mesure de l'apprentissage. Or, plus la classe $\mathcal{F}$ est riche (beaucoup de formes possibles), plus il devient facile de trouver une fonction qui « colle » très bien aux données d'apprentissage, y compris aux erreurs de mesure et au bruit. C'est ce qu'on appelle le *sur-apprentissage* (*overfitting*) : la règle apprend des détails accidentels du jeu d'apprentissage au lieu d'apprendre la tendance générale.

En conséquence :

- Si $\mathcal{F}$ est très flexible, on peut obtenir un risque empirique très petit (erreur d'entraînement faible), mais la performance sur des nouvelles données peut être mauvaise (erreur de test élevée), car le modèle a « mémorisé » du bruit.
- Si $\mathcal{F}$ est trop rigide, la règle ne peut pas s'adapter : elle commet une erreur d'entraînement déjà élevée et généralise mal aussi (sous-ajustement).

Pour illustrer ce phénomène, utilisons un jeu de données construit à partir d'une sinusoïdale parfaite et ajoutons-y aléatoirement du bruit à la sortie (ici sur P = 21 points) :

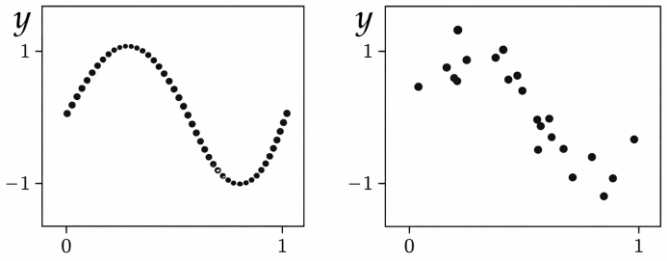


Figure 9 : Sinusoïde originale (gauche) et jeu de données bruité (droite)

La Figure 10 illustre l'ajustement non linéaire entièrement optimisé de deux modèles appliqués à ces données, utilisant respectivement un modèle polynomial (ligne du haut) et un réseau de neurones (ligne du bas).

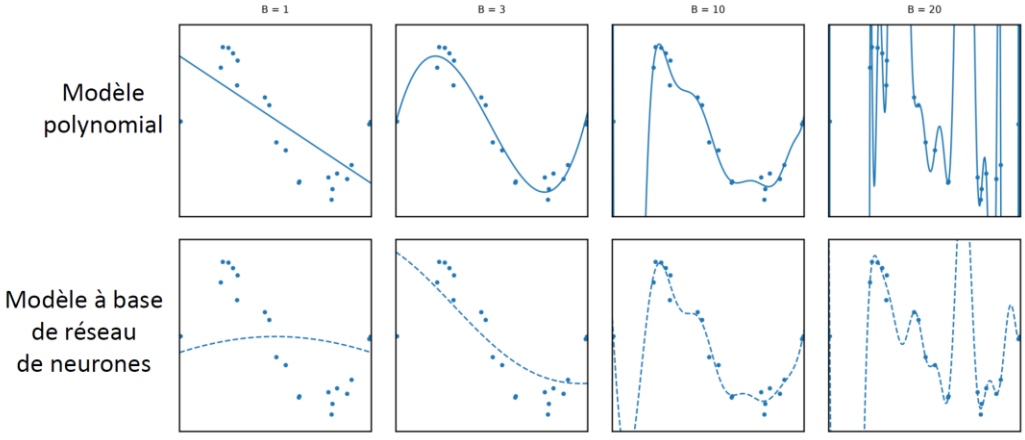


Figure 10 : Ajustement des différents modèles

On remarque que les deux modèles sous-ajustent les données lorsqu'on n'utilise qu'un seul paramètre B = 1 (colonne de gauche). Ce sous-ajustement est une conséquence directe de l'utilisation de modèles de faible flexibilité, qui produisent des ajustements trop simples pour représenter correctement la structure sous-jacente des données qu'ils cherchent à approximer.

On constate aussi que chaque modèle s'améliore lorsque l'on augmente B, mais seulement jusqu'à un certain point : au-delà, chaque modèle ajusté devient trop complexe, commence à paraître excessivement irrégulier, et ressemble peu au phénomène sinusoïdal qui a initialement généré les données.

De tels modèles sur-ajustés, bien qu'ils représentent très bien les données actuelles, seront de mauvais prédicteurs pour de nouvelles données générées par le même processus.

II.3. Performances d'un modèle en régression

Dans le cadre de la régression, la mesure la plus couramment utilisée est *l'erreur quadratique moyenne* (*MSE, mean squared error*), qui utilise le carré de la norme euclidienne comme fonction de perte et définie par :

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

Le MSE est faible lorsque les réponses prédites sont très proches des réponses vraies, et il est élevé lorsque, pour certaines observations, les réponses prédites et les réponses vraies diffèrent fortement.

Après avoir défini la MSE, on introduit souvent la RMSE (*Root Mean Squared Error*), définie comme la racine carrée de la MSE. L'intérêt principal de cette transformation est d'obtenir une mesure d'erreur dans la même unité que la variable réponse Y : alors que la MSE s'exprime en « unités au carré », la RMSE se lit directement comme une erreur typique (par exemple en euros, en mètres, etc.), ce qui facilite l'interprétation et la comparaison de modèles sur une même tâche.

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

Un inconvénient de l'utilisation de l'erreur quadratique (que l'on minimise pour obtenir les paramètres optimaux d'un modèle paramétrique) est que le fait de mettre l'erreur au carré augmente l'importance des grandes erreurs. En particulier, pour des erreurs de norme supérieure à 1, la mise au carré amplifie fortement ces valeurs. Cela oblige les coefficients appris à produire un ajustement qui cherche surtout à réduire ces grandes erreurs, parfois dues à des valeurs aberrantes (*outliers*) dans le jeu de données. Cela tend à produire des modèles qui sur-ajustent (*overfit*) les valeurs aberrantes.

Notons que la RMSE ne rend pas la mesure plus robuste aux valeurs aberrantes : comme elle repose toujours sur des écarts au carré, les grandes erreurs restent fortement pénalisées. Minimiser la RMSE ou la MSE revient au même, car la racine carrée est une fonction croissante.

La Figure 11 illustre ce phénomène : le jeu de données peut dans l'ensemble être correctement décrit par un modèle linéaire, à l'exception d'une seule valeur aberrante. Les paramètres d'un modèle linéaire sont ajustés en utilisant l'erreur MSE. L'ajustement ainsi

obtenu ne représente pas bien la majorité des points : il se courbe nettement vers le haut dans le but de réduire la grande erreur quadratique associée au point aberrant isolé.

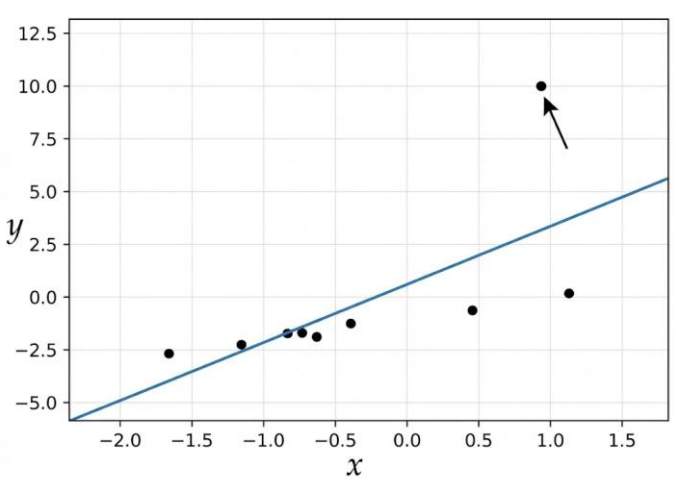


Figure 11 : Ajustement d'un modèle linéaire sur le jeu de données avec une valeur aberrante

Pour minimiser l'impact des valeurs aberrantes, une alternative consiste alors à remplacer l'erreur quadratique par une erreur absolue. La moyenne de ces écarts absolus sur un ensemble de données définit la *déviation absolue moyenne* (*MAD – Mean Absolute Deviation*) ou encore l'erreur absolue relative moyenne (*MARE – Mean Absolute Relative Error*) :

$$MAD = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad MARE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

La Figure 12 présente le résultat de l'ajustement d'un modèle linéaire en minimisant, d'une part, le critère des moindres carrés et, d'autre part, le critère des écarts absolus, en utilisant le jeu de données de la Figure 11. Le critère des moindres carrés est tracé en pointillés, et le critère des écarts absolus en trait plein. En examinant ces courbes, on constate que l'ajustement en utilisant les écarts absolus est moins influencé par la valeur aberrante et suit mieux la majorité des points. Cela constitue, à lui seul, un indice que le premier conduit à un ajustement sensiblement meilleur que les moindres carrés.

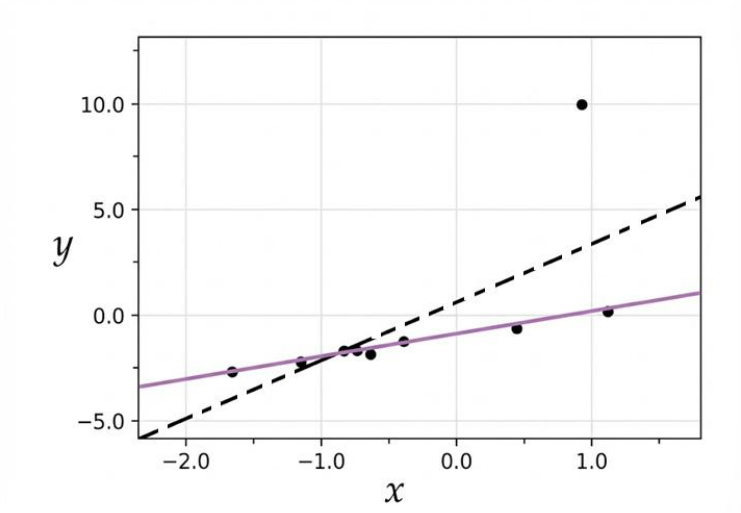


Figure 12 : Impact des deux mesures sur l'ajustement des paramètres du modèle

La MAD est surtout meilleure pour gérer les valeurs aberrantes et la MSE reste très utilisée pour plusieurs raisons :

- Ses propriétés mathématiques font qu'elle est plus simple à optimiser que la MAD et conduit à des calculs rapides ;
- Minimiser la MSE correspond à un choix naturel si l'on suppose des erreurs « gaussiennes » (bruit à peu près normal) ;
- Pénaliser les grandes erreurs n'est pas toujours souhaitable car si les grosses erreurs coûtent très cher (sécurité, contrôle, finance), on préfère parfois un critère qui « punit » davantage les écarts importants ;
- Dans beaucoup de modèles (réseaux de neurones notamment), la perte quadratique est compatible avec des méthodes d'optimisation simples et stables. La perte absolue peut fonctionner, mais elle peut rendre l'entraînement moins stable ou plus lent selon les cas.

II.4. Performances d'un modèle en classification

II.4.1. Taux de réussite

Dans le cas d'un problème de classification, on effectue en général une stratification des données : c'est une partition particulière dans laquelle la proportion de données dans chaque classe est la même que dans l'échantillon initial.

L'approche la plus courante pour quantifier la précision de la fonction de prédiction $\hat{f}$ est alors de mesurer son taux de réussite (*accuracy*), notée Acc. Il s'agit du pourcentage d'exemples du jeu de données d'apprentissage dont le label est correctement prédit par le modèle, c'est-à-dire :

$$Acc = \frac{1}{n}\sum_{i=1}^{n} \delta_{y_i,\hat{y}_i}$$

Le taux de réussite varie de 0 (aucun point n'est correctement classé) à 1 (tous les points sont correctement classés).

II.4.2. Taux de réussite équilibré en classification binaire

Le taux de réussite est une mesure de base très pertinente pour évaluer la performance d'un classifieur entraîné. Cependant, dans certains scénarios, elle peut donner une image incomplète de la qualité réelle de la résolution du problème de classification.

Par exemple, lorsque les classes d'un jeu de données sont fortement déséquilibrées — c'est-à-dire lorsqu'une classe contient beaucoup plus d'exemples que l'autre — le taux de réussite perd une grande partie de sa valeur comme métrique de qualité. En effet, si une classe domine largement, un taux de réussite proche de 1 peut être trompeur. Supposons par exemple qu'une classe représente 95 % des données : un classifieur naïf qui attribue systématiquement l'étiquette de la classe majoritaire à tous les points obtient un taux de réussite de 95 %. Pourtant, dans ce cas, « se tromper sur 5 % » revient à mal classer entièrement l'autre classe. Une manière simple d'améliorer la métrique du taux de réussite est de calculer le taux de réussite (taux de bonne classification) sur chaque classe séparément, puis de faire la moyenne.

Considérons le cas d'une classification binaire où F = {0, 1}. Si l'on note Ω_0 l'ensemble des indices des points dont le label vaut $y_i = 0$, et Ω_1 l'ensemble des indices des points dont le label vaut $y_i = 1$, on peut alors calculer, séparément pour chaque classe, son taux de réussite :

$$Acc_0 = \frac{1}{|\Omega_0|} \sum_{i \in \Omega_0} \delta_{y_i, \hat{y}_i} \qquad Acc_1 = \frac{1}{|\Omega_1|} \sum_{i \in \Omega_1} \delta_{y_i, \hat{y}_i}$$

… où $|\Omega_0|$ et $|\Omega_1|$ désignent respectivement le nombre de points appartenant aux classes 0 et 1. On peut ensuite regrouper ces deux mesures en une seule valeur en prenant la moyenne. Cette métrique combinée s'appelle le taux de réussite équilibré :

$$Acc\_equ = \frac{Acc_0 + Acc_1}{2}$$

Le taux de réussite équilibré varie entre 0 et 1. Lorsqu'elle vaut 0, aucun point n'est correctement classé ; lorsque les deux classes sont classées parfaitement, on a Acc_equ = 1. Les valeurs intermédiaires entre 0 et 1 mesurent à quel point, en moyenne, chacune des deux classes est bien classée prise séparément. Par exemple, si une classe est entièrement bien classée et l'autre entièrement mal classée (comme dans le scénario imaginaire d'un jeu de données très déséquilibré, avec 95 % de points dans une classe et 5 % dans l'autre, et où l'on prédit systématiquement la classe majoritaire), alors on aura Acc_equ = 0,5.

II.4.3. Taux de réussite équilibré en classification multi-classes

On considère une classification multi-classes avec K classes c_1, c_2, …, c_K, un ensemble de test $\{(x_i, y_i)\}_{i=1..n}$ et des prédictions $\hat{y}_i$. Pour chaque classe, on note :

$$\Omega_k = \{i \in \{1, \ldots, n\} : y_i = c_k\}$$

… l'ensemble des indices des observations dont la vraie classe est c_k. Le taux de réussite « par classe » est :

$$Acc_k = \frac{1}{|\Omega_k|} \sum_{i \in \Omega_k} \delta_{y_i, \hat{y}_i}$$

Le taux de réussite équilibré est alors la moyenne de ces taux de réussite par classe :

$$Acc\_equ = \frac{1}{K} \sum_{k=1}^{K} Acc_k$$

II.4.4. Matrice de confusion en classification binaire

D'autres métriques permettant d'évaluer la qualité d'un modèle entraîné pour une classification binaire peuvent être construites à partir de la matrice de confusion. Une matrice de confusion est un tableau récapitulatif dans lequel les résultats de classification sont ventilés selon la classe réelle (lignes) et la classe prédite (colonnes).

On note ici A le nombre de points dont le label réel 1 coïncide avec le label attribué par le classifieur entraîné. L'autre entrée diagonale D est définie de manière analogue : c'est le nombre de points dont le label prédit 0 est égal au label réel. Les entrées hors diagonale, notées B et C, représentent les deux types d'erreurs de classification, lorsque le label réel et

le label prédit ne coïncident pas. En pratique, on souhaite que ces deux valeurs soient aussi petites que possible :

Label réel	Label prédit 1	Label prédit 0
1	A	B
0	C	D

=

vrais positifs	faux négatifs
faux positifs	vrais négatifs

Figure 13: Matrice de confusion

Le taux de réussite peut s'exprimer en fonction des quantités de la matrice de confusion représentées sur la Figure 13, sous la forme :

$$Acc = \frac{A + D}{A + B + C + D}$$

… et, de même, le taux de réussite calculé séparément sur chacune des deux classes s'écrit :

$$Acc_1(sensibilité) = \frac{A}{\mathrm{A} + \mathrm{B}} \qquad Acc_0(spécificité) = \frac{D}{\mathrm{C} + \mathrm{D}}$$

Dans le jargon de l'apprentissage automatique, ces métriques calculées séparément sur chaque classe sont souvent appelées, respectivement, la *sensibilité* (taux de vrais positifs – Acc_1) et la *spécificité* (taux de vrais négatifs – Acc_0). *Le taux de réussite équilibré* peut, de même, s'exprimer comme :

$$Acc\_equ = \frac{1}{2}\frac{A}{A + B} + \frac{1}{2}\frac{D}{C + D} \ (moyenne\ de\ la\ Sensibilité\ et\ de\ la\ Spécificité)$$

On peut également définir la précision, qui est la proportion de vrais positifs parmi ceux qui sont déclarés comme positifs :

$$précision = \frac{A}{A + C}$$

Le F1-score est alors défini comme la moyenne harmonique de la précision et de la sensibilité :

$$F1 - score\ = \frac{2}{\frac{1}{precision} + \frac{1}{sensibilité}} = \frac{2A}{2A + B + C}$$

Le F1-score n'est élevé que si la précision et la sensibilité sont toutes les deux élevées ; il pénalise fortement un déséquilibre (par exemple précision très bonne mais sensibilité faible, ou l'inverse).

Dans l'exemple de la figure 14, une classification binaire sur un jeu de données classique de détection de spam est réalisée. Ce jeu de données contient P = 4601 exemples : 1813 e-mails « spam » et 2788 e-mails « non-spam ».

réel \ prédictions	non-spam	spam
non-spam	2664	124
spam	191	1622

Figure 14 : Matrice de confusion binaire

Le taux de réussite est de $\frac{2664+1622}{4601} = 93{,}2\%$. Le taux de bonne classification « non-spam » est de $\frac{2664}{2664+124} = 95{,}6\%$ et le taux de bonne classification « spam » est de $\frac{1622}{191+1622} = 89{,}5\%$. Le taux de réussite équilibré est donc la moyenne de ces deux taux et vaut 92,5%.

II.4.5. Matrice de confusion en classification multi-classes

Dans le cas de la classification, une fois l'algorithme appliqué aux données, on peut représenter la matrice de confusion multi-classes associée aux jeux de données d'entraînement et de test pour évaluer le modèle.

C'est un élément M de $\mathcal{M}_c(\mathbb{N})$ ainsi défini : pour tout $(i,j) \in [\![0, c-1]\!]^2$, m_{ij} est le nombre de données de test qui ont réellement i pour étiquette et auxquelles l'algorithme d'apprentissage statistique entraîné sur les données d'entraînement a attribué l'étiquette j.

Plus la matrice de confusion est proche d'une matrice diagonale, meilleure est la qualité de la prédiction de l'algorithme.

```
def matrice_confusion(y_true, y_pred, labels=None):
    """
    Renvoie la matrice de confusion M telle que
    M[i, j] = nombre d'exemples dont la vraie classe est labels[i]
              et la classe prédite est labels[j].
    """
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)

    if labels is None:
        labels = np.unique(np.concatenate([y_true, y_pred]))
    else:
        labels = np.asarray(labels)

    dico = {labels[j]: j for j in range(len(labels))}
    mat = np.zeros((len(labels), len(labels)), dtype=int)

    for i in range(len(y_true)):
        mat[dico[y_true[i]], dico[y_pred[i]]] += 1
    return mat, labels
```

Exemple :

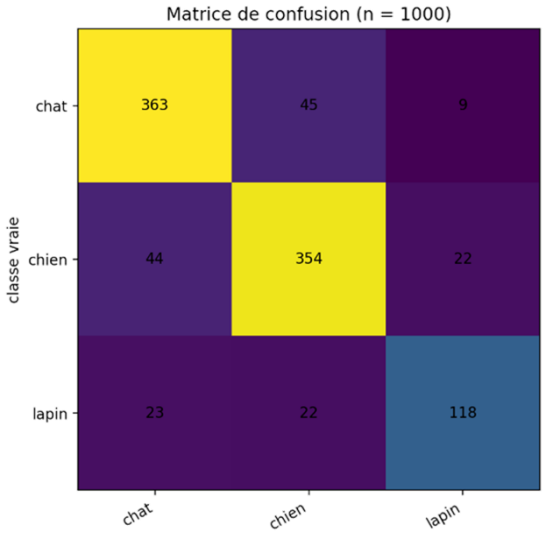
```
y_true = ["chat","chien","chien","chat","lapin"]
y_pred = ["chat","chat","chien","chat","lapin"]
M, labels = matrice_confusion(y_true, y_pred)
print(labels)
print(M)
```

```
⇨ ['chat' 'chien' 'lapin']
  [[2 0 0]
   [1 1 0]
   [0 0 1]]
   0.8 # Taux de réussite
```

Le quotient $\frac{tr(M)}{\sum_{i,j} M_{i,j}}$ donne le taux de réussite à partir de la matrice de confusion M : c'est la proportion de données dont les étiquettes ont été correctement prédites.

```
def TAUX_REUSSITE (M):
    return np.trace(M)/np.sum(M)
```

Voici un autre exemple avec un jeu de données de 1000 valeurs, où la matrice de confusion obtenue est la suivante :



Le taux de réussite est de :

$$Acc = \frac{tr(M)}{\|M\|_1} = \frac{tr(M)}{\sum_{i,j} M_{i,j}} = \frac{363 + 354 + 118}{1000} = 0{,}835\ (83{,}5\%)$$

II.5. Jeu de données pour les tests

En général, ce qui nous intéresse, c'est la précision des prédictions obtenues lorsque l'on applique le modèle à des données de test, c'est-à-dire à des données nouvelles, jamais vues pendant l'apprentissage.

En effet, supposons que l'on veuille développer un algorithme pour prédire le prix d'une action à partir de ses rendements passés. On peut entraîner la méthode à l'aide des rendements des six derniers mois. Mais on ne cherche pas vraiment à savoir si la méthode prédit correctement le prix de la semaine dernière ; on veut plutôt qu'elle prédise correctement le prix de demain ou du mois prochain.

De même, supposons que l'on dispose, pour un certain nombre de patients, de mesures cliniques (par exemple poids, pression artérielle, taille, âge, antécédents familiaux), ainsi que d'une information indiquant si chaque patient est diabétique. On peut utiliser ces patients pour entraîner une méthode d'apprentissage statistique afin de prédire le risque de diabète à partir de ces mesures. En pratique, on veut que la méthode prévoie correctement le risque de diabète pour de futurs patients, à partir de leurs mesures cliniques. On s'intéresse moins à la précision sur les patients ayant servi à l'apprentissage, puisque l'on connaît déjà leur état.

Ce qui nous intéresse vraiment n'est donc pas de savoir si $\hat{f}(x_i) \approx y_i$; nous voulons plutôt savoir si $\hat{f}(x_0) \approx y_0$, où (x_0, y_0) est une observation de test, nouvelle, qui n'a pas été utilisée pour entraîner la méthode d'apprentissage statistique. Si nous disposions d'un grand nombre d'observations de test, nous pourrions calculer :

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_{i0} - \hat{f}(x_{i0})\right)^2$$

… soit l'erreur quadratique moyenne de prédiction sur ces observations de test (x_{i0}, y_{i0}).

Pour tester le modèle et avoir une idée du taux de bons résultats qu'il renvoie, on effectue alors une partition des données en un jeu d'entraînement et un jeu de test. Le jeu d'entraînement représente généralement 70% à 90% des données.

II.6. Normalisation des données

Quand les données font intervenir des attributs d'ordres de grandeur très différents, il peut être utile d'effectuer une normalisation affine des données avant de leur appliquer un algorithme d'intelligence artificielle.

La normalisation affine des données consiste à appliquer, pour chaque caractéristique X_i, une transformation du type $X'_i = a_i X_i + b_i$ afin de rendre les variables comparables en échelle. Une forme très utilisée est la normalisation min–max, qui envoie chaque caractéristique dans [0,1] :

$$x'_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$$

C'est particulièrement important pour les méthodes fondées sur des distances (k-NN, k-moyennes) : si une caractéristique a une échelle beaucoup plus grande qu'une autre, elle domine artificiellement la distance euclidienne, et donc la décision (voisinage) ou la formation des clusters. La normalisation remet les variables sur une échelle comparable, ce qui rend l'usage de la distance plus pertinent.

La Figure 15 illustre l'effet de la normalisation des données sur la caractéristique 1, initialement dans [10, 90] et envoyé dans [0,1] après normalisation :

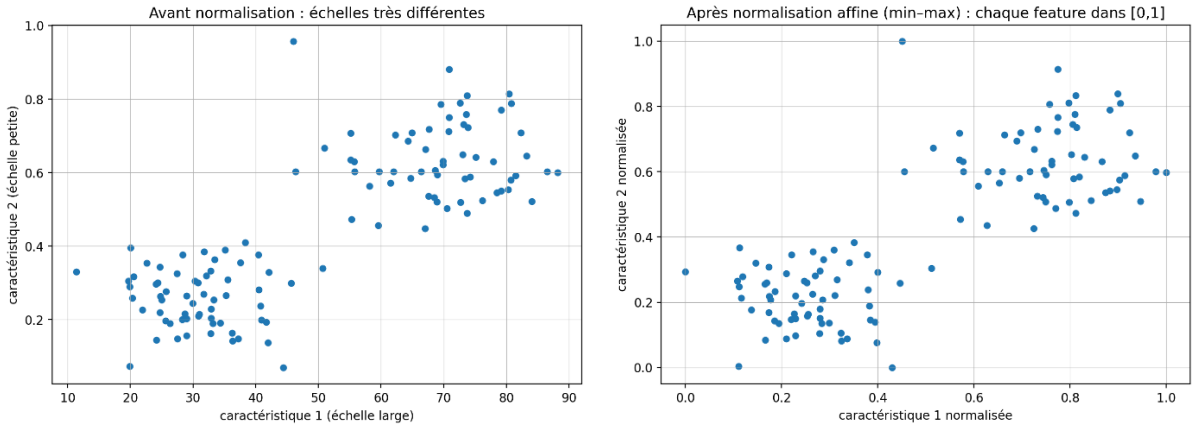


Figure 15 : Normalisation min-max des données

Pour cela on peut utiliser l'algorithme qui suit et définir un nouveau tableau normalisé :

```
def NORMALISATION_AFFINE(x):
    m,M = min(x),max(x)
    return (x-m)/(M-m)
```