1_ Intelligence Artificielle - Introduction

Compétences attendues :

- ✓ Analyser les principes d'intelligence artificielle. ⇆ *I*
- ✓ Choisir une démarche de résolution d'un problème d'ingénierie numérique ou d'intelligence artificielle. ⇆ ***I***
- ✓ Résoudre un problème en utilisant une solution d'intelligence artificielle. ⇆ ***I***

1. L'intelligence Artificielle, Qu'est-ce que c'est, à quoi ça sert ?

1.1. Définitions

Définition de l'Intelligence Artificielle :

L'intelligence artificielle (IA) est « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence ».

En première approche, l'IA peut être assimilée à un ensemble d'algorithmes permettant de réaliser des tris, des classifications…

Les domaines d'applications de l'IA sont de plus en plus nombreux, avec certains très controversés. Parmi les plus courants, on trouve notamment :

- La finance, avec une aide à la décision quant à l'acceptation éventuelle d'un prêt, la gestion d'un fond d'investissement
- La médecine, avec une aide au diagnostic de maladie et de préconisations médicamenteuses
- Le militaire, avec des systèmes d'aide à la décision et de commandement
- La robotique, avec des systèmes d'aide à la décision
- Le contrôle d'accès, avec des systèmes de reconnaissance faciale par exemple
- Le journalisme, avec des systèmes de reconnaissances de *fakenews*
- Les réseaux sociaux, avec des systèmes de suivi de la popularité d'une personne, ou l'identification de diffusion de message à caractère haineux
- Les messageries *mail* avec des propositions de réponses automatiques
- La reconnaissance de caractères ou la traduction

Exemple : Identification d'objets, en temps réel, dans une vidéo.

On remarquera sur l'exemple de gauche que l'algorithme parvient à identifier sur l'image des zones puis à identifier à quoi elles correspondent. On remarquera aussi que la flamme est identifiée comme étant un donut...

Les problèmes types de l'IA :

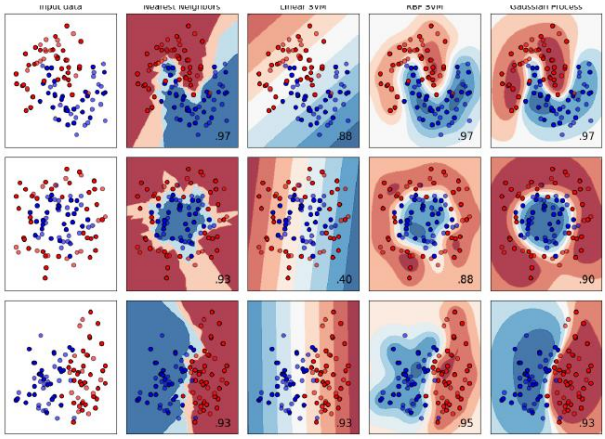
Les thèmes d'utilisation de l'IA (ou plutôt du M*achine Learning*) sont :

- **La *classification*** (idem en anglais) **(a)** : Il s'agit d'identifier à quelle catégorie un objet appartient (apprentissage supervisé).
  Applications : Détection de spam, reconnaissance d'images.

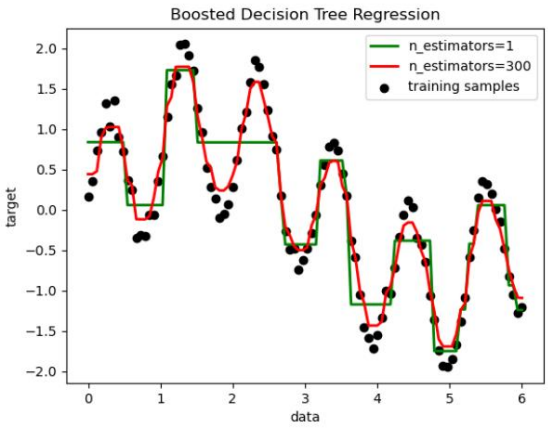
- **La *régression*** (idem en anglais) (b) : Le but est de prédire un paramètre à valeur continue associée à un objet.
  Applications : Les réponses aux médicaments, le cours de la bourse.

- **La *segmentation*** *(*ou *clustering)* **(c)** *:* Regrouper de manière automatique des objets dans des ensembles (apprentissage non supervisé).
  Applications : Segmentation marketing, regroupement des résultats des expériences.

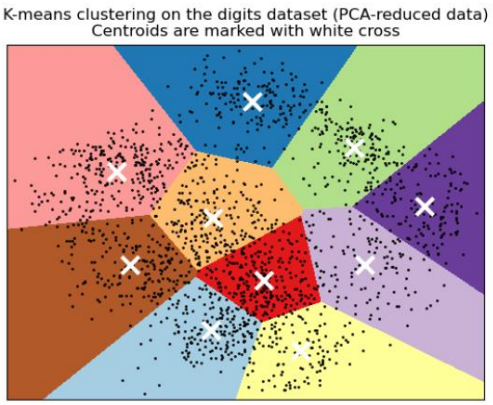
- **La *prédiction*** (idem en anglais) **(d)** : Prédire des données temporelles.
  Applications : Reconnaissance vocale, la météo ou encore la correction d'un système.



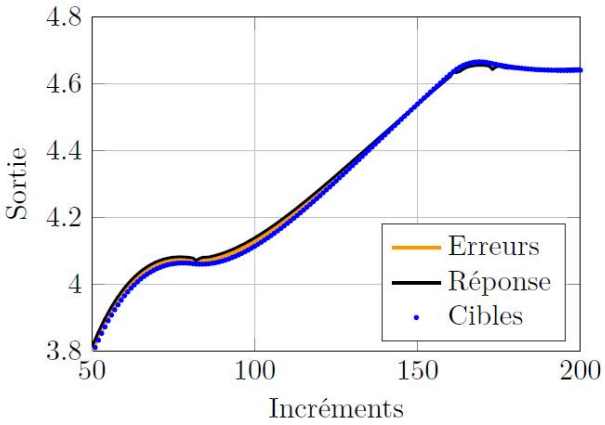
(a) Classification



(b) Régression



(c) Clustering



(d) Prédiction

Pour utiliser un algorithme d'IA il est nécessaire de disposer de données... en général beaucoup de données.

Définition des données :

Les données utilisées par un algorithme d'IA peuvent être sous différentes formes :

- Des données quantitatives continues qui peuvent prendre n'importe quelles valeurs dans un ensemble de valeurs (par exemple la température).
- Des données quantitatives discrètes qui peuvent prendre un nombre limité de valeurs dans un ensemble de valeurs (par exemple nombre de pièces d'un logement).
- Des données qualitatives (ordinales ou nominales suivant qu'on puisse les classer ou non, par exemple des couleurs, des notes à un test d'opinion, ...).

Pour pouvoir traiter ces données, il peut être nécessaire qu'elles soient organisées sous une certaine forme. On peut par exemple identifier :

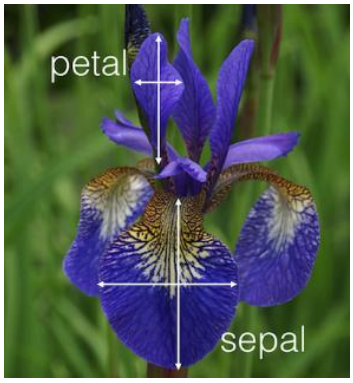
- Les données structurées (dans une base de données).
- Les données semi-structurées (dans un fichier .csv ou xml).
- Les données non structurées (image, texte ou vidéo).

Définition des observations :

On appelle observation (ou individu ou objet) une « ligne de donnée » qui va être utilisée par un algorithme d'apprentissage. Une observation est composée de caractéristiques qui varient pour chacun des observations.

Exemple :

Dans le cas d'une base de données regroupant des photos d'Iris, les données sont composées de 150 observations, chacune composée de 4 caractéristiques : longueur et largeur du sépale ainsi que longueur et largeur du pétale.



Définition de Big Data :

Mégadonnées ou données massives. Le big data désigne les ressources d'informations dont les caractéristiques en termes de volume, de vélocité et de variété imposent l'utilisation de technologies et de méthodes analytiques particulières pour générer de la valeur, et qui dépassent en général les capacités d'une seule et unique machine et nécessitent des traitements parallélisés.

Que fait-on avec ces données ?

En général les algorithmes vont chercher un lien entre ces données. Dans le cas où il existe un *lien connu* par le *Data Scientist* on parle *d'apprentissage supervisé*. Si ce *lien n'est pas connu*, on parle d'*apprentissage non supervisé* ou de *clustering*.

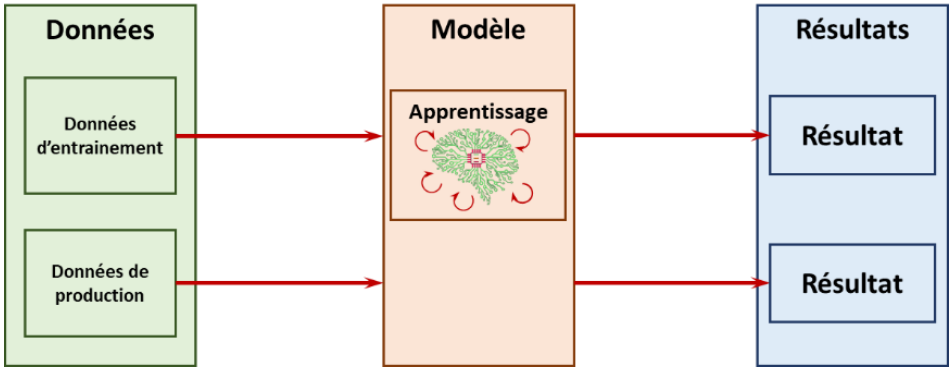
1.2. Apprentissage Automatisé – Machine Learning

Définition de l'apprentissage automatique (Machine Learning) :

L'apprentissage automatique est un champ de l'intelligence artificielle dont l'objectif est d'analyser un grand volume de données afin de déterminer des motifs et de réaliser un modèle prédictif.

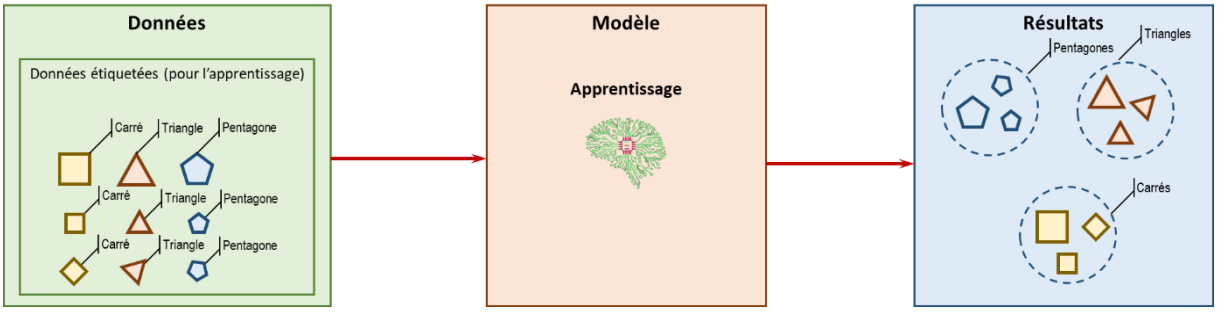
L'apprentissage comprend deux phases :

- L'entraînement (ou apprentissage) est une phase d'estimation du modèle à partir de données d'observations.
- La mise en production du modèle (inférence) est une phase pendant laquelle de nouvelles données sont traitées dans le but d'obtenir le résultat souhaité.



Définition de l'apprentissage supervisé - Apprentissage :

Tâche d'apprentissage au cours de laquelle l'algorithme (ou fonction de prédiction) va, à partir d'un ensemble de données **étiquetées**, déterminer un lien entre un ensemble les données et les étiquettes.



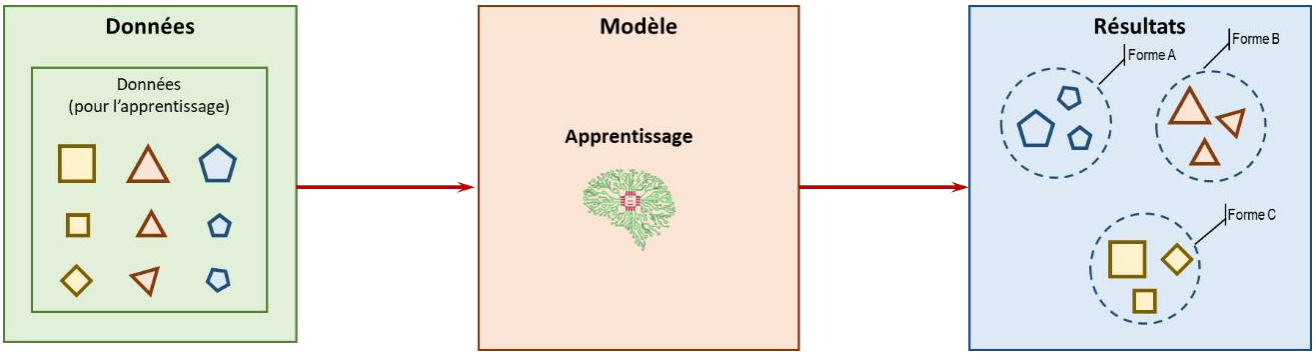
Définition de l'apprentissage supervisé - Inférence :

Tâche au cours duquel l'algorithme (ou fonction de prédiction) va, à partir d'un ensemble de données **non étiquetées**, prédire l'étiquette.

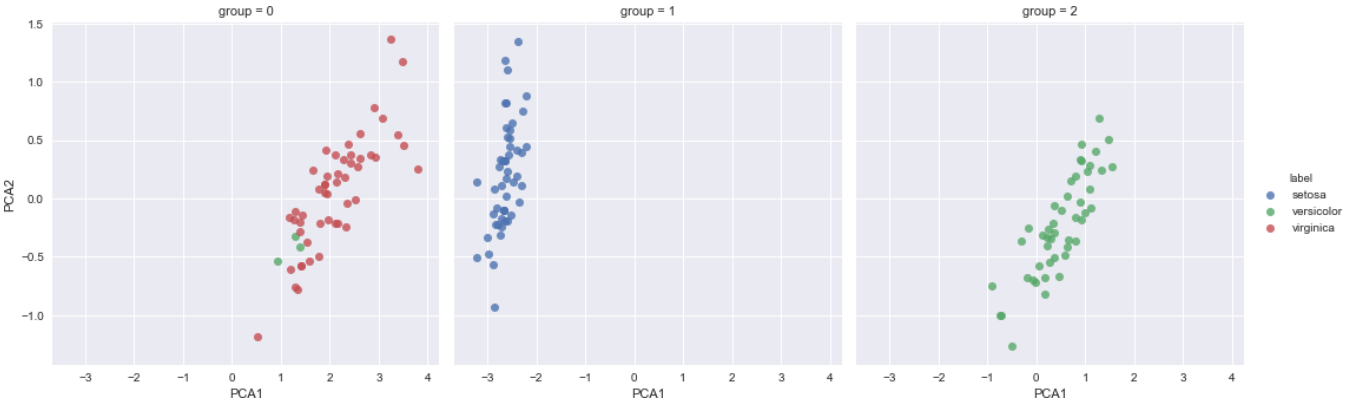
Exemple : Soit un ensemble d'images en noir et blanc représentant des chiffres de 0 à 9. L'algorithme doit dans un premier temps *apprendre* le lien entre l'image et le chiffre. Dans un second temps, l'algorithme devra déterminer le chiffre en fonction d'une image seule.

Définition de l'apprentissage non supervisé – Clustering :

Tâche d'apprentissage au cours duquel l'algorithme (ou fonction de prédiction) va, à partir d'un ensemble de données **non étiquetées**, déterminer un lien entre les données (et les regrouper).



Classification d'Iris en utilisant un algorithme d'apprentissage non supervisé :

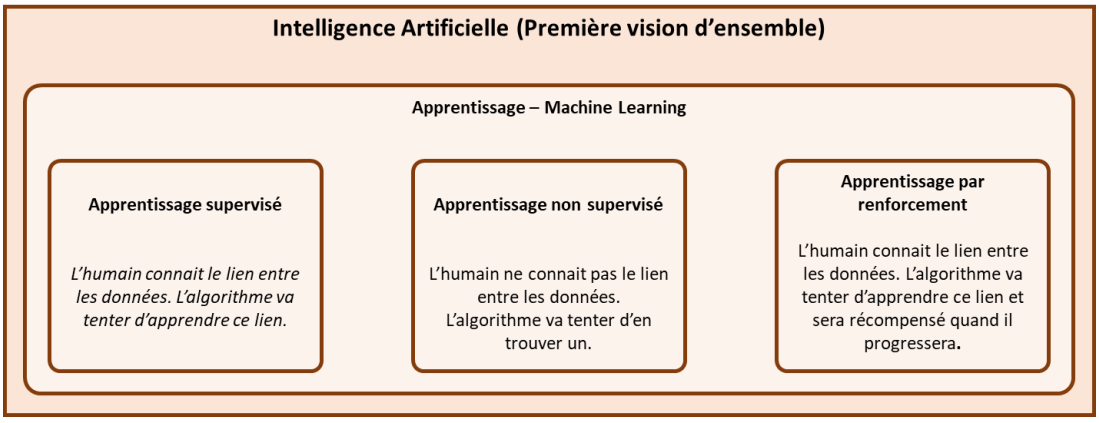


On remarque que sur 150 images d'Iris, uniquement avec les mesures des pétales, l'algorithme a réussi à retrouver les 3 différentes espèces, sans les connaître, à 3 erreurs près.

Définition de l'apprentissage par renforcement :

Si au cours de l'apprentissage supervisé, un mécanisme de récompense est mis en œuvre pour améliorer les performances du modèle, on parle d'apprentissage par renforcement.

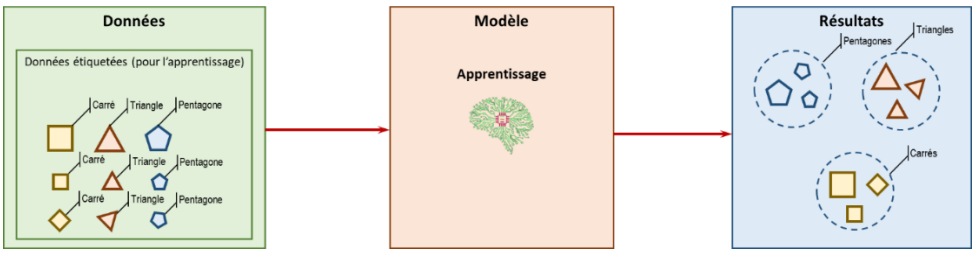
On peut donc faire une synthèse des méthodes d'apprentissage dont nous avons pris connaissance (il en existe d'autres).



1.3. Autres définitions

Définition de la classification :

En apprentissage automatique, on appelle problème de classification les problèmes où les étiquettes sont discrètes.



Dans le cas où nous souhaitons regrouper par forme, il s'agit d'un problème de classification. Les classes discrètes sont les formes (triangles, carrés, pentagones ...). Il serait aussi possible d'opérer un regroupement par couleur. Les classes seraient donc différentes.

Définition d'une régression :

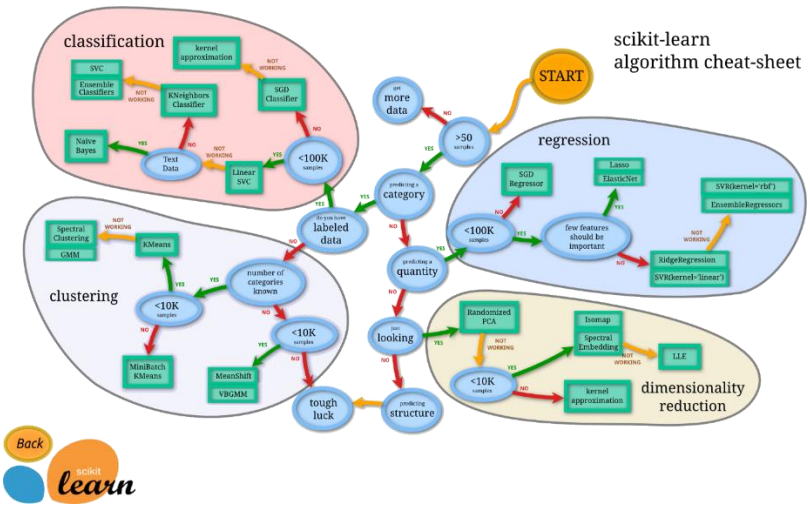
En apprentissage automatique, on appelle problème de régression les problèmes où les étiquettes sont continues.

Remarque : Dans le cas où nous souhaiterions que l'algorithme détermine l'aire ou le périmètre de la forme, les étiquette seraient alors continues sur $\mathbb{R}_*^+$.

2. Mécanismes d'apprentissages

2.1. Classifications des algorithmes d'apprentissage

La bibliothèque *scikit-learn* propose une classification de divers algorithmes en fonction des apprentissages automatiques (hors réseaux de neurones).



2.2. Validation du modèle

Une fois un algorithme d'apprentissage choisi, on se pose la question de la validation du modèle.

Quels critères et outils vont nous permettre de considérer que notre apprentissage est « bon » ?

Lors d'un problème de classification, il est par exemple possible de déterminer les écarts entre les valeurs prédites par l'algorithme et les valeurs cibles.

2.2.1. Critères de validation des problèmes de classification

Définition de la valeur prédictive positive :

Valeur prédictive positive (*accuracy classification score*) :

$$Justesse = \frac{Nombre\ de\ prédictions\ vraies}{Nombre\ de\ prédictions\ totales}$$

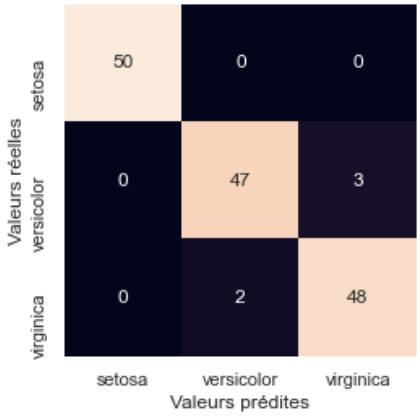
Définition des matrices de confusion :

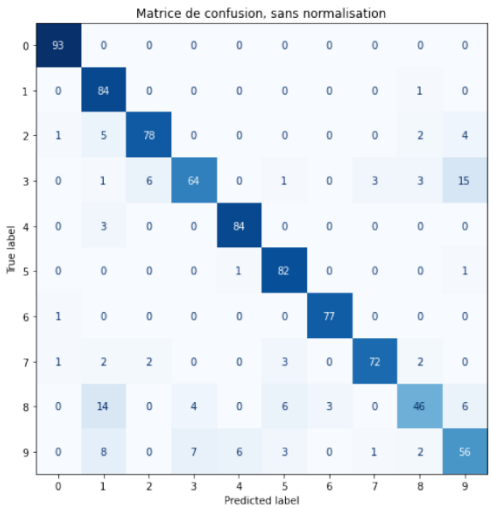
La matrice de confusion est une métrique permettant de déterminer la qualité d'une classification. En abscisses sont indiquées les valeurs prédites, et en ordonnées les valeurs réelles.

Exemple : Dans l'exemple ci-contre, nous cherchons à classifier des Iris, grâce à un algorithme, selon 3 familles : les setosa, les versicolor et les virginica.

Sur la diagonale, sont retrouvées les iris dont l'espèce a été correctement prédite.

En revanche, 3 versicolor ont été classées parmi les virginica et 2 virginica ont été classifiées dans les versicolor.





Exemple : Matrice de confusion pour la tâche de reconnaissance d'un chiffre manuscrit. La matrice de confusion permet de voir que les erreurs de prédiction sont concentrées sur les classes 3, 8 et 9 et que les confusions les plus fréquentes de la fonction de prédictions sont sur les couples 3-8, 3-9 et 8-9 (certainement dû à la présence de boucles dans le tracé des chiffres aux même endroits dans l'image).

2.2.2. Critères de validation des problèmes de régression

Définition de l'erreur moyenne quadratique (*mean squared error)* :

L'erreur moyenne quadratique est une moyenne d'écart au carré :

$$msq = \frac{1}{N}.\sum_{i=1}^{N}\|Yi - \hat{Y}i\|^2$$

en notant Yi la valeur réelle (étiquetée) et $\hat{Y}i$ la valeur estimée par l'algorithme.

La qualité d'un modèle est mesurée en comparant les prédictions effectuées sur un ensemble de données de taille N qui ne sont pas dans la base de données d'entraînement appelées données de test.

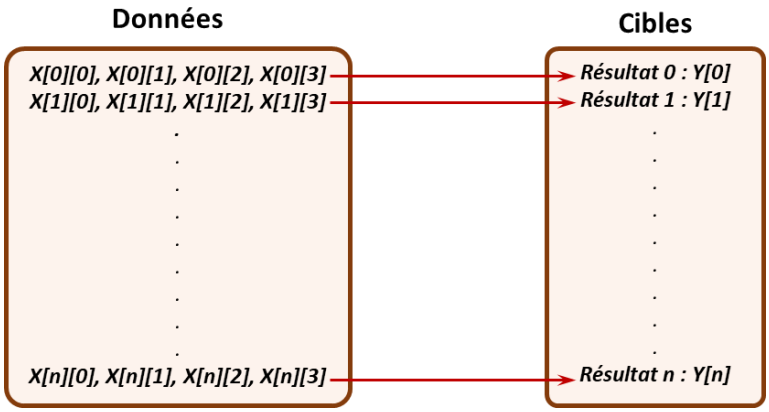
2.3. Séparation des données/Gestion des données ?

Lorsque l'on souhaite disposer d'un modèle défini par un algorithme d'IA, il faut en premier lieu disposer de données.

2.3.1. Types de données – Apprentissage supervisé

En apprentissage supervisé, il est nécessaire de connaître des données d'entrées ainsi que les sorties correspondantes (appelées aussi *cibles* ou *target*).

Données d'entrées : matrice X de n observations avec 4 caractéristiques.
Données de sortie : vecteur Y de n résultats.



Dans le cadre d'une classification, où r résultats sont possibles. On peut attribuer une valeur (entière) entre 1 et r à chacun des résultats, notamment si ceux-ci sont des données qualitatives.

Remarque : Selon la nature des sorties, on distingue plusieurs grandes classes de problèmes.

- Si les sorties y_i sont des **scalaires réels** $\forall i \in \{1, \dots, n\}\ y_i \in \mathbb{R}$, on parle alors de problème de **régression simple (ou monovariable).**
- Si les sorties y_i sont des **vecteurs de réels** (de dimension p finie) $\forall i \in \{1, \dots, n\}\ y_i \in \mathbb{R}^p$, on parle alors de problème de **régression multiple (ou multivariable).**

2.3.2. Types de données – Apprentissage non-supervisé

L'apprentissage non-supervisé est caractérisé par l'absence de sortie y fournie lors de l'apprentissage du modèle prédictif.
Pourtant on construit une fonction $f: X \to Y$ mais qui au lieu de reproduire le comportement observé sur des exemples, doit cette fois produire des sorties satisfaisant certains critères.

Exemple : On peut souhaiter prédire la partie manquante d'une donnée : les données futures pour une série temporelle ou remplacer les pixels flous d'une image par la vraie image nette.

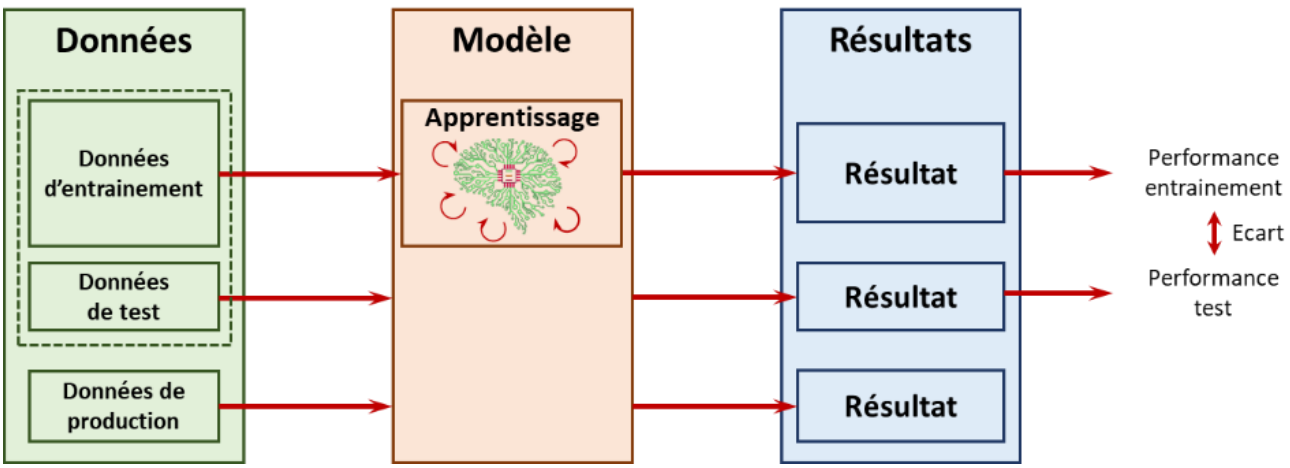
2.3.3. Séparation des données

Lors du démarrage d'un apprentissage supervisé, on dispose d'un jeu de données comprenant les données d'entrée et les cibles correspondantes. Il est d'usage de séparer ces données en deux parties :

- Les données d'entraînement permettant d'entraîner le modèle.
  Dans la pratique on utilise entre 60 et 80% des données de base.
- Les données de test, permettant de valider l'apprentissage (ou le modèle).
  Dans la pratique entre 20 et 40% des données de base.

On commence donc par réaliser un apprentissage sur les données d'entraînement. Cet entraînement produit des résultats. Connaissant les cibles correspondant aux données d'entraînement, on peut donc en déduire une performance du modèle sur le traitement des données d'entraînement.

On utilise alors le modèle en lui donnant les données de test. De même, on peut donc en déduire une performance du modèle sur le traitement des données de test.



On perçoit donc que la qualité de l'apprentissage peut dépendre du choix utilisé lors de la séparation des données.

De plus, certains choix de paramètres permettant d'affiner le modèle sont aussi liés aux données d'entraînement sélectionnées. Une des solutions pour résoudre ce problème est d'avoir recours à la validation croisée.

3. Algorithmes d'apprentissage

3.1. Algorithme des k plus proches voisins (KNN – K Nearest Neighbors)

3.1.1. Présentation

L'algorithme des *k* plus proches voisins (aussi appelé *k-nearest neighbors algorithm* ou *kNN*) permet de réaliser des opérations de régression et de classification sur un ensemble de données.

Pour une première approche, cette méthode permet d'estimer la classe d'une nouvelle donnée en déterminant la norme entre cette nouvelle donnée et l'ensemble des données du jeu d'entraînement. Parmi les *k* plus proches voisins, on recherche la classe majoritaire et on attribue cette classe à la nouvelle donnée.

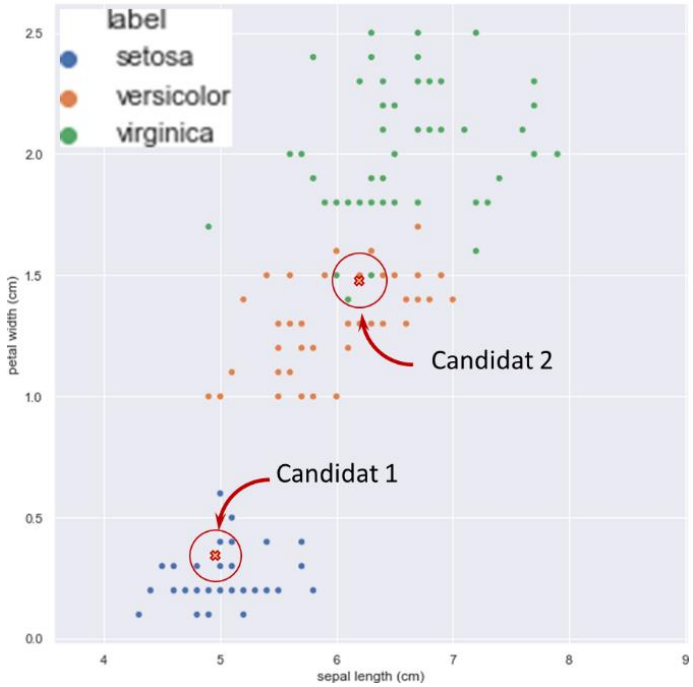
Exemple : Dans le cas ci-contre, on cherche à classifier les Iris selon 3 catégories (setosa, versicolor et virginca).

Les données étiquetées sont représentées dans le plan longueur du sépale en abscisse et largeur du pétale en ordonnée.

Soient les candidats 1 et 2, deux Iris dont on connaît les caractéristiques mais pas l'espèce.

Les 5 plus proches voisins du candidat 1 sont des setosa. Il est donc probable que le candidat 1 soit un setosa aussi.

Concernant le candidat 2, parmi les 5 plus proches voisins, 3 sont des virginica et 2 sont des versicolor. On peut faire l'hypothèse que c'est un virginica.



3.1.2. L'algorithme

Prenons comme exemple des données sous la forme *[x,y,c]* où *x* et *y* représentent des coordonnées et *c* la classe d'appartenance.

Première étape : Calculer la distance :

Il faut tout d'abord définir une fonction de distance. On peut par exemple utiliser la distance euclidienne.

Pour deux vecteurs x_1 et x_2 de taille *N*, $$\boldsymbol{d} = \sqrt{\sum_{i=1}^{N}(\boldsymbol{x_{1,i}} - \boldsymbol{x_{2,i}})^2}$$

```
import math as m
def distance(x1:list, x2:list) -> float :
    distance = 0.
    for i in range(len(x1)):
        distance += (x1[i]-x2[i])**2
    return m.sqrt(distance)
```

Deuxième étape : Détermination des plus proches voisins :

Pour cette étape, on dispose de données pour l'entraînement *data_train* et d'une donnée à tester *data_test*.
Il va falloir calculer la distance entre la donnée à tester et les données d'entraînement puis trier les données. Enfin, on retournera les *k* lignes les plus proches de *data_test*.

```
def get_voisins(data_train:list, data_test, k:int) -> list :
    distances =[]
    for ligne in data_train :
        d = distance(ligne,data_test)
        distances.append([ligne,d])
    # Tri de la liste suivant la seconde colonne (distances)
    distances.sort(key=lambda t : t[1])
    voisins = []
    for i in range(k) :
        voisins.append(distance[i][0])
    return voisins
```

Troisième étape : Prédiction :

Une fois qu'on connaît les *k* plus proches voisins, on regarde quelle est la classe majoritaire parmi ces voisins.

```
def prediction(data_train:list, data_test, k:int) -> list :
    voisins = get_voisins(data_train, data_test, k)
    sorties = [ligne[-1] for ligne in voisins]
    predict = max(set(sorties), key=sorties.count)
    return predict
```

3.2. Régression mono variable

Dans cette section, on identifiera l'espace de sortie Y à $\mathbb{R}$. La régression linéaire consiste à choisir des fonctions de prédiction f de la forme :

$$\forall x \in \mathbb{R}\ f(x) = a.x \qquad \text{où } a \text{ est une constante réelle}$$

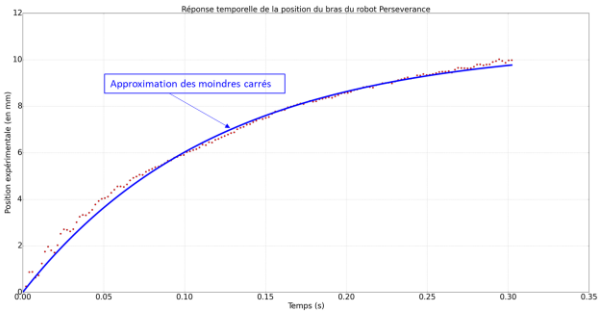
Dans le cadre de la régression linéaire, la fonction de coût L communément utilisée est l'erreur quadratique. La fonction de prédiction optimale est celle qui minimise l'erreur sur la base de données d'entraînement :

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

Sous l'hypothèse d'une solution correspondant à une fonction linéaire de la forme de l'équation $\forall x \in \mathbb{R}\ f(x) = a.x$, la constante $\hat{a}$ est obtenue en résolvant le problème de minimisation :

$$\hat{a} = \min_a \frac{1}{n}\sum_{i=1}^{n}(a.x_i - y_i)^2$$

Remarque : Une analogie peut être faite avec le TD vue en SUP en Automatique (TD 4 : Robot Perseverance) dans lequel on souhaite faire une régression linéaire aux moindres carrés pour retrouver les paramètres caractéristiques du système du premier ordre.



3.3. Régression multi-variables

On peut étendre l'étude de la régression au cas où les entrées x_i sont des vecteurs de $\mathbb{R}^d$ et les sorties y_i des vecteurs de $\mathbb{R}^p$. Le modèle linéaire liant les entrées x et les sorties y suppose l'existence d'un vecteur de paramètres $A \in \mathbb{R}^{d.p}$ tel que $y = x^T A$

Ce vecteur est calculé en minimisant l'erreur de prédiction quadratique sur la base de données d'entraînement. Ainsi si on dispose d'une base de données $(x_i, y_i)_{i\in\{1,\dots,n\}}$, on peut calculer la matrice $\hat{A}$ correspondant au modèle reproduisant les observations le plus fidèlement :

$$\hat{A} = \min_{A\in\mathbb{R}^{d.p}} \sum_{i=1}^{n}\frac{1}{n}\left\|y_i - x_i^T A\right\|^2$$

où $\min_{A\in\mathbb{R}^{d.p}} f(A)$ représente la valeur prise par A lorsque $f(A)$ atteint son minimum.

Soient $Y = \begin{pmatrix} y_1^T \\ y_2^T \\ \dots \\ y_n^T \end{pmatrix} \in \mathbb{R}^{n.p}$ et $X = \begin{pmatrix} x_1^T \\ x_2^T \\ \dots \\ x_n^T \end{pmatrix} \in \mathbb{R}^{n.d}$ On peut montrer que : $\hat{A} = \min_{A\in\mathbb{R}^{d.p}} \frac{1}{n}\|Y - XA\|^2$

La solution de ce problème a pour expression : $\hat{A} = (X^T X)^{-1} X^T Y$