10 IA-Machine learning

Sommaire

1. Introduction :

La principale activité de l'ingénieur est de résoudre des problèmes technologiques concrets, liés à la conception, la réalisation et la mise en œuvre de produits, de systèmes ou de services. Pour cela ces connaissances scientifiques basées sur les lois de la physique lui permettent de modéliser les problèmes sous forme d'équations mathématiques. Grâce à la puissance de calcul l'ordinateur les solutions sont disponibles rapidement.

Le Machine Learning est utilisé lorsque l'on ne connait pas les lois de la physique. C'est l'ordinateur qui à partir d'exemples connus (phase d'apprentissage) permettra de proposer une solution probable au problème.

Définition :
Arthur Samuel en 1959 donne la définition suivante du Machine learning :
« *Machine learning is the science of getting computers to learn without being explicity programmed.*[1] »

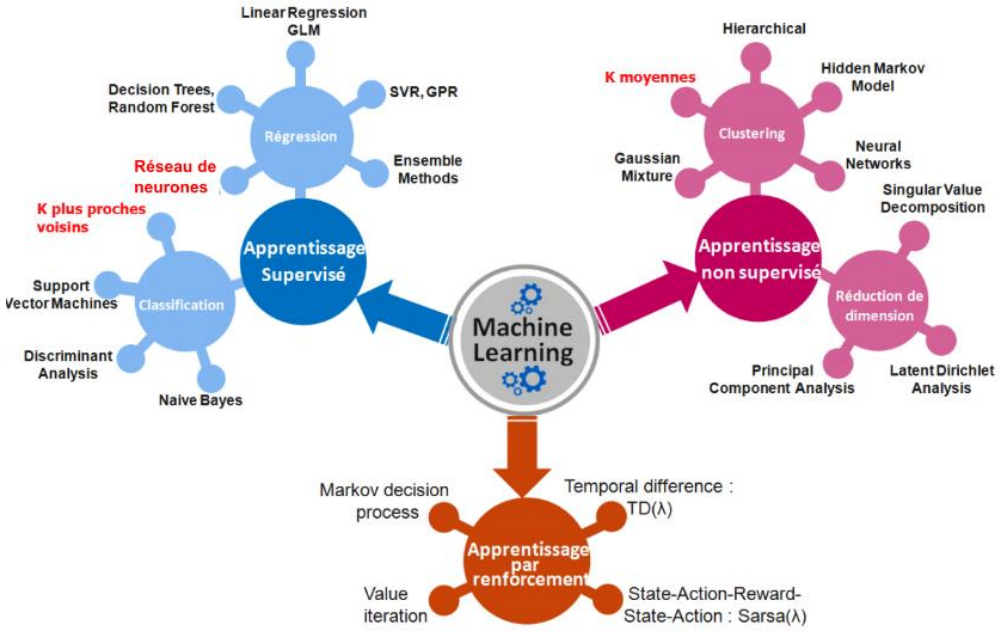
Arthur Samuel (1901-1990) est un pionnier américain de l'intelligence artificielle

Le Machine Learning consiste à laisser l'ordinateur apprendre quels calculs effectuer, plutôt que de le programmer de façon explicite.

2. Les différentes méthodes d'apprentissage :

La difficulté est alors de donner la capacité d'apprendre à l'ordinateur. Les méthodes d'apprentissages utilisées sont les suivantes :

- L'apprentissage supervisé (Supervised Learning)
- L'apprentissage non supervisé (Unsupervised Learning)
- L'apprentissage par renforcement (Reinforcement Learning)



Nota : L'apprentissage supervisé est la méthode la plus utilisée en Machine Learning.

[1] Le machine learning est le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmé.

3. Apprentissage supervisé :

On parle ainsi d'apprentissage supervisé lorsque l'on fournit à une machine beaucoup d'exemples qu'elle doit étudier (bases de données d'entrainement). De manière générale le système exploite des exemples et acquiert la capacité à les généraliser ensuite sur de nouvelles données.

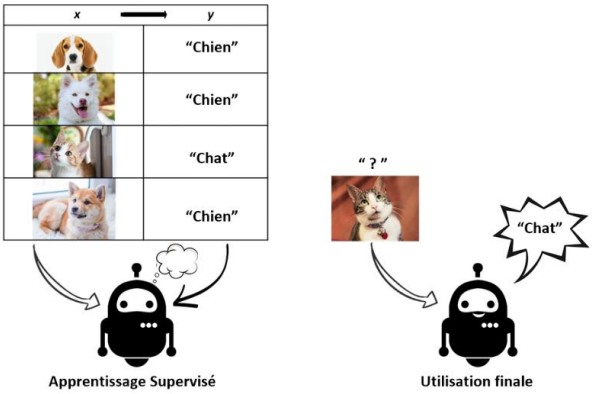
Pour mettre en œuvre l'apprentissage supervisé, il est nécessaire de prendre en compte les 4 notions suivantes :

- Le Dataset (apprendre à partir d'exemples)
- Le Modèle et ses paramètres
- Les erreurs du modèle : La Fonction Coût
- L'Algorithme d'apprentissage.

3.1 le Dataset :

On parle d'apprentissage supervisé lorsque l'on fournit à une machine beaucoup d'exemples ($\boldsymbol{x}$, $\boldsymbol{y}$) dans le but de lui faire apprendre la relation qui relie $\boldsymbol{x}$ à $\boldsymbol{y}$.

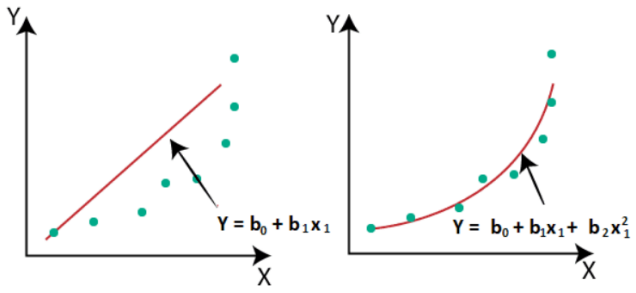
- La variable $\boldsymbol{y}$ porte le nom de *target* (la cible). C'est la valeur que l'on cherche à prédire.
- La variable $\boldsymbol{x}$ porte le nom de *feature* (facteur). Un facteur influence la valeur de $\boldsymbol{y}$, et on a en général beaucoup de *features* ($\boldsymbol{x_1}$, $\boldsymbol{x_2}$, …) dans notre *Dataset* que l'on regroupe dans une matrice $\boldsymbol{X}$.



Facteurs x_i														Cible y
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	0	0	45	Mexico	<=50K
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	35	United-States	<=50K
32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	White	Male	0	0	40	United-States	<=50K
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	<=50K
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	>50K
40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
54	Private	302146	HS-grad	9	Separated	Other-service	Unmarried	Black	Female	0	0	20	United-States	<=50K
35	Federal-gov	76845	9th	5	Married-civ-spouse	Farming-fishing	Husband	Black	Male	0	0	40	United-States	<=50K
43	Private	117037	11th	7	Married-civ-spouse	Transport-moving	Husband	White	Male	0	2042	40	United-States	<=50K
59	Private	109015	HS-grad	9	Divorced	Tech-support	Unmarried	White	Female	0	0	40	United-States	<=50K

3.2 Le modèle et ses paramètres :

En *Machine Learning*, on développe un modèle à partir des *Dataset*. Il peut s'agir d'un modèle linéaire figure de gauche, ou un modèle non-linéaire figure de droite.

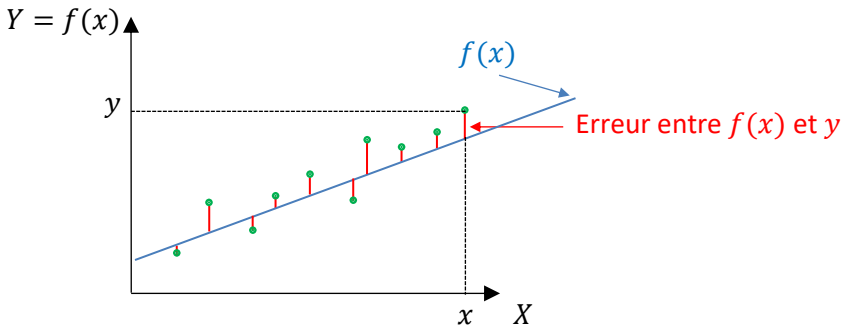


Modèle linéaire Modèle polynomial

Les valeurs b_i sont appelées les paramètres du modèle.

3.3 Les erreurs du modèle :

On appelle *Fonction Coût* l'ensemble de ces erreurs (le plus souvent on prend la moyenne quadratique des erreurs pour calculer cette fonction).



Cette fonction permet de quantifier numériquement la validité d'un modèle par rapport à des données brutes. Si nous prenons par exemple une liste de données X et associé à cette liste, une seconde liste de données Y, alors nous pouvons déterminer un modèle permettant de prédire au mieux la valeur $f(x_i)$ associée à une valeur x_i.

3.4 Algorithme d'apprentissage

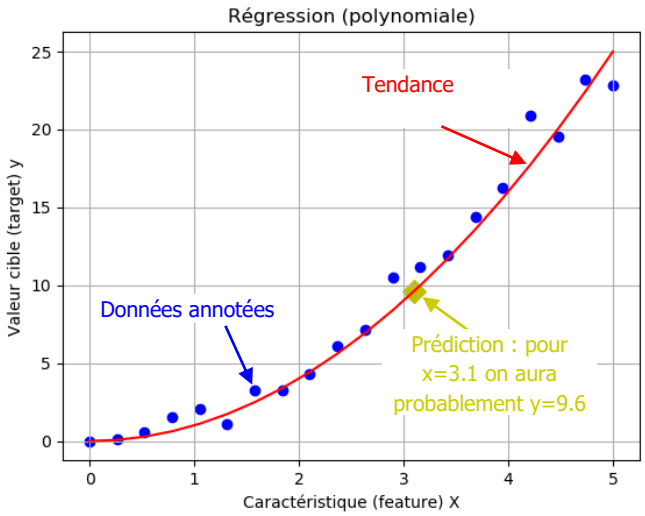
Pour obtenir un modèle satisfaisant on cherche à minimiser la *Fonction Coût*. Le but est d'alors de trouver les paramètres du modèle (les valeurs b_i) qui la minimise. Pour cela on utilise des algorithmes d'apprentissage :

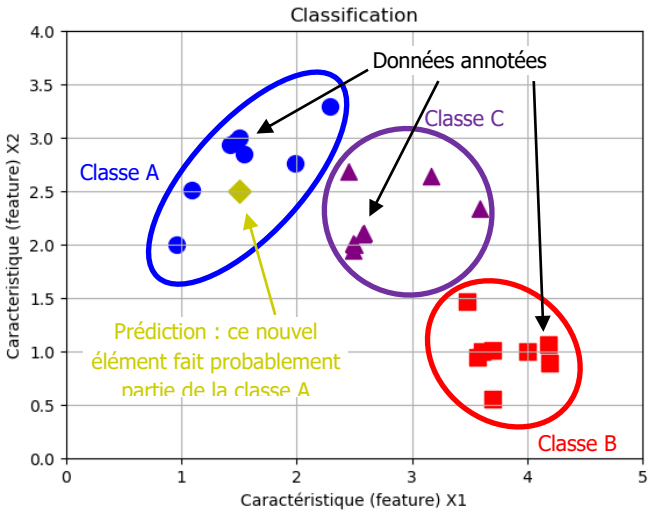
- Méthode des ***k* plus proches voisins**
- L'algorithme d'optimisation dit de **descente de gradient** qui permet de trouver le minimum de n'importe quelle fonction convexe en convergeant progressivement vers celui-ci.

4. Les applications de l'apprentissage supervisé

Avec l'apprentissage supervisé on peut développer des modèles pour résoudre 2 types de problèmes :

- Les problèmes de régression
- Les problèmes de classification





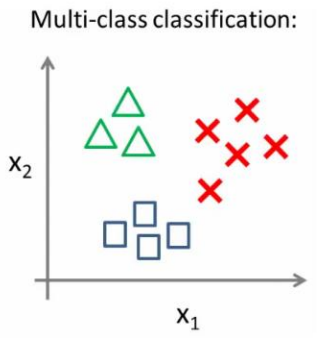
Dans un problème de **régression**, on cherche à prédire la valeur d'une variable continue, c'est-à-dire une variable qui peut prendre une infinité de valeurs. Par exemple :

- Prédire le prix d'un appartement (y) selon sa surface habitable (x)
- Prédire la quantité d'essence consommée (y) selon la distance parcourue (x)

Dans un problème de **classification**, on cherche à classer un objet dans différentes classes, c'est-à-dire que l'on cherche à prédire la valeur d'une variable discrète (qui ne prend qu'un nombre fini de valeurs). Par exemple :

- Prédire si un email est un spam ($classe\ y\ =\ 1$) ou non ($classe\ y\ =\ 0$) selon le nombre de liens présent dans l'email (x)
- Prédire si une tumeur est maligne ($y\ =\ 1$) ou bénigne ($y\ =\ 0$) selon la taille de la tumeur (x_1) et l'âge du patient (x_2)

Dans le cas d'un problème de classification, on représente souvent les classes par des symboles, plutôt que par leur valeur numérique (0, 1, …)



5. Prédiction de valeurs réelles : Cas de la régression

Le cas de la régression correspond au cas où les valeurs de sorties sont soit représentables par des valeurs réelles (cas de la régression monovariable) soit par des vecteurs dont les composantes sont des valeurs réelles (régression multivariable).

5.1 Régression linéaire monovariable

5.1.1 Modèle

On identifiera l'espace d'entrée x et celui de sortie y sont tous les deux l'ensemble des réels $\mathbb{R}$.

Définition : Un modèle de régression linéaire est une fonction de prédiction $f: x \to y$ de la forme :

$$\forall x \in \mathbb{R}\ f(x) = a \cdot x + b$$

Où a et b sont deux constantes réelles appelés paramètres du modèle.

5.1.2 Fonction Cout

Exemple : si pour les jeux de données X et Y de même taille, on cherche à déterminer un modèle linéaire ($f(x) = a \cdot x + b$), alors on cherche à déterminer les coefficients a et b qui minimisent les erreurs entre la valeur réelle et celle prédite. Soit J la fonction coût qui dépend des paramètres a et b:

$$J(a,b) = \frac{1}{n}\sum_{k=0}^{n}(f(x_k) - y_k) \quad \text{avec : } n = len(X) = len(Y)$$

On remarque que potentiellement les erreurs peuvent se compenser et on pourrait obtenir un coût nul pour une fonction ne modélisant pas bien notre jeu de données. C'est ainsi que le mathématicien allemand Carl Friedrich Gauss a introduit l'erreur quadratique moyenne (mean square loss).

$$J(a,b) = \frac{1}{2 \cdot n}\sum_{k=0}^{n}(f(x_k) - y_k)^2 \quad \text{avec : } n = len(X) = len(Y)$$

Nota : Le 2 de « $\frac{1}{2 \cdot n}$» a été ajouté pour simplifier les calculs de la descente de gradient.

5.1.3 L'algorithme d'apprentissage :

On cherchera donc à minimiser cette fonction coût, qui dépend des paramètres a et b. On est donc amené à chercher le minimum à l'aide des dérivées partielles :

$$\frac{\partial J(a,b)}{\partial a} \text{ et } \frac{\partial J(a,b)}{\partial b}$$

La recherche de ce minimum met en œuvre la méthode de la descente du gradient.

5.2 Implémentation en Python

La plupart des algorithmes permettant de construire des modèles prédictifs sont implémentées dans la librairie Python $scikit-learn$. La mise en œuvre d'un tel modèle est toujours la même quelle que soit l'algorithme employé.

1. Construction d'une base de données d'entrainement constitué d'un tableau $numpy.ndarray$ (X) dont chaque ligne est un descripteur d'une entrée et un tableau (Y) dont chaque ligne et la sortie (correspondant à l'entrée dans la même ligne de X).
2. Initialisation d'un modèle prédictif (et stockage de ce modèle dans une variable souvent appelée clf pour classifier, ce nom est utilisé même lorsqu'on traite d'un problème de régression et non de classification).
3. Calcul des paramètres du modèle à partir de la base de données d'entraînement via la commande $clf.fit(X,Y)$. Dans le cas d'une régression linéaire, Python calcule les paramètres a et b du modèle de régression linéaire et les stocke dans clf.
4. Prédire la sortie pour une nouvelle entrée x, on utilise la fonction $clf.predict(x)$.

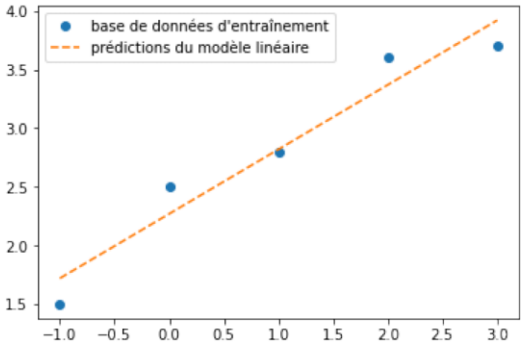
Le code ci-après illustre le calcul des paramètres d'une régression linéaire pour la base de données d'entraînement (X,Y) fournie et trace le modèle obtenu.

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Etape 1 : préparation des données
X = np.array([[-1],[0],[1],[2],[3]])
Y = np.array([1.5,2.5,2.8,3.6,3.7])
# Etape 2 : initialisation du modèle
clf = LinearRegression()
# Etape 3 : phase d'apprentissage
clf.fit(X,Y)
# Etape 4 : prédiction pour un exemple
x = np.array([[1.5]])
resultat = clf.predict(x)
print(resultat)
```

On peut ensuite utiliser le modèle pour effectuer les prédictions sur des entrées et tracer les sorties en fonctions des entrées.

```
import matplotlib.pyplot as plt

# calcul des sorties prédites par le modèle pour les entrées dans X :
predictions = clf.predict(X)
# tracé des points de la base de données d'entraînement
plt.plot(X,Y,'o')
# tracé des prédictions du modèle linéaire
plt.plot(X,predictions,'—')
plt.legend(["base de données d'entraînement","prédictions du modèle linéaire"])
plt.show()
```

Ce code permet de tracer la courbe représentée ci-dessous



Base de données d'entraînement et modèle linéaire associé.

6. Classification : Algorithme des K plus proches voisins

L'algorithme des K plus proches voisins est un algorithme d'apprentissage supervisé que l'on peut utiliser pour la classification de données (KNN peut être aussi utilisé dans le cas d'une régression).

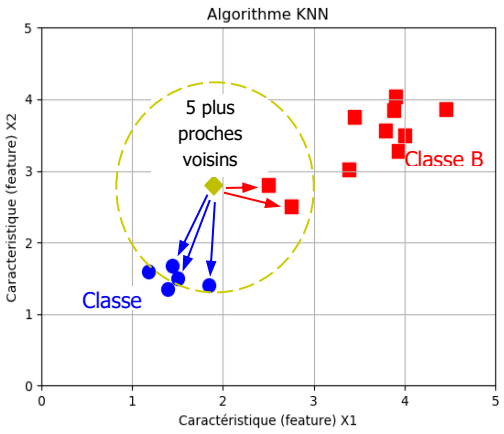
6.1 KNN n'utilise pas de modèle prédictif

Pour effectuer une prédiction, l'algorithme KNN ne va pas calculer un modèle prédictif à partir d'un Training Set comme c'est le cas pour la régression linéaire. En effet, **KNN n'a pas besoin de construire un modèle prédictif**.

6.2 Comment KNN effectue une prédiction ?

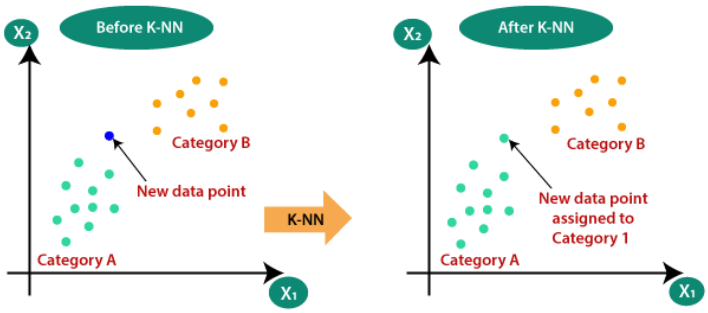
KNN utilise la proximité pour effectuer des classifications ou des prédictions sur le regroupement d'un point de données individuel

Le principe est extrêmement simple : si les K plus proches voisins d'un élément sont majoritairement d'une certaine classe, alors cet élément sera associé à cette classe.



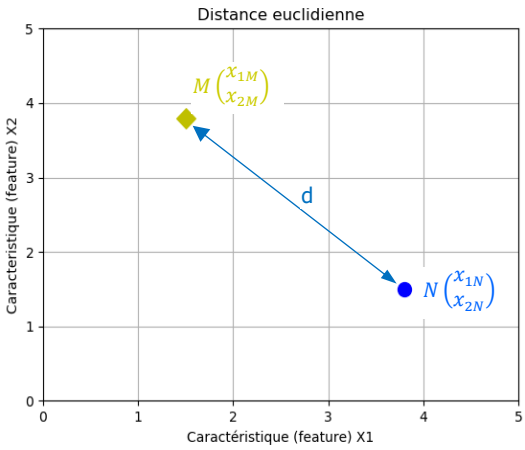
Exemple : sur la figure ci-contre, les 5 plus proches voisins de l'élément que l'on cherche à classer sont majoritairement de classe A. On peut en déduire que cet élément sera probablement lui aussi de classe A.

Nota : On peut remarquer que si on avait choisi $K = 3$ sur l'exemple précédent, la conclusion aurait été différente.



6.2.1 Notion de proximité :

La proximité avec des voisins peut être évaluée de différentes façons (distance euclidienne, distance de Manhattan, distance de Tchebychev, etc.) **Le choix de la méthode se fait en fonction du type de données**. Ainsi pour les **données quantitatives** (exemple : poids, salaires, taille, montant de panier électronique etc…) et **du même type**, la distance euclidienne est souvent choisie. En deux dimensions le modèle de calcul par ce type de méthode est le suivant :



$$D_e(MN) = \sqrt{(x_{2N} - x_{2M})^2 + (x_{1N} - x_{1M})^2}$$

Plus généralement, dans un espace de dimension n :

$$D_e(MN) = \sqrt{\sum_{i=1}^{n} (x_{iN} - x_{iM})^2}$$

6.2.2 Choix de la valeur de K :

Son choix conditionne l'efficacité de l'algorithme.

Quelques principes :

- Dans le cas de la classification binaire (Classification à faire entre uniquement deux classes), on utilise un $\boldsymbol{K}$ ***impair*** pour être sûr d'obtenir une étiquette majoritaire.
- La valeur de $\boldsymbol{K \neq 1}$ car elle trop sujette aux erreurs ou aberrations possibles dans le jeu de données d'apprentissage. De manière générale une valeur de K petite provoque un sous apprentissage (underfitting)
- Si on utilise $\boldsymbol{K = n}$ et n étant le nombre d'observations, on risque d'avoir du sur apprentissage (overfitting) et par conséquent un modèle qui se généralise mal sur des observations qu'il n'a pas encore vu.

Le choix raisonnable de K est donc un art à part entière. La technique la plus répandue (celle qui donne de meilleurs résultats pratiques) consiste à tester plusieurs valeurs de K et à choisir celle qui donne les meilleurs résultats.

6.3 Algorithme KNN

Début Algorithme

Données en entrée :

- Un ensemble de données D (Train_Set).
- Une fonction de définition distance d.
- Un nombre entier K

Pour une **nouvelle observation X** (New_Data) dont on veut prédire sa variable de sortie y

Faire :

1. Calculer toutes les distances de X avec les autres observations du jeu de données D, puis créer une liste (*L_Ind*) des indices dans l'ordre dans lequel ils doivent apparaitre pour que la liste (*Liste_D*) soit triée (ordre croissant)
2. Retenir les K observations du jeu de données D les plus proches de X en utilisant la fonction de calcul de distance d
3. Choisir la valeur d'étiquette dominante parmi les K valeurs précédentes.
4. Retourner la valeur choisie dans l'étape 3 comme étant la valeur qui a été prédite par KNN pour l'observation X.

Fin Algorithme