

# Fondement de l'Intelligence Artificielle



Abir CHaabani

[abir.chaabani@gmail.com](mailto:abir.chaabani@gmail.com)

[abir.chaabani@enicar.ucar.tn](mailto:abir.chaabani@enicar.ucar.tn)

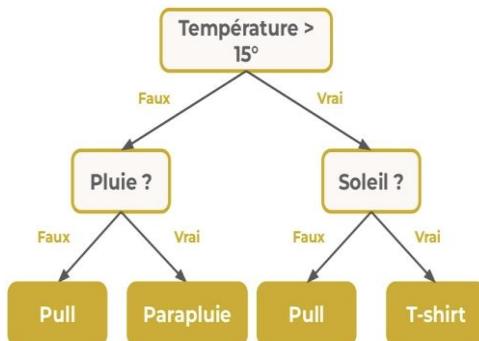
Université de Carthage - ENICARTHAGE

2 GINF 2024/2025

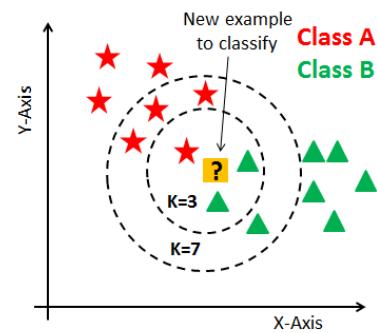
1

## Chapitre 6

### Arbre de décision



### K plus proche voisins



2

# Partie I: Les arbres de décision

3

## Motivation

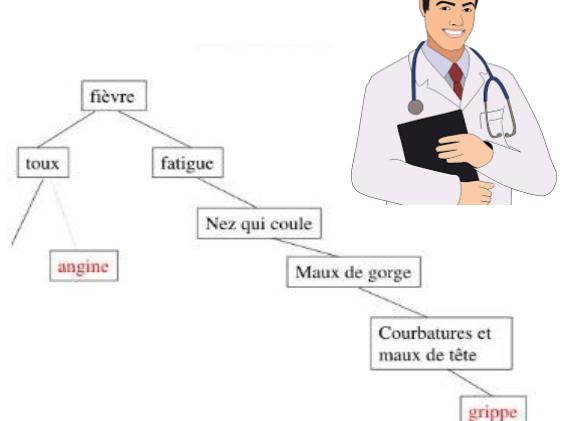
Décider comme un médecin

### Exemple 1 : Détection de la grippe

- ◊ Apparition soudaine de fièvre élevée
- ◊ Le patient est fatigué
- ◊ Rhinorrhée (nez qui coule)
- ◊ Toux
- ◊ Douleurs à la gorge
- ◊ Enrouement, douleurs dorsales, des membres et céphalées



Grippe



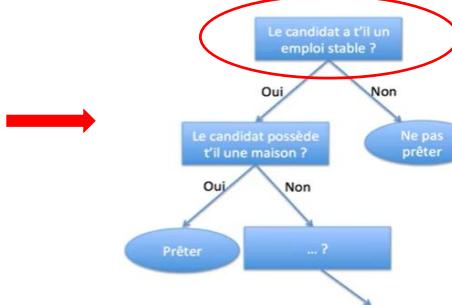
Détection de la grippe  
représentée sous forme d'arbre

# Arbre de décision

## Arbres de décision :

- Outil utilisé dans l'exploration de données et informatique décisionnelle.
- Représentation hiérarchique de la structure des données sous forme des séquences de décision (tests) en vue de la prédiction d'un résultat ou d'une classe.

**Un arbre de décision permet d'expliquer une variable cible à partir d'autres variables dites explicatives.**



Choisir la variable explicative qui permet la meilleure séparation des individus  
« Le candidat a-t-il un emploi stable »

Produire un arbre qui révèlent les relations hiérarchiques entre les variables par un algorithme itérative qui vise à diviser itérativement l'espace des données selon les variables les plus discriminantes.

# Arbre de décision

## Exemple 2 : dois-je sortir le chien ou non

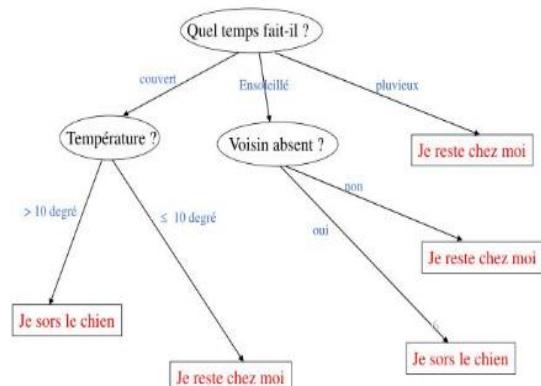


### Attributs, caractéristiques, variables (Features) :

- ◊ quel temps fait-il ? {pluvieux, ensoleillé, couvert} : attribut nominal
- ◊ Température extérieure : attribut continu (numérique)
- ◊ Voisin parti avec son chat : attribut booléen

### Décision à prendre

- ◊ Sortir ou non le chien



# Arbre de décision

Il existe deux principaux types d'arbre de décision :

1. Les arbres de classification (**Classification Tree**) permettent de prédire à quelle classe la variable cible appartient, dans ce cas la prédiction est une **étiquette de classe**.



2. Les arbres de régression (**Regression Tree**) permettent de prédire une quantité réelle (Ex: le prix d'une maison ou la durée de séjour d'un patient dans un hôpital), dans ce cas la prédiction est une **valeur numérique**.



7

## Arbre de décision: Classification

### Définition

- ◆ Un arbre de décision est un ensemble de **règles de classification** organisées de manière (**arborescente /hierarchique**), où chaque décision (ou test) est associée à un attribut, et chaque chemin dans l'arbre mène à **une prédiction de classe**.

### Motivation

- ◆ Offrir un **modèle prédictif compréhensible et interprétable** par l'utilisateur, contrairement à d'autres modèles plus complexes

### Principe de fonctionnement

- ◆ Prédire la valeur d'un **attribut cible (ou variable dépendante)** à partir d'un ensemble de **valeurs d'attributs explicatifs (ou variables prédictives)**, en divisant l'espace de données selon les valeurs des attributs.

8

# Arbre de décision: Classification

## Structure

- ❖ Un arbre est équivalent à un ensemble de règles de décision : un modèle facile à comprendre.
- ❖ Un arbre est composé :
  - ❖ **de noeuds** : classes d'individus de plus en plus fines depuis la racine.  
représentent un **test** sur un attribut.
  - ❖ **d'arcs** : prédictats de partitionnement de la classe source.
  - ❖ **Feuilles** : indiquent la **classe prédictée** (ou une valeur dans le cas de la régression).

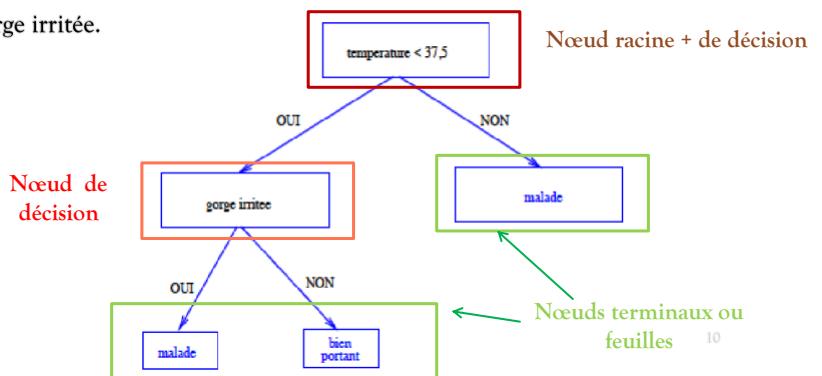
9

# Arbre de décision: Classification

- ❖ Exemple : Décider si un patient est **malade** ou **bien portant** selon sa **température** et s'il a **la gorge irritée**.

- ❖ Arbre de décision :

- ❖ 2 classes : malade ; bien portant
- ❖ 2 variables : température, gorge irritée.



# Arbre de décision: Classification

## ❖ Vocabulaire :

- ❖ **Noeud interne, intermédiaire ou test (noeud de décision)** : chaque nœud intermédiaire est défini par un test construit à partir d'une variable. Le test est applicable à toute description d'une instance et généralement un test sur un seul attribut.
- ❖ **Noeud terminal ou feuille** : étiqueté par une classe.
- ❖ **Arcs** issus d'un nœud interne : réponses possibles au test du nœud.
- ❖ Arbre de décision et apprentissage :
  - ❖ Tout arbre de décision définit un **classifieur**.
  - ❖ Le classifieur se traduit immédiatement en termes de règle de décision

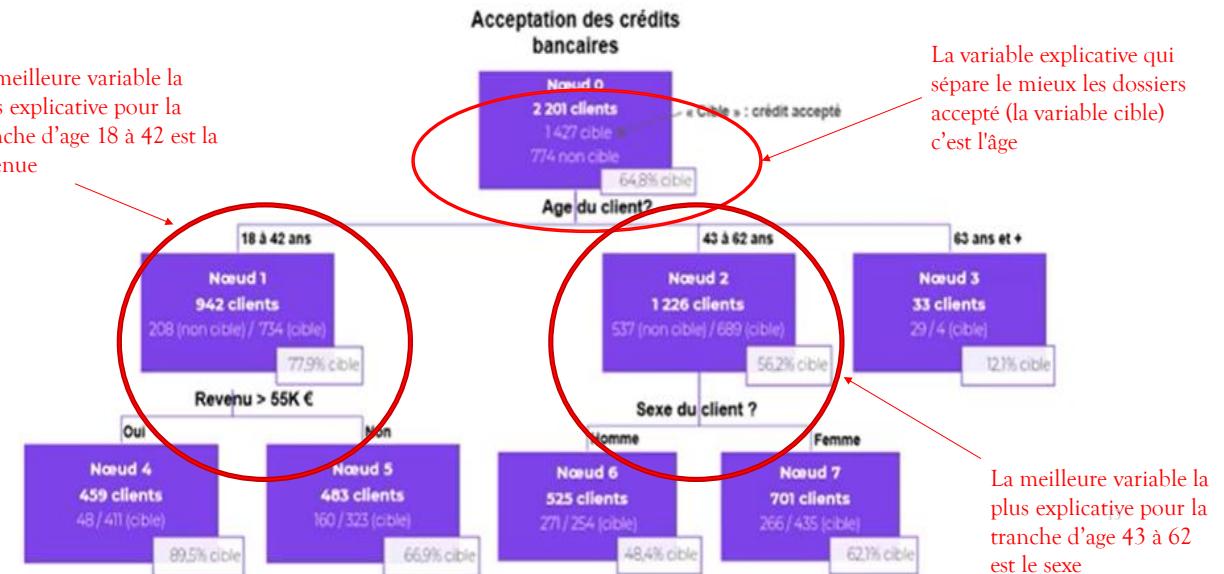
11

# Arbre de décision: Construction

- ❖ Démarrer avec un arbre vide et construire l'arbre de manière **inductive et descendante** : **Top Down Induction of Decision Trees (TDIDT)**:
  1. Démarrer avec un arbre vide (juste la racine).
  2. Choisir le meilleur attribut pour séparer les données (**en fonction d'un critère de segmentation**)
  3. Partitionner l'ensemble d'apprentissage selon les valeurs de cet attribut (**les algorithmes évaluent les différentes variables d'entrée et choisir celle qui maximise un critère donné**)
  4. Répéter récursivement pour chaque sous-ensemble jusqu'à l'arrêt.
- ❖ Critères d'arrêt :
  - ❖ échantillon pur (toutes les instances ont la même classe)
  - ❖ plus d'attributs à tester
- ❖ Algorithmes :
  - ❖ **ID3** (Quinlan, 1979), **CART** (Breiman et al., 1984), **C4.5** (Quinlan, 1993)

12

# Arbre de décision: Exemple



# Arbre de décision: Exemple

**Comment Identifier les attributs qui réduisent le mieux l'incertitude pour diviser les données?**

**Solution:** L'entropie de Shannon (Algorithmes ID3) exprime la quantité d'information.

L'entropie de Shannon quantifie l'incertitude ou le "désordre" dans un ensemble de données.

Suppose des attributs nominaux discrets.

Peut-être étendu à des attributs continus.

$$H(X) = H_2(X) = - \sum_{i=1}^n P_i \log_2 P_i.$$

Exemple: Si un jeu de données contient 50% "Oui" et 50% "Non" :

- $H = -[0.5\log_2(0.5) + 0.5\log_2(0.5)] = 1$  bit (désordre maximal, forte incertitude).
- Si 100% "Oui" :  $H=0$  (purité totale, aucune incertitude, tous les exemples appartiennent à une seule classe)

# Arbre de décision: Exemple

**Comment Identifier les attributs qui réduisent le mieux l'incertitude pour diviser les données?**

**Solution:** L'entropie de Shannon (Algorithmes ID3) exprime la quantité d'information. L'entropie de Shannon quantifie l'incertitude ou le "désordre" dans un ensemble de données. Suppose des attributs nominaux discrets. Peut-être étendu à des attributs continus.

$$H(X) = H_2(X) = - \sum_{i=1}^n P_i \log_2 P_i.$$

Exemple: Si un jeu de données contient 50% "Oui" et 50% "Non" :

- $H = -[0.5\log_2(0.5) + 0.5\log_2(0.5)] = 1$  bit (désordre maximal, forte incertitude).
- Si 100% "Oui" :  $H=0$  (purité totale, aucune incertitude, tous les exemples appartiennent à une seule classe)

15

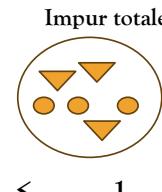
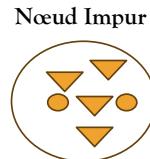
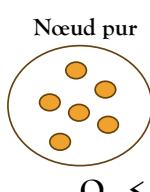
# Arbre de décision: Exemple

**Comment Identifier les attributs qui réduisent le mieux l'incertitude pour diviser les données?**

deux mesures utilisées pour évaluer  
l'impureté d'un nœud dans un arbre de  
décision

**Entropie de Shanon**

**Indice de Gini**



16

# Arbre de décision: Algorithme ID3

Quinlan, J. R., *Induction of Decision Trees*. *Mach. Learn.* 1, (Mar. 1986), pp. 81-106

- ID3 (*Iterative Dichotomiser 3*) introduit par Quinlan.
- Commence par le placement de tous les exemples d'apprentissage dans le nœud racine.
- Ensuite, chaque nœud est coupé sur un des attributs restants (qui n'a pas encore été testé).
- Le choix de cet attribut se fait à travers une mesure d'homogénéité par rapport à la variable cible.

 Cette mesure est le **Gain d'information** obtenu par le découpage.

17

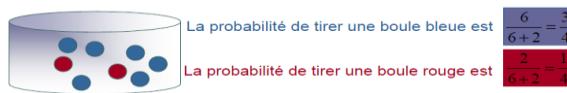
# Arbre de décision: Algorithme ID3

L'algorithme de construction de l'arbre se base sur Gain d'information qui utilise l'entropie de shanon pour décider quel est le meilleur attribut

On suppose que la variable cible a  $k$  valeurs distinctes (les étiquettes de classe). Pour un nœud  $S$  (interne ou feuille), on calcule son entropie par rapport à la cible comme suit :

- Partitionner  $S$  sur les valeurs de la cible en  $k$  groupes :  $C_1, \dots, C_k$ .
- Calculer  $p_i, i=1 \dots k$ , la probabilité qu'un élément de  $S$  se retrouve dans  $C_i$  ( $p_i \approx |C_i|/|S|$  ou  $|C_i|$  est la taille du groupe  $C_i$ ).

Exemple



- $P_i = \frac{|i|}{|S|}$
- Entropie:  $E(S) = -\sum_{i=1}^K p_i \log_2(p_i)$
- $0 \leq E(S) \leq 1$

18

# Arbre de décision: Algorithme ID3

$H(S)$  mesure l'écart de la distribution de la variable cible par rapport à la distribution uniforme :

$H(S)=0$  si  $S$  est homogène  $\rightarrow$  une seule classe (tous les éléments sont dans la même classe : toutes les probabilités  $p_i$  sont égales à 0, sauf une qui est égale à 1).

$H(S)=\max$   $\rightarrow$  classes équiprobables (si toutes les probabilités  $p_i$  sont égales = tous les groupes  $C_i$  ont la même taille :  $p_1=\dots=p_n=1/m$ ).

19

# Arbre de décision: Algorithme ID3

Étant donnée un vecteur d'attribut, on peut diviser l'ensemble  $S$  en plusieurs sous ensembles

Le gain d'information est mesuré en se basant sur la différence entre l'entropie originale de l'ensemble  $S$  et celles après sa division en se basant sur un attribut  $a$



Pour calculer le gain d'information dans un noeud interne  $S$  sur l'attribut  $a$  :

1. Partitionner  $S$  sur les valeurs de l'attribut  $a$  en  $k$  sous-groupes :  $S_1, \dots, S_m$  ( $m$  est le nombre de valeurs distinctes de l'attribut  $a$ ),
2.  $p_i$  : la probabilité qu'un élément de  $S$  appartient à  $S_i$  ( $p_i \approx |S_i|/|S|$ ),
3. Calculer  $GI(S; a) = H(S) - \sum_{i=1}^k p_i H(S_i)$  le gain d'information sur l'attribut  $a$  (déterminant mesure le gain obtenu suite au partitionnement selon l'attribut  $a$ .)

L'attribut ayant le plus gain d'information est celle sélectionnée le meilleur

La valeur avec entropie nulle est considérée comme feuille de l'arbre

20

# Arbre de décision: Algorithme ID3

## Algorithme générale

Avec ces précisions, l'**algorithme ID3** commence par la racine. Ensuite pour le nœud S en train d'être testé :

1. Calculer le gain d'information pour chaque attribut pas encore utilisé,
2. Choisir l'attribut **a** de gain d'information maximal,
3. Créer un test (décision) sur cet attribut dans le nœud S et générer les sous-nœuds  $S_1, \dots, S_m$  correspondant à la partition sur l'attribut choisi **a**,
4. Récurrence sur les nœuds qui viennent d'être créés.

## Sortie de la récursivité :

Tous les éléments de S sont dans la même classe ( $H(S)=0$ ) : S devient nœud feuille.

Pas d'attributs non utilisés : nœud feuille sur la classe majoritaire.

$S=\emptyset$  : nœud feuille sur la classe majoritaire du parent (ce cas est nécessaire pour le classement de nouveaux échantillons).

# Arbre de décision: Algorithme ID3\_exemple

❖ **Exemple :** Classification d'un ensemble de jours (J1, ..., J14) en deux catégories : « Oui », et « Non ».

Attributs	temps	température	humidité	vent
Valeurs possibles	ensoleillé, nuageux, pluvieux	chaude, fraîche, douce	haute, normale	oui, non

N°	temps	température	humidité	vent	jouer
1	ensoleillé	chaude	haute	non	non
2	ensoleillé	chaude	haute	oui	non
3	nuageux	chaude	haute	non	oui
4	pluvieux	douce	haute	non	oui
5	pluvieux	fraîche	normale	non	oui
6	pluvieux	fraîche	normale	oui	non
7	nuageux	fraîche	normale	oui	oui
8	ensoleillé	douce	haute	non	non
9	ensoleillé	fraîche	normale	non	oui
10	pluvieux	douce	normale	non	oui
11	ensoleillé	douce	normale	oui	oui
12	nuageux	douce	haute	oui	oui
13	nuageux	chaude	normale	non	oui
14	pluvieux	douce	haute	oui	non

la classe

# Arbre de décision: Algorithme ID3\_exemple

❖ On calcule la probabilité de chaque classe:

$$\diamond P(\text{jouer}=\text{oui}) = 9/14$$

$$\diamond P(\text{jouer}=\text{non}) = 5/14$$

❖ On calcule, ensuite, l'entropie de l'ensemble des données :

$$\diamond H(S) = - P(\text{jouer}=\text{oui}) * \log_2(P(\text{jouer}=\text{oui})) - P(\text{jouer}=\text{non}) * \log_2(P(\text{jouer}=\text{non}))$$

$$\diamond H(S) = - 9/14 * \log_2(9/14) - 5/14 * \log_2(5/14)$$

$$\diamond H(S) = 0.41 + 0.53 = 0.94$$

❖ Pour chaque caractéristique (temps, température, humidité, vent), on calcule le gain d'information.

23

# Arbre de décision: Algorithme ID3\_exemple

**Caractéristique "temps" :**

divide les données sur 3 sous ensembles :

On calcule la probabilité de chaque ensemble:

$$P(S_{\text{ensoleillé}}) = 5/14$$

$$P(S_{\text{nuageux}}) = 4/14$$

$$P(S_{\text{pluvieux}}) = 5/14$$

On calcule l'entropie de chaque ensemble:

$$H(S_{\text{ensoleillé}}) = - 2/5 * \log_2(2/5) - 3/5 * \log_2(3/5) = 0.971$$

$$H(S_{\text{nuageux}}) = - 4/4 * \log_2(4/4) - 0/4 * \log_2(0/4) = 0$$

$$H(S_{\text{pluvieux}}) = - 3/5 * \log_2(3/5) - 2/5 * \log_2(2/5) = 0.971$$

temps	jouer (oui)	jouer (non)	Total
ensoleillé	2	3	5
nuageux	4	0	4
pluvieux	3	2	5

- **Le gain d'information de la caractéristique “temps”:**

✓  $GI(S_{\text{temps}}) = H(S) - (P(S_{\text{ensoleillé}}) * H(S_{\text{ensoleillé}}) + P(S_{\text{nuageux}}) * H(S_{\text{nuageux}}) + P(S_{\text{pluvieux}}) * H(S_{\text{pluvieux}}))$

$$GI(S_{\text{temps}}) = 0.94 - (5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971)$$

24

$$GI(S_{\text{temps}}) = 0.247$$

# Arbre de décision: Algorithme ID3\_exemple

## ◆ Caractéristique "température" :

- ◆ divise les données sur 3 sous ensembles :
- ◆ On calcule la probabilité de chaque ensemble:
  - ◆  $P(S_{\text{chaude}}) = 4/14$
  - ◆  $P(S_{\text{douce}}) = 6/14$
  - ◆  $P(S_{\text{fraîche}}) = 4/14$

température	jouer (oui)	jouer (non)	Total
chaude	2	2	4
douce	4	2	6
fraîche	3	1	4

- ◆ On calcule l'entropie de chaque ensemble:

- ◆  $H(S_{\text{chaude}}) = -2/4 * \log_2(2/4) - 2/4 * \log_2(2/4) = 1$
- ◆  $H(S_{\text{douce}}) = -4/6 * \log_2(4/6) - 2/6 * \log_2(2/6) = 0.39 + 0.53 = 0.918$
- ◆  $H(S_{\text{fraîche}}) = -3/4 * \log_2(3/4) - 1/4 * \log_2(1/4) = 0.31 + 0.5 = 0.81$

## ◆ Le gain d'information de la caractéristique "température":

- ◆  $GI(S, \text{température}) = H(S) - (P(S_{\text{chaude}}) * H(S_{\text{chaude}}) + P(S_{\text{douce}}) * H(S_{\text{douce}}) + P(S_{\text{fraîche}}) * H(S_{\text{fraîche}}))$
- ◆  $GI(S, \text{température}) = 0.94 - (4/14 * 1 + 6/14 * 0.918 + 4/14 * 0.81)$
- ◆  $GI(S, \text{température}) = 0.029$

25

# Arbre de décision: Algorithme ID3\_exemple

## ◆ Caractéristique "humidité" :

- ◆ divise les données sur 2 sous ensembles :
- ◆ On calcule la probabilité de chaque ensemble:
  - ◆  $P(S_{\text{haute}}) = 7/14$
  - ◆  $P(S_{\text{normale}}) = 7/14$

humidité	jouer (oui)	jouer (non)	Total
haute	4	3	7
normale	6	1	7

- ◆ On calcule l'entropie de chaque ensemble:

- ◆  $H(S_{\text{haute}}) = -4/7 * \log_2(4/7) - 3/7 * \log_2(3/7) = 0.98$
- ◆  $H(S_{\text{normale}}) = -6/7 * \log_2(6/7) - 1/7 * \log_2(1/7) = 0.59$

## ◆ Le gain d'information de la caractéristique "humidité":

- ◆  $GI(S, \text{humidité}) = H(S) - (P(S_{\text{haute}}) * H(S_{\text{haute}}) + P(S_{\text{normale}}) * H(S_{\text{normale}}))$
- ◆  $GI(S, \text{humidité}) = 0.94 - (7/14 * 0.98 + 7/14 * 0.59)$
- ◆  $GI(S, \text{humidité}) = 0.152$

26

# Arbre de décision: Algorithme ID3\_exemple

## ◆ Caractéristique "vent" :

- ◆ divise les données sur 2 sous ensembles :
- ◆ On calcule la probabilité de chaque ensemble :
- ◆  $P(S_{oui}) = 6/14$
- ◆  $P(S_{non}) = 8/14$

vent	jouer (oui)	jouer (non)	Total
oui	3	3	6
non	6	2	8

- ◆ On calcule l'entropie de chaque ensemble:

$$\begin{aligned} \text{◆ } H(S_{oui}) &= -3/6 * \log_2(3/6) - 3/6 * \log_2(3/6) = 1 \\ \text{◆ } H(S_{non}) &= -6/8 * \log_2(6/8) - 2/8 * \log_2(2/8) = 0.811 \end{aligned}$$

## ◆ Le gain d'information de la caractéristique “vent”:

- ◆  $GI(S, vent) = H(S) - (P(S_{oui}) * H(S_{oui}) + P(S_{non}) * H(S_{non}))$
- ◆  $GI(S, vent) = 0.94 - (6/14 * 1 + 8/14 * 0.811)$
- ◆  $GI(S, vent) = 0.892$

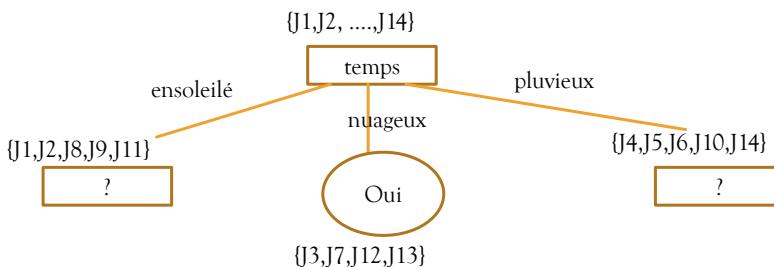
27

# Arbre de décision: Algorithme ID3\_exemple

## ◆ Le gain d'information des toutes les caractéristiques :

temps	température	humidité	vent
GI	0.247	0.029	0.152

- ◆ La première caractéristique à vérifier dans l'arbre sera “temps”.
- ◆ Comme l'entropie du temps étant “nuageux” est 0, cet ensemble contient des échantillons de la même classe → Feuille.



28

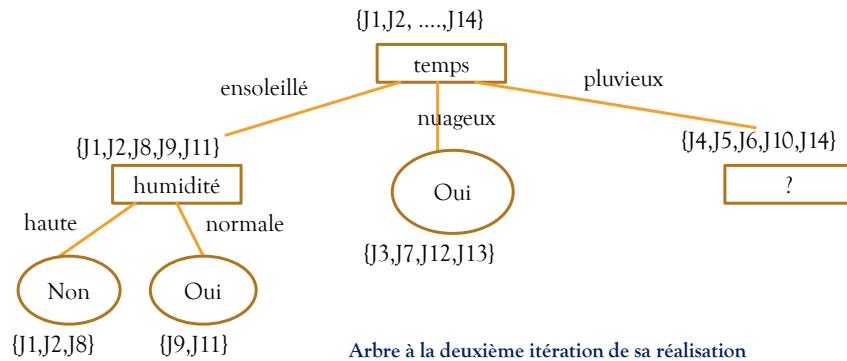
Arbre à la première itération de sa réalisation

# Arbre de décision: Algorithme ID3\_exemple

❖ La procédure continue sur les sous-noeuds ainsi obtenus, jusqu'à ce qu'on obtienne des noeuds uniformes (noeuds purs).

- ❖  $GI(S_{\text{Ensoleillé}}, \text{Température}) = 0.571$
- ❖  $GI(S_{\text{Ensoleillé}}, \text{Humidité}) = 0.971$
- ❖  $GI(S_{\text{Ensoleillé}}, \text{Vent}) = 0.019$

Le plus grand gain est pour **Humidité**.  $\rightarrow$  Le gain est égal à l'entropie de  $S_{\text{Ensoleillé}}$ .  $\rightarrow$  Tous les fils de Humidité donneront une étiquette.



29

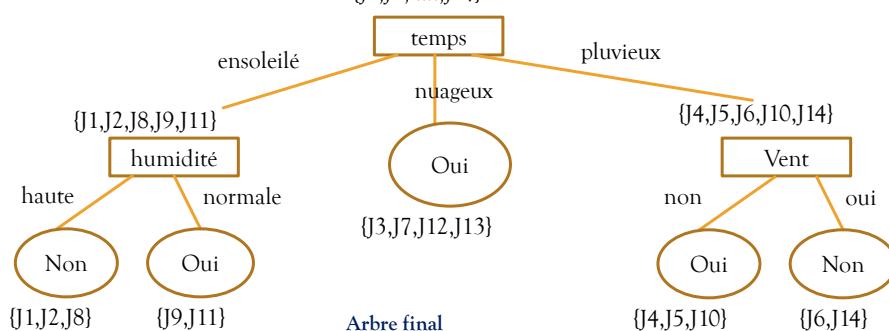
Arbre à la deuxième itération de sa réalisation

# Arbre de décision: Algorithme ID3\_exemple

❖ La procédure continue sur les sous-noeuds ainsi obtenus, jusqu'à ce qu'on obtienne des noeuds uniformes.

- ❖  $GI(S_{\text{Pluvieux}}, \text{Température}) = 0.019$
- ❖  $GI(S_{\text{Pluvieux}}, \text{Humidité}) = 0.019$
- ❖  $GI(S_{\text{Pluvieux}}, \text{Vent}) = 0.971$

Le plus grand gain est pour **Vent**.  $\rightarrow$  Le gain est égal à l'entropie de  $S_{\text{Pluvieux}}$ .  $\rightarrow$  Tous les fils de Humidité donneront une étiquette.



30

Arbre final

# ID3: Limites

## 1. Solution globale non garantie

- ID3 utilise une approche gloutonne : à chaque étape, il choisit l'attribut avec le **meilleur gain d'information local**. Cela ne garantit pas l'**optimalité globale** de l'arbre final.  
=> Amélioration possible : backtracking, mais cela augmente considérablement la complexité.

## 2. Peu efficace avec des attributs numériques continus

- ID3 a été conçu pour des attributs catégoriels.
- Ne gère pas directement les attributs numériques → nécessite de discréteriser les valeurs continues en intervalles fixes.

## 3. Améliorations :

- Algorithmes comme C4.5 (successeur d'ID3) gèrent les attributs continus

31

# C4.5 (Iterative Dichotomiser 4.5)

## Critère de Gini : Algorithme CART (Classification And Regression Trees)

- Suppose des attributs continus
- Suppose plusieurs valeurs de division pour chaque attribut
- Il répond aux questions suivantes :
  1. Comment choisir la variable à segmenter parmi les variables explicatives disponibles ?
  2. Lorsque la variable est continue, comment déterminer le seuil de coupe ?
  3. Comment déterminer la bonne taille de l'arbre ?

### Caractéristiques

- Le critère de division est le gain d'information normalisé maximal (différence d'entropie avant et après la division)
- Chaque attribut peut avoir un poids (coût)
- Traitement de variables continues en cherchant des seuils qui maximise le gain d'information
- Traitement de valeurs manquantes
- Étape d'élagage après la création pour remplacer des branches inutiles par des feuilles

32

# Indice de gini

❖ Pour un nœud **p** donnée :  $Gini(p) = 1 - \sum_{k=1}^c (P(k|p))^2$

- ❖ Plus l'indice de Gini est **bas**, plus le nœud est **pur**.
- ❖ L'attribut **a** (le test) le plus discriminant doit maximiser le gain du nœud **p** :

$$Gain(S,a) = i(\text{avant séparation}) - [P_{\text{fils1}} * i(\text{fils1}) + \dots + P_{\text{filsn}} * i(\text{filsn})]$$

**i** = la fonction pour mesurer le degré de désordre (Entropie, Gini)

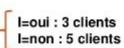
33

# Indice de gini

## ❖ Exemple :

Client	M	A	R	E	I
1	moyen	moyen	village	oui	oui
2	élevé	moyen	bourg	non	non
3	faible	âgé	bourg	non	non
4	faible	moyen	bourg	oui	oui
5	moyen	jeune	ville	oui	oui
6	élevé	âgé	ville	oui	non
7	moyen	âgé	ville	oui	non
8	faible	moyen	village	non	non

Indice de Gini avant séparation au NIVEAU DE LA RACINE :

8 clients 

$$IG(\text{avant séparation}) = 1 - ((3/8)^2 + (5/8)^2) = 0.46875$$

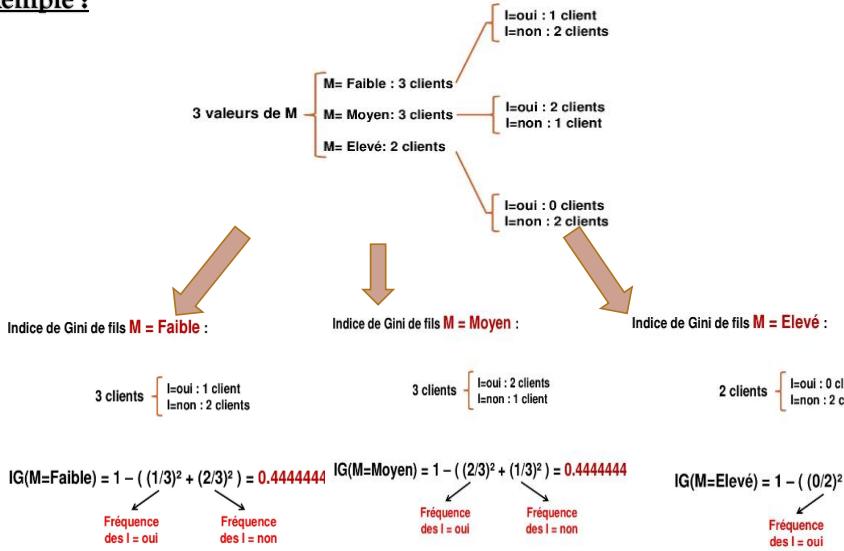
Fréquence des I = oui      Fréquence des I = non

34

# Indice de gini

Indice de Gini de la variable M (Moyenne des montants sur le compte client) :

## Exemple :

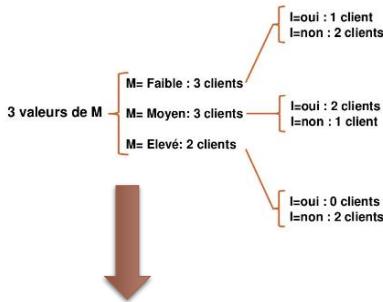


35

# Indice de Gini

Indice de Gini de la variable M (Moyenne des montants sur le compte client) :

## Exemple :



36

# Gain d'information : limites

❖ **Problématique:** Le gain d'information a tendance à favoriser les attributs qui possèdent un grand nombre de valeurs différentes (attributs multivalués). En effet, ces attributs divisent les données en beaucoup de petits sous-ensembles qui peuvent être purement classés, ce qui donne un gain artificiellement élevé, même si cette séparation ne généralise pas bien. (**Lorsqu'un attribut a plusieurs valeurs possibles, son gain peut être très élevé, car il classifie parfaitement les objets**)

❖ **Exemple :** un attribut comme "numéro de sécurité sociale" séparerait chaque individu de manière unique, mais cela ne serait pas pertinent pour la classification.



Pour corriger ce problème, on utilise une autre mesure le **rapport de gain** (**ratio de gain**) qui pénalise les attributs multivalués

37

# L'algorithme C4.5

- ❖ L'algorithme **C4.5** utilise une extension du gain d'information connue par **rapport de gain** pour sélectionner le meilleur attribut.
- ❖ Le rapport de gain fait face au problème de biais en normalisant le gain d'information à l'aide de l'information de division.
- ❖ La proportion de gain (gain d'information proportionnel) divise le gain d'information par une mesure proportionnelle à la taille de la partition générée par les valeurs d'un attribut --> (s'il y a beaucoup de valeurs, le dénominateur sera plus large).

$$\text{Gain Ratio } (S, A) = \frac{\text{Gain}(S, A)}{\text{Split Info}(S, A)}$$

$$\text{Split Info } (S, A) = - \sum_{i=1}^k \frac{|S_i|}{|S|} \times \log_2 \frac{|S_i|}{|S|}$$

avec:

- $S$ : Ensemble d'exemples.  $|S|$ : nombre d'exemples valant  $A_i$  pour l'attribut  $A$ .
- $k$ : nombre de valeurs de l'attribut  $A$

38



On sélectionne l'attribut offrant le plus de ratio de gain.

# L'algorithme C4.5

## ❖ Exemple :

N°	temps	température	humidité	vent	jouer
1	ensoleillé	chaude	haute	non	non
2	ensoleillé	chaude	haute	oui	non
3	nuageux	chaude	haute	non	oui
4	pluvieux	douce	haute	non	oui
5	pluvieux	fraîche	normale	non	oui
6	pluvieux	fraîche	normale	oui	non
7	nuageux	fraîche	normale	oui	oui
8	ensoleillé	douce	haute	non	non
9	ensoleillé	fraîche	normale	non	oui
10	pluvieux	douce	normale	non	oui
11	ensoleillé	douce	normale	oui	oui
12	nuageux	douce	haute	oui	oui
13	nuageux	chaude	normale	non	oui
14	pluvieux	douce	haute	oui	non

$$\begin{aligned}
 GI(S, \text{temps}) &= H(S) - (P(S_{\text{ensoleillé}}) * \\
 &H(S_{\text{ensoleillé}}) + P(S_{\text{nuageux}}) * H(S_{\text{nuageux}}) + \\
 &P(S_{\text{pluvieux}}) * H(S_{\text{pluvieux}})) \\
 &= 0.247
 \end{aligned}$$

$$\text{Split Info (S, Temps)} = - \sum_{i=1}^3 \frac{|S_i|}{|S|} \times \log_2 \frac{|S_i|}{|S|}$$

$$\begin{aligned}
 \text{Split Info (S, Temps)} &= -5/14 * \log_2 5/14 - 4/14 \\
 &\quad * \log_2 4/14 - 5/14 * \log_2 5/14 \\
 &= 1.576
 \end{aligned}$$

$$\text{Gain Ratio (S, Temps)} = \frac{0.247}{1.576} = 0.157$$

39

# Arbre de décision

## ❖ C4.5 (Iterative Dichotomiser 4.5)

- ❖ Extension de l'algorithme ID3 par Ross Quinlan, utilisé pour le classement.
- ❖ Parmi les améliorations :
  - ❖ Transformer les caractéristiques continues (numériques) en caractéristiques nominales dynamiquement.
  - Élagage des arbres après la création : C4.5 utilise une procédure qui permet de transformer en feuilles des sous-arbres qui ne contribuent à un meilleur résultat (score « accuracy ») de l'apprentissage.
  - ❖ Limitation de la profondeur de l'arbre par la taille minimale du nœuds (paramètre de construction).
  - ❖ Chaque attribut peut avoir un poids (coût).

## ❖ C5.0 (Iterative Dichotomiser 5.0)

- ❖ Extension commerciale de C4.5, toujours par Ross Quinlan.
- ❖ Vitesse et utilisation mémoire.
- ❖ Optimisé pour des bases de données de très grande taille.
- ❖ Arbres plus petits et pondération des cas et erreurs de classement.

40

# Algorithme CART

❖ **CART** (Classification and Regression Trees) : similaire à celui de C4.5 avec quelques différences :

- ❖ Il supporte la régression.
- ❖ Il utilise d'autres critères pour sélectionner la meilleure caractéristique. Il essaye de minimiser une fonction de coût.
- ❖ Il utilise le pré-élagage en utilisant un critère d'arrêt.
- ❖ Il génère des arbres de décision binaires.

41

# Élagage

- ❖ Contrôler la complexité du nombre des branches et des feuilles pour réaliser un arbre de décision.
- ❖ Minimiser la taille de l'arbre.
- ❖ Trouver le nombre optimale  $k$  de nœuds.
- ❖ Une méthode régularisation ou de sélection des modèles.
  
- ❖ Techniques d'élagage :
  - ❖ Pré-élagage.
  - ❖ Post-élagage.

42

# Élagage

## Pré-élagage:

L'idée : arrêter la construction de l'arbre avant qu'il ne devienne trop complexe.

- ◊ Arrêter de diviser un nœud quand la pureté des points qui domine est non parfaite mais suffisante.
- ◊ Arrêter quand il y a une classe majoritaire dans le nœud.
- ◊ Utiliser un seuil pour détecter une classe dominante.
- ◊ Avantages:
  - ◊ Rapide
- ◊ Inconvénients :
  - ◊ Arrêter la construction de l'arbre peut donner un arbre sous optimal.

43

# Élagage

## Post élagage

L'idée : on construit d'abord l'arbre entier, puis on supprime les branches inutiles.

- ◊ Finir la construction de l'arbre.
- ◊ Simplifier l'arbre en remontant des feuilles vers la racine pour trouver ou élaguer.
- ◊ Utiliser des critères de qualité qui mesure un compromis l'erreur obtenue et la complexité de l'arbre.

Utiliser un ensemble de validation pour mesurer l'erreur à chaque nœud.

- ◊ Avantages:
  - ◊ arbre souvent plus performant
- ◊ Inconvénients :
  - ◊ plus coûteux en temps.

44

## Partie II: K plus proche voisins

45

### K plus proches voisins

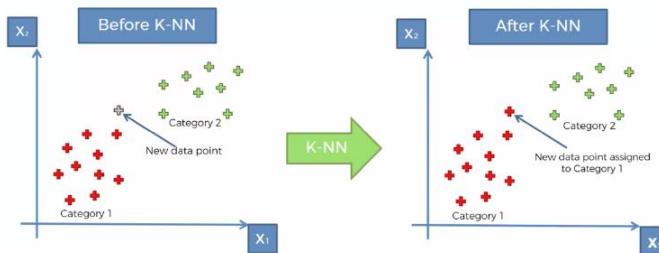
- Algorithme d'apprentissage supervisé.
- L'algorithme des **k plus proches voisins** (**k-ppv**) ou en anglais **k-Nearest Neighbors** (**k-nn**) est un algorithme de classification et de régression.
- Son fonctionnement peut être assimilé à l'analogie suivante
  - “*dis moi qui sont tes voisins,je te dirais qui tues...*”.
- K-NN ne va pas calculer un modèle prédictif à partir d'un *Training Set*
- Il n'existe pas de phase d'apprentissage proprement dite.
- K-NN **se base sur le jeu de données** pour produire un résultat.

46

# K plus proches voisins

À partir d'un ensemble  $E$  de données labellisées, k-ppv consiste à affecter, à un point à classer (n'appartenant pas à  $E$ ), la classe majoritaire de ses  $K$  plus proches voisins:

- ✓ En effet, pour une observation, qui ne fait pas parti du jeu de données, qu'on souhaite prédire, l'algorithme va chercher les  **$K$  instances du jeu de données les plus proches de notre observation.**
- ✓ Ensuite pour ces  $K$  voisins, l'algorithme se basera sur leurs variables de sortie (output variable) pour calculer la valeur de la variable  $y$  de l'observation qu'on souhaite prédire.



47

## K plus proches voisins: principe

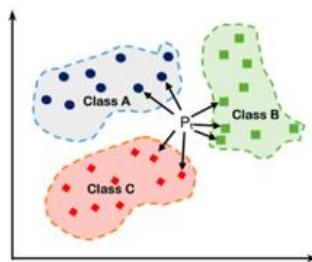
On suppose que l'ensemble  $E$  contient  $n$  données labellisées et  $u$ , une autre donnée n'appartenant pas à  $E$  qui ne possède pas de label. Soit  $d$  une fonction qui renvoie la distance (qui reste à choisir) entre la donnée  $u$  et une donnée quelconque appartenant à  $E$ . Soit un entier  $k$  inférieur ou égal à  $n$ .

Le principe de l'algorithme de k-plus proches voisins est le suivant:

On calcule les distances entre la donnée  $u$  et chaque donnée appartenant à  $E$  à l'aide de la fonction  $d$ .

On choisit les  $k$  données du jeu de données  $E$  les plus proches de  $u$ .

On attribue à  $u$  la classe qui est la plus fréquente parmi les  $k$  données les plus proches.



48

# K plus proches voisins: principe

## K-NN pour la Régression :

- Lorsqu'on utilise K-NN pour la régression, la prédiction pour un nouveau point de données est la **moyenne** des valeurs 'y' (la variable cible) des K observations les plus proches dans les données

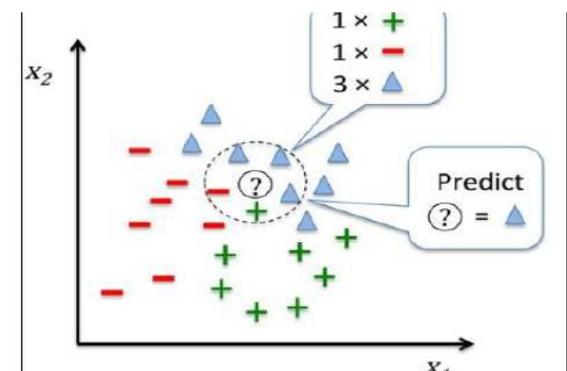
## K-NN pour la Classification :

- Lorsqu'on utilise K-NN pour la classification, la prédiction pour un nouveau point de données est déterminée par le **mode** (**la classe la plus fréquente**) des valeurs 'y' (les étiquettes de classe) parmi ses K voisins les plus proches dans les données d'entraînement.

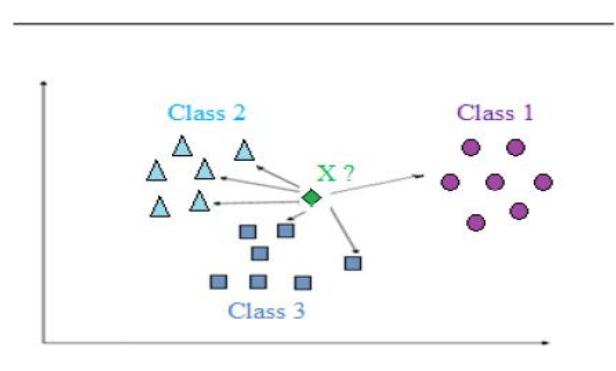
49

# K plus proches voisins: principe

## ❖ Exemples :



- Il y a 1 voisin de la classe '+', 1 voisin de la classe '-' et 3 voisins de la classe 'triangle'. Cela signifie que K=5 dans cet exemple .
- La classe 'triangle' est la plus fréquente (3 occurrences), La prédiction est donc "Predict (?) = triangle".



- Les voisins les plus proches soient principalement des triangles (Classe 2) et des carrés (Classe 3), avec potentiellement quelques cercles (Classe 1) plus éloignés.

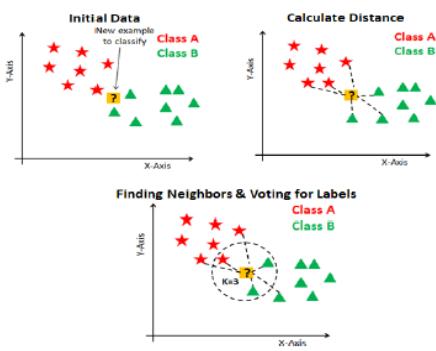
50

# K plus proches voisins: algorithme

## Début Algorithme

### Données en entrée :

- un ensemble de données D .
- une fonction de définition distance  $d$ .
- Un nombre entier K



Pour une nouvelle observation X dont on veut prédire sa variable de sortie y Faire :

1. Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D
2. Retenir les K observations du jeu de données D les proches de X en utilisation le fonction de calcul de distance  $d$
3. Prendre les valeurs de y des K observations retenues :
  1. Si on effectue une régression, calculer la moyenne (ou la médiane) de y retenues
  2. Si on effectue une classification , calculer le mode de y retenues
4. Retourner la valeur calculé dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X.

51

## Fin Algorithme

# Notion de distance

- Le choix de la distance est primordial au bon fonctionnement de la méthode.
- Les distances les plus simples permettent d'obtenir des résultats satisfaisants (lorsque c'est possible).
- Propriétés de la distance:
  - ✓ Réflexivité:  $d(A,B)=0 \iff A = B$
  - ✓ Non négativité:  $d(A, B) \geq 0$
  - ✓ Symétrie:  $d(A,B) = d(B,A)$
  - ✓ Inégalité triangulaire:  $d(A,B) \leq d(A,C) + d(C,B)$

52

# Calcul de similarité

L'algorithme **k-NN** nécessite une **fonction de calcul de distance** pour mesurer la similarité entre deux observations.

- Plus deux points sont proches, plus ils sont considérés comme similaires, et inversement.

**Fonctions de distance couramment utilisées :**

- ✓ **Distance euclidienne** : adaptée aux données numériques continues.
- ✓ **Distance de Manhattan** : somme des valeurs absolues des différences, utile pour les données en grille.
- ✓ **Distance de Minkowski** : généralisation des distances euclidienne et de Manhattan.
- ✓ **Distance de Jaccard** : mesure la dissimilarité entre deux ensembles (utile pour des attributs binaires).
- ✓ **Distance de Hamming** : compte les différences entre deux chaînes de caractères (ou vecteurs binaires).

**Choix de la distance :**

Le choix de la fonction de distance dépend du **type de données manipulé** (numérique, catégoriel,<sup>53</sup> binaire, etc.).

# Fonction de calcul de distance

- Pour les **données quantitatives de même nature** (par exemple : salaires, taille, montant d'un panier électronique, etc.), la **distance euclidienne** est un excellent choix car elle mesure la proximité géométrique dans un espace continu.
- En revanche, la **distance de Manhattan** est plus appropriée lorsque les données sont **hétérogènes** ou de **types différents** (exemples : sexe, longueur, poids). Elle est moins sensible aux grandes variations et plus robuste dans des cas mixtes.

Il est **inutile d'implémenter soi-même** les fonctions de distance :

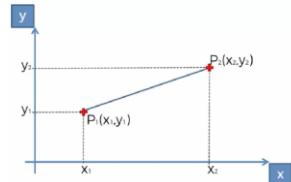
➔ Les bibliothèques de Machine Learning comme **Scikit-learn** (Python) proposent déjà des fonctions optimisées pour ces calculs.

# Fonction de calcul de distance

## La distance euclidienne

Distance qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points:

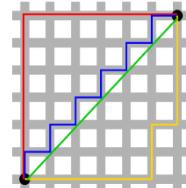
$$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$



## La distance Manhattan

La **distance de Manhattan** calcule la **somme des valeurs absolues** des différences entre les coordonnées de deux points.

$$D_m(x, y) = \sum_{i=1}^k |x_i - y_i|$$



55

# Fonction de calcul de distance

## La distance Hamming

La **distance de Hamming** est utilisée principalement pour des données binaires ou des chaînes de caractères de même longueur.

Elle permet de quantifier la différence entre deux séquences de symboles. C'est une distance au sens mathématique du terme. A deux suites de symboles de même longueur, elle associe le nombre de positions où les deux suites diffèrent.

$$D(x, y) = \sum_{i=1}^n \mathbf{1}(x_i \neq y_i)$$

### Exemples :

Considérons les mots binaires suivants : **a** = (0001111) et **b** = (1101011)

ALORS  $d = 1+1+0+0+1+0+0 = 3 \rightarrow$  La distance entre **a** et **b** est égale à 3 car 3 bits diffèrent.

La distance de Hamming entre "1011101" et "1001001" est 2.

La distance de Hamming entre "2143896" et "2233796" est 3.

La distance de Hamming entre "ramer" et "cases" est 3.

56

# Comment choisir la valeur de K

- Le choix de la valeur de K à utiliser pour effectuer une prédiction avec k-NN dépend du jeu de données.
  - En règle générale, moins on utilisera de voisins (**un nombre K petit**) plus on sera sujette au sous apprentissage (underfitting).
  - Plus on utilise de voisins (**un nombre K grand**) plus, sera fiable notre prédiction. Cependant , le voisinage peut contenir des objets de plusieurs classes.
- ✓ Si on utilise K nombre de voisins avec  $K=N$  et N étant le nombre d'observations, on risque d'avoir du overfitting et par conséquent un modèle qui se généralise mal sur des observations qu'il n'a pas encore vu:
- ✓ Le modèle prédit toujours la classe majoritaire,
  - ✓ Ce qui donne une précision trompeuse

57

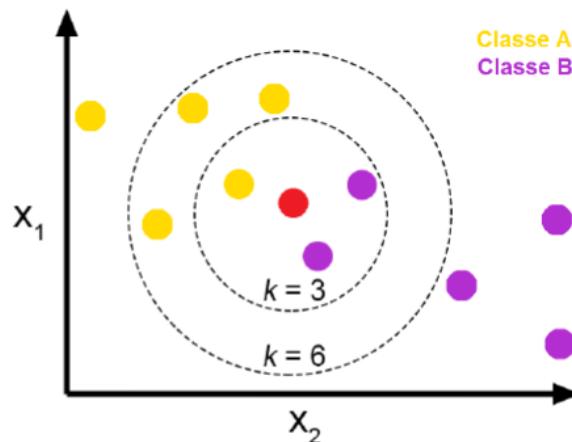
# Comment choisir la valeur de K

**Le choix de K est donc dépendant des données en entrée :**  
 s'il y a une grande dispersion (grand écart-type) → K « grand »  
 si les données sont entrelacées et que le point à classer est dans le nuage → K « moyen »  
 si le point à classer est en dehors des nuages on peut choisir → K « petit »

- Choix du nombre k de voisins déterminé par utilisation d'un ensemble test ou par validation croisée.
- Une heuristique fréquemment utilisée est de prendre k égal au nombre d'attributs plus 1.

58

# Comment choisir la valeur de K



- Pour  $K=3$ , la classe prédictive pour le point rouge serait la Classe B
- Pour  $K=6$ , la classe prédictive pour le point rouge serait la Classe A

59

## K-NN exemple

Client	Age	Revenu	Nombre cartes de crédit	Classe (Réponse)
Mohamed	35	350	3	Non
Ali	22	500	2	Oui
Samia	63	2000	1	Non
Sami	59	1700	1	Non
Meriem	25	400	4	Oui
Lotfi	37	500	2	?

60

# K-NN exemple

Client	Age	Revenu	Nombre cartes de crédit	Classe (Réponse)	Distance(Client, Lotfi)
Mohamed	35	350	3	Non	Sqrt((35-37) <sup>2</sup> +(350-500) <sup>2</sup> +(3-2) <sup>2</sup> )=150.01
Ali	22	500	2	Oui	Sqrt((22-37) <sup>2</sup> +(500-500) <sup>2</sup> +(2-2) <sup>2</sup> )= 15
Samia	63	2000	1	Non	Sqrt((63-37) <sup>2</sup> +(2000-500) <sup>2</sup> +(1-2) <sup>2</sup> )=1500.22
Sami	59	1700	1	Non	Sqrt((59-37) <sup>2</sup> +(1700-500) <sup>2</sup> +(1-2) <sup>2</sup> )=1200.2
Meriem	25	400	4	Oui	Sqrt((25-37) <sup>2</sup> +(400-500) <sup>2</sup> +(4-2) <sup>2</sup> )=100.74
Lotfi	37	500	2	?	

61

# K-NN exemple

Client	Age	Revenu	Nombre cartes de crédit	Classe (Réponse)	Distance(Client, Lotfi)
Mohamed	35	350	3	Non	Sqrt((35-37) <sup>2</sup> +(350-500) <sup>2</sup> +(3-2) <sup>2</sup> )=150.01
Ali	22	500	2	Oui	Sqrt((22-37) <sup>2</sup> +(500-500) <sup>2</sup> +(2-2) <sup>2</sup> )= 15
Samia	63	2000	1	Non	Sqrt((63-37) <sup>2</sup> +(2000-500) <sup>2</sup> +(1-2) <sup>2</sup> )=1500.22
Sami	59	1700	1	Non	Sqrt((59-37) <sup>2</sup> +(1700-500) <sup>2</sup> +(1-2) <sup>2</sup> )=1200.2
Meriem	25	400	4	Oui	Sqrt((25-37) <sup>2</sup> +(400-500) <sup>2</sup> +(4-2) <sup>2</sup> )=100.74
<b>Lotfi</b>	<b>37</b>	<b>500</b>	<b>2</b>	<b>Oui</b>	<b>?</b>

62

Il faut normaliser puis calculer les distances!

→ on ramène toutes les valeurs de la variable entre 0 et 1, tout en conservant les distances entre les valeurs.

Formule :

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \in [0, 1]$$

# K-NN exemple

## ❖ Avantages

- ❖ Simple et facile à implémenter et à utiliser.
- ❖ Compréhensible : La classification est facile à expliquer.
- ❖ Robuste aux données bruitées.
- ❖ Des applications intéressantes.
- ❖ Efficace pour des classes réparties

## ❖ Inconvénients

- ❖ Nécessité de capacité de stockage et de puissance de calcul.
- ❖ Pas de modèle construit.
- ❖ Prend du temps pour classer un nouvel objet: Comparaison des distances du nouvel objet avec tous les autres de l'ensemble d'apprentissage.
- ❖ Choix du K.

63

# K-NN exemple

## ❖ Plusieurs extensions de K plus proches voisins:

- ❖ Système de classification hybrides
- ❖ Fuzzy K-NN
- ❖ Belief K-NN
- ❖ ...

64