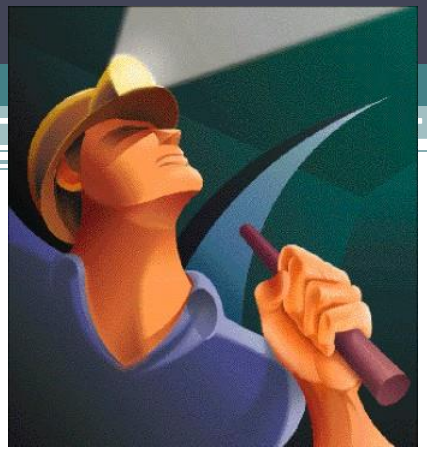
Arbres de décision

©Lotfi Ben Romdhane, *Ph.D.*
ISITCom / U. de Sousse / Tn

3DNI



Sommaire

- Concepts de base
- Algorithme d'induction des arbres de décision
- Règles de décisions
- Elagage de l'arbre
- Discussion

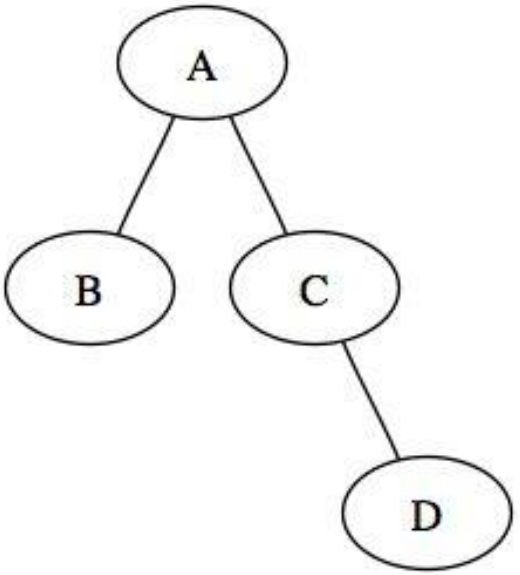
Historique

- Fin des années 1970 et début des années 1980, ***J. Ross Quilann*** a développé un algorithme pour la génération des arbres de décision et appelé **ID3** (*Iterative Dichotomizer*)
- Après, il a proposé **C4.5** et **C5** (version commercialisées)
- En 1984, **CART** (*Classification and Regression Trees*) par Breinman, *et al*

RULEQUEST RESEARCH
data mining tools

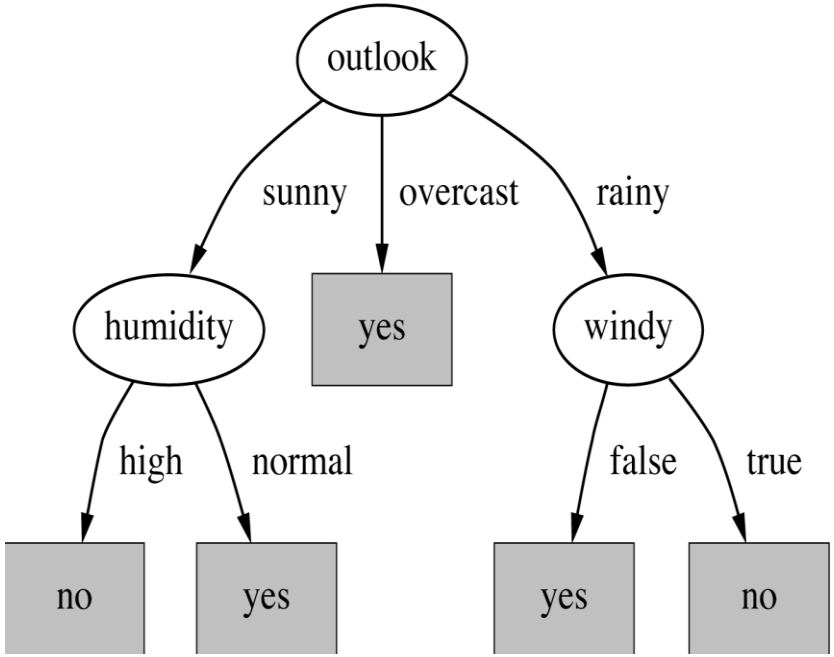
Concepts de base (1)

- Un **arbre** est une structure de données définie comme suit
  - une liste de nœuds ayant une structure hiérarchique avec la relation père/fils
  - un *nœud racine :* pas de père
  - plusieurs *nœuds feuilles*: pas de fils - appelés aussi *nœuds externes*
  - nœuds internes: non feuilles



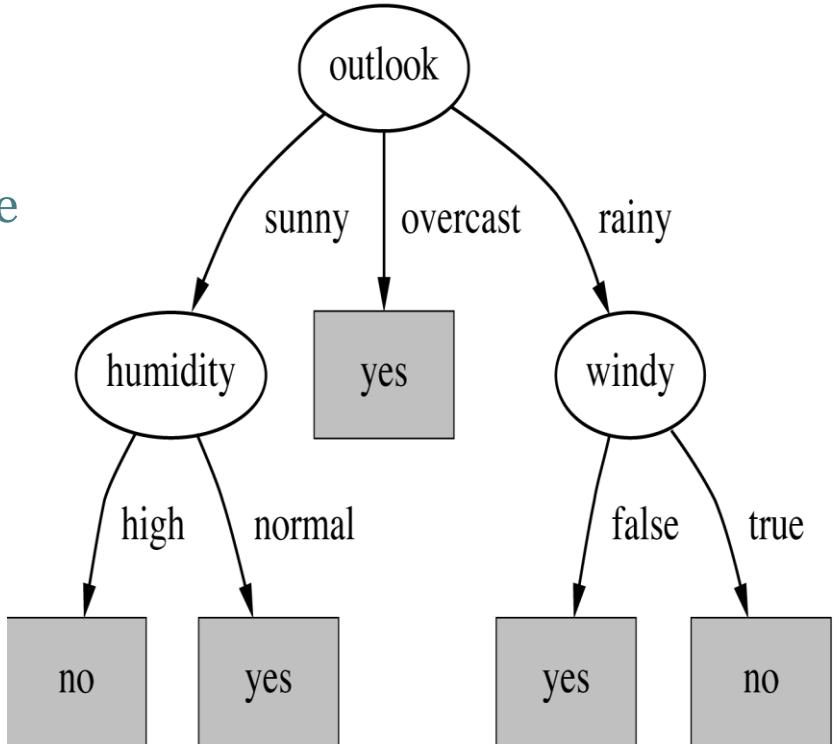
Concepts de base (2)

- Un **arbre de décision** est un arbre dans lequel
  - chaque **nœud interne** modélise un **test sur un attribut**
  - chaque *branche* de l'arbre représente **un résultat de ce test**
  - chaque **nœud externe** (feuille) représente une **classe** (catégorie)
- Dans l'arbre suivant, on a deux classes {Play Golf (Yes), Not play Golf (No)}



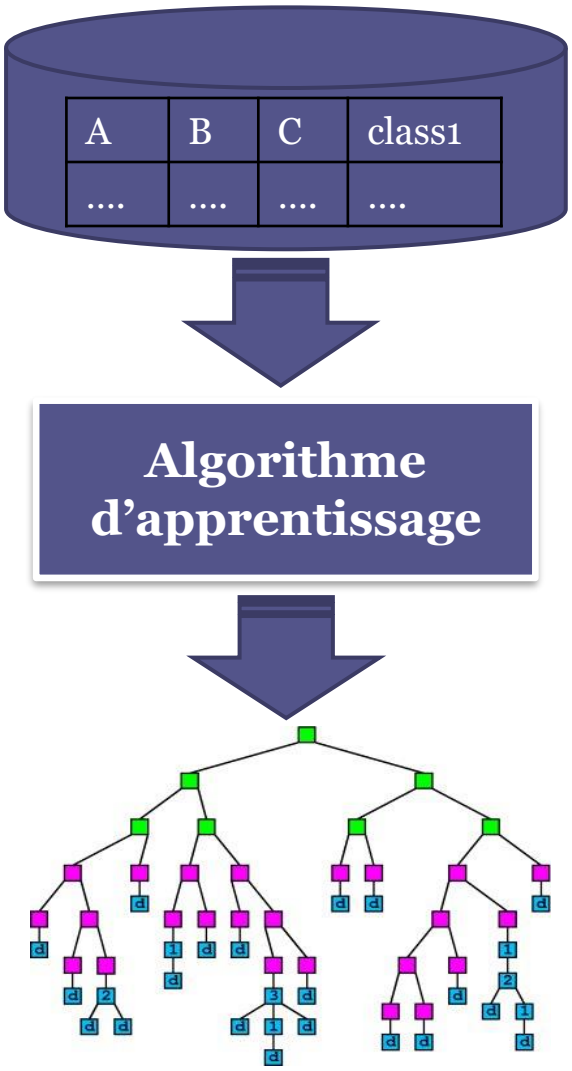
Concepts de base (3)

- un **arbre de décision est** utile pour effectuer des prédictions sur des données *X* non étiquetées (pour lesquelles, les classes sont encore inconnues)
  - un chemin est construit en partant de la racine jusqu'à un nœud feuille (qui représente la classe prédite)
  - *X = (outlook = sunny, humidity=normal, windy=true)* : classe prédite = "YES" (play golf)
  - *X = (outlook = rainy, humidity=normal, windy=true)* : classe prédite = "NO" (not to play golf)



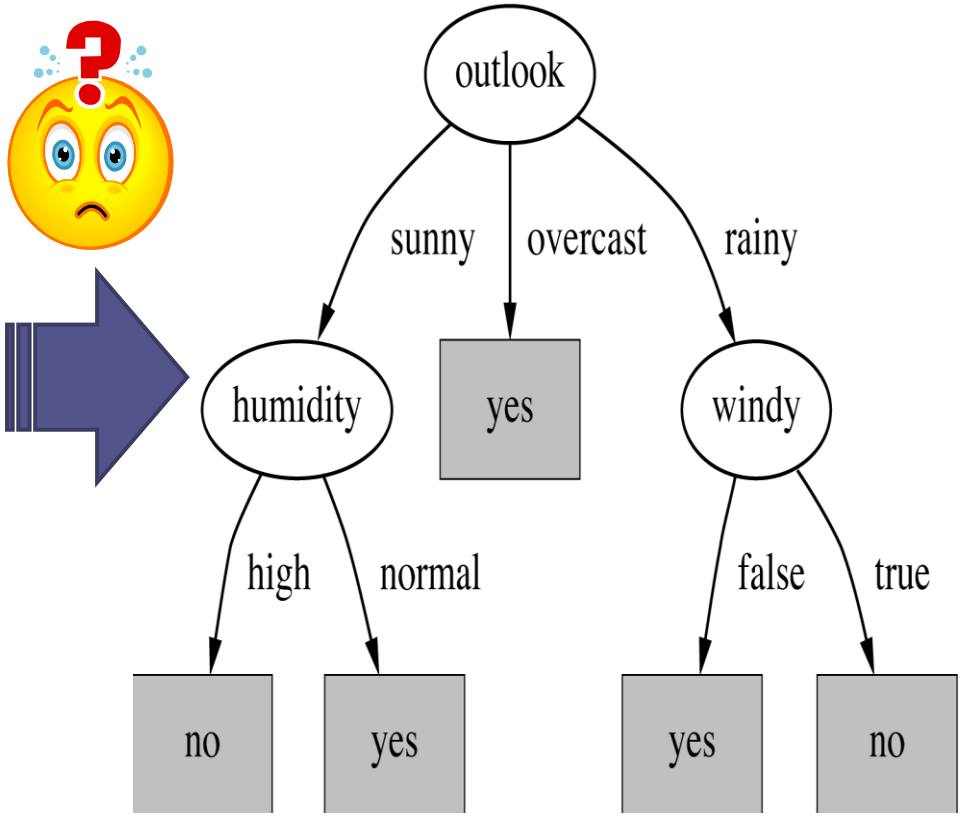
Concepts de base (4)

- Un arbre de décision est *apprise* à partir d'un ensemble de données *D*
- Chaque donnée dans D est un tuple : ensemble de valeurs et *la classe associée*
- *D = {X1, X2, …, Xn}* est un ensemble d'apprentissage avec *X = (val1, val2, …, valn, classe)*



Concepts de base (5)

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No



Algorithme d'induction (1)

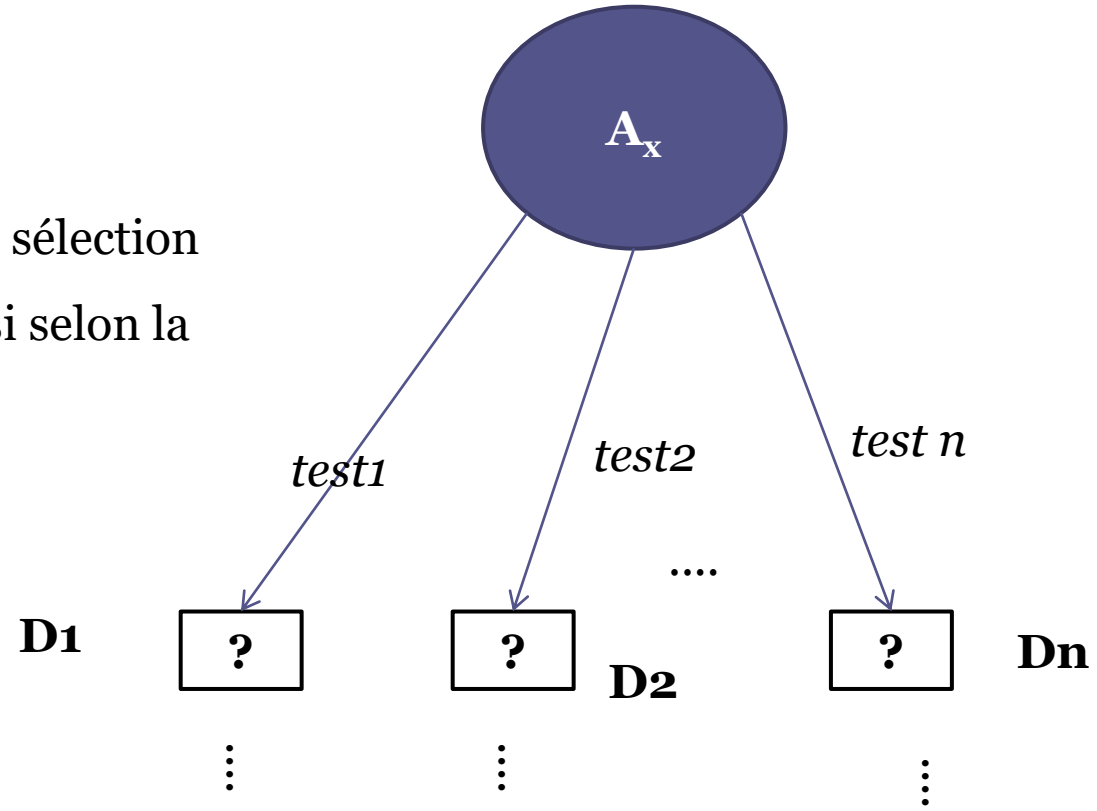
- La majorité des algorithmes adoptent une approche « ***glouton*** » (*greedy method*)
  - Un **algorithme glouton** suit le principe de faire, *étape par étape, un choix optimum local*, dans l'espoir d'obtenir un résultat optimum global
- L'arbre de décision est construit de la racine vers les feuilles (*top-down*) en suivant une approche «***diviser pour régner*** »
  - une technique algorithmique consistant à diviser un problème de grande taille en plusieurs sous-problèmes analogues, mais de tailles inférieure
  - l'étape de subdivision est appliquée récursivement.

Algorithme d'induction (2)

- La stratégie d'induction est la suivante
  1. choisir un attribut de test (division) selon une **méthode de sélection**
  2. créer un nœud et l'étiqueter avec cet attribut
  3. à partir de ce nœud, développer une branche *pour chaque test possible* pour cet attribut
  4. diviser l'ensemble de données D selon chaque test possible (branche)
  5. Récursivement, construire l'arbre

Algorithme d'induction (3)

- **D** = ensemble de données
- **A** = ensemble d'attributs
- **S**= méthode (algorithme) de sélection
- $\mathbf{A_x}$ = meilleure attribut choisi selon la méthode *S*



Algorithme d'induction (4)

Algorithme *Generer_Arbre_Decision (D,A,S)*

- *entrée*
  - ***D*** : un ensemble de données
  - ***A*** : ensemble d'attributs décrivant les données
  - ***S*** : une méthode (algorithme) de sélection du meilleur attribut
- *sortie*
  - ***T*** : un arbre de décision T

Algorithme d'induction (5)

DEBUT

1. créer un nœud N
2. **Si** tous les tuples de D admettent la même classe (D est un ensemble pur) **alors**
   - N est un nœud terminal (feuille)
   - étiqueter N avec la classe de D
3. **Si** (A=∅) **alors**
   - N est un nœud terminal (feuille)
   - étiqueter N soit avec la classe majoritaire de D; soit avec une distribution de classes ($classe_i$, $fréquence_i$)

Algorithme d'induction (6)

4. Trouver le meilleure attribut de division en appliquant la méthode S
   - $a_{best} \leftarrow \mathrm{S}(D, A)$
   - trouver c_{best} , le meilleur critère de test pour a_{best} parce que le même attribut peut avoir plusieurs critères de test possibles
5. Si le meilleur attribut est discret alors réduire l'ensemble des attributs A
   - $\mathrm{A} \leftarrow \mathrm{A} \backslash \{a_{best}\}$

Algorithme d'induction (7)

6. *N* est un nœud interne (non-feuille)
   - étiqueter N avec l'attribut de test a_{best} et son meilleur critère de test c_{best}
7. **Pour** chaque test possible t_j pour c_{best} **faire**
   - calculer D_j, l'ensemble des tuples de *D* qui passent le test t_j
   - **Si** (Dj=∅) **alors**
     - N est un nœud feuille
     - étiqueter N avec la classe majoritaire dans *D* ou une distribution de classes
   - **Sinon** *Générer_Arbre_Decision*(D_j, *A*, *S*)

FIN.

Algorithme d'induction (8)

- Le rôle de la méthode de sélection ***S*** est de choisir le meilleur attribut de division avec le meilleur critère de test
- Intuitivement, le meilleur attribut de division est celui qui permet de générer *les ensembles les plus purs*
  - génère un arbre de hauteur minimale
  - permet d'atteindre rapidement la décision (feuille)

Algorithme d'induction (9)

- L'algorithme d'induction s'arrête sur l'une des conditions suivantes
  1. L'ensemble de données *D* est pure: tous les tuples sont de la même classe
  2. L'ensemble de données *D* est vide : plus de tuples pour la branche de test correspondante
  3. L'ensemble d'attributs *A* est vide
     - créer un nœuds feuille et l'annoter avec la classe majoritaire ou une distribution de classes

Algorithme d'induction (10)

- |D| est le nombre de tuples
- |A| est le nombre d'attributs
- La complexité algorithmique de l'algorithme d'induction est :

$$T(A,D) = O\left(|A| \times |D| \times \log_2\left(|D|\right)\right)$$

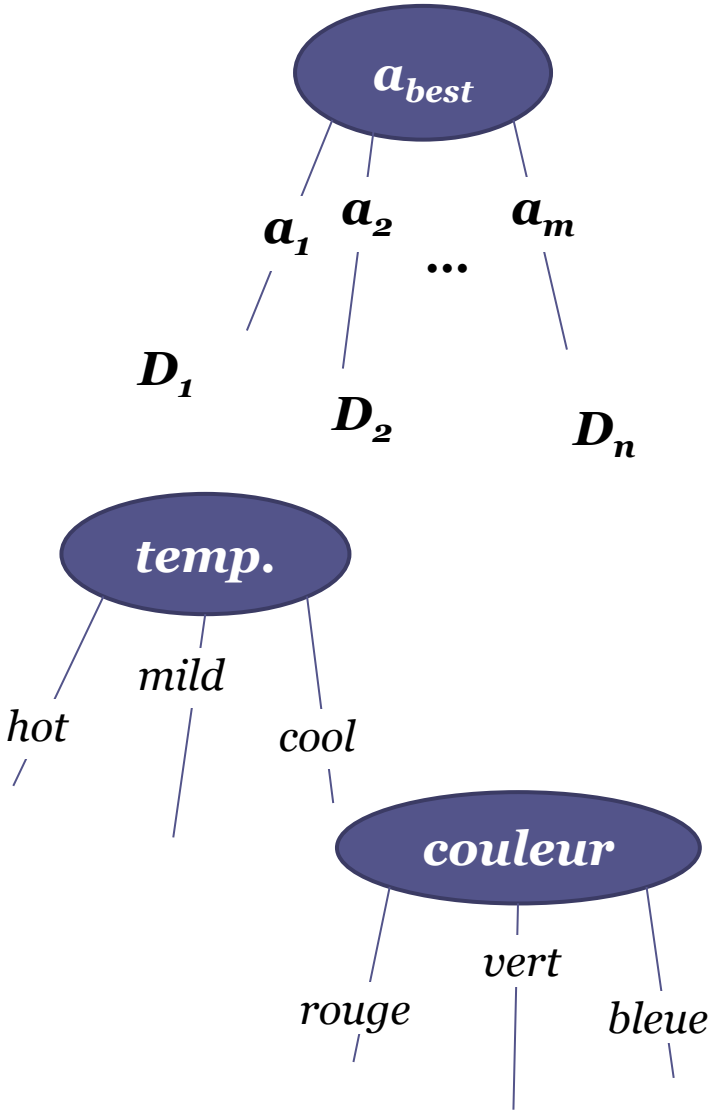
  - au pire des cas, on va développer un arbre à deux branches (deux tests), donc de hauteur $log_2(D)$
  - a chaque niveau, on a un calcul de O(|A|*|D|): on teste tous les attributs en utilisant tous les tuples de D

Formes de tests (1)

- Il y a trois formes de tests possibles dépendamment de la nature de l'attribut a_{best}
- **1er Cas : a_{best} est un attribut discret**
  - un attribut est dit discret s'il prend ses valeurs sur un ensemble fini
  - les tests sont des tests d'égalité par rapport aux différentes discrètes
  - on *développe une branche pour chaque valeur discrète*

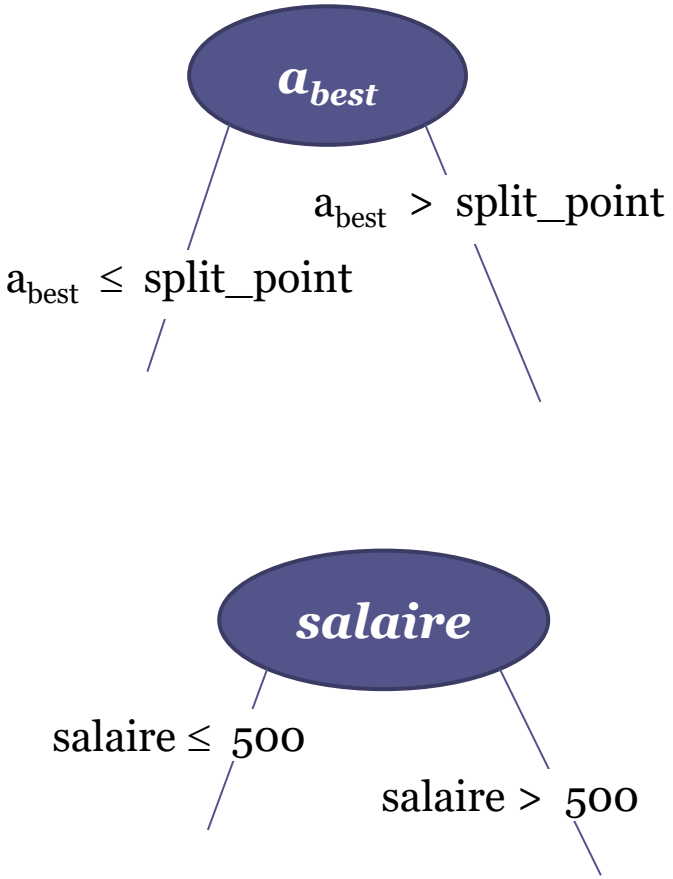
Formes de tests (2)

- Di est l'ensemble des tuples de D pour lesquels $a_{best} = a_i$
- Ainsi, dans chaque ensemble D_i, tous les tuples admettent exactement la même valeur pour a_{best}
- Ainsi, il n'a aucune utilité dans l'ensemble A
  - il est supprimé à l'étape 5 de l'algorithme



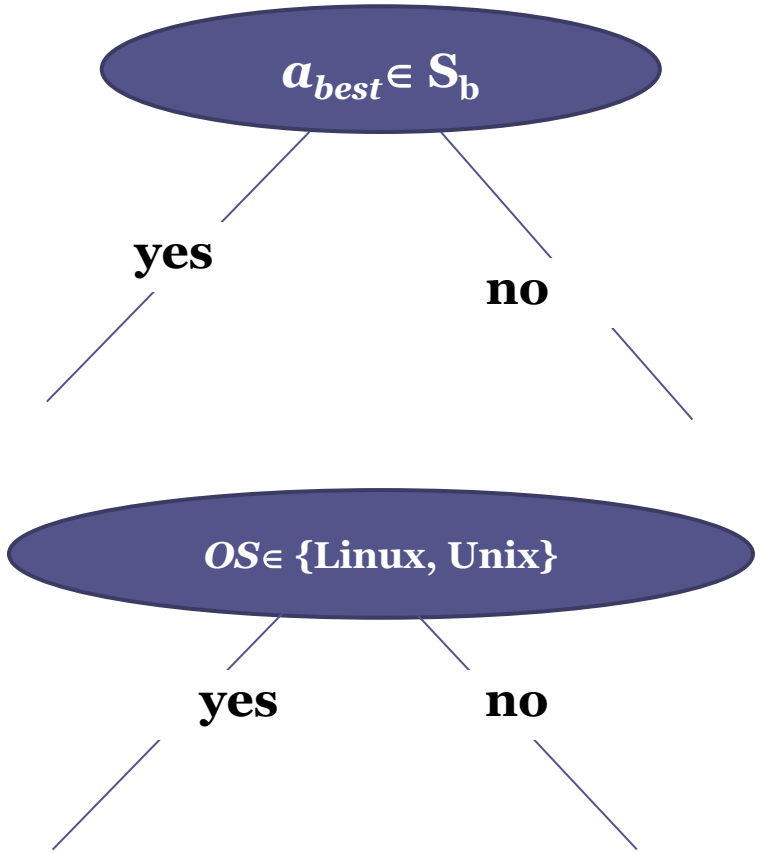
Formes de tests (3)

- **2ème cas : a_{best} est un attribut continu**
- Dans ce cas, on va calculer *un « point de division »* (split point)
- on développe deux branches chacune correspondant à un test possible : ≤ et >
  - l'égalité peut être placée à gauche ou à droite
- il y a plusieurs méthodes pour choisir le point de « point de division »



Formes de tests (3)

- **3ème cas: a_{best} est un attribut discret et on désire un arbre binaire**
  - un arbre est binaire si chaque nœud interne admet deux fils au max
- Dans ce cas, on doit déterminer un « ensemble de division » (*split subset*)
- les tests sont des tests d'appartenance ensemblistes
- Il y a plusieurs méthodes pour déterminer le meilleur « ensemble de division »



Méthodes de sélection

- Il y a plusieurs méthodes de sélection du meilleur attribut
- Ces méthodes sont des heuristiques modélisant différemment la même idée
  - le meilleur attribut est celui qui donne les *partitions les plus pures*
- **Gain informationnel**
- *Ratio du gain informationnel*
- *Index de Gini*
- …

Gain Informationnel (1)

- C'est une méthode basée sur le travail de *Claude Shannon* sur la *théorie de l'information*
- D est un ensemble d'apprentissage possédant m classes distinctes C_i (i=1..m)
- $D_{Ci} \subseteq D$ est l'ensemble des tuples ayant pour classe C_i
- P_i est la probabilité qu'un tuple arbitrairement choisi dans D appartienne à la classe C_i

$$p_i = \left|D_{C_i}\right| / |D|$$

Gain Informationnel (2)

- On définit **l'entropie** de l'ensemble D par :

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

  - cette entropie mesure la quantité d'information nécessaire D pour classer un tuple dans D
  - $Info(D) = 0$: ensemble pur
- Maintenant, on a besoin de calculer la quantité d'information nécessaire pour classer un tuple après avoir choisi un attribut de division
  - dépend de la nature de l'attribut

Gain Informationnel (3)

1er cas : attribut discret

- $V_a = \{a_1, a_2, .., a_v\}$ est l'ensemble des valeurs distinctes prises par a dans D
- D est divisé en v *sous-ensembles* $(D_1, ..., D_v)$ tel que D_j est composé des tuples ayant $(a = a_j)$
- On définit l'entropie de D sachant qu'on a choisi a comme attribut de division par :

$$Info_a(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Gain Informationnel (4)

- Remarquons que si on a produit uniquement des partitions pures alors $Info_a(D) = 0$
- On définit le ***gain informationnel*** par :

$$Gain(a) = Info(D) - Info_a(D)$$

  - le gain informationnel est la différence entre l'entropie de D avant le partitionnement et après le partitionnement selon un attribut (a)
- Le meilleur attribut est celui qui donne *un gain informationnel maximal*

$$a_{best} = \arg\max_{a \in A} Gain(a)$$

Gain Informationnel (5)

2ème cas : attribut continu

- pour un attribut continu, on détermine le meilleur «point de division » (*split point*)
  - trier les valeurs de a dans l'ordre croissant
    - $a_1 <= a_2 <= \ldots <= a_v$
  - on définit un « point de division » comme la moyenne de deux valeurs consécutives :

$$sp_i = (a_i + a_{i+1}) / 2$$

- ainsi, on obtient $(v-1)$ possibilités

Gain Informationnel (6)

- Pour chaque valeur possible sp_i (i=1..v-1)
  - diviser D en deux sous-ensembles : D_1 (≤ split point) et D_2 (> split point)
  - calculer l'entropie de D divisé selon sp_i

$$Info_a^{sp_i}(D) = \frac{|D_1|}{|D|} \times Info(D_1) + \frac{|D_2|}{|D|} \times Info(D_2)$$

- le meilleur « point de division » est celui qui donne *l'entropie minimale*

$$sp_{best} = \arg \min_{i=1..v-1} Info_a^{sp_i}(D)$$

Gain Informationnel (7)

3ème cas : attribut discret et on veut générer un arbre binaire

- dans ce cas, on détermine le meilleur « ensemble de division » (*split subset*)
- $V_a = \{a_1, a_2, .., a_v\}$ est l'ensemble des valeurs discrètes
- On considère $\boldsymbol{S}$ l'ensemble de tous les sous-ensembles possible de V_a et $S = \{S_1, S_2, ..., S_x\}$
  - remarquons que le nombre de sous-ensembles possibles est assez élevé : $\mathbf{2^v-1}$

Gain Informationnel (8)

- les tests sont sous la forme "**a ∈ S_i**?" avec $S_i \in S$
- chaque **S_i** divise D en deux sous-ensembles :
  - D_1 ($\in S_i$) et D_2 ($\notin S_i$)
- on calcul l'entropie de D divisé selon Si par :

$$Info_a^{S_i}(D) = \frac{|D_1|}{|D|} \times Info(D_1) + \frac{|D_2|}{|D|} \times Info(D_2)$$

- Le meilleur ensemble de division est celui qui donne l'entropie minimale

$$sp_{best} = \arg \min_{i=1..v-1} Info_a^{S_i}(D)$$

Exemple (1)

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

- D = ensemble d'apprentissage
- Deux classes: C_1 = "No", C_2 = "Yes"
- A = {Outlook, Temp., Humidity, Windy}: tous les attributs sont discrets
  - $V_{Outlook}$ = {sunny, overcast, rain}
  - $V_{Temp.}$ = {hot, cool, mild}
  - $V_{Humidity}$ = {high, normal}
  - V_{windy} = {false, true}
- p_1=P (Play=No)= 5 /14 = 0.36
- p_2=P (Play=Yes)=9 /14 = 0.64
- **Info(D)**= - 0.36 x $\log_2(0.36)$ - 0.64 x $\log_2(0.64)$ = **0.940**

Exemple (2)

- on va construire le premier nœud de l'arbre : nœud racine
  - on calcul le gain informationnel de chaque attribut
- **attribut « outlook »**
  - $V_{Outlook}$ = {sunny, overcast, rain}

$$Info_{Outlook}(D) = \frac{5}{14}\left(-\frac{3}{5}\log_2\left(\frac{3}{5}\right)-\frac{2}{5}\log_2\left(\frac{2}{5}\right)\right)+$$

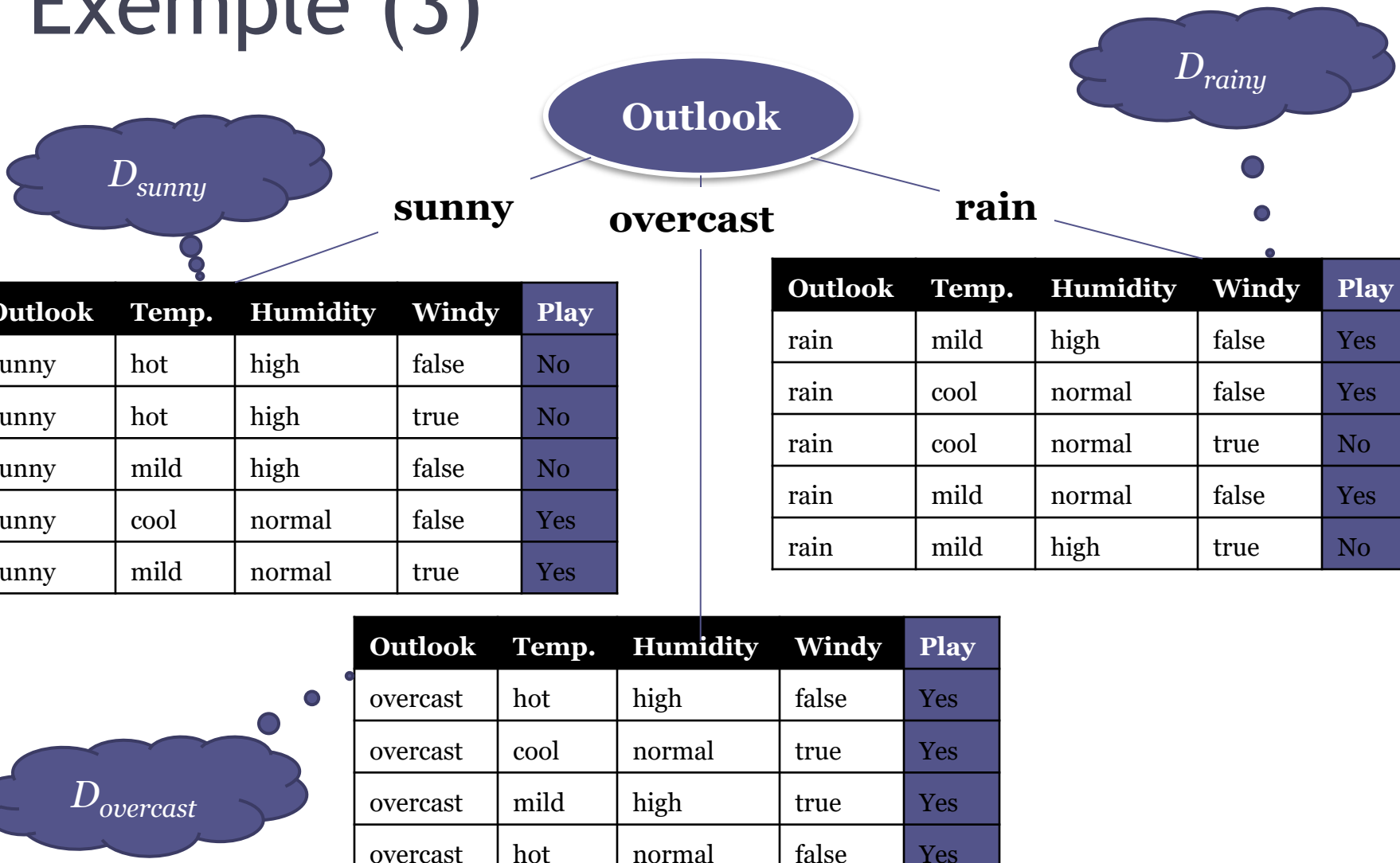
$$\frac{4}{14}\left(-0\log_2(0)-\frac{4}{4}\log_2\left(\frac{4}{4}\right)\right)+$$

$$\frac{5}{14}\left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right)-\frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)=0.694$$

$$Gain(outlook)=0.940-0.694=0.246$$

Outlook	C_1	C_2
D_{sunny}	3	2
$D_{overcast}$	0	4
D_{rainy}	2	3

Exemple (3)



Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

Outlook	Temp.	Humidity	Windy	Play
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
rain	mild	normal	false	Yes
rain	mild	high	true	No

Outlook	Temp.	Humidity	Windy	Play
overcast	hot	high	false	Yes
overcast	cool	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes

Exemple (4)

- Nous répétons la même démarche pour le reste des attributs
  - *Gain(Temp.) = 0.029*
  - *Gain(Humidity) = 0.151*
  - *Gain(Windy)=0.048*
- L'attribut qui possède le gain maximal est « outlook » qui a un gain de 0.246
- L'attribut « *outlook* » va constituer la racine de l'arbre

Temp.	C_1	C_2
D_{hot}	2	2
D_{mild}	2	4
D_{cool}	1	3

Humidity	C_1	C_2
D_{high}	4	3
D_{normal}	1	6

Windy	C_1	C_2
D_{false}	2	6
D_{true}	3	3

Exemple (5)

sunny

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

overcast

Outlook	Temp.	Humidity	Windy	Play
overcast	hot	high	false	Yes
overcast	cool	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes

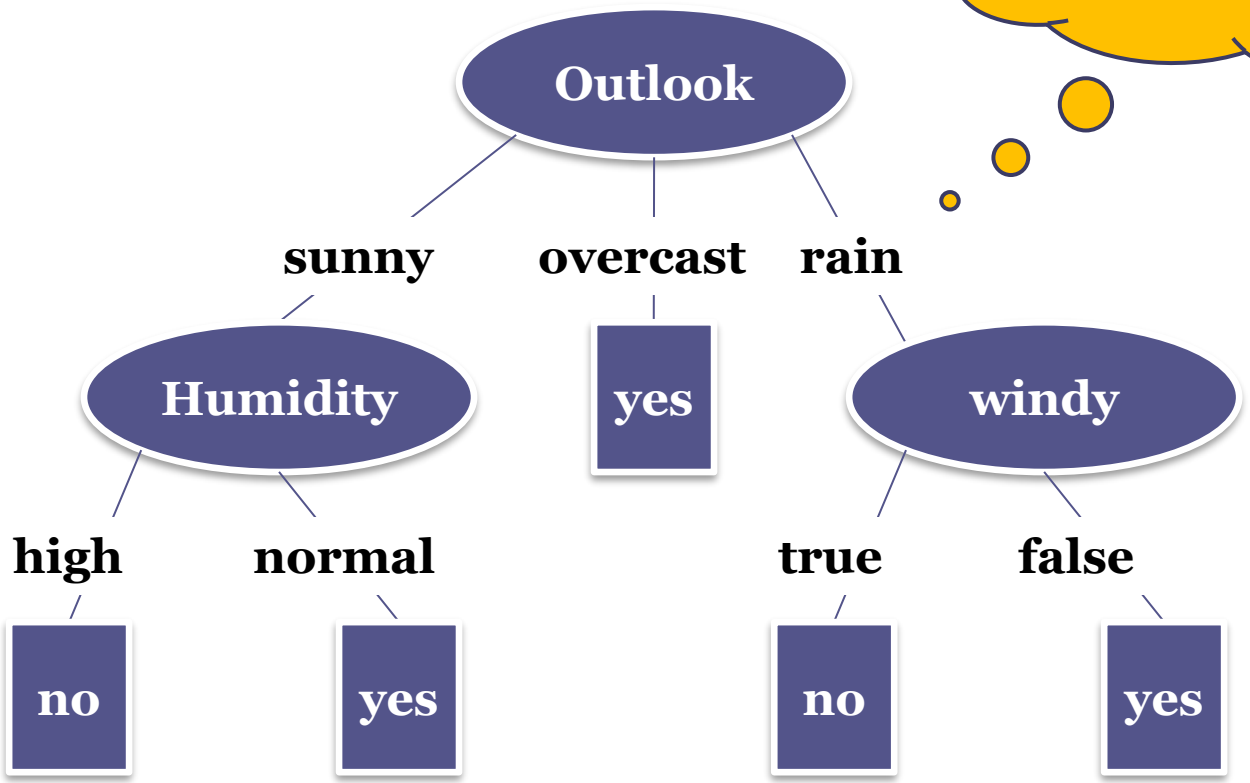
un ensemble pur: on a un noeud feuille avec la classe “YES”

rain

Outlook	Temp.	Humidity	Windy	Play
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
rain	mild	normal	false	Yes
rain	mild	high	true	No

Exemple (6)

L'attribut « temp. » ne joue aucun rôle dans la décision



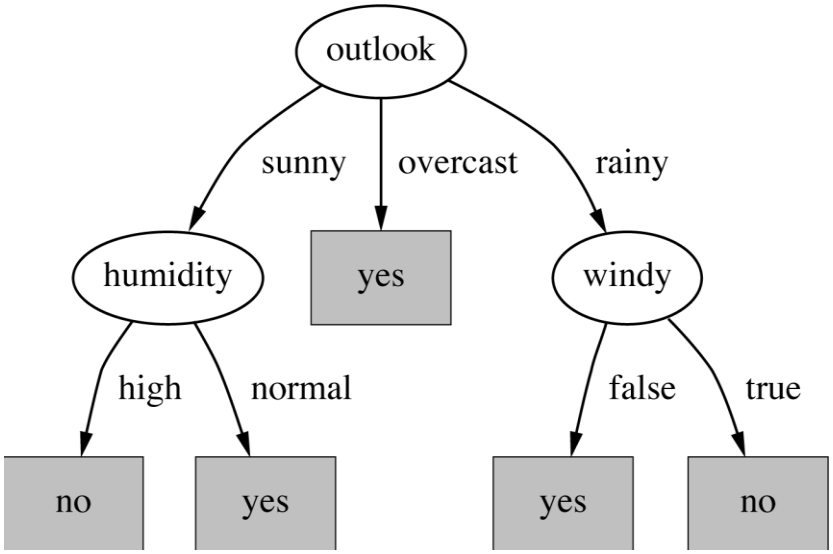
Règles de décision (1)

- Une règle de décision, dite de *classification*, est une proposition logique sous la forme :

  SI $P_1 \wedge P_2 \wedge \ldots \wedge P_n$ **ALORS** P_c

  - où chaque P_i est une proposition logique s'évaluant à VRAI/FAUX
- Un arbre de décision peut-être transformée en un ensemble de règles de décision
  - plus simples à interpréter par l'être humain
- **Stratégie**
  - chaque chemin de l'arbre de la racine à une feuille est représentée à l'aide d'une règle
  - Chaque arc de l'arbre (test) est modélisée à l'aide d'une proposition logique

Règles de décision (2)

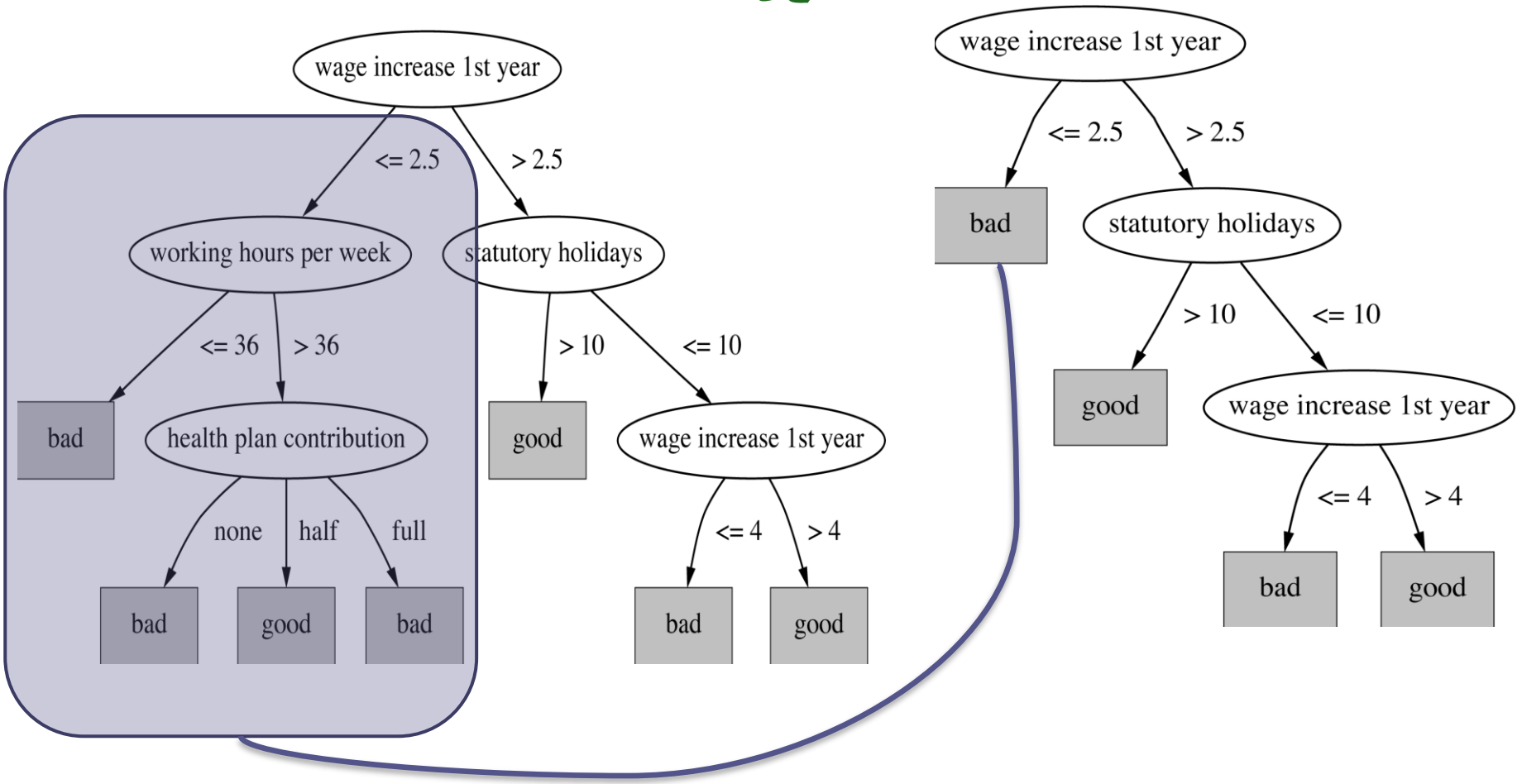


- ***R1:*** SI (Outlook=sunny) ∧ (Humidity=High) ALORS (Play=No)
- ***R2:*** SI (Outlook=sunny) ∧ (Humidity=Normal) ALORS (Play=Yes)
- ***R3:*** SI (Outlook=overcast) ALORS (Play=Yes)
- ***R4***: SI (Outlook= rain) ∧ (Windy=True) ALORS (Play=No)
- ***R5:*** SI (Outlook=rain) ∧ (Wind= false) ALORS (Play=Yes)

Elagage (1)

- L'arbre peut devenir assez complexe à interpréter
  - Beaucoup de branches vont interpréter les anomalies dans les données d'apprentissage
- L'élagage consiste à supprimer certaines parties de l'arbre
  - certaines branches (sous-arbres) sont « remplacées » par des nœuds feuilles (décisions)
    - **post-élagage:** construire l'arbre en sa totalité et puis le simplifier
    - **pré-élagage:** simplifier l'arbre pendant sa construction: éviter de développer certaines branches

Elagage (2)


wage increase 1st year
<= 2.5
> 2.5
working hours per week
statutory holidays
<= 36
> 36
> 10
<= 10
bad
health plan contribution
good
wage increase 1st year
none
half
full
<= 4
> 4
bad
good
bad
bad
good
wage increase 1st year
<= 2.5
> 2.5
bad
statutory holidays
> 10
<= 10
good
wage increase 1st year
<= 4
> 4
bad
good

Post-élagage(1)

- Soit T_{max} l'arbre obtenu à partir de l'ensemble d'apprentissage
- Construire une suite d'arbres $\{T_{max}, T_1, T_2, \ldots, T_n\}$ en partant des feuilles et en remontant vers la racine *en transformant un nœud en feuille à chaque étape*
- Comparer le **coût** du nouvel arbre à celui du précédent
  - arrêter l'élagage si le coût est supérieur

Post-élagage(2)

- Minimisez le critère d'erreur suivant :

$$\bar{w}(T_k, v) = \frac{n(k) - MC_{ela}(v,k)}{nt(v,k)(MC(v,k) - 1)}$$

$MC_{ela}(v,k)$	Nombre d'exemples de l'ensemble d'apprentissage mal classés par le nœud v de T_k dans l'arbre non élagué
$n(k)$	Nombre d'exemples de l'ensemble d'apprentissage mal classés par le nœud v de T_k dans l'arbre élagué à v
$MC(v,k)$	Nombre de feuilles du sous-arbre de T_k situé sous le nœud v
$nt(v,k)$	Nombre de feuilles de T_k

Post-élagage (3)

Algorithme post-élagage

- entrée : T_{max} (arbre construit)
- sortie : T_{opt} (arbre élagué)

DEBUT

k ← 0

$T_k \leftarrow T_{max}$

Tant Que (T_k a plus d 'un nœud interne) **faire**

pour chaque nœud v de T_k **faire**

calculer le critère $w(T_k,v)$ sur l 'ensemble d 'apprentissage

finPour

choisir le nœud v_m pour lequel ***le critère est minimal***

T_{k+1} se déduit de T_k en y remplaçant v_m par une feuille

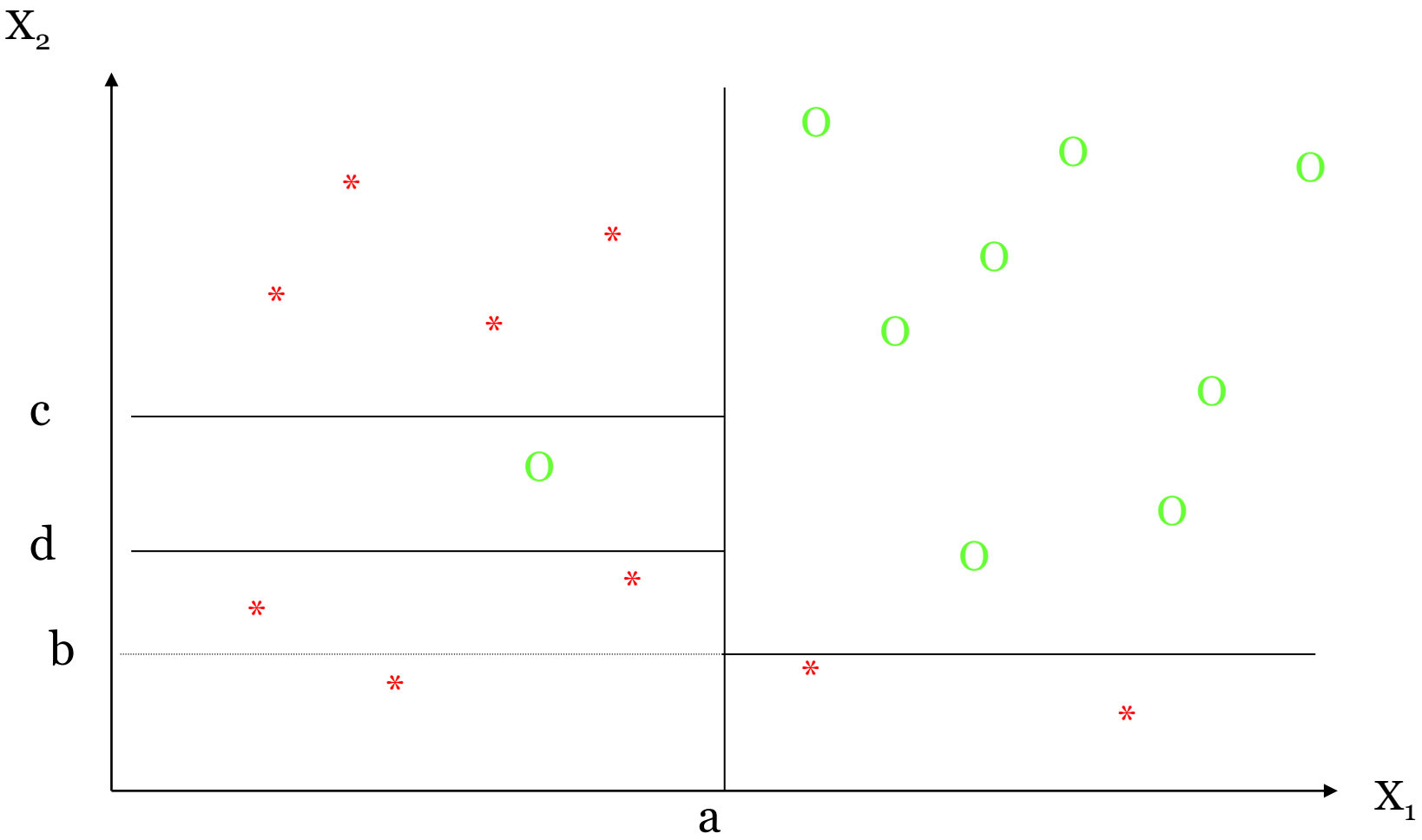
k← k+1

Fin TantQue

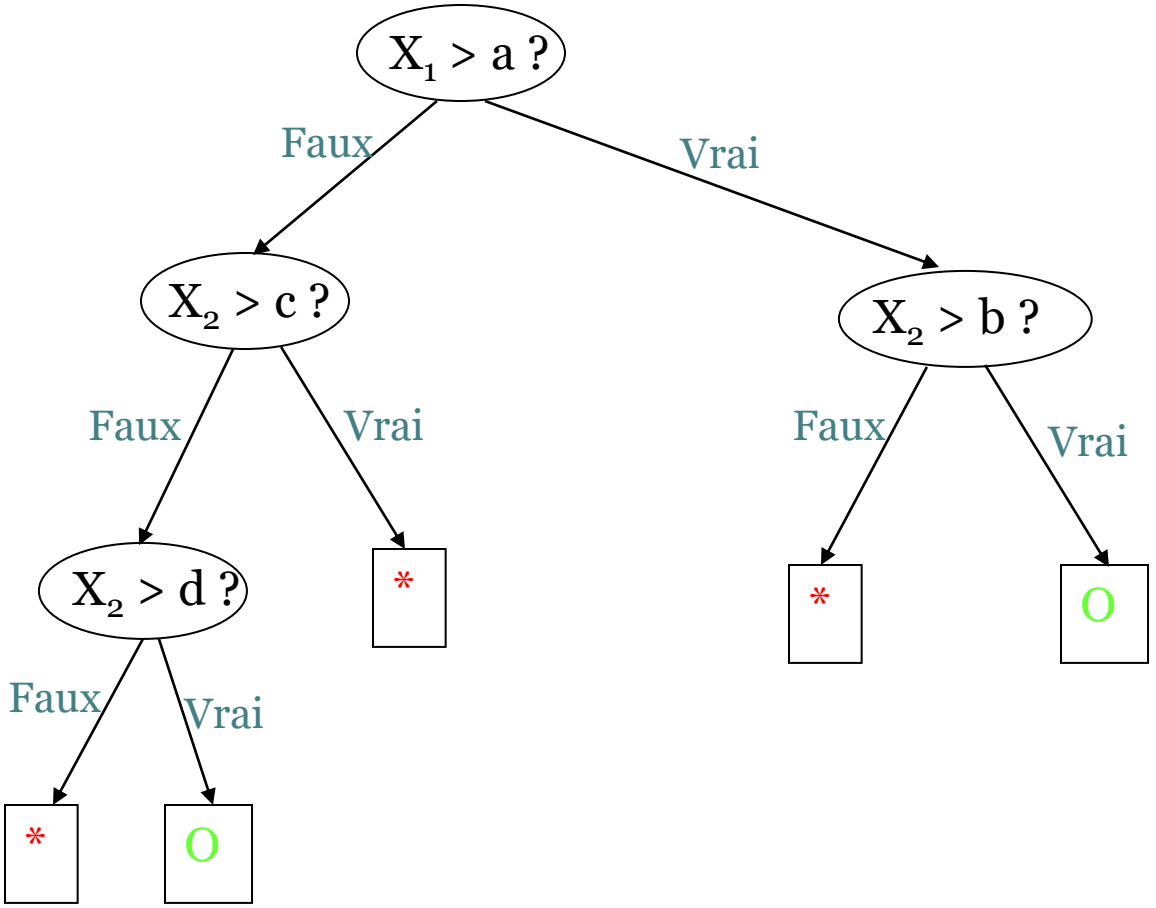
T_{opt} ← l'arbre $\in \{T_{max}, T_1, ..., Tn\}$ qui a la plus petite erreur sur l'ensemble de validation

FIN.

Post-élagage (4)

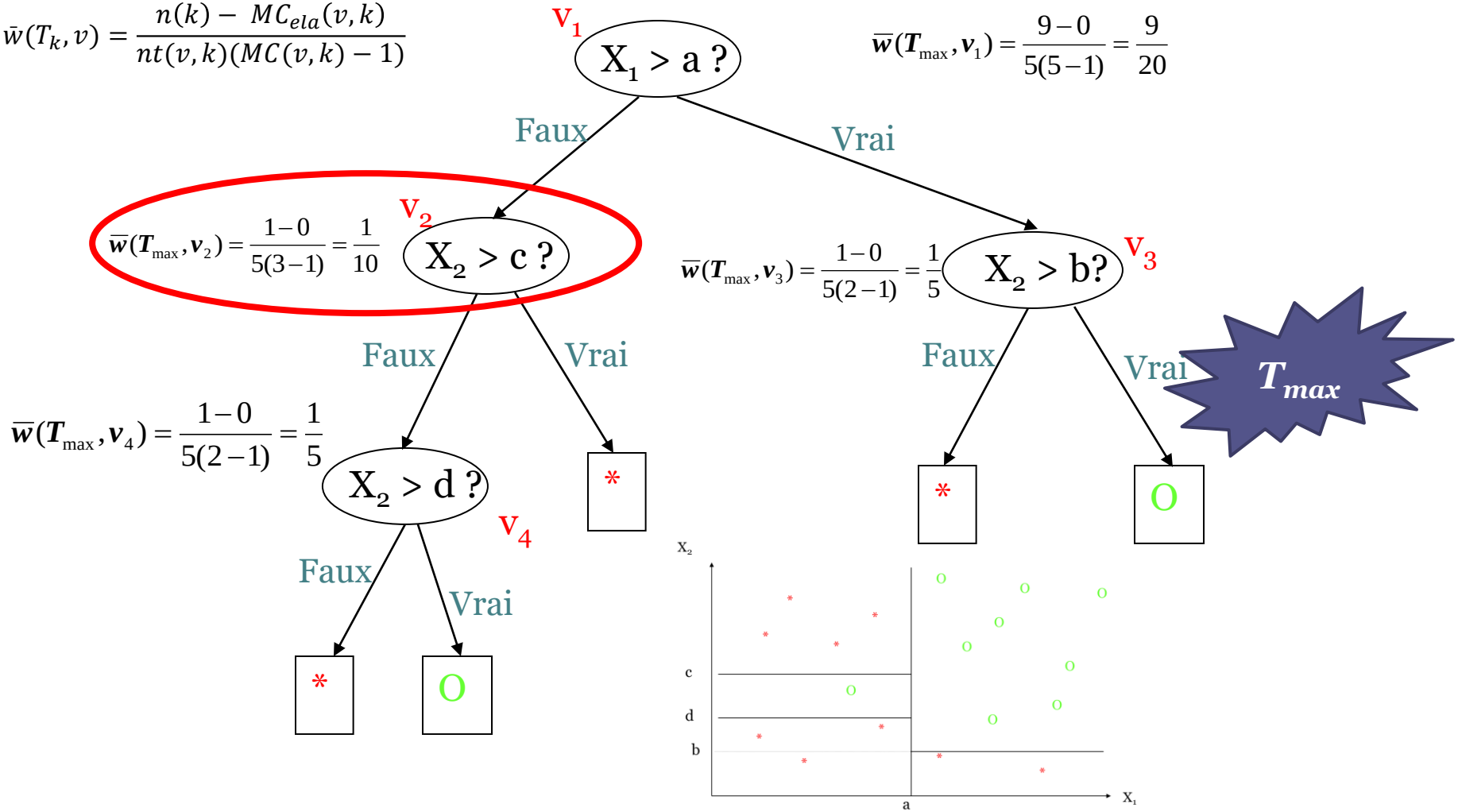


Post-élagage (5)



Post-élagage (6)

$$\bar{w}(T_k, v) = \frac{n(k) - MC_{ela}(v,k)}{nt(v,k)(MC(v,k) - 1)}$$



$$\bar{w}(T_{max}, v_1) = \frac{9-0}{5(5-1)} = \frac{9}{20}$$

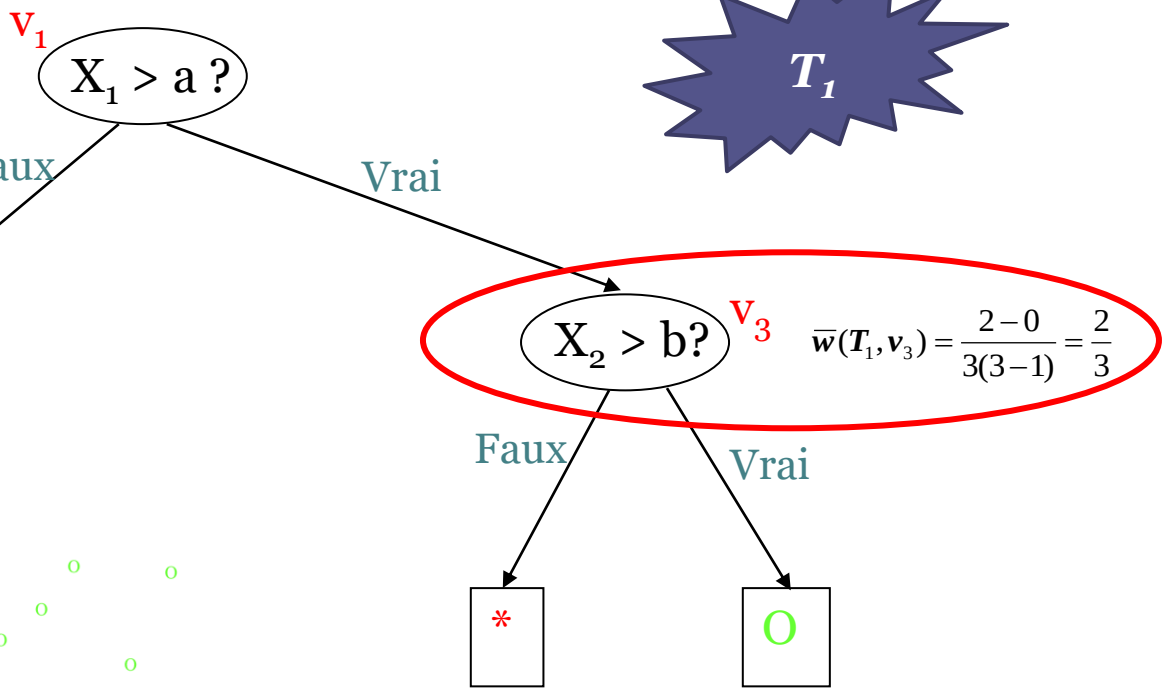
$$\bar{w}(T_{max}, v_2) = \frac{1-0}{5(3-1)} = \frac{1}{10}$$

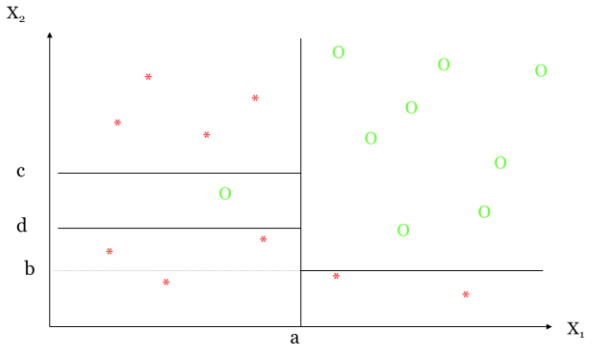
$$\bar{w}(T_{max}, v_3) = \frac{1-0}{5(2-1)} = \frac{1}{5}$$

$$\bar{w}(T_{max}, v_4) = \frac{1-0}{5(2-1)} = \frac{1}{5}$$

Post-élagage (7)

$$\bar{w}(T_1, v_1) = \frac{9-1}{3(3-1)} = \frac{4}{3}$$

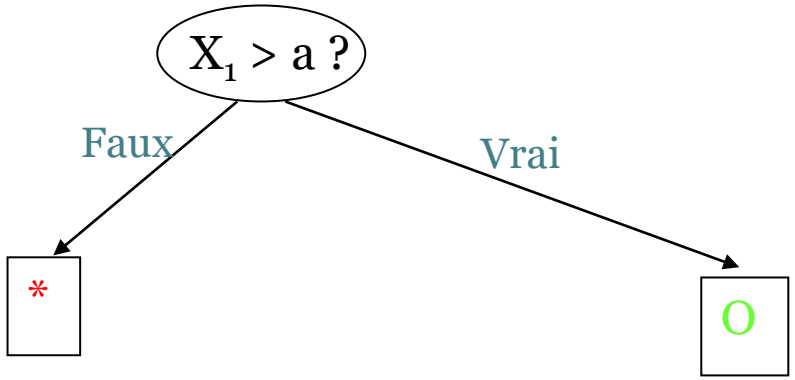




$$\bar{w}(T_k, v) = \frac{n(k) - MC_{ela}(v,k)}{nt(v,k)(MC(v,k) - 1)}$$

Post-élagage (8)

T_2



Choisir, à partir d'un ensemble de validation, le meilleur arbre parmi T_{max}, T_1 *et* T_2. C'est à dire celui minimisant l'erreur de classification

pré-élagage

- consiste à proposer des **critères d'arrêt** lors de la phase d'induction de l'arbre
  - le nombre de tuples est faible ($|D| < seuil$)
  - la valeur du gain informationnel est négligeable
  - utiliser un test statistique : test du χ^2 permettant de mesurer l'indépendance statistique de la population d'un nœud par rapport à une classe
- On risque une perte « importante » des connaissances (certaines branches de l'arbre)