IFT 615 – Intelligence Artificielle

Été 2022

Apprendre des arbres de décision

Professeur: Froduald Kabanza

Assistants: D'Jeff Nkashama & Jean-Charles Verdier

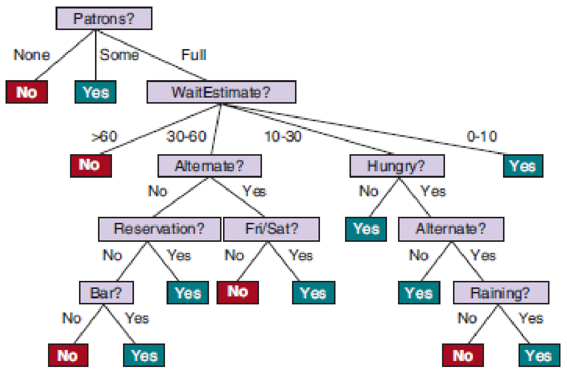
Université de Sherbrooke

Arbre de décision *vs* réseau de neurones

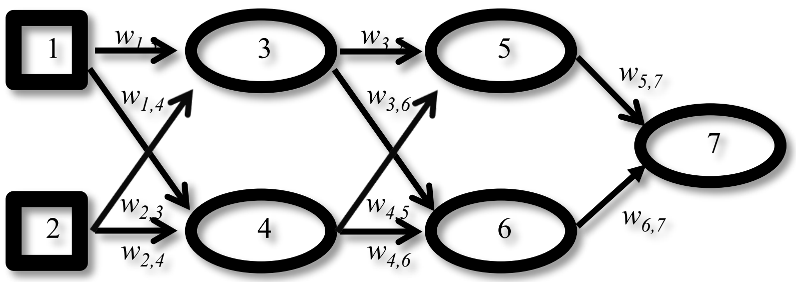
- Avec les réseaux de neurones, les arbres de décision sont actuellement les deux types de représentations les plus utilisées pour l'apprentissage supervisé dans l'industrie.

- Tout comme un réseau de neurones, un arbre de décision est un modèle paramétrique.

- Un *arbre de décisions* est un modèle symbolique, plus facile à interpréter.

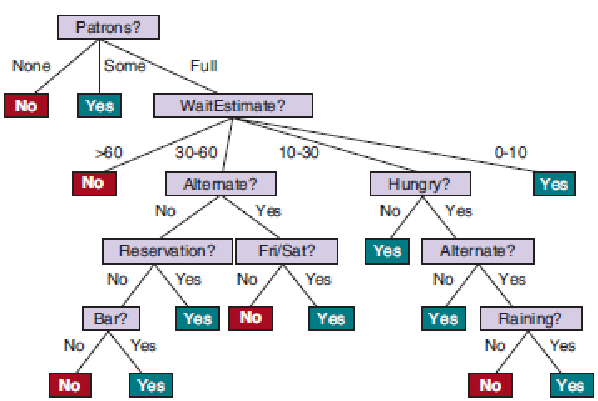


VS

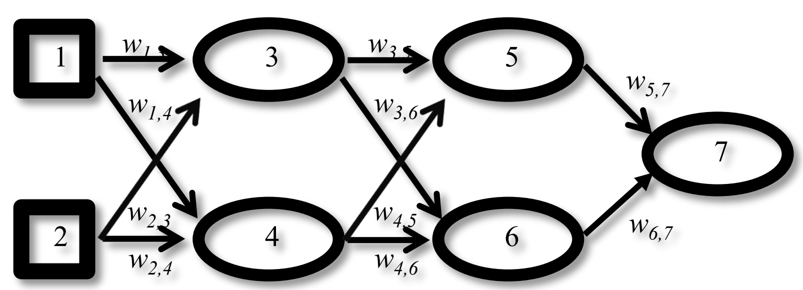


Arbre de décision *vs* réseau de neurones

- De façon général, un modèle (appris) est une fonction $y = f(x)=f(x_1, .., x_n)$:
  - $x=[x_1, ...,x_n]$: entrée (c.-a.d-. x_i sont les variables d'entrée)
  - y : cible d'apprentissage.
- Pour un réseau neurones, la fonction $f \equiv f_w$ est représentée par le vecteur de poids w : $y = f_w(x)=f_w(x_1, .., x_n)$.
- Dans le cas d'un arbre de décision, la fonction $f \equiv f_T$ est représentée par un arbre de décision T. Chaque nœud intérieur est un test sur une variable d'entrée. Chaque feuille est une valeur pour la variable cible.



VS



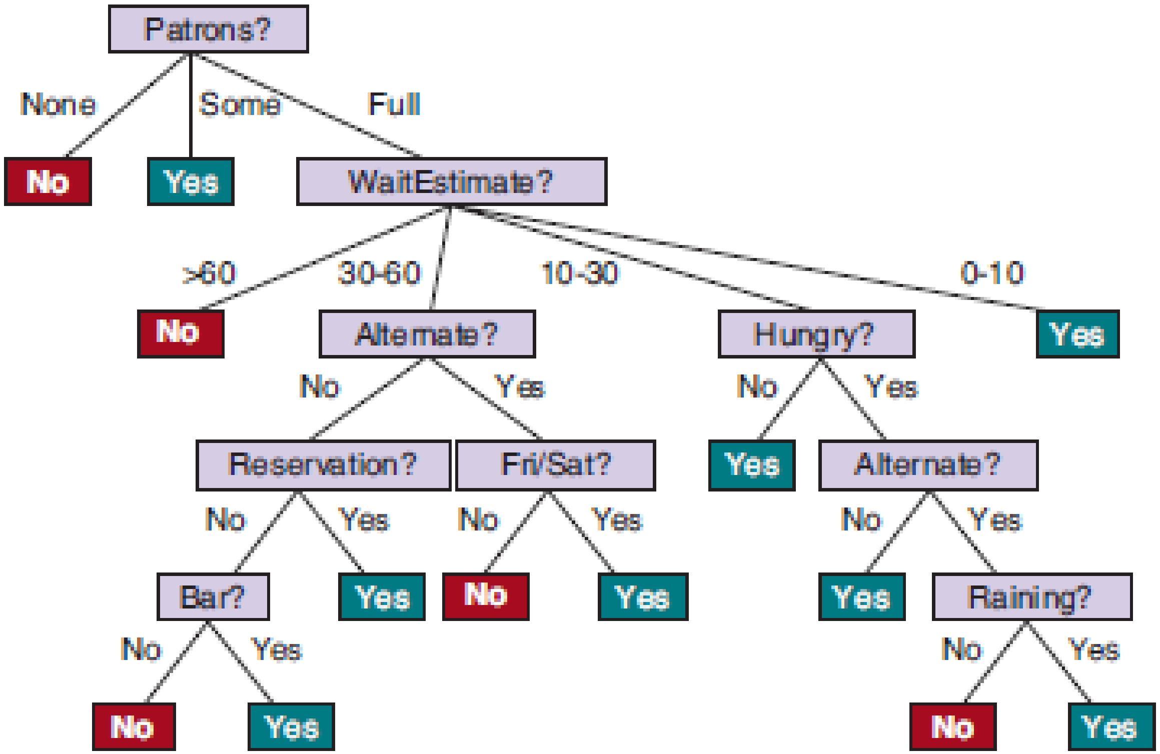
Sujets couverts par cette leçon

- Définition d’un arbre de décision

- Algorithme d’apprentissage d’un arbre de décision

Arbre de décision

- Un arbre de décision est une représentation d'un flux d'exécution tel chaque nœud intérieur représente un test sur une variable x_i du vecteur d'entrées x et un nœud feuille représente la valeur de la variable cible y.



Exemple 1 – Attente dans un resto

$x_i \equiv [x_{i1}, \; x_{i2}, \; x_{i3}, \; x_{i4}, \; x_{i5}, \; x_{i6}, \; x_{i7}, \; x_{i8}, \; x_{i9}, \; x_{i10}]$

$\equiv$ [Alt, Bar, Fri, Hun, Pat, Price, Rain, Res, Type, Est]

$y_i \equiv$ [WillWait]

- *Alternate*: Il y un resto alternatif tout proche ou non
- *Bar*: le resto a un bar confortable pour y attendre ou non
- *Fri / Sat*: On est vendredi ou samedi ou non
- *Hungry*: J'ai beaucoup faim ou non
- *Patrons*: Achalandage en ce moment (valeurs: *None*, *Some*, *Full*)
- *Price*: la gamme de prix du resto ($, $$, $$$)
- *Raining*: Il pleut ou non
- *Reservation*: J'ai réservé ou non
- *Type*: Type de resto (French, Italian, Thai ou Burger)
- *WaitEstimate*: Temps d'attente (valeurs: 0-10, 01-30, 30-60, > 60 min)

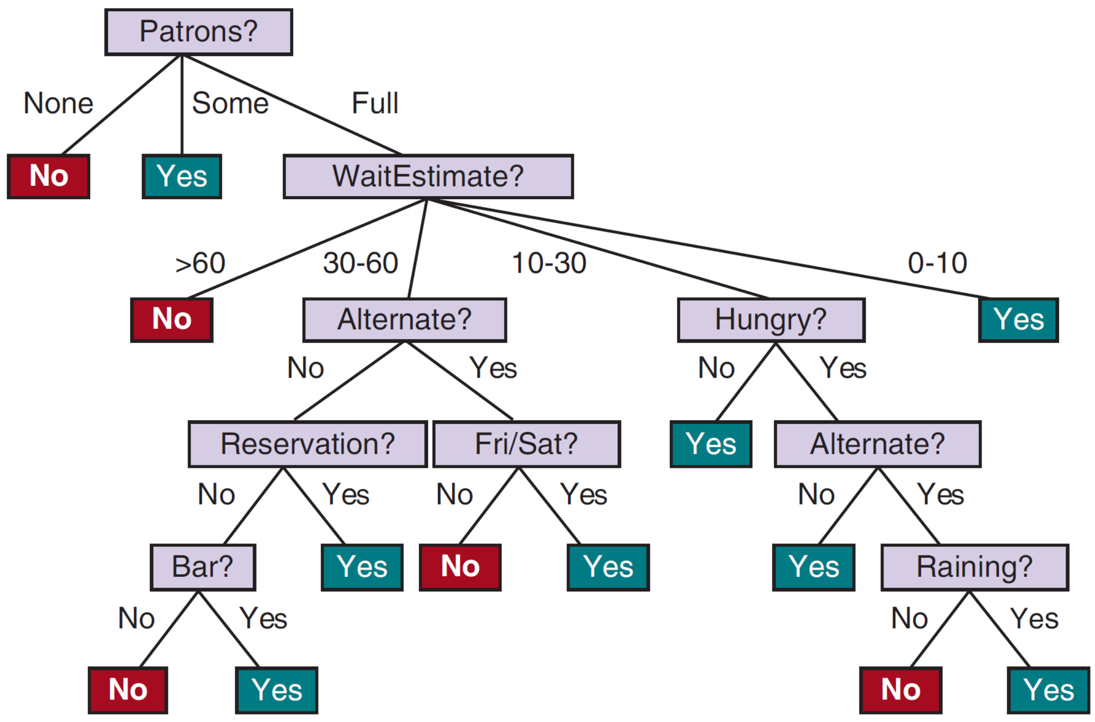
Exemple 1 – Attente dans un resto

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Jeu de données pour l'application du resto

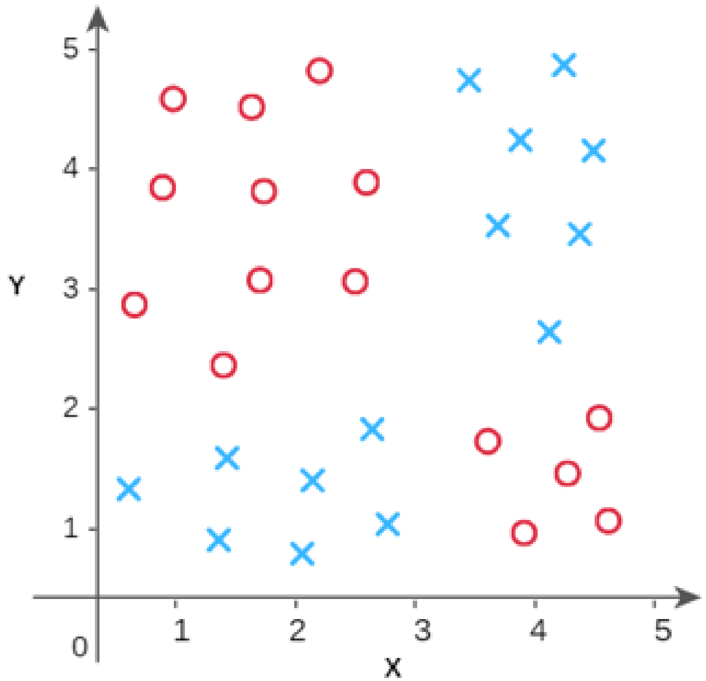
Exemple – Attente dans un resto

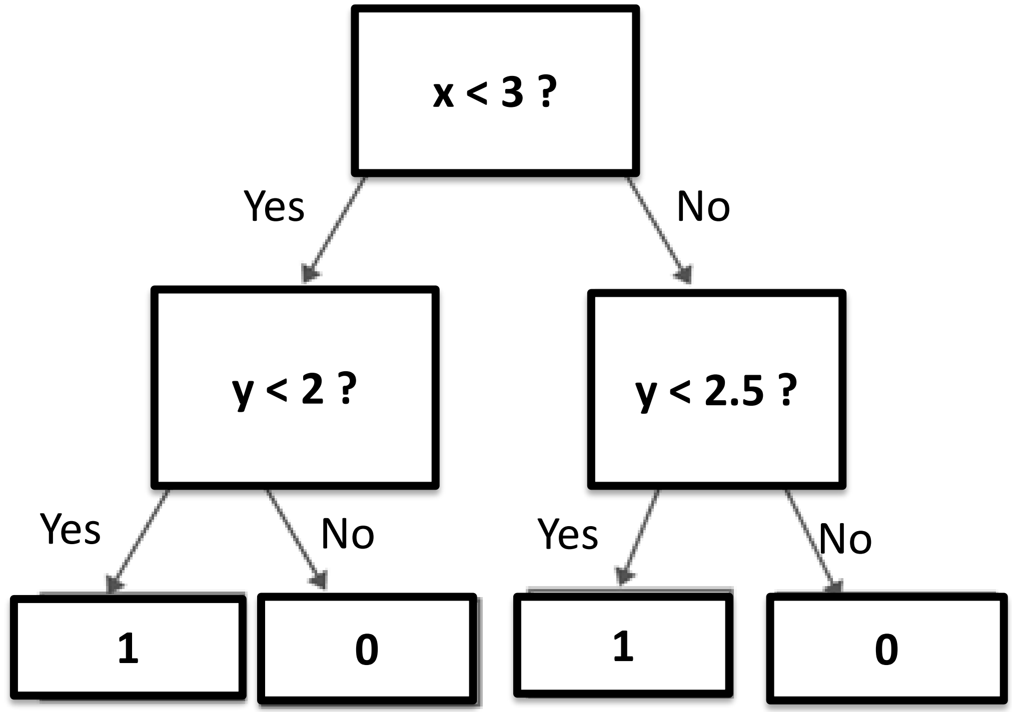
- Un arbre de décision est une représentation d'un flux d'exécution tel chaque nœud intérieur représente un test sur une variable x_i du vecteur d'entrées x et un nœud feuille représente la valeur de la variable cible y.



Arbre de decision pour l'application du resto

Exemple – Générique





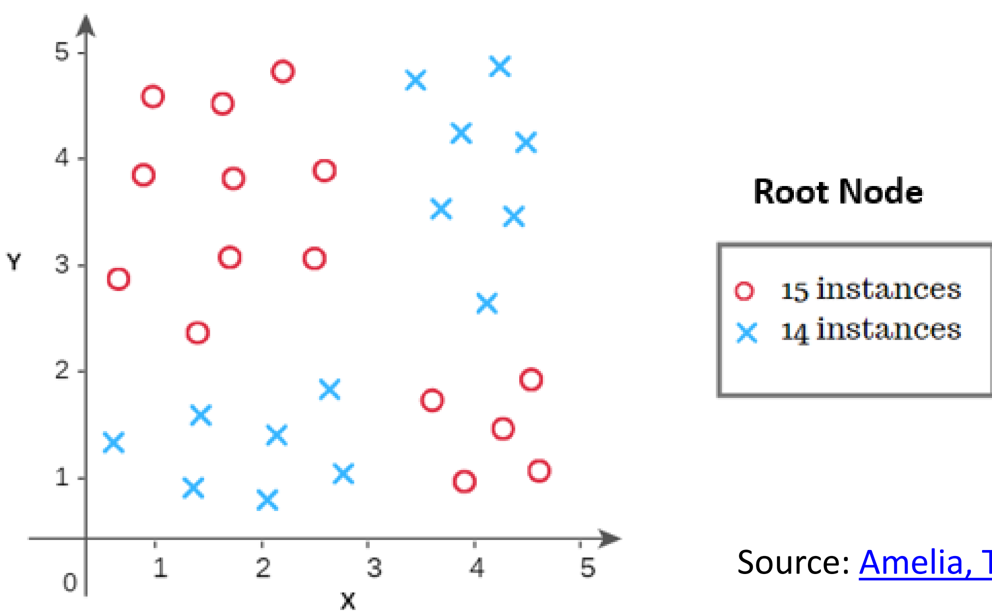
Source: [Amelia, Towards Data Science, 2019]

Algorithme d'apprentissage (définition préliminaire)

- Un nœud correspond à un test sur une variable et un ensemble d'exemples.
- Le nœud racine correspond à tous les exemples.
- Itérativement
  - Choisir une variable non encore choisie jusque là
  - Lui appliquer un test sur ses valeurs
  - Pour chaque valeur du test
    - » Générer un enfant contenant les exemples consistent avec le test

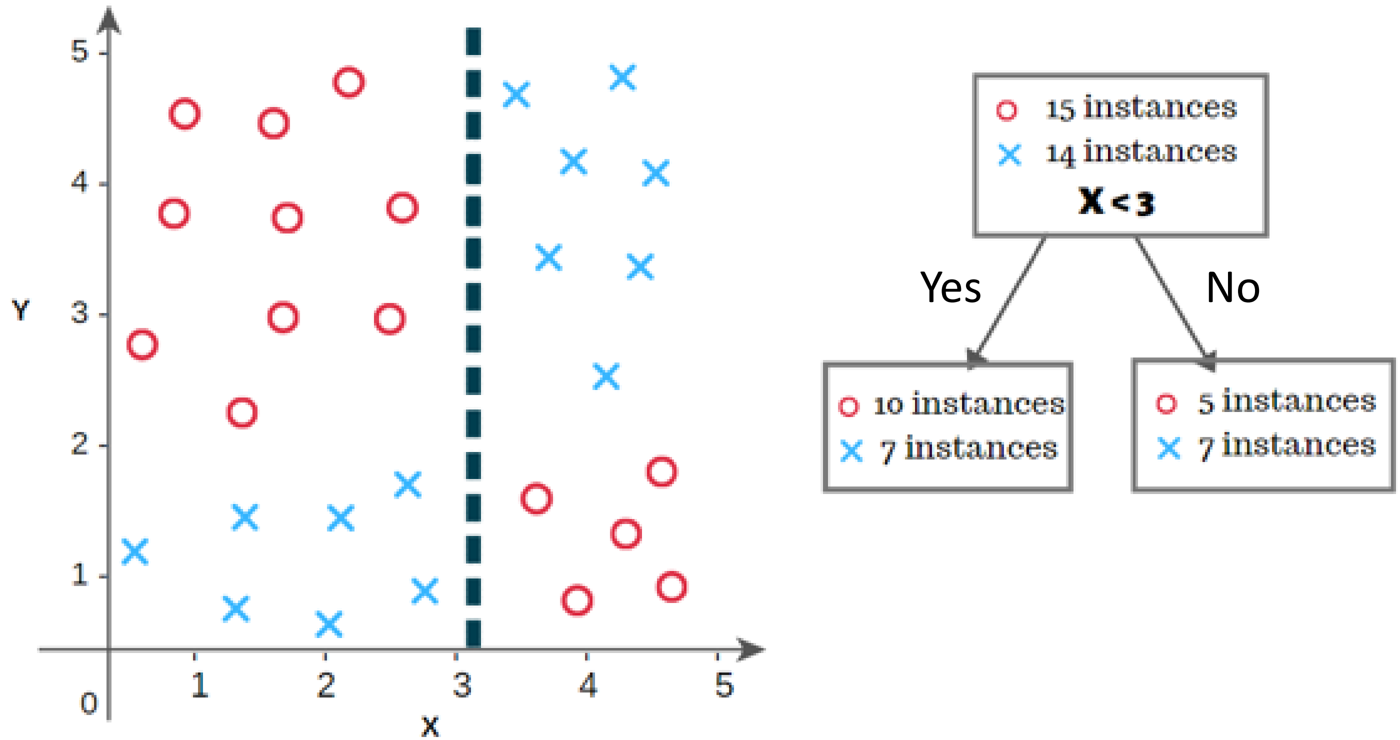
Exemple 1 - Générique

- Itérativement
  - Choisir un attribut non encore choisi jusque là
  - Lui appliquer un test sur ses valeurs
  - Pour chaque valeur du test
    - » Générer un enfant contenant les exemples consistent avec le test



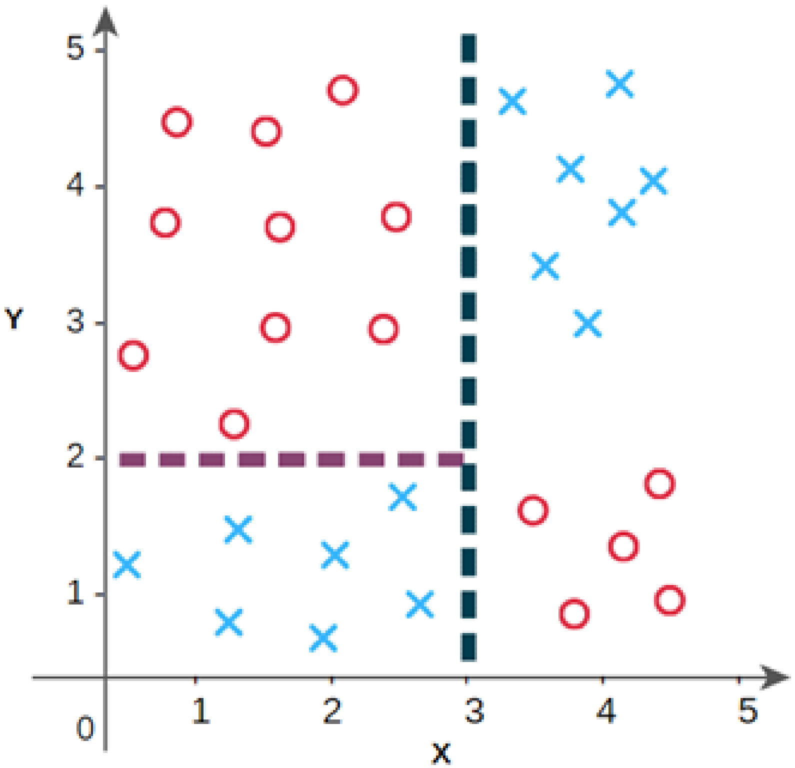
Source: [Amelia, Towards Data Science, 2019]

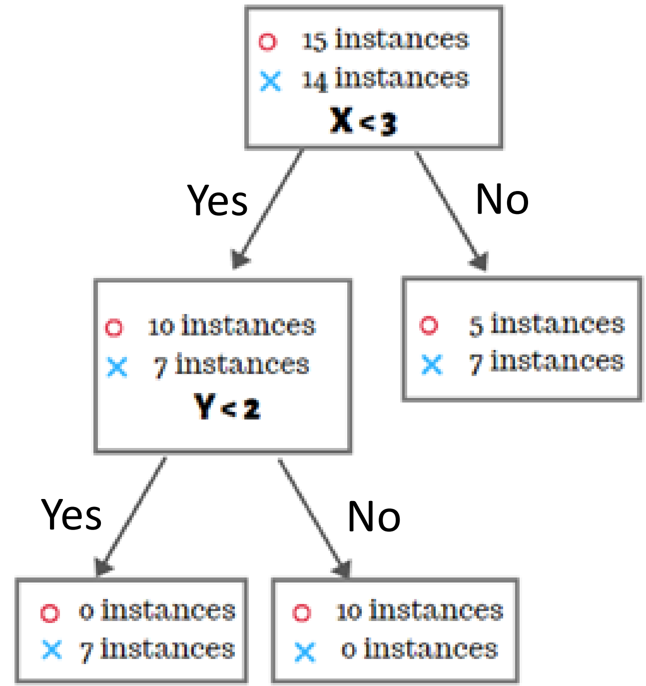
Exemple 1 – Générique



Source: [Amelia, Towards Data Science, 2019]

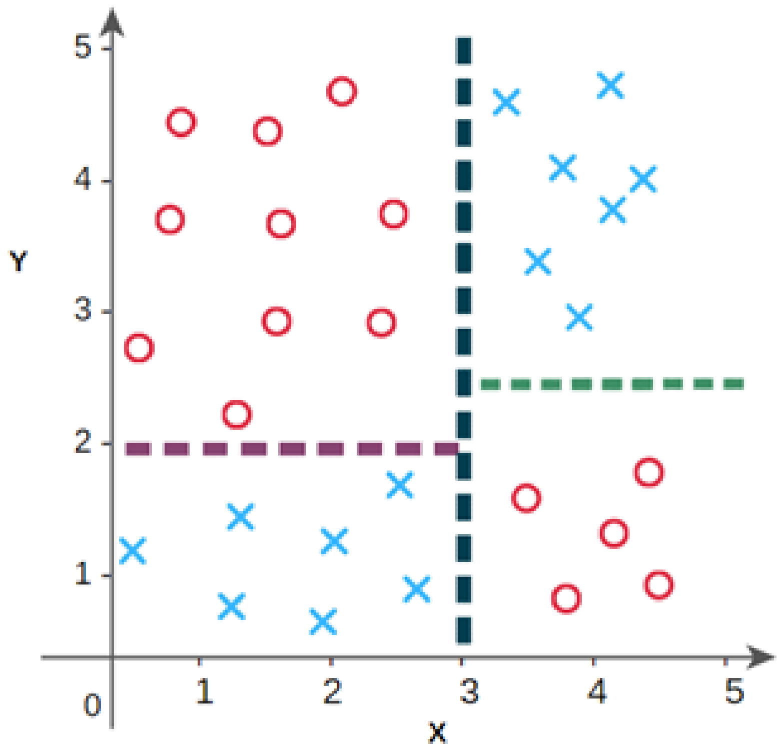
Exemple 1 – Générique

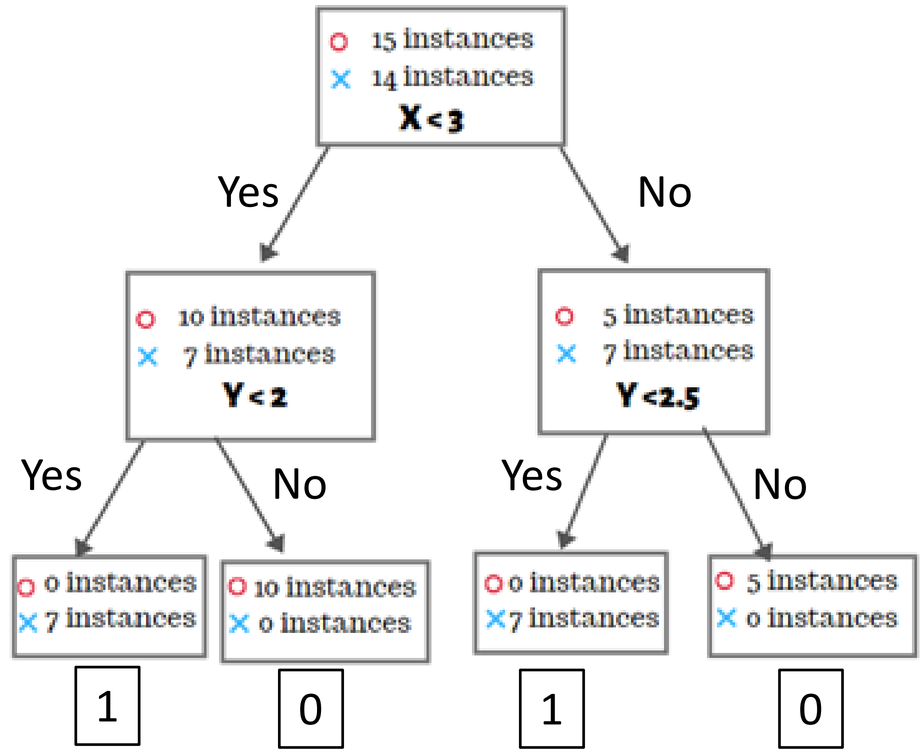




Source: [Amelia, Towards Data Science, 2019]

Exemple 1 – Générique

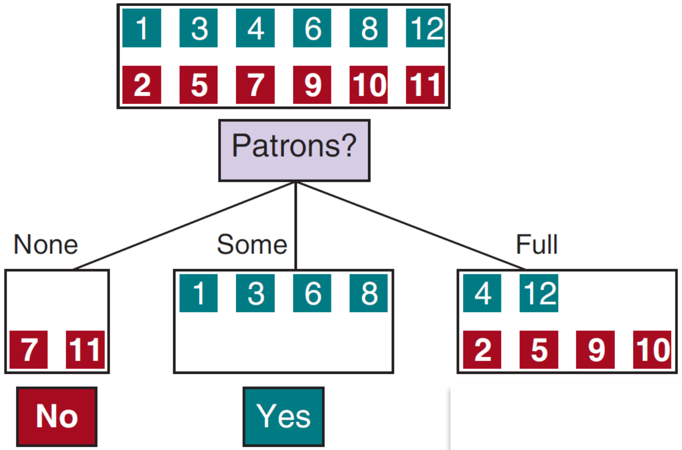




Source: [Amelia, Towards Data Science, 2019]

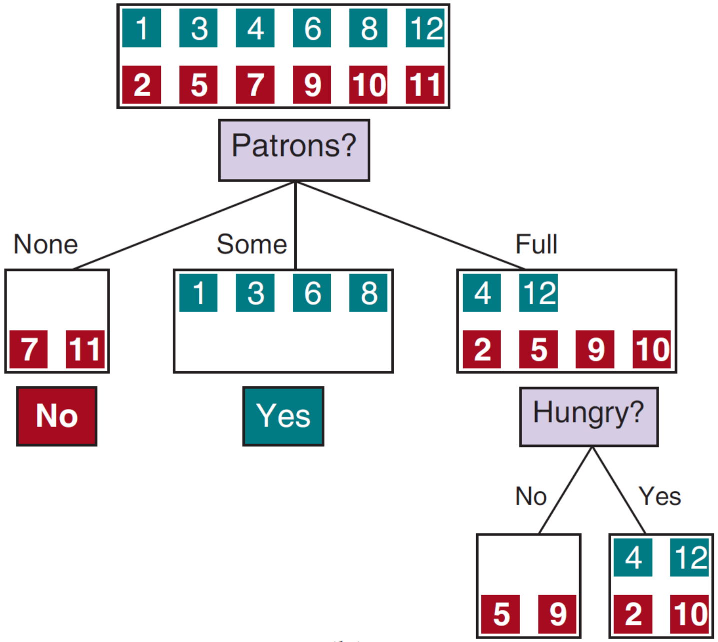
Exemple 2 – Attente dans un resto

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$



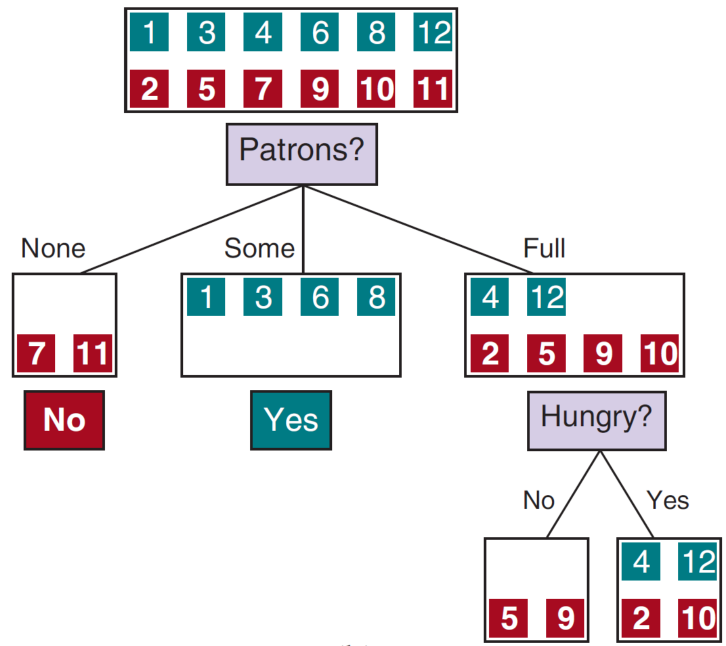
Exemple 2 – Attente dans un resto

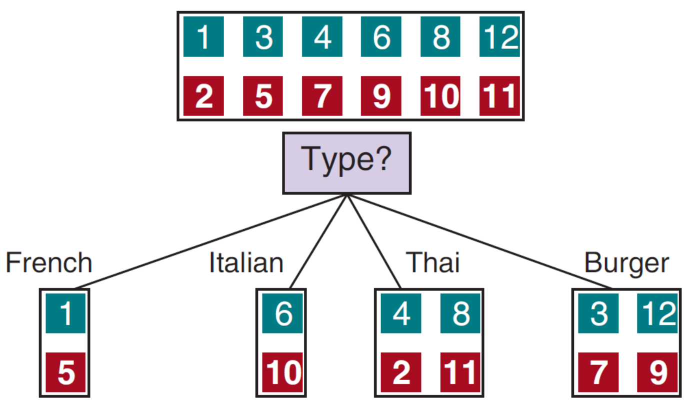
Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$



Exemple 2 – Attente dans un resto

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$





Algorithme d'apprentissage (définition formelle)

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
      A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
      tree ← a new decision tree with root test A
      for each value v of A do
          exs ← {e : e ∈ examples and e.A = v}
          subtree ← LEARN-DECISION-TREE(exs, attributes − A, examples)
          add a branch to tree with label (A = v) and subtree subtree
      return tree
```

Expressivité des arbres de décision

- Un arbre de décision binaire est équivalent à une formule propositionnelle de la forme

$$\textit{Sortie} \Leftrightarrow (\textit{Chemin}_1 \vee \textit{Chemin}_2 \vee \ldots),$$

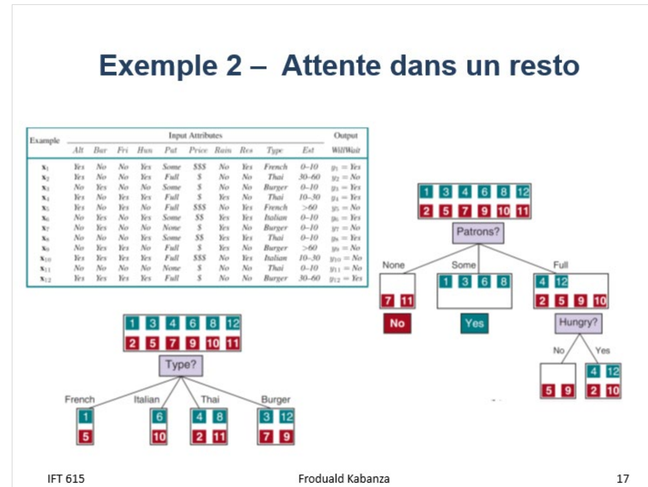
où $\textit{Chemin}_i$ est une conjonction de la forme $(A_m = v_x \wedge A_n = V_y \wedge \ldots)$ d'assignations *attribute-valeur* correspondants aux tests le long du chemin de la racine à la feuille.

- C'est une forme normale conjonctive.
- Cela veut dire que toute formule de logique propositionnelle peut être exprimée par un arbre de décision.

- La surface de décision d'un arbre de décision est un ensemble de rectangles.

Choix de l’attribut test

- On veut choisir l’attribut qui réduit le plus l’incertitude dans les exemples restants à classer



Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 = Yes$
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 = No$
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 = Yes$
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 = Yes$
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 = No$
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 = Yes$
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 = No$
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 = Yes$
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 = No$
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} = No$
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} = No$
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} = Yes$

- Cela nous amène à définir d’abord les concepts:
  - Entropie comme mesure d’incertitude
  - Gain d’information en terme de réduction de l’entropie

- On va alors choisir l’attribut qui apporte le plus de gain d’information

Entropie

- L'entropie d'une variable aléatoire V ayant les valeurs possibles v_k avec la distribution de probabilité $P(v_k)$ est définie comme étant

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = - \sum_k P(v_k) \log_2 P(vk)$$

Exemples

- L'entropie d'un choix pile ou face:

  $H(PileOuFace) = - (0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$ *bit*

- Si la pièce est biaisée à 99% pour la face :

  $H(PileOuFaceBiaise) = - (0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08$ *bits*

- L'entropie d'un dé à 4 faces non pipé

  $H(de\text{-}4) = - (0.25 \log_2 0.25 + 0.25 \log_2 0.25$
  $+ 0.25 \log_2 0.25 + 0.25 \log_2 0.25) = 2$

Entropie

- $H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = - \sum_k P(v_k) \log_2 P(vk)$

- Notons B(q), l'entropie d'une variable aléatoire binaire qui est vraie avec une probabilité q :

$$B(q) = - (q \log_2 q + (1 - q) \log_2 (1 - q))$$

- Par exemple :

$$H(PileOuFace) = B(0.5) = - (0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ bit}$$

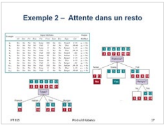
- Pour un arbre de décision, si un ensemble de données contient *p* exemples positifs et *n* exemples négatifs, l'entropie de la sortie de l'arbre de décision sur cet ensemble est:

$$H(Sortie) = B(\frac{p}{p+n})$$

Entropie

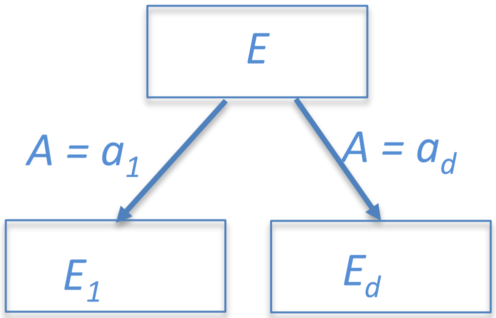
- $B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$ Entropie pour une variable booléenne vrai avec une probabilité *q*
- $H(Sortie) = B(\frac{p}{p+n})$ Entropie pour un ensemble de données, avec *p* exemples positifs et *n* exemples négatives
- Un attribut A avec d valeurs distinctes sépare l'ensemble d'entrainement E en sous-ensembles E_1 … E_d.
- Chaque sous-ensemble E_k a p_k exemples positifs et n_k exemples négatifs. Si on suit la $k^{ième}$ branche, on aura besoin de

  $B(\frac{p_k}{p_k+nk})$ bits supplémentaires pour répondre à la question



- Un exemple choisi aléatoirement sera dans E_k (c.-à-d., aura la $k^{ème}$ valeur de l'attribut) avec une probabilité $(\frac{p_k+nk}{p+n})$
- L'entropie restant après avoir testé A est donc

$$Remainder(A) = \sum_{k=1}^{d} (\frac{p_k+nk}{p+n})\ B(\frac{p_k}{p_k+nk})$$



Gain d'information et choix d'attribut

- $B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$ Entropie pour une variable booléenne vrai avec une probabilité *q*
- $H(Sortie) = B(\frac{p}{p+n})$ Entropie pour un ensemble de données, avec *p* exemples positifs et *n* exemples négatives
- L'entropie restant après avoir testé A est donc

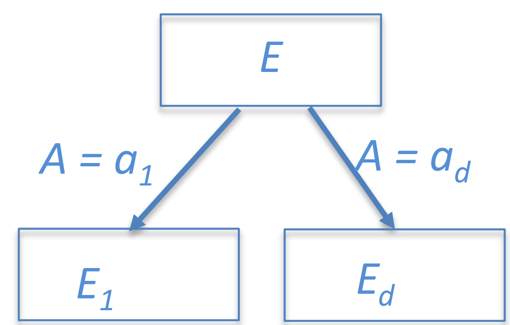
$$Remainder(A) = \sum_{k=1}^{d} \left(\frac{p_k + nk}{p+n}\right) B\left(\frac{p_k}{p_k+nk}\right)$$

- Le gain d'information dans l'attribut A est

$$Gain\ (A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

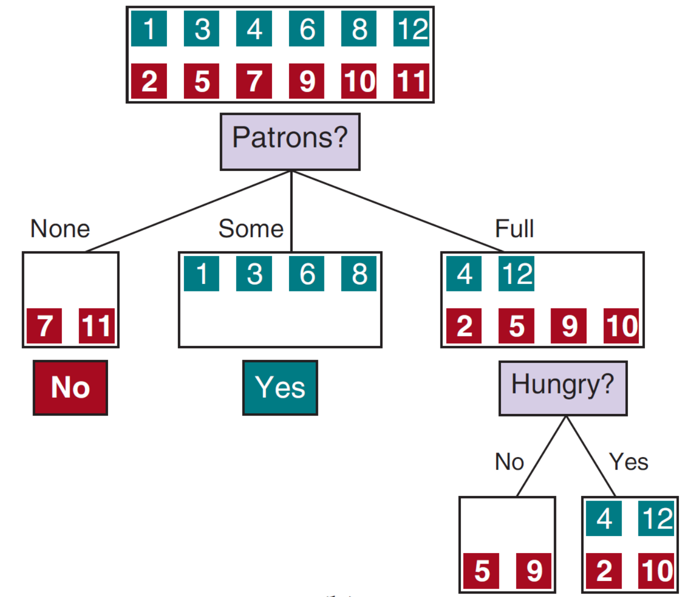
- La fonction *IMPORTANCE (a, examples)* dans l'algorithme retourne *Gain(a)*.
- Ainsi, on choisi l'attribut avec le plus de gain d'information.

E a *p* exemples positifs
et *n* exemples négatifs
E_k a p_k exemples positifs
et n_k exemples négatifs



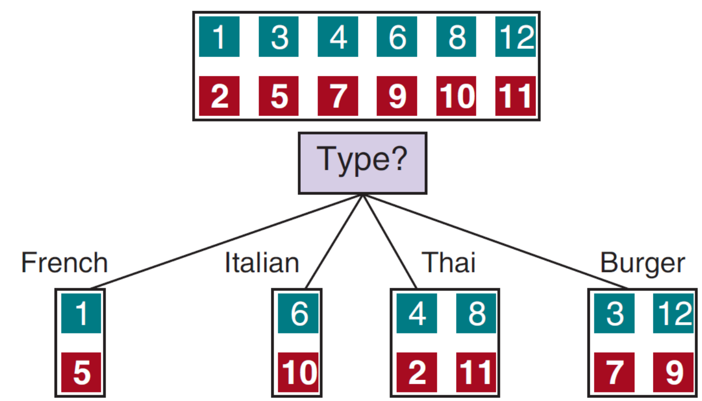
Exemple

- $B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$
- $H(Sortie) = B(\frac{p}{p+n})$
- $Remainder(A) = \sum_{k=1}^{d} (\frac{p_k + n_k}{p+n}) \; B(\frac{p_k}{p_k + nk})$
- $Gain\ (A) = B(\frac{p}{p+n})$ - $Remainder(A)$



- $Gain\ (Patrons) =$

$$1 - [\frac{2}{12} B(\frac{0}{2}) + \frac{4}{12} B(\frac{4}{4}) + \frac{6}{12} B(\frac{2}{6})] \approx 0.541 \ bits$$



- $Gain\ (Type) =$

$$1 - [\frac{2}{12} B(\frac{1}{2}) + \frac{2}{12} B(\frac{1}{2}) + \frac{4}{12} B(\frac{2}{4}) + \frac{4}{12} B(\frac{2}{4})]$$

$\approx 0 \ bits$

Généralisation et surapprentissage

Pour éviter le surapprentissage, plusieurs techniques sont utilisées.

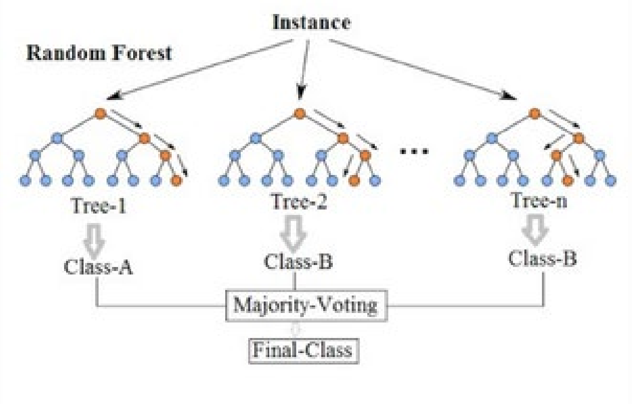
- Élaguer des nœuds qui ne paraissent pas pertinents.
  - En utilisant un test de signification statistique (voir Section 19.3.4; pas couvert pour l'examen)
  - En utilisant d'autres heuristiques:
    - Le nombre minimum d'exemples qu'un nœud doit avoir
    - La profondeur limite de l'arbre
    - Un ratio entre la classe minoritaire et la classe majoritaire.
- Réduction de dimensionnalité (e.g., en utilisant *PCA – Principal Component Analysis* – que nous ne voyons pas dans ce cours)

Généralisation et surapprentissage

- Les arbres de décisions sont sensibles aux petites variations dans les données.

- On peut atteindre plus de robustesse en utilisant plusieurs arbres de décision et en agrégeant leurs décisions pour avoir la décision finale. C'est ce qu'on appelle l'algorithme *random forest*.

Source : [wikipedia]



- C'est un cas particulier de bagging, qui est à son tour un cas particulier d'apprentissage ensembliste (ensemble learning)

Conclusion

- Un arbre de décision est une représentation symbolique d'un modèle d'apprentissage, dans laquelle flux d'exécution suit un chemin de l'arbre en effectuant des tests associées aux nœuds intérieurs pour arriver à une décision sur les feuilles.

- L'algorithme d'apprentissage base peut être améliorer de plusieurs façon pour éviter le surapprentissage et l'instabilité.

- *Random Forest* – une technique d'apprentissage ensembliste avec des arbres de décision -- est un des plus dans l'industrie.

Sujets couverts par le cours

Concepts et algorithmes

K-NN
Régression linéaire avec le Percepron
Régression logistique avec le Perceptron
Réseau de neurones
Abre de décision

Applications

Vision par ordinateur

Traitement du Langage naturel

agents intelligents

- Apprentissage automatique
- Raisonnement probabiliste
- Raisonnement logique
- Planification et jeu compétitifs
- Recherche heuristique globale
- Processus de décision de Markov
- Éthique et IA
- Recherche heuristique locale

Vous devriez être capable de...

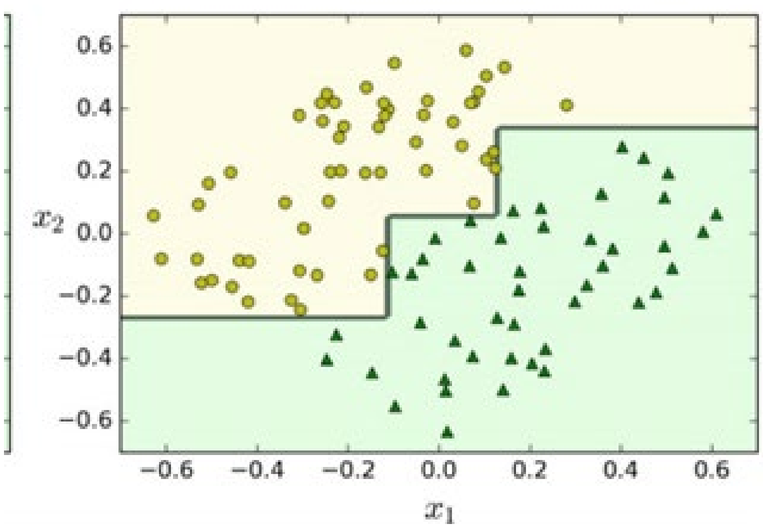
- Décrire ce qu'un arbre de décision.

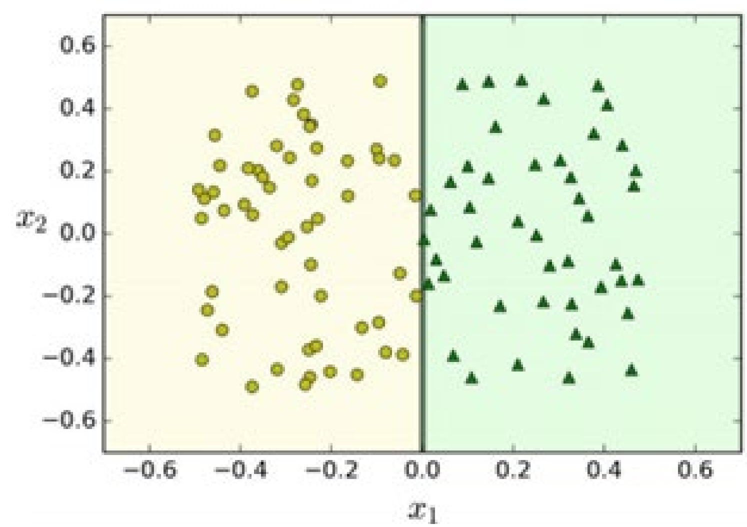
- Décrire et simuler l'algorithme d'apprentissage d'un arbre de décision sur un exemple.

    - Expliquer et appliquer le calcul de l'entropie et du gain d'information choix pour choisir le prochain attribut durant l'algorithme d'apprentissage.

Généralisation et surapprentissage

- Réduction de dimensionnalité (e.g., en utilisant *PCA – Principal Component Analysis* – que nous ne voyons pas dans ce cours)





Source: [Amelia, Towards Data Science, 2019]