

# Arbres de décision

## Module 5

---

# Objectifs

---

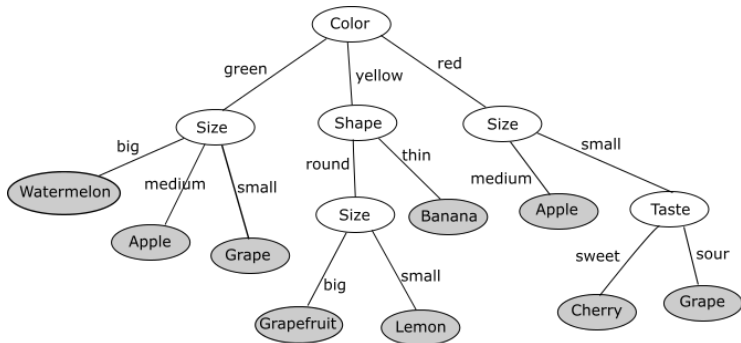
- construire un arbre de décision aussi bien pour la régression que pour la classification
- combiner plusieurs arbres efficacement avec Random Forest

# Arbres de décision

---

# Introduction

Modèle de classification ou régression qui classe un input dans une de ses feuilles pour rendre sa prédiction :



## Les arbres de décision

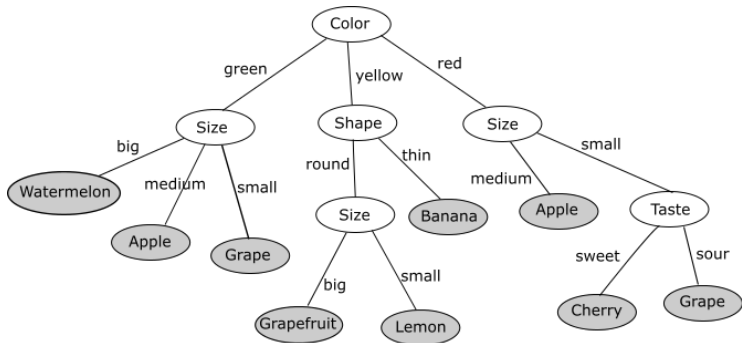
- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)
- sont interprétables
- sont très rapides durant l'inférence
- ne nécessitent pas de normalisation des données
- leur apprentissage est hautement parallélisable

→ Couteau-suisse du machine learning tabulaire.

- peuvent overfit les données, mais l'ensembling résoud ce problème
- sont sensibles aux déséquilibres de classe

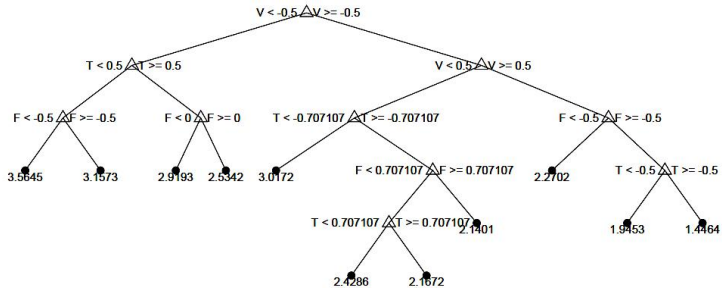
→ Si les classes ne sont pas équilibrées, peut-être les resampler.

# Arbres de classification





# Arbres de régression



Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set
- tant qu'il reste des nœuds non-traités :
  - choisir un nœud non traité
  - si le nœud remplit des conditions de feuille finale, ne rien faire
  - sinon, créer deux branches à partir du nœud non traité pour répartir les instances dans deux nouveaux nœuds

Conditions de feuilles finales : contient  $n_{min}$  éléments, est déjà à profondeur  $p_{max}$ , splitterait sans décroître assez l'entropie...

En fonction de la tâche, une fois arrivé dans la feuille de fin :

**Classification** classe majoritaire

**Régression** moyenne des valeurs cibles

Splits possibles d'une feature donnée :

**Catégorielle** chaque catégorie vs le reste

**Ordinale/Continue** milieu de chaque valeur ou quantiles

# Évaluation de la qualité d'un split

En fonction de la tâche :

**Régression** coût si on rendait la moyenne des instances comme résultat

$$Loss = \sum |\hat{y} - y| \approx variance$$

**Classification** Entropie de Shannon :

$$Loss = - \sum_{x \in X} P_x * \log_2(P_x)$$

$= 0 \Rightarrow$  il n'y a pas d'incertitude  
maximale quand on a une distribution uniforme

## Exemple — démarrage

ID, jardinage, jeux vidéos, chapeaux,  
âge

1	0	1	1	13
2	0	1	0	14
3	0	1	0	15
4	1	1	1	25
5	0	1	1	35
6	1	0	0	49
7	1	1	1	68
8	1	0	0	71
9	1	0	1	73

Première étape : création du nœud  
de départ

1, 2, 3, 4, 5, 6, 7, 8, 9

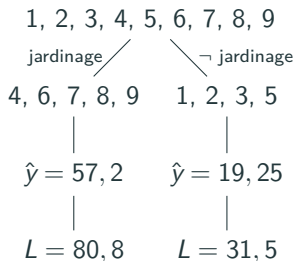
|

## Exemple — split

ID, jardinage, jeux vidéos, chapeaux,  
âge

1	0	1	1	13
2	0	1	0	14
3	0	1	0	15
4	1	1	1	25
5	0	1	1	35
6	1	0	0	49
7	1	1	1	68
8	1	0	0	71
9	1	0	1	73

Split du premier nœud. Il faut tester  
3 splits. Split sur jardinage :



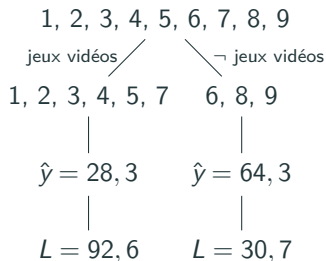
Loss totale : 122,3

## Exemple — split

ID, jardinage, jeux vidéos, chapeaux,  
âge

1	0	1	1	13
2	0	1	0	14
3	0	1	0	15
4	1	1	1	25
5	0	1	1	35
6	1	0	0	49
7	1	1	1	68
8	1	0	0	71
9	1	0	1	73

Split du premier nœud. Il faut tester  
3 splits. Split sur jeux vidéos :



Loss totale : 123,3

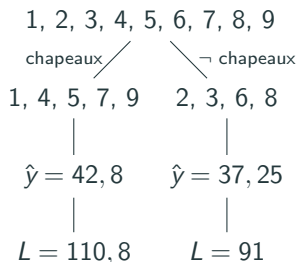


## Exemple — split

ID, jardinage, jeux vidéos, chapeaux,  
âge

1	0	1	1	13
2	0	1	0	14
3	0	1	0	15
4	1	1	1	25
5	0	1	1	35
6	1	0	0	49
7	1	1	1	68
8	1	0	0	71
9	1	0	1	73

Split du premier nœud. Il faut tester  
3 splits. Split sur chapeaux :



Loss totale : 201,8

## Exemple — split

ID, jardinage, jeux vidéos, chapeaux,  
âge

1	0	1	1	13
2	0	1	0	14
3	0	1	0	15
4	1	1	1	25
5	0	1	1	35
6	1	0	0	49
7	1	1	1	68
8	1	0	0	71
9	1	0	1	73

**122,3** jardinage

**123,3** jeux vidéos

**201,8** chapeaux

→ On split donc sur jardinage

## Exemple — split

ID, jardinage, jeux vidéos, chapeaux,  
âge

1	0	1	1	13
2	0	1	0	14
3	0	1	0	15
4	1	1	1	25
5	0	1	1	35
6	1	0	0	49
7	1	1	1	68
8	1	0	0	71
9	1	0	1	73

Résultat après le premier split :

1, 2, 3, 4, 5, 6, 7, 8, 9  
jardinage /      \ ¬ jardinage  
4, 6, 7, 8, 9      1, 2, 3, 5

À vous de jouer !

Fait par :

- la profondeur maximum
- le nombre minimum d'instances dans chaque feuille
- une baisse d'entropie maximale à chaque split
- le nombre minimum d'instances pour split
- le pruning

# Random Forest

---

- les arbres de décision overfit facilement
- ils sont rapides à apprendre
- en combiner beaucoup est faisable et réduit la variance

→ création d'une forêt (ensemble d'arbres) aléatoire

Produire des arbres décorrélés et moyenner leurs prédictions pour réduire la variance.

# Outil 1 — bagging (row sampling)

Bootstrap **aggregating** (Bagging) :

- tirer un échantillon du dataset avec replacement
- entraîner un arbre sur cet échantillon
- répéter  $B$  fois

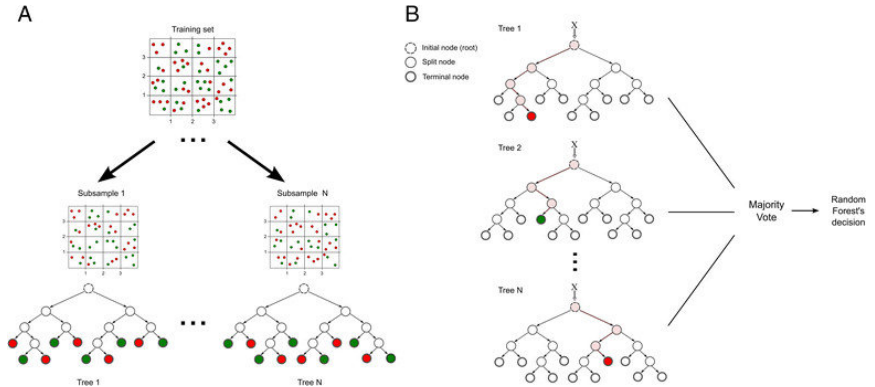
Le bagging s'appelle aussi row sampling.



## Outil 2 — random subspace method (column sampling)

- à chaque split, considérer seulement un sous-ensemble des features
- valeurs conseillées :
  - classification :  $\lfloor \sqrt{m} \rfloor$  features par split
  - régression :  $\lfloor \frac{m}{3} \rfloor$  features par split, 5 exemples par node minimum

# Random Forest



- Pas de sur-apprentissage en augmentant le nombre d'arbres
- Une fois appris, le modèle est très rapide

## Conclusion

---

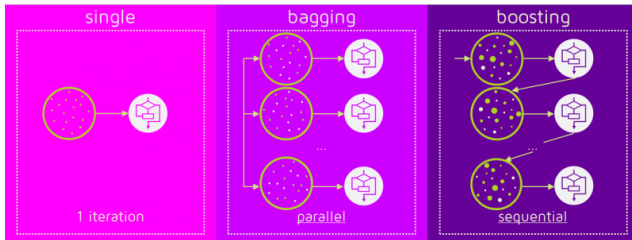
- les arbres sont interprétables, rapides à entraîner, combinables.
- random forest combine des arbres faibles en un prédicteur versatile

# Gradient boosted trees

---

# Introduction

Arbres qui s'améliorent successivement.



[quantdare.com/what-is-the-difference-between-bagging-and-boosting/](https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/)

# Approximation 1

1. partir d'un arbre grossier
2. entraîner un nouvel arbre sur les résiduels du premier
3. concaténer le nouvel arbre au premier
4. goto 2.

Quel est l'effet d'entraîner sur les résiduels ?



1. partir d'un arbre grossier
2. entraîner un nouvel arbre sur les **pseudo-résiduels** du premier
3. concaténer le nouvel arbre au premier
4. goto 2.

- choisir une fonction de coût
- calculer les pas de descente de gradient étant donné les couples  $(\hat{y}_i, y_i)$
- **se servir de ces valeurs comme de résiduels**

→ Intérêt : pouvoir utiliser n'importe quel loss dérivable.

# Approximation 3

1. partir d'un arbre grossier
2. entraîner un nouvel arbre sur les pseudo-résiduels du premier
3. **calculer un multiplicateur pour que l'arbre produit minimise le coût**
4. concaténer le nouvel arbre au premier
5. goto 2.

1. partir d'un arbre grossier
2. entraîner un nouvel arbre sur les pseudo-résiduels du premier
3. calculer un multiplicateur pour que l'arbre produit minimise le coût
4. **appliquer un learning rate**
5. concaténer le nouvel arbre au premier
6. goto 2.

- séquence d'arbres qui s'entraînent à corriger les erreurs de l'arbre d'avant
- modélisation de la correction de l'erreur par un pas de descente de gradient pour plus de flexibilité.

- row sampling
- column sampling
- tree structure cost

- modèle extrêmement performant et versatile
- entraînement parallélisable

Sujet à l'overfit si pas assez régularisé (tree structure cost) et randomisé (row & column sampling)