Arbres de décision

Daniel Aloise <daniel.aloise@polymtl.ca>

8 octobre 2020

Arbre de décision

permettent de faire de la classification

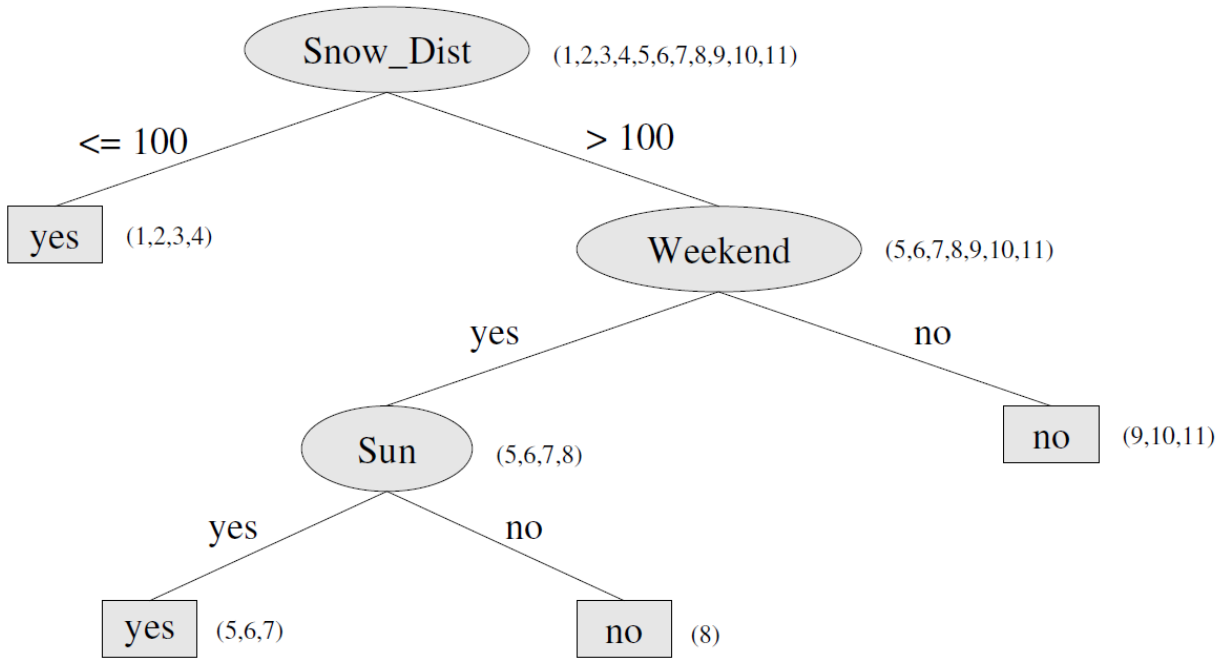
- Méthode simple qui donne très souvent de bons résultats
- Par un séquence de décisions, on arrive à la fin à classer l'exemple fourni
- À chaque étape de la séquence de prise de décision, on se prononce sur une seule des features

Ensemble d'entraînement

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

Arbre de décision

moins ou égal de 100 km, oui on va aller skier



- Pour $Snow\_Dist > 100$, $Weekend = yes$ et $Sun = yes$, la réponse est yes
- Le but est de trouver l'arbre avec l'erreur minimum (l'arbre qui minimise l'erreur)

Ensemble d'entraînement

- Remarque que dans l'ensemble d'entraînement présenté les décisions sur les journées 6 et 7 sont contradictoires

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

3 personnes sont allé skier

1 personne n'est pas allé

Arbres de décision

Algorithme exhaustive optimal créer tous les arbres, les tester sur les données et enfin choisir l'arbre avec l'erreur minimale.

Désavantage temps de calcul prohibitive pour de données possédant beaucoup de features

Solution heuristique gloutonne

au lieu de maximiser le gain final, on va maximiser le gain de la prochaine étape

Apprentissage d'arbre de décision

valeur par defaut. Ex: non , on ne va pas skier car il y a + de gens qui ne vont pas skier

fonction CREER-ARBRE-DECISION(*exemples, features, défaut*)
  si *exemples* = ∅ **retourner** *défaut*
  sinon si tous les exemples ont la même classification **alors**
    retourner cette classification
  sinon si features = ∅ **alors**
    retourner la valeur majoritaire parmi les exemples
  sinon

    meilleur_feature ← CHOISIR-FEATURE*(features, exemples)*

    arbre ← nouvel arbre avec attribut *meilleur_feature* comme racine

    m ← la valeur majoritaire parmi les exemples
    pour chaque valeur v_i de *meilleur_feature* **faire**

      $exemples_i$ ← { e ∈ *exemples* | valeur de *meilleur_feature* pour *e* = v_i }
      features' = *feature* – {*meilleur_feature* }
      sous-arbre = CREER-ARBRE-DECISION($exemples_i$,*features', m*)
      ajouter une branche à *arbre* avec attribut v_i et *sous-arbre*
    retourner *arbre*

Comment choisir le meilleur feature ?

- L'entropie mesure le montant d'incertitude d'une distribution de probabilité

Entropie

$$H(p(v_1), \ldots, p(v_k)) = \sum_{i=1}^{k} -p(v_i) \log(p(v_i))$$

- Exemple : une pièce de monnaie

IMPORTANT
si jamais 2 possibilités on utilise Log base 2
si 3 possiblités on utilise Log base 3

$$H(p(0), p(1)) = -\frac{1}{2}\log(1/2) - \frac{1}{2}\log(1/2) = 1$$

- Supposons maintenant une monnaie "chargée" dont la distribution de probabilité est (2/3,1/3)

$$H(p(0), p(1)) = -\frac{2}{3}\log(2/3) - \frac{1}{3}\log(1/3) \approx 0.90$$

Comment choisir le meilleur feature ?

- Soit N le nombre d'exemples de notre jeu de données
- Soit un feature A ayant les valeurs possibles $a_1, \ldots, a_k$
- Supposons que l'ensemble d'exemples contient n_i items pour chaque valeur possible a_i
- Initialement la valeur de l'entropie est

$$H = \sum_{i=1}^{k} -\frac{n_i}{N} \log\left(\frac{n_i}{N}\right)$$

- Notez que $\frac{n_i}{N}$ est une approximation de la probabilité $p(A = a_i)$

Comment choisir le meilleur feature ?

- On appele gain (G) la différence entre l'entropie initiale et l'espérance de l'entropie après que le feature A est choisi pour la séparation

- Notre critère glouton dans l'algorithme de construction d'une arbre de décision va toujours choisir le feature qui maximise le gain

Exemple

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

- L'entropie initiale vaut

$$H(6/11, 5/11) = 0.994$$

incertitude très élevée

Exemple

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

- L'entropie initiale vaut

$$H(6/11, 5/11) = 0.994$$

- Si on sépare avec le feature *Snow_Dist*, l'espérance d'entropie résultante vaut :

$$\frac{4}{11} H(Snow\_Dist \leq 100) + \frac{7}{11} H(Snow\_Dist > 100)$$

Exemple

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

- $H(Snow\_Dist \leq 100) = H(1, 0) = 0$

  4/4, donc (1,0) car Skiing est tout yes pour les 4

Exemple

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

- $H(Snow\_Dist > 100) = H(2/7, 5/7) = 0.863$

Exemple

- Alors, pour *Snow_Dist* le gain est de :

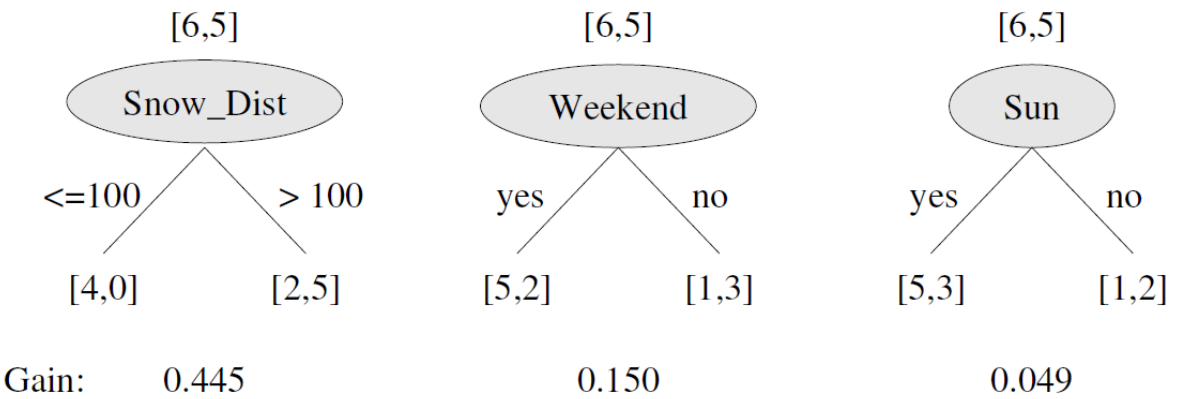
$$G(Snow\_Dist) = 0.994 - \left(\frac{4}{11} \times 0 + \frac{7}{11} \times 0.863\right) = 0.445$$

à quel point je gagne en certitude en choisissant cet attribut là

Exemple

- Alors, pour *Snow_Dist* le gain est de :

$$G(Snow\_Dist) = 0.994 - \left(\frac{4}{11} \times 0 + \frac{7}{11} \times 0.863\right) = 0.445$$



- *Snow_Dist* est bien le feature choisi

Exemple

- Les deux valeurs pour le feature *Snow_Dist*, i.e. ≤ 100 et > 100 génèrent deux arêtes dans l'arbre
- Pour le sous-ensemble $Snow\_Dist \leq 100$ la classification est bien sûr yes
- Pour $Snow\_Dist > 100$, il n'y a pas un résultat évident
- Donc, l'algorithme continue de façon récursive

à faire d'autres itérations

Exemple

- À partir des deux features encore disponibles *Sun* et *Weekend*, le meilleur doit être choisi selon le critère du gain
- Notez que notre analyse est maintenant restreinte aux sept exemples pour lesquels $Snow\_Dist > 100$

Day	*Snow_Dist*	*Weekend*	*Sun*	*Skiing*
1	≤ 100	yes	yes	yes
2	≤ 100	yes	yes	yes
3	≤ 100	yes	no	yes
4	≤ 100	no	yes	yes
5	> 100	yes	yes	yes
6	> 100	yes	yes	yes
7	> 100	yes	yes	no
8	> 100	yes	no	no
9	> 100	no	yes	no
10	> 100	no	yes	no
11	> 100	no	no	no

Exemple

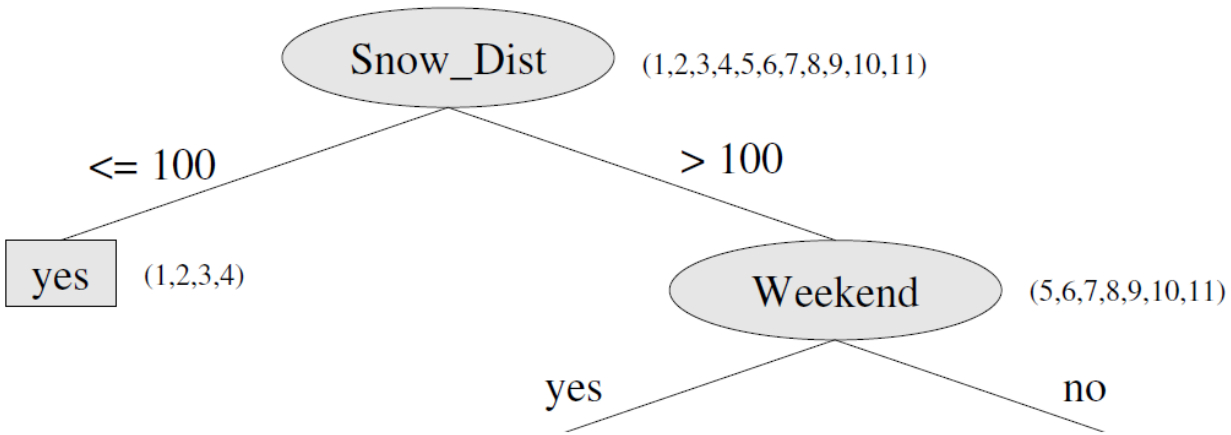
- À partir des deux features encore disponibles *Sun* et *Weekend*, le meilleur doit être choisi
- On calcule donc :

$$\begin{aligned} G(D_{>100}, \mathit{Weekend}) &= 0.863 - \left(\frac{4}{7} \times H(2/4, 2/4) + \frac{3}{7} \times H(1, 0)\right) \\ &= 0.292 \end{aligned}$$

$$\begin{aligned} G(D_{>100}, \mathit{Sun}) &= 0.863 - \left(\frac{5}{7} \times H(2/5, 3/5) + \frac{2}{7} \times H(1, 0)\right) \\ &= 0.170 \end{aligned}$$

Exemple

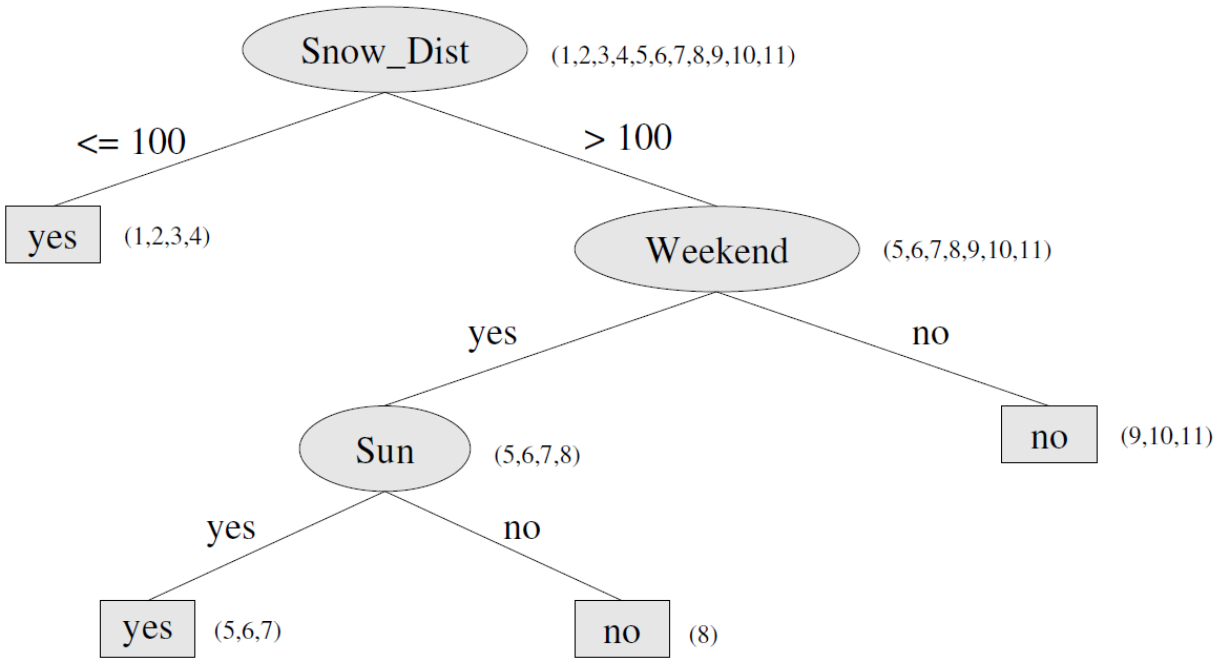
- Alors, Weekend est choisi comme racine de la nouvelle sous-arbre



Exemple

- Pour *Weekend* = *no*, l'arbre finit avec la décision no
- Pour *Weekend* = *yes*, le feature *Sun* mene à un gain de 0.171
- La construction de l'arbre finit parce qu'il nous reste plus de features à considérer

Exemple



Sommaire

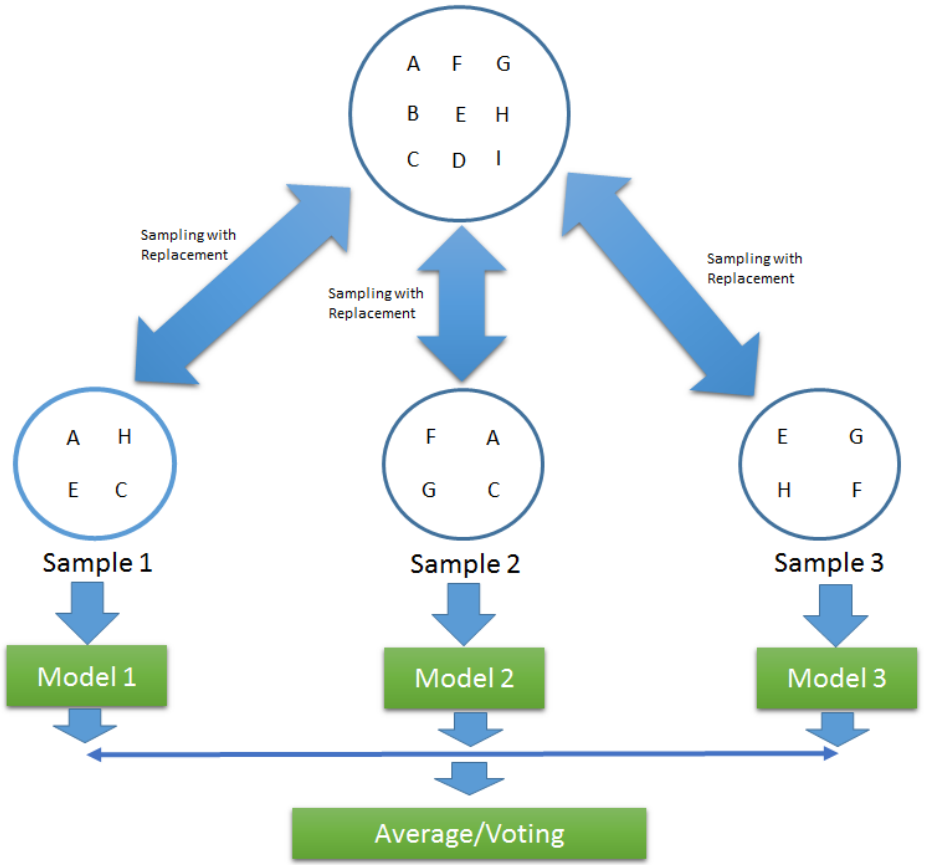
- Méthode très populaire pour l'apprentissage supervisé
- Facile à utiliser et rapide
- L'usager peut "comprendre" l'arbre de décision
- Peut aussi souffrir d'un surapprentissage
- 1) Mieux vaut s'arrêter lorsque le gain d'information est faible, au lieu de zéro
- 2) Une autre stratégie consiste à construire l'arbre complet, puis à éliminer quelques ramifications de gain faible

Ensemble d'arbres de décision

- Combine plusieurs arbres de décision pour produire une meilleure performance prédictive
- Le principe de base du modèle d'ensemble est qu'un groupe de modèles faibles se réunit pour former un modèle fort
- Le vote majoritaire entre plusieurs classificateurs augmente la robustesse et nous permet de quantifier notre niveau de confiance
  - il y a une grande différence dans la classe majoritaire apparaissant dans 501 des 1000 arbres par rapport à 947 d'entre eux

Bagging

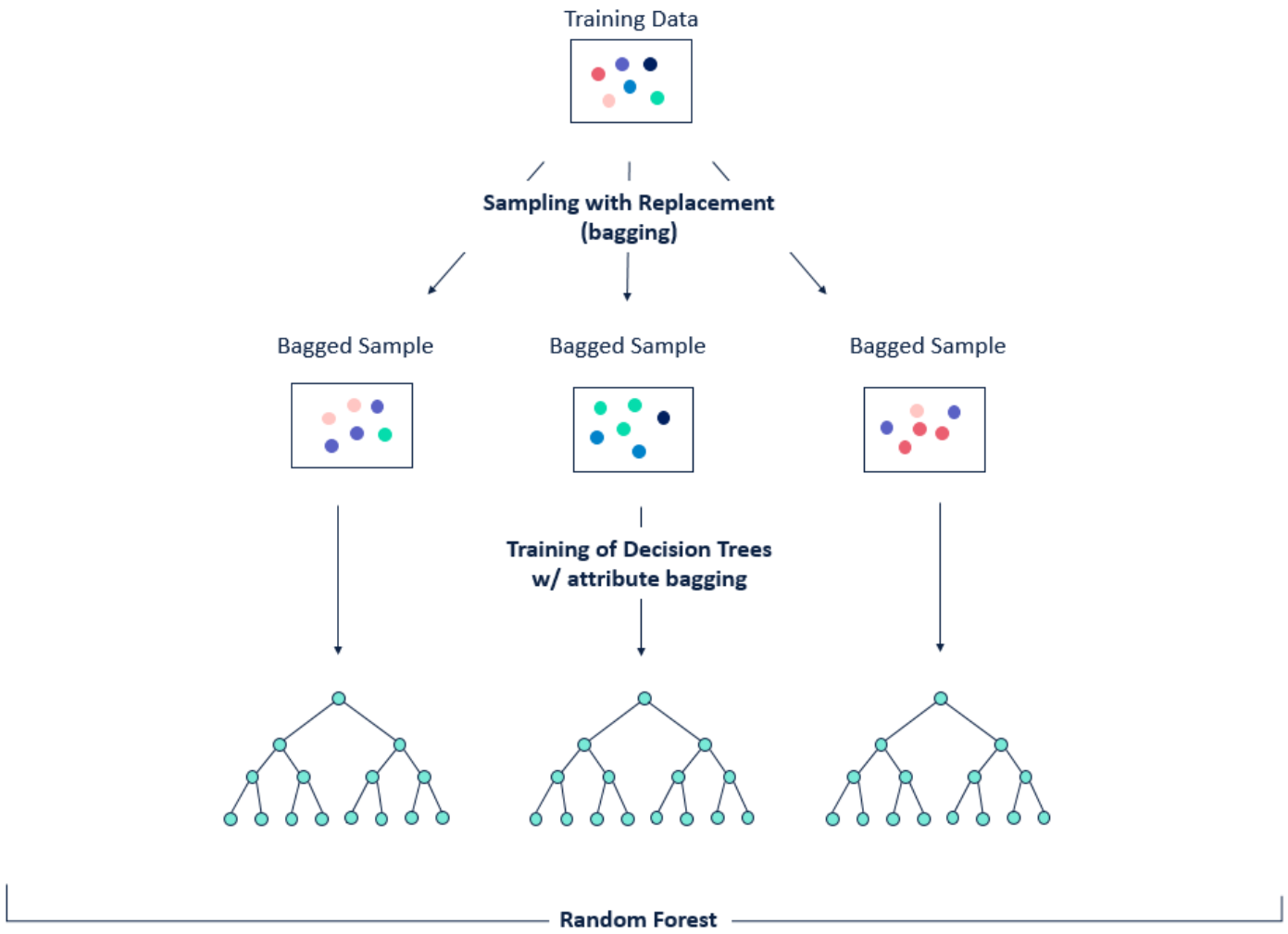
- Bagging choisit des sous-ensembles de données choisis au hasard pour entraîner chaque arbre
- Diminue le surapprentissage



Forêt d'arbres de décision

- Extension de la technique de bagging
- Effectue également la sélection aléatoire des features plutôt que d'utiliser toutes les features pour construire les arbres.

Forêt d'arbres de décision



Forêt d'arbres de décision

- Diminue la corrélation entre les arbres
- Un feature très dominant oblige chaque arbre de décision à le choisir pour les premiers splits, ce qui fait que tous les arbres se comportent de façon similaire
- La forêt d'arbres est composée par des arbres de décision non corrélés.