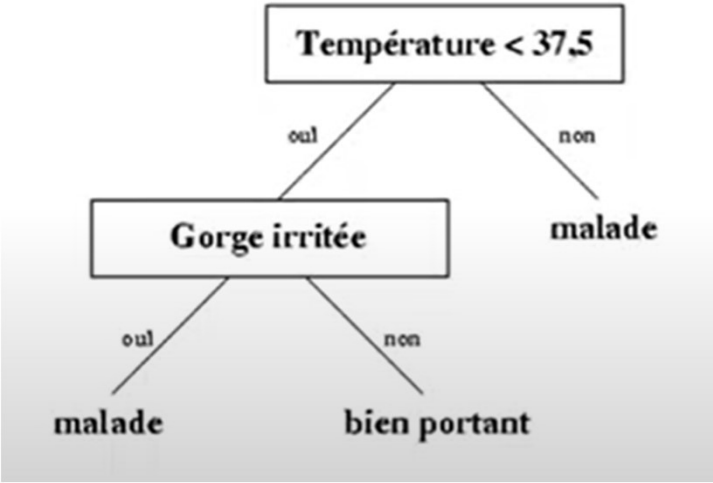
Les arbres de décision

Plan

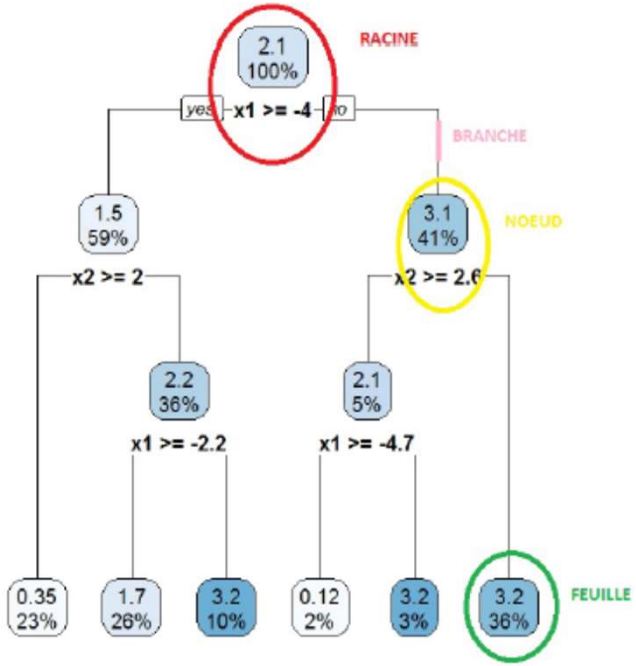
- Principe et concepts de base
- CART pour la classification
- CART pour la régression
- Optimisation
- Autres Arbres de décision
- Extensions

Arbres de Décision: **Principes de base**

- ✓ L'arbre de décision est un célèbre algorithme de Machine Learning **supervisé,**
- ✓ L'arbre de décision est utilisé à la fois pour la
  - Classification
  - Régression



Arbre de classification



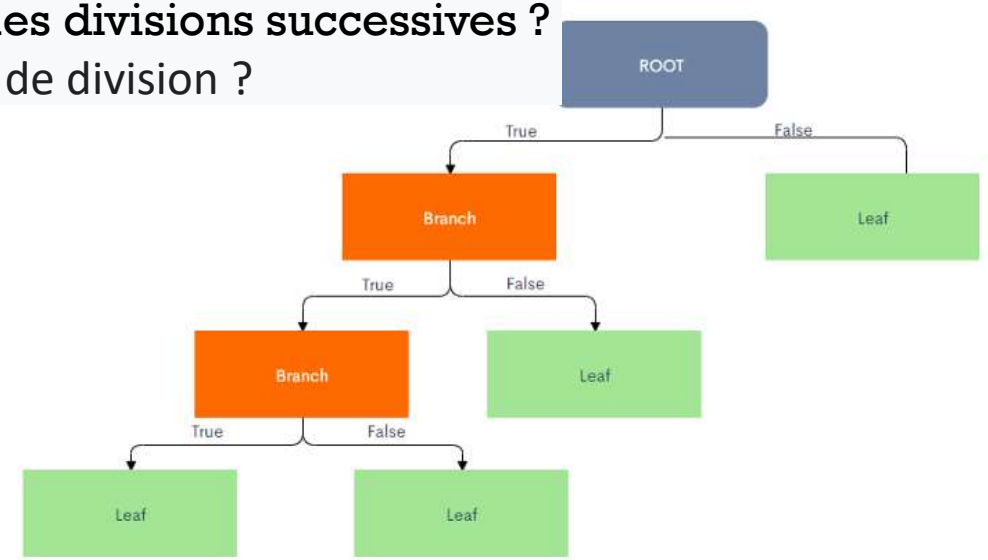
Arbre de Régression

Arbres de Décision: **Terminologies**

- ✓ **Nœud racine** : C'est le nœud de début de l'arbre à partir duquel toute la population se divise en fonction de différentes caractéristiques.
- ✓ **Nœud de branche** : Il s'agit du nœud situé sous le nœud racine qui se divise davantage.
- ✓ **Nœud feuille** : Le nœud qui ne peut pas être divisé davantage pour prendre des décisions

- ✓ Un nœud, autre que les feuilles, présente une condition évaluée à **true** ou **false**,
- ✓ Par défaut, lorsque la condition est True, la sous-branche est affichée à gauche et lorsque la condition est False, elle est affichée à droite.

Critère de construction: **Comment définir les divisions successives ?**
Condition d'arrêt: Quand arrêter le principe de division ?



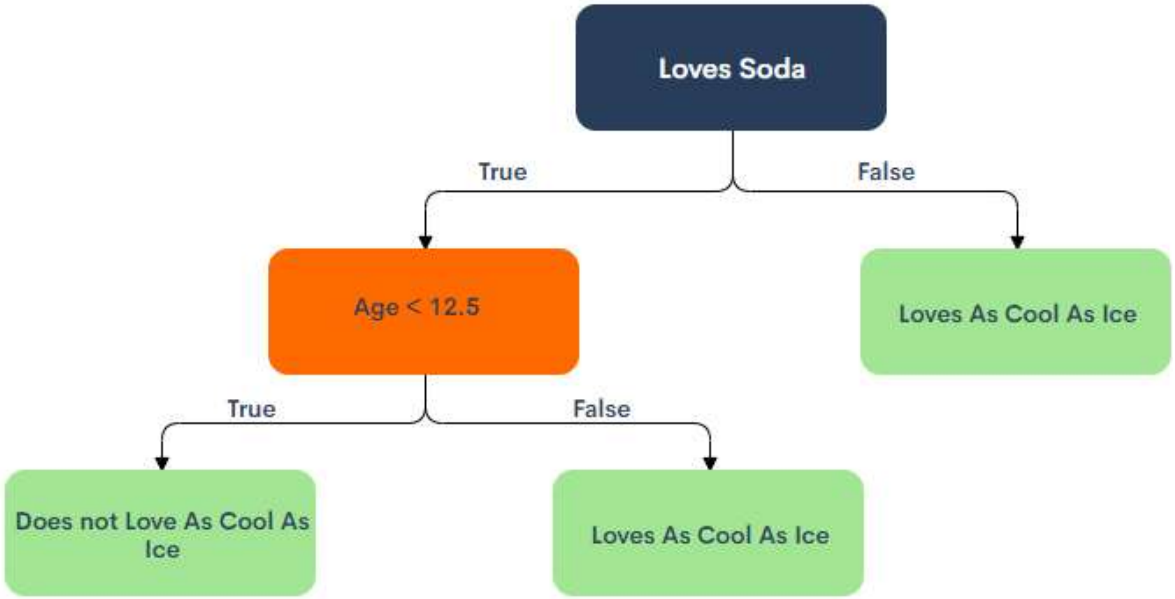
Arbres de Décision: **Terminologies**

- ✓ Chaque nœud teste un attribut (feature),
- ✓ Chaque branche correspond à une valeur d'attribut,
- ✓ Chaque feuille correspond à une classe unique ou à une classe majoritaire,

- ✓ Le choix de la racine = La sélection du feature le plus pure (le plus homogène)

- ✓ Le choix d'un nœud de branche= La sélection du feature le plus pure parmi les features restants,

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Arbres de Décision: **Algorithme**

Procédure : construire-arbre(X)

- Si tous les points de X appartiennent à la même classe alors créer une feuille portant le nom de cette classe

Sinon

- choisir le meilleur attribut pour créer un nœud
- le test associé à ce noeud sépare X en des parties : X1....Xn
- construire-arbre(X1)
- .....
- construire-arbre(Xn)

Fin

Arbres de Décision: **Algorithme**

- Déroulement de la construction :

- Recherche de la variable qui sépare le mieux
- Applique la séparation à la population
- Obtention de nouveaux nœuds

- Arrêt de l'approfondissement de l'arbre lorsque les conditions d'arrêts sont rencontrées

Arbres de Décision: **Algorithme**

- **Conditions d'arrêt**

✓ **Arriver à des feuilles pures**
✓ **Atteindre un seuil de nombre de populations où on arrête de diviser (*min_Samples_split*)**
✓ **Il n'y a plus de features à explorer**
✓ **Atteindre une profondeur déterminée (*max_depth*)**

Arbres de Décision: **Algorithme**

- **Comment choisir le meilleur attribut pour la division?**

Pour CART, gini index à minimiser
Pour ID3, Information gain à maximiser

Arbres de Décision: **l'algorithme CART**

CART: **C**lassification **A**nd **R**egression **T**ree

Pour sélectionner les nœuds les plus pures parmi les features, on utilise:

- ✓ Pour la classification: **GINI index**
  - ✓ C'est un index qui calcule l'impureté d'un feature,
  - ✓ Compris entre 0 et 1,
  - ✓ 0 correspond à un feature pure, 1 correspond à un feature totalement impure
- ✓ Pour la régression: **RSS (Residual Sum of Squares) ou MSE (Mean Squared Error)**
  - ✓ RSS et MSE calculent l'impureté d'un feature,
  - ✓ Le choix d'un nœud correspond à la sélection du feature qui minimise RSS (ou MSE)

Algorithme CART **Pour Classification: Exemple**

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

« Loves Cool As Ice » = variable target et les autres features= les variables indépendantes.
Ici, nous essayons de prédire si un individu « Loves Cool As Ice » en se basant sur les autres paramètres à l'aide de CART.

Ainsi, la première étape consiste à décider quelle variable est la plus appropriée pour le nœud racine, ce qui se fait en comprenant si une variable particulière est capable de prédire la variable cible de la manière la plus efficace ou non.

Algorithme CART **Pour Classification: Exemple**

1ère étape: quel Feature est le plus approprié pour le nœud racine,
Calculer le Gini Index pour chaque feature

$$Gini_{impurity} = \sum_{i=1}^{m} p_i \times (1 - p_i) = \sum_{i=1}^{m} p_i - {p_i}^2 = \sum_{i=1}^{m} p_i - \sum_{i=1}^{m} {p_i}^2 = \mathbf{1 - \sum_{i=1}^{m} {p_i}^2}$$

Avec m le nombre total de classes

Dans notre cas, on a deux classes:
Yes or No classes,

$$Gini_{impurity} = \mathbf{1 - \sum_{i=1}^{m} {p_i}^2 = 1 - {p_{yes}}^2 - {p_{No}}^2}$$

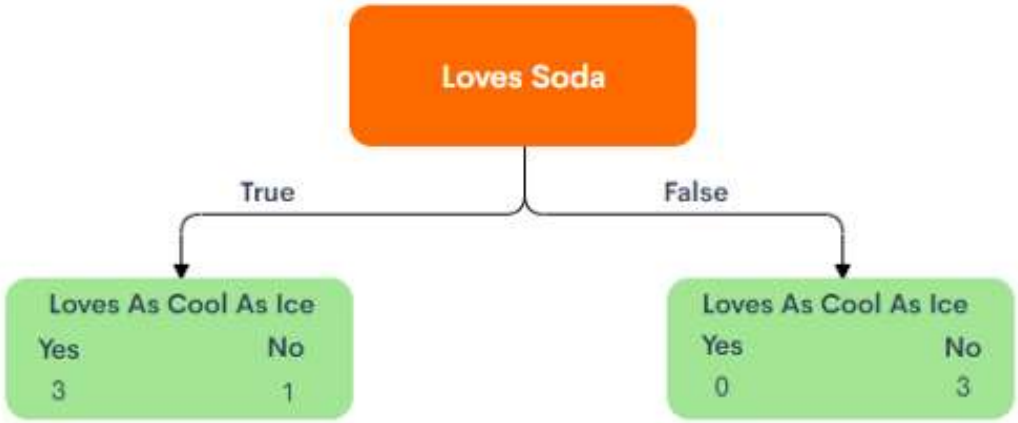
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Algorithme CART **Pour Classification: Exemple**

1ère étape: quel Feature est le plus approprié pour le nœud racine,
Calculer le Gini Index pour chaque feature

$$Gini_{impurity} = 1 - \sum_{i=1}^{m} {p_i}^2 = 1 - {p_{yes}}^2 - {p_{No}}^2$$

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Gini Impurity = $1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2$ =0,375

$$Gini\ Impurity = 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0$$

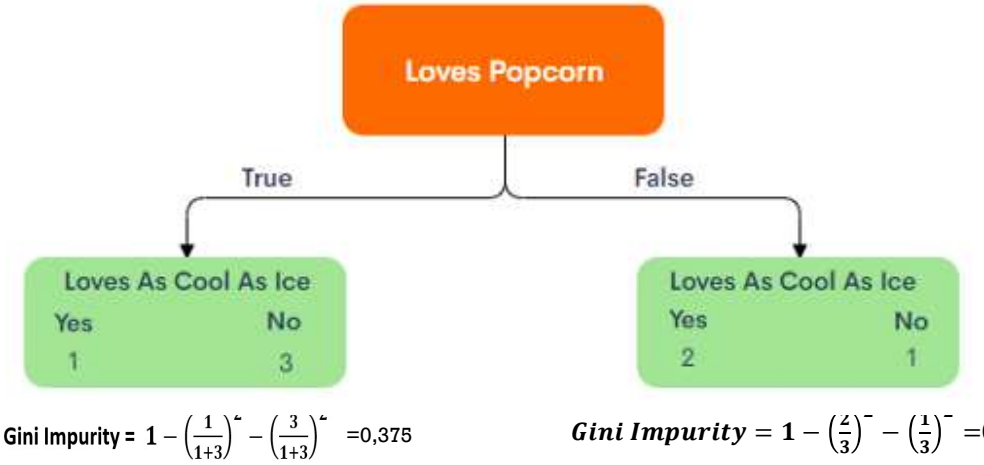
Total Gini impurity d'un noeud=moyenne des gini index multiplié par leurs poids

$$Gini\ impuriy = \left(\frac{4}{4+3}\right) \times 0{,}375 + \left(\frac{3}{4+3}\right) \times 0 = 0{,}214$$

Algorithme CART **Pour Classification: Exemple**

1ère étape: quel Feature est le plus approprié pour le nœud racine,
Calculer le Gini Index pour chaque feature

$$Gini_{impurity} = 1 - \sum_{i=1}^{m} {p_i}^2 = 1 - {p_{yes}}^2 - {p_{No}}^2$$



Gini Impurity = $1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2$ =0,375

$Gini\ Impurity = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2$ =**0,455**

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

Total Gini impurity d'un noeud=moyenne des gini index multiplié par leurs poids

$$Gini\ impuriy = \left(\frac{4}{7}\right) \times 0{,}375 + \left(\frac{3}{7}\right) \times 0{,}455 = 0{,}409$$

Algorithme CART **Pour Classification: Exemple**

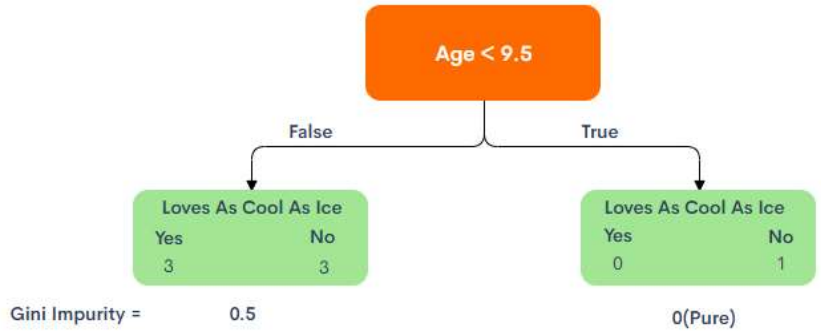
1ère étape: quel Feature est le plus approprié pour le nœud racine,
Calculer le Gini Index pour chaque feature

Question: Que faire pour une colonne numérique

	Age	Loves Cool As Ice
	7	No
9.5		
	12	No
15		
	18	Yes
26.5		
	35	Yes
36.5		
	38	Yes
44		
	50	No
66.5		
	83	No

La moyenne de chaque deux paires doit être calculée,

Considérer chaque moyenne comme un seuil et calculer le gini index pour chaque cas,



Total Gini Impurity: $-\frac{6}{7}(0.5) + \frac{1}{7}(0)$

Algorithme CART **Pour Classification: Exemple**

1ère étape: quel Feature est le plus approprié pour le nœud racine,
Calculer le Gini Index pour chaque feature

Question: Que faire pour une colonne numérique comme Age?

	Age	Loves Cool As Ice	Gini Impurity
	7	No	
9.5			0.429
	12	No	
15			0.343
	18	Yes	
26.5			0.476
	35	Yes	
36.5			0.476
	38	Yes	
44			0.343
	50	No	
66.5			0.429
	83	No	

Ainsi, comme nous pouvons le voir ici, l'impureté Gini de 0,343 est minimale, donc dans ce cas, 15 ou 44 peuvent être choisis comme valeur de seuil.

Algorithme CART **Pour Classification: Exemple**

1ère étape: quel Feature est le plus approprié pour le nœud racine,
Calculer le Gini Index pour chaque feature

Donc, maintenant, afin de décider du nœud racine final,
Gini- Impurity de tous les features indépendants est comparé.

Feature names	Total Gini Impurity
Loves Popcorn	0.405
Loves Soda	0.214
Age < 15	0.343

Ainsi, dans ce cas, le « Loves Soda » devient le nœud racine de l'arbre de décision.

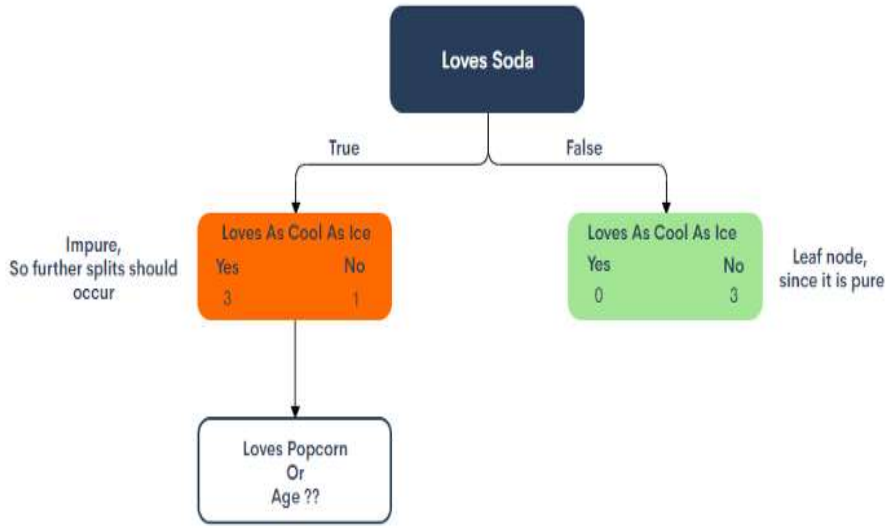
Algorithme CART **Pour Classification: Exemple**

2ème étape: Continuer le même processus pour trouver les autres branches,

- ✓ **Il faut penser à ne considérer que les lignes qui suivent la condition du nœud racine.**

Conditions d'arrêt :

- ✓ **Arriver à des feuilles pures ou bien**
- ✓ **Atteindre un seuil de nombre de populations où on arrête de diviser ou**
- ✓ **Il n'y a plus de features à explorer**
- ✓ **Atteindre une profondeur déterminée**



Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

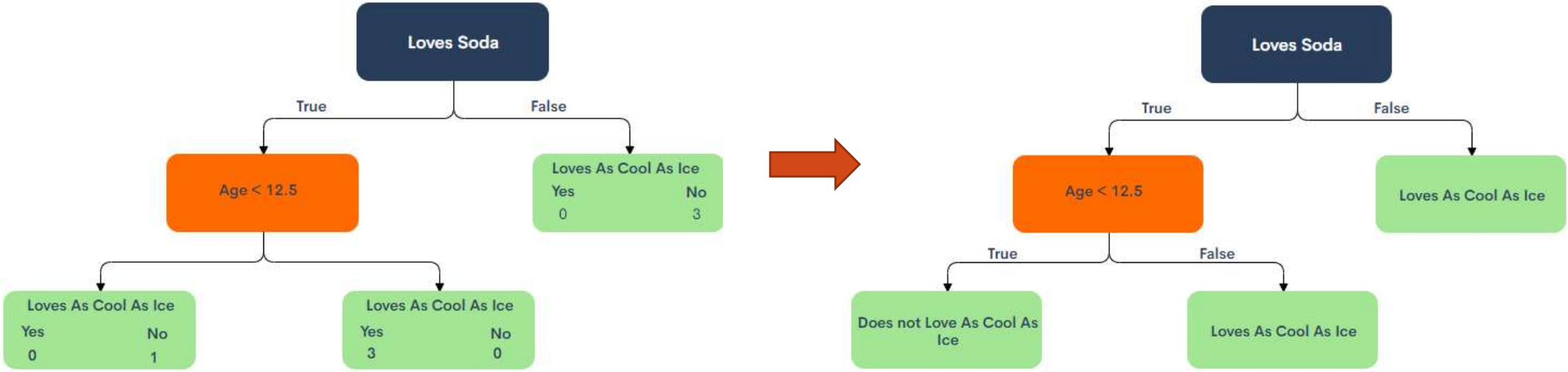
Arbre de décision: **l'algorithme CART: Exemple**

2ème étape: Continuer le même processus pour trouver les autres branches,

- ✓ **Il faut penser à ne considérer que les lignes qui suivent la condition du nœud racine.**

Conditions d'arrêt :

- ✓ **Arriver à des feuilles pures ou bien**
- ✓ **Atteindre un seuil de nombre de populations où on arrête de diviser ou**
- ✓ **Il n'y a plus de features à explorer**
- ✓ **Atteindre une profondeur déterminée**



Algorithme CART **Pour Régression**

Et si la valeur target était une valeur numérique? Comment procéder?

Study hours one week before exam	Sleeping hours last night	Gender	Marks
2	8	M	20
20	4	F	30
20	7	M	80
40	3	F	50
17	8	M	77

Algorithme CART **Pour Régression**

Répéter

1. Chaque feature divise les données en deux parties, Calculer la moyenne des targets des Deux divisions,

2. Calculer pour chaque feature la RSS (ou MSE) par rapport à la moyenne de chaque division

3, Prendre comme nœud, le feature dont le RSS (ou MSE) est minimal

Jusqu'à condition d'arrêt

RSS=Somme des carrés de la différence entre la valeur moyenne et la valeur prédite.
MSE=Moyenne des carrés de la différence entre la valeur moyenne et la valeur prédite

Condition d'arrêt = - Atteindre un nombre seuil de population à ne pas diviser ou
- Atteindre une profondeur bien déterminée
- Il n y a plus de features à explorer

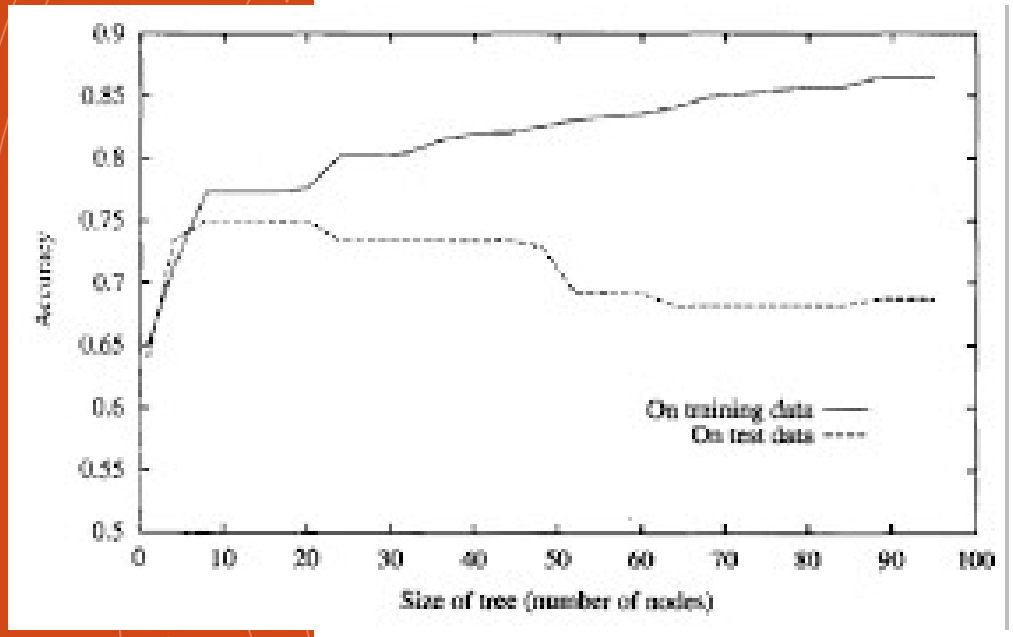
Algorithme CART **Pour Régression**

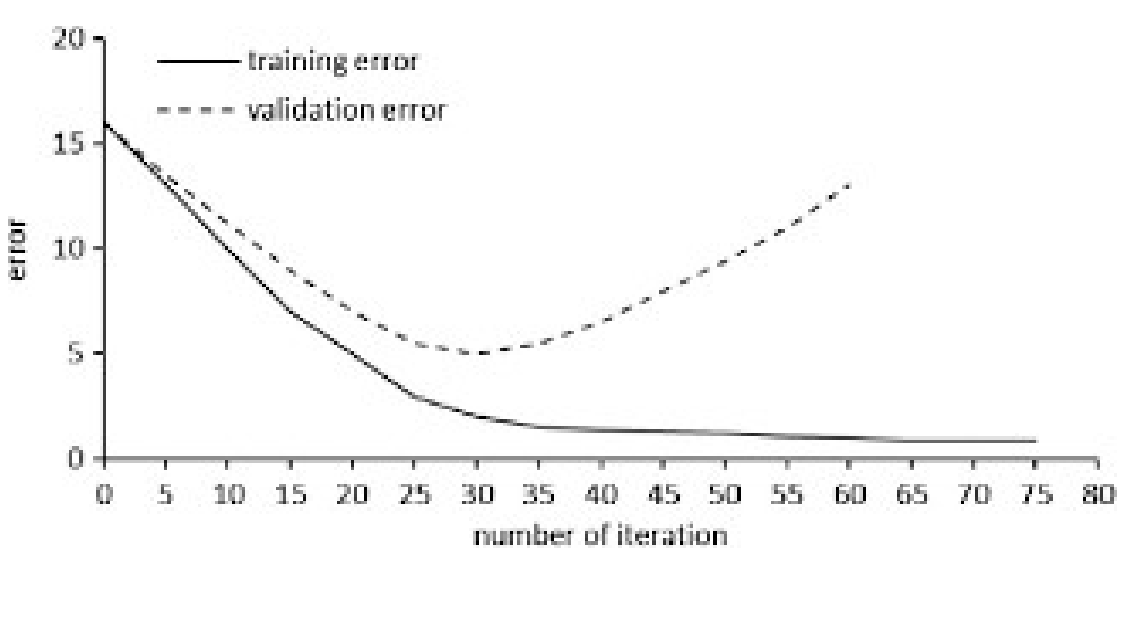
Exercice: Construire l'arbre CART de régression suivante:

Study hours one week before exam	Sleeping hours last night	Gender	Marks
2	8	M	20
20	4	F	30
20	7	M	80
40	3	F	50
17	8	M	77

Arbre de décision: **Problème Overfitting**

Par défaut, le modèle d'arbre de décision peut croitre jusqu'à sa pleine profondeur.
Ce qui produit le problème d'overfitting,





Solution d'Overfitting : **Pruning (Elagage)**

Le **pruning** (élagage) fait référence à une technique pour supprimer les parties de l'arbre de décision afin d'empêcher sa croissance jusqu'à sa pleine profondeur.

En ajustant les hyperparamètres du modèle d'arbre de décision, on peut élaguer les arbres et les empêcher du surapprentissage.

Il existe deux types de Pruning:

1. **Pre-Pruning**
2. **Post-Pruning**

Solution d'Overfitting : **Pre-Pruning**

La technique de pré-élagage fait référence à l'arrêt précoce (Early Stopping) de la croissance de l'arbre de décision.
La technique de pré-élagage consiste à régler les hyperparamètres du modèle d'arbre de décision avant le pipeline d'entraînement.

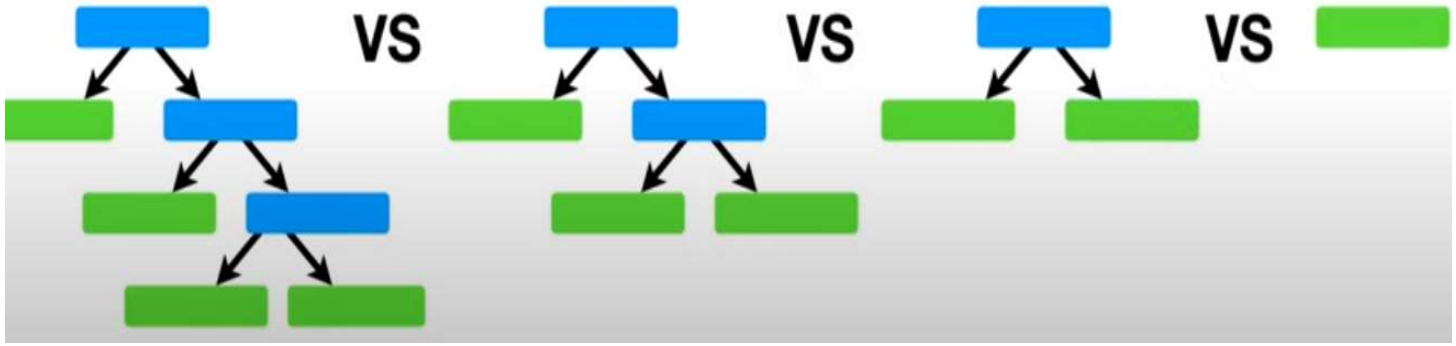
Les hyperparamètres incluant ***max_depth***, ***min_samples*_leaf**, **min_samples_split** peuvent Être réglé pour arrêter tôt la croissance de l'arbre et empêcher le modèle de surajustement

Solution d'Over-fitting : **Post-Pruning**

1. Créer l'arbre de décision jusqu'à sa pleine profondeur,

2. Répéter
   Supprimer une branche de l'arbre (Commencer par les branches en profondeur)
   Remplacer la branche supprimée par un nœud feuille
   Calculer le ***CCP*** (**Cost Complexity Pruning**) de l'arbre résultant
   jusqu'à remonter à la racine
3. Sélectionner l'arbre qui minimise le **CCP**

CCP=RSS+ αT, avec α un hyper paramètre, **T** est le nombre de feuilles dans l'arbre



Arbres de décision : **Autres Algorithmes**

- ID3 (*Iterative Dichotomiser 3*): Arbre de classification de même principe que le CART mais avec un critère de division différent qui est ***Information_Gain***
- C4.5, C5 (successeurs d'ID3)

- ***Information_Gain= mesure le gain en entropie avant et après la division des données,***
- ***On sélectionne comme nœud l'attribut qui maximise le Information_Gain***

$$Info(D) = -\sum_{i=1}^{m} p_i \log(p_i) \text{ où } p_i \text{ représente la probabilité qu'un tuple de } A \text{ appartient à } D$$

$$Info_A(D) = \sum_{j=1}^{V} \frac{|D_j|}{|D|} \times \text{info}(D_j)$$

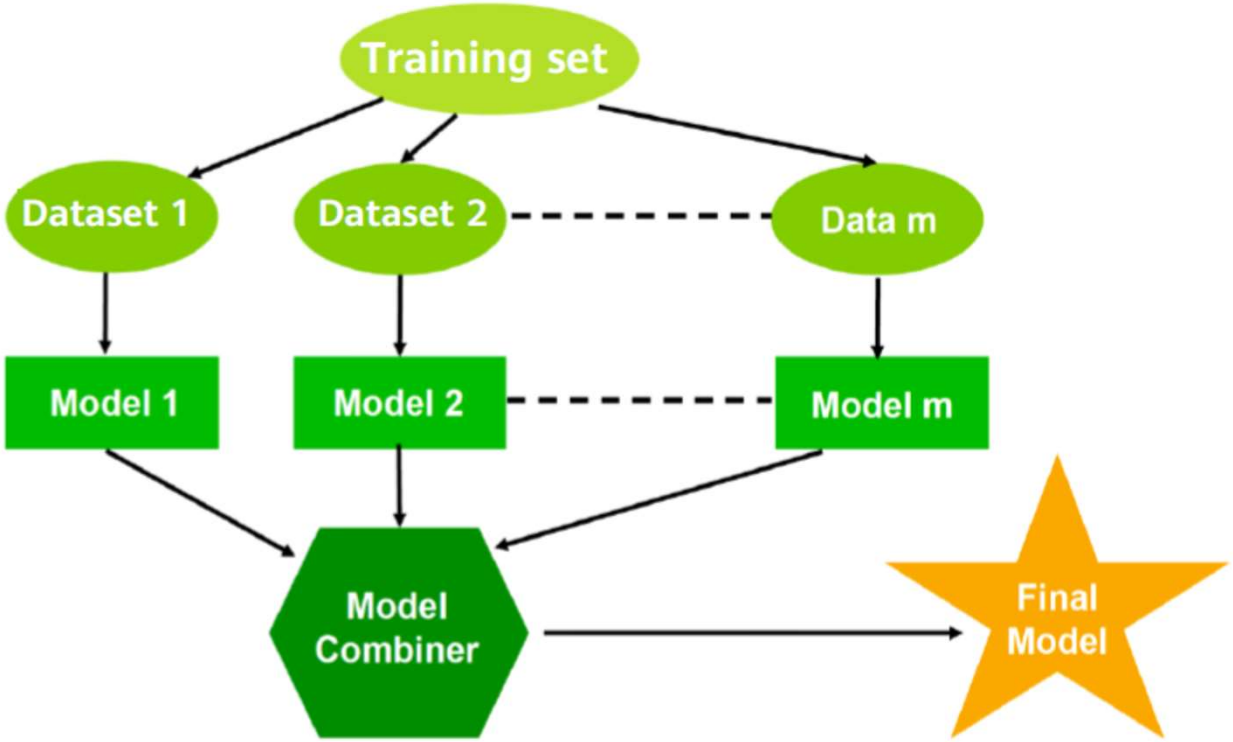
Gain = Info(D)- $Info_A(D)$

Info(D) est la Moyenne d' information demandée pour identifier les classes d'un tuple en D.

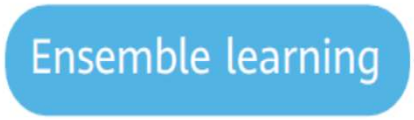
|Dj|/|D| agit comme le poids de la jth partition

Arbres de décision : **Méthodes Ensemblistes**

Définition: L'apprentissage ensembliste est un paradigme d'apprentissage automatique qui permet d'entrainer plusieurs modèles pour résoudre le même problème et de les combiner pour Atteindre un modèle meilleur .



Arbres de décision : **Méthodes Ensemblistes**



Bagging (Bootstrap aggregating) (Ex: Random forests)

- Construit indépendamment plusieurs modèles basiques, puis fait la **moyenne de leurs prédictions**.
- En moyenne, un modèle composite est généralement meilleur qu'un modèle unique en raison de la contribution à une **plus petite variance**.
- la génération de ces modèles peut être naturellement **parallélisée**

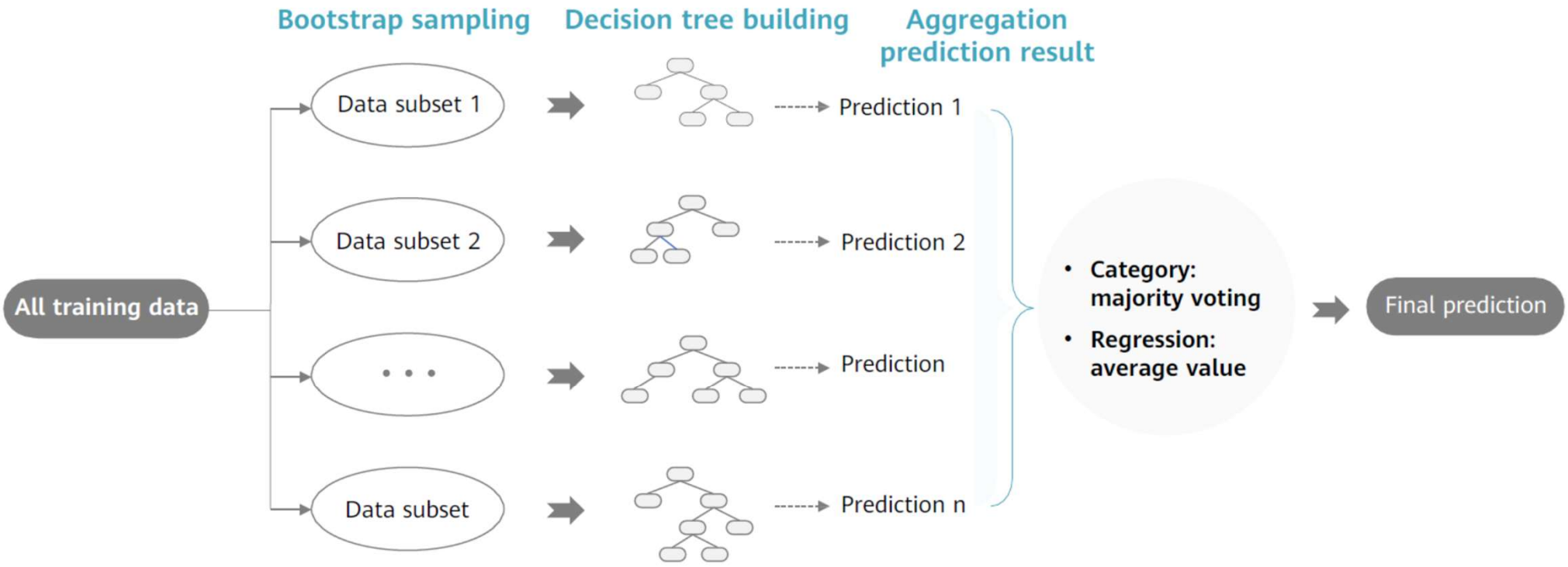
Boosting (AdaBoost, XGBoost, GBDT)

- Construit les modèles basiques en séquence pour réduire progressivement le biais.
- Le modèle composite peut parfaitement s'ajuster aux données, ce qui peut également entraîner un overfitting.

Bagging : **Random Forest**

Random forest=Bagging+CART decision tree

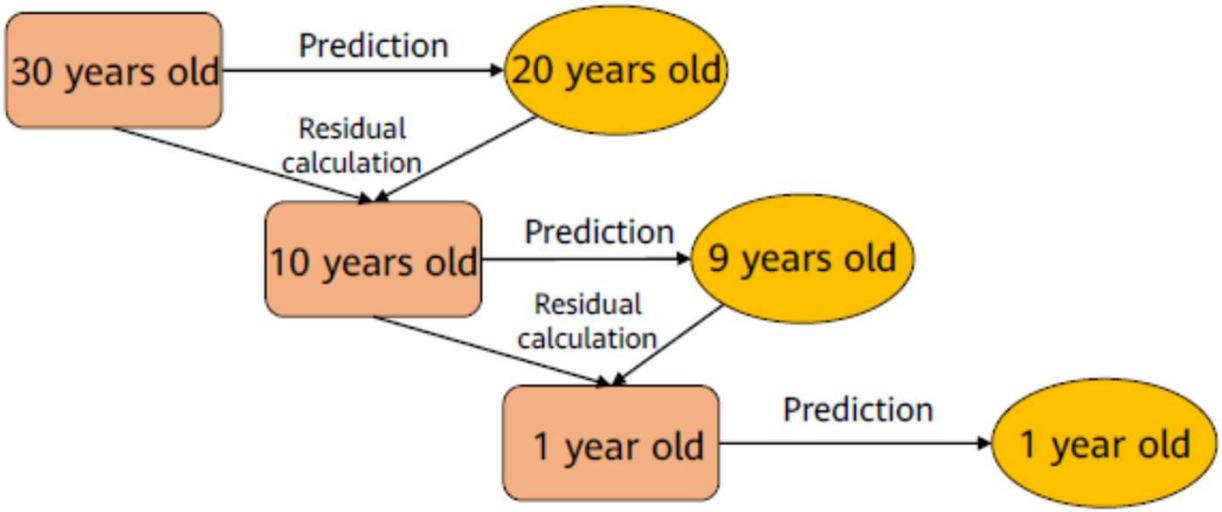
- créent plusieurs arbres de décision et les combinent pour rendre les prévisions plus précises et plus stables.
- Random forest peut être utilisé à la fois pour la régression et la classification



Boosting : **GBDT**

In essence, the residual of the error function to the predicted value is fit by the next basic learner. (The residual is the error between the predicted value and the actual value.)

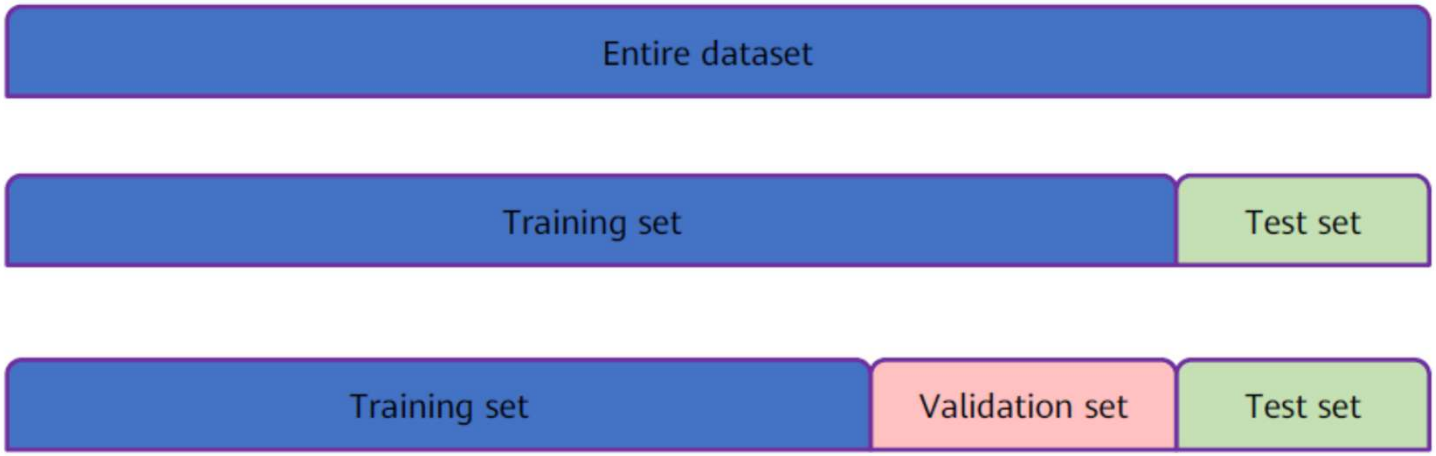
- During model training, GBDT requires that the sample loss for model prediction be as small as possible.



Cross Validation(1)

- **Cross validation**: It is a statistical analysis method used to validate the performance of a classifier. The basic idea is to divide the original dataset into two parts: training set and validation set. Train the classifier using the training set and test the model using the validation set to check the classifier performance.
- **k-fold cross validation ($K-CV$):**
  - Divide the raw data into k groups (generally, evenly divided).
  - Use each subset as a validation set, and use the other $k-1$ subsets as the training set. A total of k models can be obtained.
  - Use the mean classification accuracy of the final validation sets of k models as the performance indicator of the $K-CV$ classifier.

Cross Validation(2)



- Note: The K value in K-fold cross validation is also a hyperparameter.

Implémenter l'arbre CART de classification en utilisant la bibliothèque Scikit-Learn

https://www.datacamp.com/community/tutorials/decision-tree-classification-python