Tree-Based Model and ensemble method

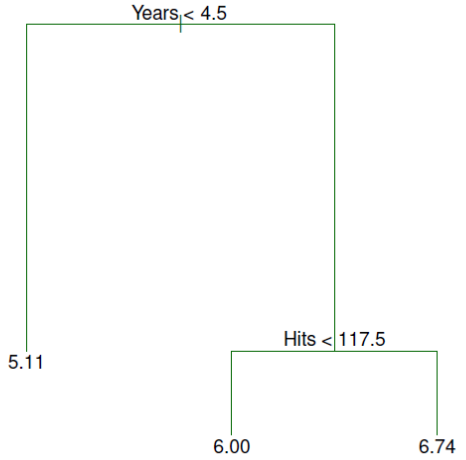
Simple trees, Bagging, Random Forest

Introduction

- Tree-based Model : construction of one tree (very interpretable but weak interpretation accuracy)

- Ensemble Method : Combine several trees to improve the accuracy
    - Bagging/Random Forest : contruct several indepedant trees then mean (reduce variance)
    - Boosting : construct sequentially (correct the error of the previous one)

Tree based model



- Salary in function of experiences and performance
- Over simplification

- Build of tree based model

  Divide the **predictors** space $(X_1, X_1, \dots, X_p)$ into J distinct and non overlapping, **regions** $(R_1, R_2, \dots, R_j)$

  For each new obs, the prediction is the mean of y_i in the region it falls into

  Example :

  First split on yearsExperience < 5

  $R_1 = \{X_1 < 5\}$

  $R_2 = \{X_1 \geq 5\}$

  Then split R_2 on perfScore < 80

  $R_{2a} = \{X_1 \geq 5, X_2 < 80\}$

  $R_{2b} = \{X_1 \geq 5, X_2 > 80\}$

Final regions : R_1 , R_{2a}, R_{2b}

- If a new X falls in region R_j: $\hat{y}(X) =$ mean of all y_i in R_j for regression / majority class in classification

- Predictor region spaces

  High dimensional rectangles / boxes

- Goal :

  Find boxes $R_1, R_1, \dots, R_j$ that minimize the RSS given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y_{R_j}})^2$$

- $\widehat{y_{R_j}}$: mean response for the training observations within the jth box

Construction de l’arbre (Exemple concret)

Âge	y
20	0
22	0
25	0
30	1
35	1
40	1

🔎 Rappel : Indice de Gini

En **classification**, l’arbre cherche à créer des groupes homogènes.

- Si un nœud contient plusieurs classes, son impureté est mesurée par :

$$Gini = 1 - \sum_{k=1}^{K} p_k^2$$

où p_k = proportion d’observations de la classe k dans le nœud.

- Exemple :
  - Nœud avec 50% de 0 et 50% de 1 → $Gini = 1 - (0.5^2 + 0.5^2) = 0.5$ → très impur.
  - Nœud avec 100% de 0 → $Gini = 1 - (1^2) = 0$ → parfait (homogène).

🔎 Étape 1 : tester des splits possibles

On trie les données par âge et on teste les **coupures possibles** (entre les valeurs distinctes).

Candidats pour couper :

- $age < 21$
- $age < 23.5$
- $age < 27.5$
- $age < 32.5$
- $age < 37.5$

🔎 Étape 3 : calcul pour un split

Exemple : split $age < 27.5$.

- Groupe gauche : {20, 22, 25 → y=0,0,0} → homogène (tous 0).
- Groupe droit : {30, 35, 40 → y=1,1,1} → homogène (tous 1).

Impureté = 0 → parfait ✅.

Donc ce split est **excellent**.

Plus gini proche de 1 mieux c’est

🔎 Étape 2 : mesurer la qualité d’un split

En classification, on veut des **groupes homogènes** (que des 0, ou que des 1 si possible).

On peut mesurer ça avec :

- **Impureté de Gini** : $Gini = 1 - \sum p_k^2$, où p_k est la proportion de classe k.
- Ou **entropie**.
- En régression : on prend la **variance intra-groupe** (ou MSE).

🔎 Étape 4 : répéter récursivement

On continue le même processus **dans chaque groupe** : tester les splits, choisir celui qui réduit le plus l’erreur, etc.

On arrête quand :

- Les nœuds sont homogènes.
- Ou on atteint une profondeur max / nombre minimal d’observations.

Nœuds homogènes = tous la même classe ou proche de y en regression

⚡ Exemple très simplifié

- Supposons 6 individus avec `Age` et `Revenu`.
- Pour `Age`, le meilleur split est à 27.5 (réduit Gini de 0.3).
- Pour `Revenu`, le meilleur split est à 40k (réduit Gini de 0.25).

👉 Comme 0.3 > 0.25, l'arbre choisit de splitter **par Age < 27.5** à la racine.

🌳 Ensuite…

- Dans le **sous-groupe gauche** (Age < 27.5), l'arbre recommence :
  - Il regarde à nouveau toutes les features (Age, Revenu).
  - Il teste tous les splits possibles *dans ce sous-groupe*.
  - Il choisit le meilleur.
- Idem dans le **sous-groupe droit** (Age ≥ 27.5).

⚠️ Donc tu n'as pas "Age d'abord puis Revenu ensuite par défaut".

C'est **dynamique** : à chaque nœud, l'algorithme choisit la feature qui donne le meilleur gain.

🌳 Pourquoi on peut re-regarder Age après avoir déjà split dessus ?

Parce que le **premier split sur Age** ne coupe pas toutes les valeurs possibles d'Age, il ne fait qu'une **frontière**.

Exemple :

- Racine : `Age < 27.5 ?`
  - Sous-nœud gauche : individus jeunes (ex: 20, 22, 25)
  - Sous-nœud droit : individus plus âgés (ex: 30, 35, 40, …)

👉 Dans le sous-nœud droit, tu as encore des individus d'âges différents (30 vs 40 par exemple).

Donc l'algo peut très bien re-splitter sur Age, par exemple `Age < 37.5 ?`, pour mieux séparer les 30 des 40.

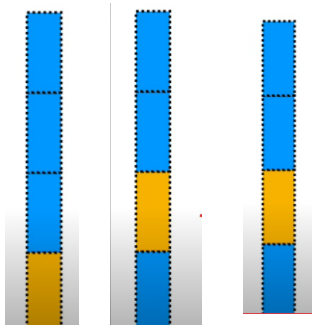
Technical details

- Recursive Binary Splitting : at each step we choose the best variable and best threshold that reduce the RSS
- Greedy / Top down : local better choice not global
- Regression Criteria : RSS / Clasification Criteria Gini,entropia
- Often the tree is too complex : Pruning / cross validation

Pruning / Cross Validation

- To avoid overfitting (too big depth): Pruning (cut the tree too find a better bias/variance balance)
- How : remove some of the leaves aand replace it by a leave that is an average of larger number observation
- How to Prune Regression Trees, Clearly Explained!!!
- Low depth tree : ++bias – variance / Low depth tree : --bias +variance
- Pruning generate a sequence of candidate subtrees

Cross validation



- Method where we **split** the data into several parts, **train** the model on some parts and **test** it on the others

- Then use CV to compare performance of each sub trees (find the optimal depth/ size after pruning :hyperparameters)

- We keep the one that minimize the error

- *On construit d'abord un arbre trop grand, puis on le réduit (pruning). La cross-validation permet de choisir la taille optimale de l'arbre. (CV permet de retrouver les valeurs optimales des hyperparamètres de manière plus efficaces)*

Method to find hyperparameters

- ***Grid search*** *: you define a grid* $\lambda \in \{0.01, 0.1, 1, 10\}$, deep of the tree $\in \{2, 3, 4, 5\}$ *and test all combinations choose the one that minimise the error*

- ***Grid Search CV*** *: you define a grid of possible hyperparameters, for each value you do a CV ; you choose the hpp that maximises score. More efficient/ robust*

- ***Grid search CV*** *more efficient than Grid search but less than Random search CV / Bayesian optimization (Optuna,Hyperopt …)*

- ***Random search CV*** *: you fix a fix number of trials (ex : 50 random draw in the spae of hpp=) more speed*
- ***Bayesian optimization*** *: Not random, the algorithm learns a it goes along which areas of the hyperparameter space seem promishing*

Ensemble method (Bagging Random Forest)

- Even after pruning a simple tree has high variance

- To improve stability and accuracy we combine many simple trees together (weak learners) :

  Different methods exist : bagging, Random Forests, Boosting (Elyes)

Bagging

- Bagging : procedure for reducing the variance of statistical learning method
- How : by averaging : it reduces variance(if each model is noisy combining many of them smooths out the fluctuations)

  Mean of n independant observation of variance σ^2 is $\frac{\sigma^2}{n}$

- If we could make the same experience with a large number of datasets different from the true population : we could train a tree on each dataset :

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

In practice we do not have accees to multiple training test -> this is why we will artificially simulate train set (boostrap)

(tirage with remise)

La méthode artificielle (le **bootstrap**) c'est tout simple :

1. Tu as ton dataset de taille n (par ex. 100 observations).
2. Tu crées un nouvel échantillon aussi de taille n, en **tirant au hasard avec remise** dans le dataset :
   - Chaque observation a une chance $1/n$ d'être tirée à chaque fois.
   - Comme c'est **avec remise**, certaines observations apparaissent plusieurs fois, d'autres pas du tout.
3. Tu répètes ça B fois (par ex. 100 ou 500 fois).
   → Tu obtiens B jeux de données artificiels.
4. Tu entraînes un modèle (ex. un arbre) sur chaque jeu.
5. Tu fais la **moyenne (ou vote)** des prédictions.

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

- $\hat{f}^{*b}$: bth bootstrapped training set

Random Forest

- Pb of bagging : if a variable is dominant all the subset will choose the same variable to start : trees are highly correlated
- RF : Add random ; each time a random sample of m predictors is chosen
- We often choose to take $\mathrm{m} = \sqrt{p}$ for classification and m=$\frac{p}{3}$ for regression
- So the probability that the strong predictor is not choosen is $\frac{p-m}{p}$ (proba of bing chosen is $\frac{m}{p}$)
- At the end we also proceed to a mean

Exemple d'arbre trop sensible aux données d'entrées

Understand bagging and RF through data

Observation (individu)	Age (feature 1)	Revenu (feature 2)	Ville (feature 3)	y (cible)
1	25 25	30k	Paris	0
2	40	70k	Lyon	1
3	32	50k	Paris	0

Bagging (random only on individuals): we drag individuals randomly with replacement. Example : Take I1 one time / I2 3 times (bootstrapped dataset) / all the features are always considered to build the tree and find better split

Random Forest : Same but at each nodes we only see Age/ Revenu or Ville/ Revenu information