

PAPER TITLE

1. GUIDE D'ENTRAÎNEMENT DES FORÊTS ALÉATOIRES

Cette section rassemble et synthétise des recommandations sur l'entraînement des forêts aléatoires disponibles dans la littérature, en particulier dans Probst et al. (2019).

1.1. Mode de construction d'une forêt aléatoire.

Le processus pour construire une Random Forest se résume comme suit:

- Sélectionnez le nombre d'arbres à construire (`n_trees`).
- Pour chaque arbre, effectuez les étapes suivantes :
 - Générer un échantillon bootstrap à partir du jeu de données.
 - Construire un arbre de décision à partir de cet échantillon:
 - À chaque nœud de l'arbre, sélectionner un sous-ensemble aléatoire de caractéristiques (`mtry`).
 - Trouver la meilleure division parmi ce sous-ensemble et créer des nœuds enfants.
 - Arrêter la croissance de l'arbre selon des critères de fin spécifiques (comme une taille minimale de nœud), mais sans élaguer l'arbre.
- Agréger les arbres pour effectuer les prédictions finales :
 - Régression : la prédiction finale est la moyenne des prédictions de tous les arbres.
 - Classification : chaque arbre vote pour une classe, et la classe majoritaire est retenue.

1.2. Quelle implémentation utiliser?.

Il existe de multiples implémentations des forêts aléatoires. Le présent document présente et recommande l'usage de deux implémentations de référence: le *package* R `ranger` et le *package* Python `scikit-learn`. Il est à noter qu'il est possible d'entraîner des forêts aléatoires avec les algorithmes `XGBoost` et `LightGBM`, mais il s'agit d'un usage avancé qui n'est recommandé en première approche. Cette approche est présentée dans la partie **REFERENCE A LA PARTIE USAGE AVANCE**.

1.3. Quels sont les hyperparamètres des forêts aléatoires?

Cette section décrit en détail les principaux hyperparamètres des forêts aléatoires listés dans le tableau Table 1. Les noms des hyperparamètres utilisés sont ceux figurant dans le *package* R `ranger`, et dans le *package* Python `scikit-learn`. Il arrive qu'ils portent un nom différent dans d'autres implémentations des *random forests*, mais il est généralement facile de s'y retrouver en lisant attentivement la documentation.



Hyperparamètre		Description
 ranger	 scikit-learn	
mtry	max_features	Le nombre de variables candidates à chaque noeud
replacement		L'échantillonnage des données se fait-il avec ou sans remise?
sample.fraction	max_samples	Le taux d'échantillonnage des données
min.node.size	min_samples_leaf	Nombre minimal d'observations nécessaire pour qu'un noeud puisse être partagé
num.trees	n_estimators	Le nombre d'arbres
splitrule	criterion	Le critère de choix de la règle de division des noeuds intermédiaires
min.bucket	min_samples_split	Nombre minimal d'observations dans les noeuds terminaux
max.depth	max_depth	Profondeur maximale des arbres

Table 1: Les principaux hyperparamètres des forêts aléatoires

- Le **nombre de variables candidates à chaque noeud** contrôle l'échantillonnage des variables lors de l'entraînement. La valeur par défaut est fréquemment \sqrt{p} pour la classification et $p/3$ pour la régression. C'est l'hyperparamètre qui a le plus fort effet sur la performance de la forêt aléatoire. Une valeur plus basse aboutit à des arbres plus différents, donc moins corrélés (car ils reposent sur des variables différentes), mais ces arbres peuvent être moins performants car ils reposent parfois sur des variables peu pertinentes. Inversement, une valeur plus élevée du nombre de variables candidates aboutit à des arbres plus performants, mais plus corrélés. C'est en particulier le cas si seulement certaines variables

sont très prédictives, car ce sont ces variables qui apparaîtront dans la plupart des arbres.

- Le **taux d'échantillonnage** et le **mode de tirage** contrôlent le plan d'échantillonnage des données d'entraînement. Les valeurs par défaut varient d'une implémentation à l'autre; dans le cas de `ranger`, le taux d'échantillonnage est de 63,2% sans remise, et de 100% avec remise. L'implémentation `scikit-learn` ne propose pas le tirage sans remise. Ces hyperparamètres ont des effets sur la performance similaires à ceux du nombre de variables candidates, mais dans une moindre ampleur. Un taux d'échantillonnage plus faible aboutit à des arbres plus différents et donc moins corrélés (car ils sont entraînés sur des échantillons très différents), mais ces arbres peuvent être peu performants car ils sont entraînés sur des échantillons de petite taille. Inversement, un taux d'échantillonnage élevé aboutit à des arbres plus performants mais plus corrélés. Les effets de l'échantillonnage avec ou sans remise sur la performance de la forêt aléatoire sont moins clairs et ne font pas consensus. Les travaux les plus récents suggèrent toutefois qu'il est préférable d'échantillonner sans remise (Probst, Wright, & Boulesteix (2019)).
- Le **nombre minimal d'observations dans les noeuds terminaux** contrôle la taille des noeuds terminaux. La valeur par défaut est faible dans la plupart des implémentations (entre 1 et 5). Il n'y a pas vraiment de consensus sur l'effet de cet hyperparamètre sur les performances. En revanche, il est certain que le temps d'entraînement décroît fortement avec cet hyperparamètre: une valeur faible implique des arbres très profonds, avec un grand nombre de noeuds. Il peut donc être utile de fixer ce nombre à une valeur plus élevée pour accélérer l'entraînement, en particulier si les données sont volumineuses.
- Le **nombre d'arbres** par défaut varie selon les implémentations (500 dans `ranger`, 100 dans `scikit-learn`). Il s'agit d'un hyperparamètre particulier car il n'est associé à aucun arbitrage en matière de performance: la performance de la forêt aléatoire croît avec le nombre d'arbres, puis se stabilise à un niveau approximativement constant. Le nombre optimal d'arbres est donc intuitivement celui à partir duquel la performance de la forêt ne croît plus (ce point est détaillé plus bas). Il est important de noter que ce nombre optimal dépend des autres hyperparamètres. Par exemple, un taux d'échantillonnage faible et un nombre faible de variables candidates à chaque noeud aboutissent à des arbres peu cor-

réels, mais peu performants, ce qui requiert probablement un plus grand nombre d'arbres.

- **Le critère de choix de la règle de division des noeuds intermédiaires:** la plupart des implémentations des forêts aléatoires retiennent par défaut l'impureté de Gini pour la classification et la variance pour la régression, même si d'autres critères de choix ont été proposés dans la littérature. A ce stade, aucun critère de choix ne paraît systématiquement supérieur aux autres en matière de performance. Le lecteur intéressé pourra se référer à la discussion détaillée dans Probst, Wright, & Boulesteix (2019).

1.4. Comment entraîner une forêt aléatoire?.

Comme indiqué dans la partie **REFERENCE A AJOUTER**, la performance prédictive d'une forêt aléatoire varie en fonction de deux critères essentiels: elle croît avec le pouvoir prédictif moyen des arbres, et décroît avec la corrélation entre les arbres. La recherche d'arbres très prédictifs pouvant aboutir à augmenter la corrélation entre eux, l'objectif de l'entraînement d'une forêt aléatoire revient à trouver le meilleur arbitrage possible entre pouvoir prédictif et corrélation.

REFERENCES

Probst, P., Wright, M. N., & Boulesteix, A.-L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3), e1301.