PAPER TITLE

1. Que sont les méthodes ensemblistes?

1.1. **L'union fait la force.**

Plutôt que de chercher à construire d'emblée un unique modèle très complexe, les approches ensemblistes vise à obtenir un modèle très performant en combinant un grand nombre de modèles simples.

Il existe quatre grandes approches ensemblistes:

- le *bagging*;
- la *random forest*;
- le *stacking*;
- le *boosting*.

Le présent document se concentre sur deux approches: la *random forest* et le *boosting*.

Les méthodes ensemblistes désignent un ensemble d'algorithmes d'apprentissage supervisé (notamment les forêts aléatoires et le *boosting*) développés depuis le début des années 2000. Ces méthodes consistent à entraîner plusieurs modèles de base, puis à combiner les résultats obtenus afin de produire une prédiction consolidée. Les modèles de base, dits "apprenants faibles" ("weak learners"), sont généralement peu complexes. Le choix de ces modèles et la manière dont leurs prédictions sont combinées sont des facteurs clés pour la performance de ces approches.

Les méthodes ensemblistes peuvent être divisées en deux grandes familles selon qu'elles s'appuient sur des modèles entrainés en parallèle ou de manière imbriquée ou séquentielle. Lorsque les modèles sont *entrainés en parallèle*, chaque modèle de base est entraîné en utilisant soit un échantillon aléatoire des données d'entraînement, soit un sous-ensemble des variables disponibles, et le plus souvent une combinaison des deux, auquel cas on parle de forêt aléatoire. Les implémentations les plus courantes des forêts aléatoires sont les *packages* `ranger` en `R` et `scikit-learn` en Python. Lorsque les modèles de base sont *entrainés de manière séquentielle*, chaque modèle de base vise à minimiser l'erreur de prédiction de l'ensemble des modèles de base précédents. Les implémentations les plus courantes du *boosting* sont actuellement XGBoost, CatBoost et LightGBM.

2. Pourquoi utiliser des méthodes ensemblistes?

Les méthodes ensemblistes sont particulièrement bien adaptées à de nombreux cas d'usage de la statistique publique, pour deux raisons. D'une part, elles sont conçues pour s'appliquer à des données tabulaires (enregistrements en lignes, variables en colonnes), structure de données omniprésente dans la statistique publique. D'autre part, elles peuvent être mobilisées dans toutes les situations où on utilise une régression linéaire ou une régression logistisque (imputation, repondération…).

Les méthodes ensemblistes présentent trois avantages par rapport aux méthodes économétriques traditionnelles (régression linéaire et régression logistique):

- Elles ont une **puissance prédictive supérieure**: alors que les méthodes traditionnelles supposent fréquemment l'existence d'une relation linéaire ou log-linéaire entre y et $\mathbf{X}$, les méthodes ensemblistes ne font quasiment aucune hypothèse sur la relation entre y et $\mathbf{X}$, et se contentent d'approximer le mieux possible cette relation à partir des données disponibles. En particulier, les modèles ensemblistes peuvent facilement modéliser des **non-linéarités** de la relation entre y et $\mathbf{X}$ et des **interactions** entre variables explicatives *sans avoir à les spécifier explicitement* au préalable, alors que les méthodes traditionnelles supposent fréquemment l'existence d'une relation linéaire ou log-linéaire entre y et $\mathbf{X}$.
- Elles nécessitent **moins de préparation des données**: elles ne requièrent pas de normalisation des variables explicatives et peuvent s'accommoder des valeurs manquantes (selon des techniques variables selon les algorithmes).
- Elles sont généralement **moins sensibles aux valeurs extrêmes et à l'hétéroscédasticité** des variables explicatives que les approches traditionnelles.

Elles présentent par ailleurs deux inconvénients rapport aux méthodes économétriques traditionnelles. Premièrement, bien qu'il existe désormais de multiples approches permettent d'interpétrer partiellement les modèles ensemblistes, leur interprétabilité reste globalement moindre que celle d'une régression linéaire ou logistique. Deuxièmement, les modèles ensemblistes sont plus complexes que les approches traditionnelles, et leurs hyperparamètres doivent faire l'objet d'une optimisation, par exemple au travers d'une validation croisée. Ce processus d'optimisation est généralement plus complexe et plus long que l'estimation d'une régression linéaire ou logistique. En revanche, utiliser des

méthodes ensemblistes ne requiert pas de connaissances avancées en informatique ou de puissance de calcul importante.

> **i** Et par rapport au *deep learning*?
>
> Si les approches de *deep learning* sont sans conteste très performantes pour le traitement du langage naturel et le traitement d'image, leur supériorité n'est pas établie pour les applications reposant sur des données tabulaires. Les comparaisons disponibles dans la littérature concluent en effet que les méthodes ensemblistes à base d'arbres sont soit plus performantes que les approches de *deep learning* (Grinsztajn et al. (2022), Shwartz-Ziv & Armon (2022)), soit font jeu égal avec elles (McElfresh et al. (2024)). Ces études ont identifié trois avantages des méthodes ensemblistes: elles sont peu sensibles aux variables explicatives non pertinentes, robustes aux valeurs extrêmes des variables explicatives, et capables d'approximer des fonctions très irrégulières. De plus, dans la pratique les méthodes ensemblistes sont souvent plus rapides à entraîner et moins gourmandes en ressources informatiques, et l'optimisation des hyperparamètres s'avère souvent moins complexe (Shwartz-Ziv & Armon (2022)).

3. Comment fonctionnent les méthodes ensemblistes?

Quatre temps:

- les arbres de décision et de régression (CART);
- les forêts aléatoires;
- le boosting.

3.1. **Le point de départ: les arbres de décision et de régression.**

Présenter *decision tree* et *regression tree*. Reprendre des éléments du chapitre 9 de https://bradleyboehmke.github.io/HOML/

Principes d'un arbre:

- fonction constante par morceaux;
- partition de l'espace;
- interactions entre variables.

Illustration, et représentation graphique (sous forme d'arbre et de graphique).

3.2. **Critères de performance et sélection d'un modèle.**

La performance d'un modèle augmente généralement avec sa complexité, jusqu'à atteindre un maximum, puis diminue. L'objectif est d'obtenir un modèle qui minimise à la fois le sous-apprentissage (biais) et le sur-apprentissage (variance). C'est ce qu'on appelle le compromis biais/variance. Cette section présente très brièvement les critères utilisés pour évaluer et comparer les performances des modèles.

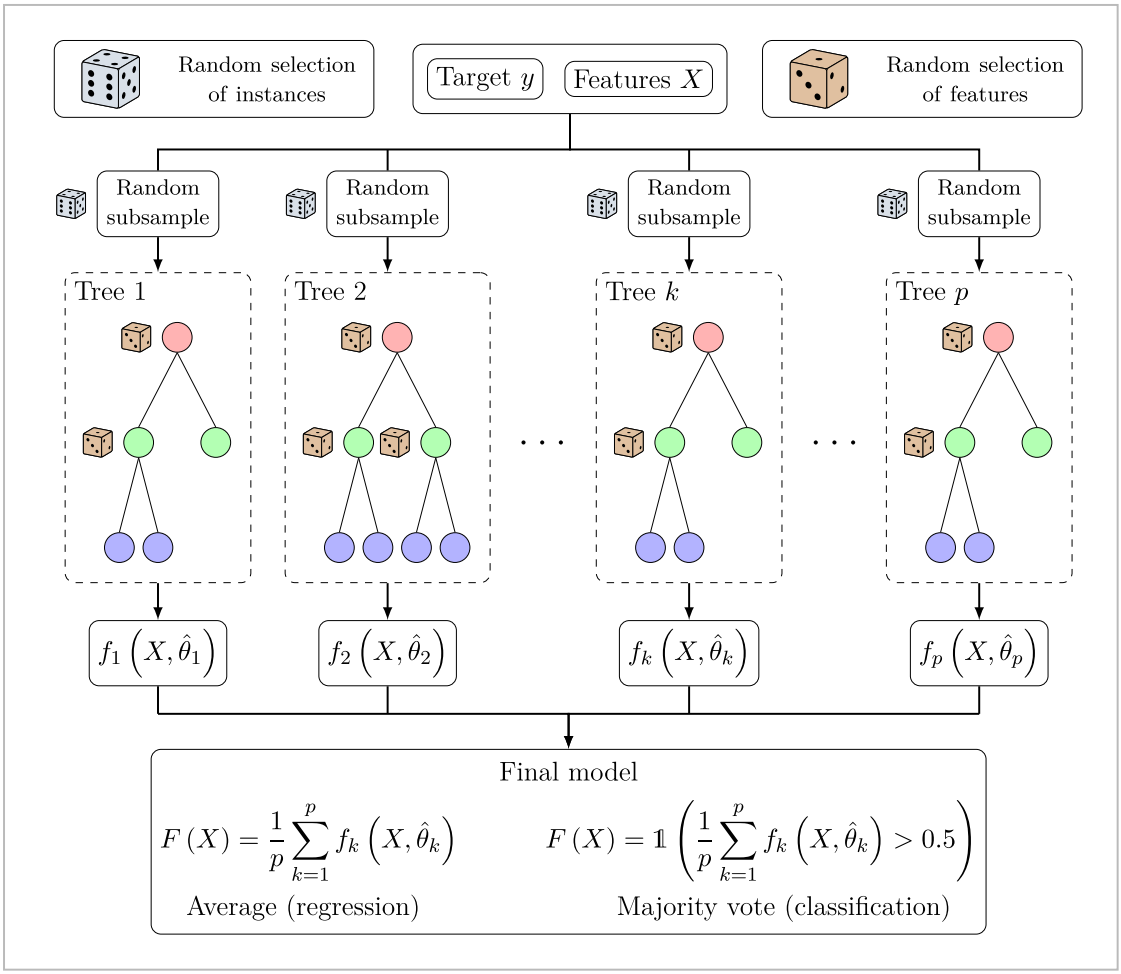

Figure 1: Représentation schématique d'un algorithme de forêt aléatoire

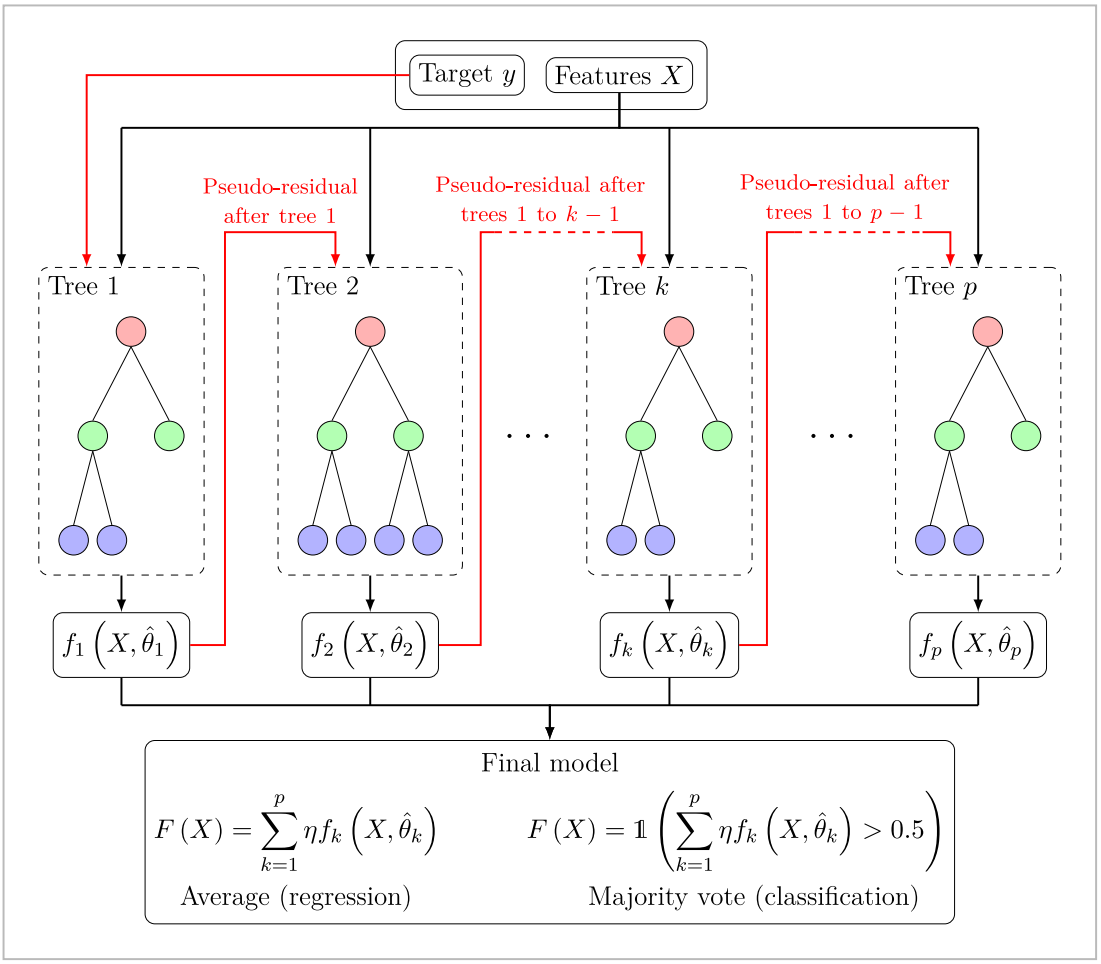

Figure 2: Représentation schématique d'un algorithme de *boosting*

4. Le *bagging*, les *random forests* et le *boosting*

Il existe plusieurs types de méthodes ensemblistes, toutes ayant en commun la combinaison de modèles élémentaires. Le présent document présente les 3 principales méthodes : le Bagging, la Random Forests et le Boosting.

4.1. Le *bagging* (Bootstrap Aggregating).

Présenter le *bagging* en reprenant des éléments du chapitre 10 de https://bradleyboehmke.github.io/HOML. Mettre une description de l'algorithme en pseudo-code?

- Présentation avec la figure en SVG;
- Illustration avec un cas d'usage de classification en deux dimensions.

Le bagging, ou Bootstrap Aggregating, est une méthode ensembliste qui comporte trois étapes principales :

- Création de sous-échantillons : À partir du jeu de données initial, plusieurs sous-échantillons sont générés par échantillonnage aléatoire avec remise (bootstrapping). Chaque sous-échantillon a la même taille que le jeu de données original, mais peut contenir des observations répétées, tandis que d'autres peuvent être omises. Cette technique permet de diversifier les données d'entraînement en créant des échantillons variés, ce qui aide à réduire la variance et à améliorer la robustesse du modèle.
- Entraînement parallèle : Un modèle distinct est entraîné sur chaque sous-échantillon de manière indépendante. Cette technique permet un gain d'efficacité et un meilleur contrôle du surapprentissage (overfitting).
- Agrégation des prédictions : Les prédictions des modèles sont combinées pour produire le résultat final. En classification, la prédiction finale est souvent déterminée par un vote majoritaire, tandis qu'en régression, elle correspond généralement à la moyenne des prédictions. En combinant les prédictions de plusieurs modèles, le bagging renforce la stabilité et la performance globale de l'algorithme, notamment en réduisant la variance des prédictions.

Le bagging appliqué aux arbres de décision est la forme la plus courante de cette technique.

Le bagging est particulièrement efficace pour réduire la variance des modèles, ce qui les rend moins vulnérables au surapprentissage. Cette caractéristique est particulièrement utile dans les situations où la robustesse et la capacité de généralisation des modèles sont cruciales. De plus, comme le bagging repose sur des processus indépendants, l'exécution est plus plus rapide dans des environnements distribués.

Cependant, bien que chaque modèle de base soit construit indépendamment sur des sous-échantillons distincts, les variables utilisées pour générer ces modèles ne sont pas forcément indépendantes d'un modèle à l'autre. Dans le cas du bagging appliqué aux arbres de décision, cela conduit souvent à des arbres ayant une structure similaire.

Les forêts aléatoires apportent une amélioration à cette approche en réduisant cette corrélation entre les arbres, ce qui permet d'augmenter la précision de l'ensemble du modèle.

4.2. Les *random forests.*

Expliquer que les *random forests* sont une amélioration du *bagging*, en reprenant des éléments du chapitre 11 de https://bradleyboehmke.github.io/HOML/

- Présentation avec la figure en SVG;
- Difficile d'illustrer avec un exemple (car on ne peut pas vraiment représenter le *feature sampling*);
- Bien insister sur les avantages des RF: 1/ faible nombre d'hyperparamètres; 2/ faible sensibilité aux hyperparamètres; 3/ limite intrinsèque à l'overfitting.

4.3. Le *boosting.*

Reprendre des éléments du chapitre 12 de https://bradleyboehmke.github.io/HOML/ et des éléments de la formation boosting.

Le *boosting* combine l'**approche ensembliste** avec une **modélisation additive par étapes** (*forward stagewise additive modeling*).

- Présentation;
- Avantage du boosting: performances particulièrement élevées.
- Inconvénients: 1/ nombre élevé d'hyperparamètres; 2/ sensibilité des performances aux hyperparamètres; 3/ risque élevé d'overfitting.
- Préciser qu'il est possible d'utiliser du subsampling par lignes et colonnes pour un algoithme de boosting. Ce point est abordé plus en détail dans la partie sur les hyperparamètres.

References

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data?. *Advances in Neural Information Processing Systems*, *35*, 507–520.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., & White, C. (2024). When do neural nets outperform boosted trees on tabular data?. *Advances in Neural Information Processing Systems*, *36*.

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, *81*, 84–90.