17-11-2021

Les méthodes d’ensemble

Présenté par Miyuki, Pierre et Anne-Sophie

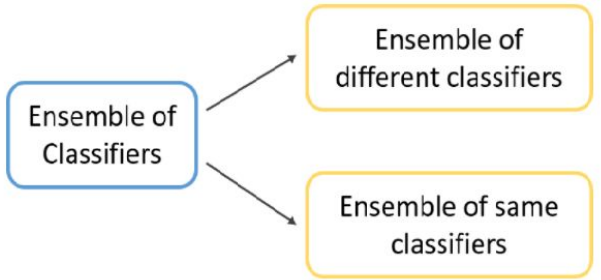
Formation Développeur Data IA - École Microsoft IA by Simplon

Sommaire

- Le principe général des méthodes d'ensemble
- Qu’est-ce que le "Bagging" ?
- Qu’est-ce que le "Pasting" ?
- L'évaluation "Out-Of-Bag"
- La méthode de "Random Subspaces"
- La méthode de "Random Patches"
- RandomForest et concepts présentés précédemment utilisés par ce modèle
- Ensemble des paramètres de la fonction "RandomForestRegressor" de la librairie Scikit-Learn

Principe général des méthodes d'ensemble

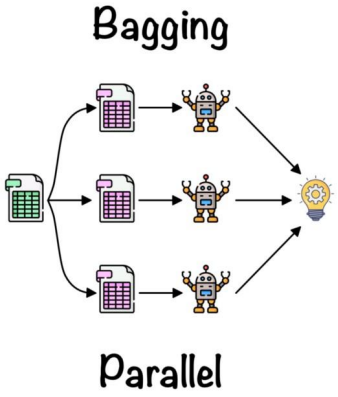
Combiner des modèles avec des performances faibles permet d'obtenir un modèle prédictif plus efficace



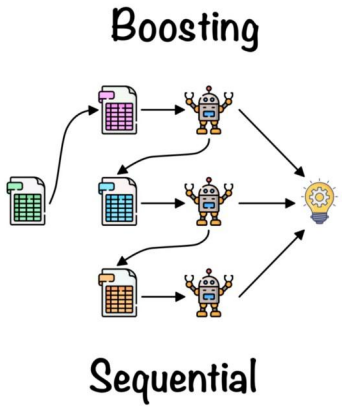
Les méthodes d'ensemble sont un paradigme d'apprentissage automatique qui repose sur la combinaison de plusieurs modèles (souvent appelés "**apprenants faibles**"). L'objectif est d' accroître les performances du modèle et de parvenir à un niveau de précision bien supérieur à celui qui serait réalisé si on utilisait n'importe lequel de ces algorithmes pris séparément.

- ➔ Utiliser un seul et même type d'algorithmes permet de produire un **résultat homogène**
- ➔ Utiliser des algorithmes entièrement différents (RandomForest, SVC, régression logistique) permet d'obtenir un **résultat hétérogène**.

Principe général des méthodes d'ensemble



Avec la **méthode ensembliste séquentielle**, les modèles sont entraînés à la suite en leur permettant d'apprendre des erreurs passées, avant de les combiner à la fin du processus. Ce qui est possible en affectant un poids un peu plus élevé aux observations erronées du premier modèle, pour leur donner plus d'importance dans l'entraînement du suivant, et ainsi de suite pour les suivants.



Avec la **méthode ensembliste parallèle**, les modèles sont entraînés en simultané (en même temps), dans l'idée d'exploiter à la fin les différences d'observations entre ces modèles indépendants, au moment de leur combinaison.

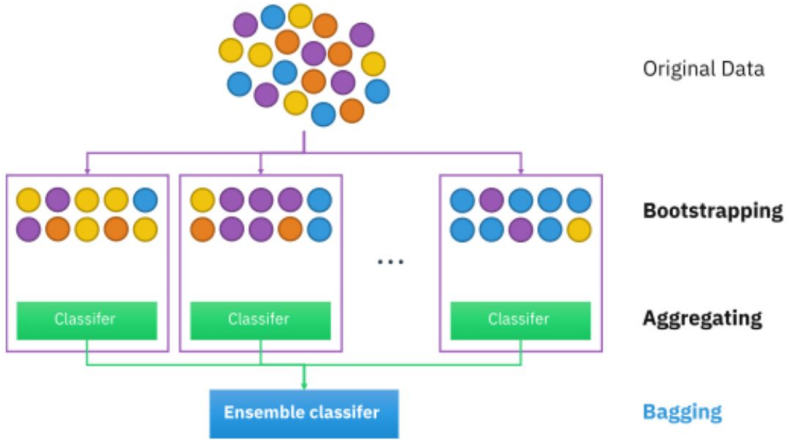
Qu’est-ce que le bagging ?

- *Bagging = Bootstrap + aggregating*
- Méthode d’ensemble introduite par Breiman en 1996
- Utilisée pour réduire la variance au sein d'un ensemble de données bruyant

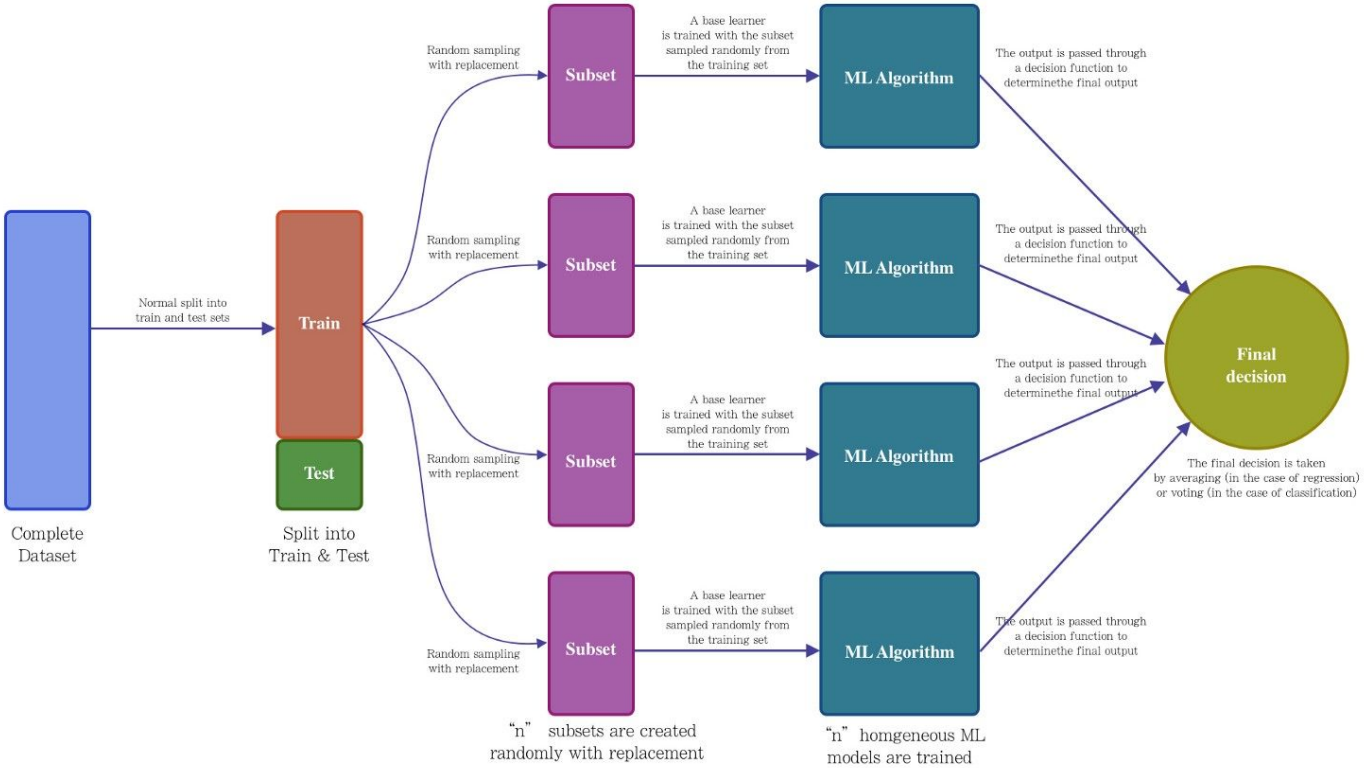
1. **Le bootstraping** : Le principe est de créer de « nouveaux échantillons » par tirage au hasard dans le dataset d’origine, **avec remise** (une même instance peut donc apparaître plusieurs fois dans un sous-échantillon).

2. **L’entraînement parallèle** : Ces échantillons bootstrap sont ensuite entraînés indépendamment et en parallèle les uns avec les autres en utilisant des apprenants faibles ou de base.

3. **L'agrégation** : On obtient ainsi un ensemble de modèles dont il convient de moyenner (lorsqu’il s’agit d’une régression) ou de faire voter (pour une classification) les différentes prédictions.

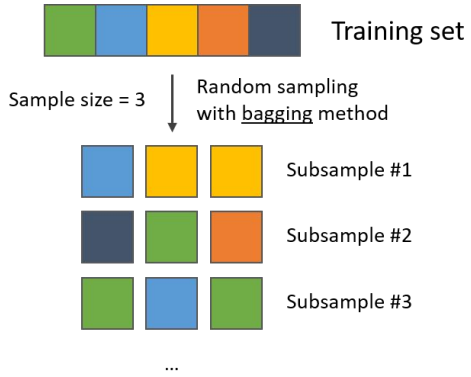


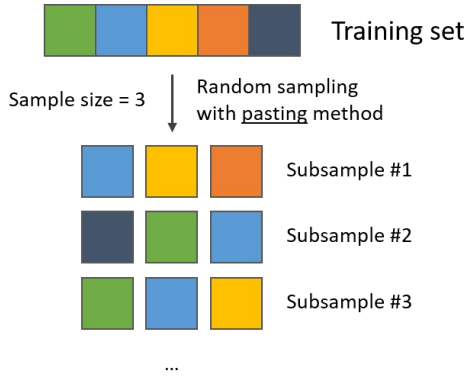
Qu’est-ce que le “bagging” ?



Qu’est-ce que le “pasting” ?

Le “pasting” est une méthode d’ensemble semblable au “bagging” à la différence qu’elle n’effectue **pas de remplacement** lors de la création de ses sous-échantillons. Cela signifie que, dans un échantillon, la même instance ne peut apparaître qu’une seule fois.

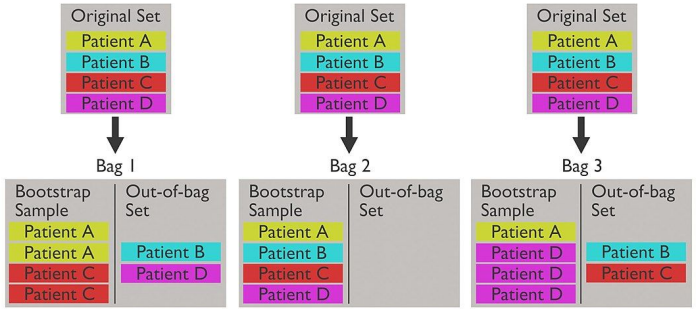




L’évaluation out-of-bag

L' évaluation out-of-bag ou out-of-bag error est une **méthode de mesure de l'erreur des modèles d'apprentissage automatique utilisant le bagging**. Le bagging utilise le sous-échantillonnage avec remplacement pour créer des échantillons d'apprentissage à partir desquels le modèle doit apprendre. L'erreur OOB est l'erreur de prédiction moyenne sur chaque échantillon d'apprentissage x_i, en utilisant uniquement les arbres qui n'avaient pas x_i dans leur échantillon bootstrap. Le bagging permet de définir une estimation out-of-bag de l'amélioration des performances de prédiction en évaluant les prédictions sur les observations qui n'ont pas été utilisées dans la construction de l'apprenant.

Lorsque l' agrégation bootstrap est effectuée, deux ensembles indépendants sont créés. Un ensemble, l'échantillon bootstrap, est constitué des données choisies pour être « dans le sac » par échantillonnage avec remise. L'ensemble hors sac comprend toutes les données non choisies dans le processus d'échantillonnage. Lorsque ce processus est répété, comme lors de la création d'une RandomForest , de nombreux échantillons d'amorçage et ensembles OOB sont créés. Les ensembles OOB peuvent être agrégés en un seul ensemble de données, mais chaque échantillon n'est considéré comme hors sac que pour les arbres qui ne l'inclue pas dans leur échantillon bootstrap. L'image ci-dessous montre que pour chaque sac échantillonné, les données sont séparées en deux groupes.



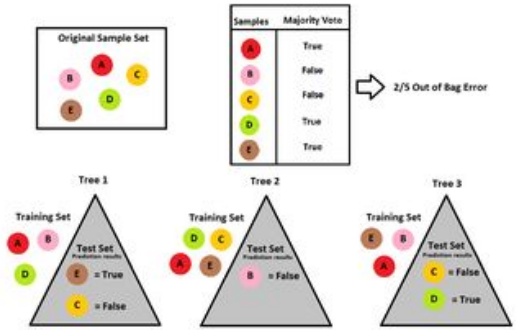
L’évaluation out-of-bag

Étant donné que chaque ensemble hors sac n'est pas utilisé pour entraîner le modèle, il s'agit d'un bon test pour les performances du modèle. Le calcul spécifique de l'erreur OOB dépend de la mise en œuvre du modèle, mais un calcul général est le suivant.

1. Trouver tous les modèles (ou arbres, dans le cas d'une forêt aléatoire) qui ne sont pas entraînés par l'instance OOB.
2. Prener le vote majoritaire du résultat de ces modèles pour l'instance OOB, par rapport à la vraie valeur de l'instance OOB.
3. Compiler l'erreur OOB pour toutes les instances de l'ensemble de données OOB.

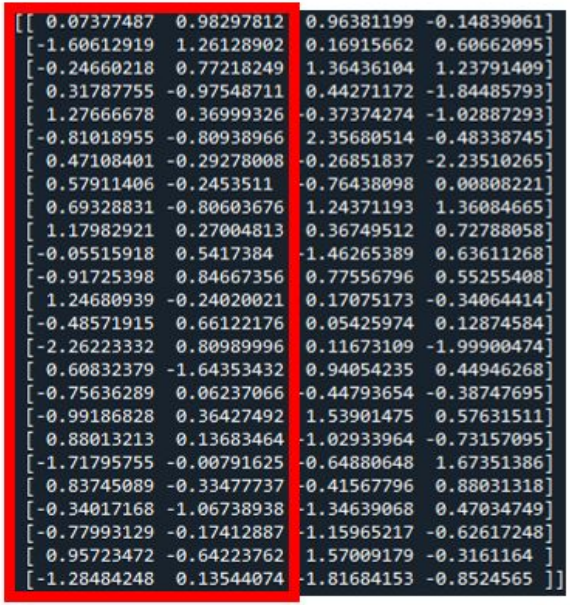
Le processus du bagging peut être personnalisé pour répondre aux besoins d'un modèle. Pour garantir un modèle précis, la taille de l'échantillon d'apprentissage bootstrap doit être proche de celle de l'ensemble d'origine.] En outre, le nombre d'itérations (arbres) du modèle (forêt) doit être pris en compte pour trouver la véritable erreur OOB. L'erreur OOB se stabilisera sur de nombreuses itérations, donc commencer avec un nombre élevé d'itérations est une bonne idée.

Montré dans l'exemple à droite, l'erreur OOB peut être trouvée en utilisant la méthode ci-dessus une fois la forêt configurée.



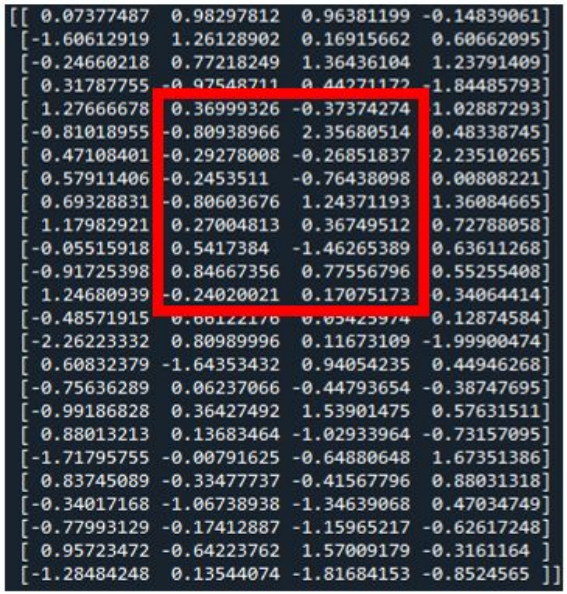
La méthode de “Random Subspaces”

- ➔ Technique utilisée pour introduire une variation parmi les prédicteurs dans un modèle d'ensemble.
- ➔ Permet de diminuer la corrélation entre les prédicteurs afin d’augmenter les performances du modèle d'ensemble.
- ➔ Également connue sous le nom de **feature bagging**.
- ➔ Il crée des sous-ensembles de l'ensemble d'apprentissage qui ne contiennent que certaines fonctionnalités.
- ➔ Le nombre de caractéristiques choisi est échantillonné au hasard à partir de l'ensemble d'apprentissage avec remise.
- ➔ Ces sous-ensembles sont ensuite utilisés afin d'entraîner les prédicteurs d'un ensemble.



La méthode de “Random Patches”

- ➔ **Random Patch = Random Subspace + bagging**
- ➔ Les objectifs de la méthode de Random Patches sont étroitement liés à celui du Bagging et du Random Subspace.
- ➔ Les caractéristiques d'échantillonnage entraînent une plus grande diversité parmi les prédicteurs d'un ensemble et diminuent donc la variance du modèle d'ensemble.
- ➔ C'est la raison pour laquelle la méthode de Random Subspaces et la méthode des de Random Patchs sont utilisées lors de l'apprentissage de certains modèles d'ensemble.



RandomForest et concepts présentés précédemment utilisés par ce modèle

- **Constitués de plusieurs arbres de décisions indépendants**
- **RandomForest = tree bagging + feature sampling**

L'algorithme RandomForest est constitué d'un **ensemble d'arbres de décision indépendants**. Chaque arbre dispose d'une vision parcellaire du problème du fait d'un double tirage aléatoire :

➔ Le **tree bagging** : un tirage aléatoire avec remplacement sur les observations (les lignes de la base de données).
➔ Le **feature sampling** : un tirage aléatoire sur les variables (les colonnes de la base de données).

Tree bagging

On retrouve ici le Bagging qui se détermine par 3 étapes clés :

- Construction de n arbres de décisions en tirant aléatoirement n échantillons d'observations,
- Entraînement de chaque arbre de décision,
- Pour faire une prévision sur de nouvelles données, il faut appliquer chacun de n arbres et prendre la majorité parmi les n prévisions.

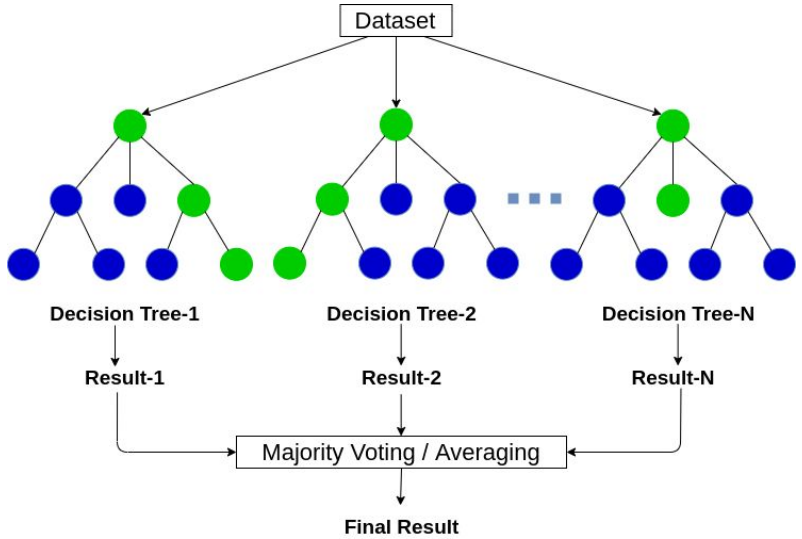
RandomForest et concepts présentés précédemment utilisés par ce modèle

Feature sampling

C'est un processus de tirage aléatoire sur les variables (colonnes de données). Par défaut, on tire Racine n variables pour un problème à n variables au total. Ce processus permet de baisser la corrélation entre les arbres qui pourrait perturber la qualité des résultats. En statistique, on dit que le feature sampling permet de réduire la variance de l'ensemble créé.

A la fin, tous ces arbres de décisions indépendants sont assemblés. La prédiction faite par le RandomForest pour des données inconnues est alors la moyenne (ou le vote, dans le cas d'un problème de classification) de tous les arbres.

Le RandomForest fonctionne sur le principe suivant : plutôt que d'avoir un estimateur complexe capable de tout faire, il utilise plusieurs estimateurs simples (de moins bonne qualité individuelle). Chaque estimateur a une vision parcellaire du problème. Ensuite, l'ensemble de ces estimateurs est réuni pour obtenir la vision globale du problème. C'est l'assemblage de tous ces estimateurs qui rend performante la prédiction.



RandomForest et concepts présentés précédemment utilisés par ce modèle

Critère de division/split

Un arbre de décisions construit des sous-populations par séparations successives des feuilles d'un arbre. Il existe différents critères de séparation pour construire un arbre :

- Le **critère de Gini** organise la séparation des feuilles d'un arbre en se focalisant sur la classe la plus représentée dans le jeu de données : il faut la séparer le plus rapidement possible.
- Le **critère d'entropie** est basé sur la mesure du désordre (comme en thermodynamique) qui règne dans la population étudiée. La construction de l'arbre vise à baisser l'entropie globale des feuilles de l’arbre à chaque étape.

Ensemble des paramètres de la fonction "RandomForestRegressor" de la librairie Scikit-Learn

- **n_estimators** (default = 100) : Nombre d'arbre dans la forêt

- **criterion** {"gini", "entropie"}, default = "gini" : cette fonction permet de mesurer la qualité d'un split. Les critères pris en charge sont "gini" pour l'impureté de Gini et « entropie » pour le gain d'informations. Remarque : ce paramètre est spécifique à l'arborescence.

- **max_depth** int, default = None : La profondeur maximale de l'arbre. Si Aucun, les nœuds sont étendus jusqu'à ce que toutes les feuilles soient pures ou jusqu'à ce que toutes les feuilles contiennent moins d'échantillons min_samples_split.

- **max_features** {"auto", "sqrt", "log2"}, int ou float, defaultt = "auto" : Le nombre de caractéristiques à prendre en compte lors de la recherche du meilleur partage :
    - Si int, tenez compte des max_features caractéristiques à chaque division.
    - Si float, alors max_features est une fraction et les caractéristiques sont prises en compte à chaque division.round(max_features * n_features)
    - Si "auto", alors max_features=sqrt(n_features).
    - Si « sqrt », alors max_features=sqrt(n_features)(identique à « auto »).
    - Si "log2", alors max_features=log2(n_features).
    - Si aucun, alors max_features=n_features.

Ensemble des paramètres de la fonction "RandomForestRegressor" de la librairie Scikit-Learn

- **bootstrap** : Méthode d'échantillonnage des points de données (avec ou sans remise)

- **oob_score** bool, par défaut=Faux : S'il faut utiliser des échantillons hors sac pour estimer le score de généralisation. Uniquement disponible si bootstrap=True.

- **n_jobs** entier , default = None : Le nombre de tâches à exécuter en parallèle. Ce paramètre indique au moteur combien de processeurs il est autorisé à utiliser. Une valeur de "-1" signifie qu'il n'y a pas de restriction alors qu'une valeur de "1" signifie qu'il ne peut utiliser qu'un seul processeur.

- **random_state** int, instance RandomState ou aucun, default = None : Contrôle à la fois le caractère aléatoire de l'amorçage des échantillons utilisés lors de la création d'arbres (if bootstrap=True) et l'échantillonnage des caractéristiques à prendre en compte lors de la recherche de la meilleure division à chaque nœud (if).

Ensemble des paramètres de la fonction "RandomForestRegressor" de la librairie Scikit-Learn

- **warm_start** bool, default = False : Lorsqu'il est défini sur True, réutiliser la solution de l'appel précédent pour ajuster et ajouter plus d'estimateurs à l'ensemble, sinon, ajuster simplement une toute nouvelle forêt.

- **max_samples** entier ou flottant, default = None : Détermine quelle fraction de l'ensemble de données d'origine est donnée à un arbre individuel