ENSEMBLE METHOD

= approach that combines many simple "building blocks" models in order to obtain a single model.

→ weak learners (since they usually lead to mediocre results on their own)

1. BAGGING / BOOTSTRAP AGGREGATION

procedure for reducing the variance of a statistical learning method (frequently used in the context of decision trees).

- Recall: averaging a set of observations reduces variance (if each obs $Z_1, \dots Z_n$ has variance σ^2, the variance of the mean $\bar{Z}$ is given by σ^2/n)

⇨ BOOTSTRAPING uses ↑ as intuition and takes repeated samples (with replacement) from the training dataset. It then trains a model on the b^{th} bootstrapped training set in order to get $\hat{f}^{*b}(x)$. Finally it averages all the predictions to get

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

Classification: $$\hat{f}_{bag}(x) = \underset{1 \le k \le K}{\arg\max} \sum_{b=1}^{B} \text{Ind}(f_b(x) = k)$$

↳ record class predicted by each of the B trees and take a majority vote: the overall prediction is the most commonly occurring class among the B predictions.

Note: - Bagging is only useful for high variance / low bias models!

- Using a very large value of B will not lead to overfitting (B = 100 is usually sufficiently large that the error has settled down).
- On average, each bagged tree makes use of around two thirds of the observations

↳ Why? if we draw items (with replacement) from an original set of size n, the probability of choosing any one item on the first draw is

$$P(x_1) = \frac{1}{n} \quad \text{and} \quad \overline{P(x_1)} = 1 - \frac{1}{n}$$

The probability of never choosing this item on any of the draws is

$$P = \left(1 - \frac{1}{n}\right)^n$$

As n gets larger

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \simeq 0{,}368$$

- the remaining 1/3 of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations.
  - It is possible to predict the response of the i^{th} observation using each of the trees in which that observation was OOB
    - ↳ around B/3 predictions for the i^{th} obs
    - ↳ if we average these predictions / take a majority vote we get a single OOB prediction for the i^{th} observation.
    - ↳ if we do this for all n observations we can use these predictions to compute the MSE or classification error.
      (similar to leave-one-out cross validation)

2. RANDOM FORESTS ← even higher variability between weak learners.
   ↳ Similar to bagging but forces each split to consider only a subset of the predictions

   m = predictor subset size

   On average $\frac{(p-m)}{p}$ of the splits will not even consider the strong predictor (the trees are "decorrelated")

   Note: - when m = p, random forest amounts simply to bagging

3. BOOSTING → (designed for binary classification $y \in \{-1, 1\}$)?
   Each tree is grown using information from previously grown trees. (the trees are grown sequentially).
   Boosting does not involve bootstrap sampling. Instead, each tree is fit on a modified version of the original dataset.

   Each tree attempts to capture a signal that is not yet accounted for by the current set of trees.

   Note: - Boosting can overfit if B is too large!
   - The shrinkage parameter controls the rate at which boosting learns.

how it works: (ADABOOST)
GIVEN: labeled training data with labels $y_i \in \{-1, 1\}$

and a set / class H of T possible weak classifiers.

Initialize: We introduce a weight $w_j^{(1)}$ for each training sample and set

$$w_j^{(1)} = \frac{1}{m} \quad \text{for each } j$$

For $t = 1, 2, \ldots T$

1) We train a weak classifier $h_t \in H$ using the data and weights associated with them. We then choose the classifier with the lowest error

$$\epsilon_t = \sum_{j=1}^{m} w_j^{(t)} \, \text{Ind}\left(y_j \neq h_t(x_j)\right)$$

2) Update the weights (they increase for samples that are misclassified and decrease for samples that are correctly classified)

$$w_j^{(t+1)} = w_j^{(t)} \exp\left(-\alpha_t \, y_j \, h_t(x_j)\right)$$

α_t → reliability coefficient

$$\alpha_t = \log_e\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

(α always > 0)

note the sign!
sign of output $h_t(x_j)$ is the same as label → ⊕
otherwise ↳ ⊖

correct answer → $y_j h_t(x_j) > 0$
→ $\exp(-\alpha_t y_j h_t(x_j)) < 0$
→ weight less important

3) Normalize weights so that they sum to 1.

4. DROPOUT

Regularization method for Artificial Neural Networks.
→ Can be seen as an ensemble method.