III – Trees and Ensembling methods

Contents:

- *A first non-linear model: decision trees*
- *Ensembling: bagging and boosting*
- *Random forest and boosted trees*

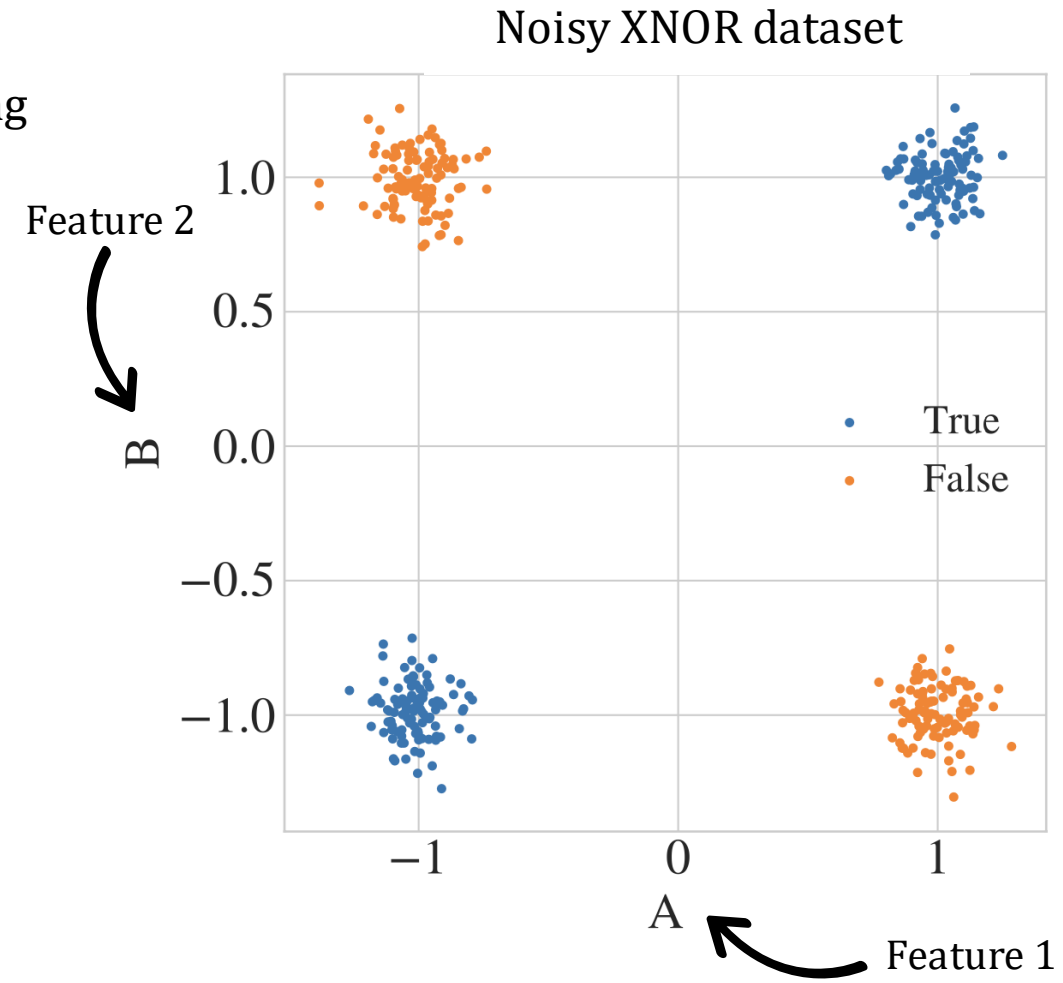
A classification task: the XOR dataset

Decision trees

- Consider a **classification task** on an artificial dataset replicating the XNOR function

A	B	XNOR
True	True	True
True	False	False
False	True	False
False	False	True

- Data are $n = 500$ couples $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right) \Longrightarrow$ **Supervised learning**
- The target variable $y \in \{-1,1\}$ is discrete $\Longrightarrow$ **Classification**
- A linear classification would not be able to learn such a function*
- **Decision trees** to the rescue!

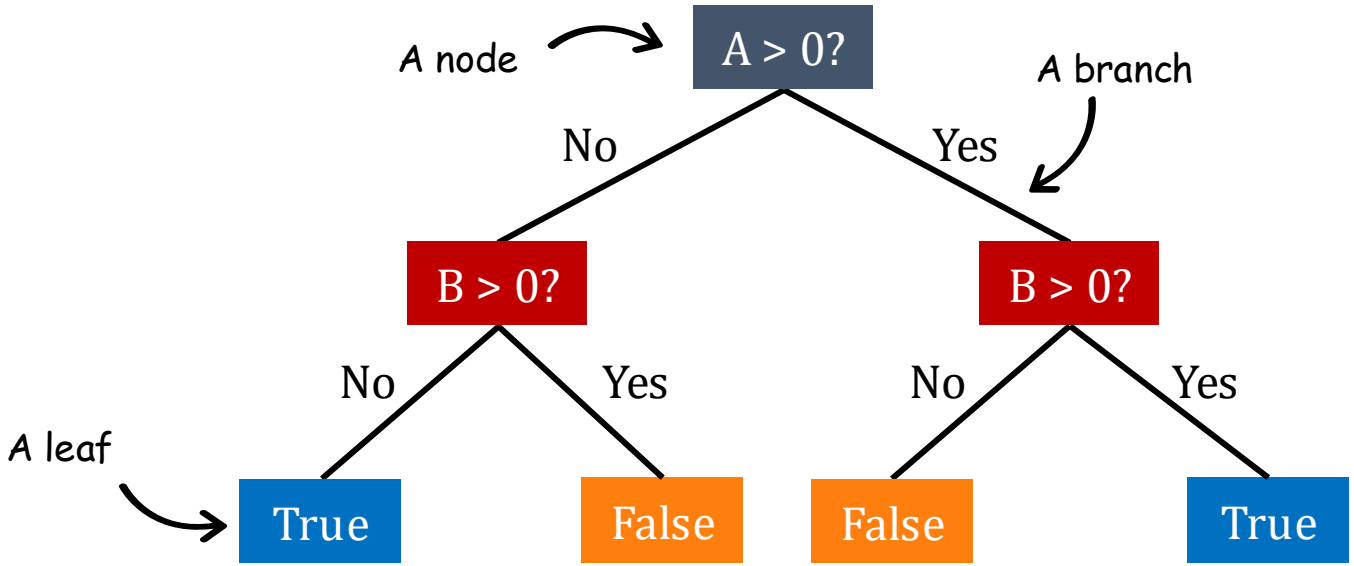


*In fact, an alternative (not discussed here) would be to use **kernels** to find a higher dimensional embedding ϕ that makes the classes linearly separable and then use a linear classifier.

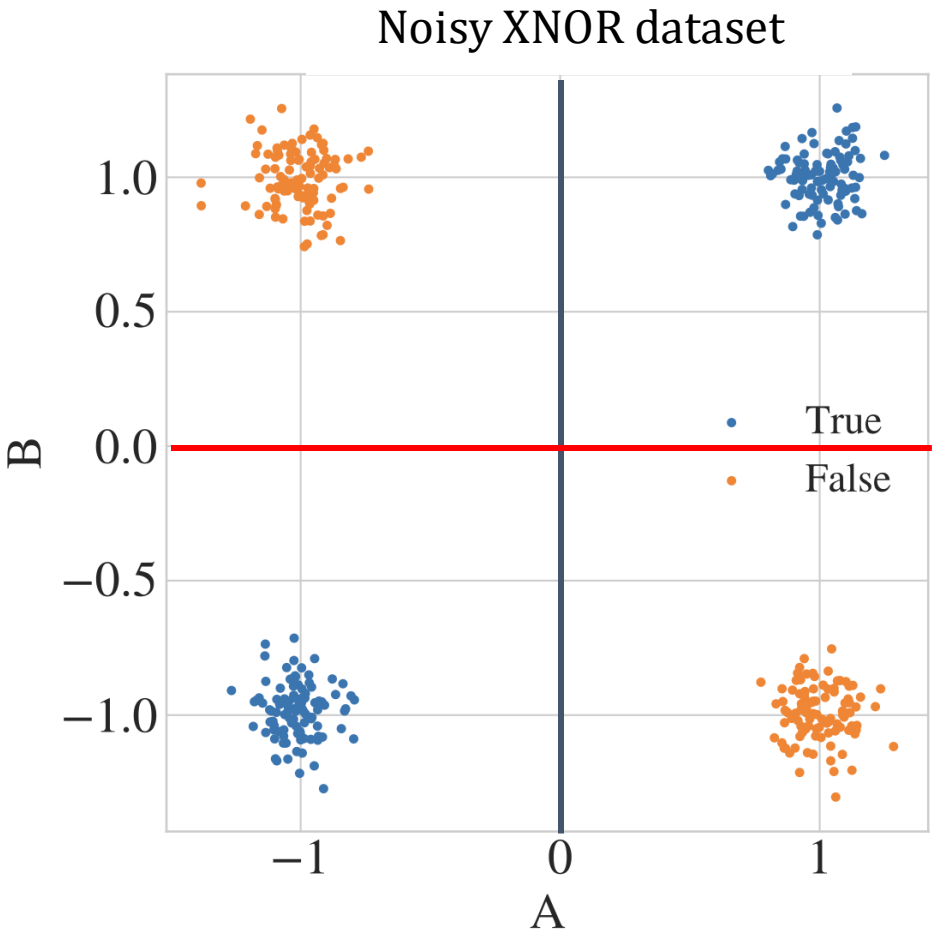
Intuition behind decision trees

Decision trees

- Decision trees incrementally ask questions about the features to split the problem into smaller, simpler (binary) decisions



- All root and inner nodes question the value of a feature, and branches split the dataset into **different regions to which a datapoint can belong uniquely**



A more formal view of decision trees

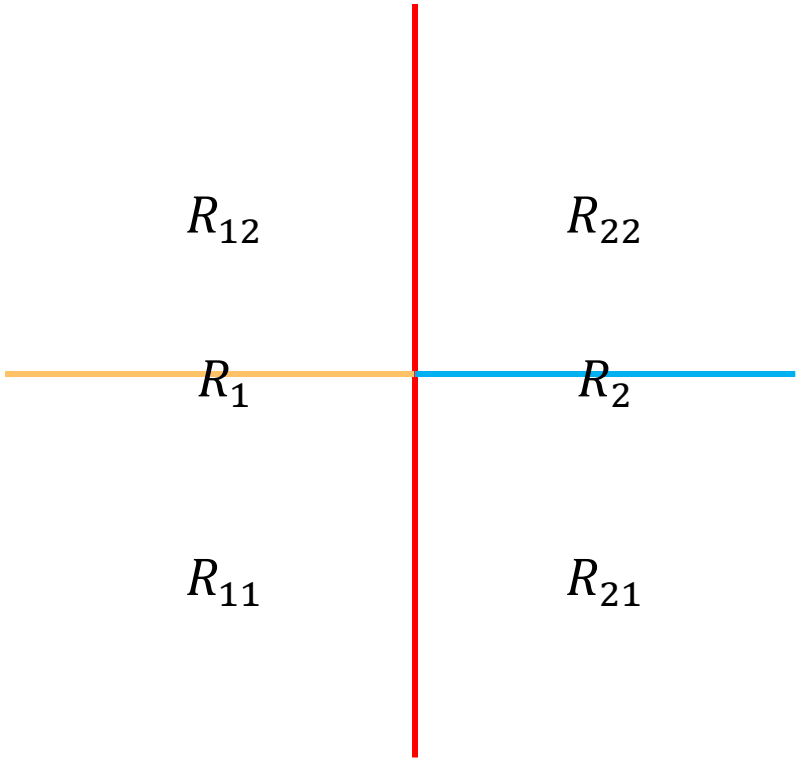
Decision trees

- More formally, at a given node in parent region R_i asking a question about the j^{th} feature, we create two regions:

$$R_{i1} = \{\boldsymbol{x} \mid x_j < \alpha_i^j, \boldsymbol{x} \in R_i\}$$
$$R_{i2} = \{\boldsymbol{x} \mid x_j \geq \alpha_i^j, \boldsymbol{x} \in R_i\}$$

- The parameters $\boldsymbol{\theta}$ of decision trees are the threshold values at each nodes (the sequence of α)
- Decision tree minimise a criterion at each node of the tree

Noisy XNOR dataset



Decision trees

- To find the best possible parameters, decision trees minimize a cost function through **a measure of "impurity" at each node until a stopping criterion is met** (depth of the tree, minimum number of samples in nodes)
- For instance, popular measures are the **cross-entropy** (for classification) or the **squared error** (for regression)

Classification

$$\ell(R_i) = -\sum_{k=1}^{q} \rho_k^i \log_2 \rho_k^i \,, \qquad \rho_k^i = \frac{\left|\{\boldsymbol{x}^{(j)} \mid \boldsymbol{x}^{(j)} \in R_i,\ y^{(j)} = k\}\right|}{\left|\{\boldsymbol{x}^{(j)} \mid \boldsymbol{x}^{(j)} \in R_i\}\right|}$$

Proportion of points in class k at node i

→ If the node is *pure*, $\rho_k^i = 1$ for a single class giving an entropy of 0, if the node is mixed with several classes, then entropy is large

Regression

$$\ell(R_i) = \frac{1}{N}\sum_{j=1}^{N} (y_i - m)^2 \,, \qquad m = \frac{1}{N}\sum_{x^{(j)} \in R_i}^{N} y^{(j)}$$

Average value in the region

→ Takes the value in the region that minimizes the average squared error

Decision trees

Classification

$$\ell(R_i) = -\sum_{k=1}^{q} \rho_k^i \log_2 \rho_k^i \,, \qquad \rho_k^i = \frac{\left|\{\boldsymbol{x}^{(j)} \mid \boldsymbol{x}^{(j)} \in R_i,\ y^{(j)} = k\}\right|}{\left|\{\boldsymbol{x}^{(j)} \mid \boldsymbol{x}^{(j)} \in R_i\}\right|}$$

Proportion of points in class k at node i

Regression

$$\ell(R_i) = \frac{1}{N}\sum_{j=1}^{N} (y_i - m)^2 \,, \qquad m = \frac{1}{N}\sum_{x^{(j)} \in R_i}^{N} y^{(j)}$$

Average value in the region

- At each node, we look for the split into left/right regions maximizing the **gain**

$$G_i(\alpha_t^j) = \ell(R_i) - \left(\frac{N_L}{N}\ell(R_{i1}) + \frac{N_R}{N}\ell(R_{i2})\right)$$

N_L, N_R: number of datapoints in left/right regions

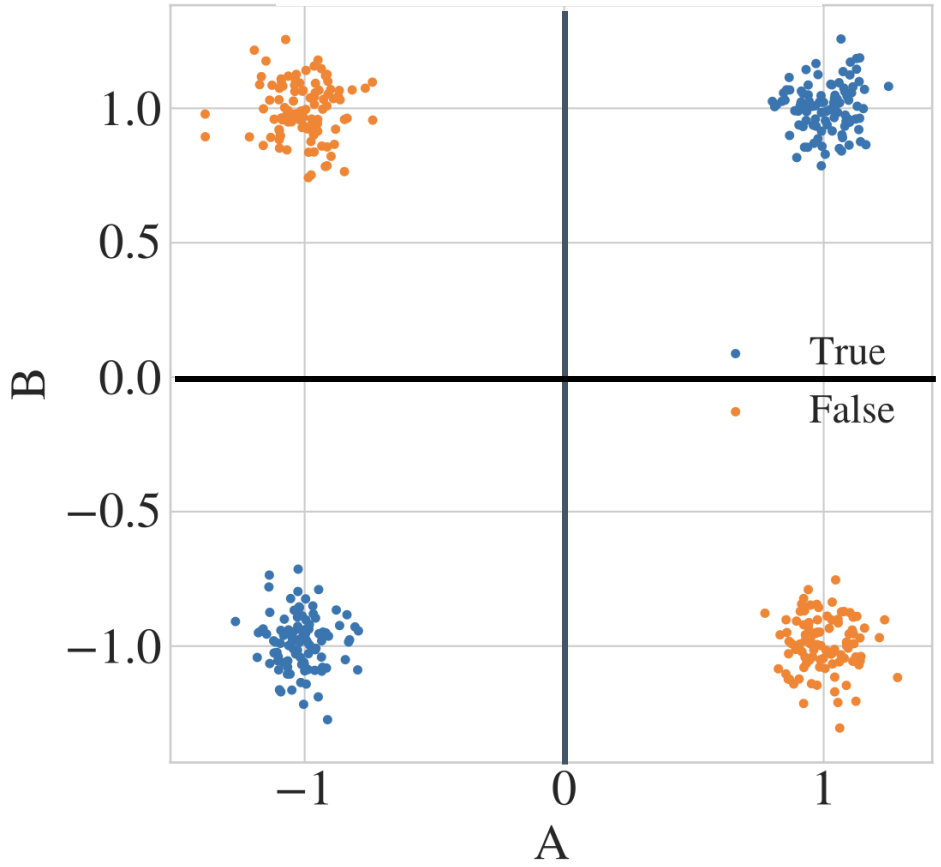
\+ can handle categorical values, easy to interpret, fast to compute, both regression and classification

− **Shallow trees: high bias estimators (underfit), deep trees: high variance estimators (overfit)**

Decision trees

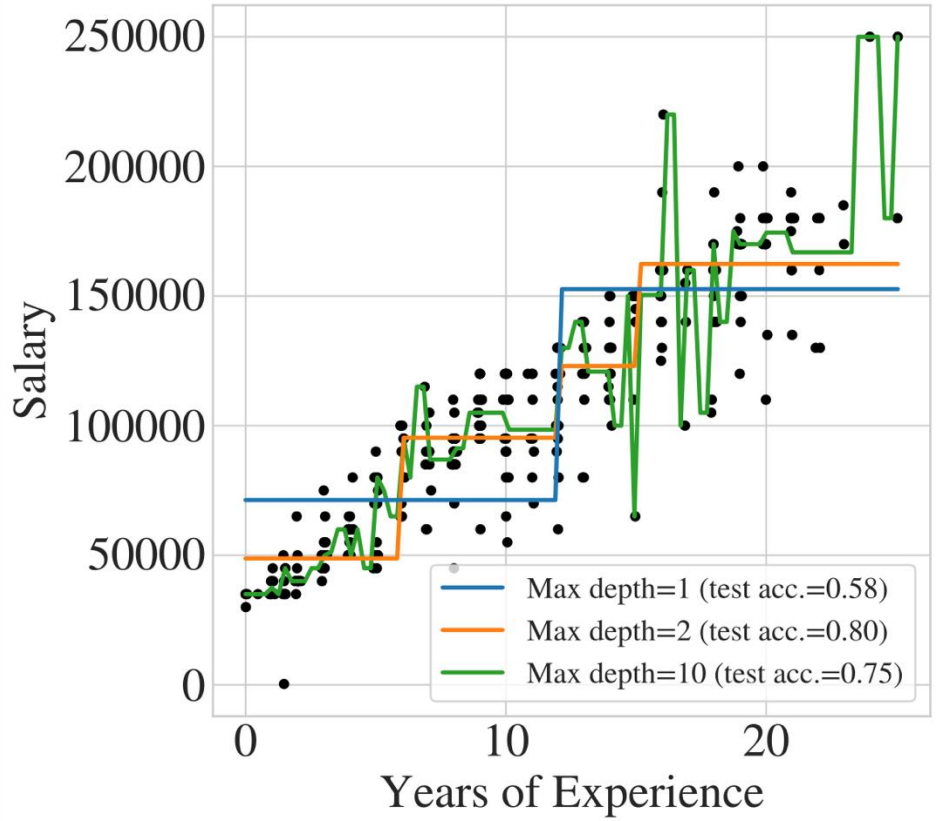
Classification

Noisy XNOR dataset



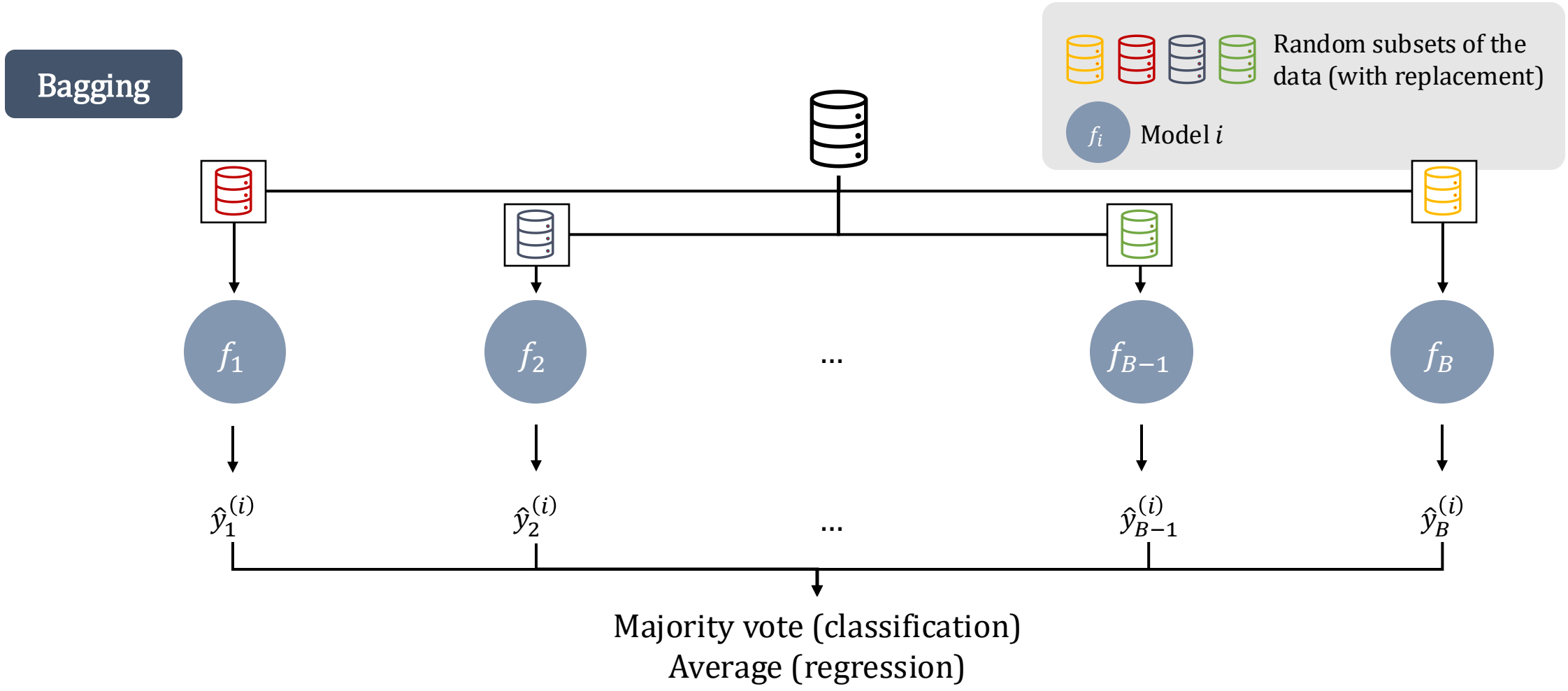
Regression

Salary prediction



Ensembling methods: bagging

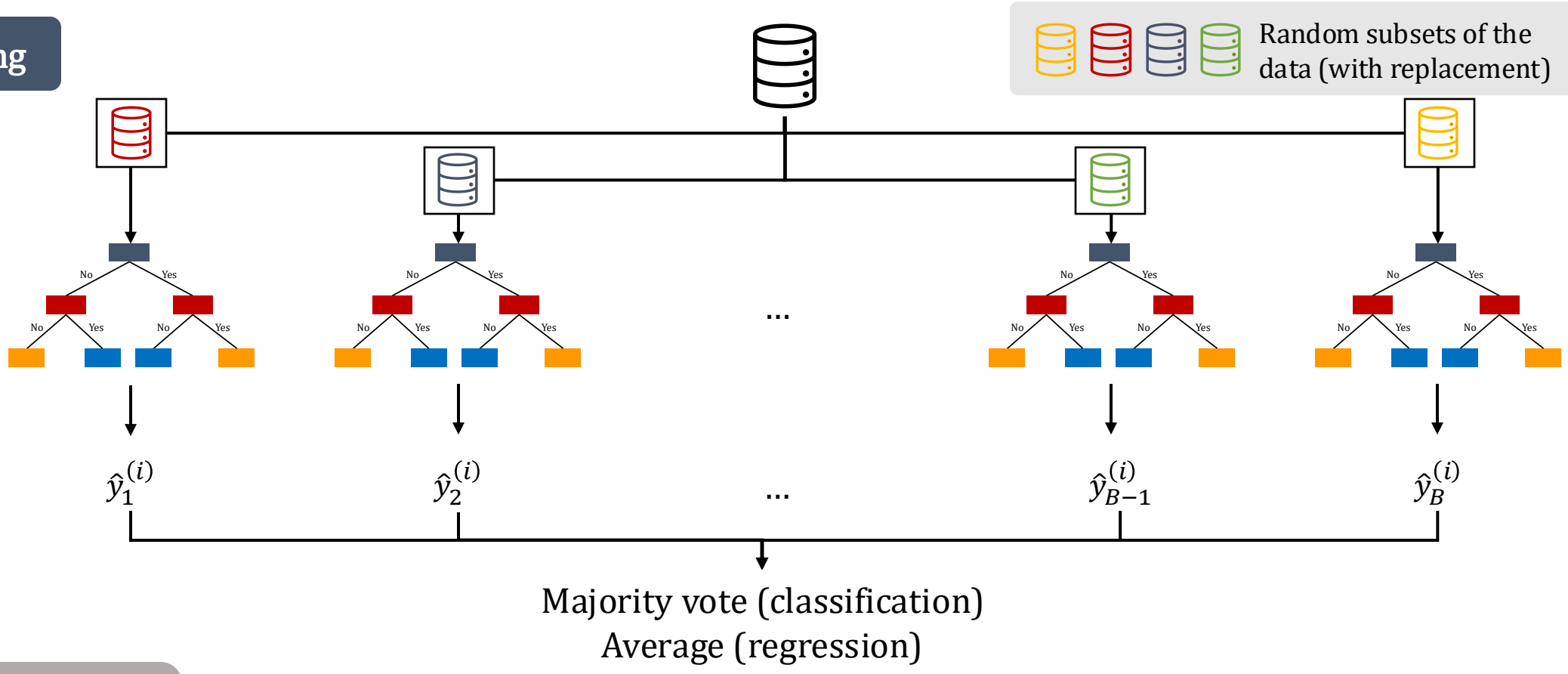
- To circumvent the problems that can have weak learners like decision trees, **ensembling methods** were proposed



- To reduce the variance, models need to be **uncorrelated**: this is achieved by using **random sampling of the dataset**

Ensembling methods: random forest

Bagging



Random Forest = Bagging of decision trees + feature bagging

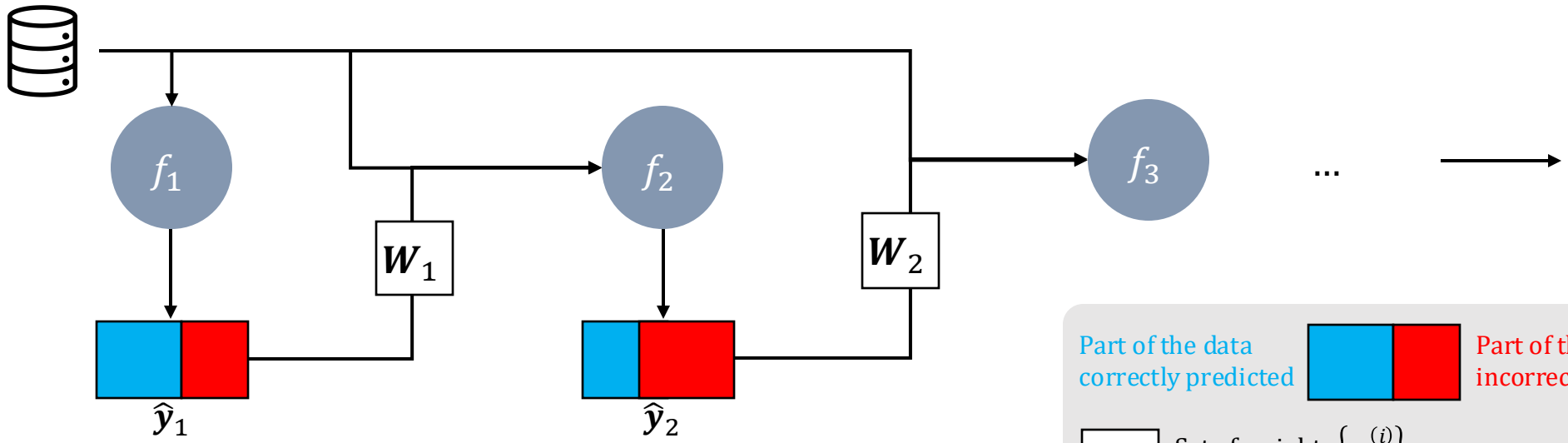
\+ Can still handle categorical values, both regression and classification, **reduced variance**

− More expansive to compute (need to train B trees instead of one), harder to interpret

Ensembling methods: boosting

- While bagging trains high-variance models in parallel to reduce the variance of the combined estimate, **boosting trains high-bias models sequentially to reduce the overall bias**

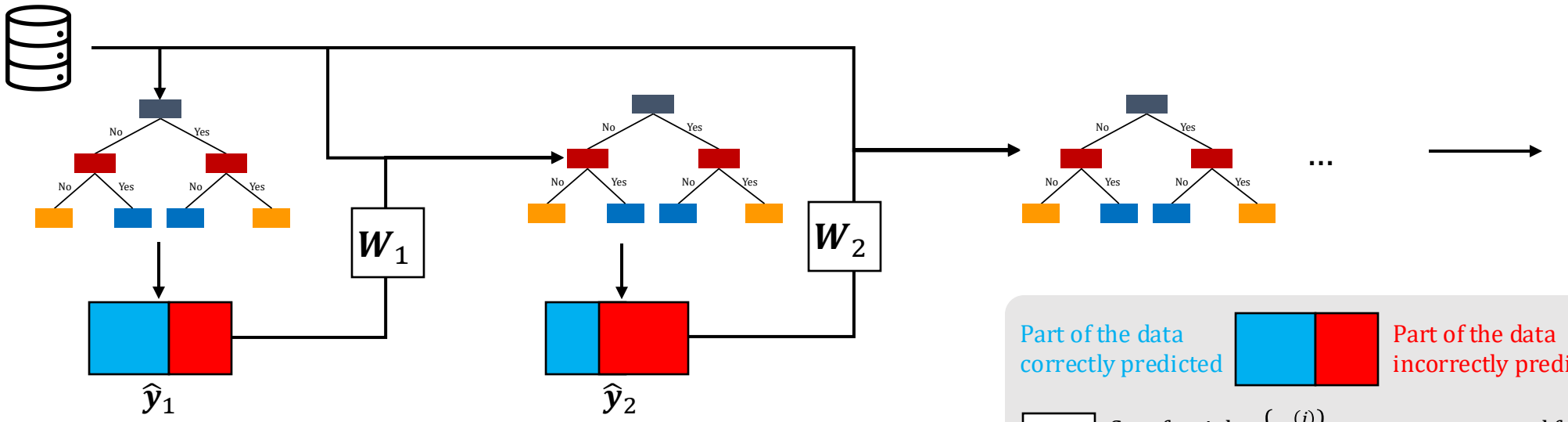
Boosting



- Successive learners f_i are fed by data $\boldsymbol{X}_i$, a weighted version of the initial dataset $\boldsymbol{X}$, **giving more weights to the errors committed by the previous model $\boldsymbol{f_{i-1}}$**
- The output is, as in bagging, **a linear combination of all the learners** weighted by the contribution of each tree
- The choice of weighting and training depends on the algorithm and context (see Adaboost or gradient boosting)

Ensembling methods: boosted trees

Boosting



Part of the data correctly predicted

Part of the data incorrectly predicted

W_i Set of weights $\left\{w_j^{(i)}\right\}_{j=1,\ldots,n_{\text{train}}}$ computed from the mistakes made by the model $i-1$

Boosted trees = Boosting of decision trees

\+ Both regression and classification (residuals or weighted classification error), **reduced bias**, good performances

\− More expansive to compute, increased variance, subject to overfitting

Comparison on our regression problem