Pour une botanomancie rigoureuse: lire l'importance dans les feuilles des forêts (aléatoires) et en extraire des préceptes élémentaires.

Erwan Scornet (Ecole Polytechnique)

StatLearn - 5th April 2023

Why do we need interpretability?

Why do we need interpretability?

Machine learning is used for **decision support**.

Predicting is not enough

Understanding predictions is vital

- for Machine learning to be accepted (sensible applications in health, justice, defense)
- To improve algorithms (e.g., detect unfairness and try to correct it)

Keywords: trust, transparency, accountability, fairness, ethics.

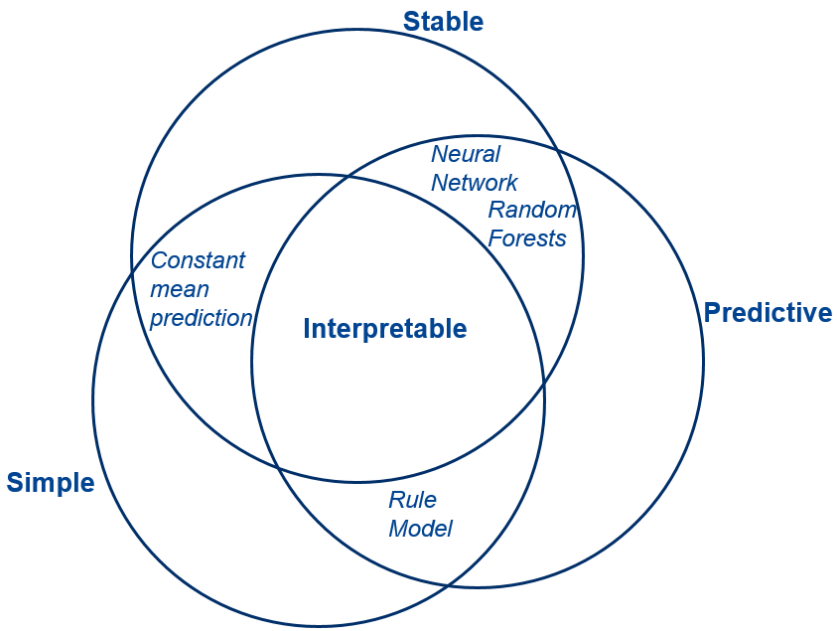
NIPS2017 debate: Interpretability is necessary for Machine learning

https://www.youtube.com/watch?v=93Xv8vJ2acI

Interpretable Models

- No agreement about a rigorous definition of interpretability (Lipton, 2016; Doshi-Velez and Kim, 2017; Murdoch et al., 2019)

- Minimum requirements for interpretability
  1. Simplicity (Murdoch et al., 2019)
  2. Stability (Yu, 2013)
  3. Predictivity (Breiman, 2001b)



Existing Approaches

- Black-box models

E.g. Neural networks, Random forests

Combined with post-processing
E.g. variable importance
sensitivity analysis
local linearization

Hard to operationalize

Existing Approaches

- Black-box models

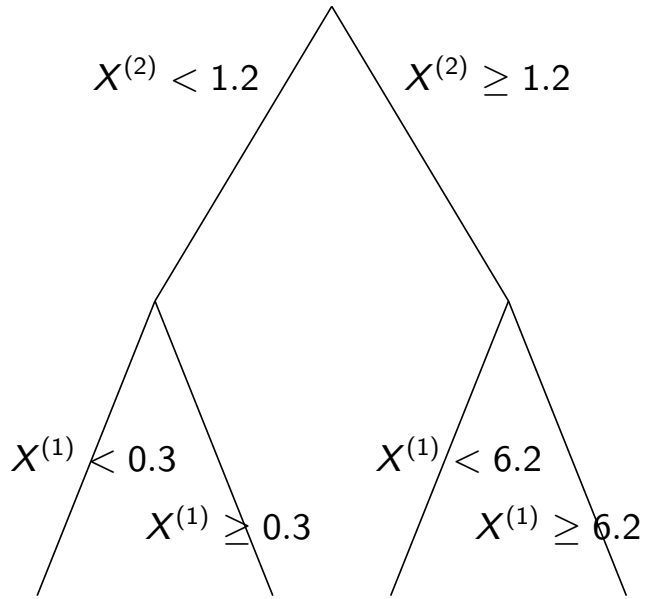
E.g. Neural networks, Random forests

Combined with post-processing
E.g. variable importance
sensitivity analysis
local linearization

Hard to operationalize

- Interpretable models

E.g. decision trees, decision rules



Unstable

Outline

Decision trees

Decision tree: a tool to help you taking a decision via asking a sequence of questions.

Decision trees

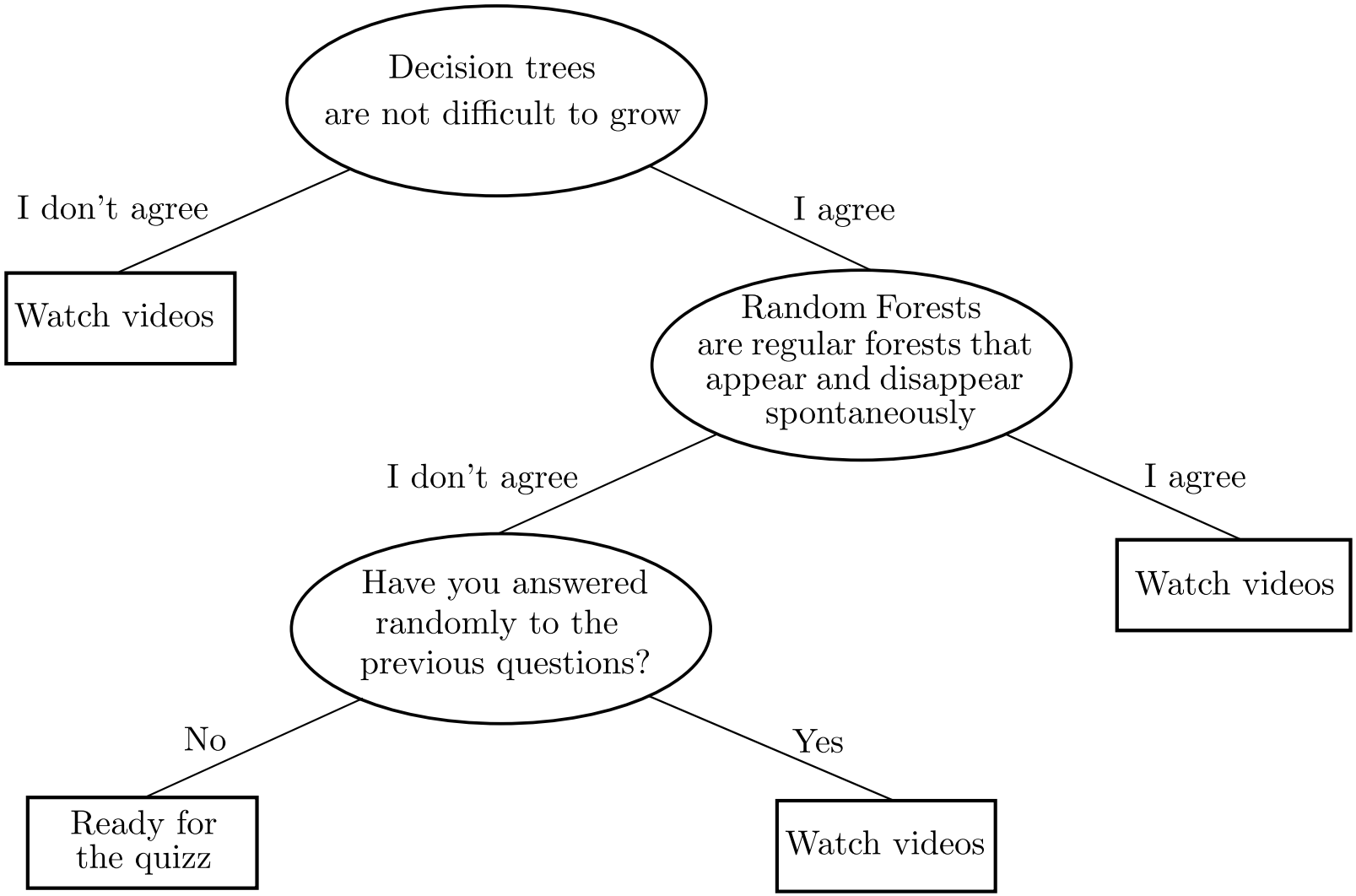
Decision tree: a tool to help you taking a decision via asking a sequence of questions.

A first example - Should you (re)watch the videos on decision trees?

Decision trees

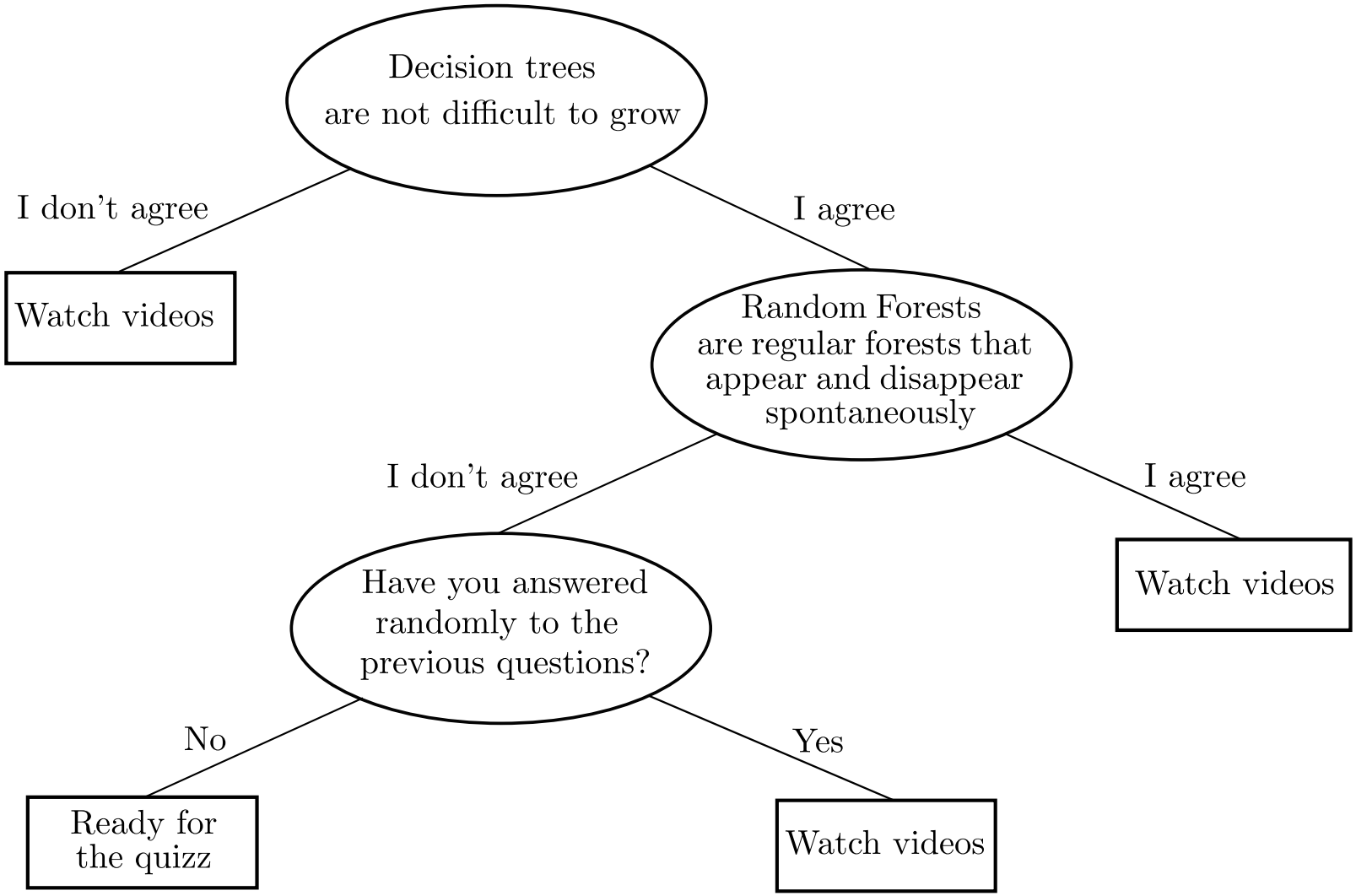
Decision tree: a tool to help you taking a decision via asking a sequence of questions.

A first example - Should you (re)watch the videos on decision trees?



Decision trees

A first example - Should you (re)watch the videos on decision trees?



$\rightarrow$ Such a tree comes from common sense or from domain experts.

Decision trees

Can we collect data to automatically create a decision tree, without domain experts?

Decision trees

Can we collect data to automatically create a decision tree, without domain experts?

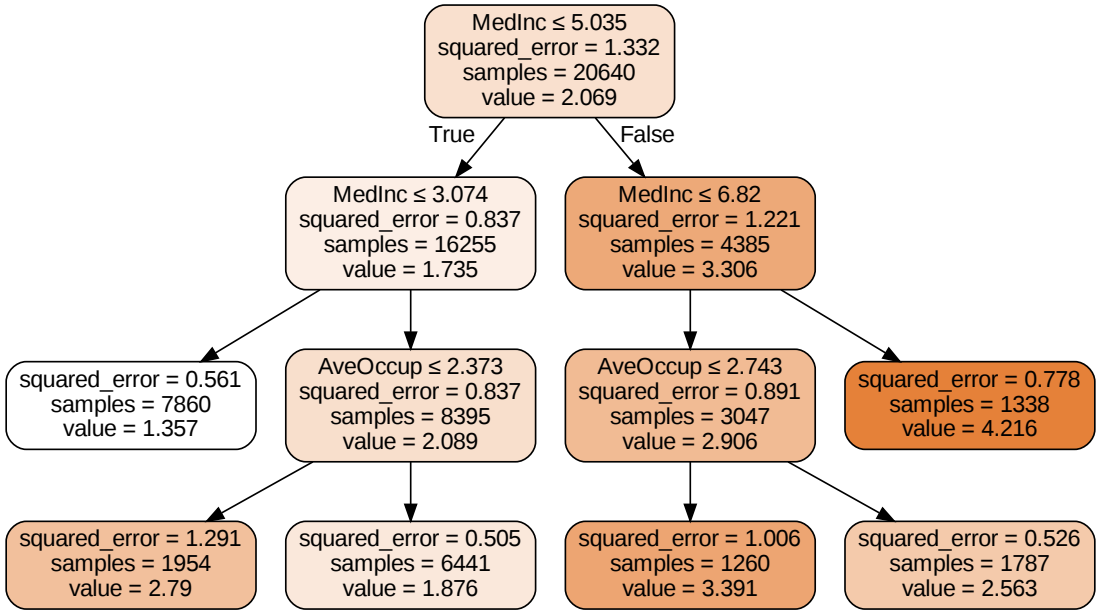
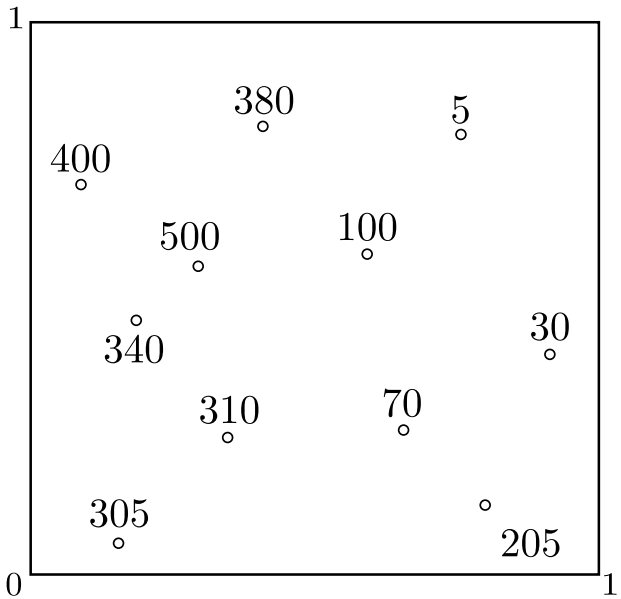


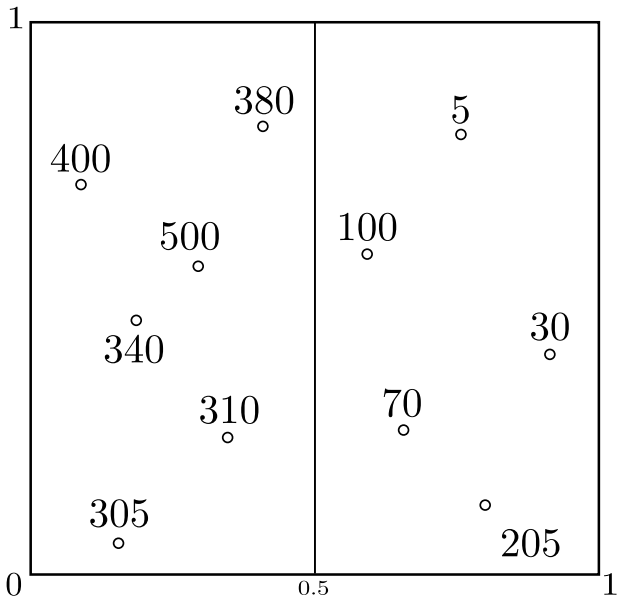
Figure: Output of a decision tree trained on a real-estate data set

Construction of Decision trees - regression



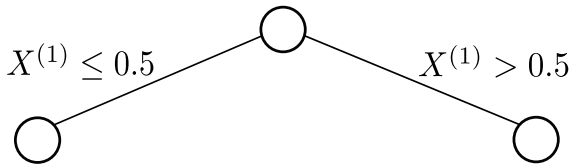
$k = 0$

Construction of Decision trees - regression

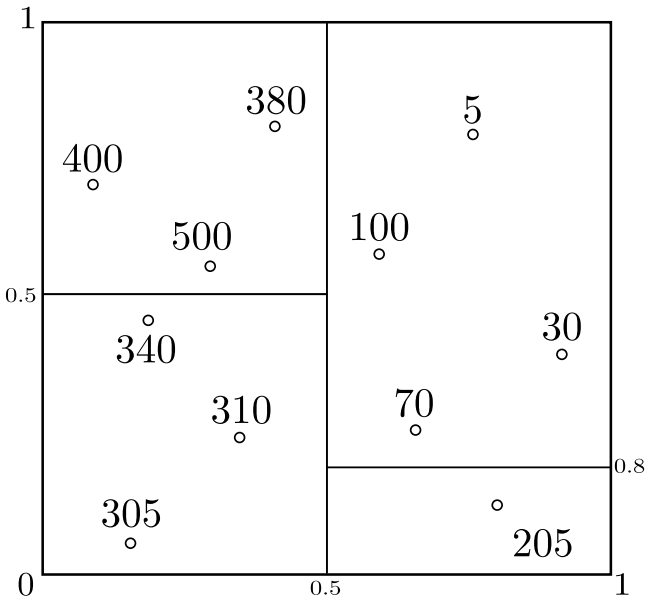


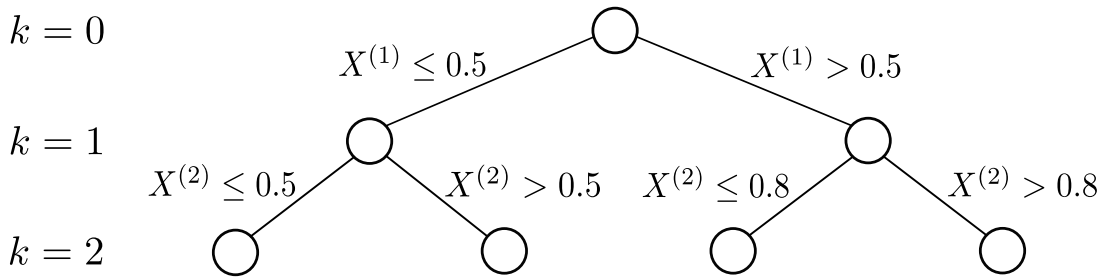
$k = 0$

$k = 1$

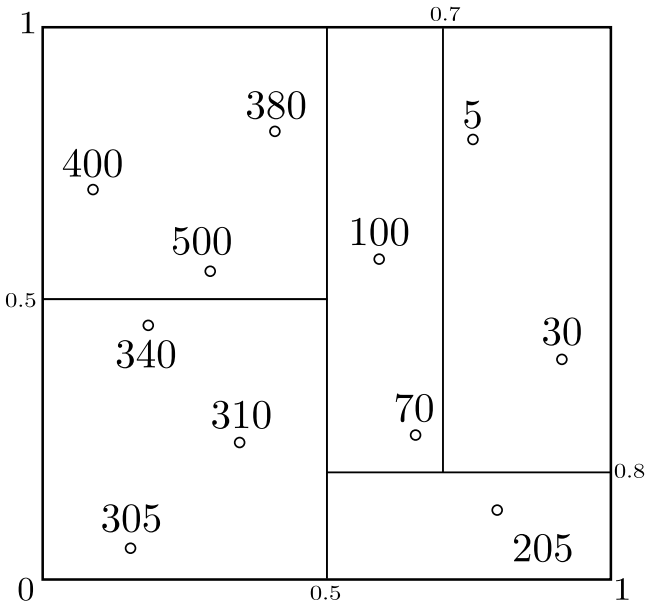


Construction of Decision trees - regression





Construction of Decision trees - regression

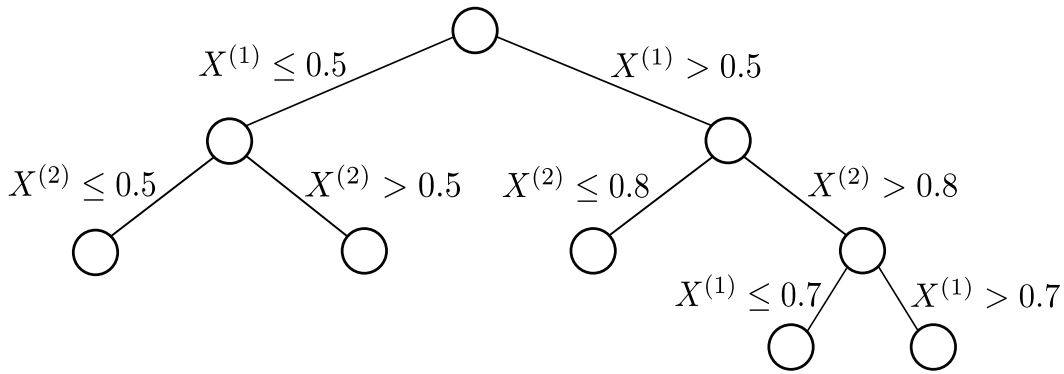


$k = 0$

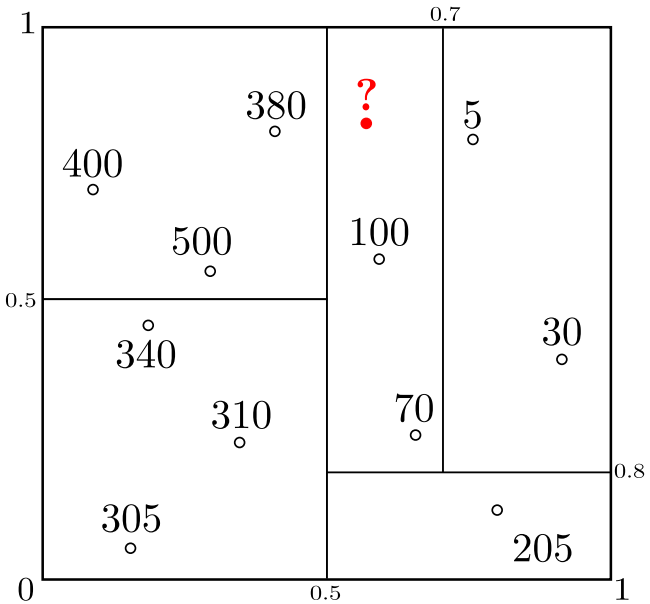
$k = 1$

$k = 2$

$k = 3$



Construction of Decision trees - regression

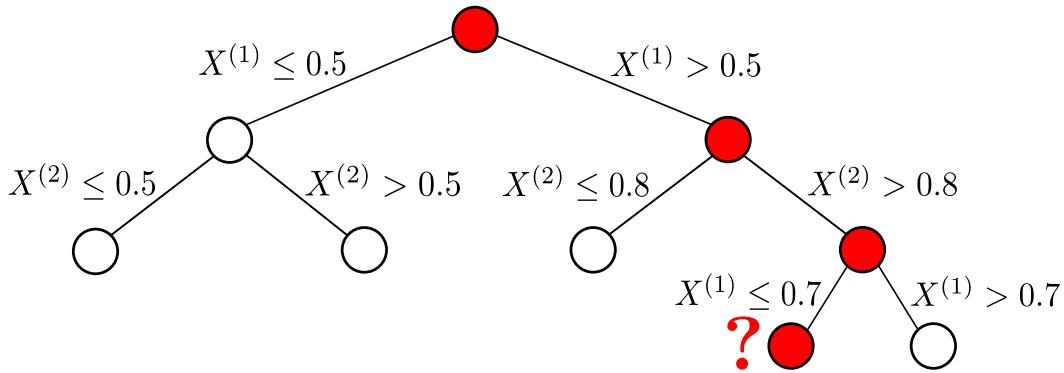


$k = 0$

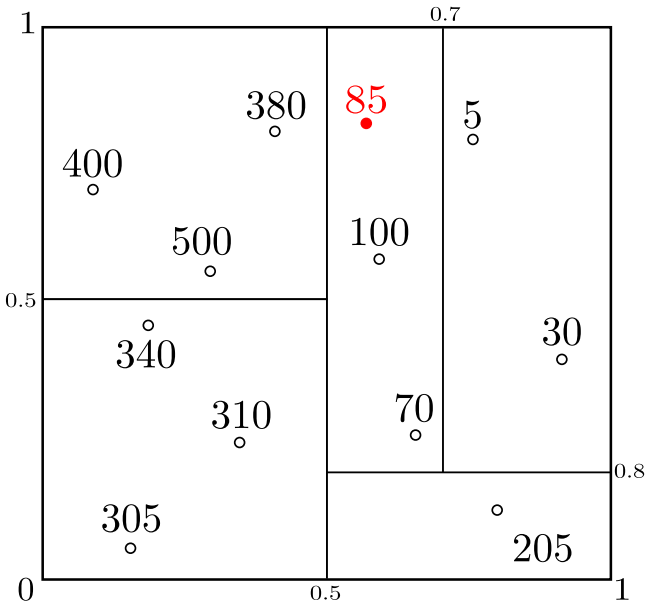
$k = 1$

$k = 2$

$k = 3$



Construction of Decision trees - regression

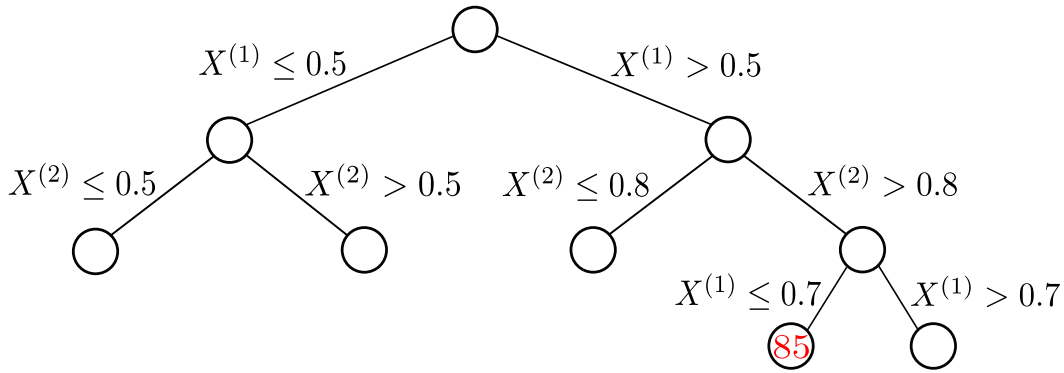


$k = 0$

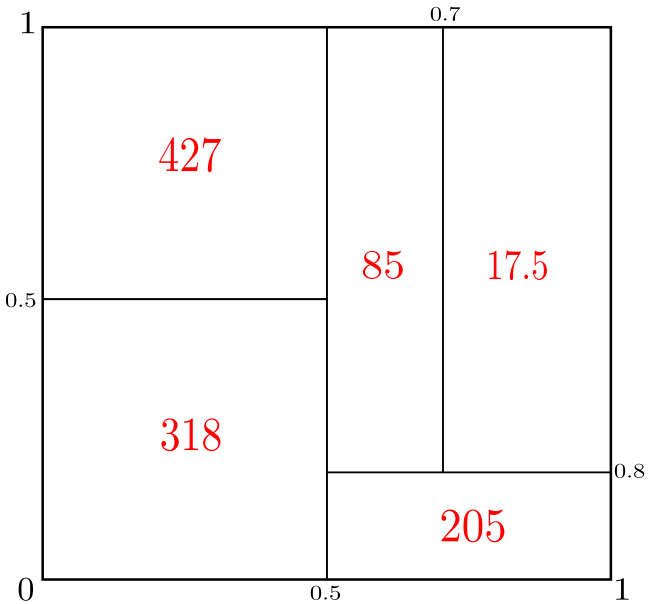
$k = 1$

$k = 2$

$k = 3$



Construction of Decision trees - regression

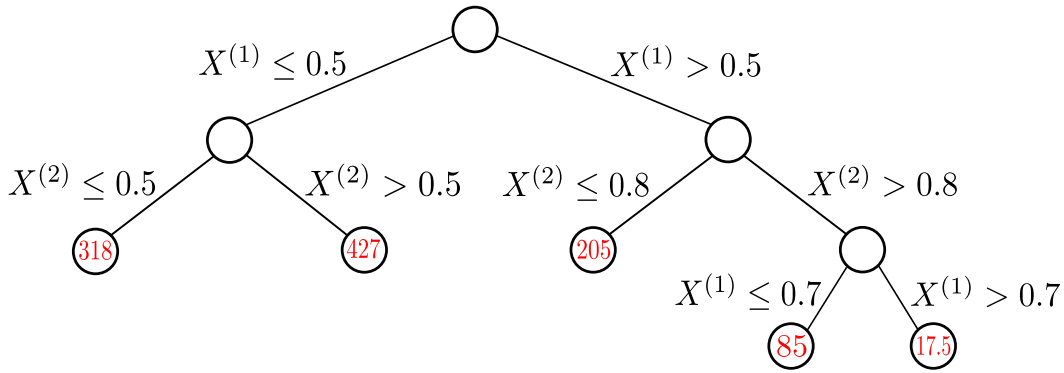


$k = 0$

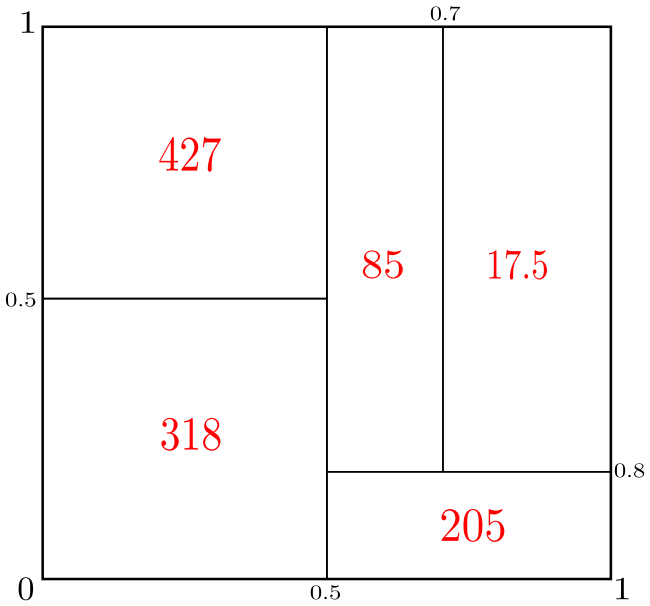
$k = 1$

$k = 2$

$k = 3$



Construction of Decision trees - regression

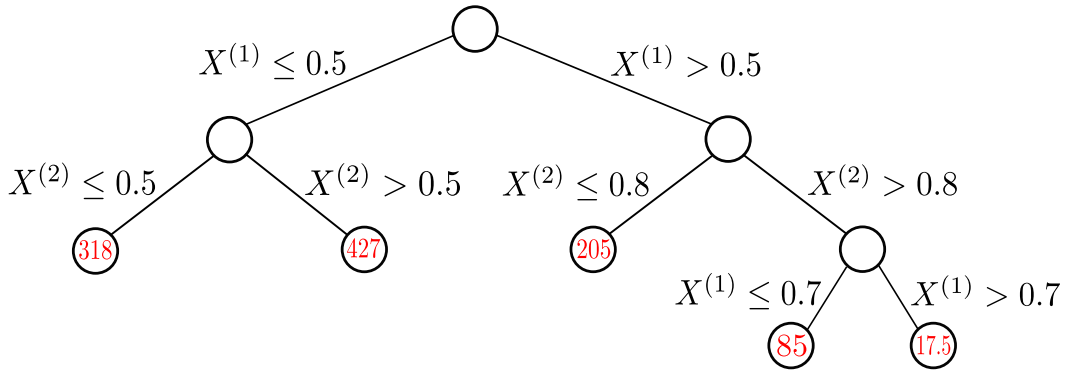


$k = 0$

$k = 1$

$k = 2$

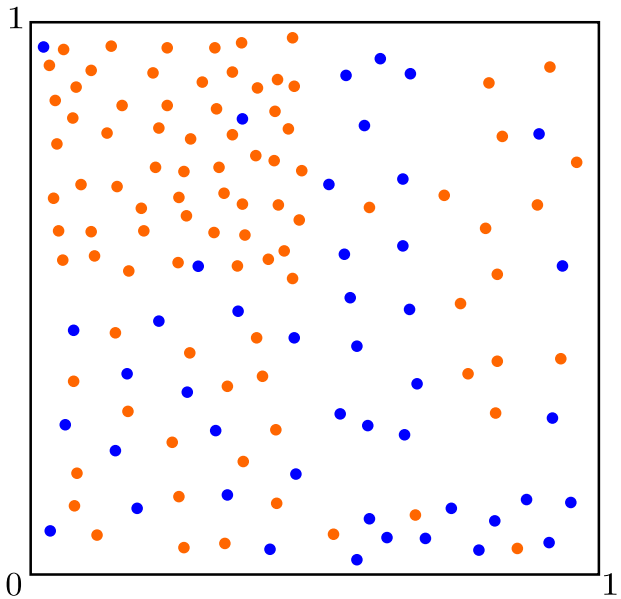
$k = 3$



Decision tree building

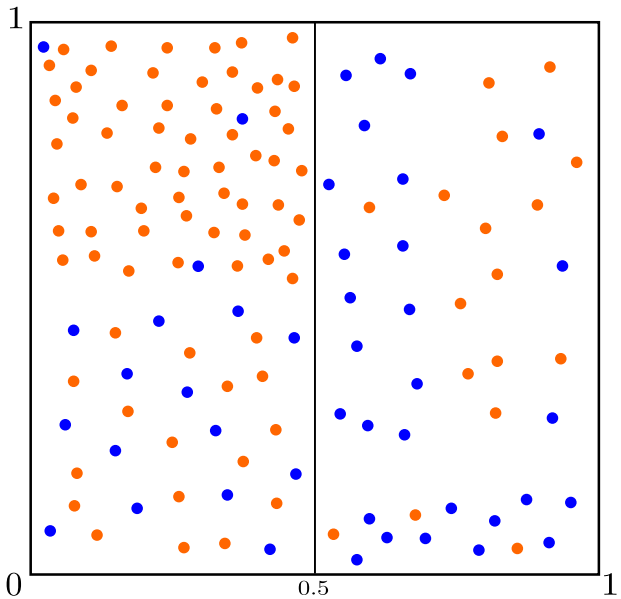
- Requires a splitting rule
- Requires a stopping rule
- Requires a prediction rule

Construction of Decision trees - classification



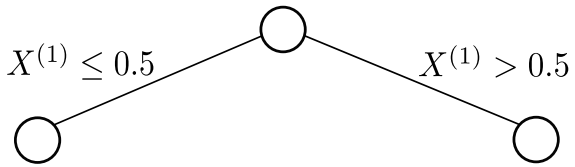
$k = 0$

Construction of Decision trees - classification

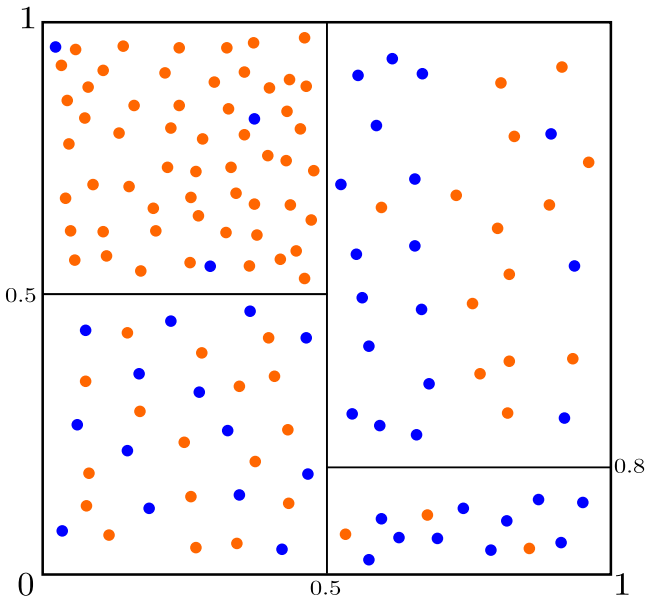


$k = 0$

$k = 1$



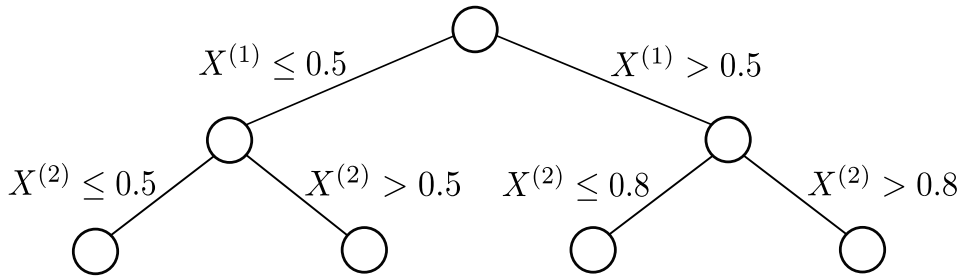
Construction of Decision trees - classification



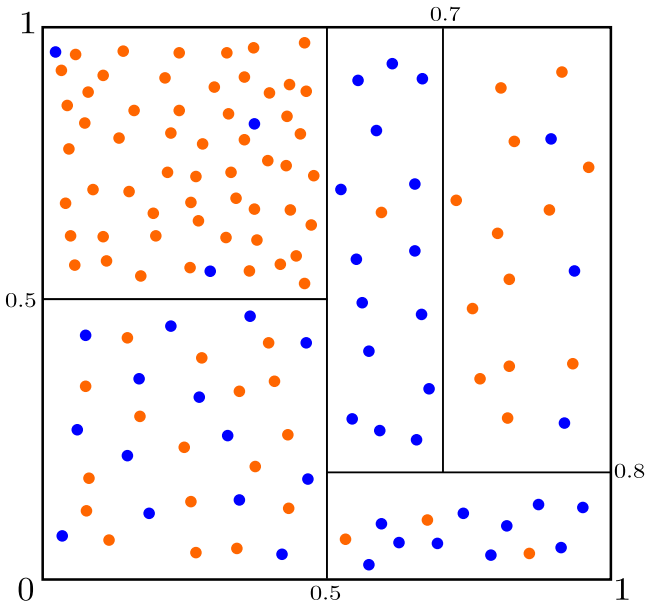
$k = 0$

$k = 1$

$k = 2$



Construction of Decision trees - classification

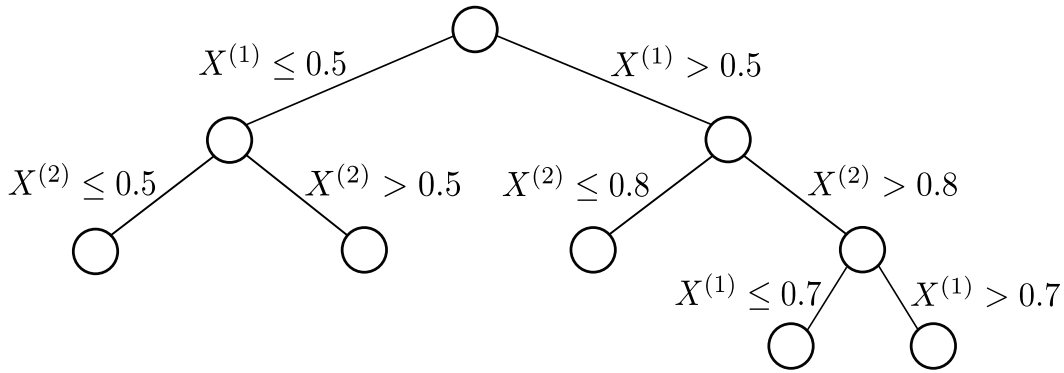


$k = 0$

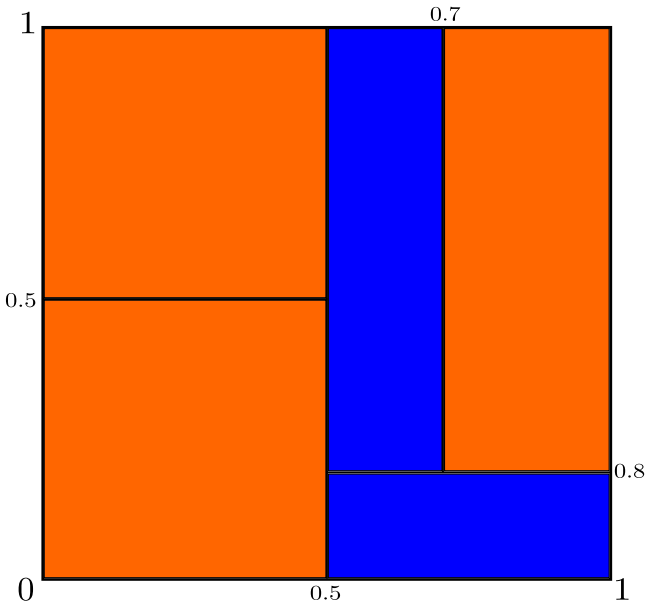
$k = 1$

$k = 2$

$k = 3$



Construction of Decision trees - classification

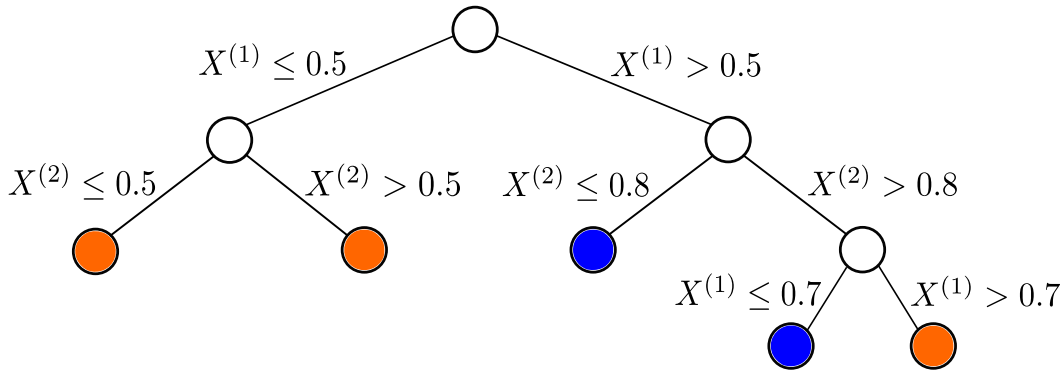


$k = 0$

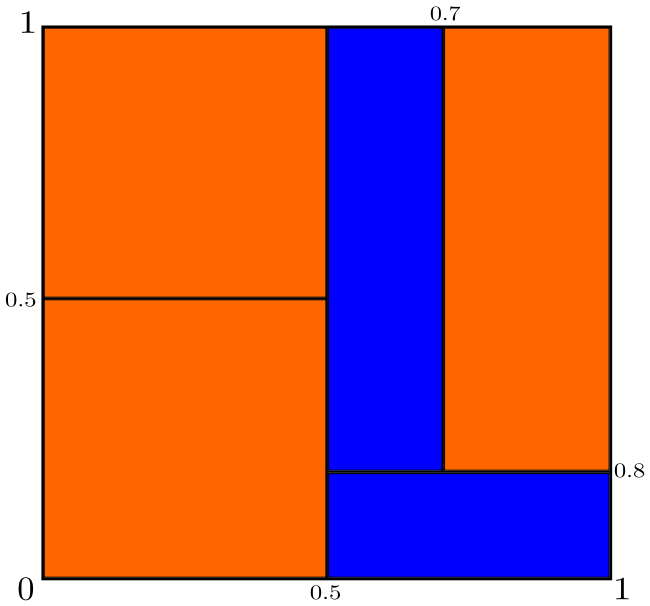
$k = 1$

$k = 2$

$k = 3$



Construction of Decision trees - classification

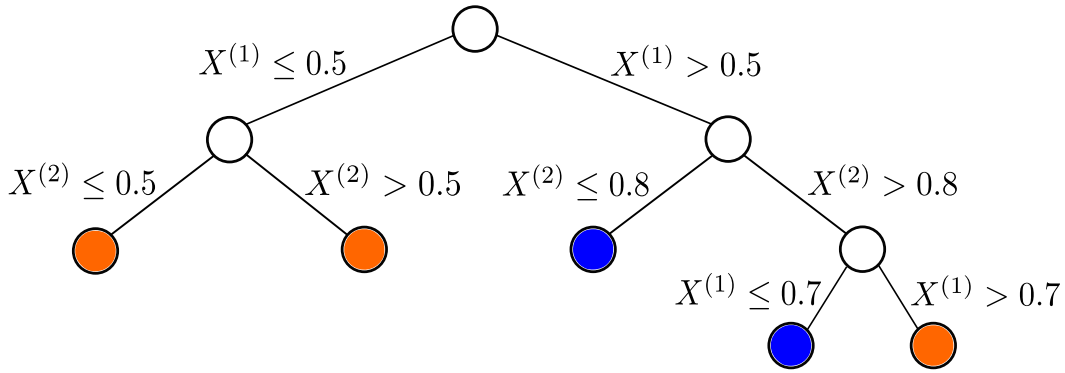


$k = 0$

$k = 1$

$k = 2$

$k = 3$



Decision tree building

- Requires a splitting rule
- Requires a stopping rule
- Requires a prediction rule

Splitting criterion

Finding the best split in a cell A requires to define an impurity criterion Imp. Based on this criterion, one can define the impurity reduction associated to a split (j, s) as

$$\Delta Imp(j, s; A) = Imp(A) - p_L Imp(A_L) - p_R Imp(A_R),$$

where p_L (resp. p_R) is the fraction of observations in A that fall into A_L (resp. A_R).

The best split $(j^\star, s^\star)$ is then chosen as

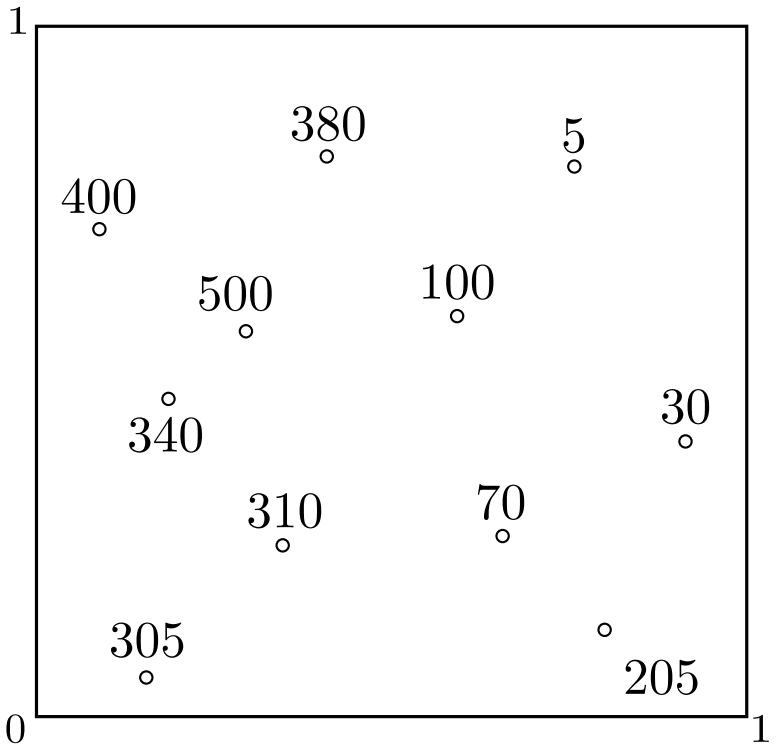
$$(j^\star, s^\star) \in \underset{j,s}{\operatorname{argmax}} \Delta Imp(j, s; A).$$

An instance of impurity measure: the empirical variance (regression)

$$\begin{aligned} Imp_V(A) &= \mathbb{V}_n[Y|X \in A] \\ &= \frac{1}{N_n(A)} \sum_{i, X_i \in A} (Y_i - \bar{Y}_A)^2, \end{aligned}$$

where $N_n(A)$ is the number of observations in the cell A and $\bar{Y}_A$ the mean of the Y_is over all observations in A.

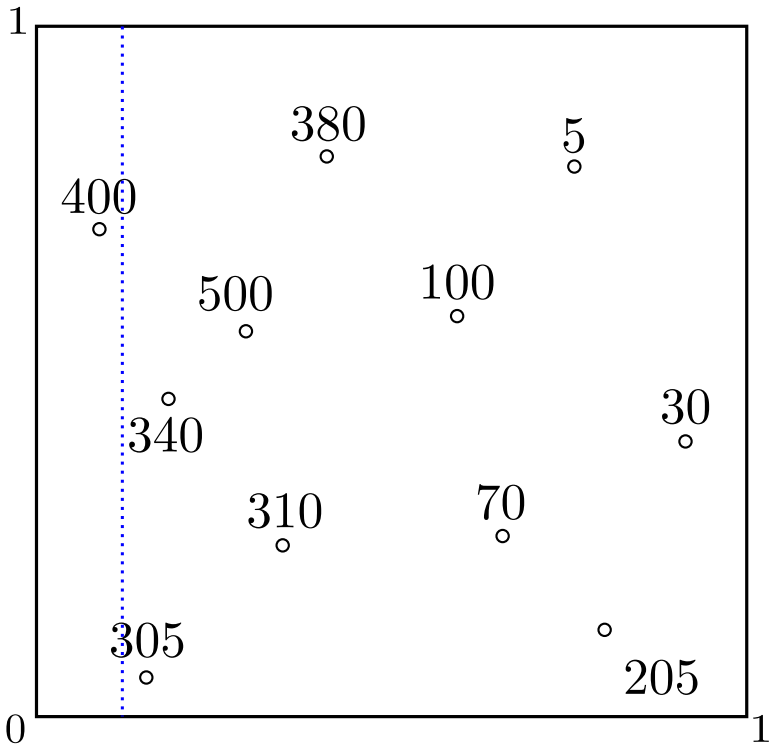
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

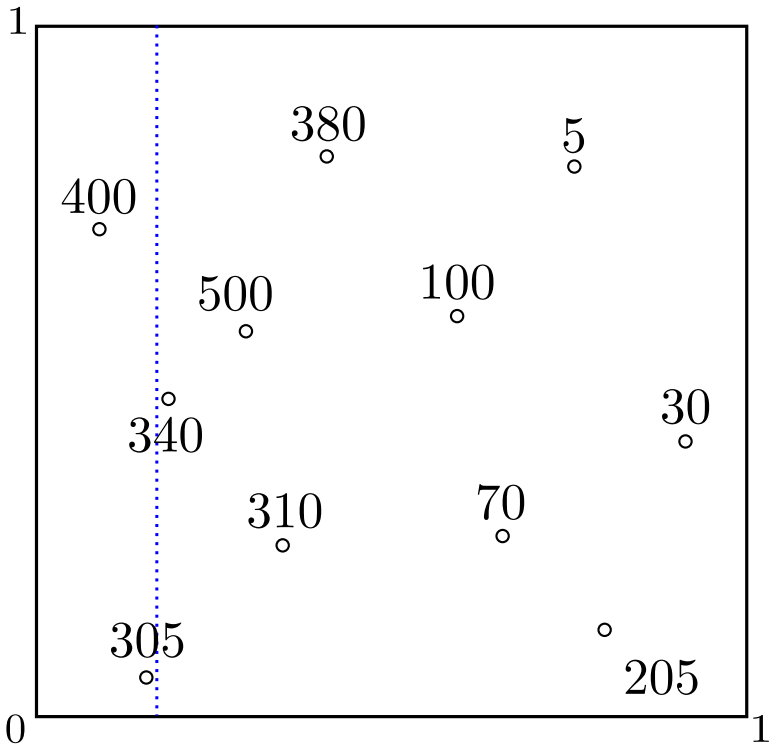
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

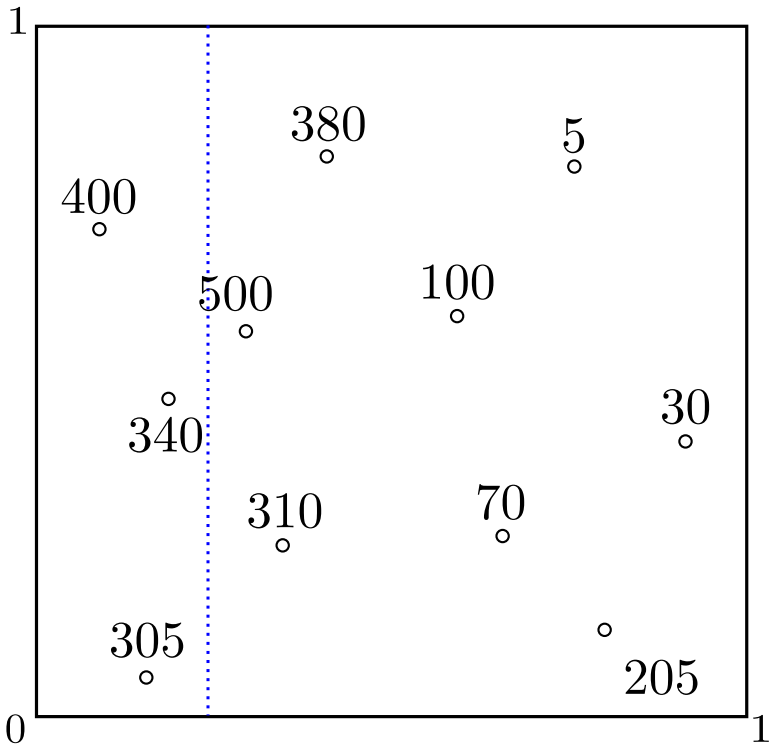
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

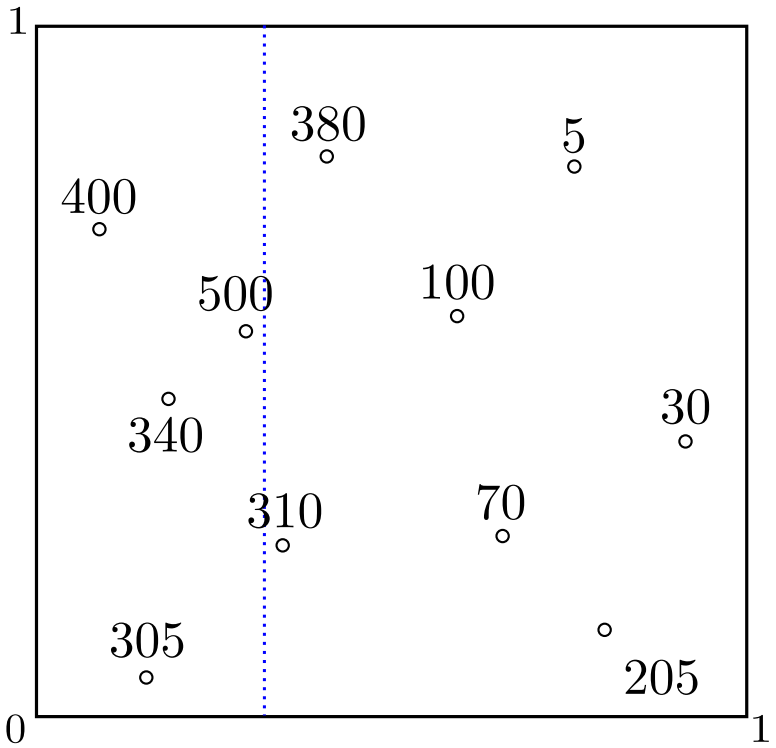
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

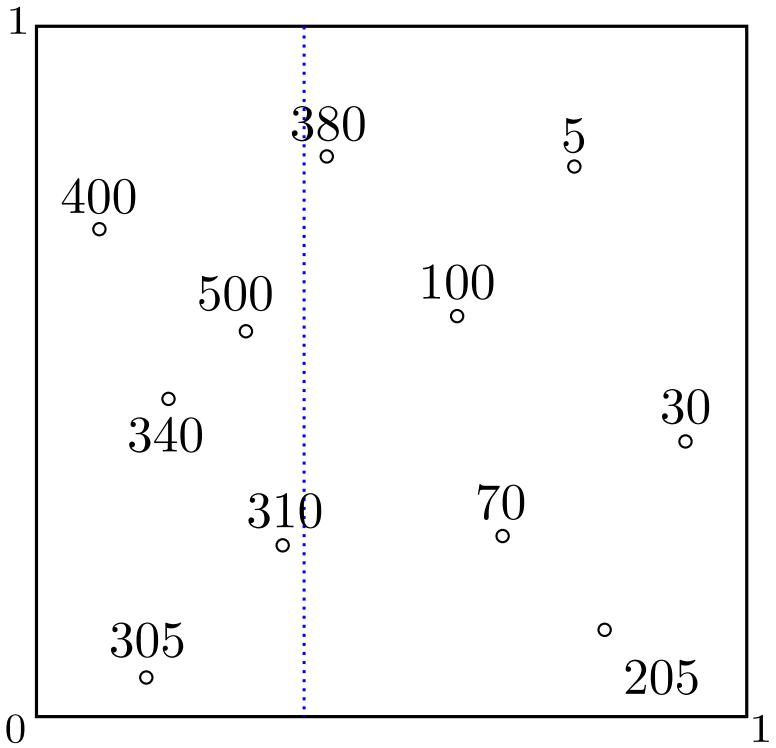
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

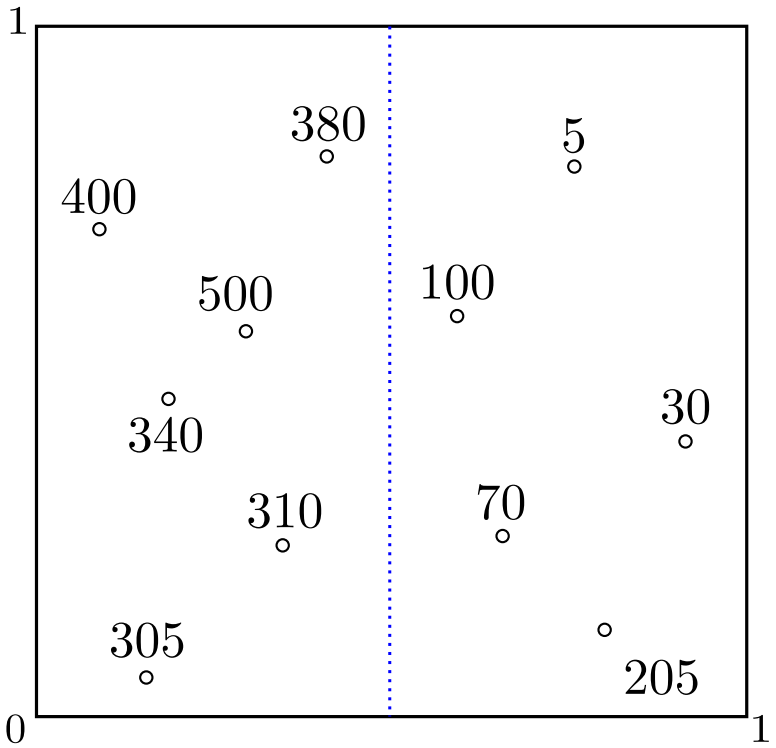
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

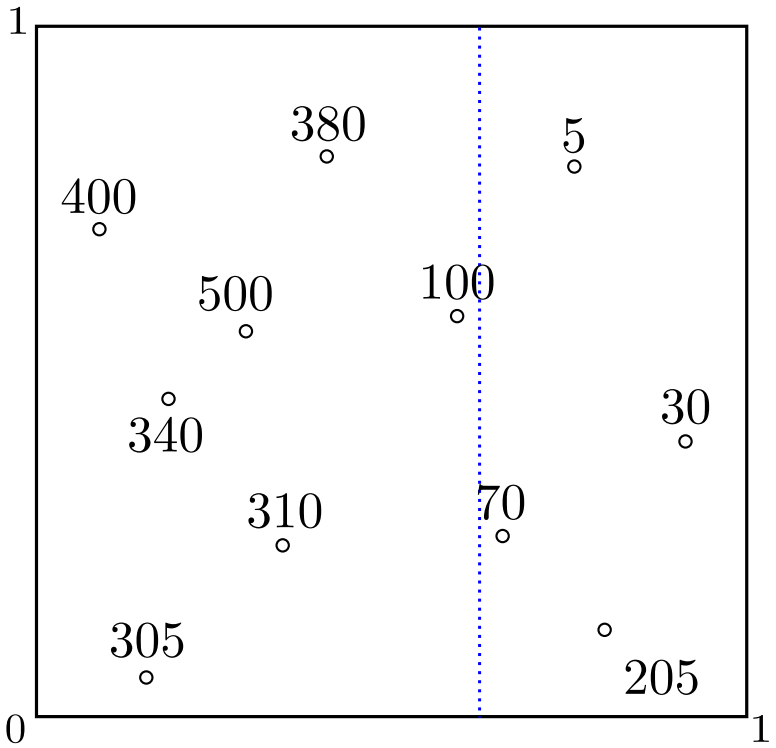
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

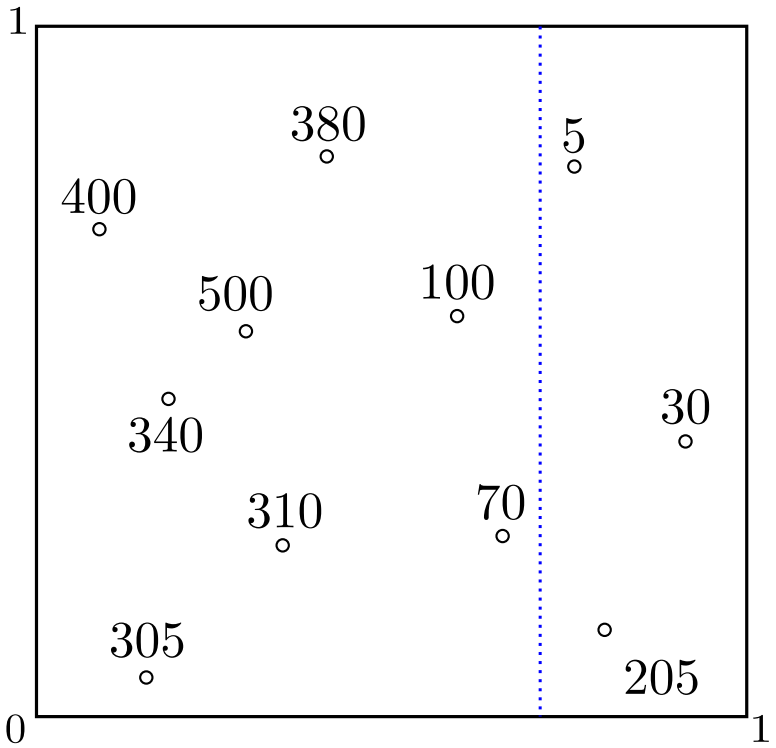
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

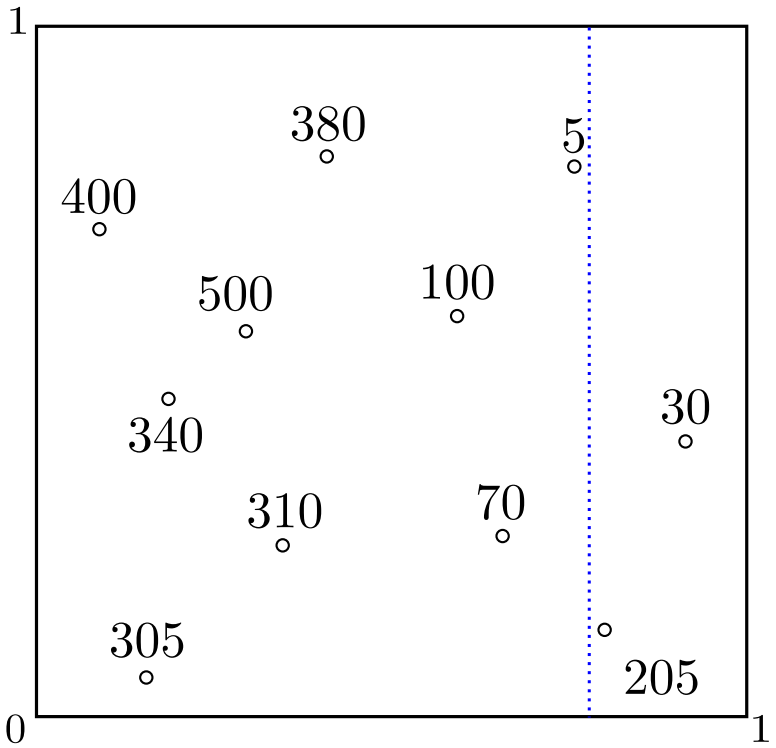
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

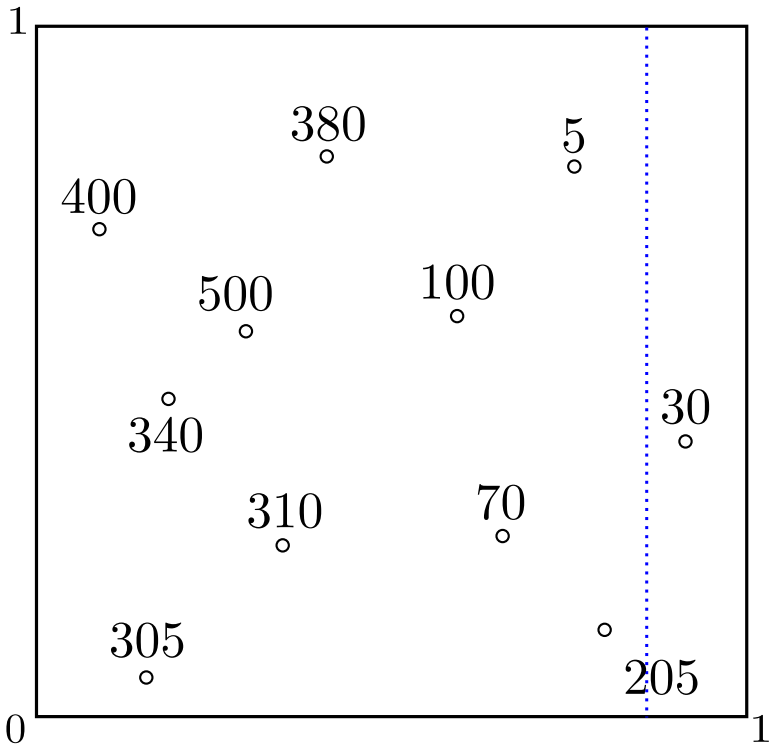
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

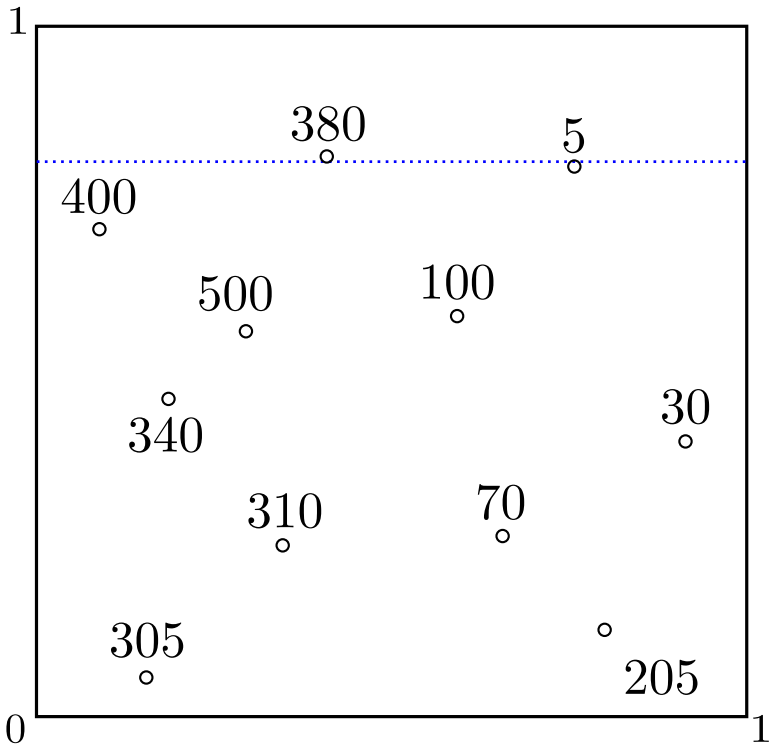
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

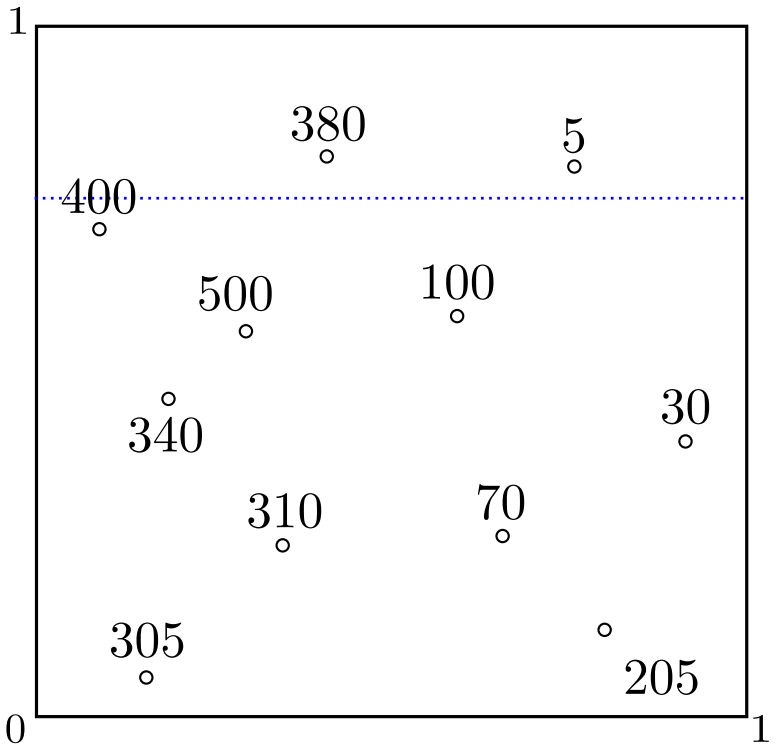
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

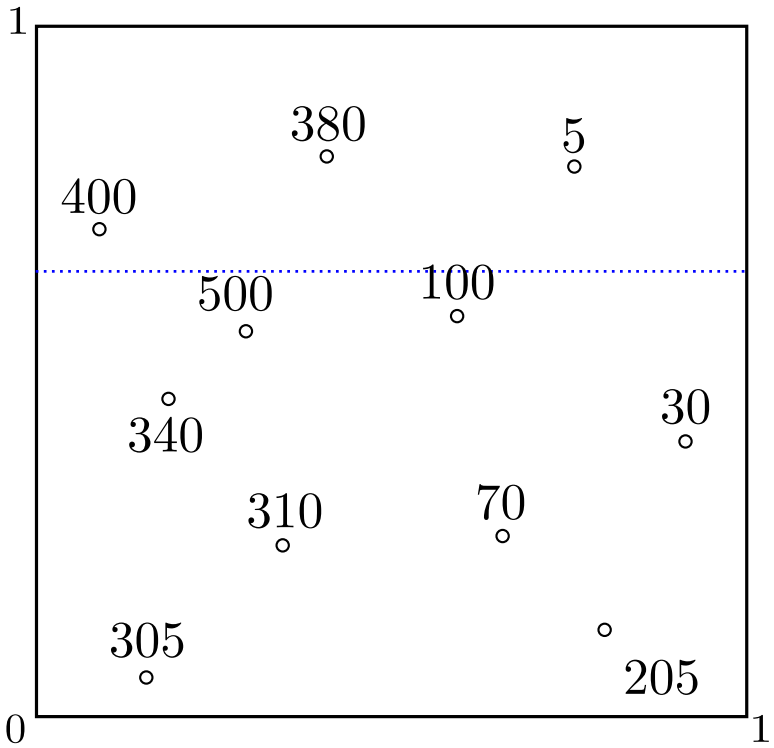
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

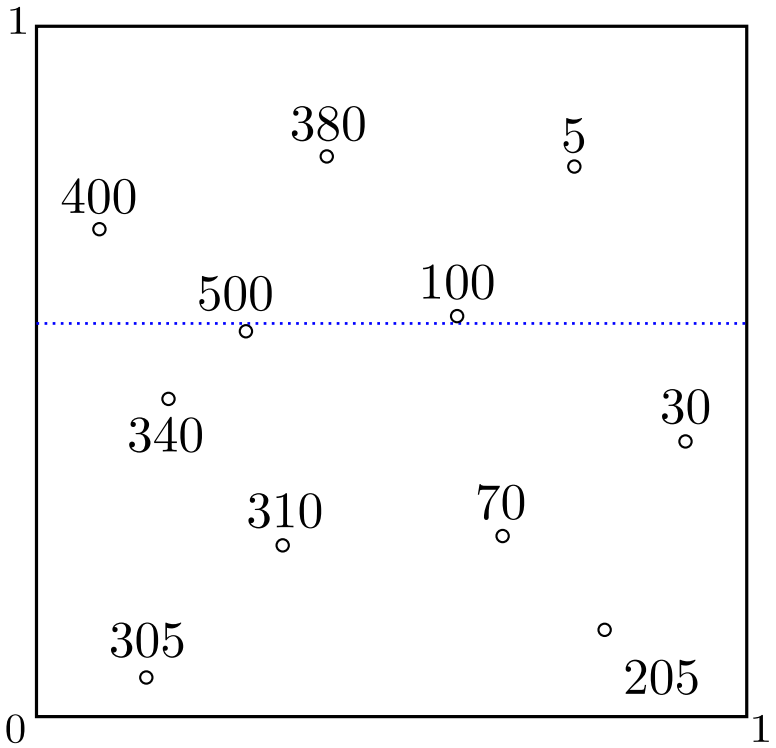
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

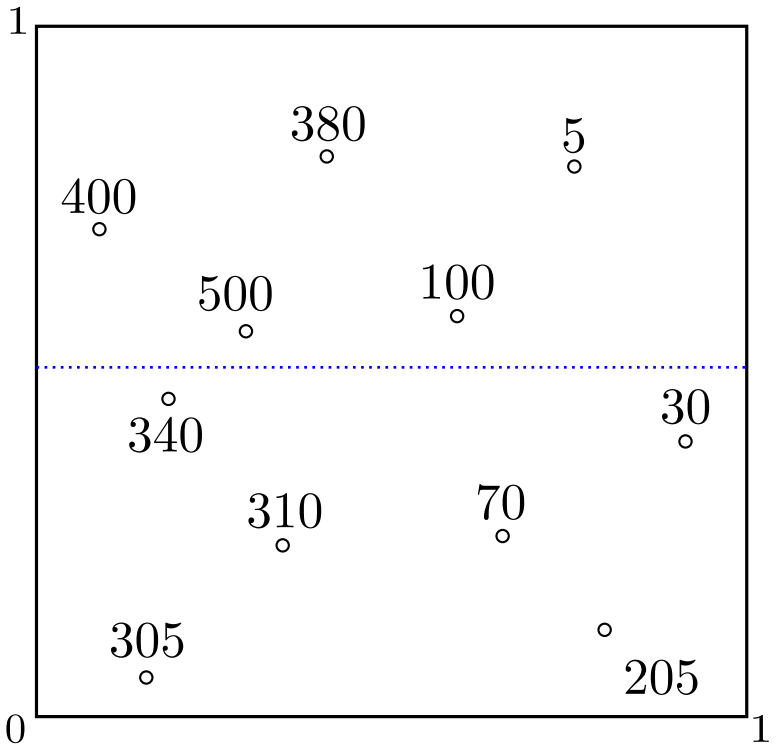
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

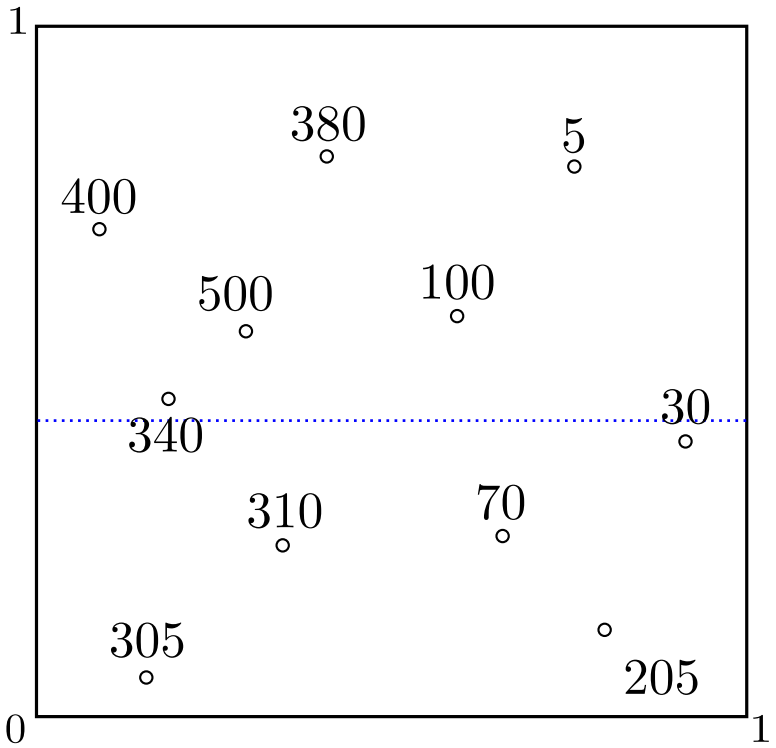
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

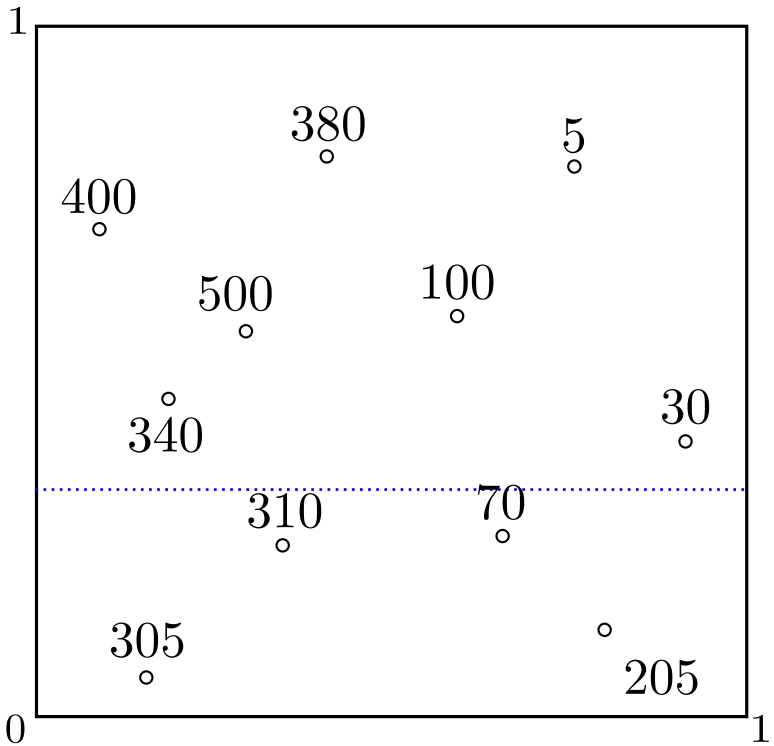
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

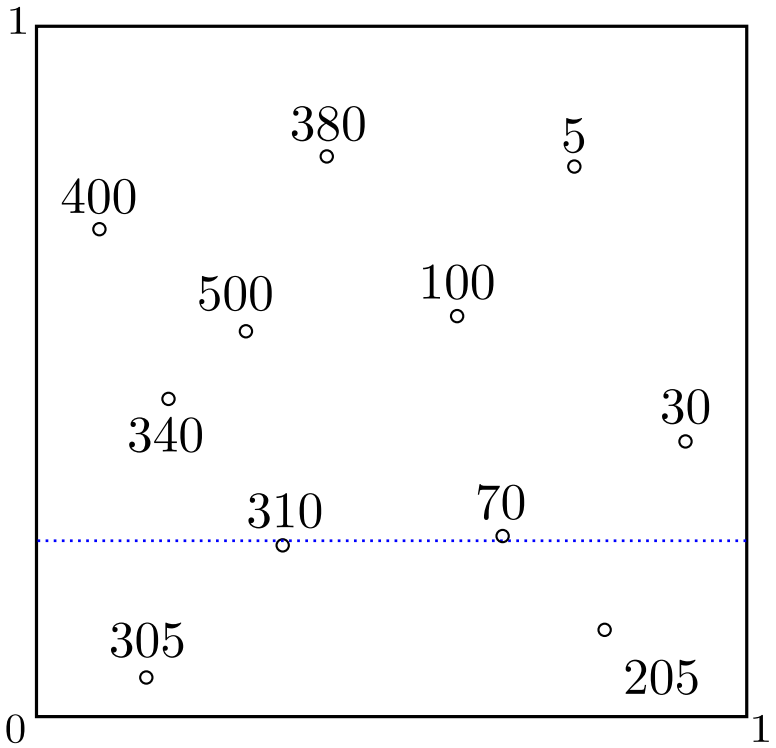
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

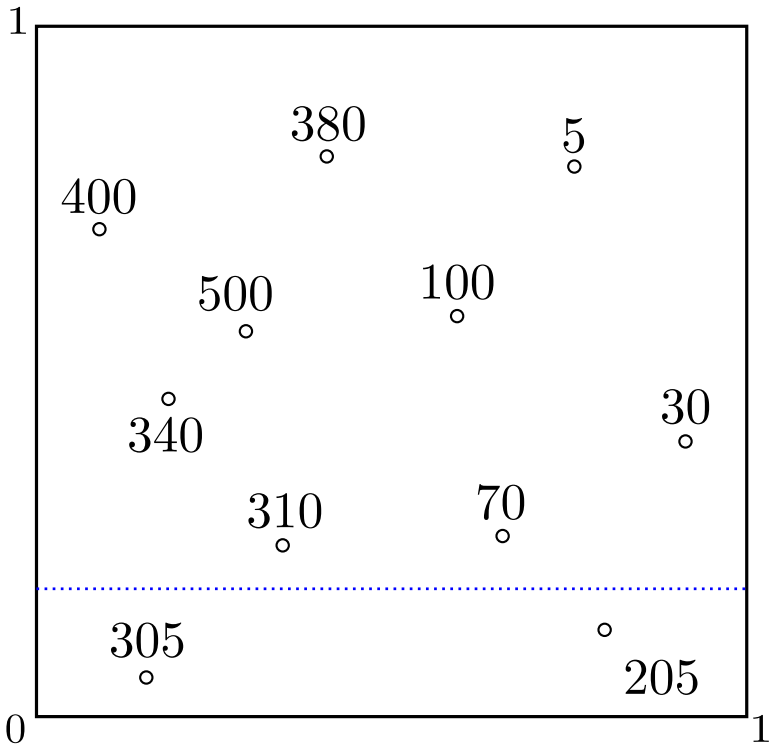
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

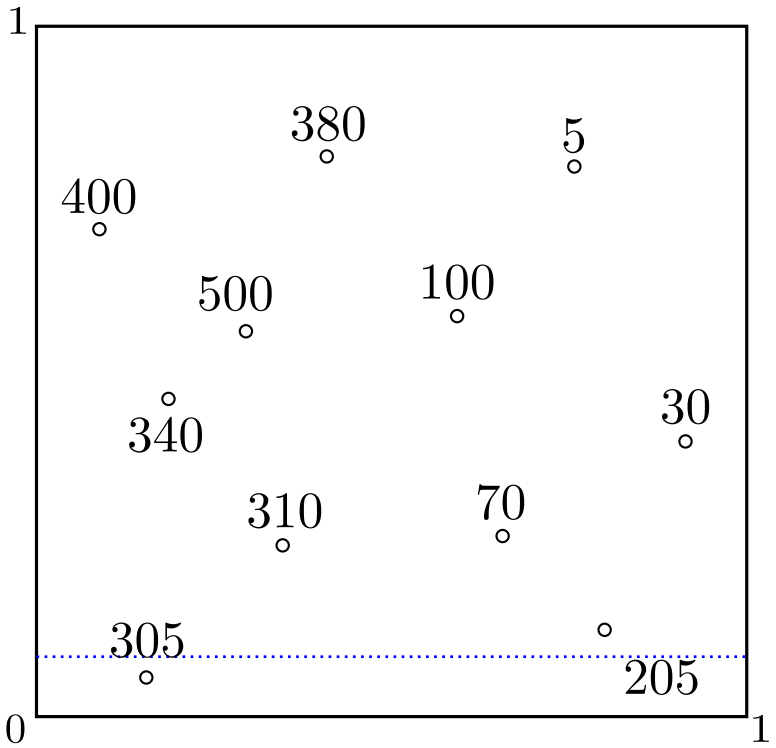
Finding the best split - an example



$k = 0$

- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

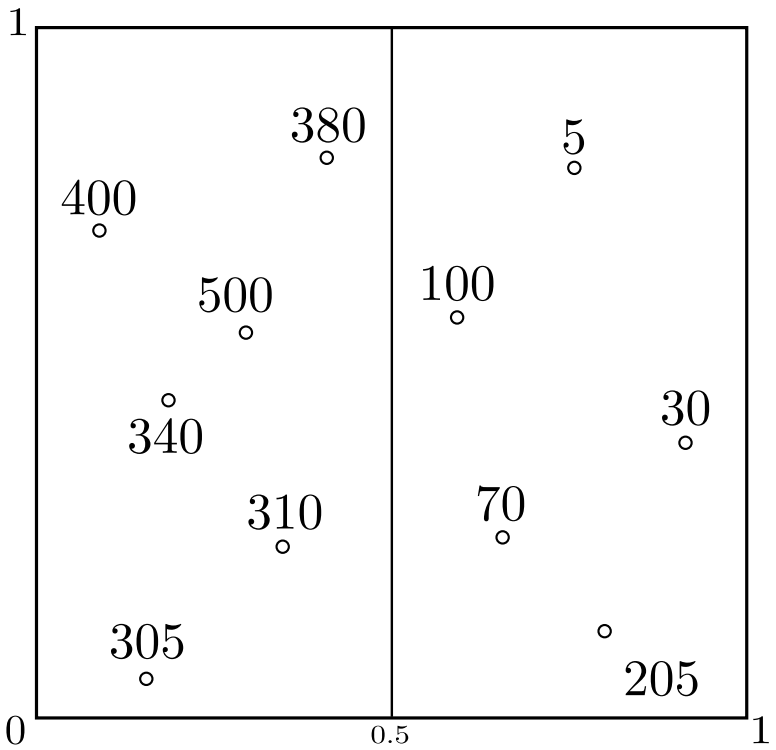
Finding the best split - an example



$k = 0$

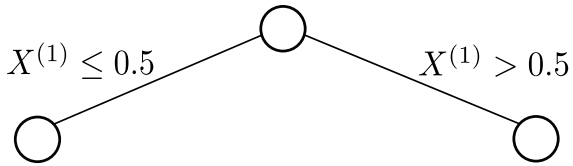
- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.

Finding the best split - an example



$k = 0$

$k = 1$



- Consider splits at the middle of two consecutive observations
- For each split, compute the decrease in impurity between the parent node and the two resulting nodes.
- Select the split maximizing the decrease in impurity

Splitting criteria

For regression:

- Variance

$$Imp_V(A) = \mathbb{V}_n[Y|X \in A].$$

- Mean absolute deviation around the median

$$Imp_{L_1}(A) = \mathbb{E}_n[|Y - Med(Y|X \in A)||X \in A].$$

For classification, letting $p_{k,n}(A)$ the proportion of observations in A such that $Y = k$:

- Misclassification error rate

$$Imp_{err}(A) = 1 - \max_{1 \leq k \leq K} p_{k,n}(A)$$

- Gini

$$Imp_G(A) = \sum_{k=1}^{K} p_{k,n}(A)(1 - p_{k,n}(A)).$$

- Entropy

$$Imp_H(A) = -\sum_{k=1}^{K} p_{k,n}(A) \log_2(p_{k,n}(A)).$$

General rule. Choose the splitting criterion corresponding to the risk you want to minimize (Variance for MSE, Entropy for cross-entropy)

Classification - which criterion to use?

We can choose between

- Misclassification rate

$$Imp_{err}(A) = 1 - \max_{1 \leq k \leq K} p_{k,n}(A)$$

- Gini

$$Imp_G(A) = \sum_{k=1}^{K} p_{k,n}(A)(1 - p_{k,n}(A)).$$

- Entropy

$$Imp_H(A) = -\sum_{k=1}^{K} p_{k,n}(A) \log_2(p_{k,n}(A)).$$

Classification - which criterion to use?

In a binary classification setting, the criterion can be rewritten as

- Misclassification rate

$$Imp_{err}(A) = 1 - \max_{k \in \{0,1\}} p_{k,n}(A)$$

- Gini

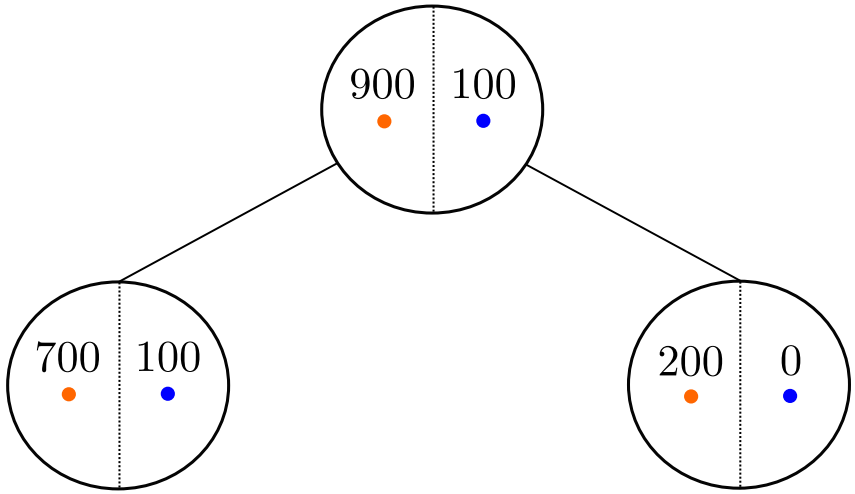
$$Imp_G(A) = 2p_{0,n}(A)(1 - p_{0,n}(A))$$

- Entropy

$$Imp_H(A) = -p_{0,n}(A)\log_2(p_{0,n}(A)) - (1 - p_{0,n}(A))\log_2(1 - p_{0,n}(A))$$

Classification - which criterion to use?

Let us take an example:



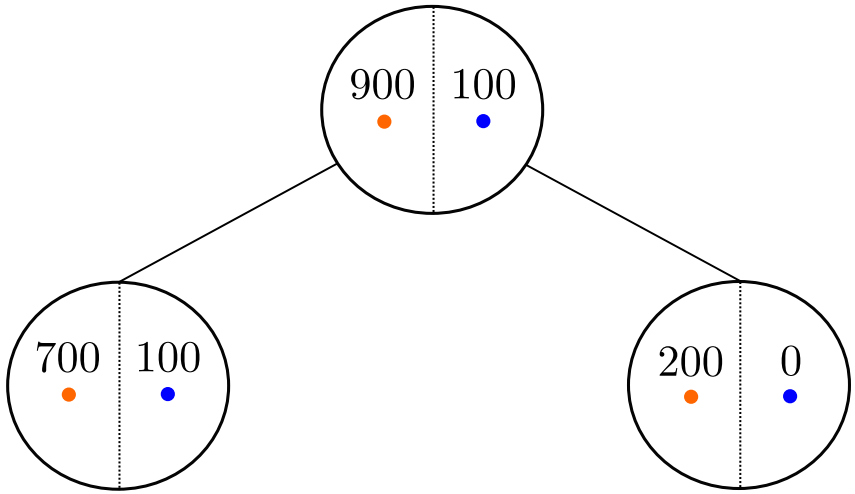
For such a split of the parent cell A, we have

$$Imp_{err}(A) = Imp_{err}(A_L) = Imp_{err}(A_R) = 0.1,$$

which leads us to believe that the split is uninformative since $\Delta Imp_{err} = 0$.

Classification - which criterion to use?

Let us take an example:



For such a split of the parent cell A, we have

$$Imp_{err}(A) = Imp_{err}(A_L) = Imp_{err}(A_R) = 0.1,$$

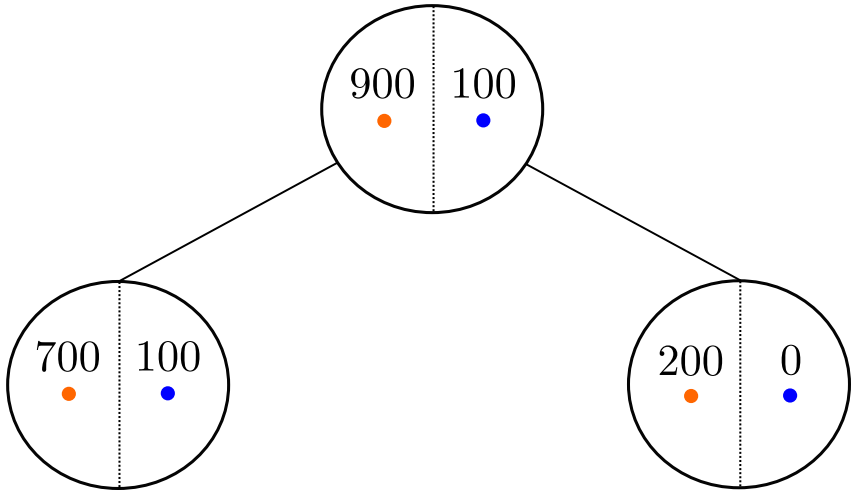
which leads us to believe that the split is uninformative since $\Delta Imp_{err} = 0$.

But the right node is pure! The decrease in impurity for the two other criterion is

$$\Delta Imp_H(A) = 0.01 \quad \text{and} \quad \Delta Imp_G(A) = 0.005.$$

Classification - which criterion to use?

Let us take an example:



For such a split of the parent cell A, we have

$$Imp_{err}(A) = Imp_{err}(A_L) = Imp_{err}(A_R) = 0.1,$$

which leads us to believe that the split is uninformative since $\Delta Imp_{err} = 0$.

This phenomenon results from the fact that the misclassification rate in the binary setting is not strictly concave, contrary to the Entrope/Gini criterion. More explanation here[1]

Misclassification criterion is not precise enough to be used for building trees.

[1] https://tushaargvs.github.io/assets/teaching/dt-notes-2020.pdf

Handling discrete features

There exist three main types of features:

- Continuous (blood pressure)
- Ordinal (Glasgow score)
- Nominal (Medical treatments)

Handling discrete features

There exist three main types of features:

- Continuous (blood pressure)
- Ordinal (Glasgow score)
- Nominal (Medical treatments)

Continuous features. The tree above was built on continuous features: splits are of the form $X^{(j)} \leq s$.

Handling discrete features

There exist three main types of features:

- Continuous (blood pressure)
- Ordinal (Glasgow score)
- Nominal (Medical treatments)

Continuous features. The tree above was built on continuous features: splits are of the form $X^{(j)} \leq s$.

Ordinal features. Construction can be directly extended to ordinal features: splits are exactly of the same form $X^{(j)} \leq s$.

Handling discrete features

There exist three main types of features:

- Continuous (blood pressure)
- Ordinal (Glasgow score)
- Nominal (Medical treatments)

Continuous features. The tree above was built on continuous features: splits are of the form $X^{(j)} \leq s$.

Ordinal features. Construction can be directly extended to ordinal features: splits are exactly of the same form $X^{(j)} \leq s$.

Nominal features. For nominal feature, it makes no sense to consider such splits: there is no natural order on treatments. We can use one-hot encoding instead.

Handling discrete features

There exist three main types of features:

- Continuous (blood pressure)
- Ordinal (Glasgow score)
- Nominal (Medical treatments)

Continuous features. The tree above was built on continuous features: splits are of the form $X^{(j)} \leq s$.

Ordinal features. Construction can be directly extended to ordinal features: splits are exactly of the same form $X^{(j)} \leq s$.

Nominal features. For nominal feature, it makes no sense to consider such splits: there is no natural order on treatments. We can use one-hot encoding instead.
One-hot encoding: Consists in creating as many new variables as the original one contains modalities.

- Advantage: can model splits of any types - increase approximation capacity of decision trees
- Drawback: exploding number of variables - expensive computations and requires large sample sizes.

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

- Choose one of the following splitting criteria: misclassification rate, Entropy or Gini

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

- Choose one of the following splitting criteria: misclassification rate, Entropy or Gini
- Consider a nominal variable X_j that can take L modalities. Reorder it so that the empirical probabilities in a given cell A satisfy

$$\mathbb{P}_n[Y = 1|X_j = C_1, X \in A] \leq \mathbb{P}_n[Y = 1|X_j = C_2, X \in A] \leq \ldots \leq \mathbb{P}_n[Y = 1|X_j = C_L, X \in A].$$

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

- Choose one of the following splitting criteria: misclassification rate, Entropy or Gini
- Consider a nominal variable X_j that can take L modalities. Reorder it so that the empirical probabilities in a given cell A satisfy

$$\mathbb{P}_n[Y = 1|X_j = C_1, X \in A] \leq \mathbb{P}_n[Y = 1|X_j = C_2, X \in A] \leq \ldots \leq \mathbb{P}_n[Y = 1|X_j = C_L, X \in A].$$

- Then the best split (that maximizes the decrease in impurity) is of the form

$$X_j \in \{C_1, \ldots, C_\ell\} \quad \text{vs} \quad X_j \in \{C_{\ell+1}, \ldots, C_L\}. \tag{1}$$

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

- Choose one of the following splitting criteria: misclassification rate, Entropy or Gini
- Consider a nominal variable X_j that can take L modalities. Reorder it so that the empirical probabilities in a given cell A satisfy

$$\mathbb{P}_n[Y = 1|X_j = C_1, X \in A] \leq \mathbb{P}_n[Y = 1|X_j = C_2, X \in A] \leq \ldots \leq \mathbb{P}_n[Y = 1|X_j = C_L, X \in A].$$

- Then the best split (that maximizes the decrease in impurity) is of the form

$$X_j \in \{C_1, \ldots, C_\ell\} \quad \text{vs} \quad X_j \in \{C_{\ell+1}, \ldots, C_L\}. \tag{1}$$

This is a result from Fisher, 1958

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

- Choose one of the following splitting criteria: misclassification rate, Entropy or Gini
- Consider a nominal variable X_j that can take L modalities. Reorder it so that the empirical probabilities in a given cell A satisfy

$$\mathbb{P}_n[Y = 1|X_j = C_1, X \in A] \leq \mathbb{P}_n[Y = 1|X_j = C_2, X \in A] \leq \ldots \leq \mathbb{P}_n[Y = 1|X_j = C_L, X \in A].$$

- Then the best split (that maximizes the decrease in impurity) is of the form

$$X_j \in \{C_1, \ldots, C_\ell\} \quad \text{vs} \quad X_j \in \{C_{\ell+1}, \ldots, C_L\}. \tag{1}$$

This is a result from Fisher, 1958

Summary. Instead of evaluating the splitting criterion on $2^{L-1} - 1$ splits, we just need to compute it on $L - 1$ splits!

A particular case: nominal features for binary classification

Nominal variables. A classic way to handle them is via one-hot encoding. Sadly, the number of dummy variables explodes.

In **binary classification**, we can do better.

- Choose one of the following splitting criteria: misclassification rate, Entropy or Gini
- Consider a nominal variable X_j that can take L modalities. Reorder it so that the empirical probabilities in a given cell A satisfy

$$\mathbb{P}_n[Y = 1|X_j = C_1, X \in A] \leq \mathbb{P}_n[Y = 1|X_j = C_2, X \in A] \leq \ldots \leq \mathbb{P}_n[Y = 1|X_j = C_L, X \in A].$$

- Then the best split (that maximizes the decrease in impurity) is of the form

$$X_j \in \{C_1, \ldots, C_\ell\} \quad \text{vs} \quad X_j \in \{C_{\ell+1}, \ldots, C_L\}. \tag{1}$$

This is a result from Fisher, 1958

Summary. Instead of evaluating the splitting criterion on $2^{L-1} - 1$ splits, we just need to compute it on $L - 1$ splits!

Extension to regression. The same procedure holds in regression by considering the average values of Y for each modality (instead of the probability in (1)).

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule

Stopping rule for splitting a cell:

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule

Stopping rule for splitting a cell:

- All samples have the same label (classification)

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule

Stopping rule for splitting a cell:

- All samples have the same label (classification)
- No reduction of the impurity criterion

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule

Stopping rule for splitting a cell:

- All samples have the same label (classification)
- No reduction of the impurity criterion
- The next split will produce cells with less than `min-samples-leaf` observations (1, by default)

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule

Stopping rule for splitting a cell:

- All samples have the same label (classification)
- No reduction of the impurity criterion
- The next split will produce cells with less than `min-samples-leaf` observations (1, by default)
- The cell contains less than `min-samples-split` observations (2, by default)

Decision tree

Now that we have defined a splitting rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule

Stopping rule for splitting a cell:

- All samples have the same label (classification)
- No reduction of the impurity criterion
- The next split will produce cells with less than `min-samples-leaf` observations (1, by default)
- The cell contains less than `min-samples-split` observations (2, by default)
- The cell has already been split `max-depth` times (∞, by default)

Decision tree

Now that we have defined a splitting and a stopping rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule (by default, one observation per leaf)

Decision tree

Now that we have defined a splitting and a stopping rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule (by default, one observation per leaf)
- Prediction rule

Prediction rule:

Decision tree

Now that we have defined a splitting and a stopping rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule (by default, one observation per leaf)
- Prediction rule

Prediction rule:

- Regression - Average of labels per leaf

$$\hat{t}_n(x_{new}) = \sum_{i=1}^{n} Y_i \frac{\mathbb{1}_{X_i \in A_n(x_{new})}}{N_n(A_n(x_{new}))}$$

Decision tree

Now that we have defined a splitting and a stopping rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule (by default, one observation per leaf)
- Prediction rule

Prediction rule:

- Regression - Average of labels per leaf

$$\hat{t}_n(x_{new}) = \sum_{i=1}^{n} Y_i \frac{\mathbb{1}_{X_i \in A_n(x_{new})}}{N_n(A_n(x_{new}))}$$

- Classification - Majority vote per leaf

$$\hat{t}_n(x_{new}) = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \sum_{i=1}^{n} \frac{\mathbb{1}_{Y_i = k} \mathbb{1}_{X_i \in A_n(x_{new})}}{N_n(A_n(x_{new}))}$$

Decision tree

Now that we have defined a splitting and a stopping rule, let us see the rest of the tree construction.

Decision tree building

- Splitting rule (Variance in regression, Gini or Entropy in classification)
- Stopping rule (by default, one observation per leaf)
- Prediction rule (average or majority vote per leaf)

Tree ensemble methods

Consist in aggregating the predictions of several decision trees:

- More robust than individual trees to changes in data
- More flexible methods / useful for modelling complex input-output relations

How to do that?

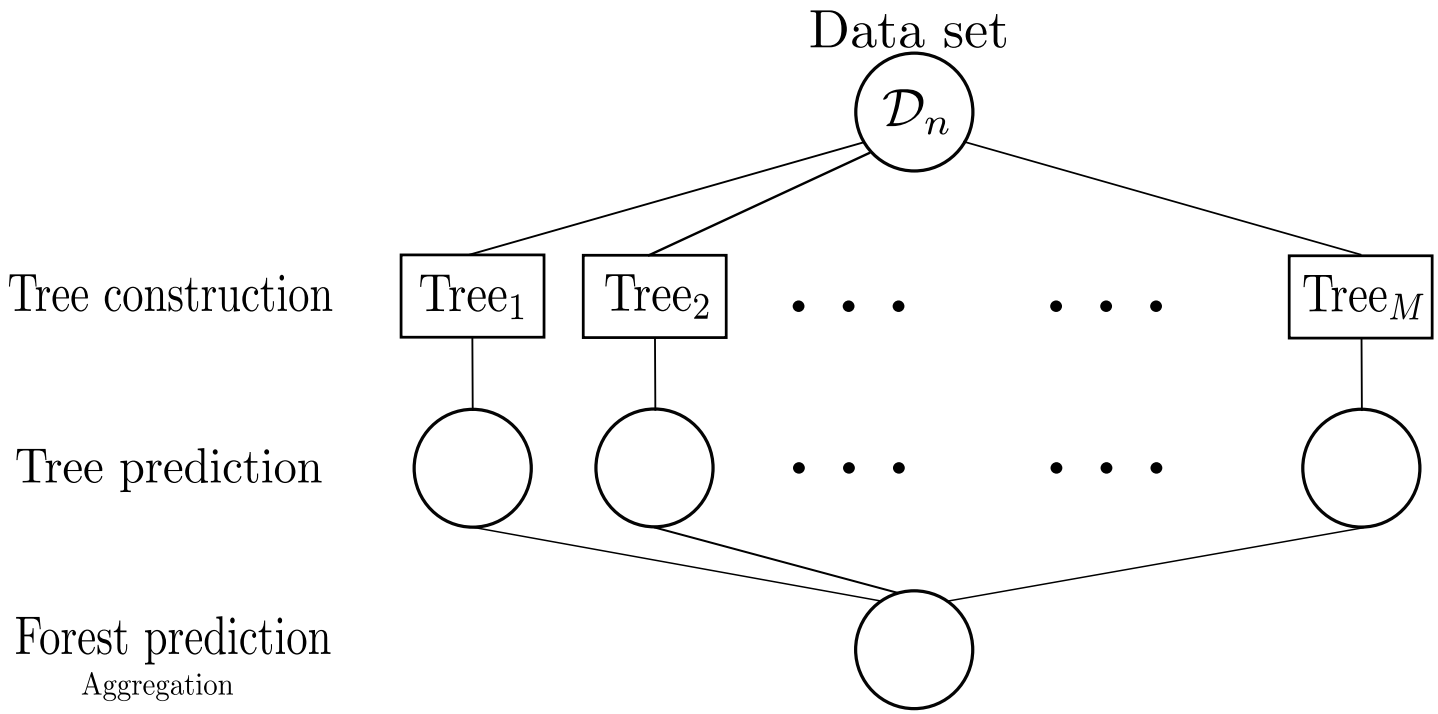
Tree ensemble methods

Consist in aggregating the predictions of several decision trees:

- More robust than individual trees to changes in data
- More flexible methods / useful for modelling complex input-output relations

How to do that?

- Random forests (parallel methods)



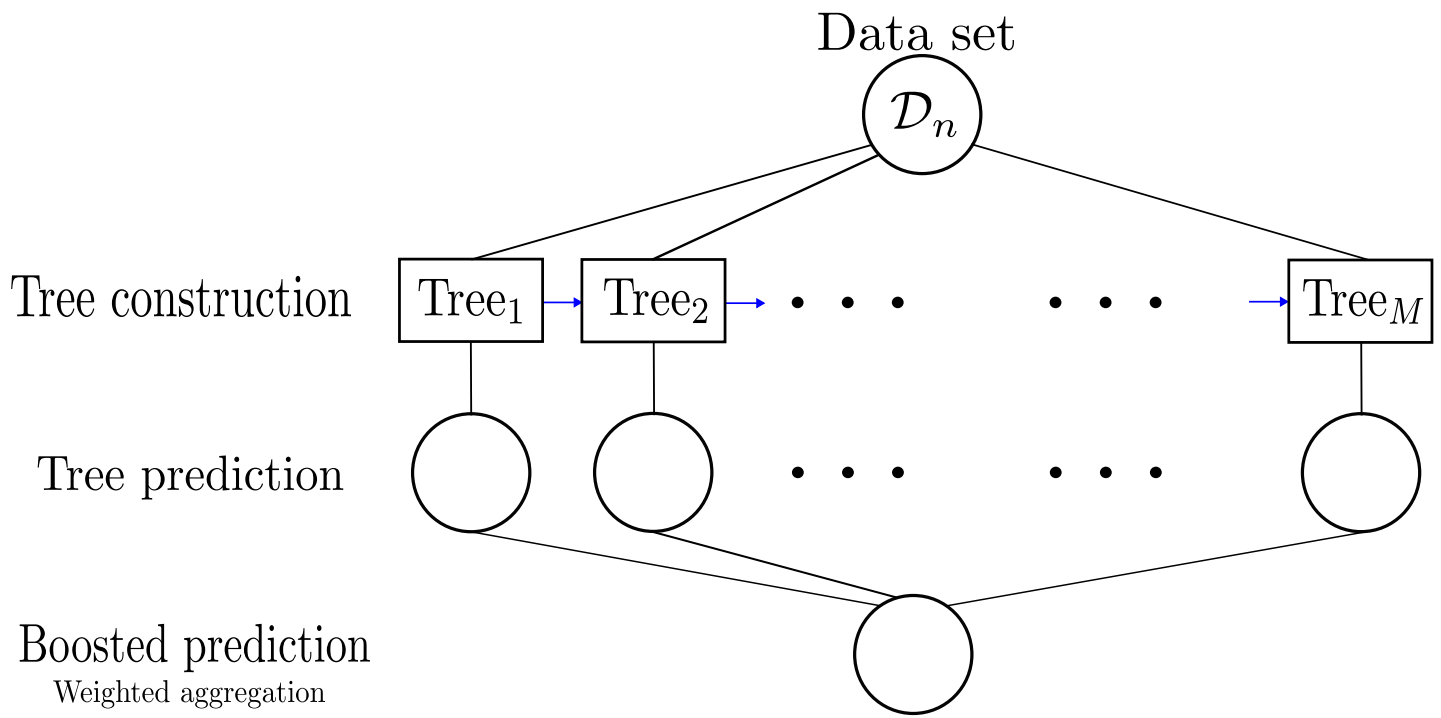
Tree ensemble methods

Consist in aggregating the predictions of several decision trees:

- More robust than individual trees to changes in data
- More flexible methods / useful for modelling complex input-output relations

How to do that?

- Random forests (parallel methods)
- Tree boosting (sequential methods)



Tree construction

Tree construction

- Input: dataset, splitting criterion.
- At each node A, select the best split by optimizing the splitting criterion / difference in impurity

$$(j^\star, s^\star) \in \underset{j \in \{1,\dots,d\}, s \in \text{range}(X^{(j)})}{\text{argmax}} \Delta Imp(j, s; A).$$

- Repeat for each cell until:
  - All labels in the cell are the same
  - No impurity reduction
  - The leaf contains one observation
- Output: a fully-grown decision tree.

Tree pruning

- Input: A fully-grown decision tree, a data set, an impurity measure.
- Choose one of the two pruning strategies:
  - Reduction Error pruning (RE, C4.5)
  - Cost complexity pruning (CART)
- Output: a pruned decision tree.

Tree construction in random forests

Tree construction

- Input: dataset, splitting criterion.
- At each node A, select the best split by optimizing the splitting criterion / difference in impurity

$$(j^\star, s^\star) \in \underset{j \in \{1,\dots,d\}, s \in \text{range}(X^{(j)})}{\text{argmax}} \Delta Imp(j, s; A).$$

- Repeat for each cell until:
  - All labels in the cell are the same
  - No impurity reduction
  - The leaf contains one observation
- Output: a fully-grown decision tree.

Tree pruning

For trees in random forests, no pruning strategy

Tree construction

Tree construction

- Input: dataset, splitting criterion.
- At each node A, select the best split by optimizing the splitting criterion / difference in impurity

$$(j^\star, s^\star) \in \underset{j \in \{1,\dots,d\}, s \in \text{range}(X^{(j)})}{\text{argmax}} \Delta Imp(j, s; A).$$

- Repeat for each cell until:
  - All labels in the cell are the same
  - No impurity reduction
  - The leaf contains one observation
- Output: a fully-grown decision tree.

Tree pruning

For trees in random forests, no pruning strategy

Such trees have a small bias (fully-grown) but a large variance (one point per leaf).
They cannot be used as single estimators!

Bagging - Averaging predictors via data set resampling

Bagging (**B**ootstrap **agg**regat**ing**) consists in running a learning algorithm on mulitple modified data sets to stabilize its performance.

Bagging - Averaging predictors via data set resampling

Bagging (**B**ootstrap **agg**regat**ing**) consists in running a learning algorithm on mulitple modified data sets to stabilize its performance.

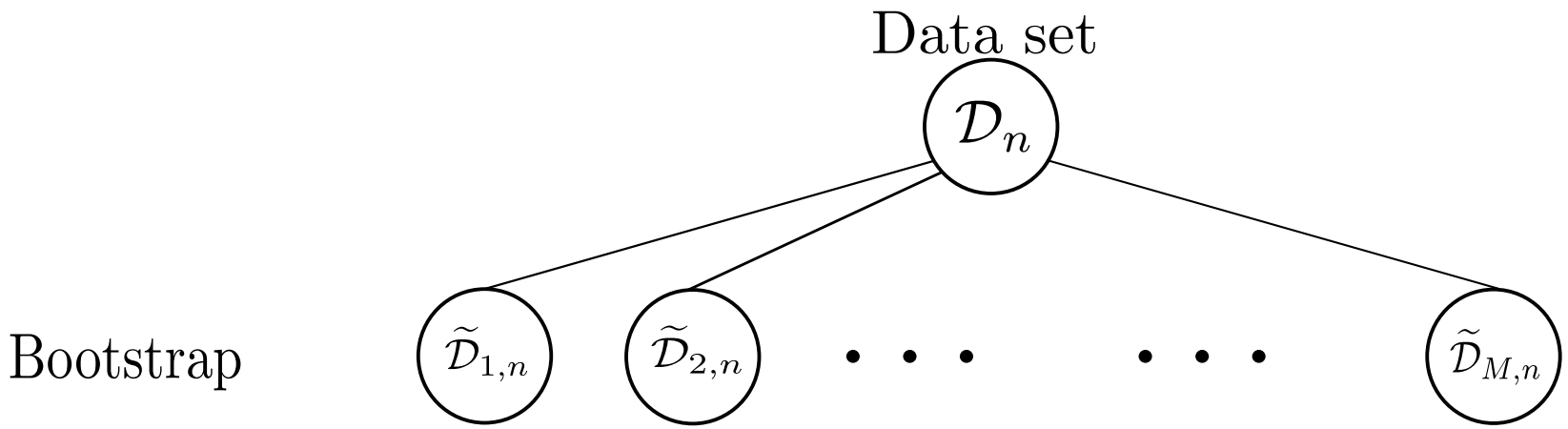
Bootstrap. A sampling scheme that consists in drawing n observations with replacement among the n original ones. Applied once, bootstrap creates one new data set, called a bootstrapped data set.

Bagging - Averaging predictors via data set resampling

Bootstrap. A sampling scheme that consists in drawing n observations with replacement among the n original ones. Applied once, bootstrap creates one new data set, called a bootstrapped data set.

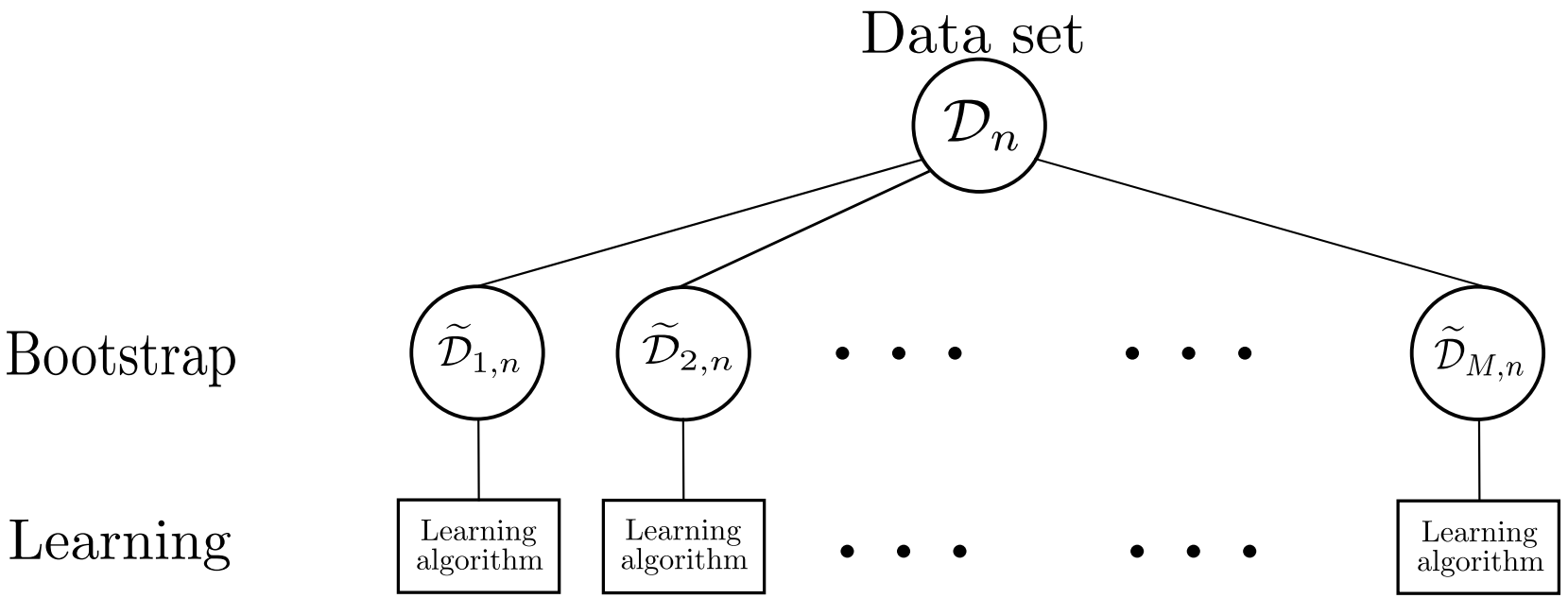
Bagging - Averaging predictors via data set resampling

Bootstrap. A sampling scheme that consists in drawing n observations with replacement among the n original ones. Applied once, bootstrap creates one new data set, called a bootstrapped data set.



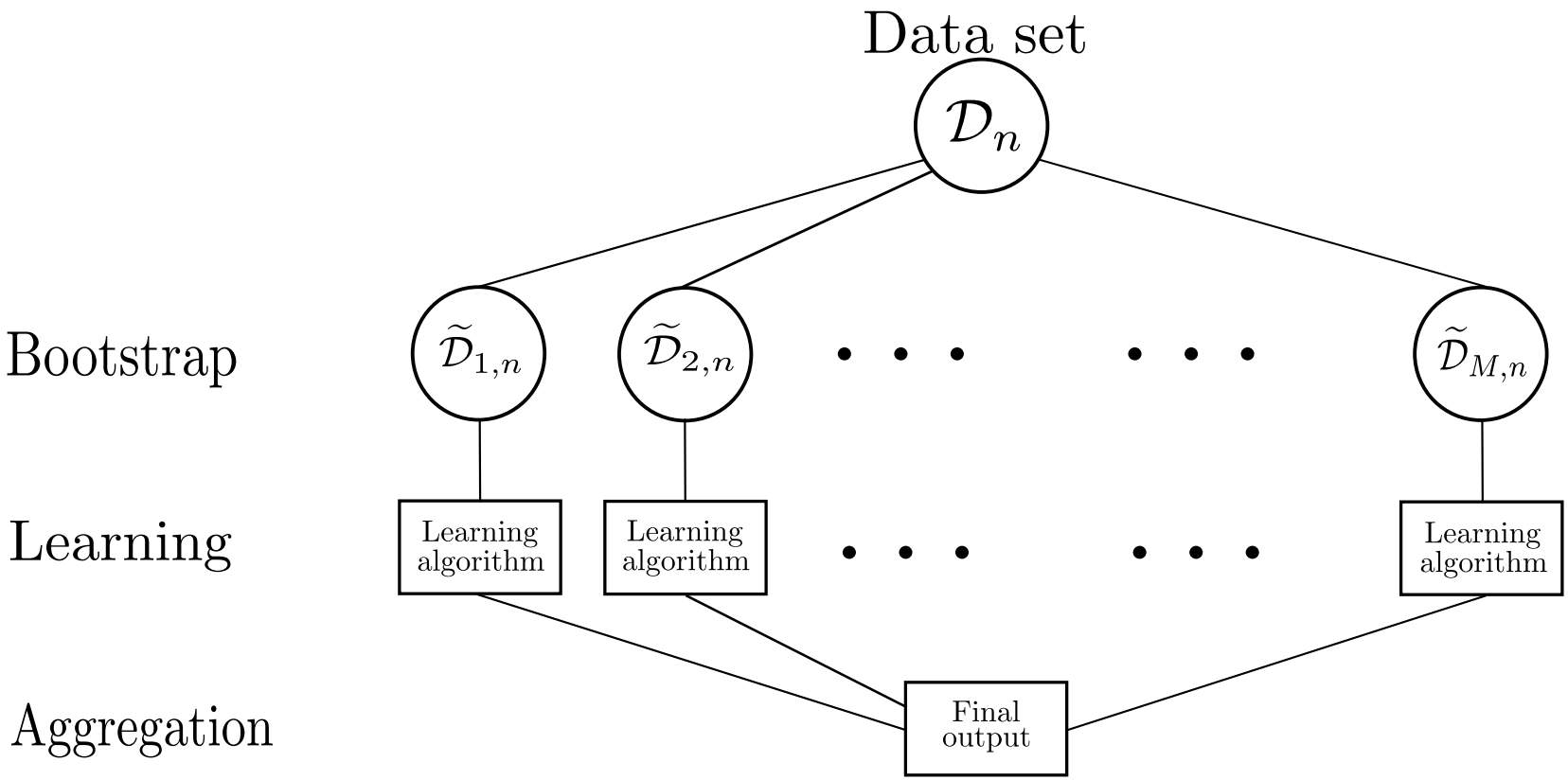
Bagging - Averaging predictors via data set resampling

Bootstrap. A sampling scheme that consists in drawing n observations with replacement among the n original ones. Applied once, bootstrap creates one new data set, called a bootstrapped data set.

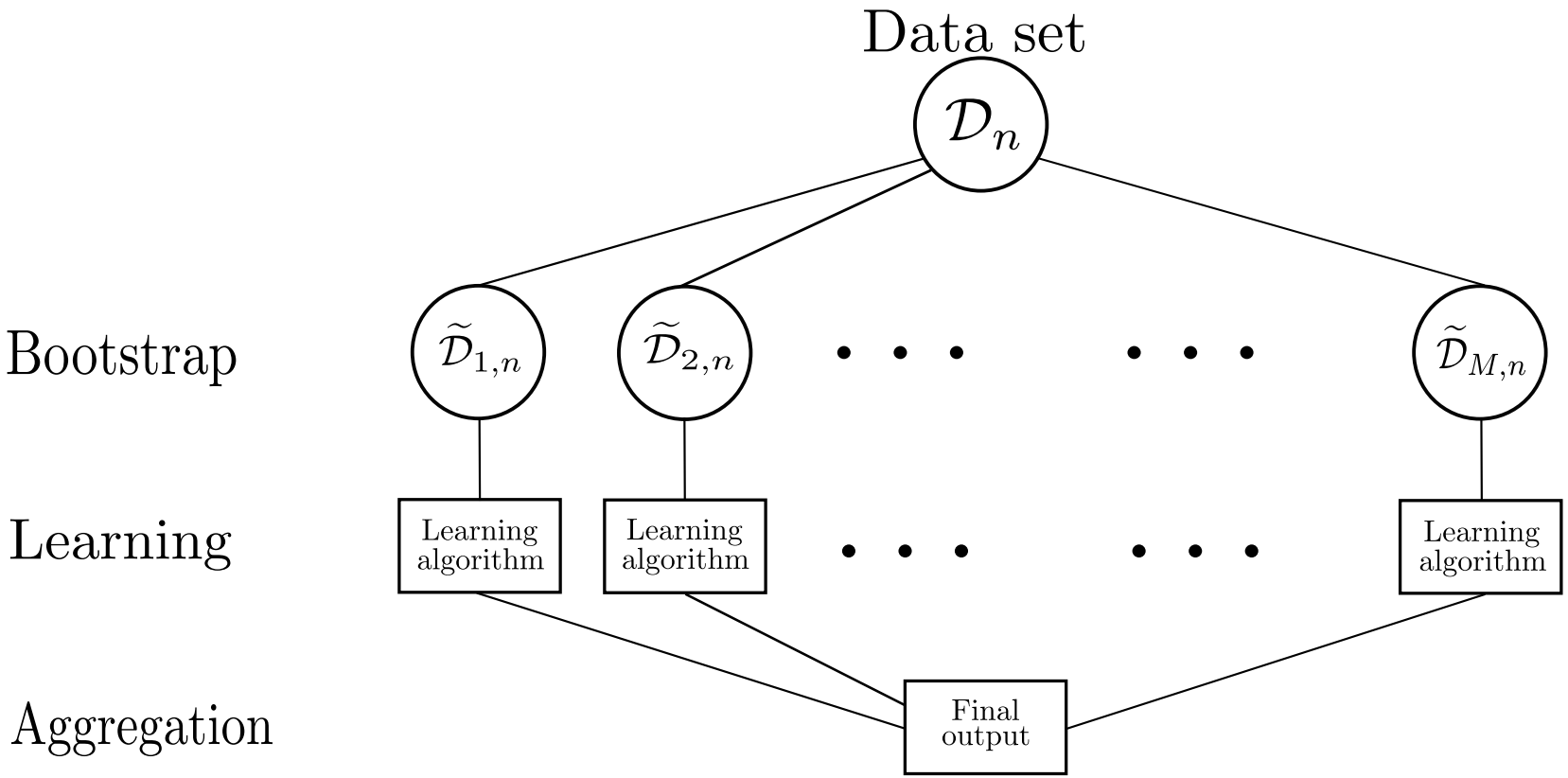


Bagging - Averaging predictors via data set resampling

Bootstrap. A sampling scheme that consists in drawing n observations with replacement among the n original ones. Applied once, bootstrap creates one new data set, called a bootstrapped data set.



Bagging - Averaging predictors via data set resampling



Interests:

- Increase stability - data modification has less impact on the final predictor
- Parallel method - computationnally efficient
- Can be applied to a wide range of learning algorithm (for example, decision trees!)

Inconvenient: individual predictors may be too correlated (built on similar observations).

Split randomization in tree construction

Random forests

Two randomization ingredients:

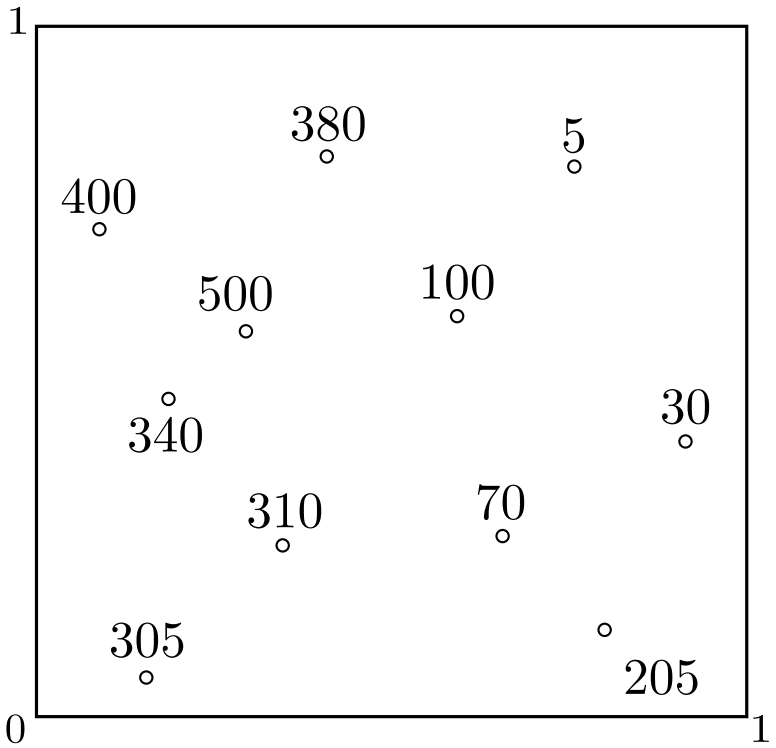
- Bagging
- Split randomization

Split randomization in tree construction

Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.

Split randomization in tree construction

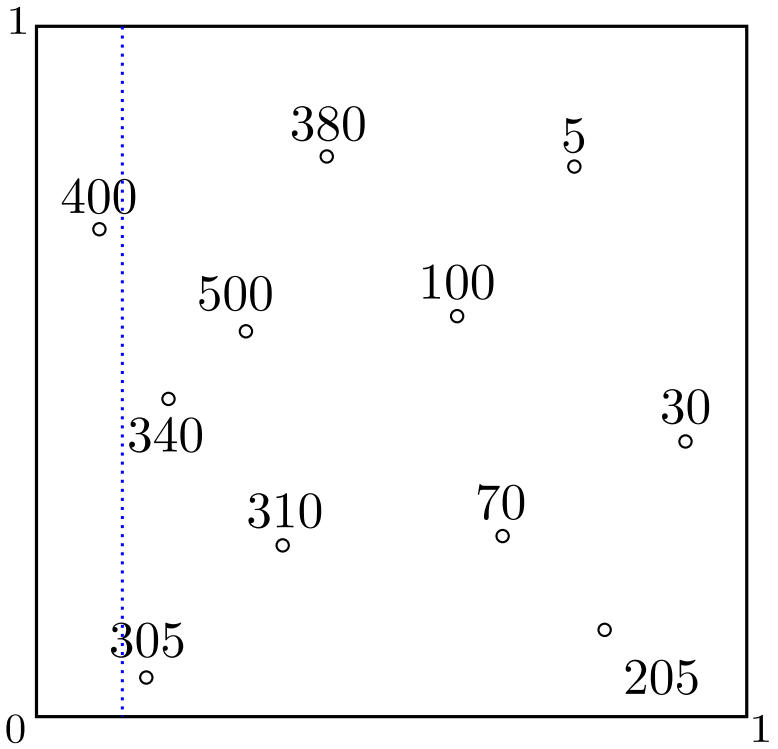
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

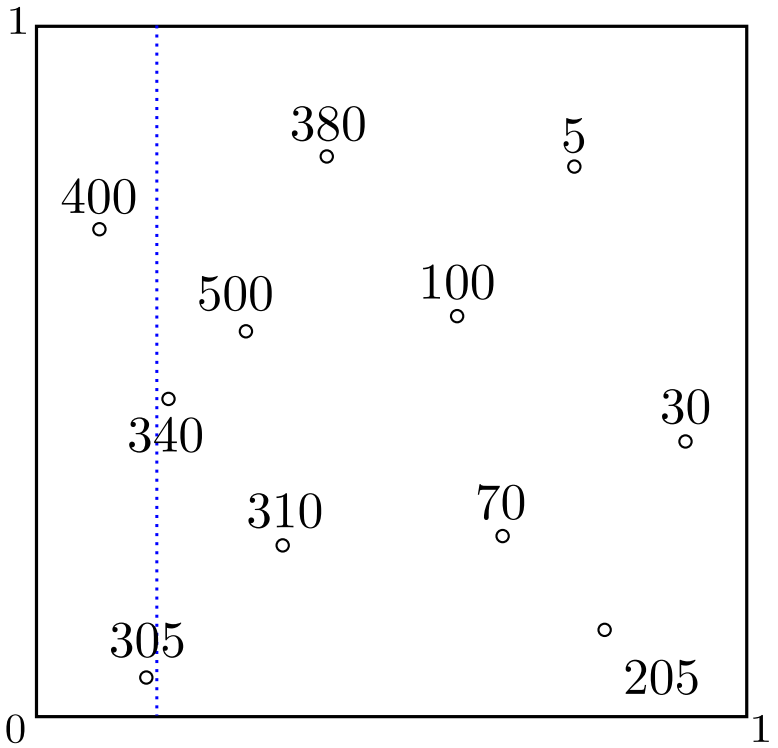
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

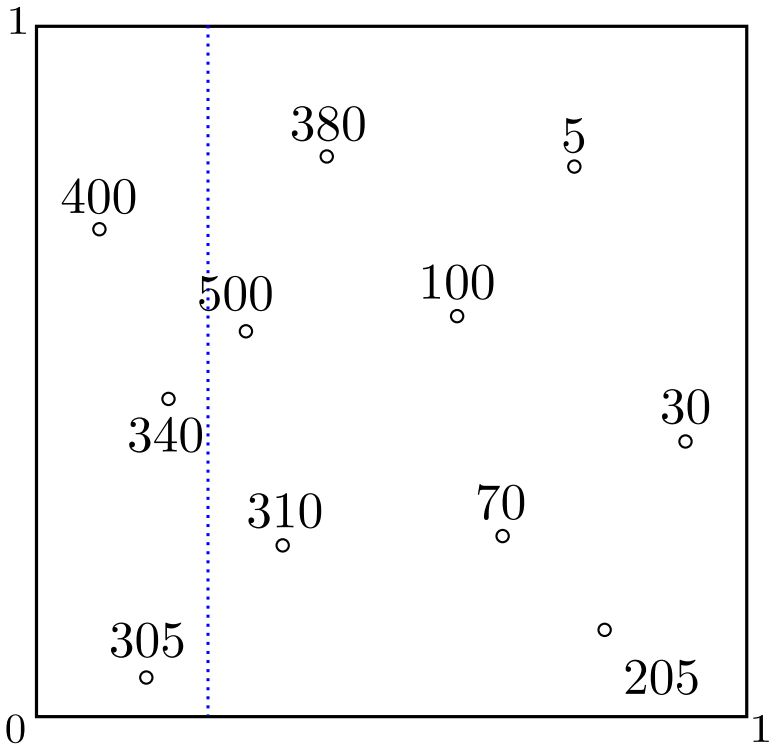
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

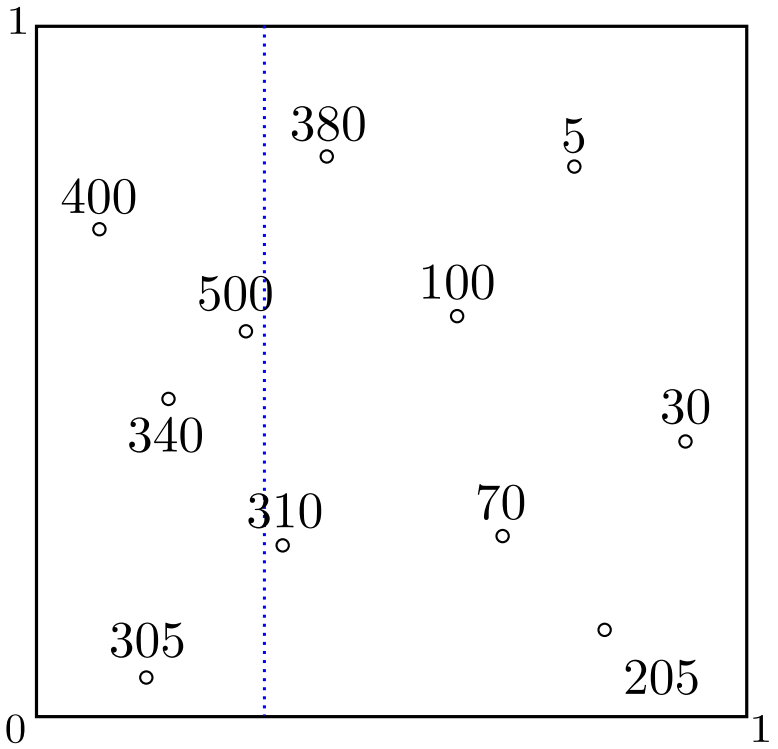
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

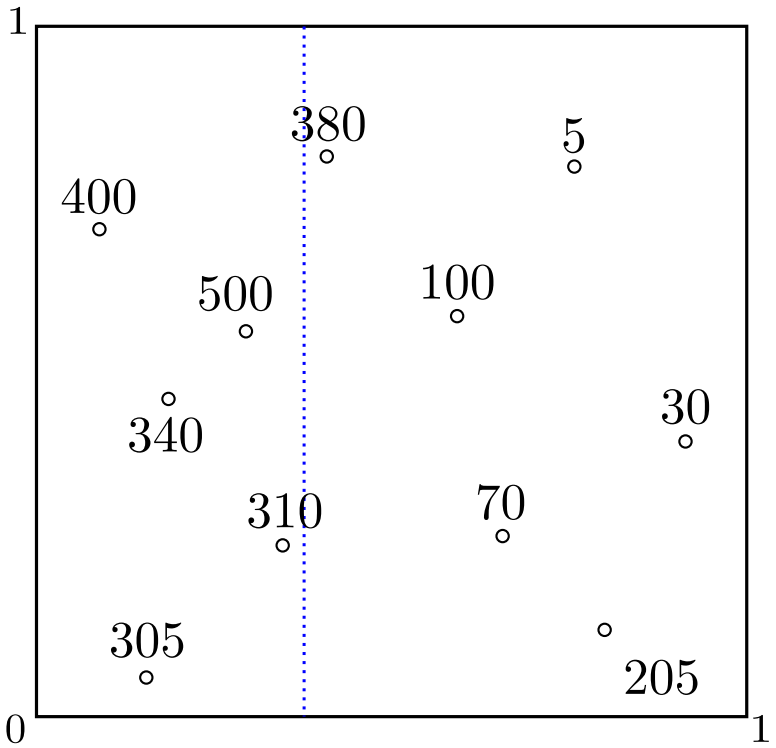
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

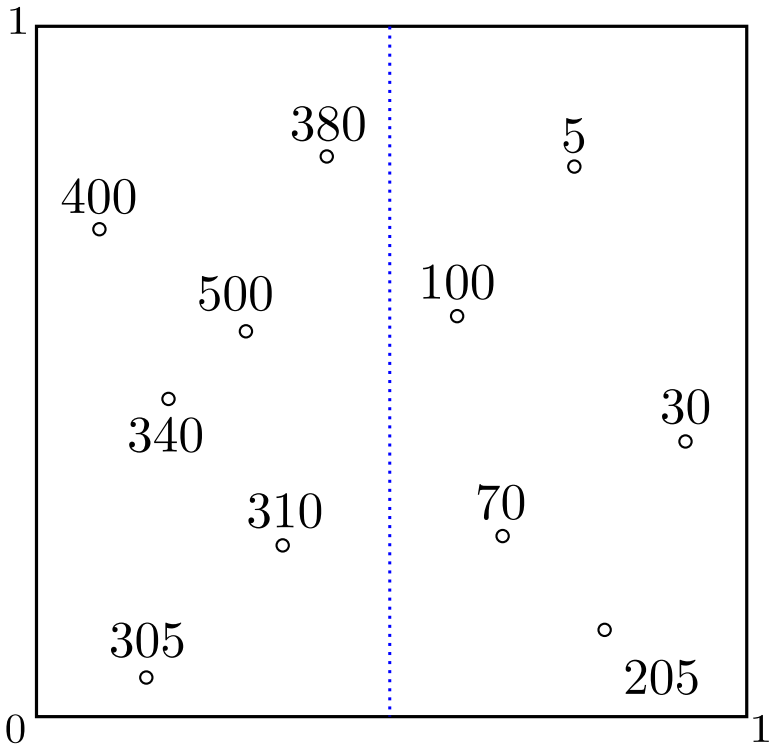
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

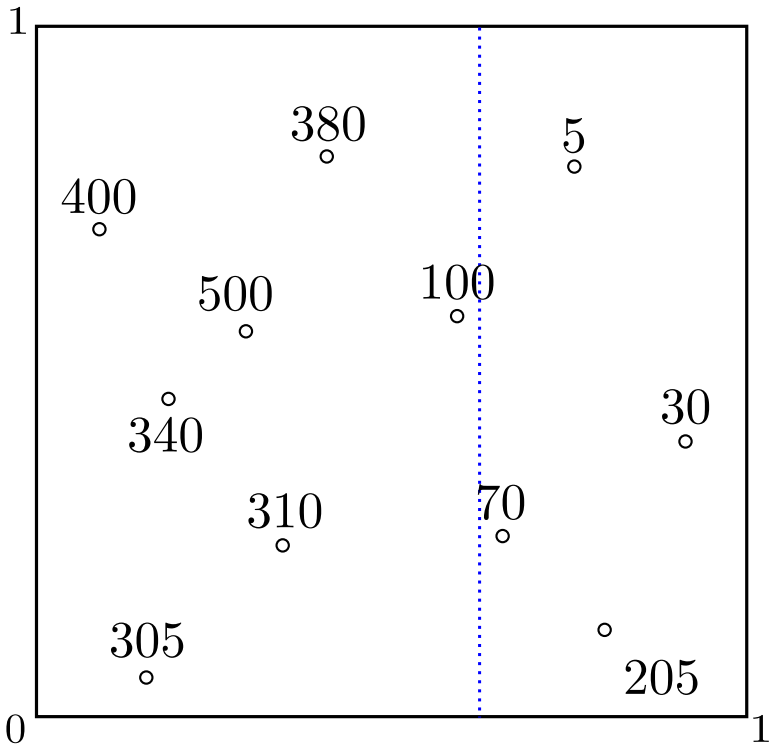
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

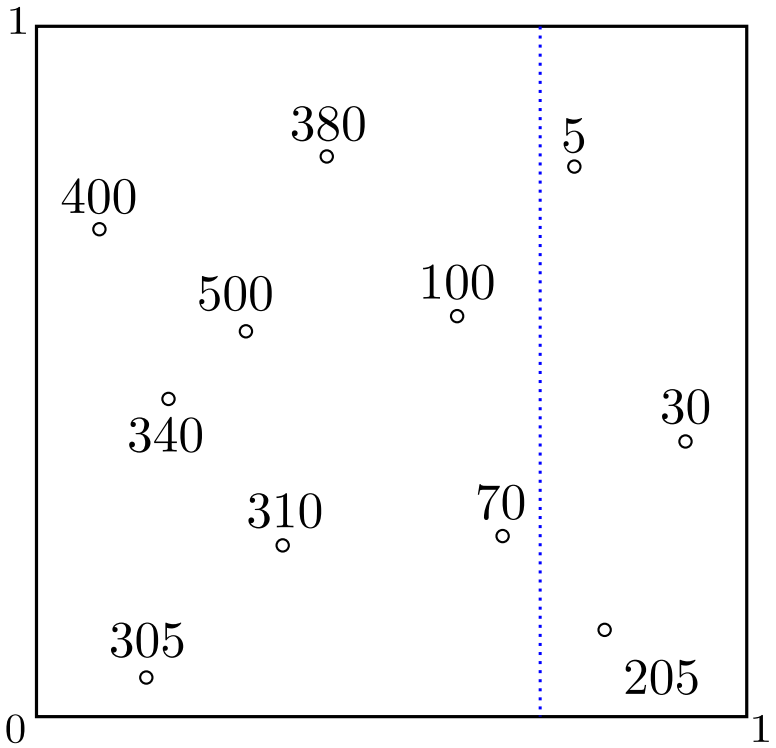
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

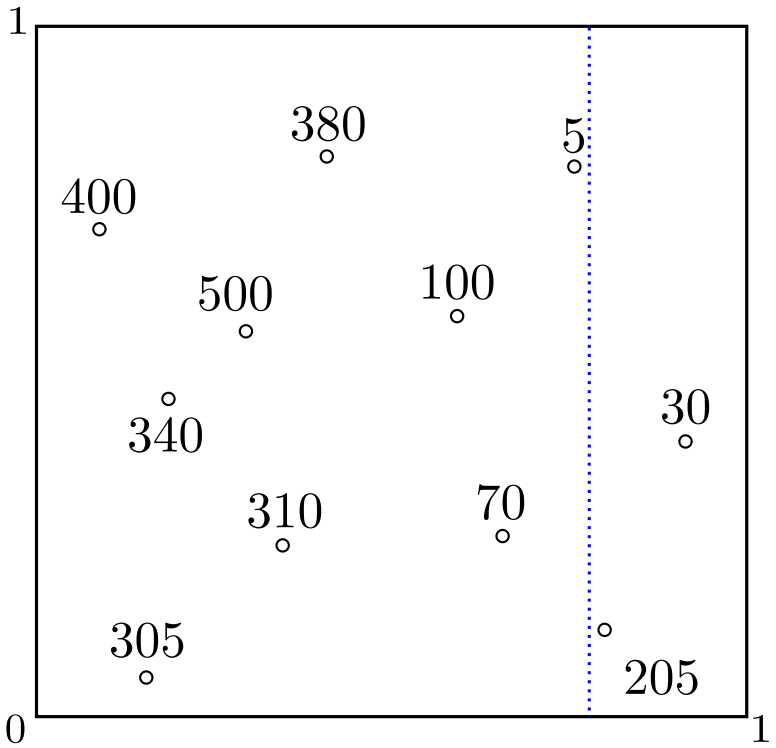
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

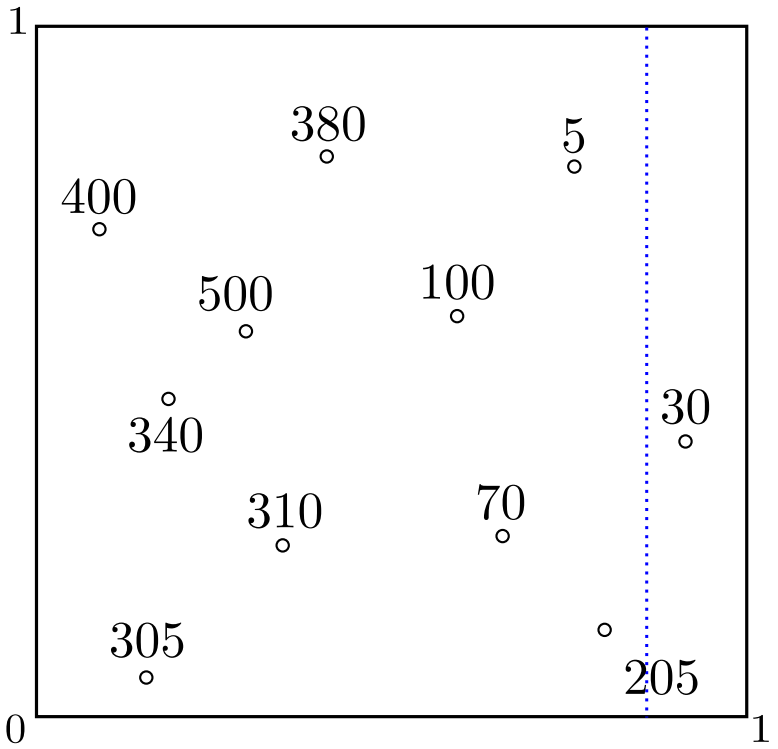
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

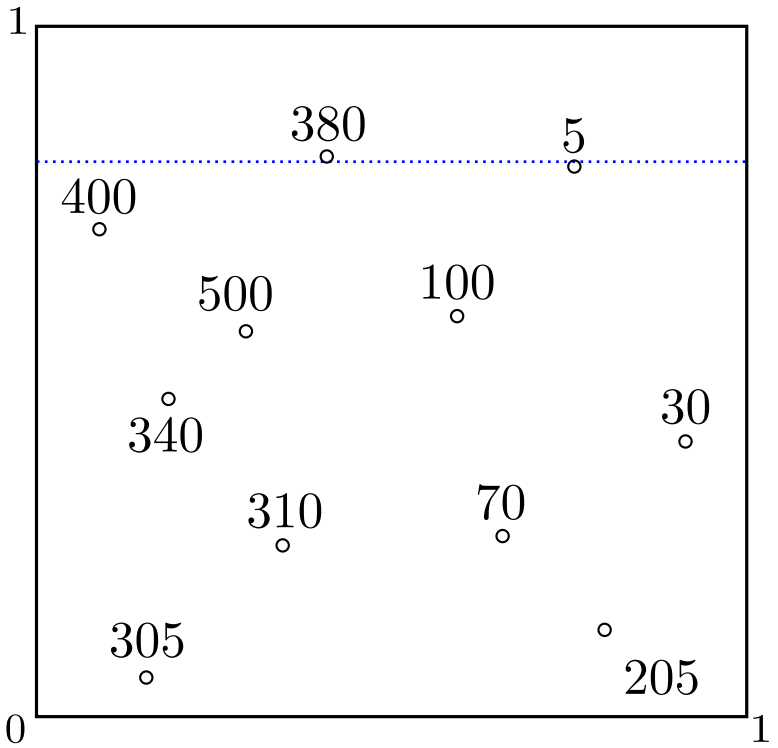
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

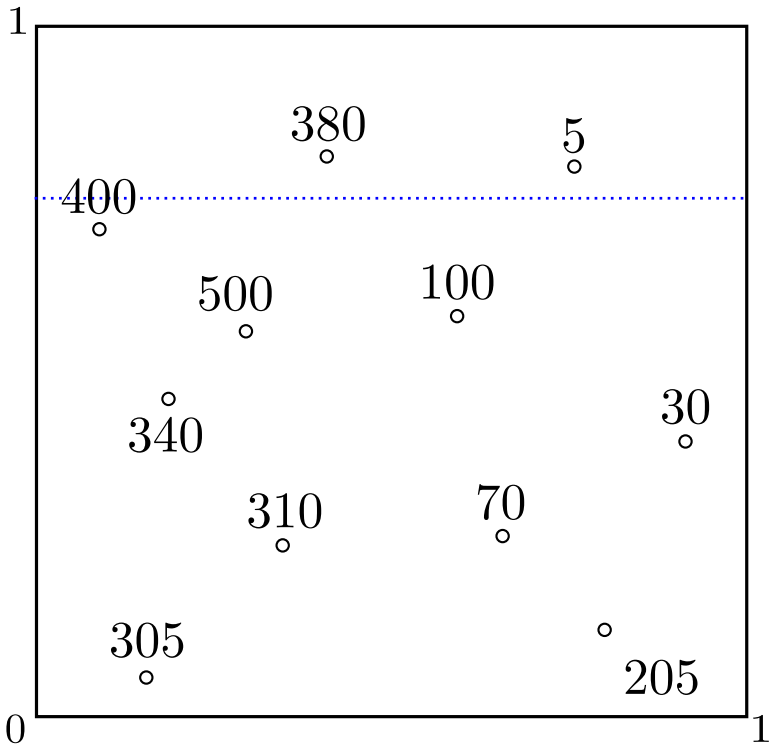
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

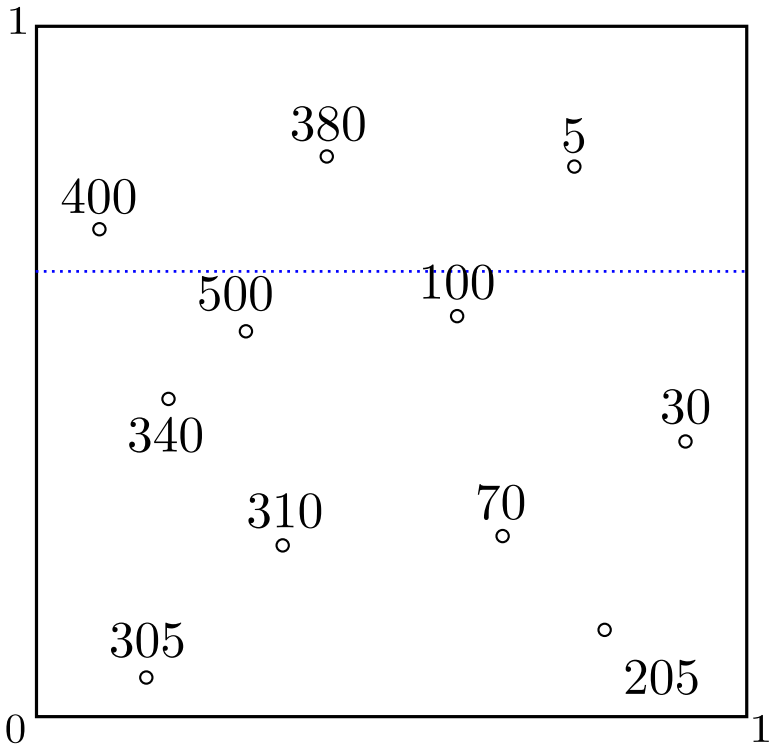
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

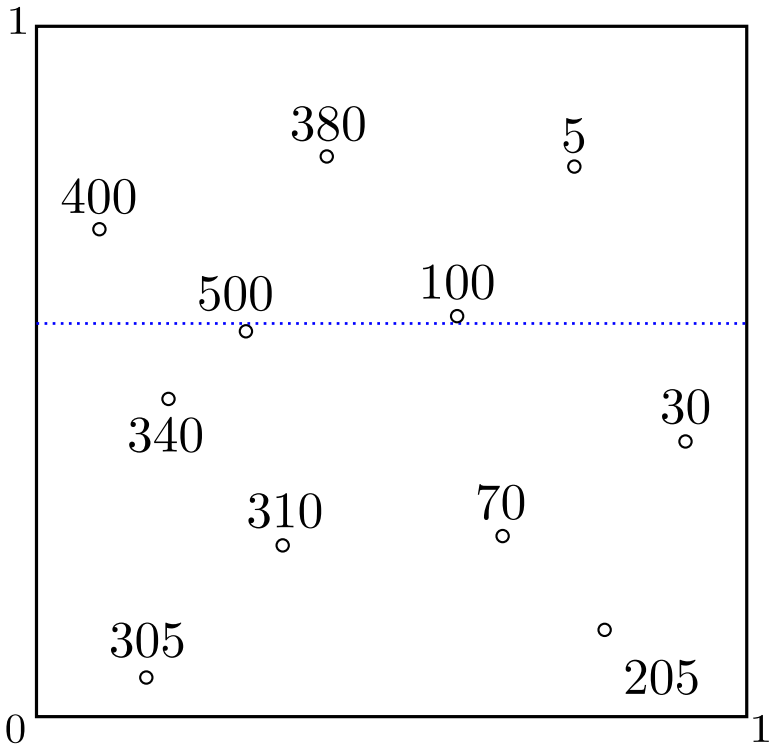
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

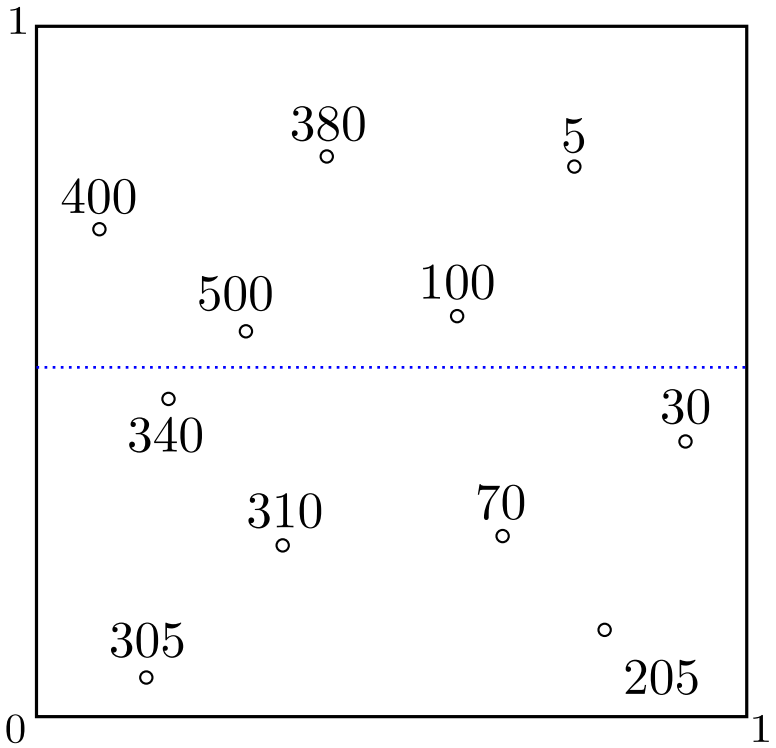
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

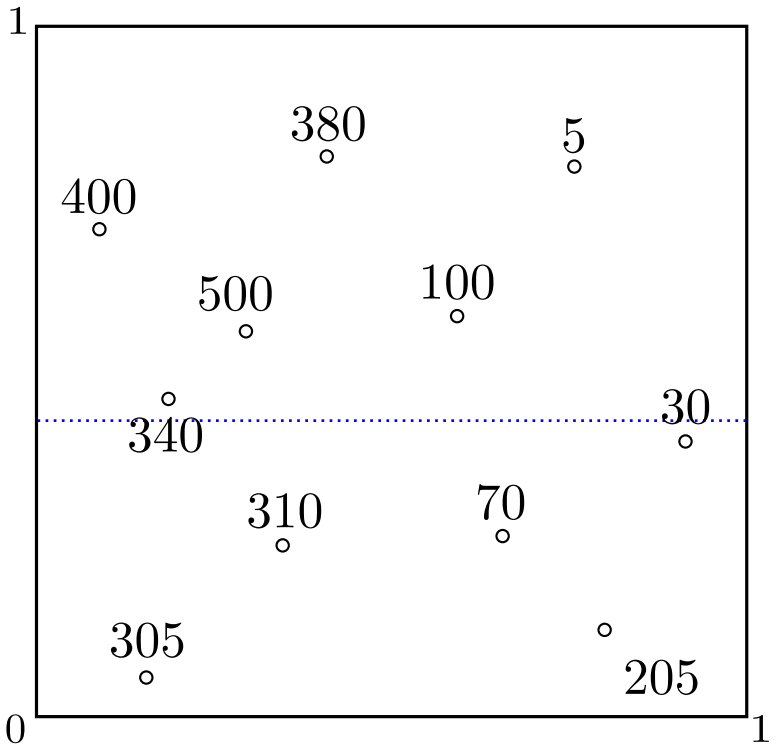
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

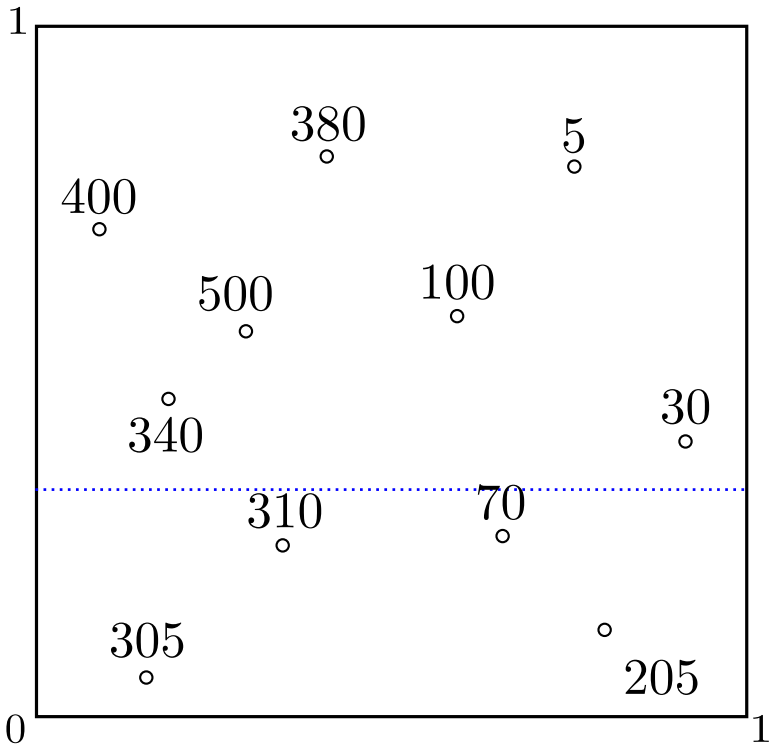
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

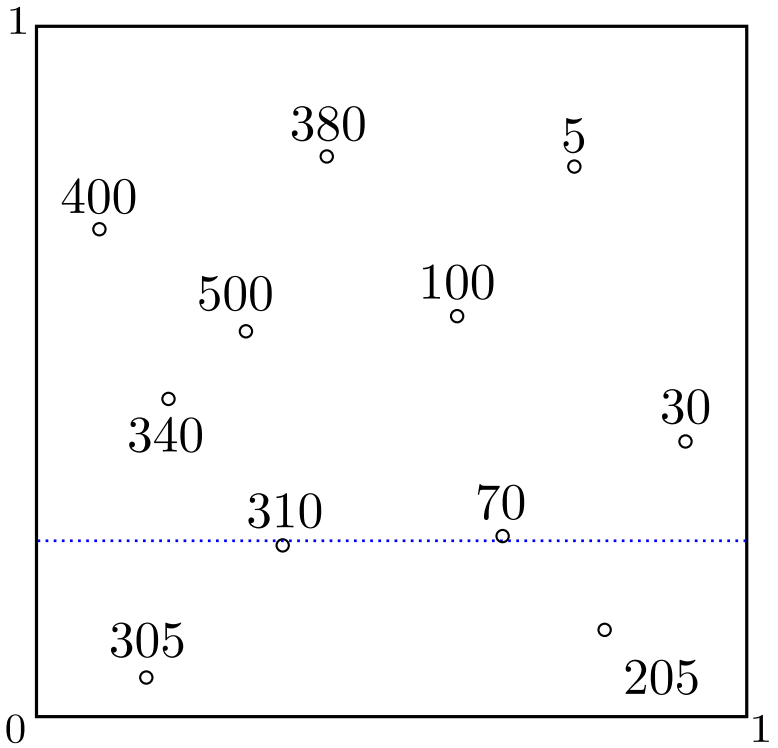
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

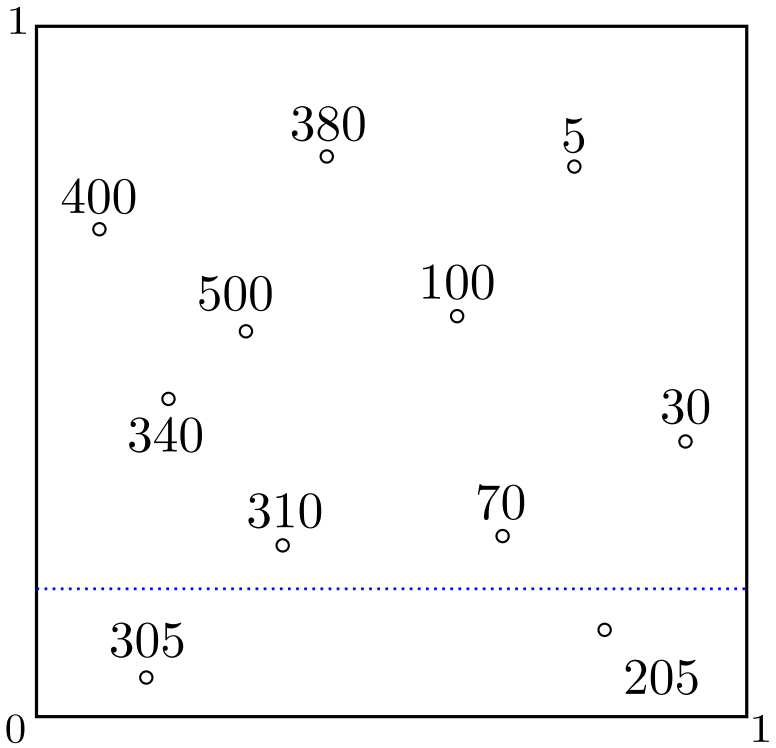
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

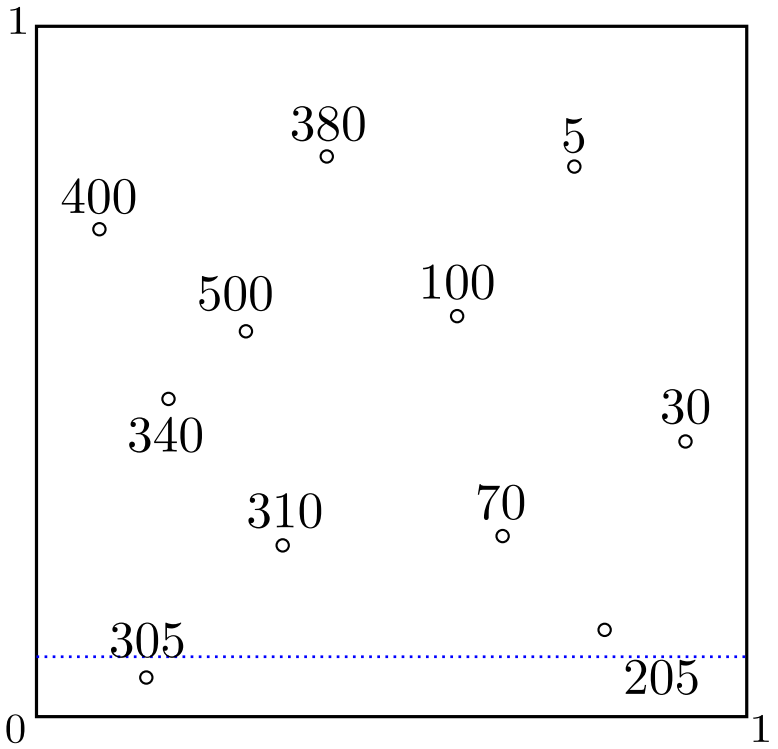
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

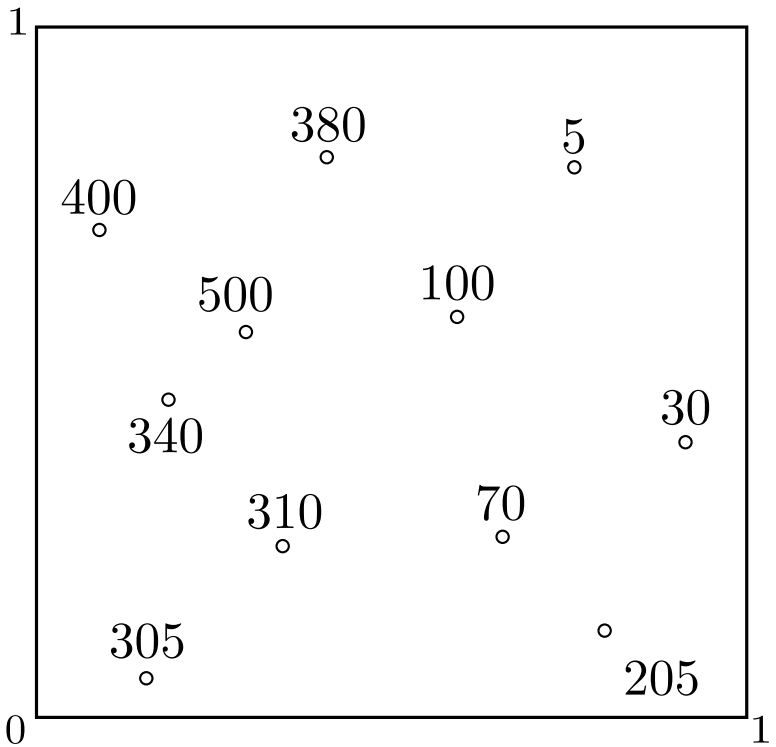
Regular split selection. In each cell of a tree, select the best split, by optimizing the splitting criterion along all directions.



$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

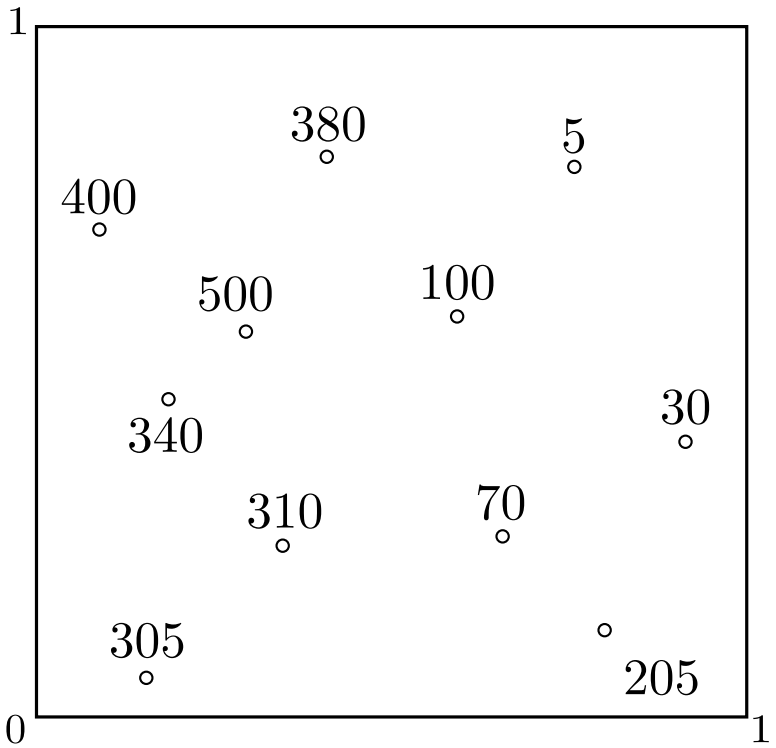


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

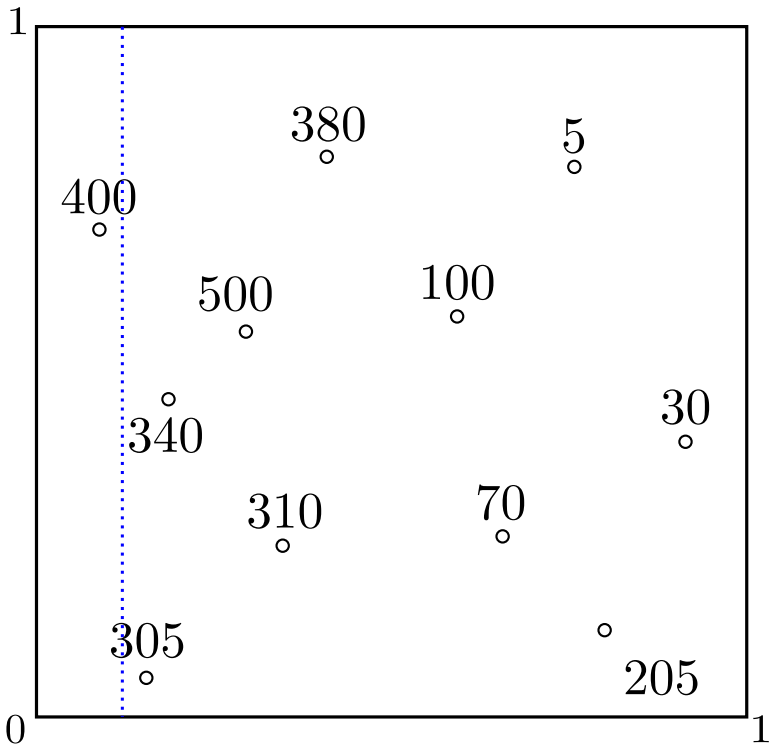


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

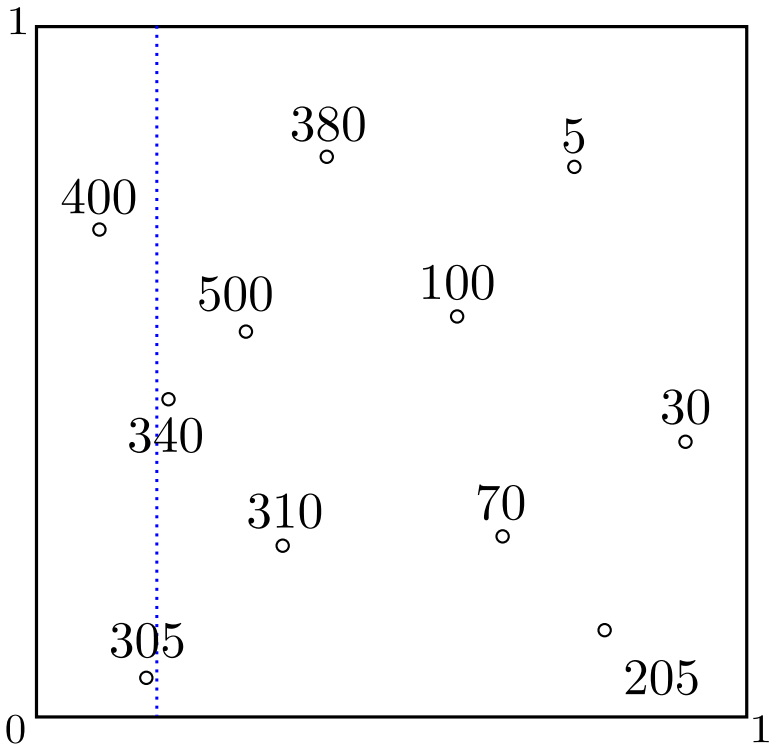


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

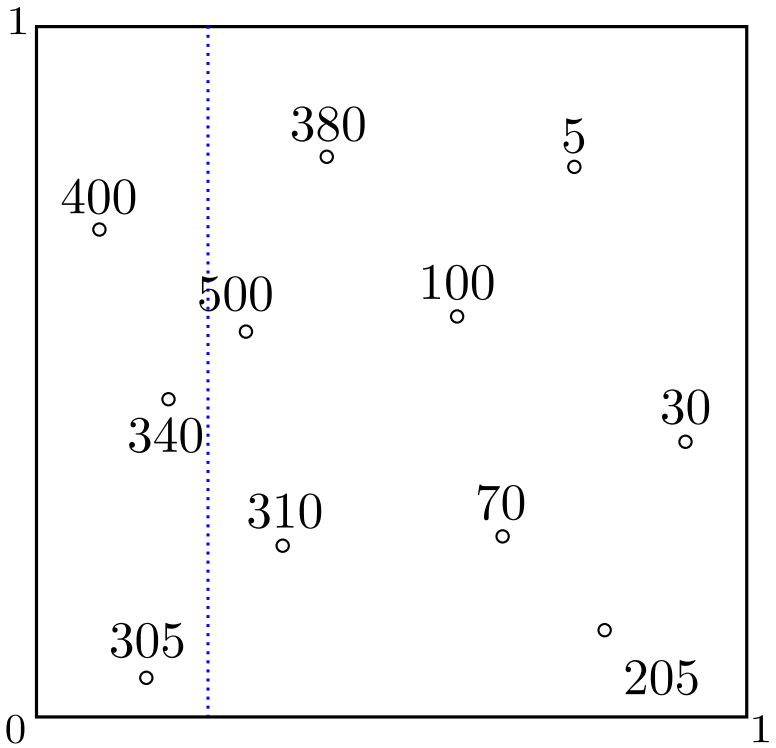


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

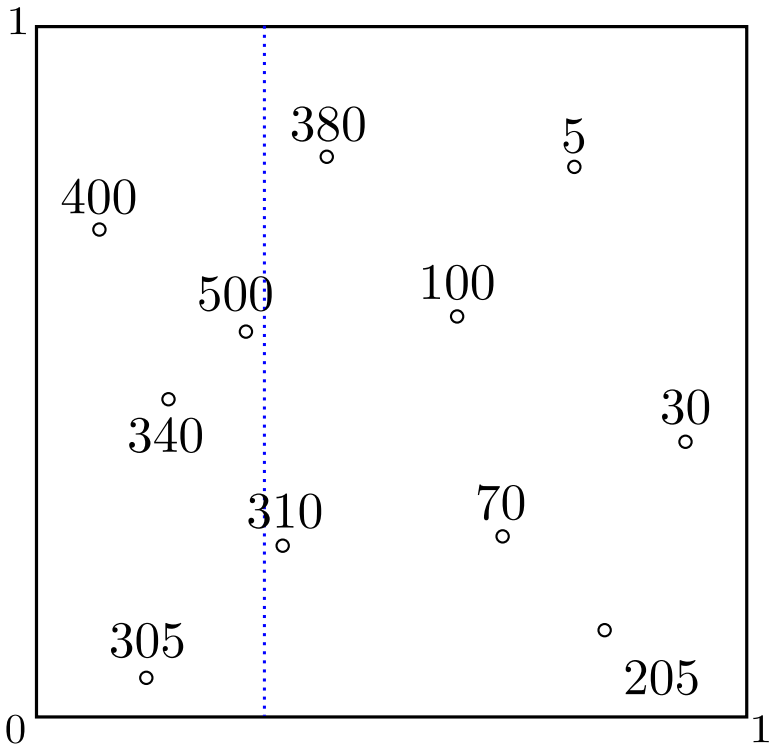


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

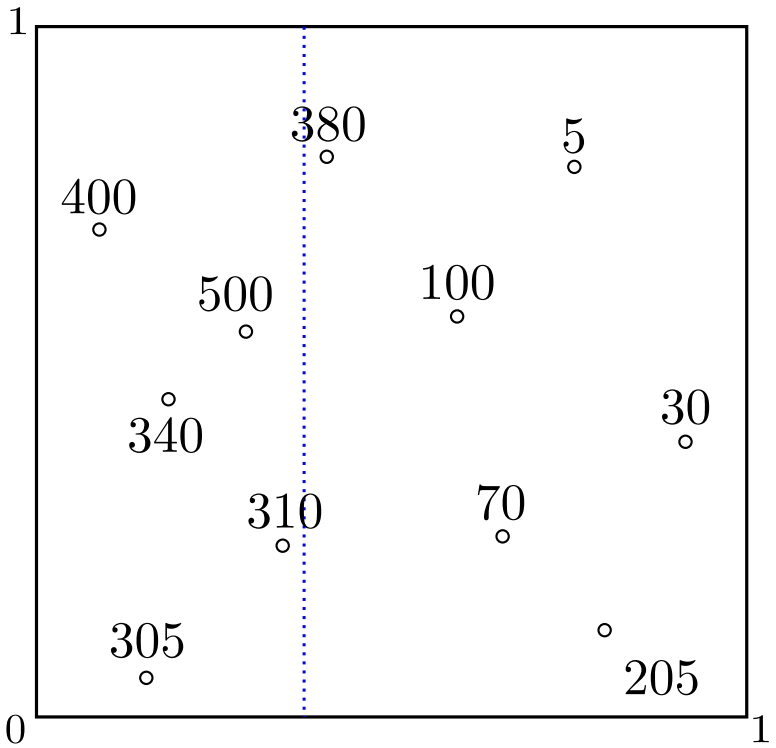


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

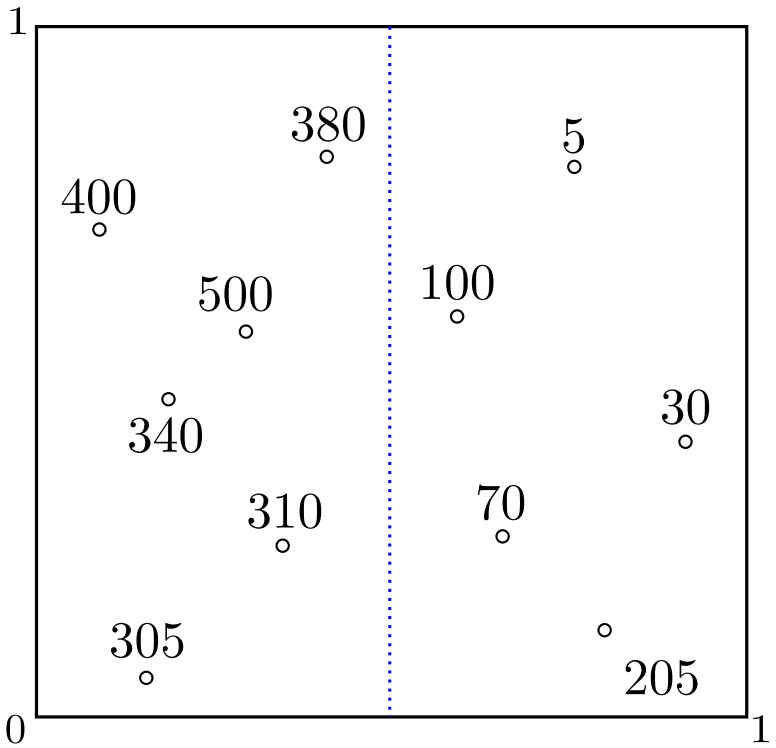


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

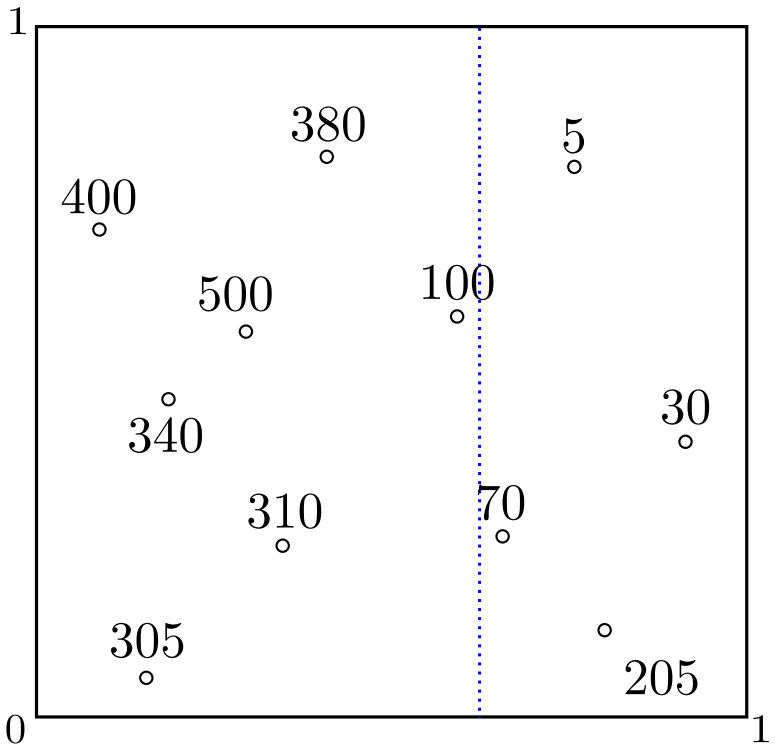


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

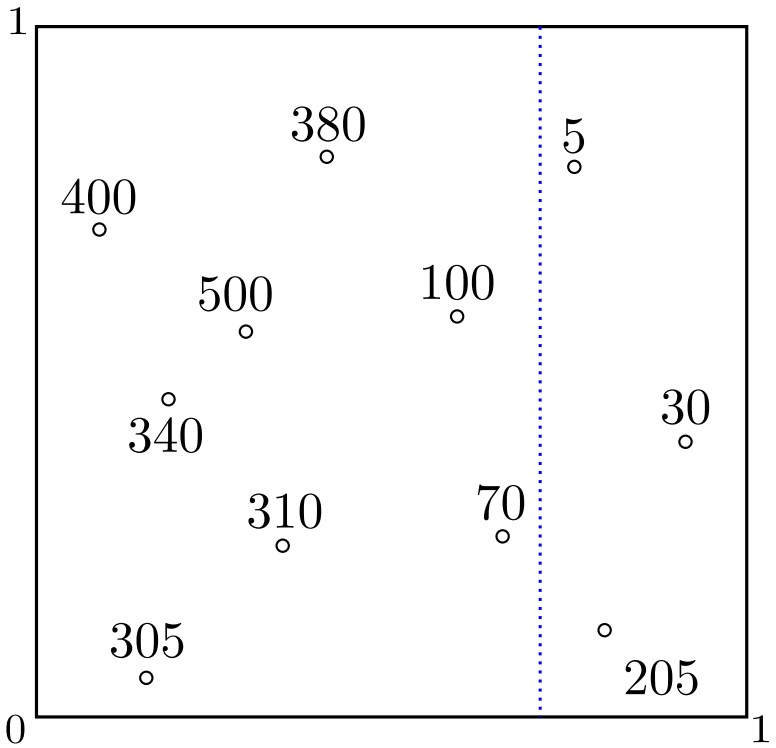


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

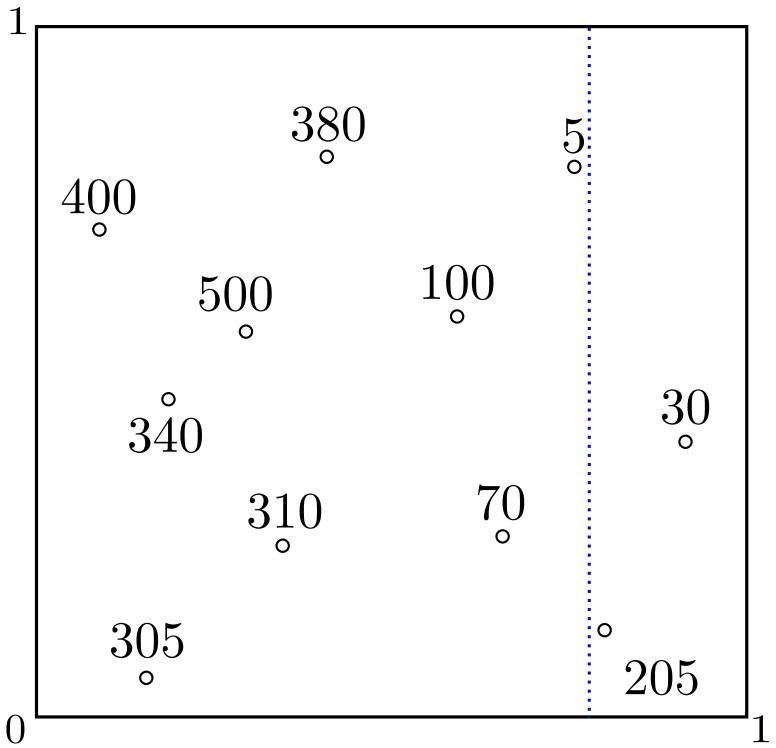


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

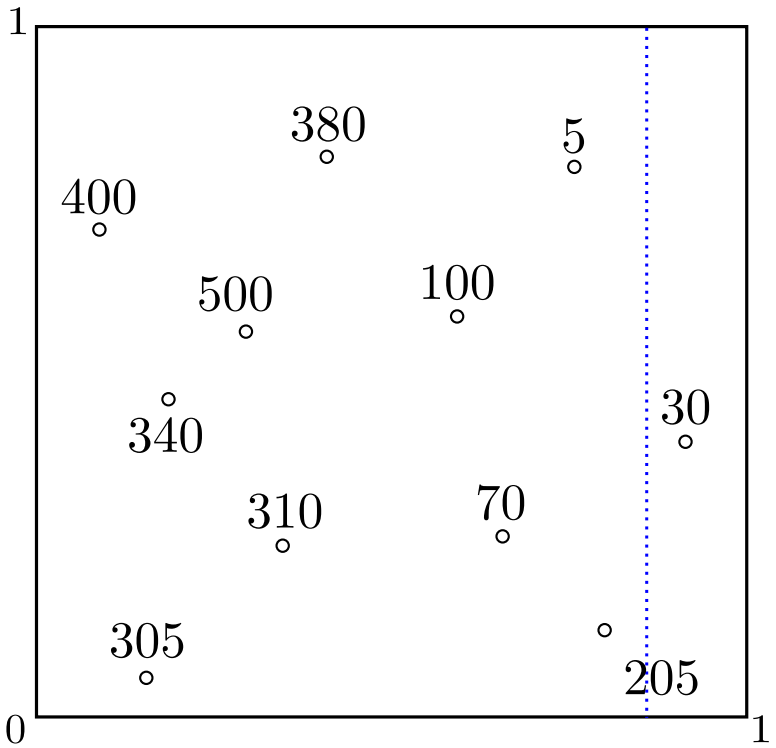


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.

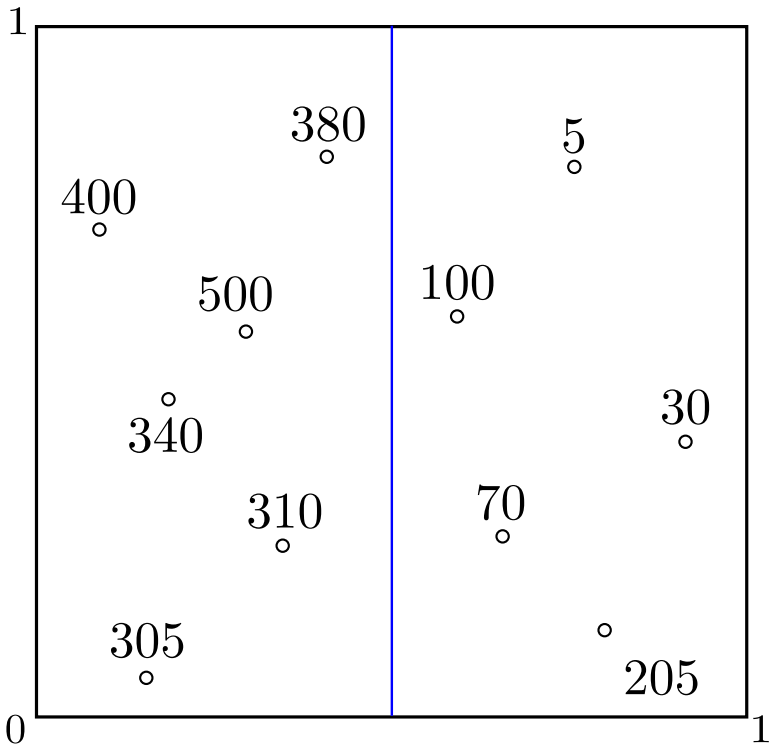


$k = 0$

Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.

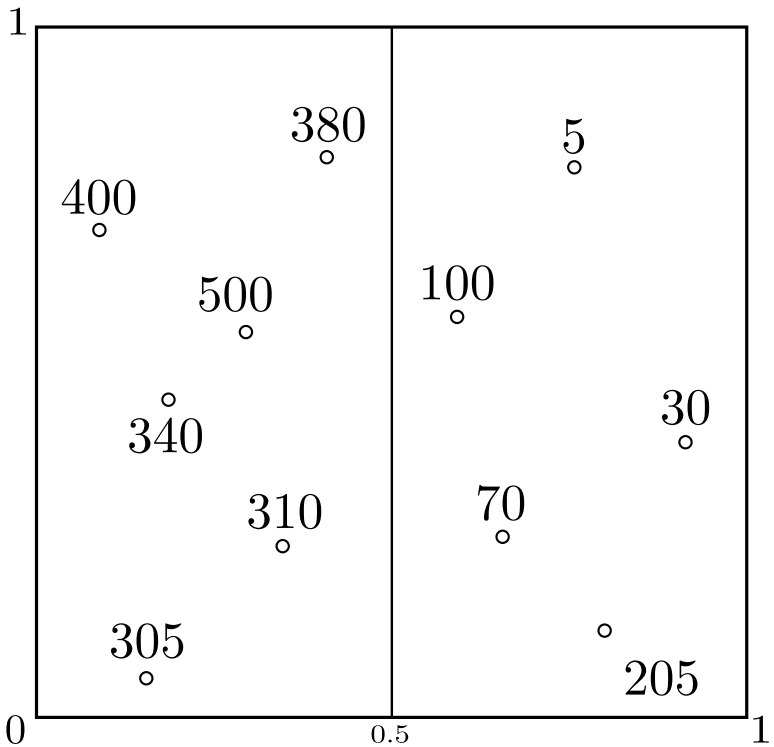
Here, for example, randomly selecting the first direction (variable $X^{(1)}$) leads to considering the following splits only.



$k = 0$

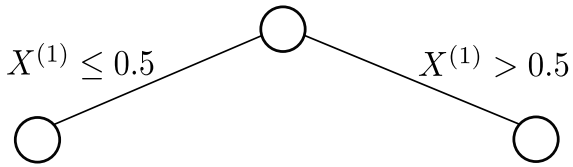
Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.



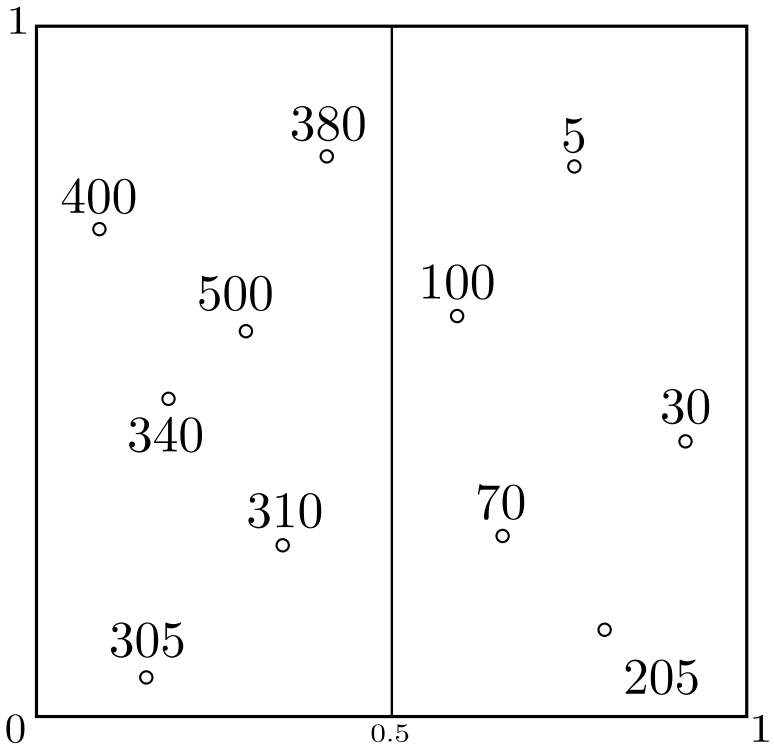
$k = 0$

$k = 1$



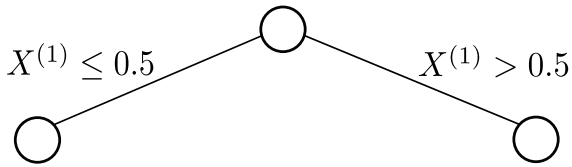
Split randomization in tree construction

Split randomization.
In each cell of a tree, select **uniformly at random a prespecified number of directions**. Select the best split (by optimizing the splitting criterion) **along these directions only**.



$k = 0$

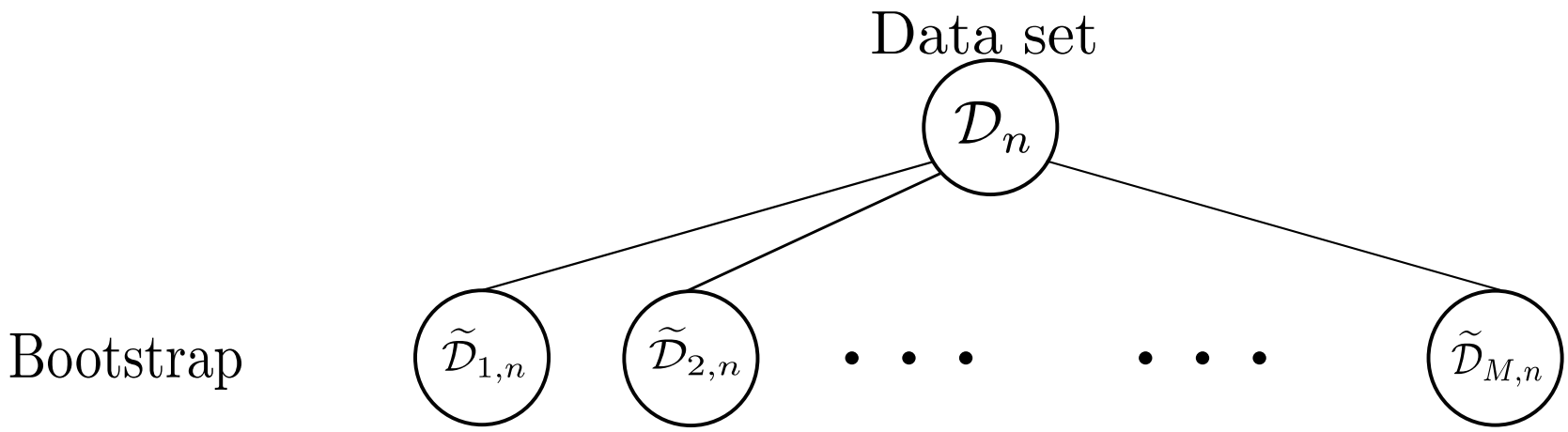
$k = 1$



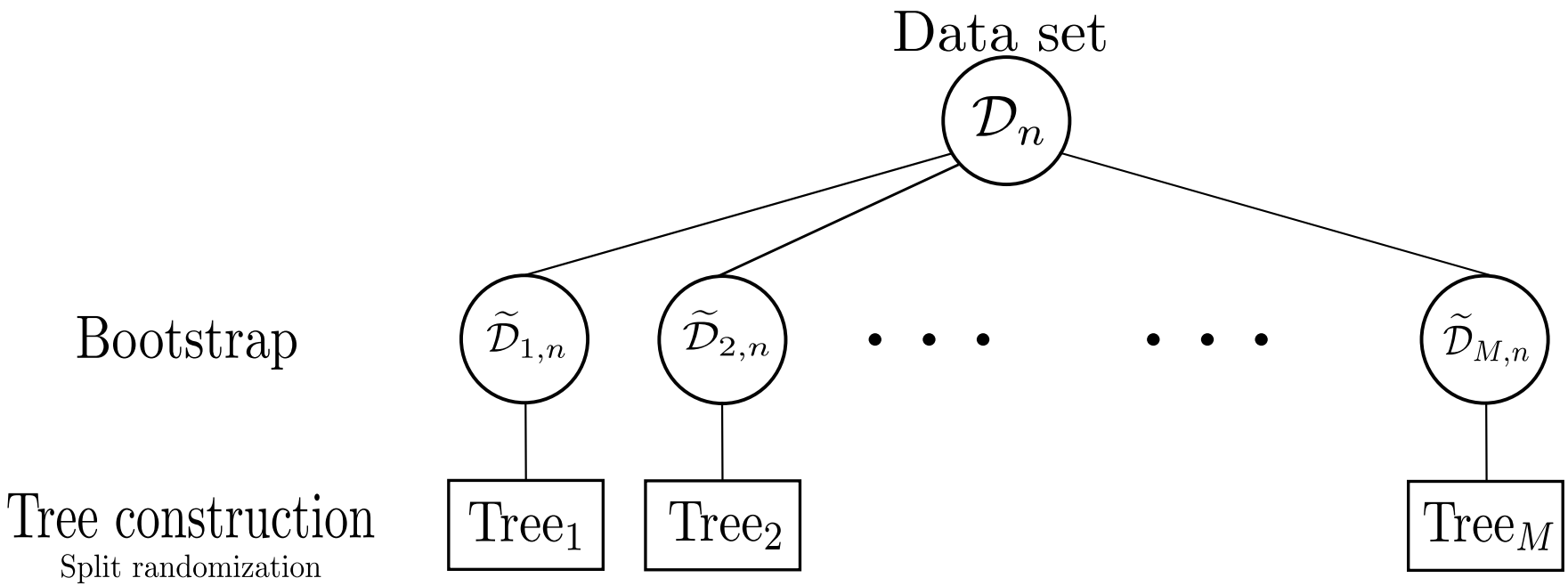
The same procedure is repeated on the resulting cells, with a new random choice of the splitting directions.

Construction of random forests

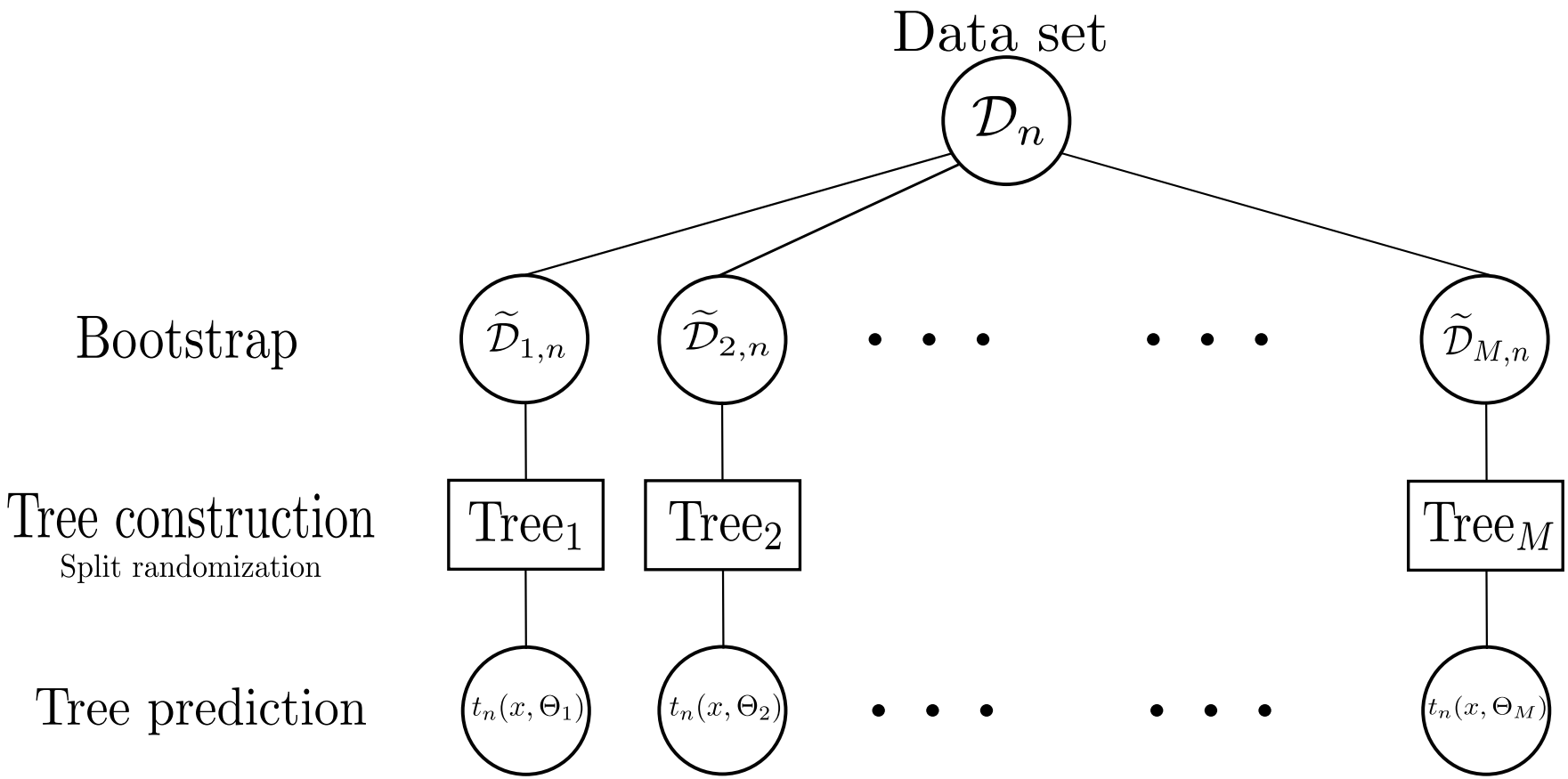
Construction of random forests



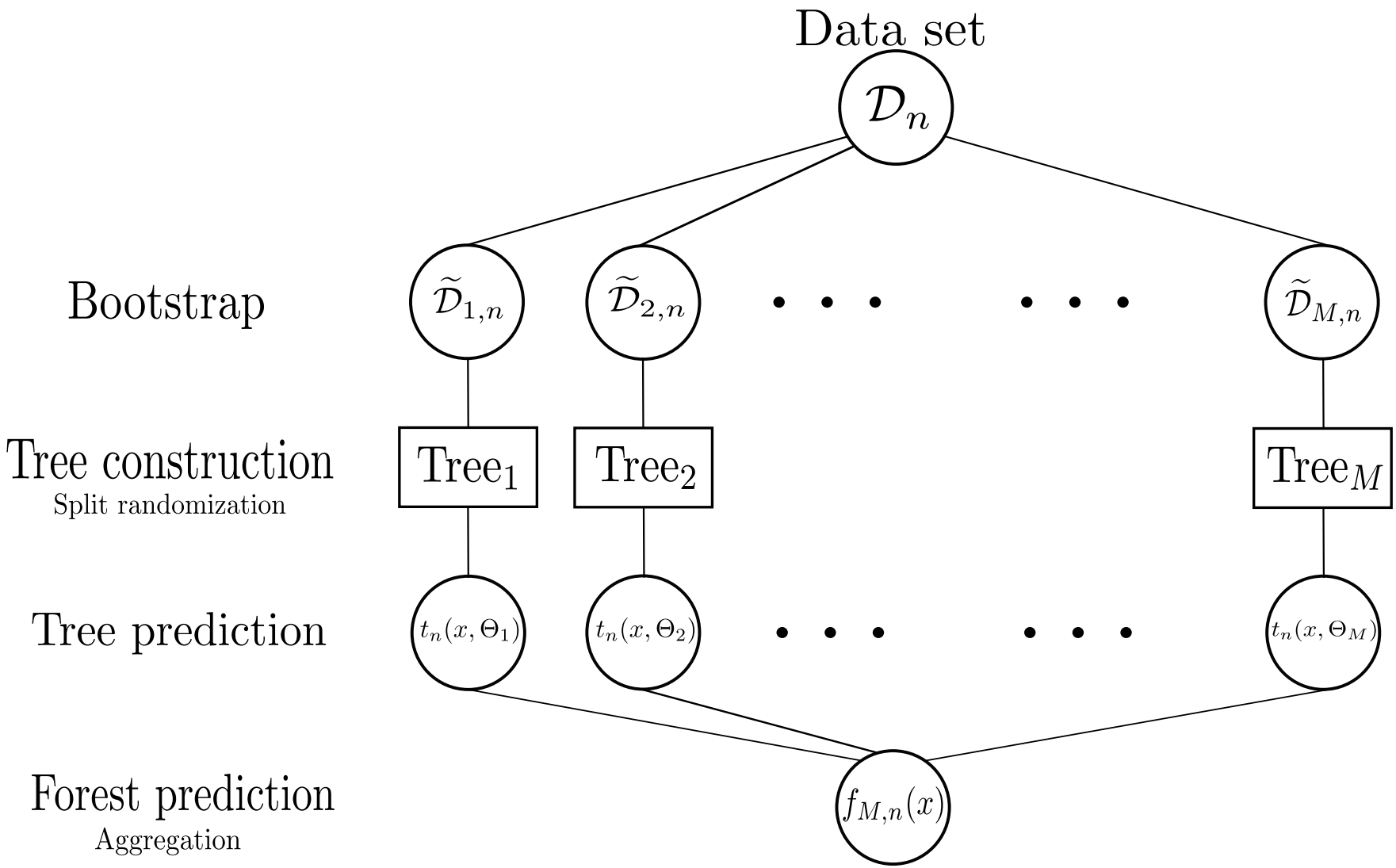
Construction of random forests



Construction of random forests



Construction of random forests



Construction of Breiman forests

- Build in parallel `n-estimators` CART as follows.

CART

Bootstrap - Select `max-samples` observations with replacement among the original sample $\mathcal{D}_n$. Use only these observations to build the tree.

For each cell,

Select randomly `max-features` coordinates among $\{1, \ldots, d\}$;
Choose the best split along previous directions, according to the splitting criterion the one minimizing the CART criterion.

Stop splitting each cell when all observations inside it have the same label or when a stopping criterion is met:

there are less than `min-sample-leaf` observation in the leaf
resulting cells would contain less than `min-sample-split` observation
cell is already split `max-depth` times
there are already `max-leaf-nodes` leaves

- Compute the forest prediction by averaging the predictions of all trees.

List of all random forest hyperparameters

See `Scikit-learn` documentation for more details: `RandomForestRegressor` / `RandomForestClassifier`.

List of all random forest hyperparameters

See `Scikit-learn` documentation for more details: `RandomForestRegressor` / `RandomForestClassifier`.

List of all hyperparameters in the forest:

- `n-estimators` $= 100$
- `criterion`='gini'
- `max-depth`=None, `min-samples-split` $= 2$, `min-samples-leaf` $= 1$, `min-weight-fraction-leaf` $= 0.0$, `min-impurity-decrease` $= 0.0$, `max-leaf-nodes`=None
  $\rightarrow$ By default, trees are fully grown with no pruning strategy
- `max-features`='sqrt' (classif.) 'None' (regression).
- `bootstrap`=True, `max-samples`=None

List of all random forest hyperparameters

See `Scikit-learn` documentation for more details: `RandomForestRegressor` / `RandomForestClassifier`.

List of all hyperparameters in the forest:

- `n-estimators` $= 100$
- `criterion`='gini'
- `max-depth`=None, `min-samples-split` $= 2$, `min-samples-leaf` $= 1$, `min-weight-fraction-leaf` $= 0.0$, `min-impurity-decrease` $= 0.0$, `max-leaf-nodes`=None
  $\rightarrow$ By default, trees are fully grown with no pruning strategy
- `max-features`='sqrt' (classif.) 'None' (regression).
- `bootstrap`=True, `max-samples`=None

Remarks.

- Due to bootstrap and split randomization, running twice RF may lead to different results. Increasing the number of trees limits this difference.
- Fixing a `random-state` with no boostrap and no split randomization will make two runs of RF identical.
- Beware, by default, split randomization is used in classification but not in regression!

Role of each hyperparameter

Number of trees.

- Larger values are better
- No statistical tradeoff between low and high values
- Limited by computational power - growing many trees is expensive
- Default values (several hundreds / thousands) usually do a good job

Role of each hyperparameter

Number of trees.

- Larger values are better
- No statistical tradeoff between low and high values
- Limited by computational power - growing many trees is expensive
- Default values (several hundreds / thousands) usually do a good job

Bootstrap size / tree shape.

- Control the bias/variance tradeoff: small bootstrap size / shallow trees lead to predictors with a large bias but a small variance
- Use small bootstrap size or shallow tree if data are very noisy
- Use default setting for modelling very complex phenomenon

$\rightarrow$ Precise tuning can help but default values are good in general

Role of each hyperparameter

Number of trees.

- Larger values are better
- No statistical tradeoff between low and high values
- Limited by computational power - growing many trees is expensive
- Default values (several hundreds / thousands) usually do a good job

Bootstrap size / tree shape.

- Control the bias/variance tradeoff: small bootstrap size / shallow trees lead to predictors with a large bias but a small variance
- Use small bootstrap size or shallow tree if data are very noisy
- Use default setting for modelling very complex phenomenon

$\rightarrow$ Precise tuning can help but default values are good in general

Split randomization

- Most complex parameter to tune
- Small values of `max-features` lead to very different trees
  $\rightarrow$ `max-features`=1 corresponds to drawing randomly the splitting direction
- Large values of `max-features` lead to similar trees
  $\rightarrow$ `max-features`=d corresponds to building the same tree (if no bootstrap is used)
- No precise heuristic, can be tuned by cross-validation.

Out-of-bag error

Idea. Evaluate the error of a random forest using the fact that each observation has not been used in all tree constructions and can thus be used as test points for such aggregated trees.

General procedure (short):

- Consider that a forest has been trained on the data set $\mathcal{D}_n$.
- For each observation $i \in \{1, \ldots, n\}$,
  - Consider the bootstrap samples that do not contain this observation.
  - For the trees that are not built using observation i, compute the predictions at X_i and aggregate them. Compute the loss of such an aggregated prediction.
- Compute the Out-of-bag error by averaging the losses over all observations.

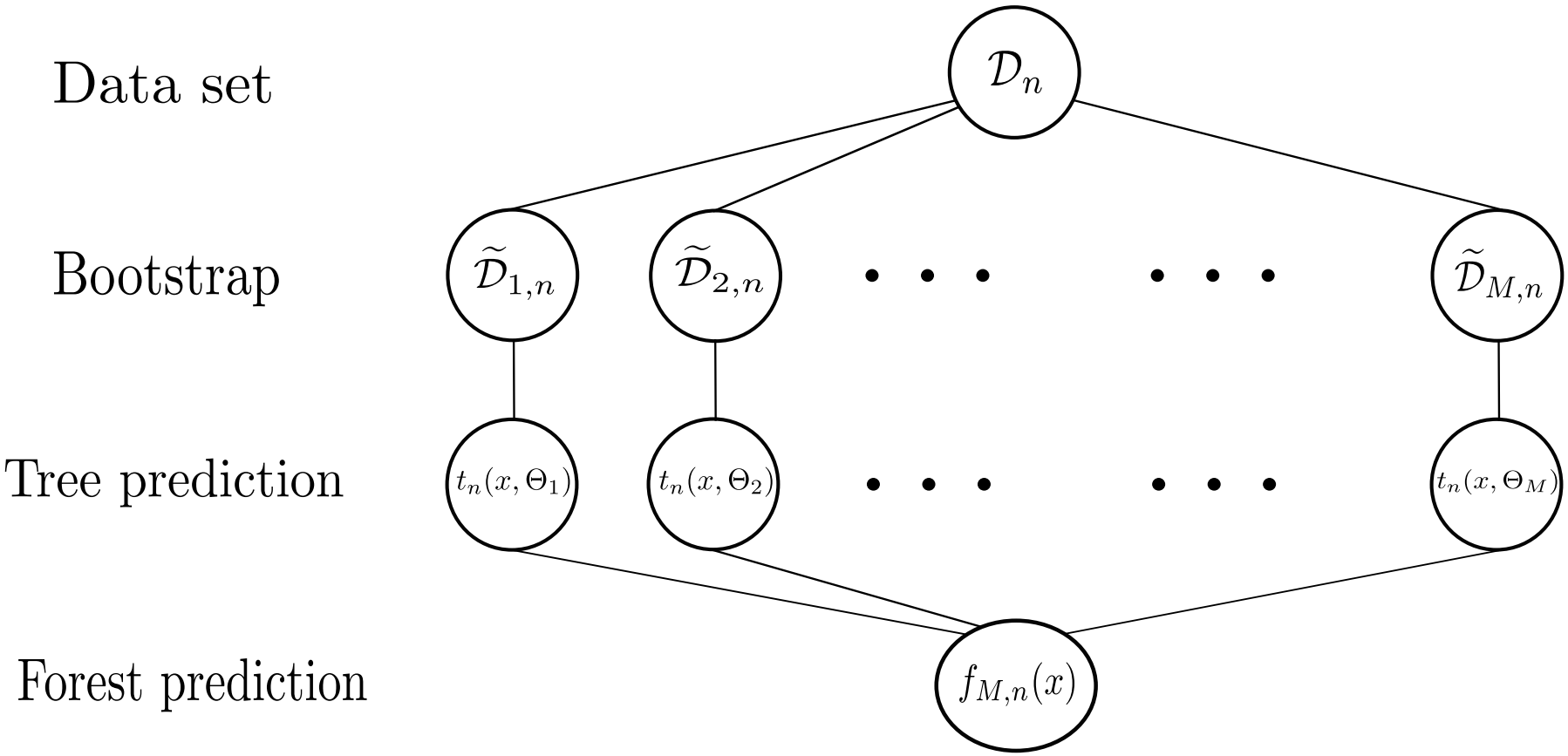
Benefits:

- No need for dividing the data set into a train and a test set
- Easily parallelizable
- Asymptotically equivalent to the risk of the forest for large M.

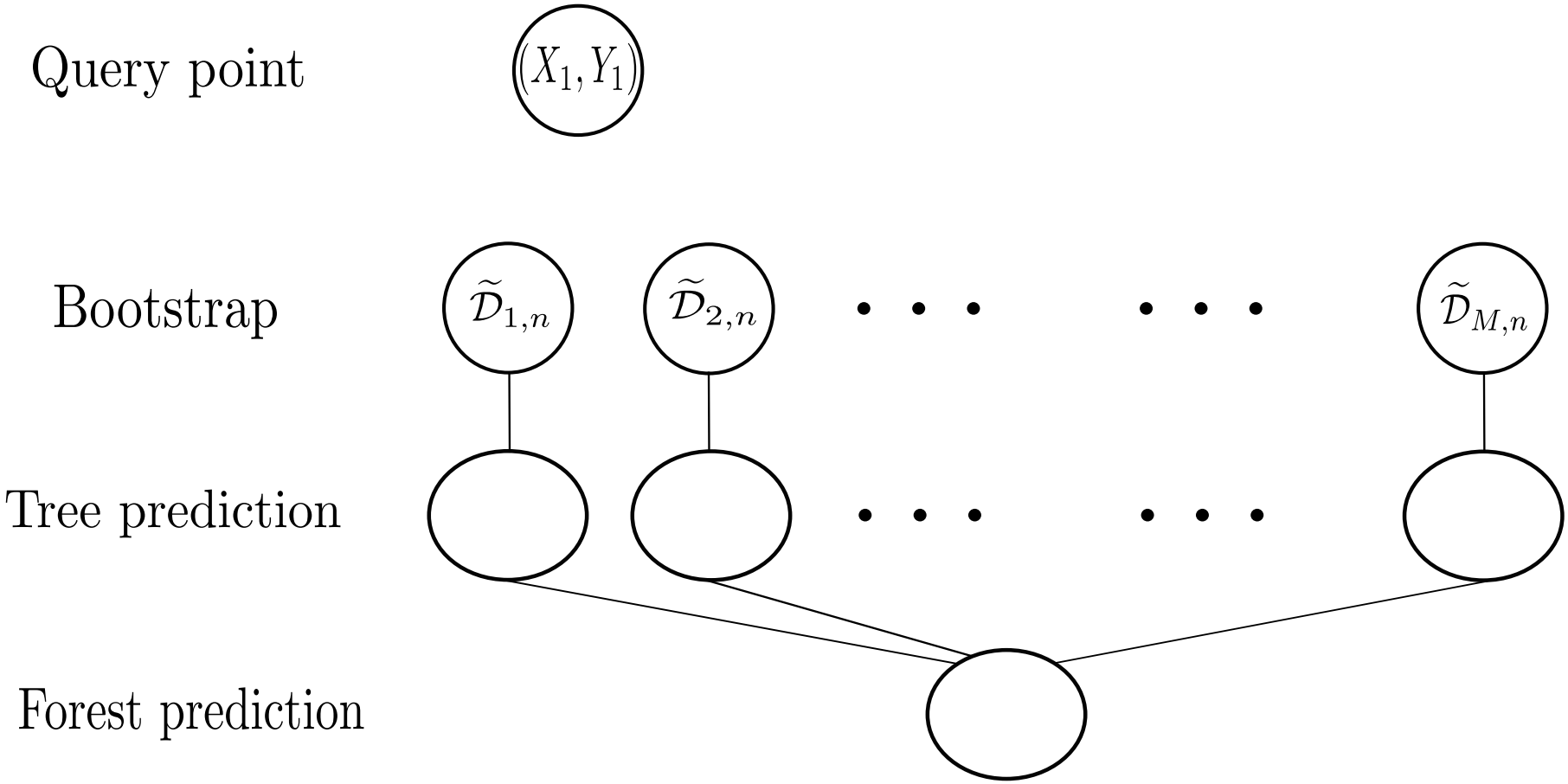
Drawback:

- Do not compute exactly the error of the whole forest but rather the aggregated error of some trees in the forest.

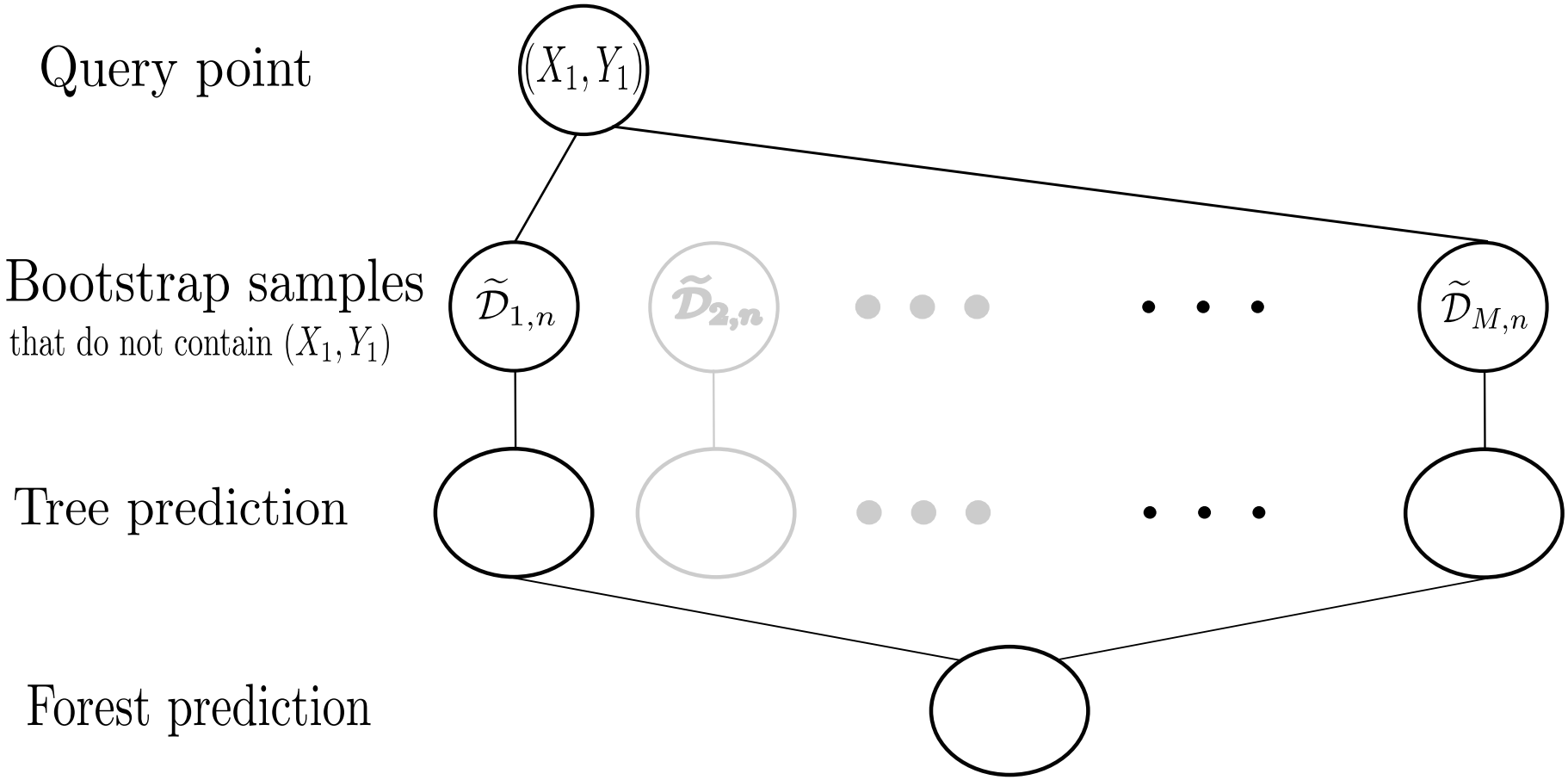
Out-of-bag procedure



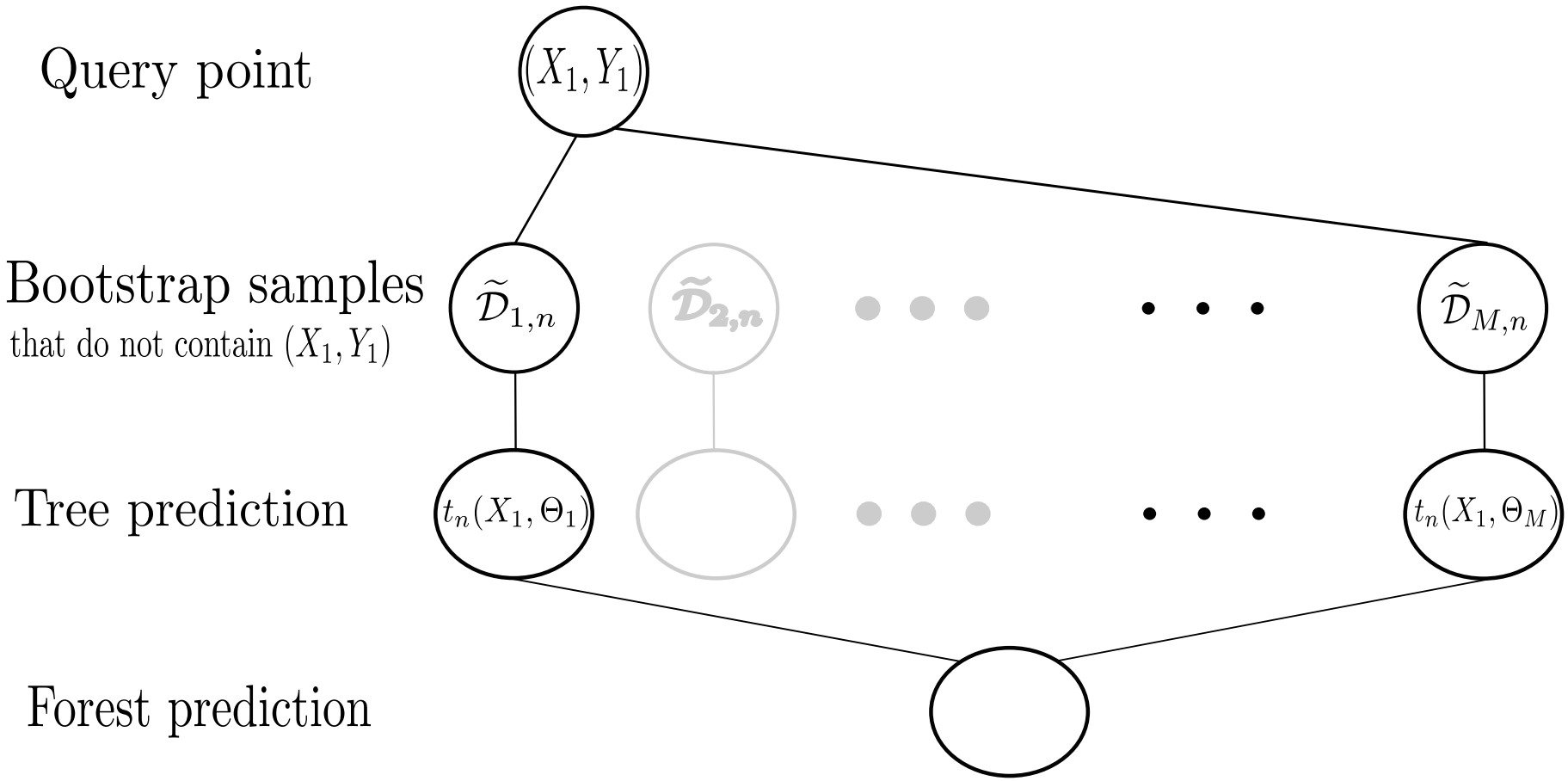
Out-of-bag procedure



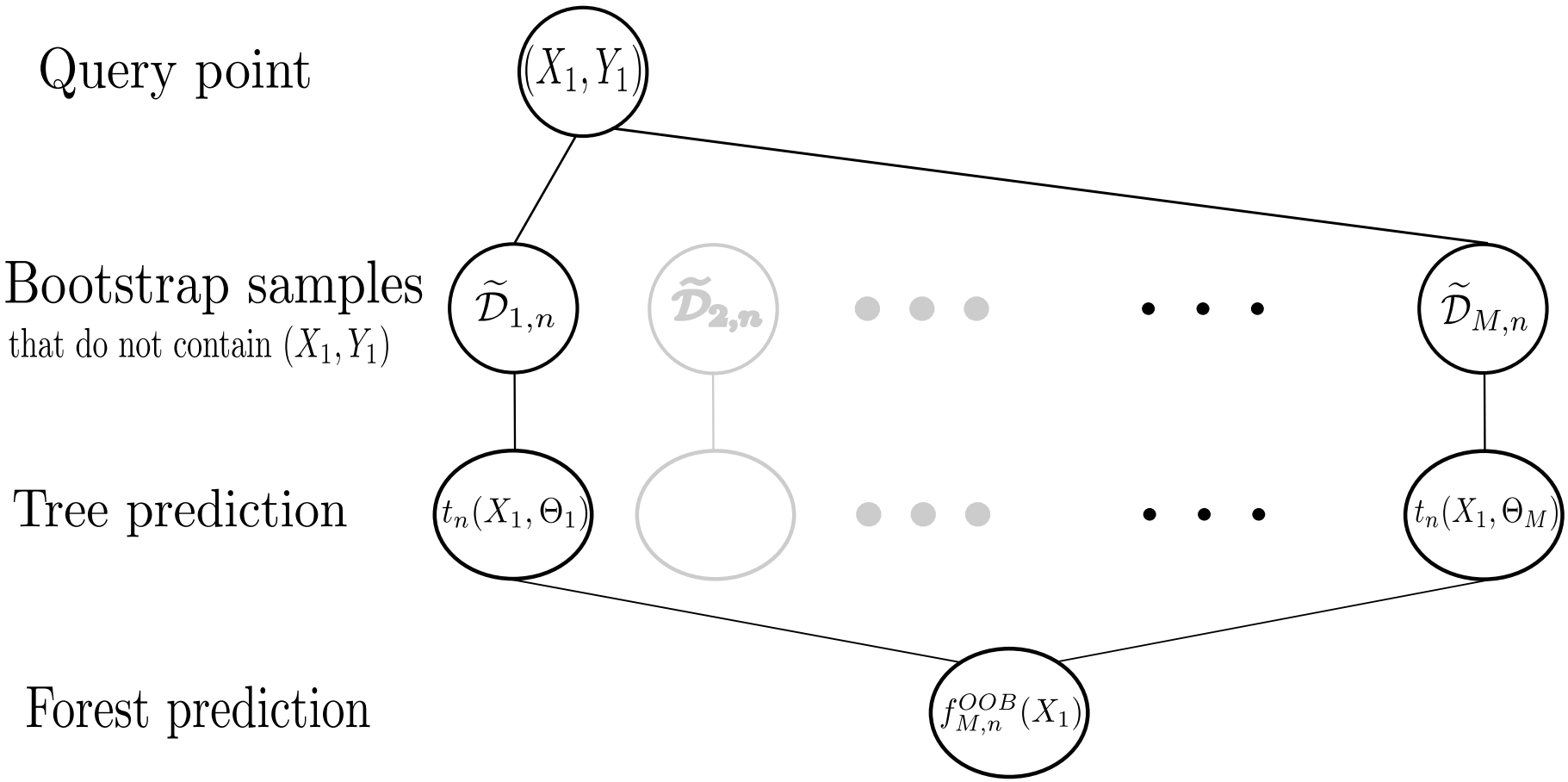
Out-of-bag procedure



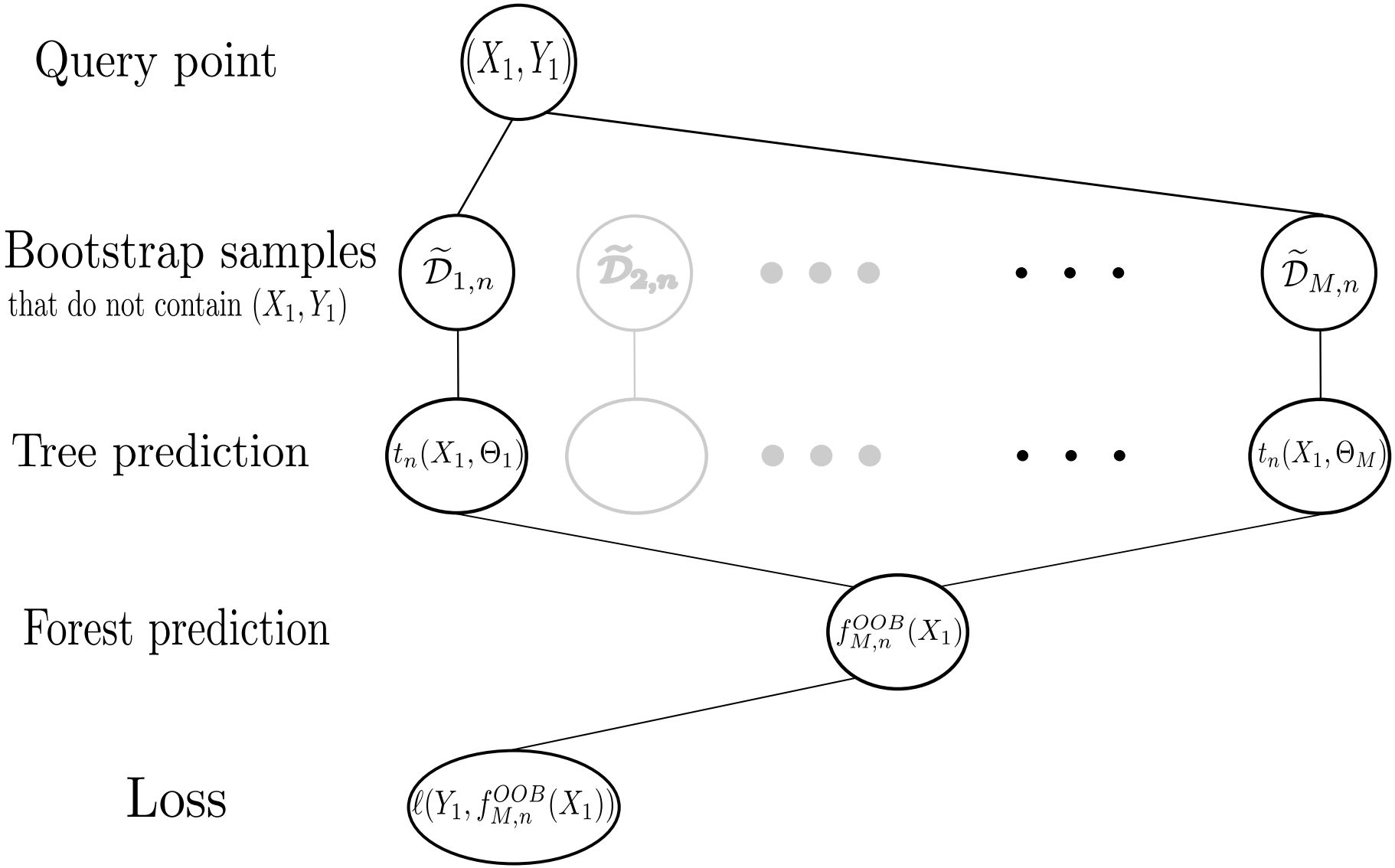
Out-of-bag procedure



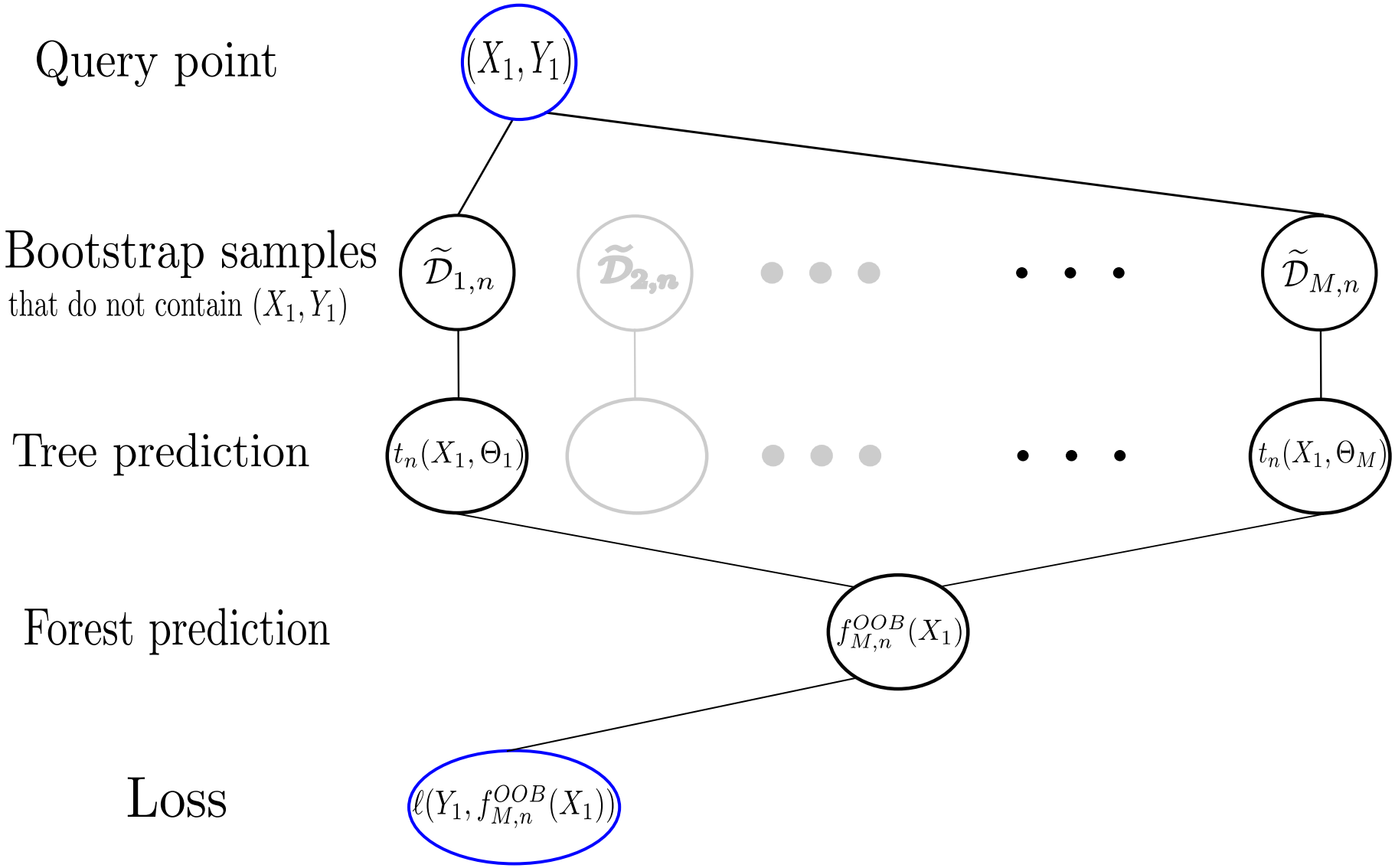
Out-of-bag procedure



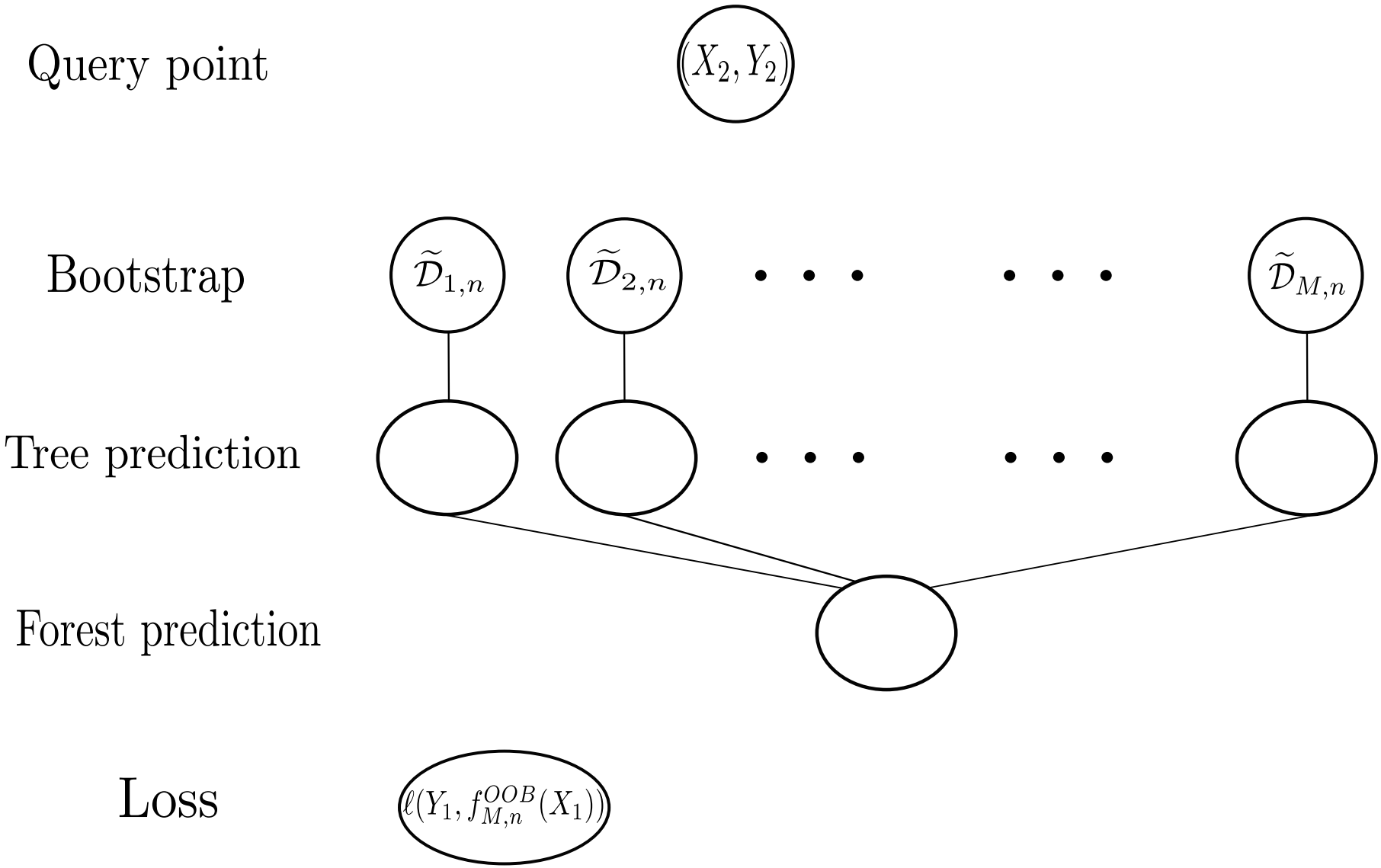
Out-of-bag procedure



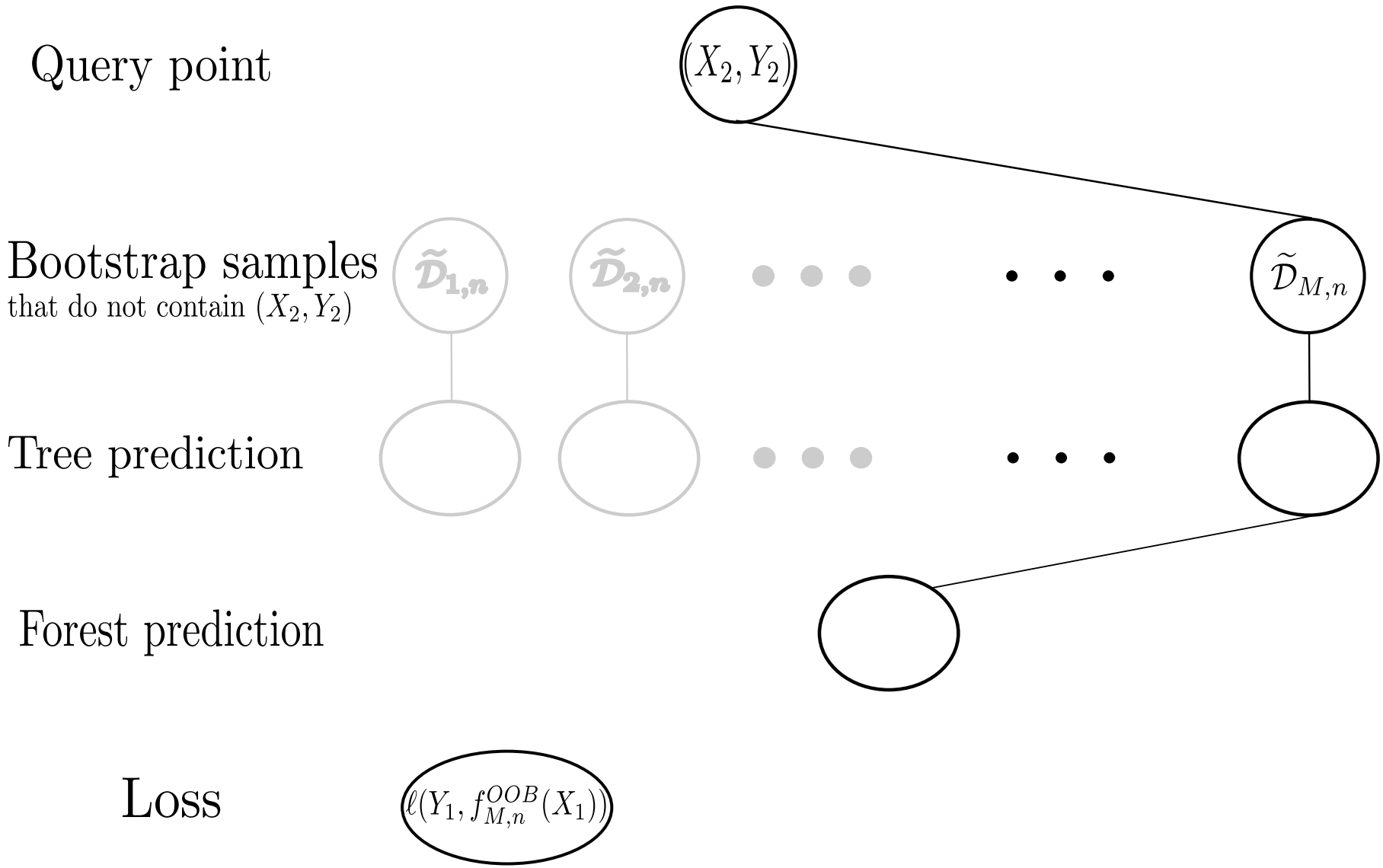
Out-of-bag procedure



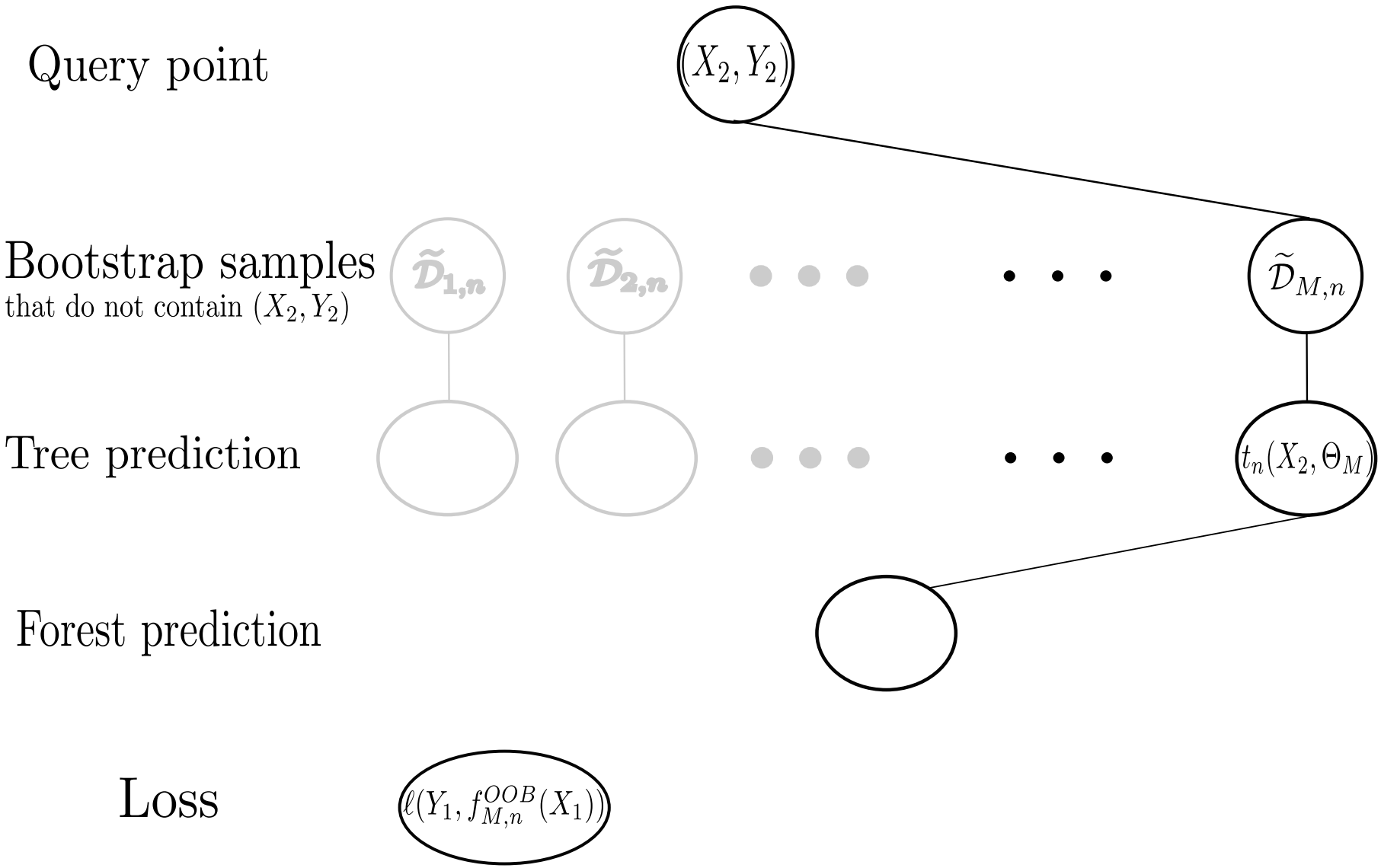
Out-of-bag procedure



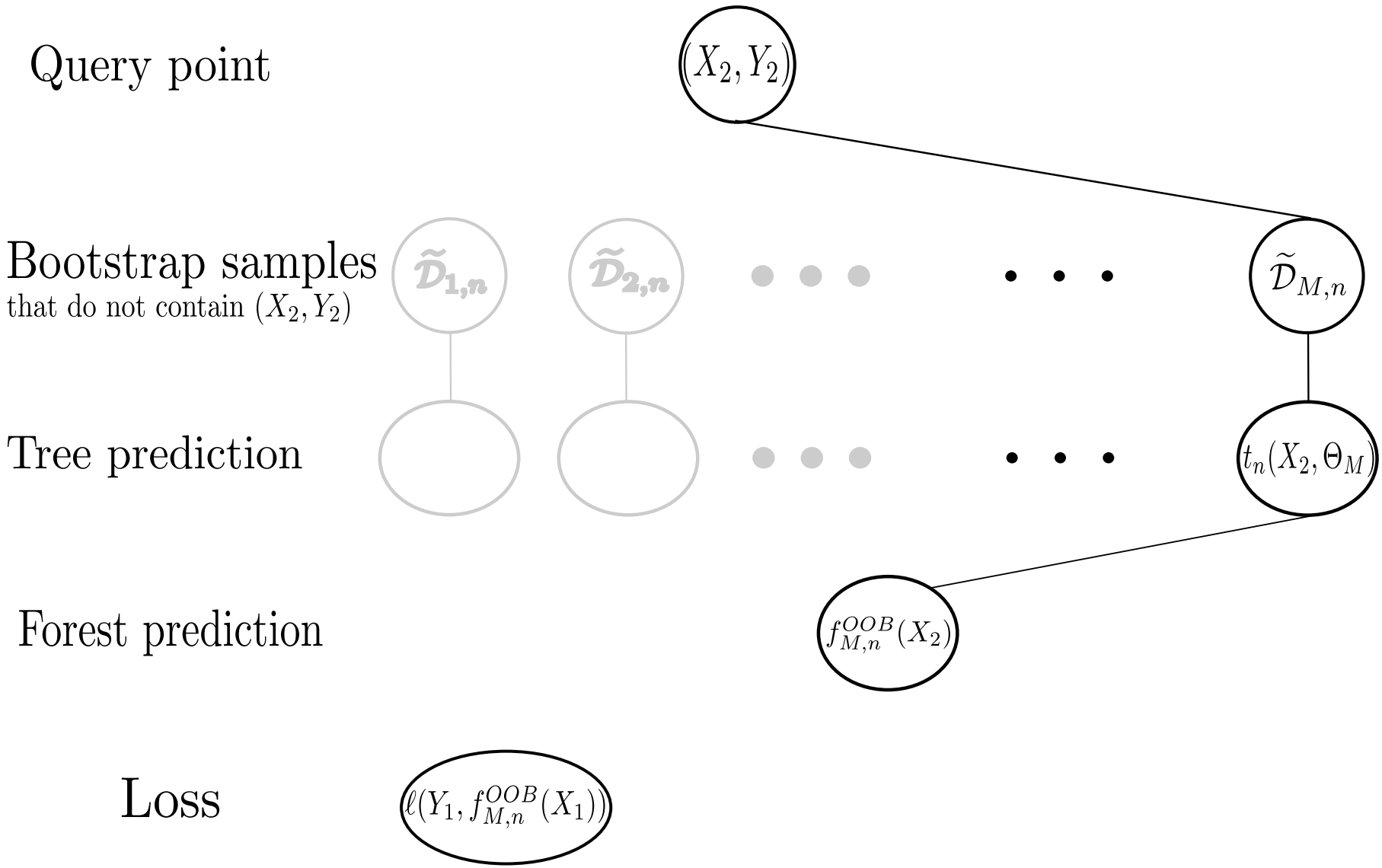
Out-of-bag procedure



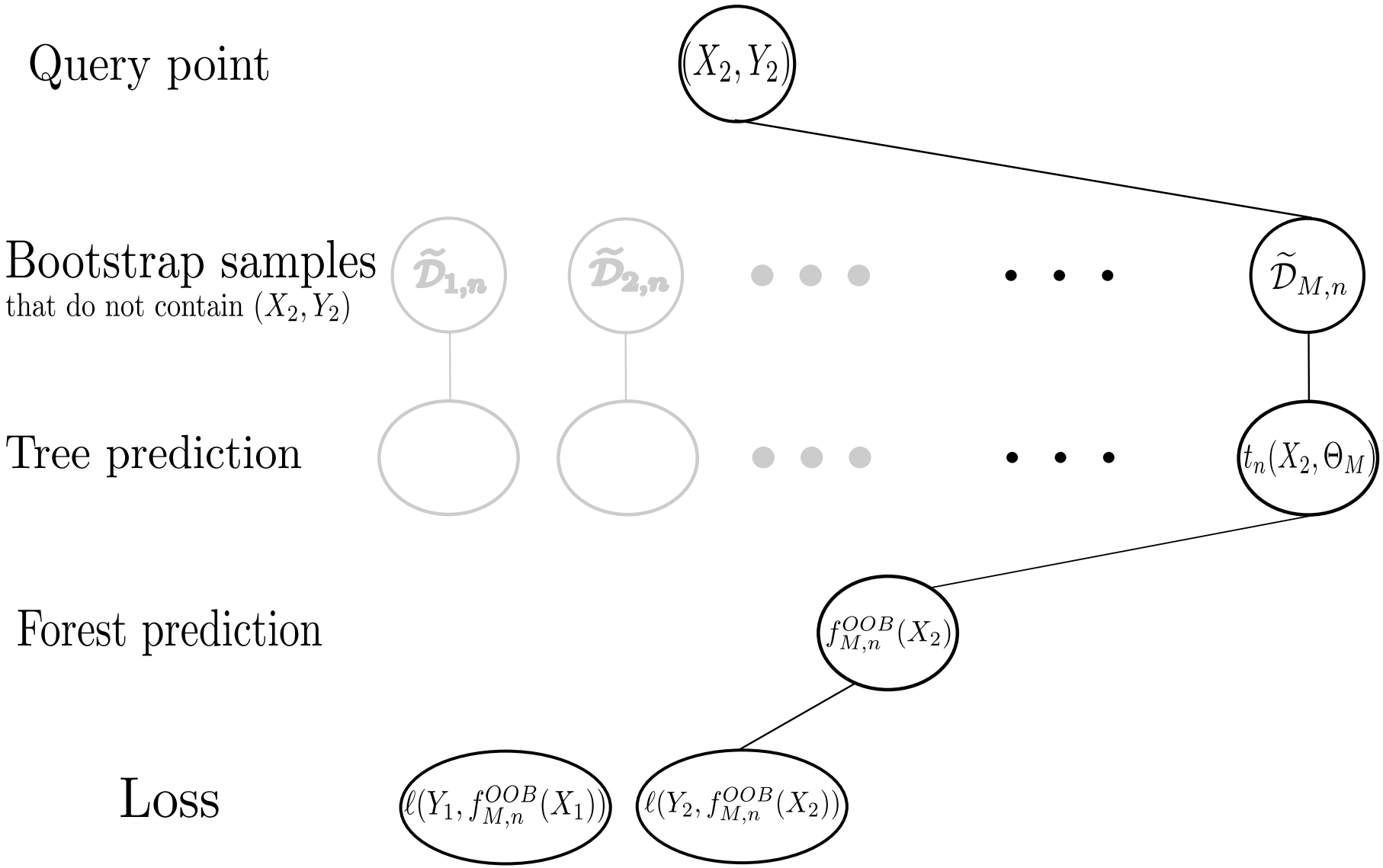
Out-of-bag procedure



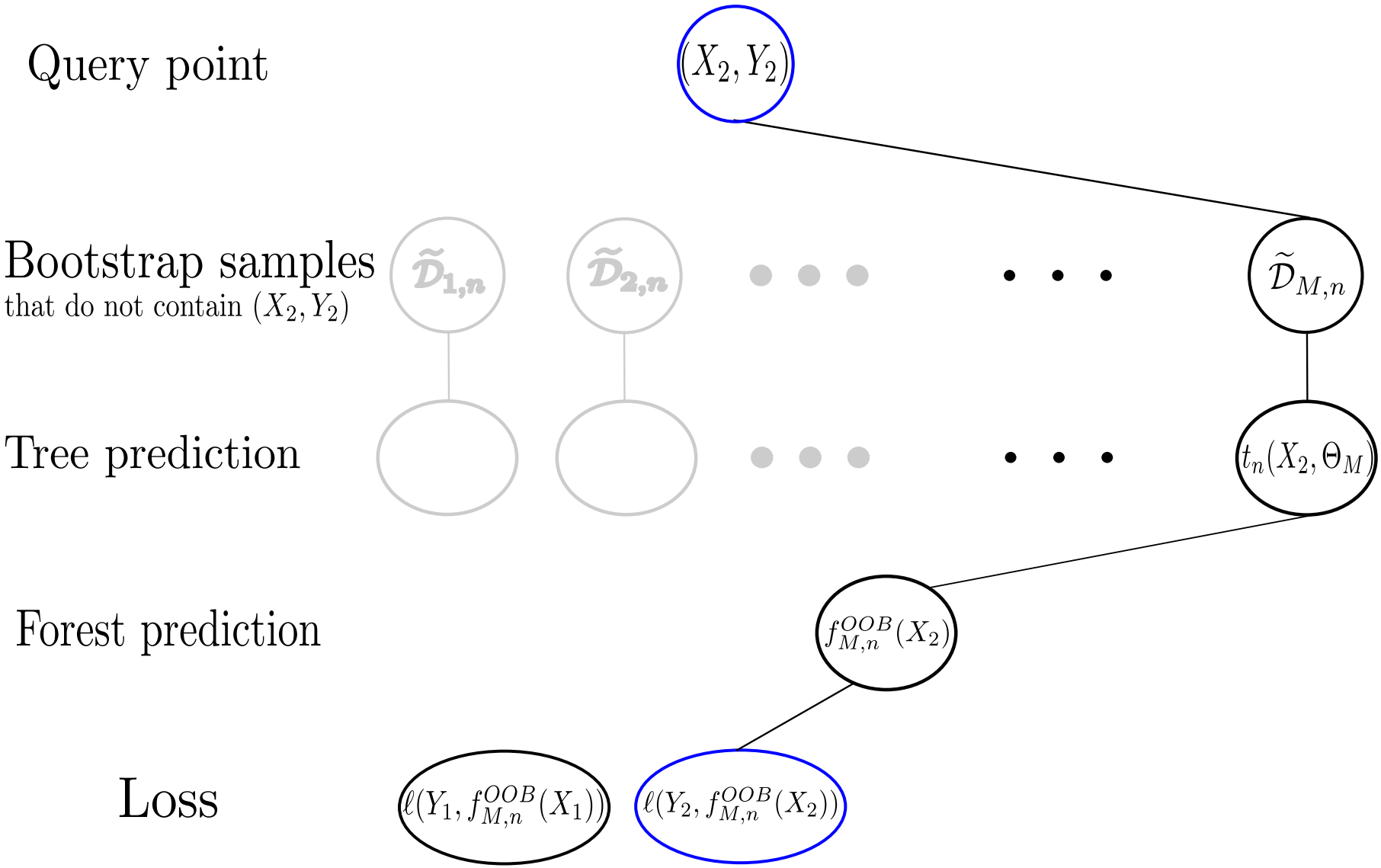
Out-of-bag procedure



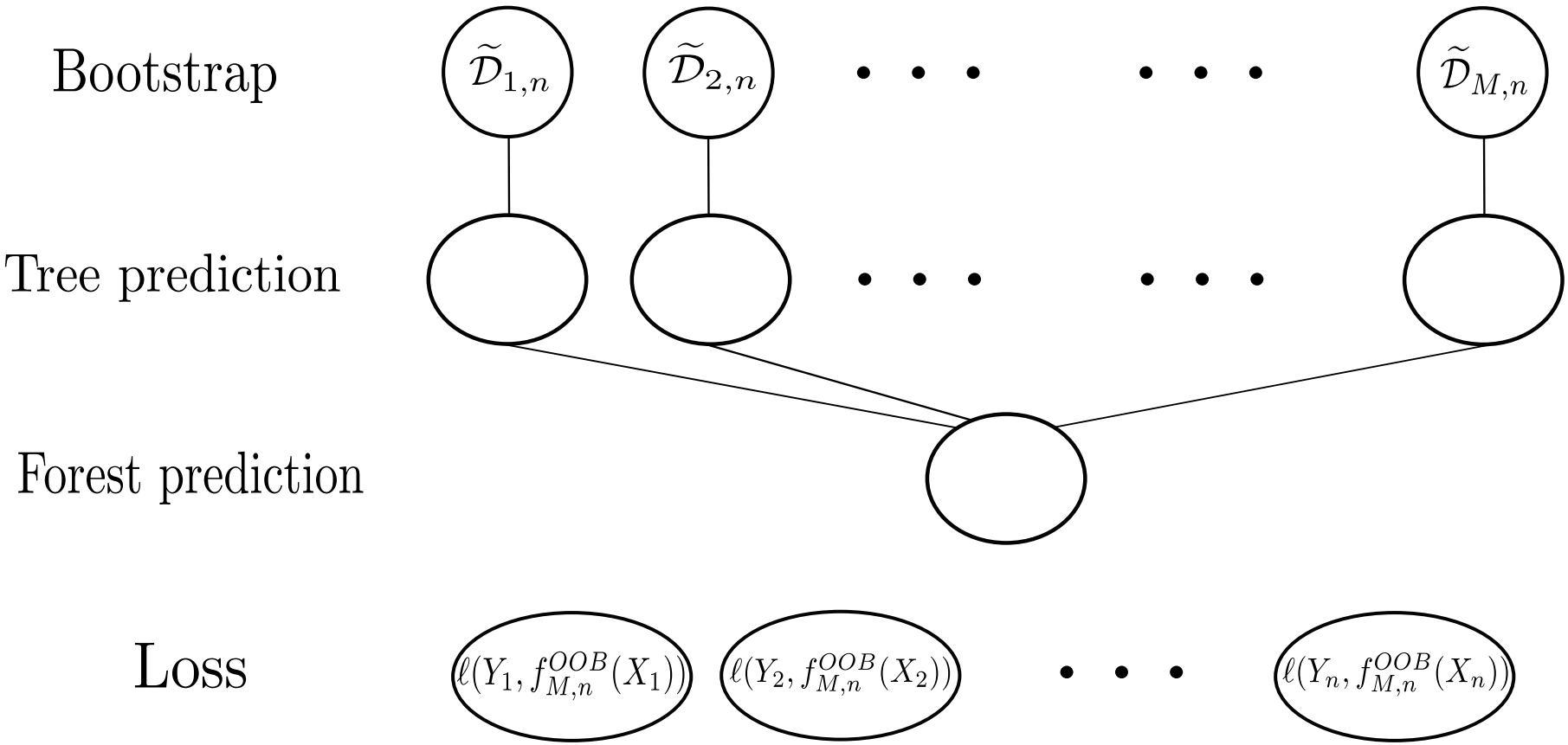
Out-of-bag procedure



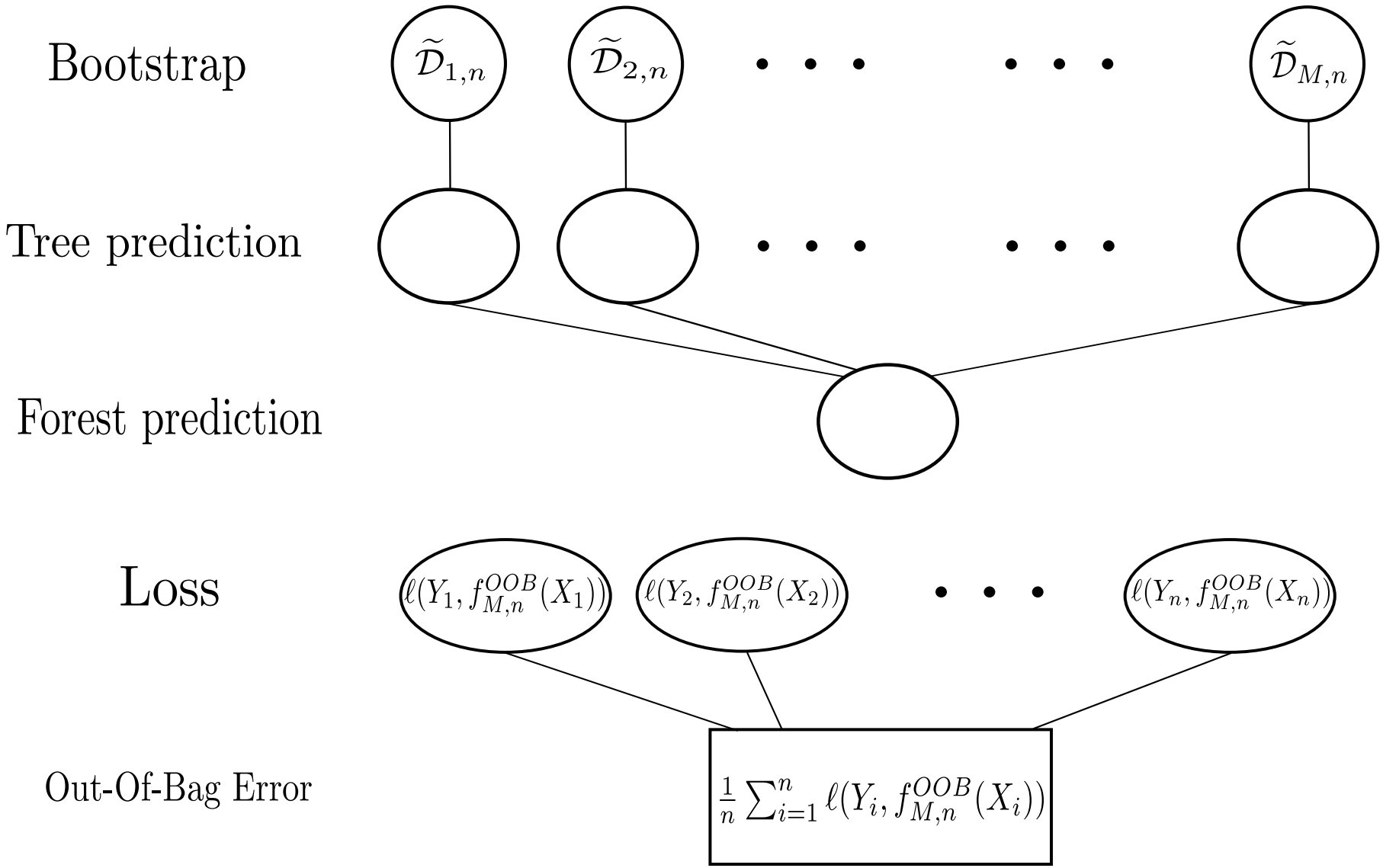
Out-of-bag procedure



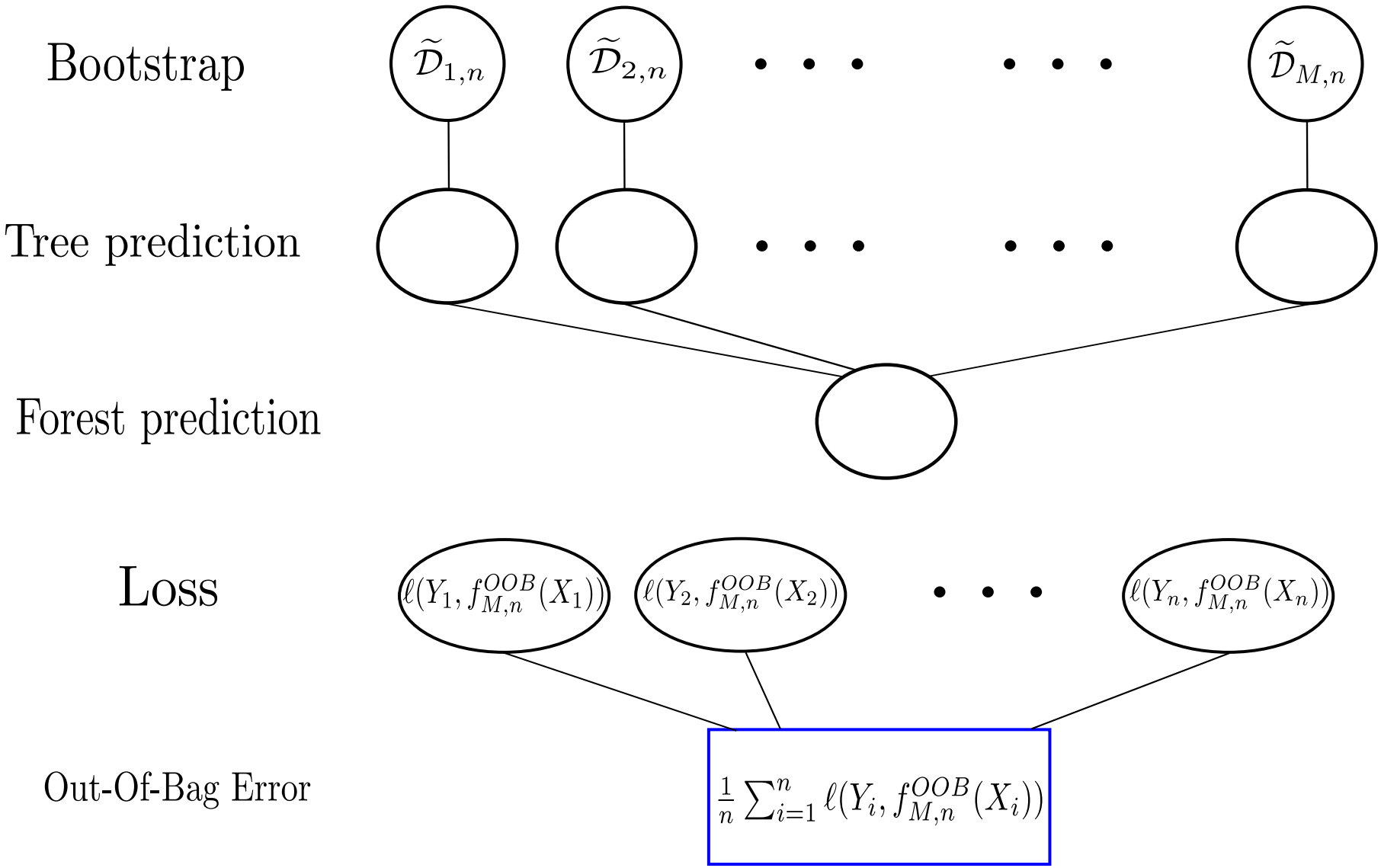
Out-of-bag procedure



Out-of-bag procedure



Out-of-bag procedure



Out-of-bag (detailed procedure)

Consider that a forest has been trained on the data set $\mathcal{D}_n$.

- For each observation $i \in \{1, \ldots, n\}$,
  - Consider the bootstrap samples that do not contain this observation, that is the set $\Lambda_{i,n} = \{m, (X_i, Y_i) \notin \widetilde{\mathcal{D}}_{m,n}\}$
  - For the trees that are not built using observation i, compute the prediction at X_i and aggregate them as

$$f_{M,n}^{(OOB)}(X_i) = \frac{1}{|\Lambda_{i,n}|} \sum_{m \in \Lambda_{n,i}} t_n(X_i, \Theta_m) \mathbb{1}_{|\Lambda_{n,i}|>0} \quad \text{in regression,}$$

$$= \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \sum_{\ell \in \Lambda_{n,i}} \mathbb{1}_{t_n(X_i,\Theta_\ell)=k} \quad \text{in classification.}$$

  - Compute the associated loss

$$\ell(f_{M,n}^{(OOB)}(X_i), Y_i) = (f_{M,n}^{(OOB)}(X_i) - Y_i)^2 \quad \text{in regression,}$$

$$= \ell(f_{M,n}^{(OOB)}(X_i), Y_i) = \mathbb{1}_{f_{M,n}^{(OOB)}(X_i) \neq Y_i} \text{in classification.}$$

- Compute the Out-of-bag error by averaging the losses over all observations, that is

$$R_n^{OOB} = \frac{1}{n} \sum_{i=1}^{n} \ell(f_{M,n}^{(OOB)}(X_i), Y_i)$$

Variable importance via random forests

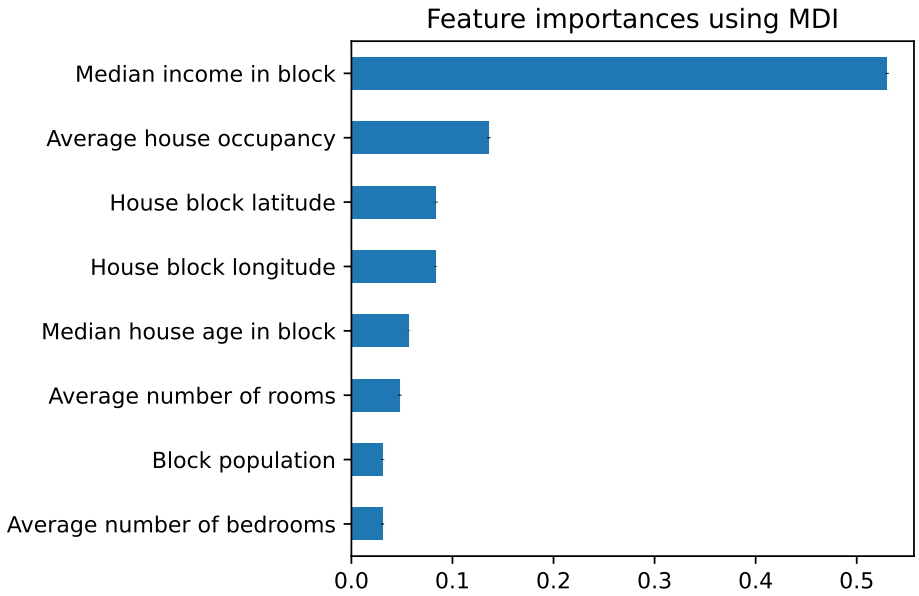


Figure: One of the two variable importance measure, Mean Decrease in Impurity (MDI) computed on the California housing data set.

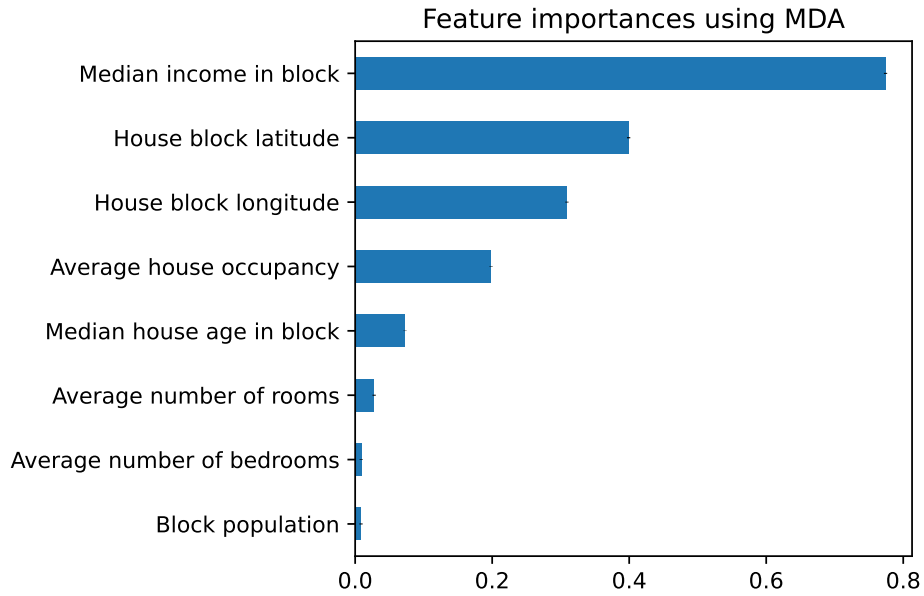


Figure: One of the two variable importance measure, Mean Decrease in Accuracy (MDA) computed on the California housing data set.

Variable importance via random forests

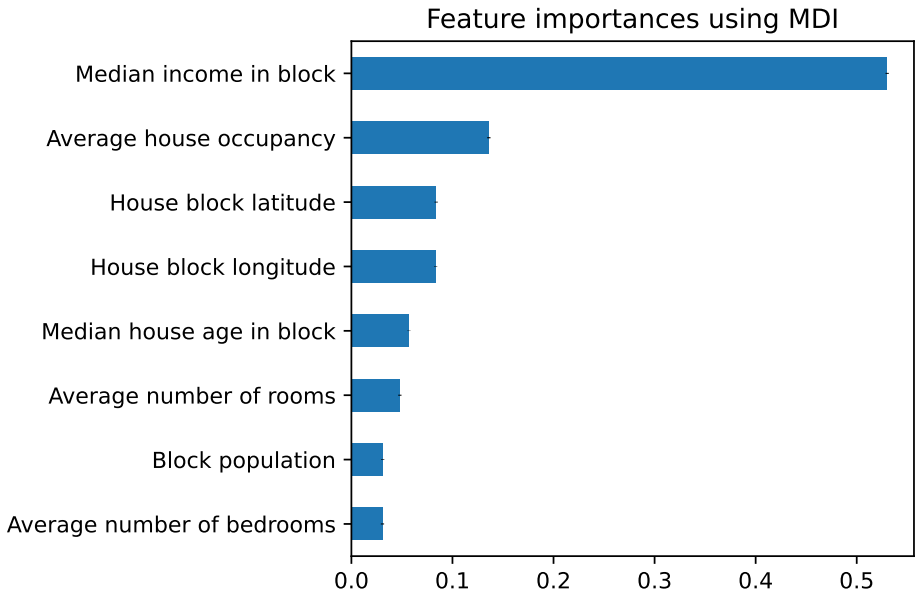


Figure: One of the two variable importance measure, Mean Decrease in Impurity (MDI) computed on the California housing data set.

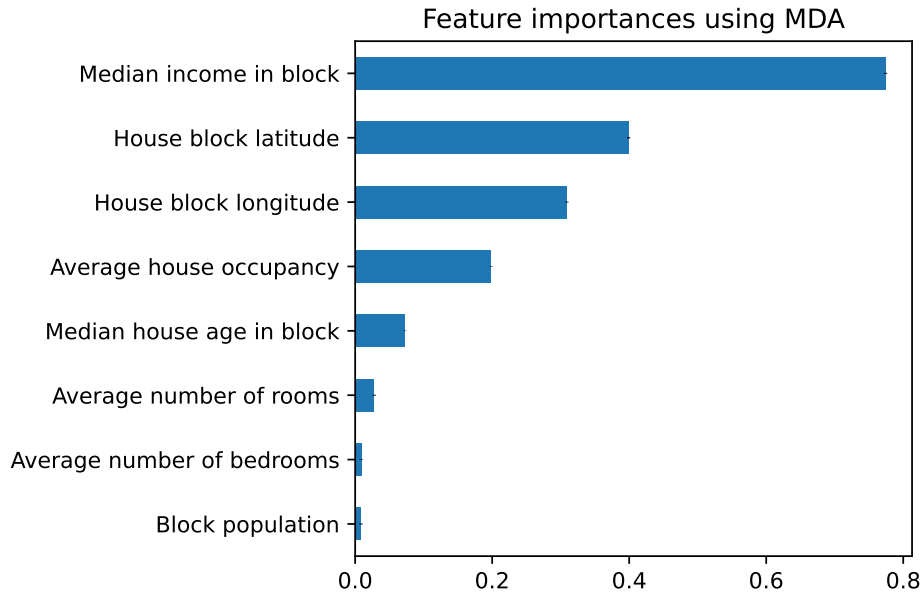


Figure: One of the two variable importance measure, Mean Decrease in Accuracy (MDA) computed on the California housing data set.

- Going beyond prediction to understand the black-box model
- Finding the input variables that are the most "linked" to the output
- Here the variable ranking is not exactly the same across these two different measures.

Variable importance - to what aim?

One single good variable importance measure does not exist. It always depend on what it is used for.

A simple example. Assume that $X \in \mathbb{R}^{10}$, $Y \in \mathbb{R}$ and $Y = X_1$ with $X_1 = g(X_2, \ldots, X_{10})$ for some function g.

- (Variable selection) If one is interested in finding the smallest set of variables leading to good predictive performance, the associated variable importance should be large for X_1 and null for $X_2, \ldots, X_{10}$.
- (Link identification) If one is interested in finding all variables linked to the output, the associated variable importance should be large for $X_1, \ldots, X_d$.

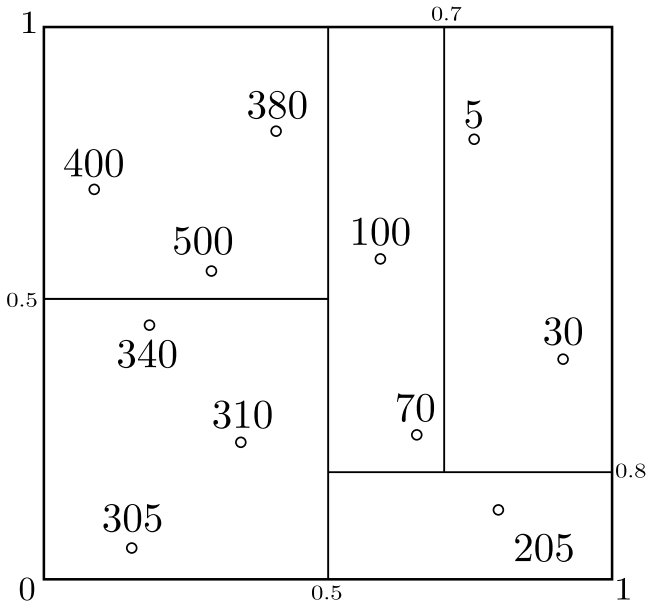
The quality of a variable importance measure depends on its final use (variable selection or link identification).

Variable importance in random forests

Two different measures often computed with random forests:

- Mean Decrease Impurity (MDI) (Breiman, 2002)
  - Tailored for decision tree methods
  - Use the decrease in impurity in each node to compute an aggregated variable importance
- Mean Decrease Accuracy (MDA) (also called *permutation importance*, see Breiman, 2001a)
  - Can be used with any supervised learning algorithm (not tree specific)
  - Permute the values of a given feature in the test set and compare the resulting decrease in predictive performance.

Mean Decrease in impurity

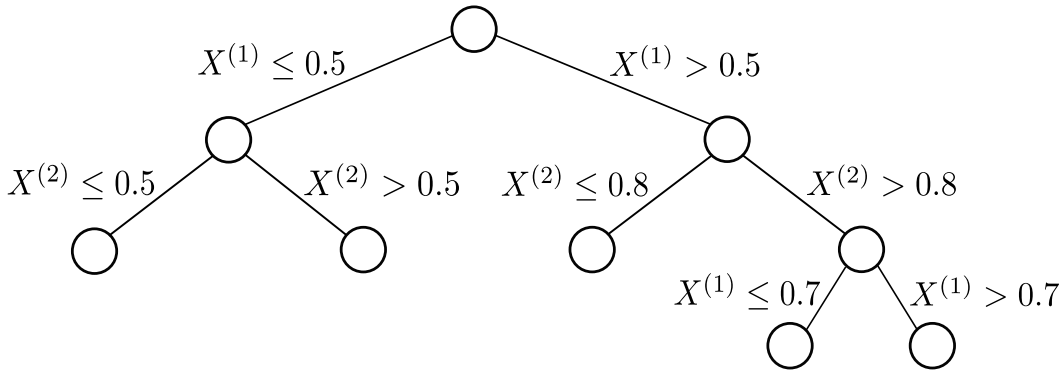


$k = 0$

$k = 1$

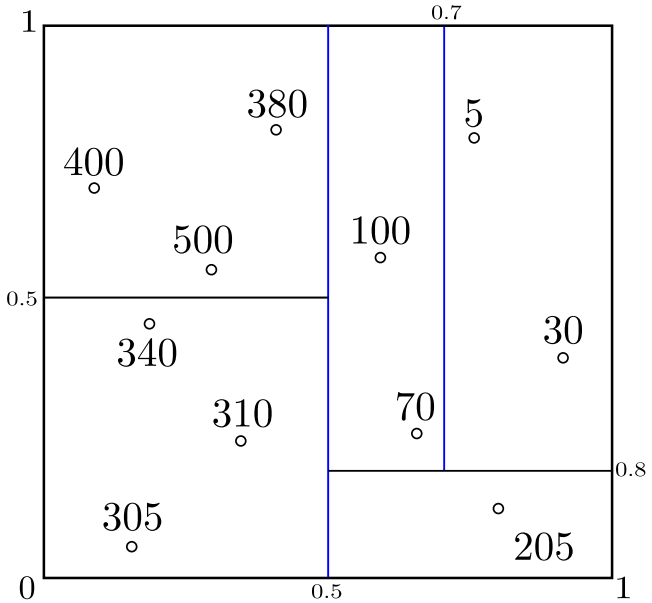
$k = 2$

$k = 3$



For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$.

Mean Decrease in impurity

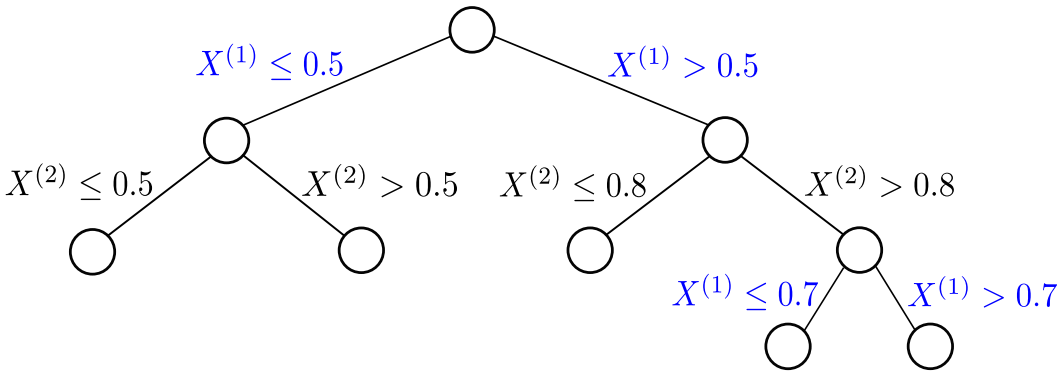


$k = 0$

$k = 1$

$k = 2$

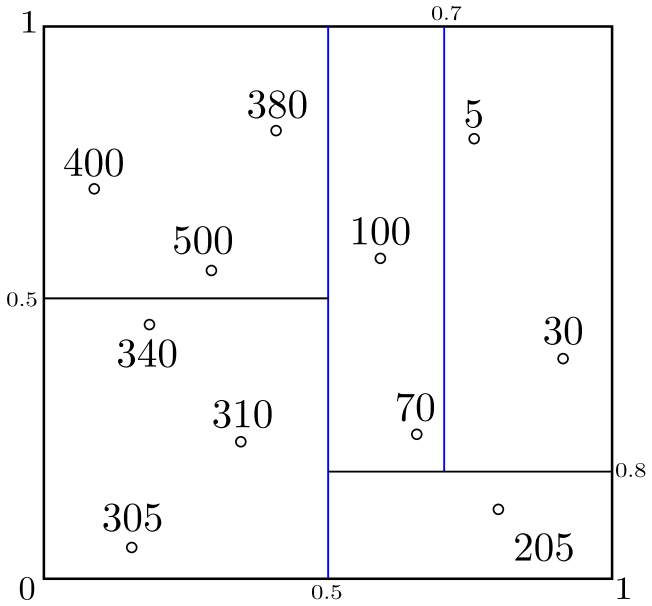
$k = 3$



For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$. We proceed as follows:

- Identify all splits that involve variable $X^{(1)}$

Mean Decrease in impurity

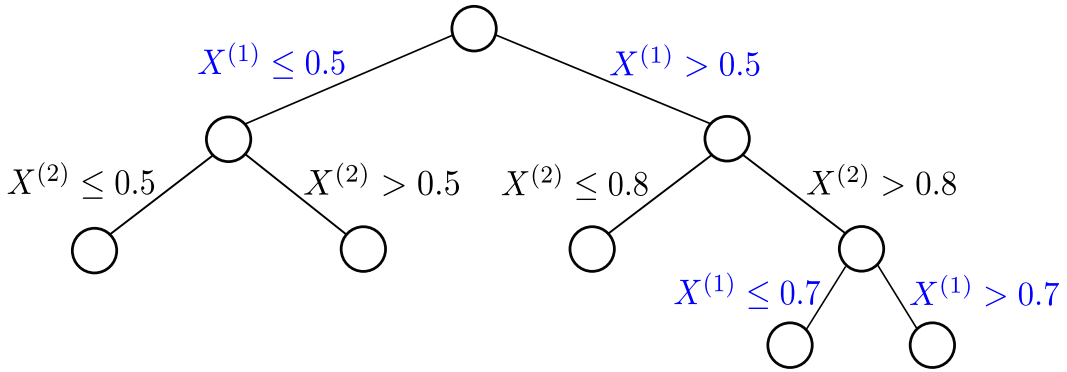


$k = 0$

$k = 1$

$k = 2$

$k = 3$



For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$. We proceed as follows:

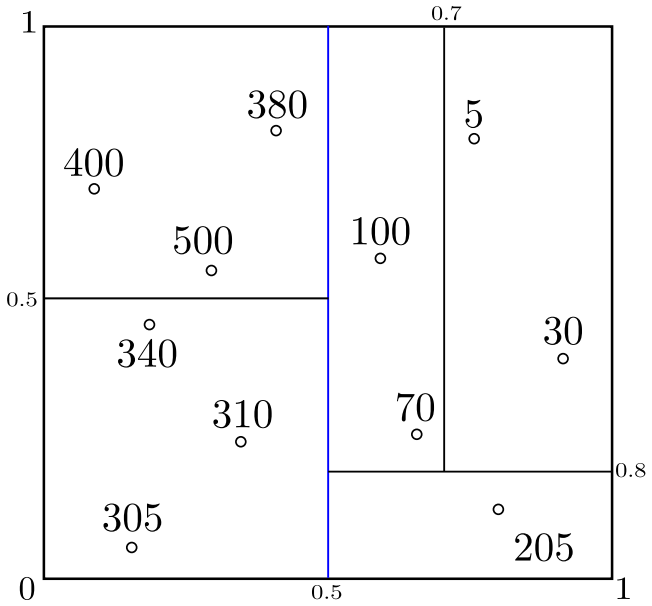
- Identify all splits that involve variable $X^{(1)}$
- For each split, compute the decrease in impurity between the parent node A and the two resulting nodes A_L and A_R:

$$\Delta Imp_n(A) = Imp_n(A) - p_{L,n} Imp_n(A_L) - p_{R,n} Imp_n(A_R),$$

where $p_{L,n}$ (resp. $p_{R,n}$) is the fraction of observations in A that fall into A_L (resp. A_R). For example,

$$Imp_{V,n}(A) = \mathbb{V}_n[Y|X \in A].$$

Mean Decrease in impurity

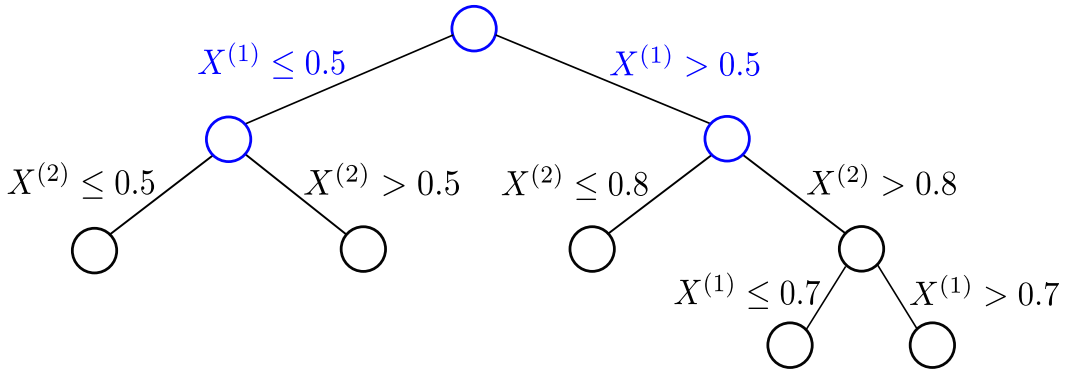


$k = 0$

$k = 1$

$k = 2$

$k = 3$



For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$. We proceed as follows:

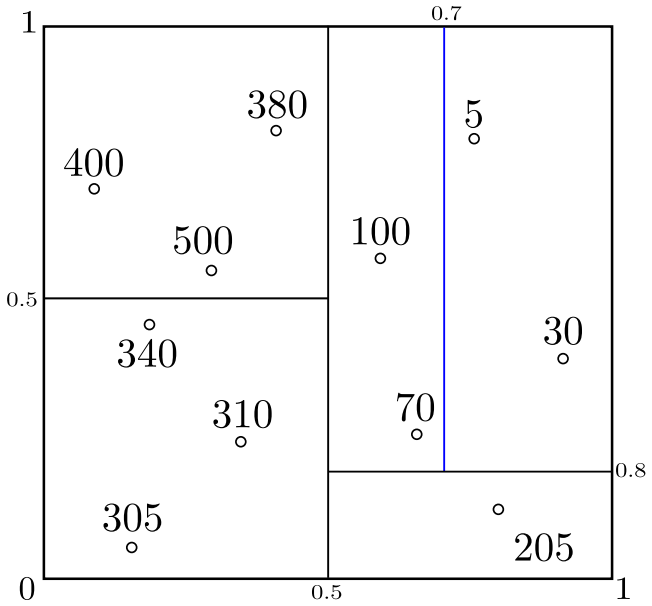
- Identify all splits that involve variable $X^{(1)}$
- For each split, compute the decrease in impurity between the parent node A and the two resulting nodes A_L and A_R:

$$\Delta Imp_n(A) = Imp_n(A) - p_{L,n} Imp_n(A_L) - p_{R,n} Imp_n(A_R),$$

where $p_{L,n}$ (resp. $p_{R,n}$) is the fraction of observations in A that fall into A_L (resp. A_R). For example,

$$Imp_{V,n}(A) = \mathbb{V}_n[Y|X \in A].$$

Mean Decrease in impurity

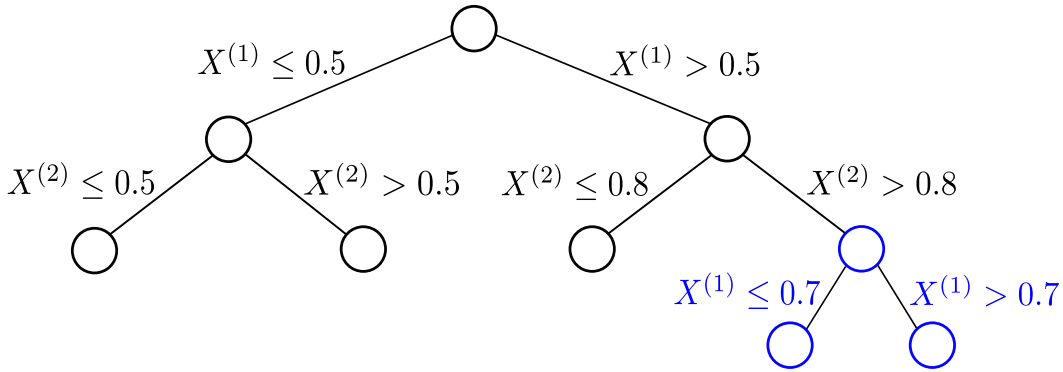


$k = 0$

$k = 1$

$k = 2$

$k = 3$



For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$. We proceed as follows:

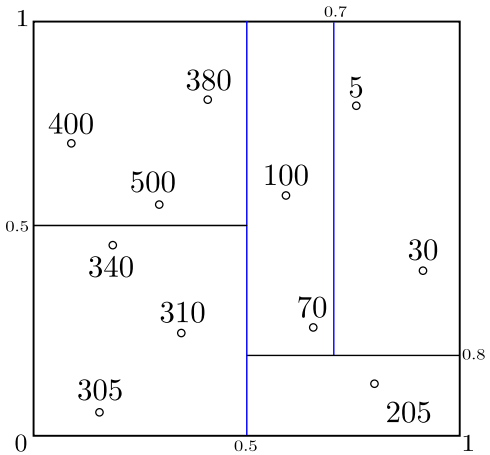
- Identify all splits that involve variable $X^{(1)}$
- For each split, compute the decrease in impurity between the parent node A and the two resulting nodes A_L and A_R:

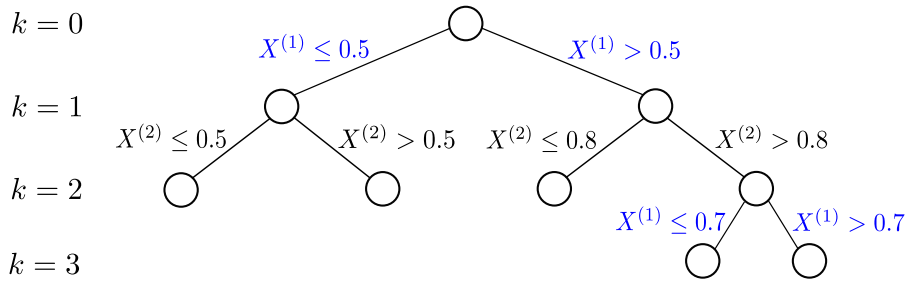
$$\Delta Imp_n(A) = Imp_n(A) - p_{L,n} Imp_n(A_L) - p_{R,n} Imp_n(A_R),$$

where $p_{L,n}$ (resp. $p_{R,n}$) is the fraction of observations in A that fall into A_L (resp. A_R). For example,

$$Imp_{V,n}(A) = \mathbb{V}_n[Y|X \in A].$$

Mean Decrease in impurity





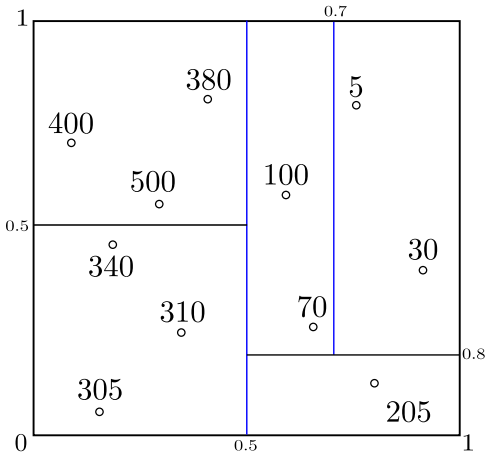
For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$. We proceed as follows:

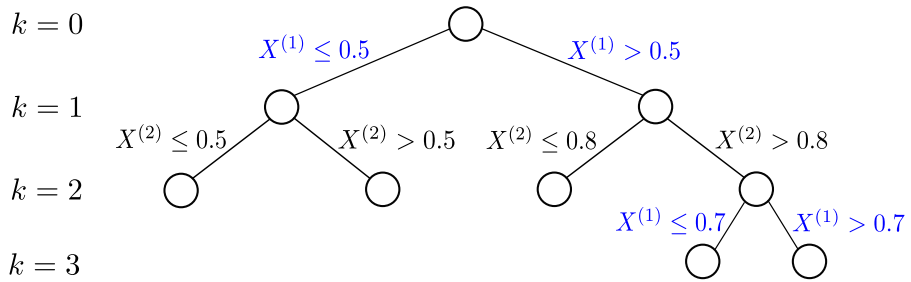
- Identify all splits that involve variable $X^{(1)}$
- For each split, compute the decrease in impurity $\Delta Imp_n(A)$ between the parent node A and the two resulting nodes A_L and A_R
- The MDI of $X^{(1)}$ computed via this tree $\mathcal{T}$ is

$$\widehat{\mathrm{MDI}}_{\mathcal{T}}(X^{(j)}) = \sum_{\substack{A \in \mathcal{T} \\ j_{n,A}=1}} p_{n,A} \; \Delta Imp_n(A), \tag{2}$$

where the sum ranges over all cells A in $\mathcal{T}$ that are split along variable j and $p_{A,n}$ is the fraction of observations falling into A

Mean Decrease in impurity





For this given trained tree $\mathcal{T}$, we want to evaluate the MDI of $X^{(1)}$. We proceed as follows:

- Identify all splits that involve variable $X^{(1)}$
- For each split, compute the decrease in impurity $\Delta Imp_n(A)$ between the parent node A and the two resulting nodes A_L and A_R
- The MDI of $X^{(1)}$ computed via this tree $\mathcal{T}$ is

$$\widehat{\mathrm{MDI}}_{\mathcal{T}}(X^{(j)}) = \sum_{\substack{A \in \mathcal{T} \\ j_{n,A}=1}} p_{n,A} \; \Delta Imp_n(A) \tag{2}$$

- The MDI of $X^{(1)}$ output by a forest is the average of the MDI of $X^{(1)}$ of each tree.

Mean Decrease in Impurity

Pros

- Easily accessible via scikit-learn as the attribute `feature_importances_` of a RandomForest object
- No extra computations needed
- Adapted to the tree building process / the predictor

Cons

- biased towards variables with many categories (see, e.g., Strobl et al., 2007; Nicodemus, 2011), variables that possess high-category frequency (Nicodemus, 2011; Boulesteix et al., 2011), biased in presence of correlated features (Nicodemus and Malley, 2009)
- Bias related to in-sample estimation (Li et al., 2019; Zhou and Hooker, 2019) - Same observations are used to build the tree and estimate the MDI
- Bias related to fully-grown tree
- No information about the quantity it is supposed to estimate!

MDA illustration

Built-in variable importance algorithm for random forests
MDA principle: decrease of accuracy of the forest when a variable is noised up

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

MDA illustration

Built-in variable importance algorithm for random forests
MDA principle: decrease of accuracy of the forest when a variable is noised up

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

MDA illustration

Built-in variable importance algorithm for random forests
MDA principle: decrease of accuracy of the forest when a variable is noised up

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

MDA illustration

Built-in variable importance algorithm for random forests
MDA principle: decrease of accuracy of the forest when a variable is noised up

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

quadratic error $= 13.7$ quadratic error $= 16.4$

$$\text{MDA}(X^{(j)}) = 16.4 - 13.7 = 2.7$$

MDA illustration

Built-in variable importance algorithm for random forests
MDA principle: decrease of accuracy of the forest when a variable is noised up

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

quadratic error $= 13.7$ quadratic error $= 16.4$

$$\text{MDA}(X^{(j)}) = 16.4 - 13.7 = 2.7$$

- $\text{MDA}(X^{(j)}) = 0 \longrightarrow$ no influence of $X^{(j)}$
- $\text{MDA}(X^{(j)})$ is high $\longrightarrow$ strong influence of $X^{(j)}$

MDA illustration

Built-in variable importance algorithm for random forests
MDA principle: decrease of accuracy of the forest when a variable is noised up

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

quadratic error $= 13.7$ quadratic error $= 16.4$

$$\text{MDA}(X^{(j)}) = 16.4 - 13.7 = 2.7$$

$\mathcal{D}_n$ used to fit the forest and compute accuracy: overfitting and inflated accuracy

Mean Decrease in Accuracy

Pros

- Can be applied to any machine learning algorithm via the function `permutation-importance` in scikit-learn
- Fast to compute (no need to retrain a forest)

Cons

- Biased in presence of correlation
- Break the links between the given covariate and the output but also between the given covariate and the other covariates

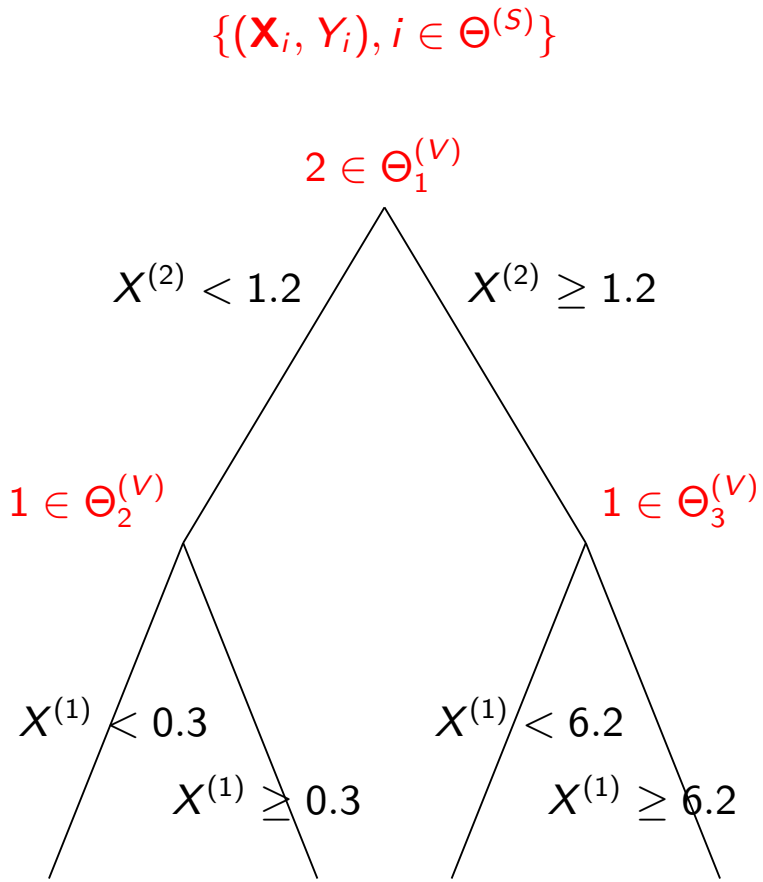
Random forests

- Regression setting
  - input vector $\mathbf{X} = (X^{(1)}, \ldots, X^{(p)}) \in \mathbb{R}^p$
  - output $Y \in \mathbb{R}$
  - dataset $\mathcal{D}_n = \{(\mathbf{X}_i, Y_i), i = 1, \ldots, n\}$, where $(\mathbf{X}_i, Y_i) \sim \mathbb{P}_{\mathbf{X},Y}$.

Random forests

- Regression setting
  - input vector $\mathbf{X} = (X^{(1)}, \ldots, X^{(p)}) \in \mathbb{R}^p$
  - output $Y \in \mathbb{R}$
  - dataset $\mathcal{D}_n = \{(\mathbf{X}_i, Y_i), i = 1, \ldots, n\}$, where $(\mathbf{X}_i, Y_i) \sim \mathbb{P}_{\mathbf{X},Y}$.

- Random forest algorithm
  - Aggregation of Θ-random trees $\Theta = (\Theta^{(S)}, \Theta^{(V)})$
  - M: number of trees
  - $m_{M,n}(\mathbf{X}, \mathbf{\Theta}_M)$: the forest estimate at $\mathbf{X}$



MDA illustration

$X^{(1)}$	$X^{(2)}$	$\dots$	$X^{(j)}$	$\dots$	$X^{(p)}$	Y
2.1	4.3	$\dots$	0.1	$\dots$	2.6	2.3
1.7	4.1	$\dots$	9.2	$\dots$	3.8	0.4
3.4	9.2	$\dots$	3.2	$\dots$	3.6	10.2
5.6	1.2	$\dots$	8.2	$\dots$	4.2	9.1
8.9	6.8	$\dots$	6.7	$\dots$	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

MDA illustration

$X^{(1)}$	$X^{(2)}$	$\dots$	$X^{(j)}$	$\dots$	$X^{(p)}$	Y
2.1	4.3	$\dots$	0.1	$\dots$	2.6	2.3
1.7	4.1	$\dots$	9.2	$\dots$	3.8	0.4
3.4	9.2	$\dots$	3.2	$\dots$	3.6	10.2
5.6	1.2	$\dots$	8.2	$\dots$	4.2	9.1
8.9	6.8	$\dots$	6.7	$\dots$	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

MDA illustration

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

MDA illustration

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

Explained variance of Y $= 16.4$

Explained variance of Y $= 13.7$

$$\text{MDA}(X^{(j)}) = 16.4 - 13.7 = 2.7$$

MDA illustration

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	0.1	...	2.6	2.3
1.7	4.1	...	9.2	...	3.8	0.4
3.4	9.2	...	3.2	...	3.6	10.2
5.6	1.2	...	8.2	...	4.2	9.1
8.9	6.8	...	6.7	...	2.9	4.5

$X^{(1)}$	$X^{(2)}$	...	$X^{(j)}$	...	$X^{(p)}$	Y
2.1	4.3	...	6.7	...	2.6	2.3
1.7	4.1	...	3.2	...	3.8	0.4
3.4	9.2	...	9.2	...	3.6	10.2
5.6	1.2	...	0.1	...	4.2	9.1
8.9	6.8	...	8.2	...	2.9	4.5

Table: Example of the permutation of a dataset $\mathcal{D}_n$ for $n = 5$.

Question: Can I use $\mathcal{D}_n$ to both fit the forest and compute accuracy ?

No: overfitting and inflated accuracy.

How to handle this in practice?

MDA versions

The explained variance estimate of MDA algorithms differ across implementations

Train-Test MDA: train data to fit the forest, and test data for accuracy

MDA versions

The explained variance estimate of MDA algorithms differ across implementations

Train-Test MDA: train data to fit the forest, and test data for accuracy

Out-of-bag (OOB) samples: $\mathcal{D}_n$ is bootstrap prior to the construction of each tree, leaving aside a portion of $\mathcal{D}_n$, which is not involved in the tree growing and defines the "out-of-bag" sample.

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

Selected samples: $\Theta_\ell^{(S)} = \{1, 3, 4\}$

MDA versions

The explained variance estimate of MDA algorithms differ across implementations

Train-Test MDA: train data to fit the forest, and test data for accuracy

Out-of-bag (OOB) samples: $\mathcal{D}_n$ is bootstrap prior to the construction of each tree, leaving aside a portion of $\mathcal{D}_n$, which is not involved in the tree growing and defines the "out-of-bag" sample.

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

OOB samples: $\{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$

MDA versions

The explained variance estimate of MDA algorithms differ across implementations

Train-Test MDA: train data to fit the forest, and test data for accuracy

Out-of-bag (OOB) samples: $\mathcal{D}_n$ is bootstrap prior to the construction of each tree, leaving aside a portion of $\mathcal{D}_n$, which is not involved in the tree growing and defines the "out-of-bag" sample.

MDA Version	Package	Error	Data
Train-Test	`scikit-learn` `randomForestSRC`	Forest	Testing dataset
Breiman-Cutler	`randomForest` (normalized) `ranger` / `randomForestSRC`	Tree	OOB sample
Ishwaran-Kogalur	`randomForestSRC`	Forest	OOB sample

Table: Summary of the different MDA algorithms.

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^{n} \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^{n} \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree
- $\mathbf{X}_{i,\pi_{j\ell}}$: i-th observation where the j-th component is permuted across the OOB sample of the ℓ-th tree

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	9.2	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	6.7	$\ldots$	2.9	4.5

$\mathbf{X}_i$

$X^{(1)}$	$X^{(2)}$	$\ldots$	$X^{(j)}$	$\ldots$	$X^{(p)}$	Y
2.1	4.3	$\ldots$	0.1	$\ldots$	2.6	2.3
1.7	4.1	$\ldots$	6.7	$\ldots$	3.8	0.4
3.4	9.2	$\ldots$	3.2	$\ldots$	3.6	10.2
5.6	1.2	$\ldots$	8.2	$\ldots$	4.2	9.1
8.9	6.8	$\ldots$	9.2	$\ldots$	2.9	4.5

$\mathbf{X}_{i,\pi_{j\ell}}$

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^n \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree
- $\mathbf{X}_{i,\pi_{j\ell}}$: i-th observation where the j-th component is permuted across the OOB sample of the ℓ-th tree

$$\widehat{\mathrm{MDA}}_{M,n}^{(BC)}(X^{(j)}) = \frac{1}{M} \sum_{\ell=1}^{M} \frac{1}{N_{n,\ell}} \sum_{i=1}^{n} \Big[\big(Y_i - m_n(\mathbf{X}_{i,\pi_{j\ell}}, \Theta_\ell)\big)^2 - \big(Y_i - m_n(\mathbf{X}_i, \Theta_\ell)\big)^2 \Big] \mathbb{1}_{i \notin \Theta_\ell^{(S)}}$$

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^n \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree
- $\mathbf{X}_{i,\pi_{j\ell}}$: i-th observation where the j-th component is permuted across the OOB sample of the ℓ-th tree

$$\widehat{\mathrm{MDA}}_{M,n}^{(BC)}(X^{(j)}) = \frac{1}{M} \sum_{\ell=1}^{M} \frac{1}{N_{n,\ell}} \sum_{i=1}^{n} \left[(Y_i - m_n(\mathbf{X}_{i,\pi_{j\ell}}, \Theta_\ell))^2 - (Y_i - m_n(\mathbf{X}_i, \Theta_\ell))^2 \right] \mathbb{1}_{i \notin \Theta_\ell^{(S)}}$$

Quadratic risk of the ℓ-th tree

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^n \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree
- $\mathbf{X}_{i,\pi_{j\ell}}$: i-th observation where the j-th component is permuted across the OOB sample of the ℓ-th tree

$$\widehat{\mathrm{MDA}}_{M,n}^{(BC)}(X^{(j)}) = \frac{1}{M} \sum_{\ell=1}^{M} \frac{1}{N_{n,\ell}} \sum_{i=1}^{n} \Big[\big(Y_i - m_n(\mathbf{X}_{i,\pi_{j\ell}}, \Theta_\ell)\big)^2 - \big(Y_i - m_n(\mathbf{X}_i, \Theta_\ell)\big)^2 \Big] \mathbb{1}_{i \notin \Theta_\ell^{(S)}}$$

Inflated quadratic risk of the ℓ-th tree where $X^{(j)}$ is permuted

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^n \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree
- $\mathbf{X}_{i,\pi_{j\ell}}$: i-th observation where the j-th component is permuted across the OOB sample of the ℓ-th tree

$$\widehat{\mathrm{MDA}}_{M,n}^{(BC)}(X^{(j)}) = \frac{1}{M} \sum_{\ell=1}^{M} \frac{1}{N_{n,\ell}} \sum_{i=1}^{n} \big[(Y_i - m_n(\mathbf{X}_{i,\pi_{j\ell}}, \Theta_\ell))^2 - (Y_i - m_n(\mathbf{X}_i, \Theta_\ell))^2 \big] \mathbb{1}_{i \notin \Theta_\ell^{(S)}}$$

Risks are computed over the OOB sample of each tree

Breiman-Cutler MDA

- $i \in \{1, \ldots, n\} \setminus \Theta_\ell^{(S)} = \{2, 5\}$: OOB sample of the ℓ-th tree
- $N_{n,\ell} = \sum_{i=1}^n \mathbb{1}_{i \neq \Theta_\ell^{(S)}} = 2$: size of the OOB sample of the ℓ-th tree
- $\mathbf{X}_{i,\pi_{j\ell}}$: i-th observation where the j-th component is permuted across the OOB sample of the ℓ-th tree

$$\widehat{\mathrm{MDA}}_{M,n}^{(BC)}(X^{(j)}) = \frac{1}{M} \sum_{\ell=1}^{M} \frac{1}{N_{n,\ell}} \sum_{i=1}^{n} \Big[\big(Y_i - m_n(\mathbf{X}_{i,\pi_{j\ell}}, \Theta_\ell)\big)^2 - \big(Y_i - m_n(\mathbf{X}_i, \Theta_\ell)\big)^2 \Big] \mathbb{1}_{i \notin \Theta_\ell^{(S)}}$$

Average over all trees

Assumptions

(A1)

The response $Y \in \mathbb{R}$ *follows*

$$Y = m(\mathbf{X}) + \varepsilon$$

where

- $\mathbf{X} = (X^{(1)}, \ldots, X^{(p)}) \in [0,1]^p$
- $\mathbf{X}$ *admits a density* f *such that* $c_1 < f(\mathbf{x}) < c_2$, *with constants* $c_1, c_2 > 0$
- m *is continuous*
- *the noise* ε *is sub-Gaussian and centered*

Assumptions

(A2): the theoretical tree is consistent
(always true with slight modifications of the forest algorithm)

Assumptions

(A2): the theoretical tree is consistent
(always true with slight modifications of the forest algorithm)

(A2)

The randomized theoretical CART tree built with the distribution of $(\mathbf{X}, Y)$ *is consistent, that is, for all* $\mathbf{x} \in [0,1]^p$*, almost surely,*

$$\lim_{k \to \infty} \Delta(m, A_k^\star(\mathbf{x}, \Theta)) = 0.$$

Assumptions

(A2): the theoretical tree is consistent
(always true with slight modifications of the forest algorithm)

(A2)

The randomized theoretical CART tree built with the distribution of $(\mathbf{X}, Y)$ *is consistent, that is, for all* $\mathbf{x} \in [0,1]^p$*, almost surely,*

$$\lim_{k \to \infty} \Delta(m, A_k^\star(\mathbf{x}, \Theta)) = 0.$$

(A3): tree partition is not too complex with respect to n

Assumptions

(A2): the theoretical tree is consistent
(always true with slight modifications of the forest algorithm)

(A2)

The randomized theoretical CART tree built with the distribution of $(\mathbf{X}, Y)$ *is consistent, that is, for all* $\mathbf{x} \in [0,1]^p$*, almost surely,*

$$\lim_{k \to \infty} \Delta(m, A_k^\star(\mathbf{x}, \Theta)) = 0.$$

(A3): tree partition is not too complex with respect to n

(A3)

The asymptotic regime of a_n*, the size of the subsampling without replacement, and the number of terminal leaves* t_n *is such that* $a_n \leq n-2$*,* $a_n/n < 1-\kappa$ *for a fixed* $\kappa > 0$*,* $\lim_{n\to\infty} a_n = \infty$*,* $\lim_{n\to\infty} t_n = \infty$*, and* $\lim_{n\to\infty} t_n \frac{(\log(a_n))^9}{a_n} = 0$*.*

MDA Convergence

Theorem (Bénard, Da Veiga, et al. (2021))

If Assumptions (A1), (A2), and (A3) are satisfied, then, for all $M \in \mathbb{N}^\star$ *and* $j \in \{1, \ldots, p\}$ *we have*

$$\widehat{MDA}^{(BC)}_{M,n}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{E}[(m(\mathbf{X}) - m(\mathbf{X}_{\pi_j}))^2]$$

$\mathbf{X}_{\pi_j}$: $\mathbf{X}$ where the j-th component is replaced by an independent copy, i.e. $\mathbf{X}_{\pi_j} = (X^{(1)}, \ldots, X'^{(j)}, \ldots, X^{(p)})$

Limit interpretation?

Sensitivity analysis

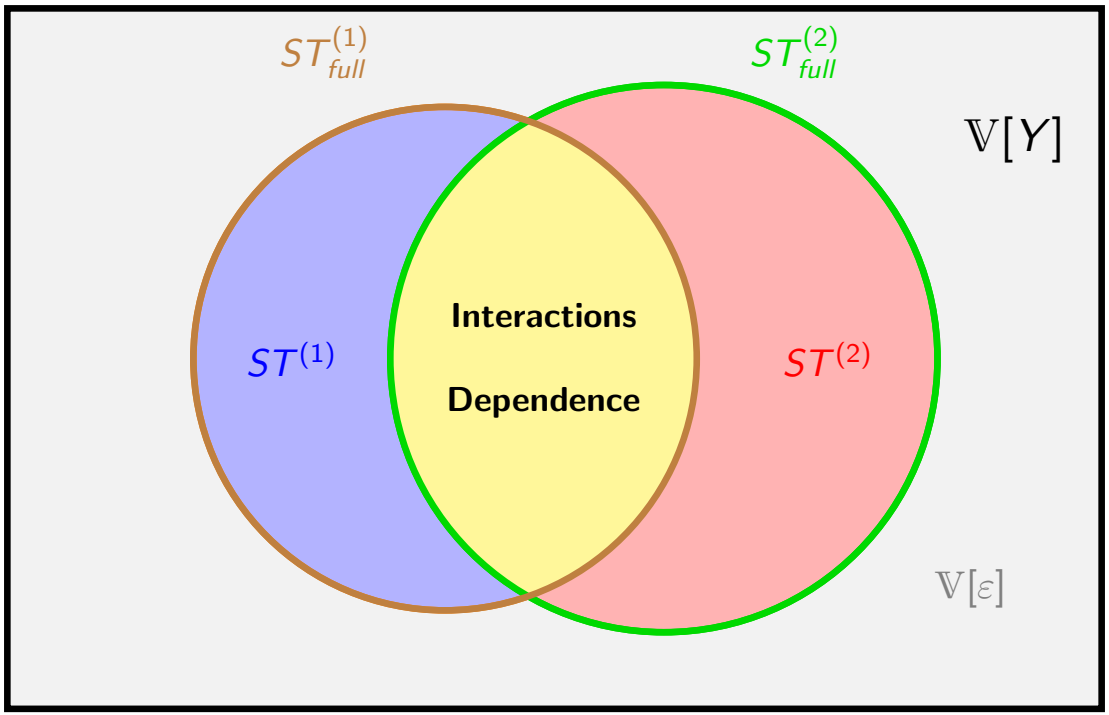


Figure: Standard and full total Sobol indices for $Y = m(X^{(1)}, X^{(2)}) + \varepsilon$.

Total Sobol index (Sobol, 1993)

$$ST^{(1)} = \frac{\mathbb{E}[\mathbb{V}(m(\mathbf{X})|\mathbf{X}^{(-1)})]}{\mathbb{V}(Y)}$$

Full total Sobol index (Mara et al., 2015; Benoumechiara, 2019)

$$ST^{(1)}_{full} = \frac{\mathbb{E}[\mathbb{V}(m(\mathbf{X}_{\pi_1})|\mathbf{X}^{(-1)})]}{\mathbb{V}(Y)}$$

MDA Decomposition

Proposition (Bénard, Da Veiga, et al. (2021))

If Assumptions (A1), (A2) and (A3) are satisfied, then for all $M \in \mathbb{N}^\star$ and $j \in \{1, \ldots, p\}$ we have

$$\widehat{MDA}_{M,n}^{(BC)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times ST^{(j)} + \mathbb{V}[Y] \times ST_{full}^{(j)} + MDA_3^{\star(j)}.$$

The term $\mathrm{MDA}_3^{\star(j)}$ is not an importance measure and is defined by

$$\mathrm{MDA}_3^{\star(j)} = \mathbb{E}[(\mathbb{E}[m(\mathbf{X})|\mathbf{X}^{(-j)}] - \mathbb{E}[m(\mathbf{X}_{\pi_j})|\mathbf{X}^{(-j)}])^2].$$

MDA Decomposition

Proposition (Bénard, Da Veiga, et al. (2021))

If Assumptions (A1), (A2) and (A3) are satisfied, then for all $M \in \mathbb{N}^\star$ *and* $j \in \{1, \ldots, p\}$ *we have*

$$(i) \quad \widehat{MDA}^{(TT)}_{M,n}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times S T^{(j)} + \mathbb{V}[Y] \times S T^{(j)}_{full} + MDA^{\star(j)}_3$$

$$(ii) \quad \widehat{MDA}^{(BC)}_{M,n}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times S T^{(j)} + \mathbb{V}[Y] \times S T^{(j)}_{full} + MDA^{\star(j)}_3.$$

If additionally $M \longrightarrow \infty$, *then*

$$(iii) \quad \widehat{MDA}^{(IK)}_{M,n}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times S T^{(j)} + MDA^{\star(j)}_3.$$

Independent inputs

If inputs X are independent: $\mathrm{MDA}_3^{\star(j)} = 0$ and $ST^{(j)} = ST_{full}^{(j)}$.

Corollary (Bénard, Da Veiga, et al. (2021))

If $\mathbf{X}$ *has independent components, and if Assumptions (A1)-(A3) are satisfied, for all* $M \in \mathbb{N}^\star$ *and* $j \in \{1, \ldots, p\}$ *we have*

$$\widehat{MDA}_{M,n}^{(TT)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} 2\mathbb{V}[Y] \times ST^{(j)}$$
$$\widehat{MDA}_{M,n}^{(BC)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} 2\mathbb{V}[Y] \times ST^{(j)}.$$

If additionally $M \longrightarrow \infty$, *then*

$$\widehat{MDA}_{M,n}^{(IK)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times ST^{(j)}.$$

This Corollary completes the result from (Gregorutti, 2015).

Additive regression function

If m is additive: $\mathrm{MDA}_3^{\star(j)} = 0$.

Corollary (Bénard, Da Veiga, et al. (2021))

If the regression function m is additive, and if Assumptions (A1)-(A3) are satisfied, for all $M \in \mathbb{N}^\star$ and $j \in \{1, \ldots, p\}$ we have

$$\widehat{MDA}_{M,n}^{(TT)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times ST^{(j)} + \mathbb{V}[Y] \times ST_{full}^{(j)}$$

$$\widehat{MDA}_{M,n}^{(BC)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times ST^{(j)} + \mathbb{V}[Y] \times ST_{full}^{(j)}.$$

If additionally $M \longrightarrow \infty$, then

$$\widehat{MDA}_{M,n}^{(IK)}(X^{(j)}) \xrightarrow{\mathbb{L}^1} \mathbb{V}[Y] \times ST^{(j)}.$$

MDA summary

- When inputs $\mathbf{X}$ are dependent and have interactions, the MDA is artificially inflated by the term MDA_3 and is therefore misleading.

MDA summary

- When inputs $\mathbf{X}$ are dependent and have interactions, the MDA is artificially inflated by the term MDA_3 and is therefore misleading.
- MDA versions have different theoretical counterparts, and thus different meanings: be careful when using forest packages !

MDA summary

- When inputs $\mathbf{X}$ are dependent and have interactions, the MDA is artificially inflated by the term MDA_3 and is therefore misleading.
- MDA versions have different theoretical counterparts, and thus different meanings: be careful when using forest packages !
- For variable selection, the total Sobol index is the relevant component

$$\mathbb{V}[Y] \times ST^{(j)} + \cancel{\mathbb{V}[Y] \times ST^{(j)}_{full}} + \cancel{\mathrm{MDA}_3^{\star(j)}}$$

MDA summary

- When inputs $\mathbf{X}$ are dependent and have interactions, the MDA is artificially inflated by the term MDA_3 and is therefore misleading.
- MDA versions have different theoretical counterparts, and thus different meanings: be careful when using forest packages !
- For variable selection, the total Sobol index is the relevant component

$$\mathbb{V}[Y] \times ST^{(j)} + \cancel{\mathbb{V}[Y] \times ST^{(j)}_{full}} + \cancel{\mathrm{MDA}_3^{\star(j)}}$$

- We develop the Sobol-MDA: a fast and consistent estimate of $ST^{(j)}$ for random forests

Sobol-MDA

Principle: **project** the partition of each tree along the j-th direction to remove $X^{(j)}$ from the prediction process.

Sobol-MDA

Principle: **project** the partition of each tree along the j-th direction to remove $X^{(j)}$ from the prediction process.

$$\widehat{\text{S-MDA}}_{M,n}(X^{(j)}) = \frac{1}{\hat{\sigma}^2_Y} \frac{1}{n} \sum_{i=1}^{n} \left[Y_i - m_{M,n}^{(-j,OOB)}(\mathbf{X}_i^{(-j)}, \boldsymbol{\Theta}_M) \right]^2 - \left[Y_i - m_{M,n}^{(OOB)}(\mathbf{X}_i, \boldsymbol{\Theta}_M) \right]^2$$

Sobol-MDA

Principle: **project** the partition of each tree along the j-th direction to remove $X^{(j)}$ from the prediction process.

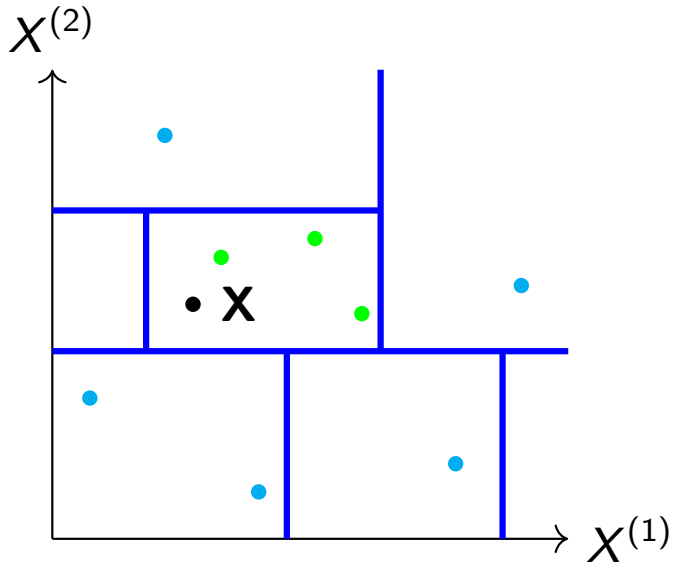


Figure: Partition of $[0,1]^2$ by a random tree (left side) projected on the subspace span by $\mathbf{X}^{(-2)} = X^{(1)}$ (right side), for $p = 2$ and $j = 2$.

$$\widehat{\text{S-MDA}}_{M,n}(X^{(j)}) = \frac{1}{\hat{\sigma}^2_Y} \frac{1}{n} \sum_{i=1}^{n} \left[Y_i - m_{M,n}^{(-j,OOB)}(\mathbf{X}_i^{(-j)}, \boldsymbol{\Theta}_M) \right]^2 - \left[Y_i - m_{M,n}^{(OOB)}(\mathbf{X}_i, \boldsymbol{\Theta}_M) \right]^2$$

Sobol-MDA

Principle: **project** the partition of each tree along the j-th direction to remove $X^{(j)}$ from the prediction process.

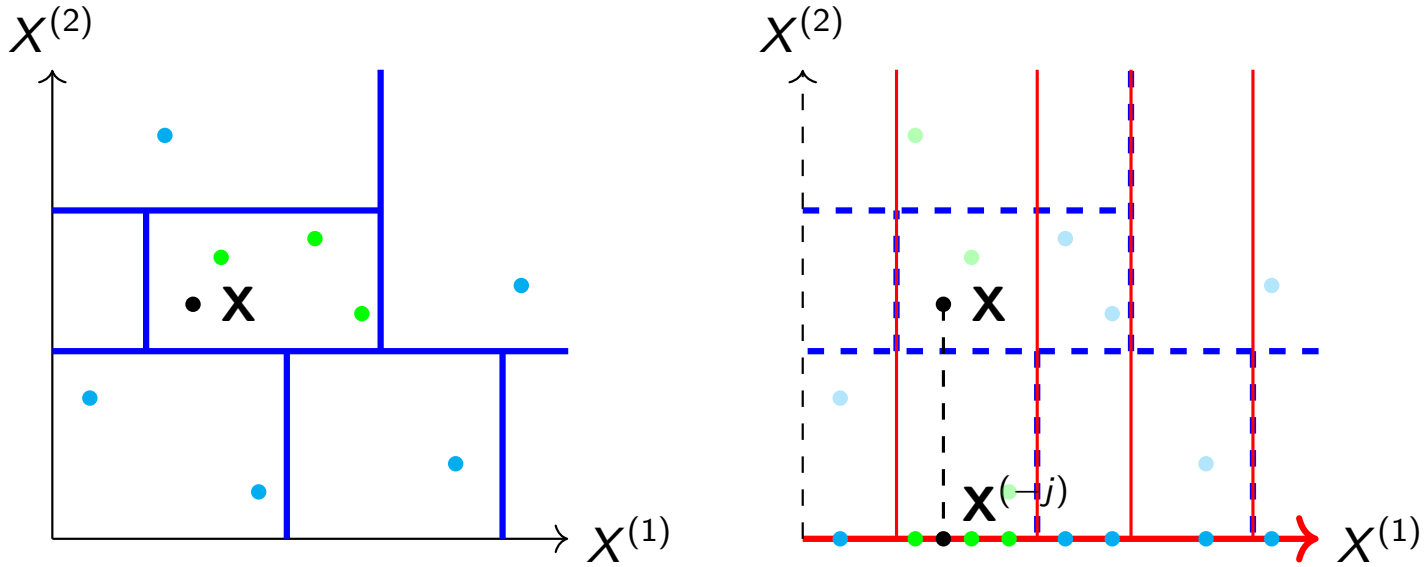


Figure: Partition of $[0,1]^2$ by a random tree (left side) projected on the subspace span by $\mathbf{X}^{(-2)} = X^{(1)}$ (right side), for $p = 2$ and $j = 2$.

$$\widehat{\text{S-MDA}}_{M,n}(X^{(j)}) = \frac{1}{\hat{\sigma}_Y^2}\frac{1}{n}\sum_{i=1}^{n}\left[Y_i - m_{M,n}^{(-j,OOB)}(\mathbf{X}_i^{(-j)}, \boldsymbol{\Theta}_M)\right]^2 - \left[Y_i - m_{M,n}^{(OOB)}(\mathbf{X}_i, \boldsymbol{\Theta}_M)\right]^2$$

Consistency of the Sobol-MDA

The Sobol-MDA recovers the appropriate theoretical counterpart for variable selection: the total Sobol index

Consistency of the Sobol-MDA

The Sobol-MDA recovers the appropriate theoretical counterpart for variable selection: the total Sobol index

Theorem (Bénard, Da Veiga, et al. (2021))

If Assumptions (A1), (A2'), and (A3') are satisfied, for all $M \in \mathbb{N}^\star$ *and* $j \in \{1, \ldots, p\}$

$$\widehat{S\text{-}MDA}_{M,n}(X^{(j)}) \xrightarrow{p} ST^{(j)}.$$

Consistency of the Sobol-MDA

The Sobol-MDA recovers the appropriate theoretical counterpart for variable selection: the total Sobol index

Theorem (Bénard, Da Veiga, et al. (2021))

If Assumptions (A1), (A2'), and (A3') are satisfied, for all $M \in \mathbb{N}^\star$ *and* $j \in \{1, \ldots, p\}$

$$\widehat{S\text{-}MDA}_{M,n}(X^{(j)}) \xrightarrow{p} ST^{(j)}.$$

(A2')

- *A node split is constrained to generate child nodes with at least a small fraction* $\gamma > 0$ *of the parent node observations.*
- *At each tree node, the number* `mtry` *of candidate variables drawn to optimize the split is set to* $mtry = 1$ *with a small probability* $\delta > 0$*. Otherwise, with probability* $1 - \delta$*, the default value of mtry is used.*

(A3')

The asymptotic regime of a_n*, the size of the subsampling without replacement, and the number of terminal leaves* t_n *is such that* $a_n \leq n - 2$*,* $a_n/n < 1 - \kappa$*, for a fixed* $\kappa > 0$*,* $\lim a_n, t_n = \infty$*,* $\lim 2^{t_n} \frac{(\log a_n)^9}{a_n} = 0$*.*

Sobol-MDA Experiments

Settings (Archer and Kimes, 2008; Gregorutti et al., 2017)

- $p = 200$ input variables
- 5 independent groups of 40 variables
- each group is a Gaussian vector, strongly correlated

Sobol-MDA Experiments

Settings (Archer and Kimes, 2008; Gregorutti et al., 2017)

- $p = 200$ input variables
- 5 independent groups of 40 variables
- each group is a Gaussian vector, strongly correlated
- 1 variable from each group involved in m

$$m(\mathbf{X}) = 2X^{(1)} + X^{(41)} + X^{(81)} + X^{(121)} + X^{(161)}.$$

- independent Gaussian noise with $\mathbb{V}[\varepsilon] = 10\%\mathbb{V}[Y]$

$$Y = m(\mathbf{X}) + \varepsilon$$

Sobol-MDA Experiments

Settings (Archer and Kimes, 2008; Gregorutti et al., 2017)

- $p = 200$ input variables
- 5 independent groups of 40 variables
- each group is a Gaussian vector, strongly correlated
- 1 variable from each group involved in m

$$m(\mathbf{X}) = 2X^{(1)} + X^{(41)} + X^{(81)} + X^{(121)} + X^{(161)}.$$

- independent Gaussian noise with $\mathbb{V}[\varepsilon] = 10\%\mathbb{V}[Y]$

$$Y = m(\mathbf{X}) + \varepsilon$$

- $n = 1000$ observations
- $M = 300$ trees

Sobol-MDA Experiments

$\widehat{\text{S-MDA}}$	
$\mathbf{X}^{(1)}$	0.035
$\mathbf{X}^{(161)}$	0.005
$\mathbf{X}^{(81)}$	0.004
$\mathbf{X}^{(121)}$	0.004
$\mathbf{X}^{(41)}$	0.002
$\mathbf{X}^{(179)}$	0.002
$\mathbf{X}^{(13)}$	0.001
$\mathbf{X}^{(25)}$	0.001
$\mathbf{X}^{(73)}$	0.001
$\mathbf{X}^{(155)}$	0.001

$\widehat{\text{BC-MDA}}/2\mathbb{V}[Y]$	
$\mathbf{X}^{(1)}$	0.048
$\mathbf{X}^{(25)}$	0.010
$\mathbf{X}^{(31)}$	0.008
$\mathbf{X}^{(14)}$	0.008
$\mathbf{X}^{(40)}$	0.007
$\mathbf{X}^{(3)}$	0.007
$\mathbf{X}^{(17)}$	0.006
$\mathbf{X}^{(26)}$	0.006
$\mathbf{X}^{(41)}$	0.006
$\mathbf{X}^{(121)}$	0.006

$\widehat{\text{IK-MDA}}/\mathbb{V}[Y]$	
$\mathbf{X}^{(1)}$	0.056
$\mathbf{X}^{(5)}$	0.009
$\mathbf{X}^{(81)}$	0.007
$\mathbf{X}^{(41)}$	0.005
$\mathbf{X}^{(161)}$	0.005
$\mathbf{X}^{(15)}$	0.005
$\mathbf{X}^{(121)}$	0.005
$\mathbf{X}^{(7)}$	0.005
$\mathbf{X}^{(4)}$	0.004
$\mathbf{X}^{(28)}$	0.004

Table: Sobol-MDA, normalized BC-MDA, and normalized IK-MDA estimates with influential variables in blue.

Conclusion

Additional experiments are available in Bénard, Da Veiga, et al. (2021) (non-linear data with interactions and dependence)

- analytical example
- backward variable selection with real data

Conclusion

Additional experiments are available in Bénard, Da Veiga, et al. (2021) (non-linear data with interactions and dependence)

- analytical example
- backward variable selection with real data

Extension. Sobol-MDA can be associated with any black-box algorithm

- fit a black box $\hat{f}$ on $\mathcal{D}_n$
- generate a large sample $\mathcal{D}'_N$ with $\hat{f}$
- run the Sobol-MDA with $\mathcal{D}'_N$

Conclusion

Additional experiments are available in Bénard, Da Veiga, et al. (2021) (non-linear data with interactions and dependence)

- analytical example
- backward variable selection with real data

Extension. Sobol-MDA can be associated with any black-box algorithm

- fit a black box $\hat{f}$ on $\mathcal{D}_n$
- generate a large sample $\mathcal{D}'_N$ with $\hat{f}$
- run the Sobol-MDA with $\mathcal{D}'_N$

Summary.

- Strong connections between the MDA and Sobol indices
- MDA does not target the appropriate quantity but Sobol-MDA does
- R/C++ package `SobolMDA`, available online on Gitlab (https://gitlab.com/drti/sobolmda), and based on the package `ranger`

SIRUS: **S**table and **I**nterpretable **RU**le **S**et

An example: SIRUS output on Titanic data set (Bénard, Biau, et al., 2021b)

Average survival rate $p_s = 39\%$.

if	`sex` is male	**then**	$p_s = 19\%$	**else**	$p_s = 74\%$
if	1^{st} or 2^{nd} `class`	**then**	$p_s = 56\%$	**else**	$p_s = 24\%$
if	1^{st} or 2^{nd} `class` **&** `sex` is female	**then**	$p_s = 95\%$	**else**	$p_s = 25\%$
if	`fare` < 10.5£	**then**	$p_s = 20\%$	**else**	$p_s = 50\%$
if	`no parents or children aboard`	**then**	$p_s = 35\%$	**else**	$p_s = 51\%$
if	2^{st} or 3^{nd} `class` **&** `sex` is male	**then**	$p_s = 14\%$	**else**	$p_s = 64\%$
if	`sex` is male **&** age ≥ 15	**then**	$p_s = 16\%$	**else**	$p_s = 72\%$

SIRUS

Principle

- Build a random forests and extract all decisions rules from all trees
- Select the rules that appear with a frequence larger than p_0
- Aggregate the rules to obtain the final estimator.

Principle

Frequent paths in random trees = strong and robust patterns in the data.

Technical detail

- Preprocessing: discretize features based on their quantiles
- Random forests: building trees of depth 2

Technical detail

- Preprocessing: discretize features based on their quantiles
- Random forests: building trees of depth 2

Probability that a Θ-random tree contains a given path $\mathscr{P} \in \Pi$

$$p_n(\mathscr{P}) = \mathbb{P}(\mathscr{P} \in T(\Theta, \mathcal{D}_n) | \mathcal{D}_n)$$

Selected paths

$$\boxed{\hat{\mathscr{P}}_{M,n,p_0} = \{\mathscr{P} \in \Pi : \hat{p}_{M,n}(\mathscr{P}) > p_0\}}$$

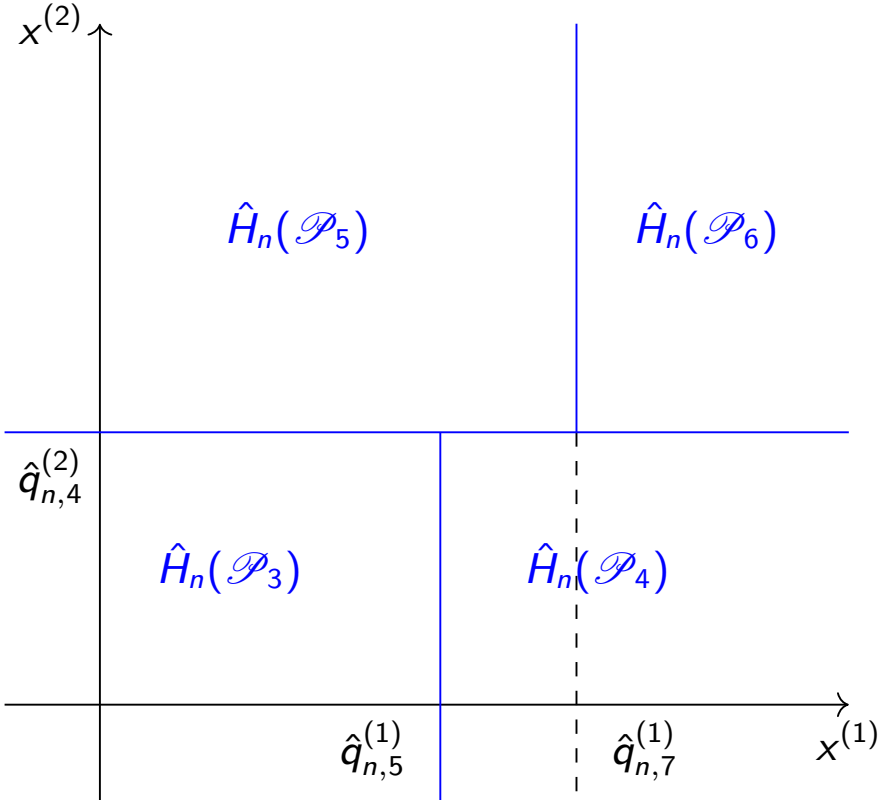
where

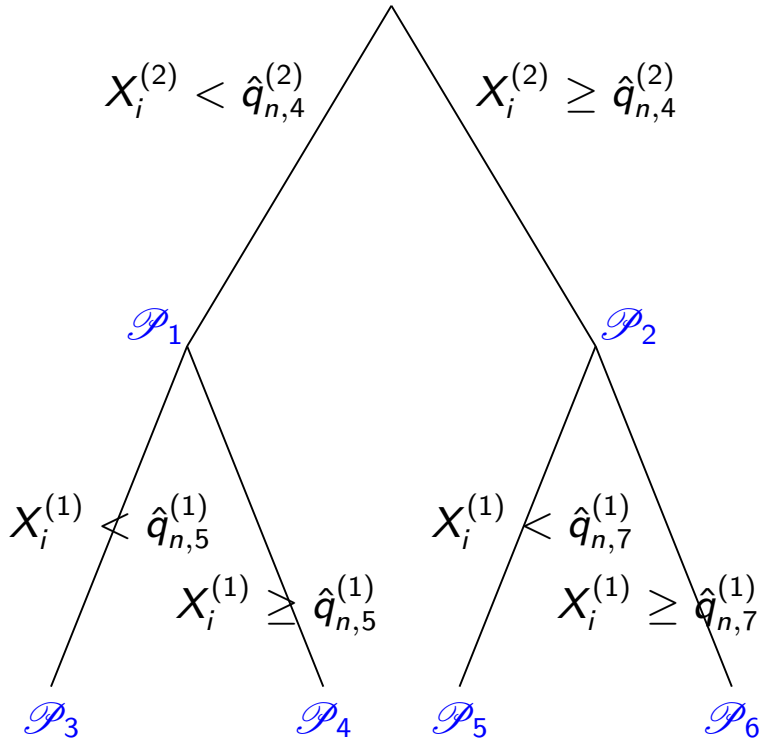
$$\hat{p}_{M,n}(\mathscr{P}) = \frac{1}{M} \sum_{\ell=1}^{M} \mathbb{1}_{\mathscr{P} \in T(\Theta_\ell, \mathcal{D}_n)}$$

is the Monte-Carlo estimate, directly computed using the random forest with M trees parametrized by $\Theta_1, ..., \Theta_M$.

SIRUS - Rule

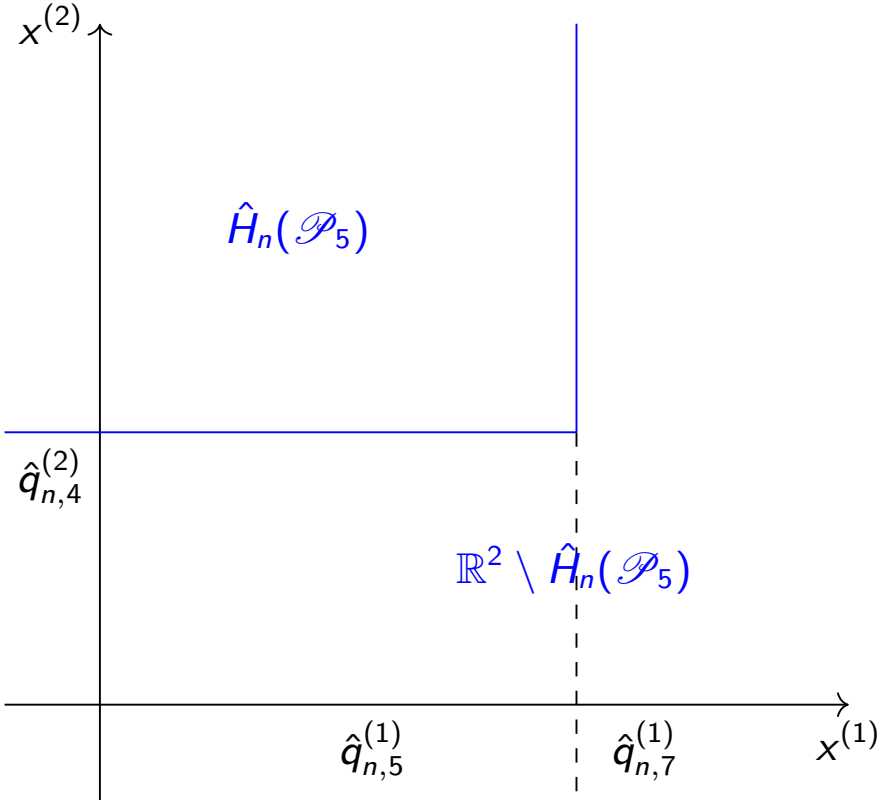
How to recover a rule from a path ?

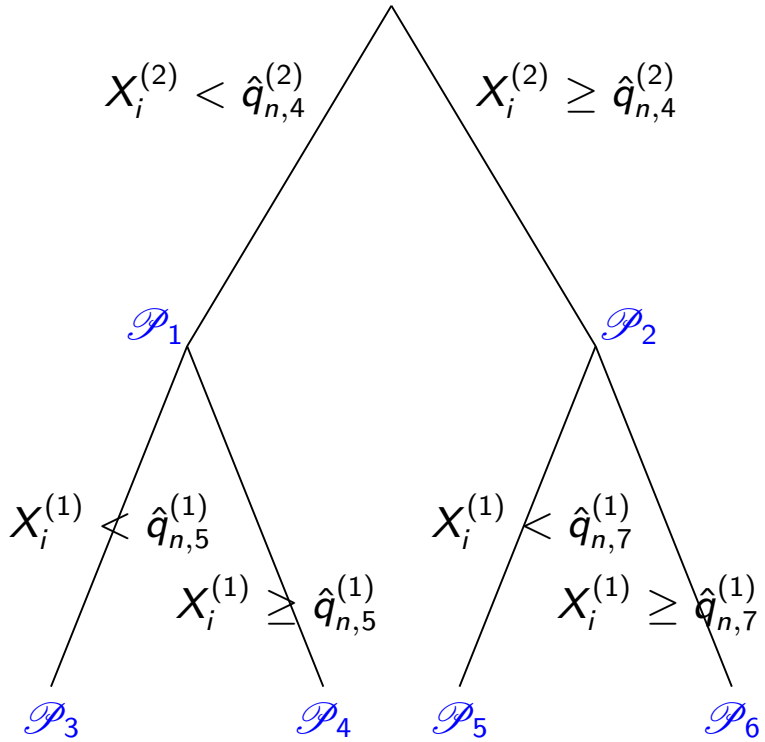




SIRUS - Rule

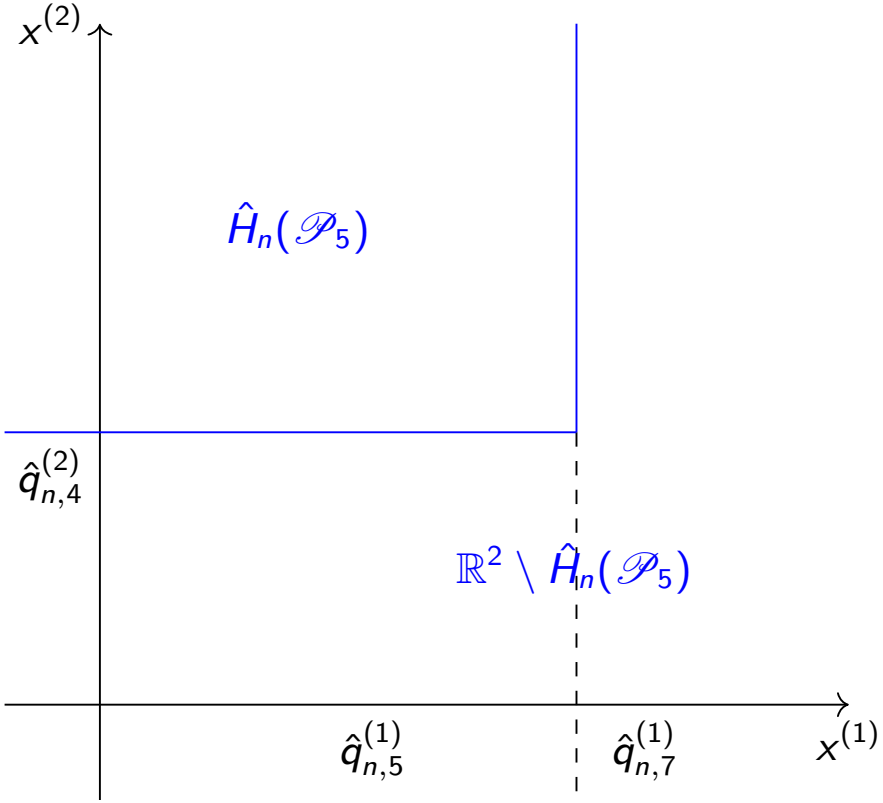
How to recover a rule from a path ?

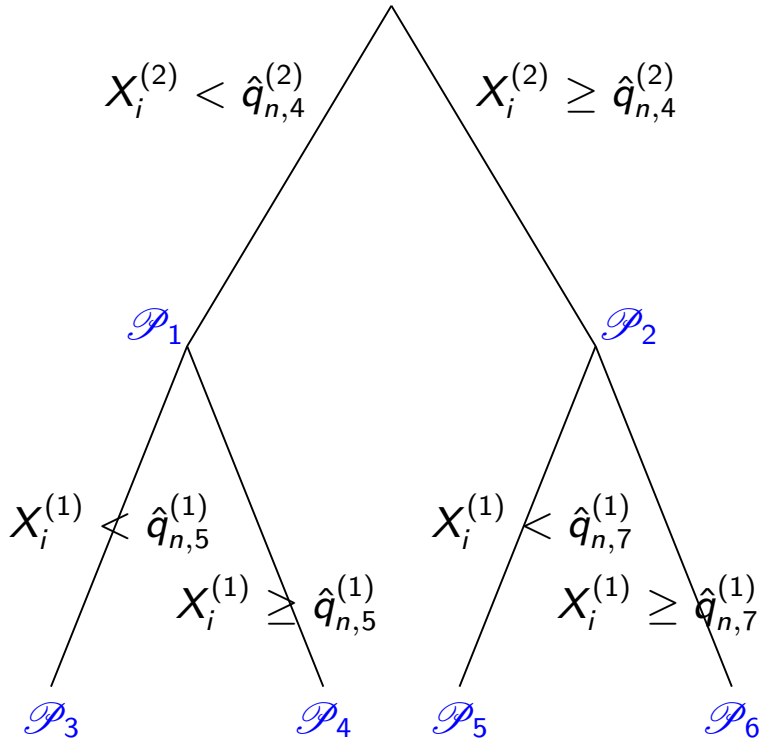




SIRUS - Rule

How to recover a rule from a path ?

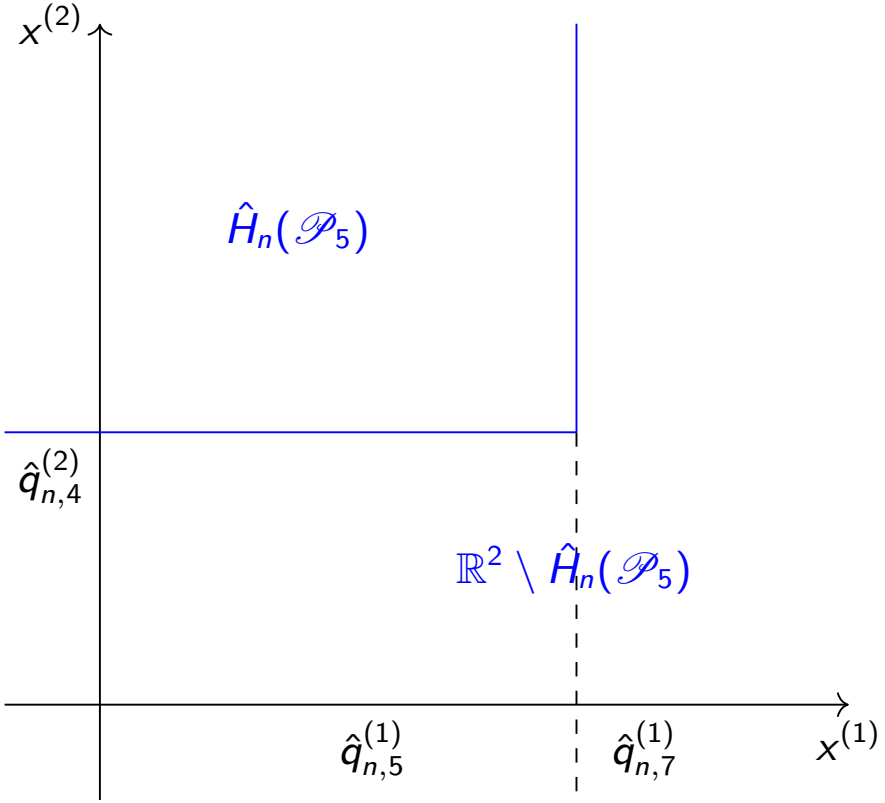


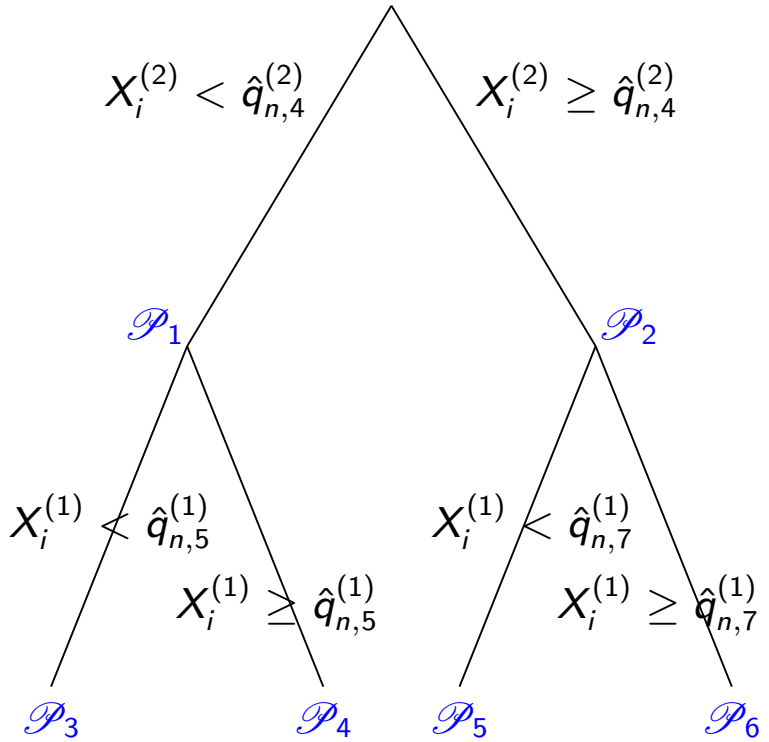


$$\forall \mathbf{x} \in \mathbb{R}^p, \quad \hat{g}_{n,\mathscr{P}}(\mathbf{x}) = \begin{cases} \frac{1}{N_n(\hat{H}_n(\mathscr{P}))} \sum_{i=1}^n Y_i \mathbb{1}_{\mathbf{x}_i \in \hat{H}_n(\mathscr{P})} & \text{if } \mathbf{x} \in \hat{H}_n(\mathscr{P}) \\ \frac{1}{n - N_n(\hat{H}_n(\mathscr{P}))} \sum_{i=1}^n Y_i \mathbb{1}_{\mathbf{x}_i \notin \hat{H}_n(\mathscr{P})} & \text{otherwise.} \end{cases}$$

SIRUS - Rule

How to recover a rule from a path ?





$$\forall \mathbf{x} \in \mathbb{R}^p, \quad \hat{g}_{n,\mathscr{P}}(\mathbf{x}) = \begin{cases} \frac{1}{N_n(\hat{H}_n(\mathscr{P}))} \sum_{i=1}^n Y_i \mathbb{1}_{\mathbf{x}_i \in \hat{H}_n(\mathscr{P})} & \text{if } \mathbf{x} \in \hat{H}_n(\mathscr{P}) \\ \frac{1}{n - N_n(\hat{H}_n(\mathscr{P}))} \sum_{i=1}^n Y_i \mathbb{1}_{\mathbf{x}_i \notin \hat{H}_n(\mathscr{P})} & \text{otherwise.} \end{cases}$$

The final classifier corresponds to the averaging of all selected rules.

Stability - definition

Define

- $\mathcal{D}'_n$, Θ' independent copies of $\mathcal{D}_n$ and Θ
- $\hat{p}'_{M,n}(\mathscr{P})$, $\hat{\mathscr{P}}'_{M,n,p_0}$ built with $\mathcal{D}'_n$, Θ'

Dice-Sorensen index

$$\hat{S}_{M,n,p_0} = \frac{2\left|\hat{\mathscr{P}}_{M,n,p_0} \cap \hat{\mathscr{P}}'_{M,n,p_0}\right|}{\left|\hat{\mathscr{P}}_{M,n,p_0}\right| + \left|\hat{\mathscr{P}}'_{M,n,p_0}\right|}.$$

Stability - a theoretical result

(A1) The subsampling rate a_n satisfies $\lim_{n\to\infty} a_n = \infty$ and $\lim_{n\to\infty} \frac{a_n}{n} = 0$.

(A2) The number of trees M_n satisfies $\lim_{n\to\infty} M_n = \infty$.

(A3) $\mathbf{X}$ has a density f with respect to the Lebesgue measure, continuous, bounded, and strictly positive.

Let $\mathcal{U}^\star = \{p^\star(\mathscr{P}), \mathscr{P} \in \Pi\}$ be the set of all theoretical probabilities of appearance of all paths.

Proposition Bénard, Biau, et al., 2021b

Assume that Assumptions (A1)-(A3) are satisfied. Then, provided $p_0 \in [0,1]\backslash\mathcal{U}^\star$, we have

$$\lim_{n\to\infty} \hat{S}_{M_n,n,p_0} = 1, \quad \text{in probability.}$$

Sketch of proof

The asymptotic stability of SIRUS comes from the two following points:

1. The bias of $\hat{p}_{M_n,n}(\mathscr{P})$ tends to zero.

2. The variance of $\hat{p}_{M_n,n}(\mathscr{P})$ tends to zero.

Sketch of proof

The asymptotic stability of SIRUS comes from the two following points:

1. The bias of $\hat{p}_{M_n,n}(\mathscr{P})$ tends to zero.
   - Prove that **CART-splitting criterion** is **consistent** and **asymptotically normal** when cuts are limited to empirical quantiles and the number of trees grows with n (A3).
2. The variance of $\hat{p}_{M_n,n}(\mathscr{P})$ tends to zero.

Sketch of proof

The asymptotic stability of SIRUS comes from the two following points:

1. The bias of $\hat{p}_{M_n,n}(\mathscr{P})$ tends to zero.
   - Prove that **CART-splitting criterion** is **consistent** and **asymptotically normal** when cuts are limited to empirical quantiles and the number of trees grows with n (A3).
2. The variance of $\hat{p}_{M_n,n}(\mathscr{P})$ tends to zero.

   The variance can be decomposed into two terms:
   - the variance generated by the Monte-Carlo randomization, which goes to 0 as the number of trees increases (A2).
   - the variance of $p_n(\mathscr{P})$, which is a bagged estimate and thus an infinite-order U-statistic. The result comes from Mentch and Hooker, 2016 since $\lim_{n\to\infty} a_n/n = 0$ (A1).

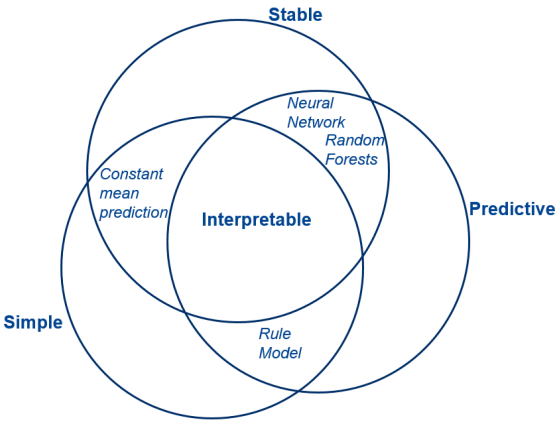
Numerical experiments

Competitors:

- CART (Breiman et al., 1984)
- Classical rule learning: RIPPER (Cohen, 1995)
- Frequent pattern mining: CBA (Classification Based on Association Rules, Liu et al., 1998), BRL (Bayesian Rule List, Letham et al., 2015)
- Tree ensemble: RuleFit (Friedman and Popescu, 2008), Node Harvest (Meinshausen, 2010).

Metrics:

- Accuracy/Error: 1-AUC
- Stability: Dice-Sorensen index
- Simplicity: Number of rules output by the procedure



Accuracy

	Black box	Decision tree	Classical rule learning	Frequent pattern mining		Tree ensemble		
Dataset	**Random Forest**	**CART**	**RIPPER**	**CBA**	**BRL**	**RuleFit**	**Node harvest**	**SIRUS**
Authentification	10^{-4}	0.02	0.02	0.14	0.009	$\mathbf{9.10^{-4}}$	0.02	0.03
Breast Wisconsin	0.009	0.06	0.07	0.05	0.02	**0.01**	**0.01**	**0.01**
Credit Approval	0.07	0.14	0.15	0.14	0.11	**0.08**	**0.07**	0.09
Credit German	0.20	0.29	0.38	0.40	0.33	**0.23**	**0.26**	**0.25**
Diabetes	0.17	0.25	0.29	0.30	0.25	**0.18**	**0.19**	**0.19**
Haberman	0.31	0.48	0.39	0.50	0.43	**0.37**	**0.34**	**0.35**
Heart C2	0.10	0.19	0.23	0.17	0.23	0.12	0.12	**0.10**
Heart H2	0.11	0.23	0.24	0.24	0.16	**0.11**	**0.11**	**0.12**
Heart Statlog	0.10	0.20	0.21	0.17	0.22	0.12	0.12	**0.10**
Hepatitis	0.12	0.48	0.39	0.36	0.33	0.20	0.23	**0.17**
Ionosphere	0.02	0.11	0.12	0.13	0.10	**0.04**	0.07	0.07
Kr vs Kp	9.10^{-4}	0.02	**0.009**	0.05	0.01	**0.009**	0.04	0.04
Liver Disorders	0.23	0.33	0.35	0.48	0.44	**0.27**	0.30	0.35
Mushrooms	0	0.007	3.10^{-5}	5.10^{-4}	$\mathbf{2.10^{-5}}$	5.10^{-4}	0.002	6.10^{-4}
Sonar	0.07	0.27	0.26	0.25	0.44	**0.12**	0.16	0.2
Spambase	0.01	0.11	0.08	0.12	0.05	**0.02**	0.04	0.07

Figure: Model error (1-AUC) over a 10-fold cross-validation for UCI datasets. Results are averaged over 10 repetitions of the cross-validation. Values within 10% of the minimum are displayed in bold, random forest is put aside.

Simplicity

	Decision tree	Classical rule learning	Frequent pattern mining		Tree ensemble		
Dataset	**CART**	**RIPPER**	**CBA**	**BRL**	**RuleFit**	**Node harvest**	**SIRUS**
Authentification	21	7	7	17	49	30	13
Breast Wisconsin	7	12	24	7	24	32	24
Credit Approval	5	4	55	4	15	27	16
Credit German	18	3	69	4	33	33	20
Diabetes	13	3	17	6	26	31	8
Haberman	2	1	2	2	3	17	5
Heart C2	10	3	34	4	23	36	20
Heart H2	5	2	29	3	12	24	12
Heart Statlog	10	3	27	4	22	35	16
Hepatitis	2	2	14	2	8	14	12
Ionosphere	4	4	38	4	20	35	15
Kr vs Kp	16	15	29	9	18	13	24
Liver Disorders	15	3	2	3	19	33	17
Mushrooms	4	8	25	11	10	22	23
Sonar	6	4	33	2	32	83	19
Spambase	14	16	126	16	68	60	21

Figure: Mean model size over a 10-fold cross-validation for UCI datasets. Results are averaged over 10 repetitions of the cross-validation.

Stability

	Decision tree	Classical rule learning	Frequent pattern mining		Tree ensemble		
Dataset	**CART**	**RIPPER**	**CBA**	**BRL**	**RuleFit**	**Node harvest**	**SIRUS**
Authentification	0.41	0.36	**0.87**	**0.86**	0.48	0.59	**0.81**
Breast Wisconsin	0.21	0.55	**0.80**	**0.78**	0.34	0.71	0.70
Credit Approval	0.52	0.32	0.43	0.52	0.25	0.23	**0.75**
Credit German	0.46	0.22	0.51	0.41	0.24	0.48	**0.66**
Diabetes	0.29	0.21	0.46	**0.73**	0.39	0.45	**0.81**
Haberman	**0.83**	0.09	**0.79**	0.50	0.46	0.52	0.65
Heart C2	0.25	0.35	0.38	0.60	0.39	0.49	**0.71**
Heart H2	0.46	0.27	0.52	**0.73**	0.29	0.29	**0.65**
Heart Statlog	0.30	0.41	0.41	**0.75**	0.35	0.48	**0.83**
Hepatitis	0.26	0.16	0.24	0.34	0.26	0.49	**0.68**
Ionosphere	**0.96**	0.39	0.13	0.70	0.17	0.33	0.69
Kr vs Kp	0.71	0.74	**0.84**	**0.80**	0.19	0.27	**0.87**
Liver Disorders	0.23	0.10	**0.91**	0.50	0.24	0.31	0.58
Mushrooms	**1**	0.84	**0.98**	0.80	0.69	0.48	0.86
Sonar	0.34	0.04	0.09	0.19	0.09	0.20	**0.55**
Spambase	0.49	0.10	0.46	**0.86**	0.28	0.66	**0.78**

Figure: Mean stability over a 10-fold cross-validation for UCI datasets. Results are averaged over 10 repetitions of the cross-validation. Values within 10% of the maximum are displayed in bold.

Conclusion on SIRUS

Method

- Build a random forest with a limited tree depth (typically two) and with quantile discretization.
- Extract the most frequent decision rules
- Aggregate these rules to obtain a predictor

Benefits.

- Output a small, stable, and predictive set of rules
- Predictive performances are on par with RF
- Stability and number of rules improved over state-of-the-art algorithms
- Works for classification and regression tasks
- `R` package `sirus` available on CRAN [2]
- Bonus: theoretical guarantees of stability

[2] https://cran.r-project.org/web/packages/sirus/index.html

Interpretability

- Often required for sensitive real-world application
- No precise meaning
- Understanding what is meant by *Interpretability* is a prerequisite to the subsequent analysis.

Interpretability needs to be interpreted!

Interpretability

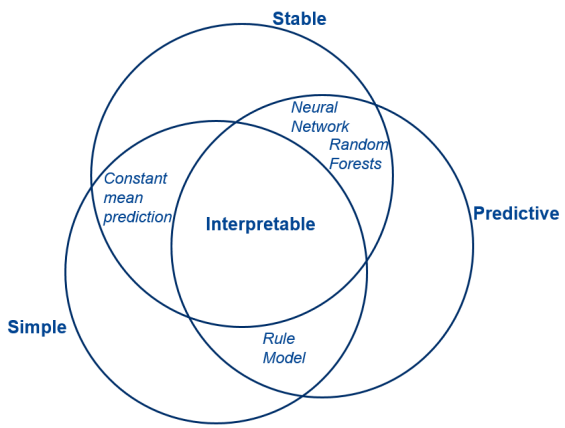
- Often required for sensitive real-world application
- No precise meaning
- Understanding what is meant by *Interpretability* is a prerequisite to the subsequent analysis.

Interpretability needs to be interpreted!

Two main approaches can be distinguished:

- Building a complex black-box model and try to understand it:
  - Variable importance
  - Local linearization

- Building a simple model, which is also stable and predictive
  - Decision rules (unstable)
  - Decision trees (unstable)
  - Linear/Logistic regression (limited predictivity)



Take-home messages on variable importance

- Different variable importances should be used depending on the task: (i) building a small predictive model or (ii) finding all variables related to the output.

Take-home messages on variable importance

- Different variable importances should be used depending on the task: (i) building a small predictive model or (ii) finding all variables related to the output.
- Variable importances are built based on heuristics, with no theoretical justification.

Take-home messages on variable importance

- Different variable importances should be used depending on the task: (i) building a small predictive model or (ii) finding all variables related to the output.
- Variable importances are built based on heuristics, with no theoretical justification.

Do not use MDI or MDA!

We do not know what quantity they are targeting

Take-home messages on variable importance

- Different variable importances should be used depending on the task: (*i*) building a small predictive model or (*ii*) finding all variables related to the output.
- Variable importances are built based on heuristics, with no theoretical justification.

Do not use MDI or MDA!

We do not know what quantity they are targeting

Alternatives that circumvent some of their flaws have been proposed:

- MDI
  - Out-of-sample estimation (Li et al., 2019; Zhou and Hooker, 2021; Loecher, 2022) with code in python:

    `https://github.com/ZhengzeZhou/unbiased-feature-importance`
- MDA
  - Rerun the model without a given covariate (expensive). Work for any predictive model (Williamson et al., 2021)
  - Use the tree structure to remove a variable from the model without needing to rerun it (Bénard, Da Veiga, et al., 2021)

Take-home messages on variable importance

- Different variable importances should be used depending on the task: (*i*) building a small predictive model or (*ii*) finding all variables related to the output.
- Variable importances are built based on heuristics, with no theoretical justification.

Do not use MDI or MDA!

We do not know what quantity they are targeting

Alternatives that circumvent some of their flaws have been proposed:

- MDI
  - Out-of-sample estimation (Li et al., 2019; Zhou and Hooker, 2021; Loecher, 2022) with code in python:
    `https://github.com/ZhengzeZhou/unbiased-feature-importance`
- MDA
  - Rerun the model without a given covariate (expensive). Work for any predictive model (Williamson et al., 2021)
  - Use the tree structure to remove a variable from the model without needing to rerun it (Bénard, Da Veiga, et al., 2021)

Anyway, remember to check the predictive performance of a model: it it is low, the model is useless and variable importances are misleading.

Take-home messages on simple models

Generalized linear models

- Linear/Logistic regression models are simple
- But easy to interpret only in the case of independent input variables!

Take-home messages on simple models

Generalized linear models

- Linear/Logistic regression models are simple
- But easy to interpret only in the case of independent input variables!

Decision trees/rules

- Easy to interpret when the number of rules/tree depth is small
- Often unstable: new runs of the algorithm on new data can change drastically the results

Take-home messages on simple models

Generalized linear models

- Linear/Logistic regression models are simple
- But easy to interpret only in the case of independent input variables!

Decision trees/rules

- Easy to interpret when the number of rules/tree depth is small
- Often unstable: new runs of the algorithm on new data can change drastically the results

SIRUS

- Stabilize decision rules by construction, based on random forests
- Good performances and stability results with a small number of rules
- Operates in regression and classification

Main references

Random forests

- Seminal papers on random forests (Breiman, 2000; Breiman, 2004)
- Overview on random forests Criminisi et al., 2011; Biau and Scornet, 2016

Variable importances

- MDI is ill-defined (Scornet, 2020)
- Use a different sample to estimate MDI (Li et al., 2019; Zhou and Hooker, 2021; Loecher, 2022)[3]
- Sobol-MDA, an alternative to MDA (Bénard, Da Veiga, et al., 2021) [4]

Decision rules

- Sirus in classification (Bénard, Biau, et al., 2021b) and regression (Bénard, Biau, et al., 2021a)
- `R` package `sirus` available on CRAN [5]

[3] https://github.com/ZhengzeZhou/unbiased-feature-importance
[4] `R` package `SobolMDA` https://gitlab.com/drti/sobolmda)
[5] https://cran.r-project.org/web/packages/sirus/index.html

Questions ?

References I

[AK08] K.J. Archer and R.V. Kimes. "Empirical characterization of random forest variable importance measures". In: *Computational Statistics & Data Analysis* 52 (2008), pp. 2249–2260.

[BDS21] C. Bénard, S. Da Veiga, and E. Scornet. "MDA for random forests: inconsistency, and a practical solution via the Sobol-MDA". In: *arXiv preprint arXiv:2102.13347* (2021).

[Bén+21a] C. Bénard, G. Biau, et al. "Interpretable random forests via rule extraction". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 937–945.

[Bén+21b] C. Bénard, G. Biau, et al. "Sirus: Stable and interpretable rule set for classification". In: (2021).

[Ben19] N. Benoumechiara. "Treatment of dependency in sensitivity analysis for industrial reliability". PhD thesis. Sorbonne Université ; EDF R&D, 2019.

[Bou+11] A.-L. Boulesteix et al. "Random forest Gini importance favours SNPs with large minor allele frequency: impact, sources and recommendations". In: *Briefings in Bioinformatics* 13 (2011), pp. 292–304.

[Bre+84] L. Breiman et al. *Classification and Regression Trees*. New York: Chapman & Hall, 1984.

References II

[Bre00] L. Breiman. *Some infinity theory for predictor ensembles*. Technical Report 577, UC Berkeley. 2000.

[Bre01a] L. Breiman. "Random forests". In: *Machine Learning* 45 (2001), pp. 5–32.

[Bre01b] L. Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical Science* 16 (2001), pp. 199–231.

[Bre02] L. Breiman. "Manual on setting up, using, and understanding random forests v3. 1". In: *Statistics Department University of California Berkeley, CA, USA* 1 (2002), p. 58.

[Bre04] L. Breiman. *Consistency for a simple model of random forests*. Technical Report 670, UC Berkeley. 2004.

[BS16] G. Biau and E. Scornet. "A random forest guided tour". In: *Test* 25 (2016), pp. 197–227.

[Coh95] William W Cohen. "Fast effective rule induction". In: *Machine learning proceedings 1995*. Elsevier, 1995, pp. 115–123.

[CSK11] A. Criminisi, J. Shotton, and E. Konukoglu. "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning". In: *Foundations and Trends in Computer Graphics and Vision* 7 (2011), pp. 81–227.

References III

[DK17] F. Doshi-Velez and B. Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv:1702.08608* (2017).

[Fis58] Walter D Fisher. "On grouping for maximum homogeneity". In: *Journal of the American statistical Association* 53.284 (1958), pp. 789–798.

[FP08] Jerome H Friedman and Bogdan E Popescu. "Predictive learning via rule ensembles". In: *The Annals of Applied Statistics* 2.3 (2008), pp. 916–954.

[GMS17] B. Gregorutti, B. Michel, and P. Saint-Pierre. "Correlation and variable importance in random forests". In: *Statistics and Computing* 27 (2017), pp. 659–678.

[Gre15] B. Gregorutti. "Random forests and variable selection : analysis of the flight data recorders for aviation safety". PhD thesis. Université Pierre et Marie Curie - Paris VI, 2015.

[Let+15] Benjamin Letham et al. "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9.3 (2015), pp. 1350–1371.

[LHM+98] Bing Liu, Wynne Hsu, Yiming Ma, et al. "Integrating classification and association rule mining.". In: *Kdd*. Vol. 98. 1998, pp. 80–86.

References IV

[Li+19] X. Li et al. "A debiased mdi feature importance measure for random forests". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8049–8059.

[Lip16] Z.C. Lipton. "The mythos of model interpretability". In: *arXiv:1606.03490* (2016).

[Loe22] Markus Loecher. "Unbiased variable importance for random forests". In: *Communications in Statistics-Theory and Methods* 51.5 (2022), pp. 1413–1425.

[Mei10] Nicolai Meinshausen. "Node harvest". In: *The Annals of Applied Statistics* (2010), pp. 2049–2072.

[MH16] L. Mentch and G. Hooker. "Quantifying uncertainty in random forests via confidence intervals and hypothesis tests". In: *Journal of Machine Learning Research* 17 (2016), pp. 841–881.

[MTA15] T. A Mara, S. Tarantola, and P. Annoni. "Non-parametric methods for global sensitivity analysis of model output with dependent inputs". In: *Environmental Modelling & Software* 72 (2015), pp. 173–183.

[Mur+19] W.J. Murdoch et al. "Interpretable machine learning: Definitions, methods, and applications". In: *arXiv:1901.04592* (2019).

References V

[Nic11] K. K. Nicodemus. "Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures". In: *Briefings in bioinformatics* 12.4 (2011), pp. 369–373.

[NM09] K. K. Nicodemus and J. D. Malley. "Predictor correlation impacts machine learning algorithms: implications for genomic studies". In: *Bioinformatics* 25.15 (2009), pp. 1884–1890.

[Sco20] E. Scornet. "Trees, forests, and impurity-based variable importance". In: *accepted for publication in Annales de l'IHP* (2020).

[Sob93] I.M. Sobol. "Sensitivity estimates for nonlinear mathematical models". In: *Mathematical Modelling and Computational Experiments* 1 (1993), pp. 407–414.

[Str+07] C. Strobl et al. "Bias in random forest variable importance measures: Illustrations, sources and a solution". In: *BMC bioinformatics* 8.1 (2007), p. 25.

[Wil+21] Brian D Williamson et al. "A general framework for inference on algorithm-agnostic variable importance". In: *Journal of the American Statistical Association* (2021), pp. 1–14.

[Yu13] B. Yu. "Stability". In: *Bernoulli* 19 (2013), pp. 1484–1500.

References VI

[ZH19] Z. Zhou and G. Hooker. "Unbiased measurement of feature importance in tree-based methods". In: *arXiv preprint arXiv:1903.05179* (2019).

[ZH21] Z. Zhou and G. Hooker. "Unbiased measurement of feature importance in tree-based methods". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15.2 (2021), pp. 1–21.