DDA3020 Machine Learning
Lecture 08 Tree-based Methods

Jicong Fan
School of Data Science, CUHK-SZ

10/14/2025

Outlines

Motivation

Parametric models:

- Until now, we have learned 3 supervised learning models, including linear regression (for regression and classification), logistic regression, and SVM.
- The commonality is that they are all parametric models:
  - In training, we define a **model** (*i.e.*, hypothesis function) over **the whole input space**, and learn its **parameters with fixed numbers** from all of the training data.
  - In testing, we use the same model and the same parameter set for any test input.
- The limitations of parametric models include:
  - The adopted model should be determined by the trainer, and it may be far from the ground-truth relationship between the input and the output.
  - The decision/prediction is difficult to explain/understand, there is no decision procedure

Motivation

Thus, we introduce **nonparametric models**

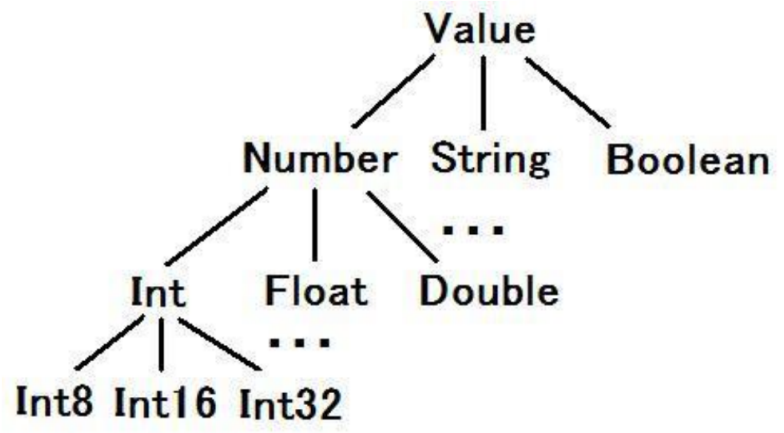
- They do not rely on strong assumptions regarding the shape of the relationship between the variables. Instead, the data are allowed to speak for themselves in determining the form of the fitted functions.
- Typical nonparametric models include k-nearest neighbors (KNN) and decision tree (DT).
- DT is a hierarchical nonparametric model. In addition to the above advantage, a special advantage of DT is the interpretability of its decision, which is a hierarchical decision process.

What Is Decision Tree?

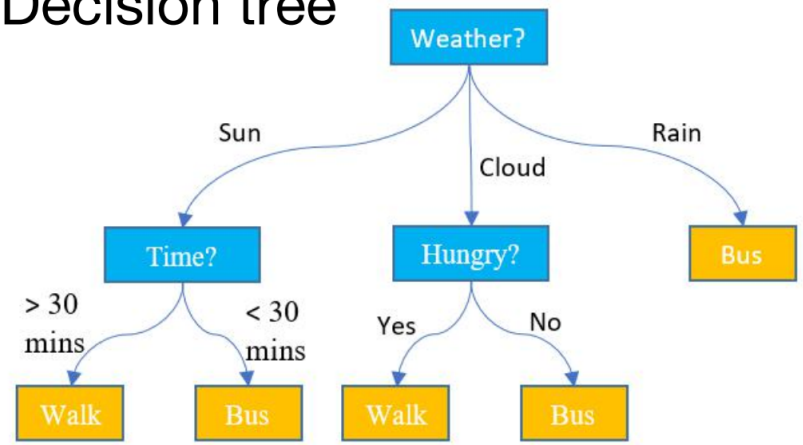
Trees

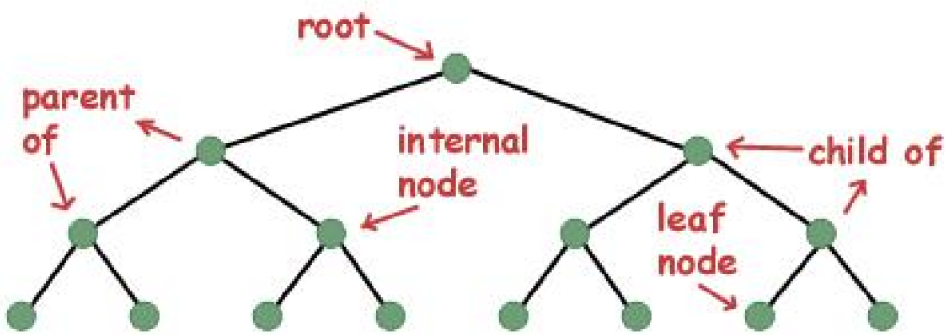
Components: root, branch, and leaf

Data structure



Decision tree



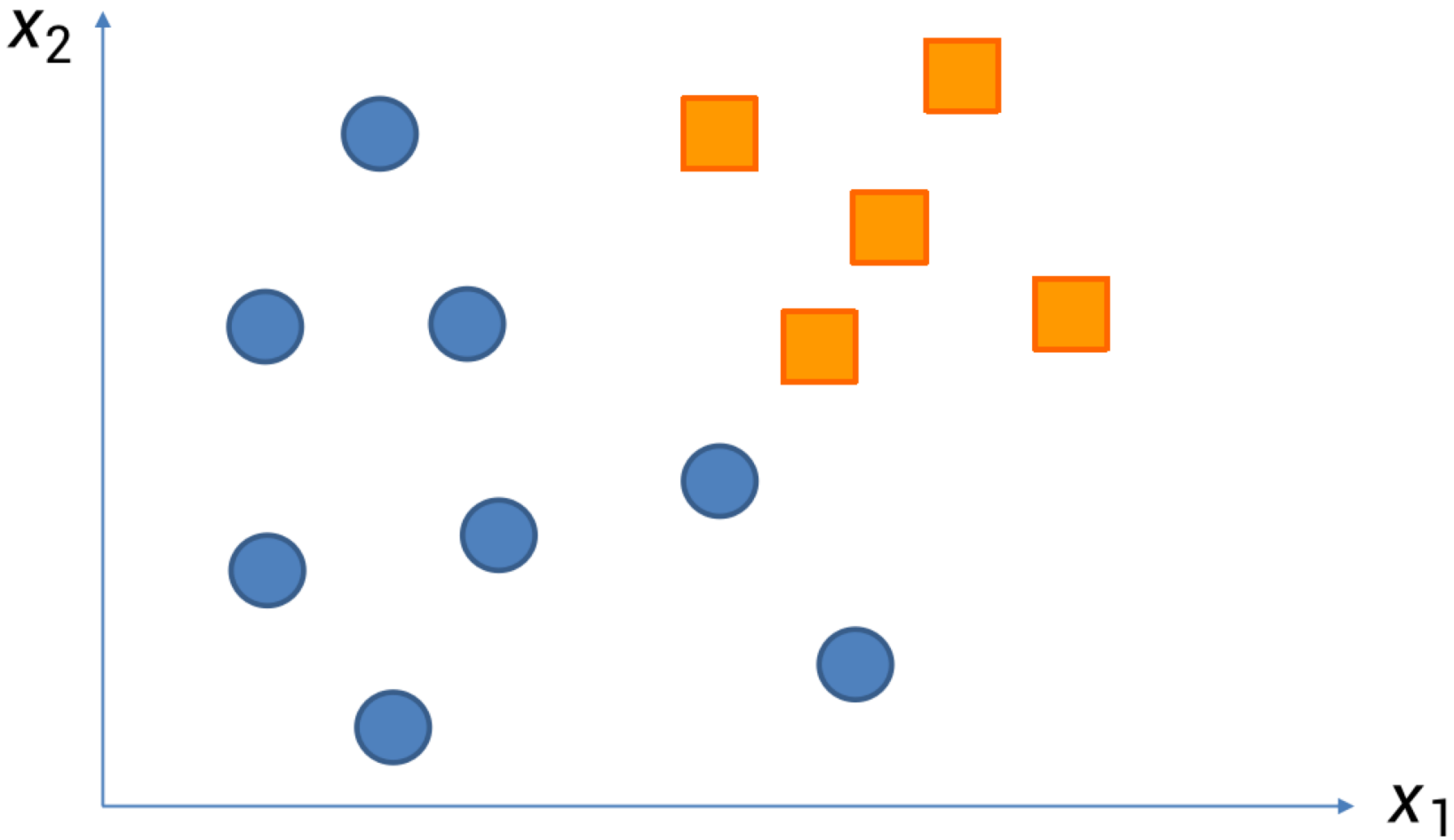


Reversed trees actually

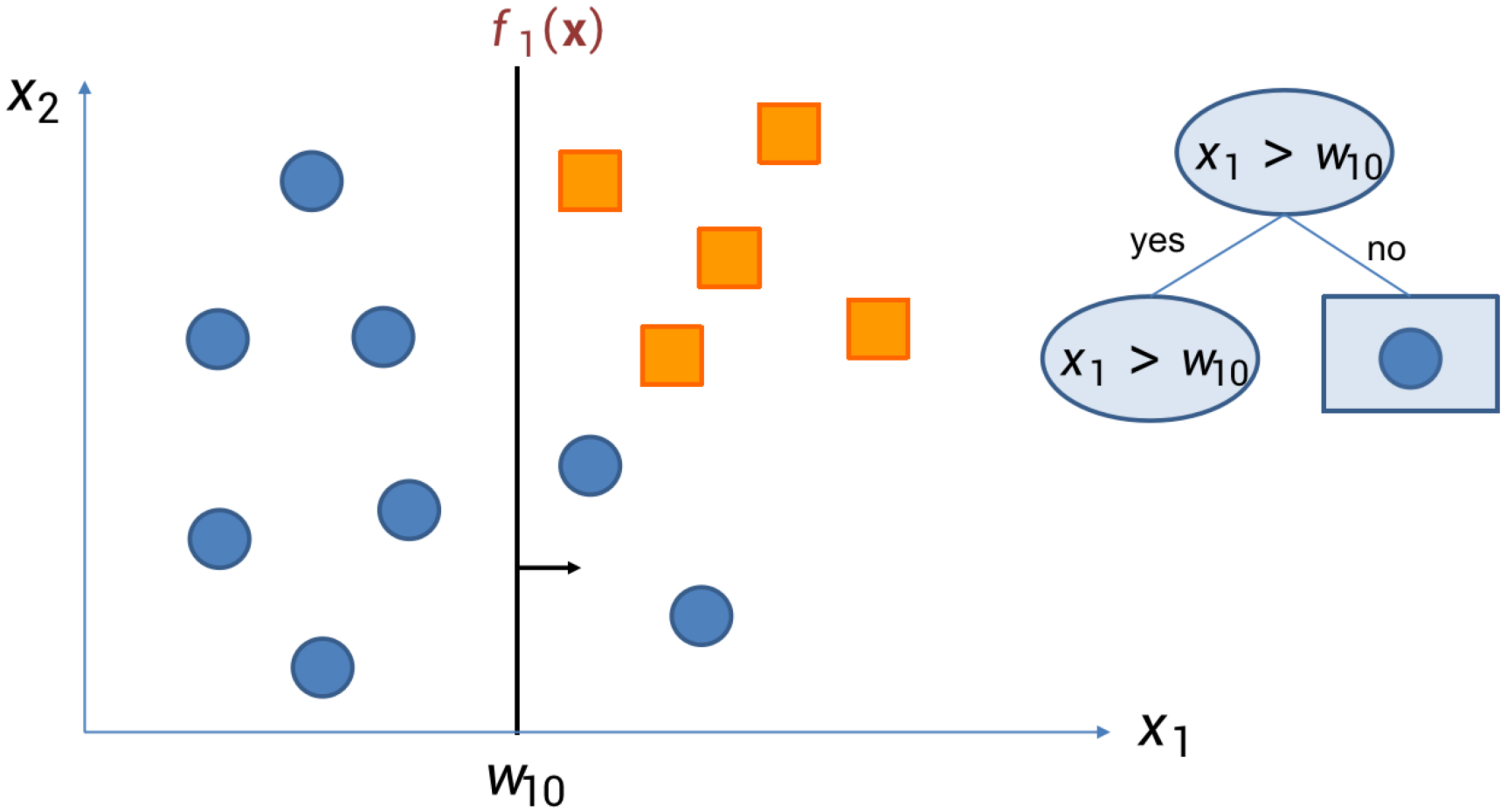
Definition

- A **decision tree** is a **hierarchical model** for supervised learning whereby the local region is identified in a sequence of **recursive splits**.
- In a **univariate tree**, in each internal node, the test uses only one of the input dimensions.
- Binary Tree and Multi-way Tree
  - Each node in a binary tree has at most two children, typically referred to as the left child and the right child.
  - Each node in a multi-way tree can have more than two children.
- Various decision tree algorithms
  - CART (classification and regression trees) [Breiman et al. 1984] — (binary tree)
  - ID3 (Iterative Dichotomiser 3) [Ross Quinlan, 1986] — (multi-way tree)

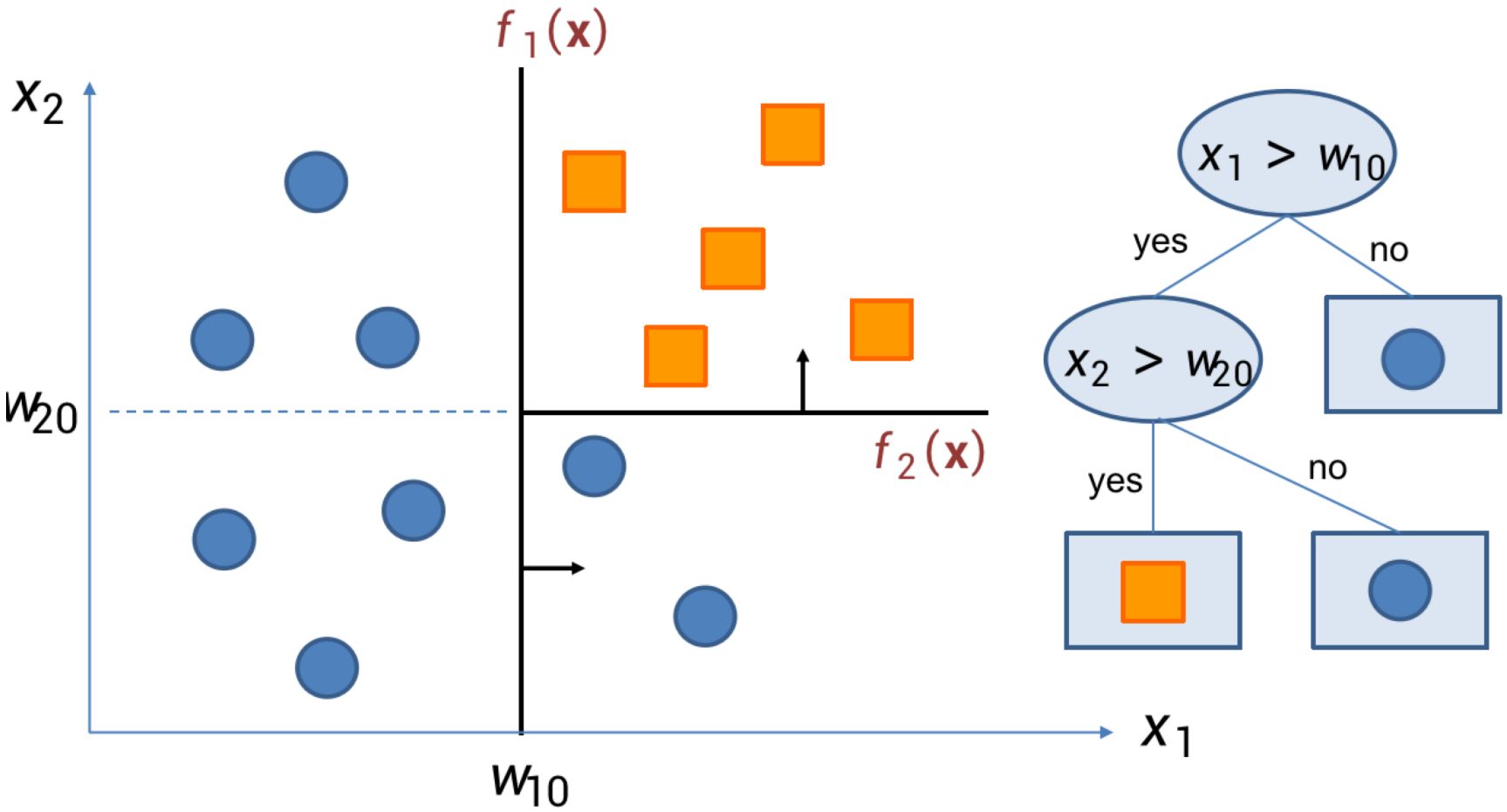
Example: A Classification Problem



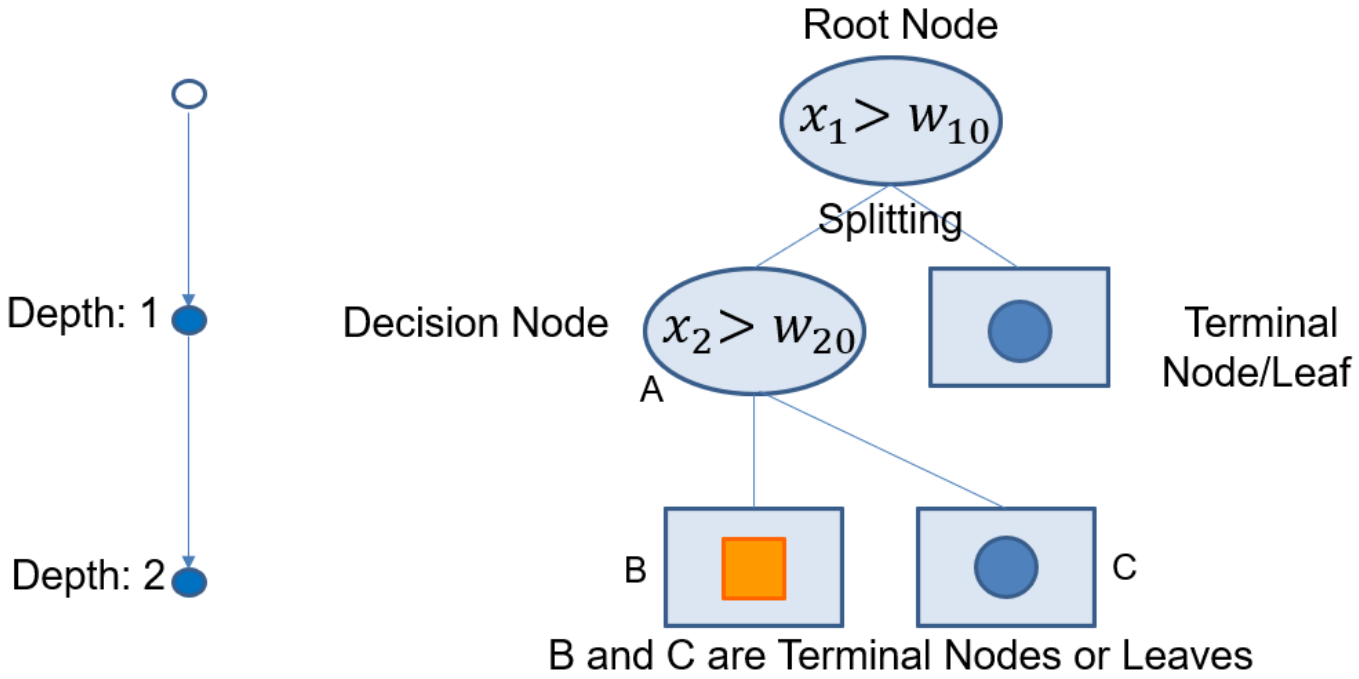
Example: A Classification Problem



Example: A Classification Problem



Basic Terminologies



- In a decision tree, each node represents a **decision point** and the branches represent the possible **outcomes**
- A-B-C forms a **sub-tree** or **branch**.
- A is **parent node** of B and C; B and C are the **child nodes** of A.
- The **depth** of a decision tree is the length of the longest path from a root to a leaf.
- The **size** of a decision tree is the number of nodes in the tree.

Tree Learning

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All leaf nodes are pure (only one class remains)
  - Or a maximum depth is reached
  - Or a performance metric is achieved
- Note: For a given training set, there exist many trees that code it with no error, and, for simplicity, we are interested in finding the smallest among them. However, finding the smallest tree is NP-complete (Quinlan 1986). Thus, we are forced to use local search procedures based on heuristics that give reasonable trees in a reasonable time.

How to choose the best attribute for a classification tree

As only one input attribute (variable) is used at each step, which attribute and what split are the best for each step? There are some rules:

- Random: an attribute chosen at random
- **Least-Values**: the attribute with the smallest number of possible values
- **Most-Values**: the attribute with the largest number of possible values
- **Impurity Measure**: the attribute that has the largest reduction of impurity

Impurity

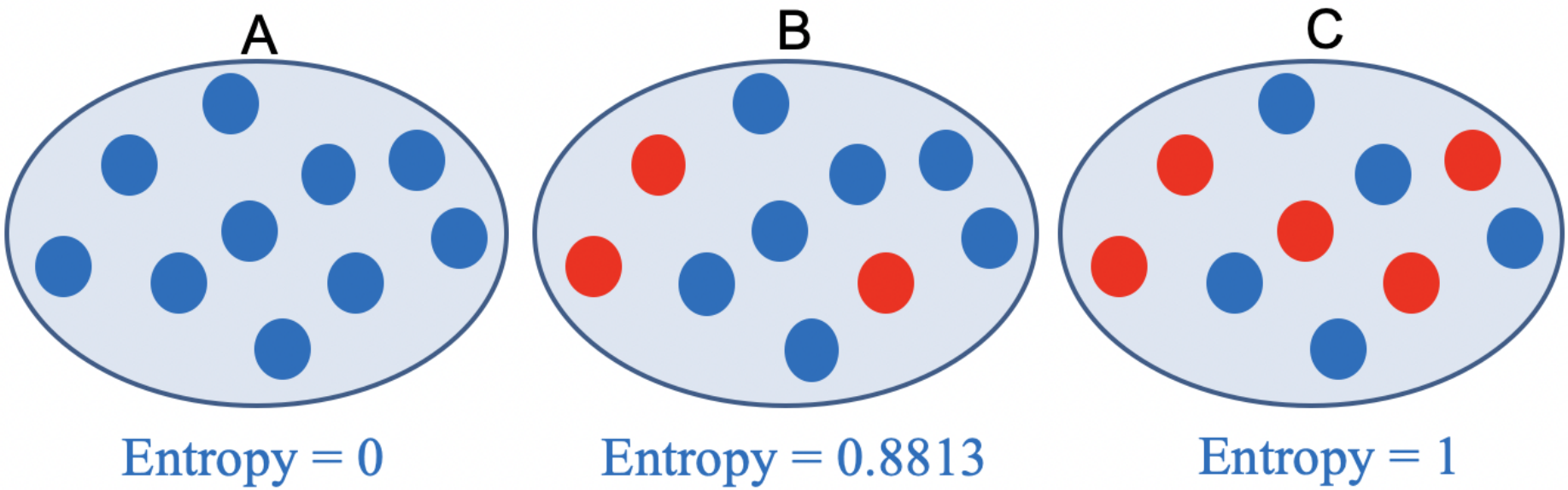
- Let $|S|$ be the number of training instances reaching node S (a set). If S is the root node, then $|S| = N$.
- Let S_i be the set of instances of S belonging to class C_i, $i = 1, \ldots, K$.
- Given that an instance reaches node S, the estimate for the probability of class C_i is

$$\widehat{P}\left(C_i \mid x, S\right) \equiv p_i = \frac{|S_i|}{|S|}$$

- Node S is pure if there exist an i such that $p_i = 1$.
- If the node is pure, we do not need to split any further and can add a leaf node labeled with the class for which p_i is 1 .

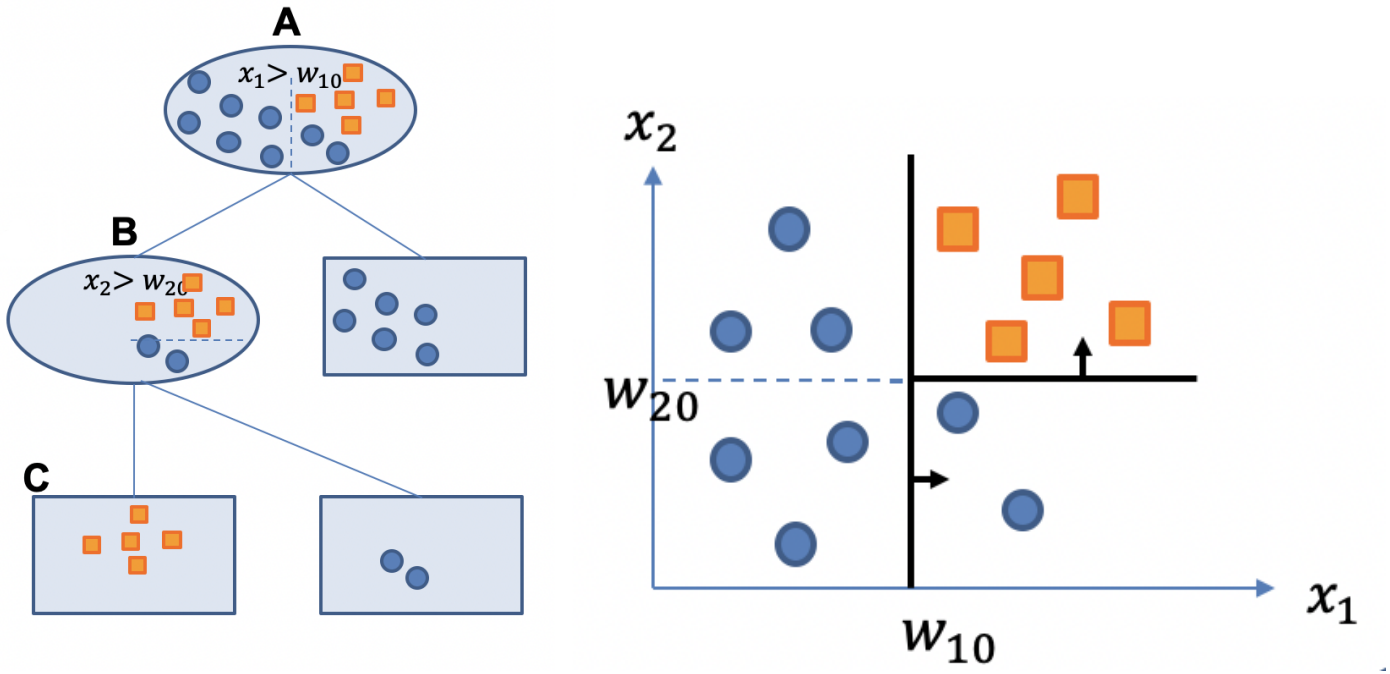
Impurity

- How to measure the impurity of one node? Entropy (or information)
- Entropy in information theory specifies the minimum number of bits needed to encode the class code of an instance.
- Entropy for multi-class node S: $\text{Info}(S) = -\sum_{i=1}^{K} p_i \log_2 p_i$.
- Entropy for binary (two-class) node : $-p \log_2 p - (1-p) \log_2 (1-p)$.



A is a pure node, B is more impure than A, and C is the most impure here.

Impurity: Entropy



Entropy (information) for parent node A : $-\left(\frac{5}{13}\right)\log_2\left(\frac{5}{13}\right)-\left(\frac{8}{13}\right)\log_2\left(\frac{8}{13}\right)=0.96$

Entropy (information) for node B : $-\left(\frac{5}{7}\right)\log_2\left(\frac{5}{7}\right)-\left(\frac{2}{7}\right)\log_2\left(\frac{2}{7}\right)=0.86$

Entropy (information) for node C : $-\left(\frac{5}{5}\right)\log_2\left(\frac{5}{5}\right)-\left(\frac{0}{5}\right)\log_2\left(\frac{0}{5}\right)\triangleq 0$

Information Gain

- Suppose we use an attribute $A \in \{a_1, a_2, \ldots, a_V\}$ to split (divide) S into V subsets: $D_1, D_2, \ldots, D_V$.
- The information of the partition by A is the conditional entropy* (the entropy of S given the condition A)

$$\text{Info}(S|A) = \sum_{v=1}^{V} \frac{|D_v|}{|S|} \times \text{Info}\,(D_v)$$

- Information gained by branching on attribute A

$$\text{Gain}(A) = \text{Info}(S) - \text{Info}(S|A)$$

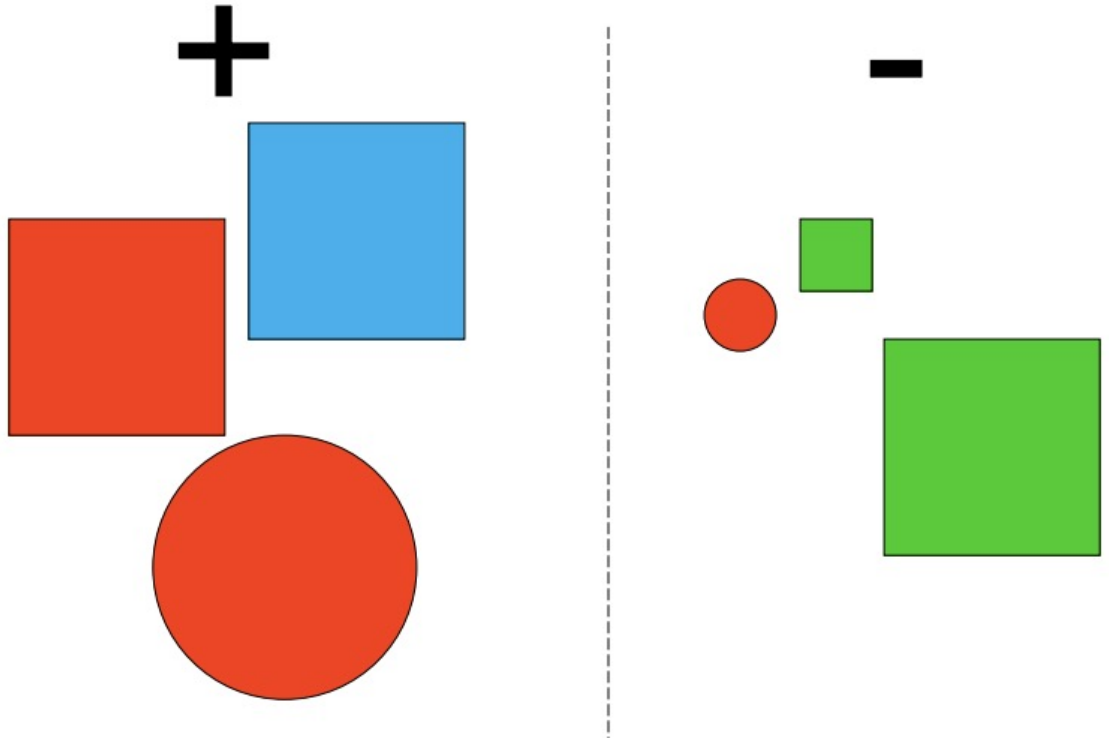
- Select the attribute with the highest information gain

* $$\mathrm{H}(Y|X) \equiv \sum_{x \in \mathcal{X}} p(x)\mathrm{H}(Y|X=x) = -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log_2 p(y|x)$$

$$= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x)p(y|x) \log_2 p(y|x) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log_2 \frac{p(x,y)}{p(x)}$$

Example: Choosing attribute via information gain

Consider a binary classification with the following training data, left is positive and right is negative. Each data is described by 3 attributes: Color, Size, Shape
What's the best attribute for the root node?

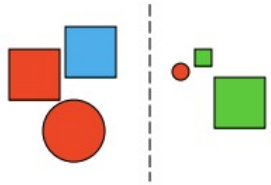


Reference: `https://pages.cs.wisc.edu/~dyer/cs540/notes/11_learning-decision-pdf`

Example: Choosing attribute via information gain

The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



- Compute the information of the root node (entire training set)

$$\text{Info}(S) = H(3/6, 3/6) = 1$$

- Compute the information of the partition by attribute 'Color'

$$\text{Info}(S|\text{Color}) = 3/6H(2/3, 1/3) + 1/6H(1/1, 0/1) + 2/6H(0/2, 2/2)$$

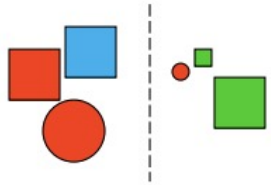
- Compute the information gain given by 'Color'

$$\text{Gain}(\text{Color}) = \text{Info}(S) - \text{Info}(S|\text{Color}) = 0.54 \text{ bits}$$

Example: Choosing attribute via information gain

The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



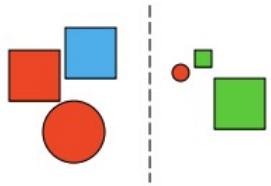
Similarly, we can compute the information gains given by 'Shape' and 'Size'. Then:

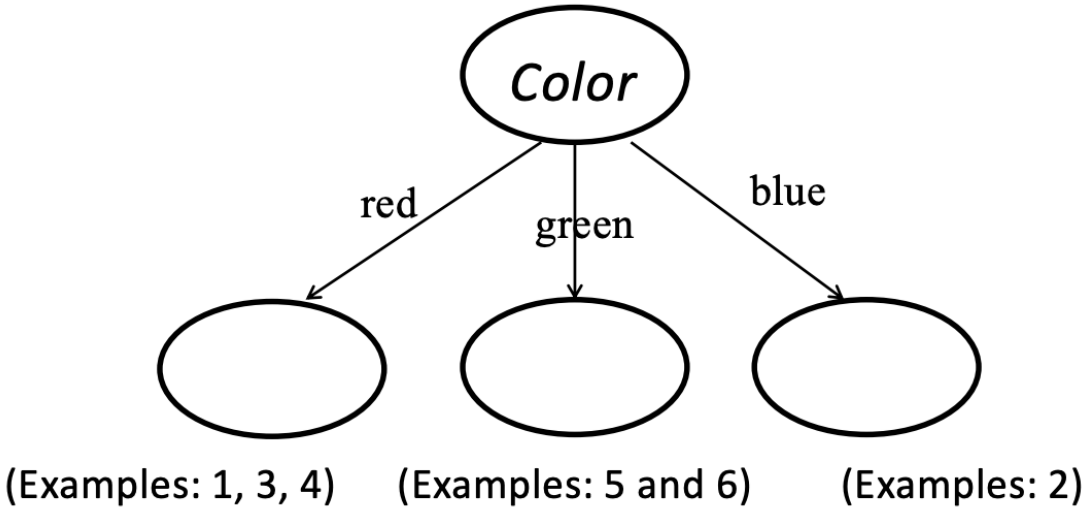
- $\text{Gain}(\text{Color}) = \text{Info}(S) - \text{Info}(S|\text{Color}) = 0.54$ bits
- $\text{Gain}(\text{Shape}) = \text{Info}(S) - \text{Info}(S|\text{Shape}) = 0$ bits
- $\text{Gain}(\text{Size}) = \text{Info}(S) - \text{Info}(S|\text{Size}) = 0.46$ bits

Finally, we select 'Color' as the best attribute at the root and split the data into three subsets.

Example: Choosing attribute via information gain

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



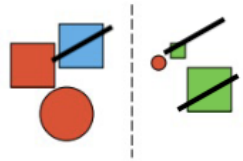


Now we only need to split the first child node (examples 1, 3, 4)

Example: Choosing attribute via information gain

The data at the first child node is shown in the following table:

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−



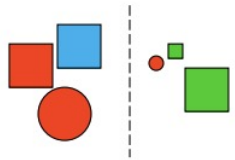
Which of 'Shape' and 'Size' should we use to split the data (denoted by S)?

- $\text{Info}(S) = H(2/3, 1/3) = (-2/3)\log 2/3 + (-1/3)\log 1/3 = 0.92$
- $\text{Info}(S|\text{Size}) = 2/3H(2/2, 0/2) + 1/3H(0/1, 1/1) = (2/3)(0) + (1/3)(0) = 0.$ So $\text{Gain}(\text{Size}) = 0.92 - 0 = 0.92.$
- $\text{Info}(S|\text{Shape}) = 1/3H(1/1, 0/1) + 2/3H(1/2, 1/2) = (1/3)(0) + (2/3)(1) = 0.67.$ So $\text{Gain}(\text{Shape}) = 0.92 - 0.67 = 0.25.$
- $\text{Info}(S|\text{Color}) = 3/3\text{H}(2/3, 1/3) + 0/3\text{H}(0/0, 0/0) = 0.92.$ So $\text{Gain}(\text{Color}) = 0.92 - 0.92 = 0.$
- Therefore, 'Size' is the best attribute.

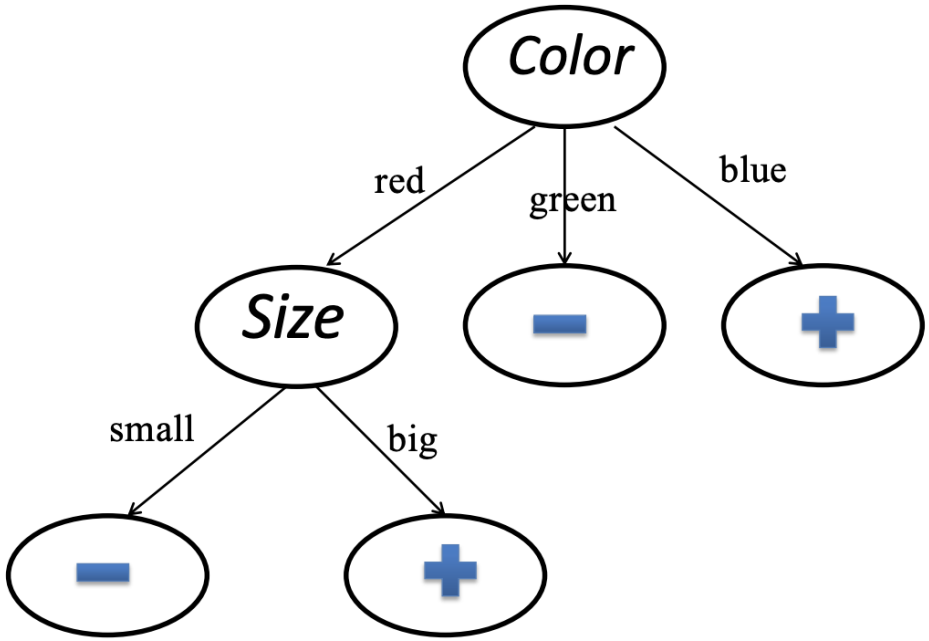
Example: Choosing attribute via information gain

Training data

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



Final decision tree



More Attribute Selection Measure

- Gini index: $\text{gini}(S) = 1 - \sum_{i=1}^{K} p_i^2$
- Suppose we use an attribute A to split S into V subsets $D_1, D_2, \ldots, D_V$, then the expected Gini index given A is defined as

$$\text{gini}(S|A) = \sum_{v=1}^{V} \frac{|D_v|}{|S|} \text{gini}\,(D_v)$$

- Reduction in impurity (Gini index):

$$\Delta\,\text{gini}(A) = \text{gini}(S) - \text{gini}(S|A)$$

- Info (entropy) v.s. Gini index
  - Gini index has a simpler computation.
  - Gini index is more interpretable.
  - Info is more effective when the classes are imbalanced.
  - Info is less sensitive to noise.

Decision Tree: An Example of Classification

Iris dataset (UCI machine learning database): 150 instances (plants), 4 features, and 3 classes.

Variables Table

Variable Name	Role	Type	Description	Units	Missing Values
sepal length	Feature	Continuous		cm	no
sepal width	Feature	Continuous		cm	no
petal length	Feature	Continuous		cm	no
petal width	Feature	Continuous		cm	no
class	Target	Categorical	class of iris plant: Iris Setosa, Iris Versicolour, or Iris Virginica		no

Decision Tree: An Example of Classification

Decision tree on Iris dataset

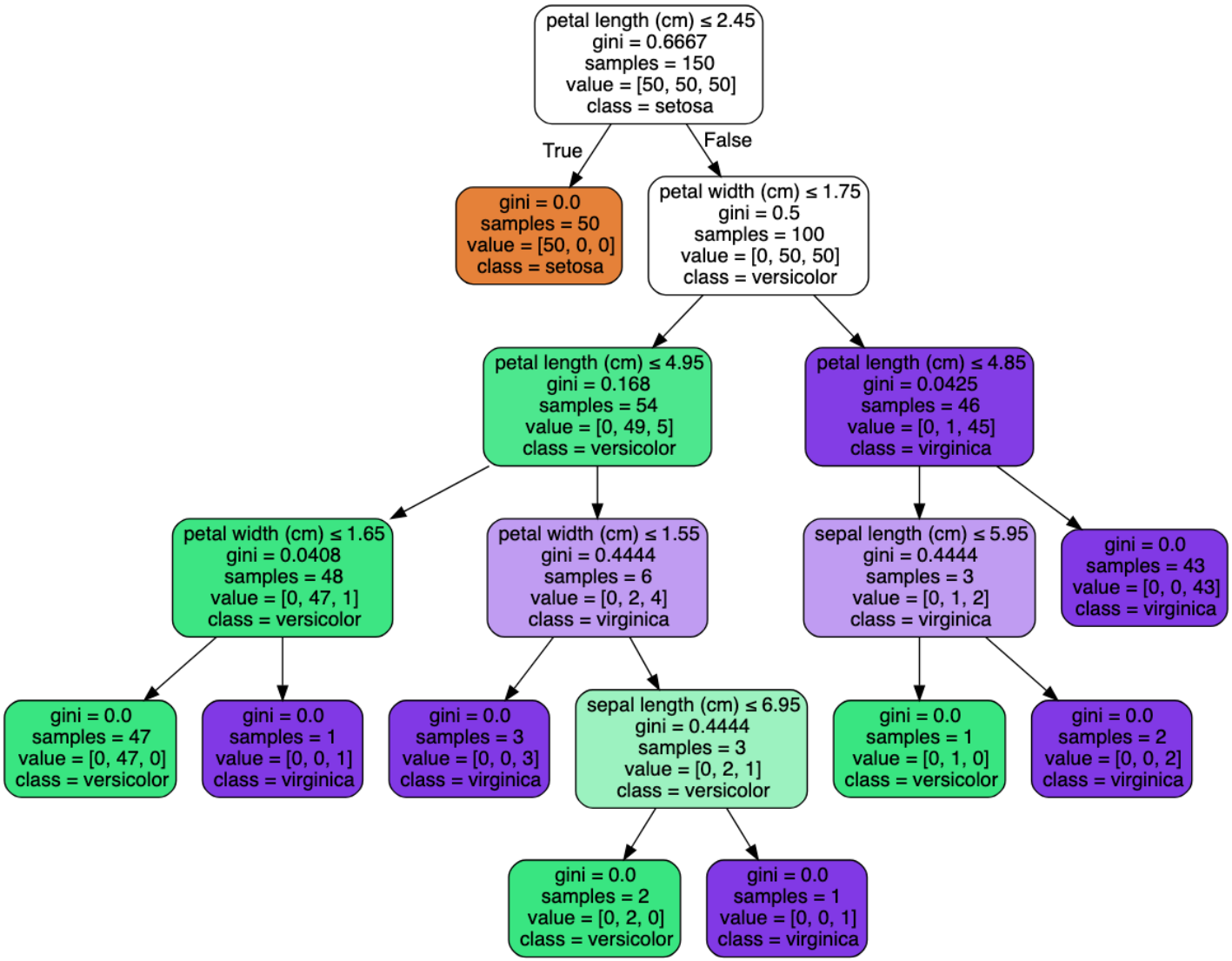


Image source: `https://scikit-learn.org/stable/modules/tree.html`

Decision Tree: An Example of Classification

Decision tree on Iris dataset

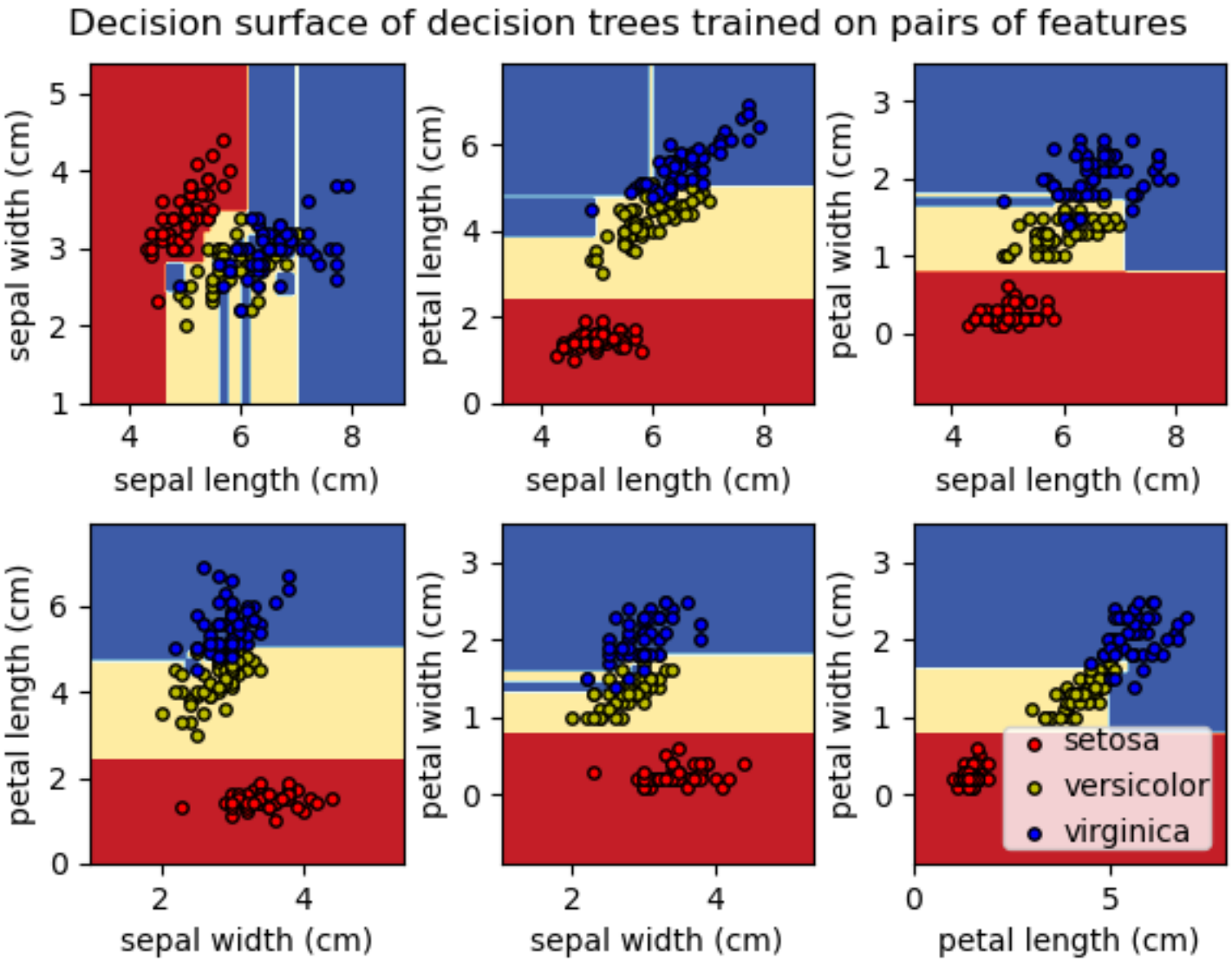


Image source: `https://scikit-learn.org/stable/modules/tree.html`

Regression Trees

- A *regression tree* is constructed in almost the same manner as a classification tree, except that the impurity measure that is appropriate for classification is replaced by a measure appropriate for regression.
- In regression, the goodness of a split is measured by the mean square error (MSE) or the sum of squared errors (SSE) from the estimated value.
- The prediction for leaf c is $\bar{y}_c = \frac{\sum_{i \in c} y_i}{N_c}$.
- Within each leave c, the MSE can be computed as $e_c = \frac{\sum_{i \in c} (y_i - \bar{y}_c)^2}{N_c}$
- The total MSE is $\mathcal{S} = \sum_{c \in \text{leaves(Tree)}} e_c$.
- The total SSE is $\mathcal{S} = \sum_{c \in \text{leaves(Tree)}} \sum_{i \in c} (y_i - \bar{y}_c)^2$. (preferred)

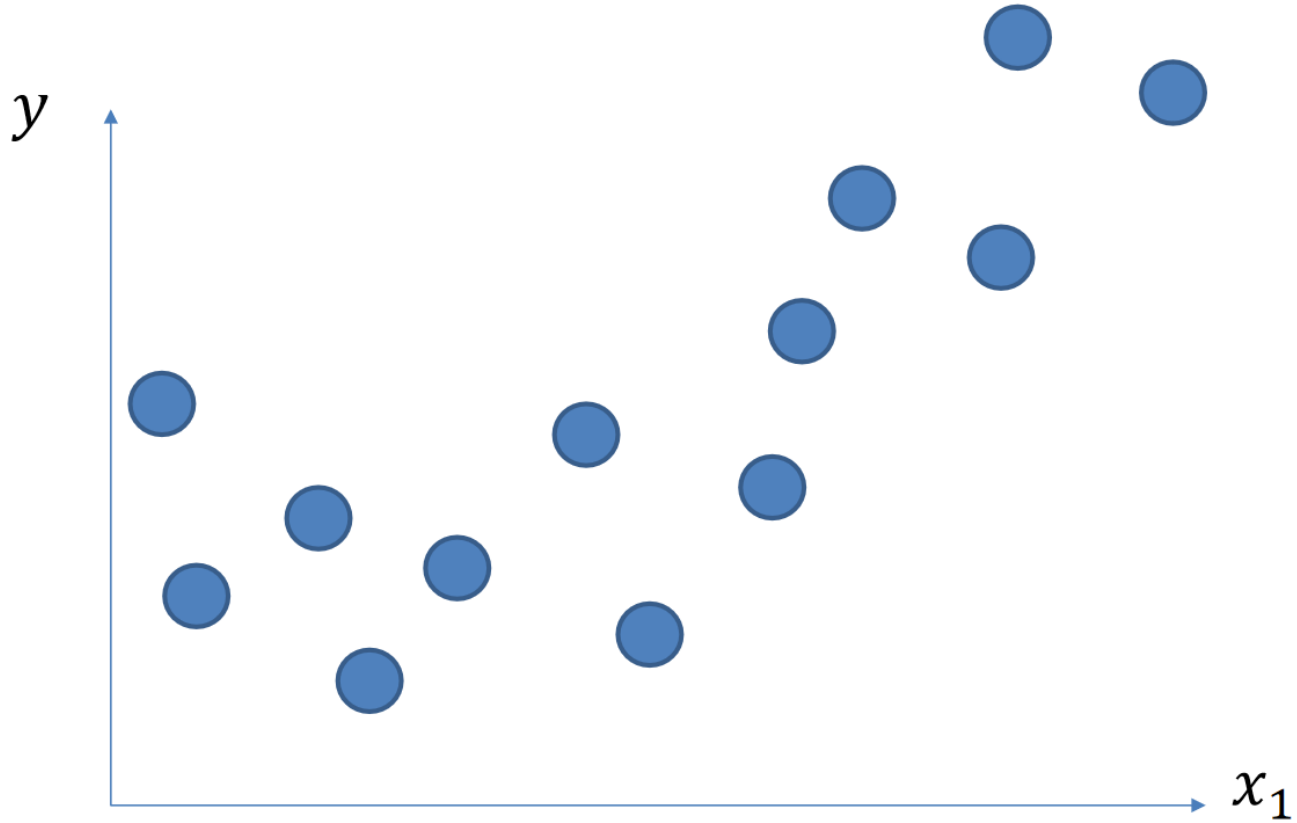
Regression Trees

Basic Regression-Tree-Growing Algorithm

1. Start with a single node containing all points. Calculate $\bar{y}$ and $\mathcal{S}$.
2. For each node,
   - If all the points in the node have the same value for all the independent variables, **stop**.
   - Otherwise, search over all binary splits of all variables for the one which will reduce $\mathcal{S}$ as much as possible.
     - If the largest decrease in $\mathcal{S}$ would be less than some threshold δ, or one of the resulting nodes would contain less than q points, **stop**.
     - Otherwise, **take that split**, creating two new nodes.
3. In each new node, go back to step 1.

Example: build a regression tree

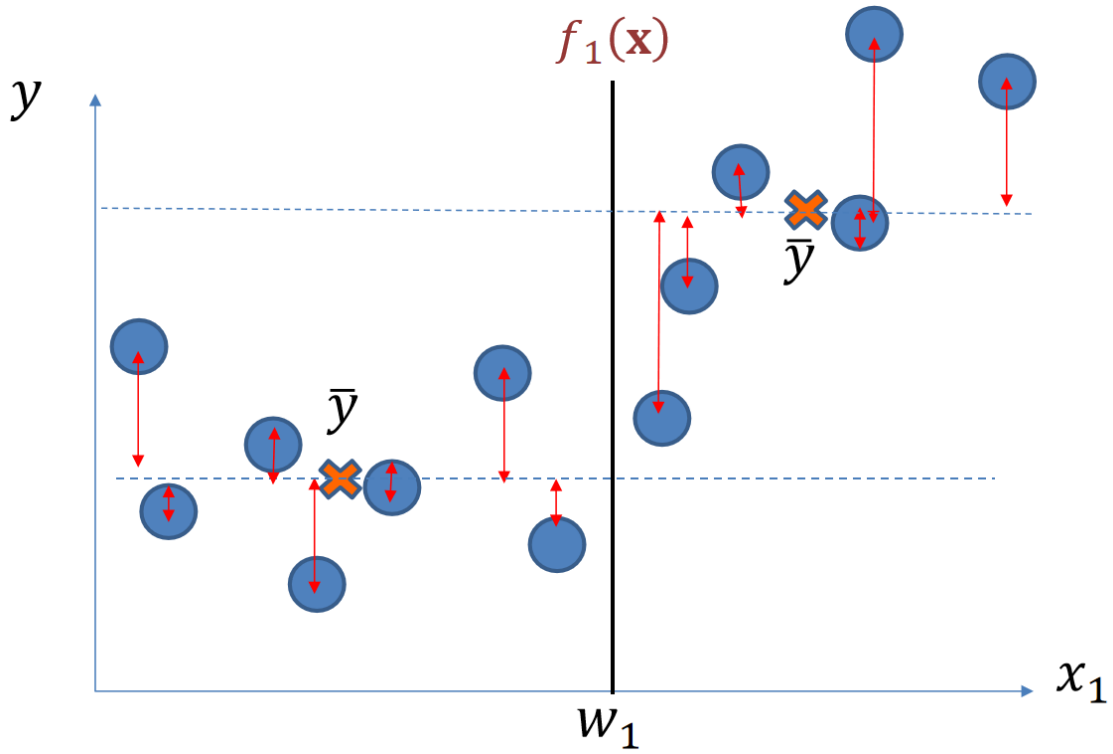
We consider building a regression tree based on the following 1-dimensional data.



- $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.
- $\mathcal{S} = \sum_{i=1}^{N}(y_i - \bar{y})^2$

Example: build a regression tree

- Consider a threshold w_1 that splits the root node into two child nodes c_L, c_R and compute:
  - $\bar{y}_L = \frac{1}{|c_L|}\sum_{i\in c_L} y_i, \quad \bar{y}_R = \frac{1}{|c_R|}\sum_{i\in c_R} y_i$
  - $\mathcal{S}_{w_1} = \sum_{i\in c_L}(y_i - \bar{y}_{c_L})^2 + \sum_{i\in c_R}(y_i - \bar{y}_{c_R})^2$

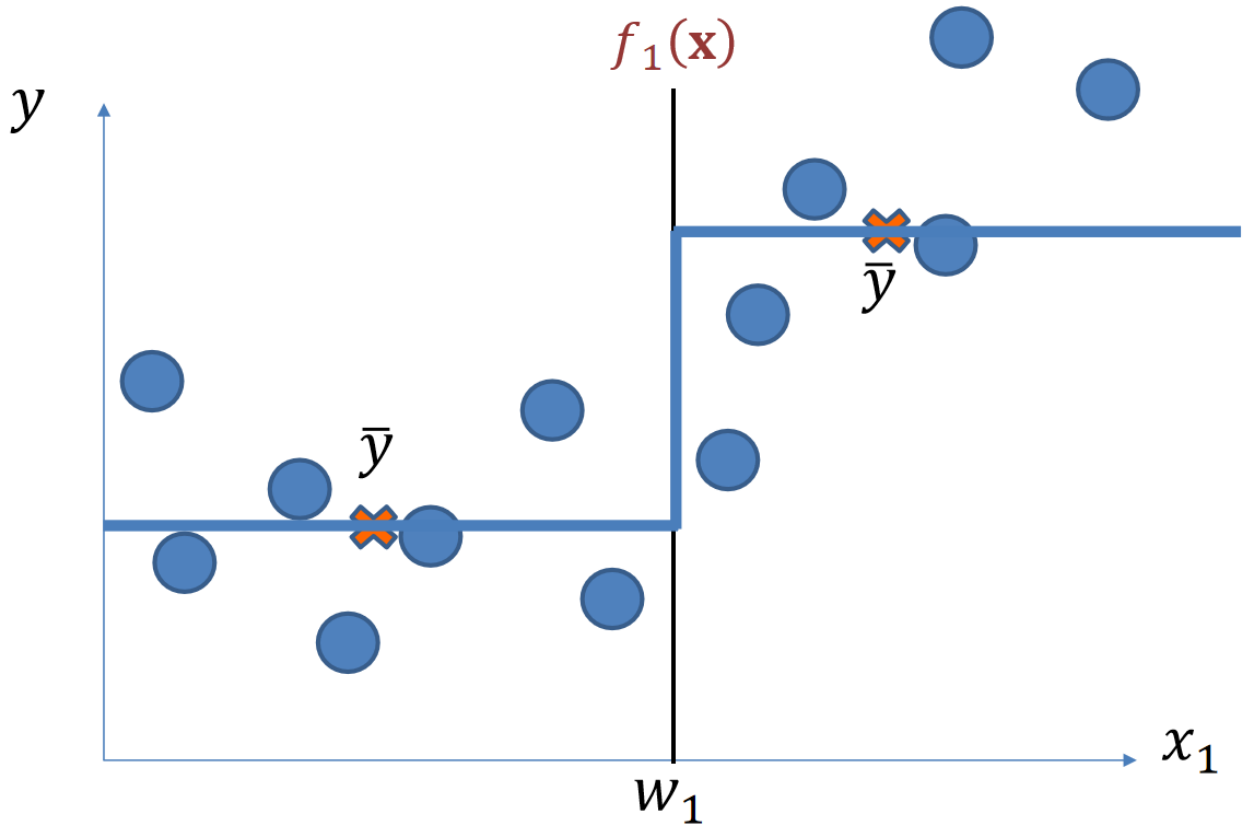


- We search for w_1 which leads to the largest decrease of $\mathcal{S}$:

$$w_1^* = \underset{w_1}{\text{argmax}} \; S - S_{w_1}$$

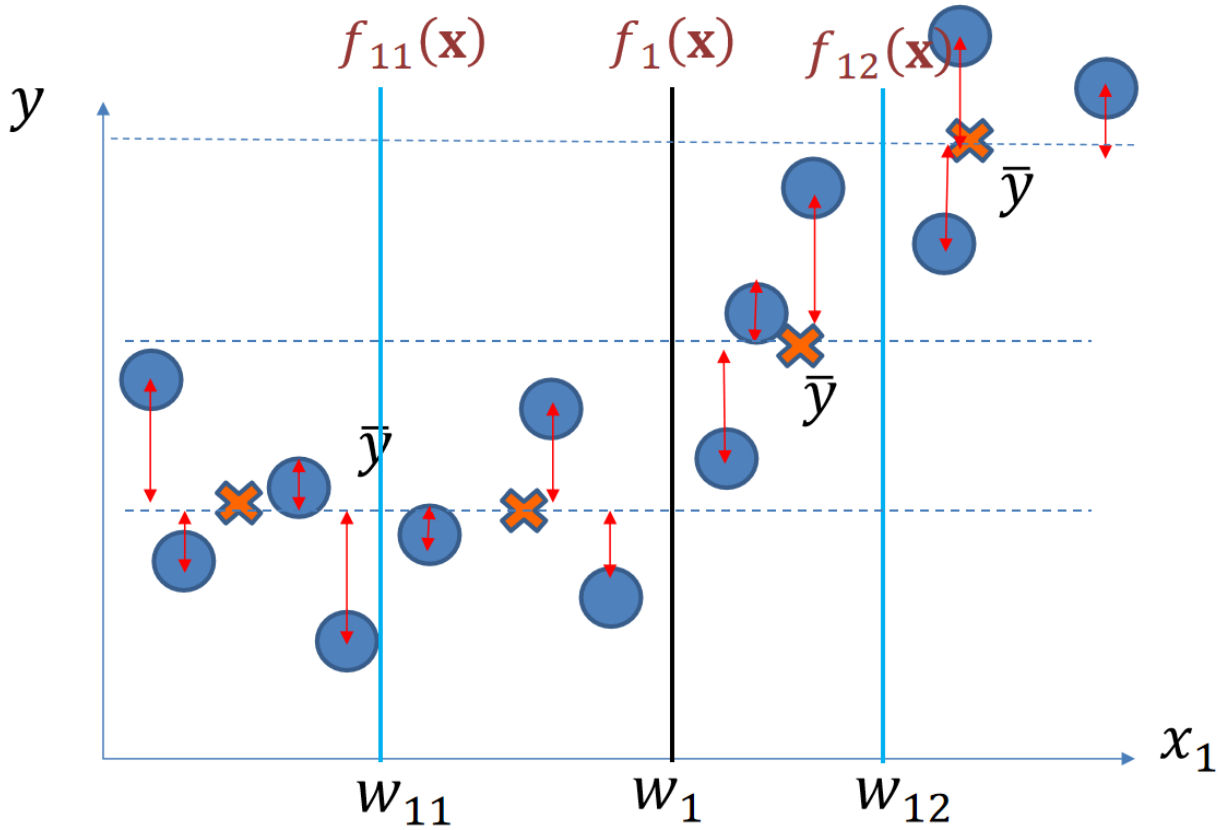
Example: build a regression tree

Repeat the above steps for each child node.



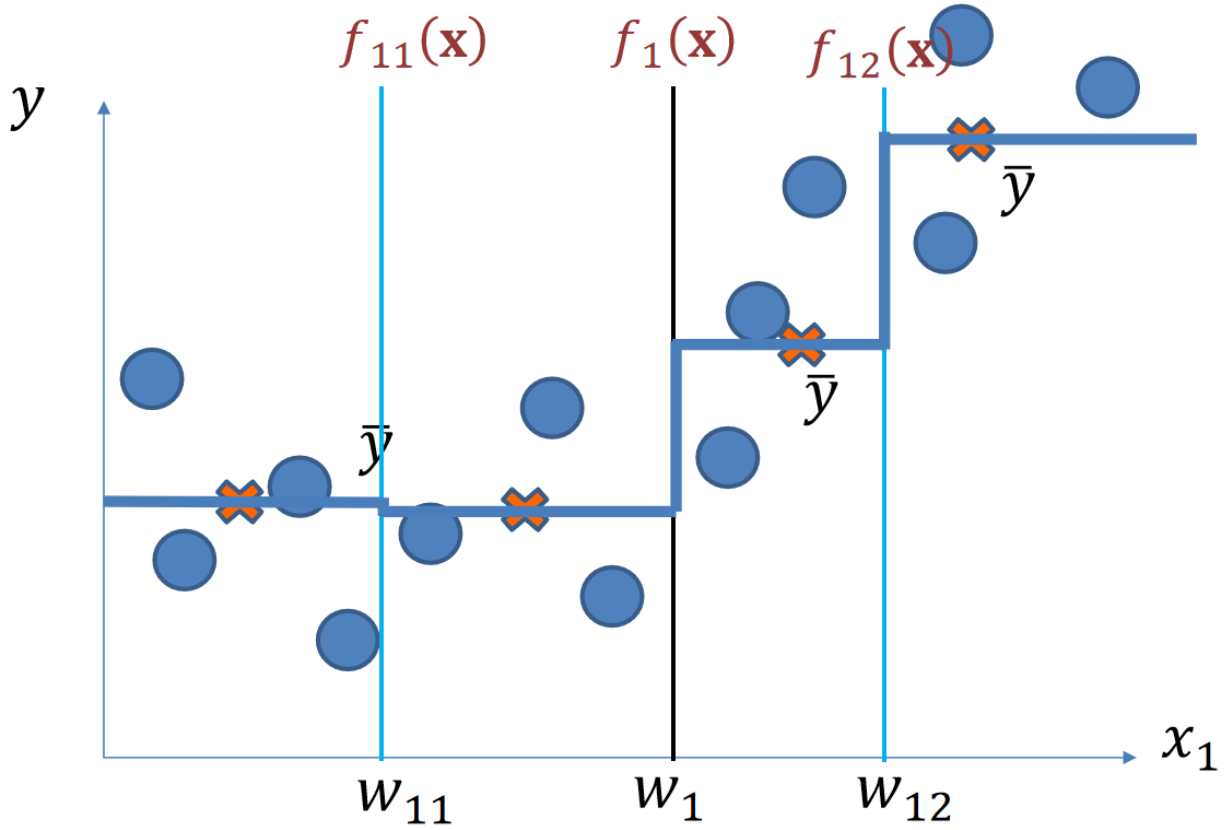
Example: build a regression tree

Repeat the above steps for each child node.



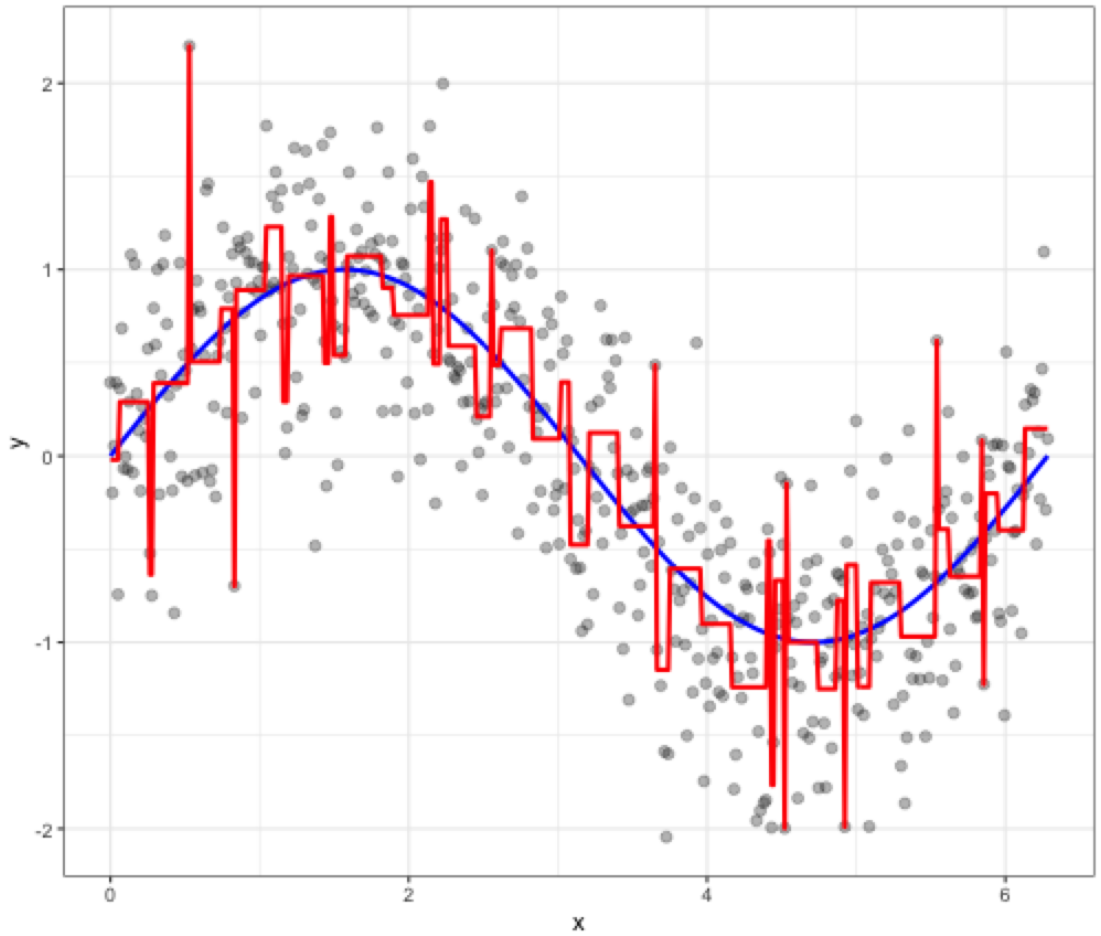
Example: build a regression tree

Repeat the above steps for each child node, until some stop conditions are satisfied.



Overfitting

Trees have a tendency to overfit to training data, such that the prediction error on testing data is likely to be high.
As shown in the following example, the prediction of the constructed tree is highly non-smooth.



Overfitting and Pruning

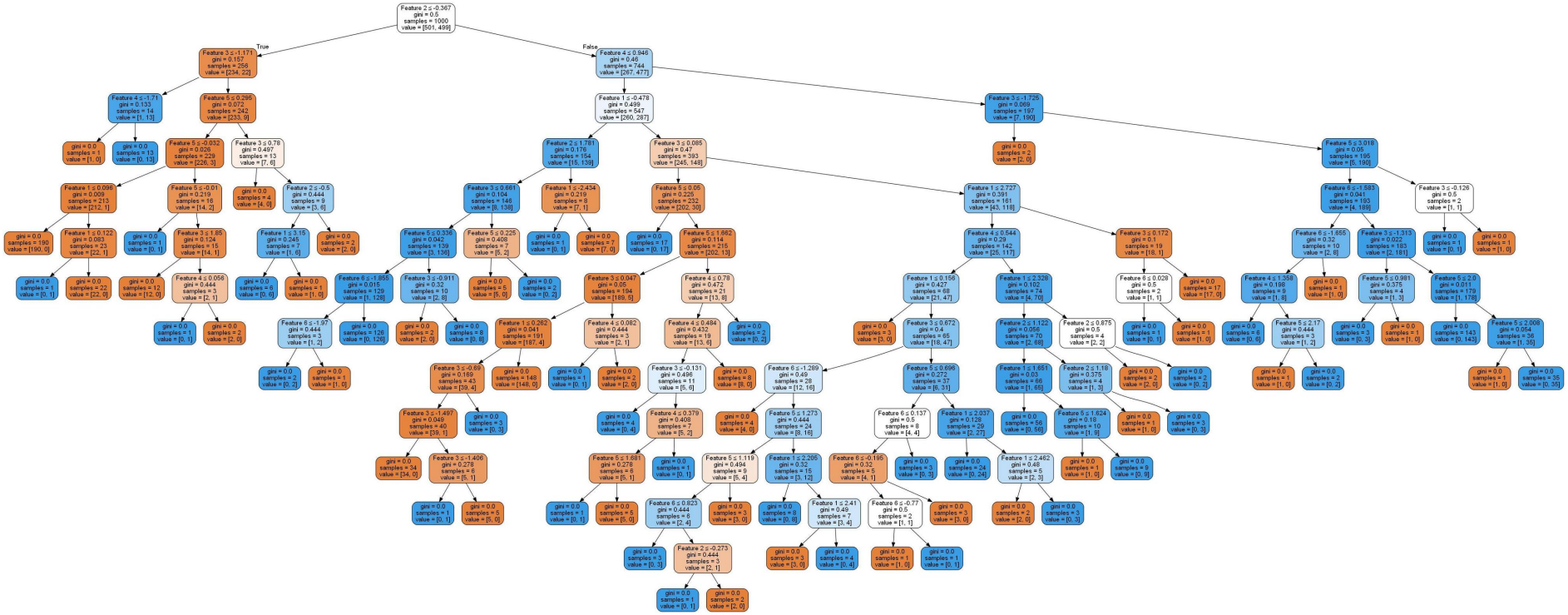
An effective approach to alleviate overfitting is pruning.
The general procedure of pruning is as follows:

- Split training data further into training and validation sets
- Grow a deep tree based on the training set
- Do until further pruning is harmful:
  - Evaluate the impact on the validation set of pruning each possible node
  - Greedily remove the node if that improves the validation accuracy most.

Reference: `https://faculty.cc.gatech.edu/~bboots3/CS4641-Fall2018/Lecture2/02_DecisionTrees.pdf`

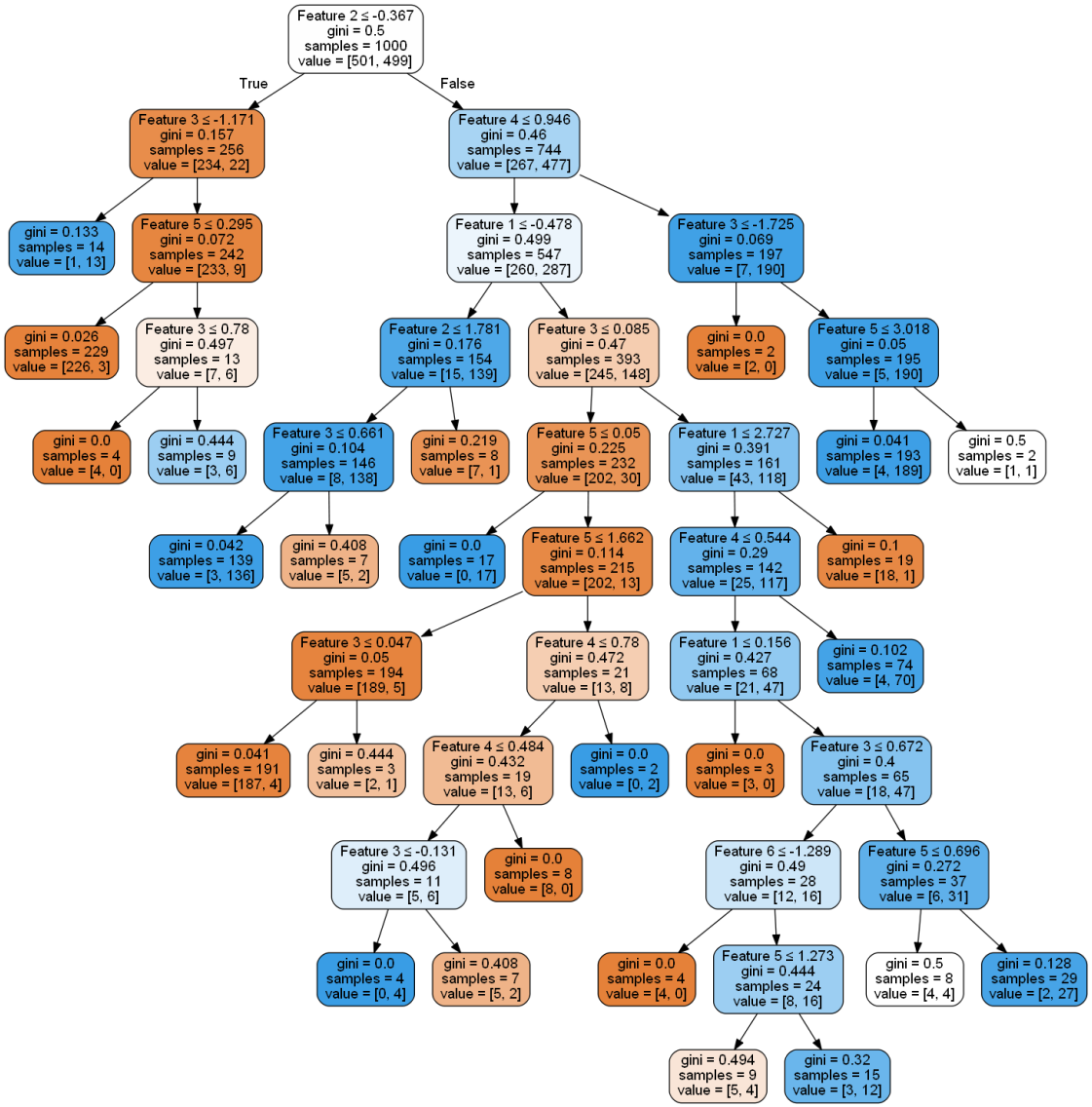
Overfitting and Pruning

A complex tree before pruning



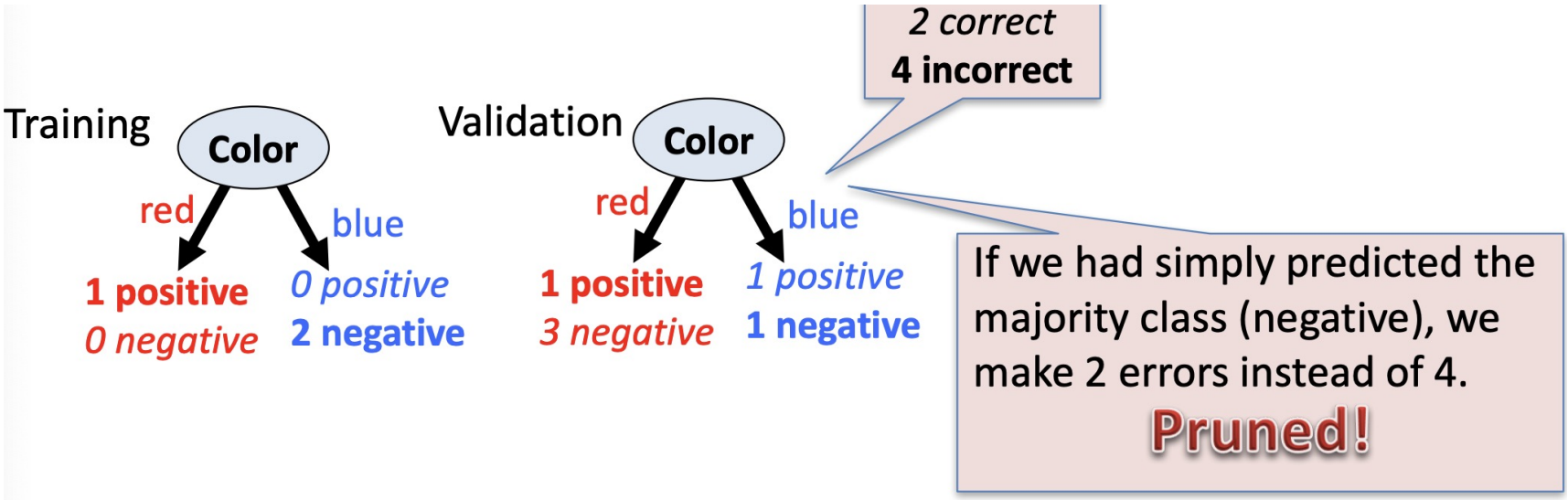
Overfitting and Pruning

A pruned tree

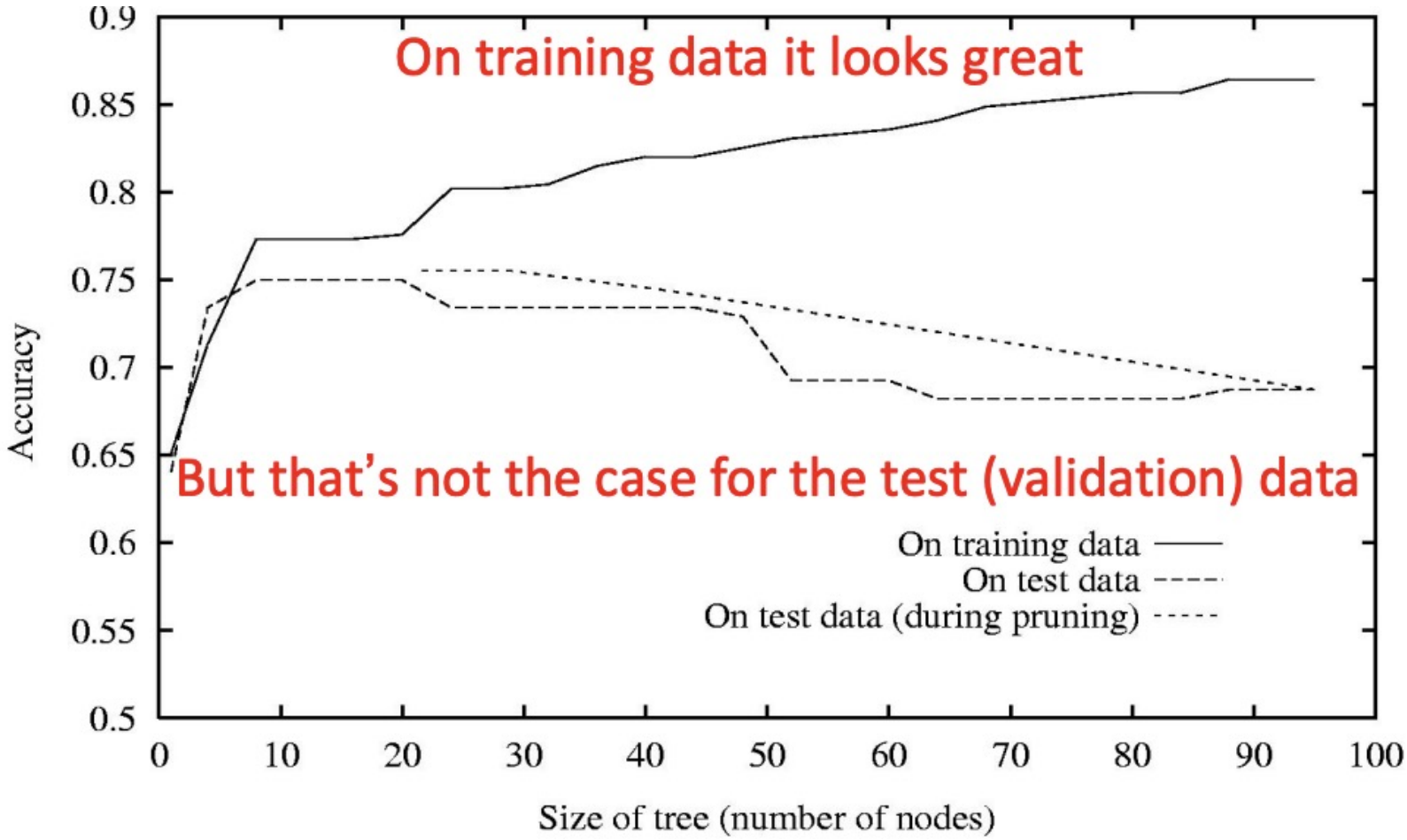


Overfitting and Pruning

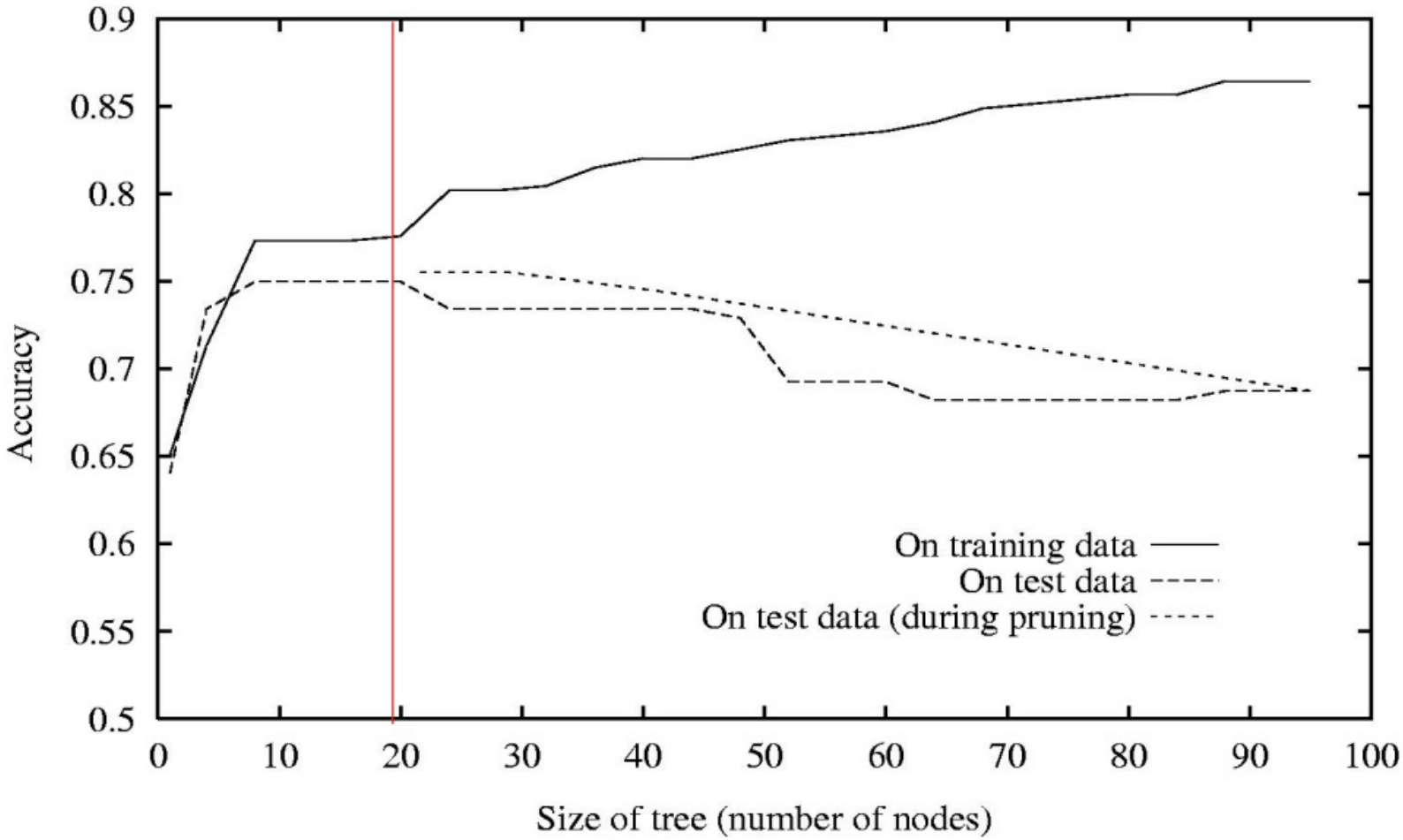
- Pruning of the decision tree is done by replacing a whole subtree with a leaf node
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf
- For example,



Effect of Reduced-Error Pruning



Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

Summary of Decision Trees

Advantages:

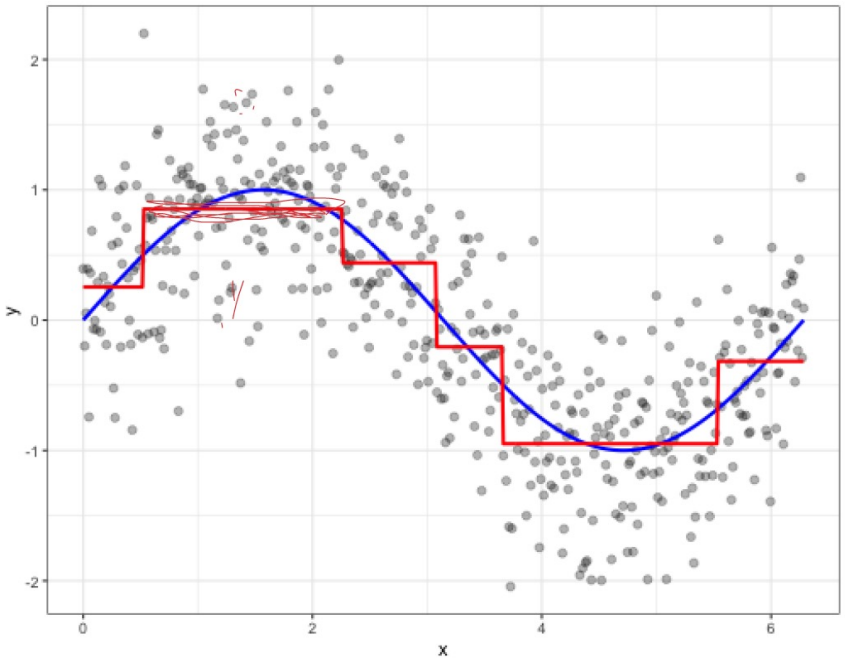
- Easy to understand (interpretability)
- Useful in data exploration (rule extraction)
- Less data cleaning/pre-processing required
- Data type is not a constraint
- Non-parametric method

Disadvantages:

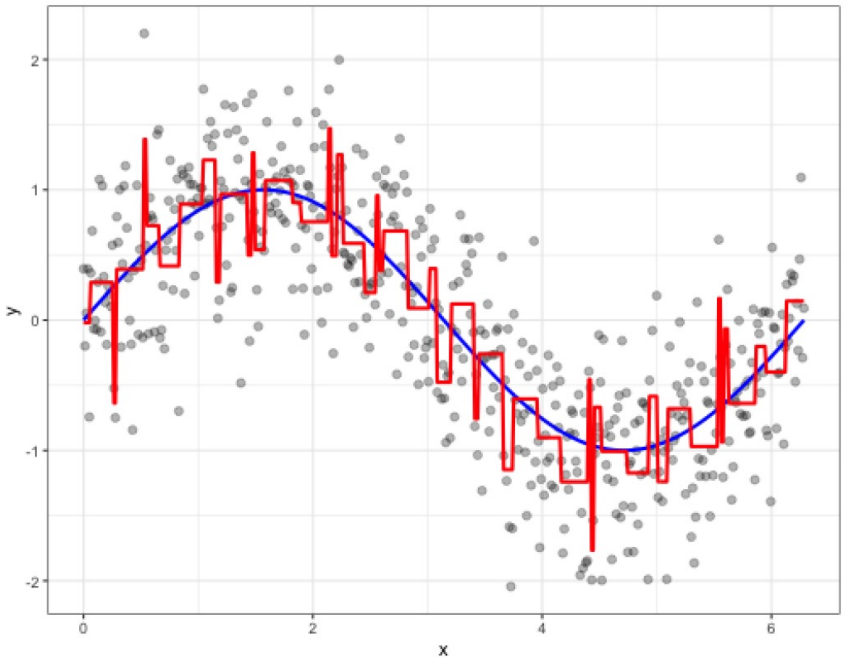
- Overfitting: Overfitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model parameters and pruning.
- Continuous variables: While working with continuous numerical variables, decision tree loses information when it categorizes variables into categories (quantization error).

Drawbacks of single decision tree

Single pruned trees are poor predictors



Single deep trees are overfitting

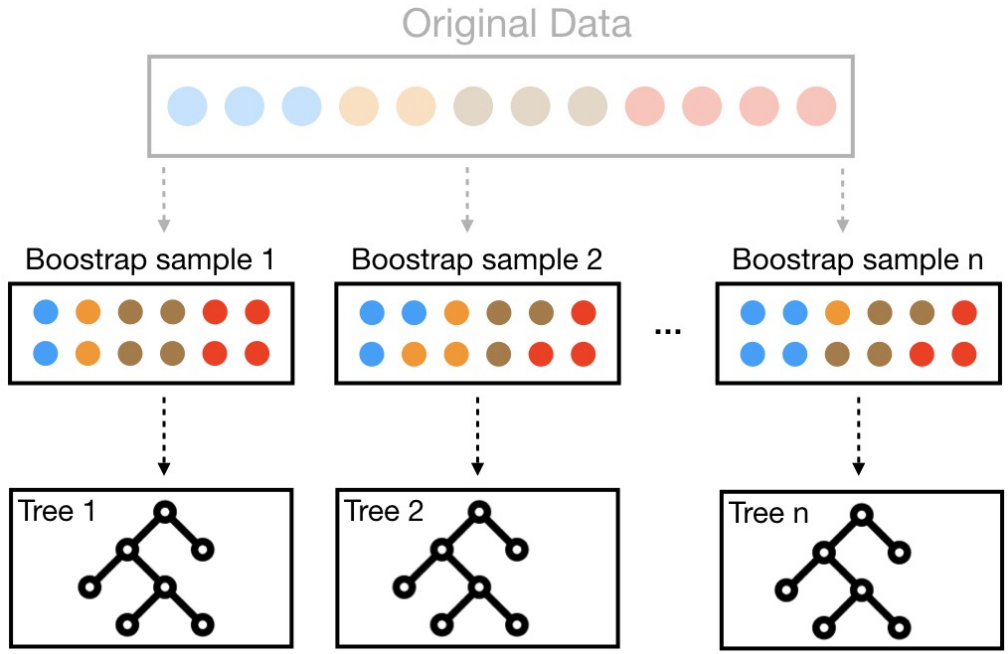


Ensemble models

- To address generalization issues of single decision trees, we introduce ensemble models.
- The basic idea is constructing many diverse decision trees, then combine their predictions as the final prediction (majority for classification, or average for regression).
- The philosophy is that the wisdom of the crowd is likely higher than singles. 三个臭皮匠，胜过诸葛亮
- We introduce two ensemble models of decision trees:
  - Bootstrap Aggregating (Bagging)
  - Random Forests

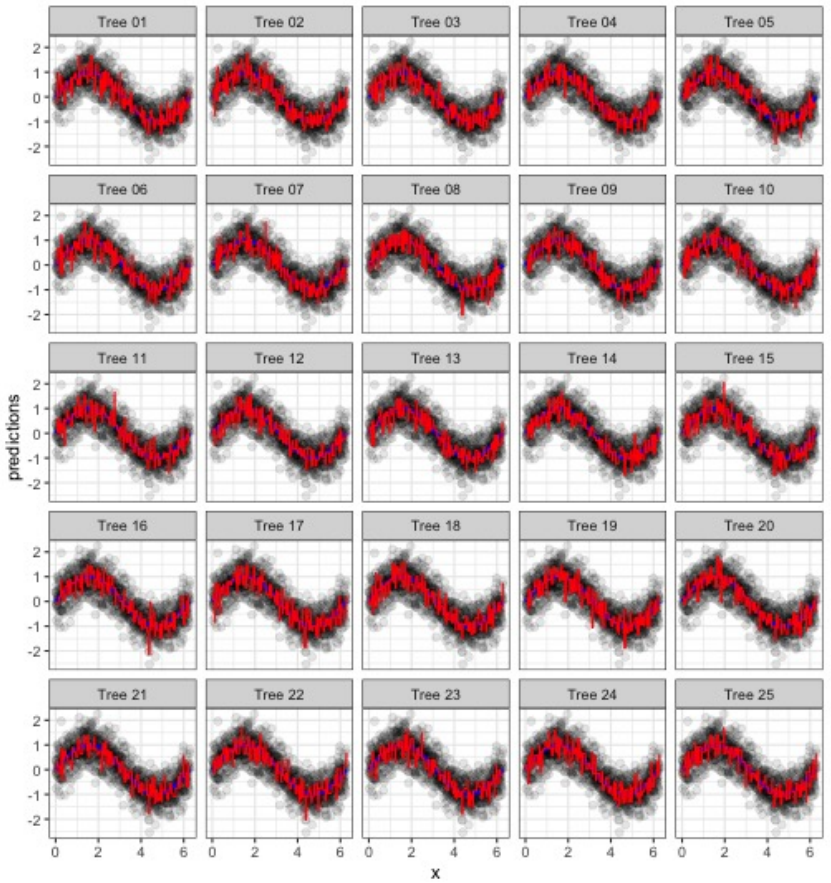
Bootstrap Aggregating: wisdom of the crowd (Bagging)

- **Step 1**: Sample records with replacement (aka "bootstrap" the training data), to obtain several diverse training data sets (data-level randomness)
- **Step 2**: Fit an overgrown tree to each resampled training data set.



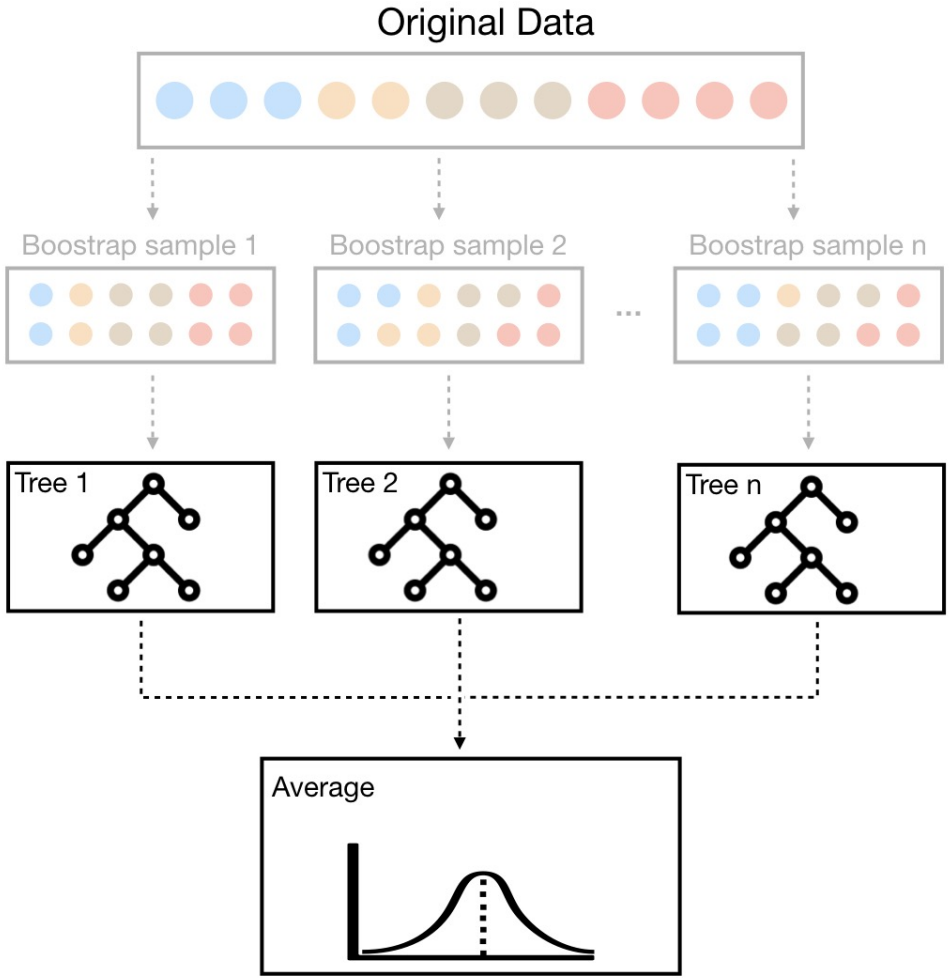
Bootstrap Aggregating: wisdom of the crowd (Bagging)

- **Step 1**: Sample records with replacement (aka "bootstrap" the training data), to obtain several diverse training data sets (data-level randomness)
- **Step 2**: Fit an overgrown tree to each resampled training data set, and we obtain several diverse decision trees



Bootstrap Aggregating (Bagging)

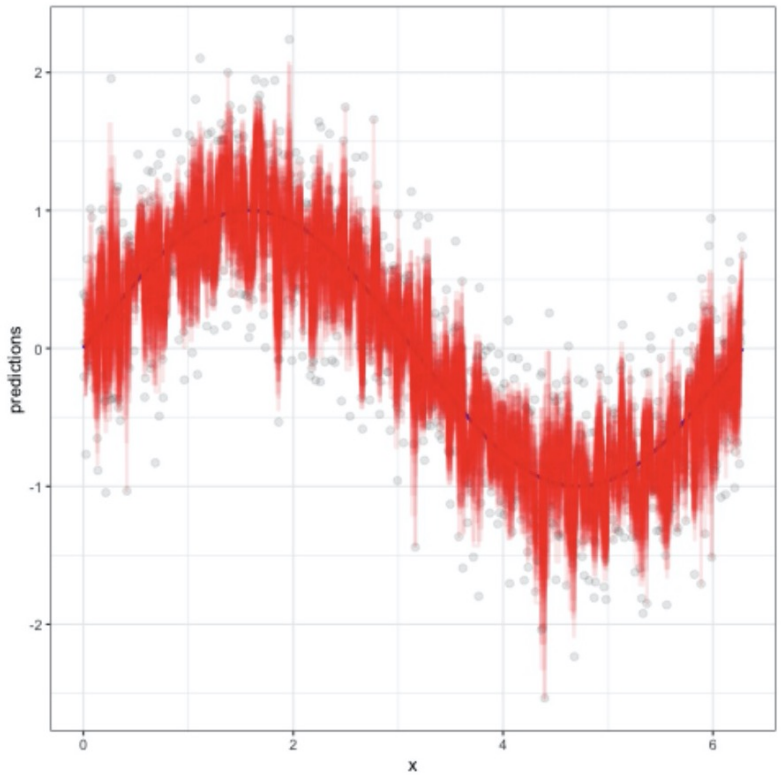
- **Step 1**: Sample records with replacement (aka "bootstrap" the training data), to obtain several diverse training data sets (data-level randomness)
- **Step 2**: Fit an overgrown tree to each resampled training data set
- Aggregate the predictions of all single trees (majority or average)



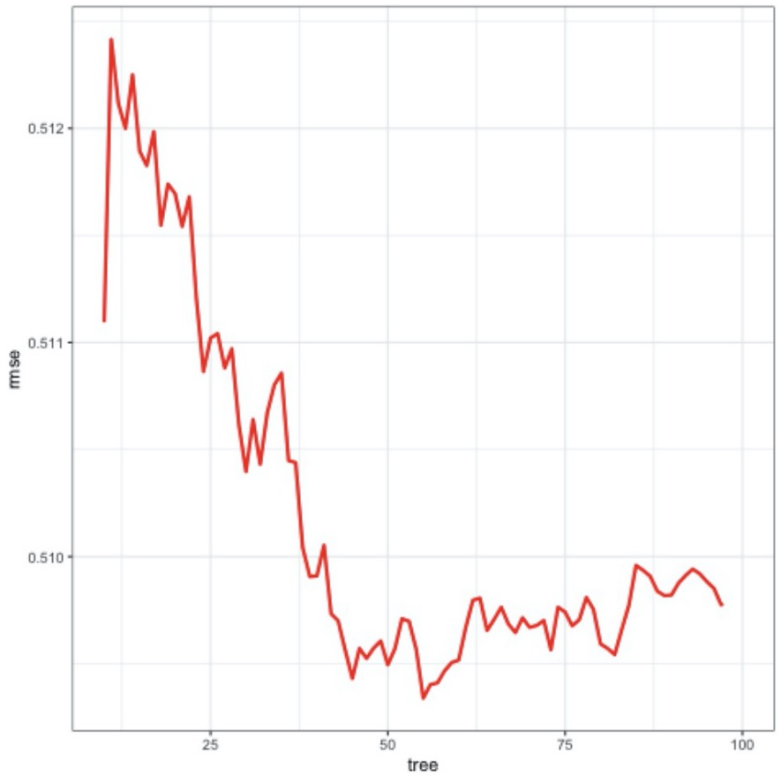
Bootstrap Aggregating (Bagging)

Bagging with more single trees will decrease the prediction error.

As we add more trees...

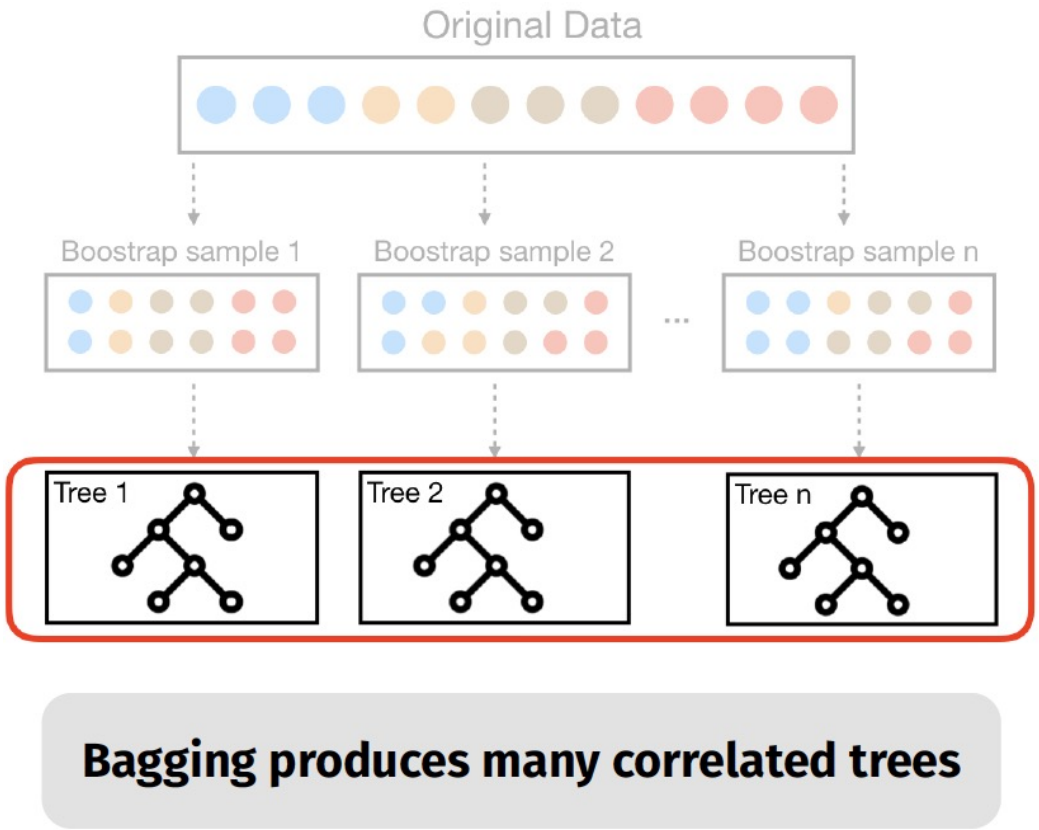


our average prediction error reduces



Bootstrap Aggregating (Bagging)

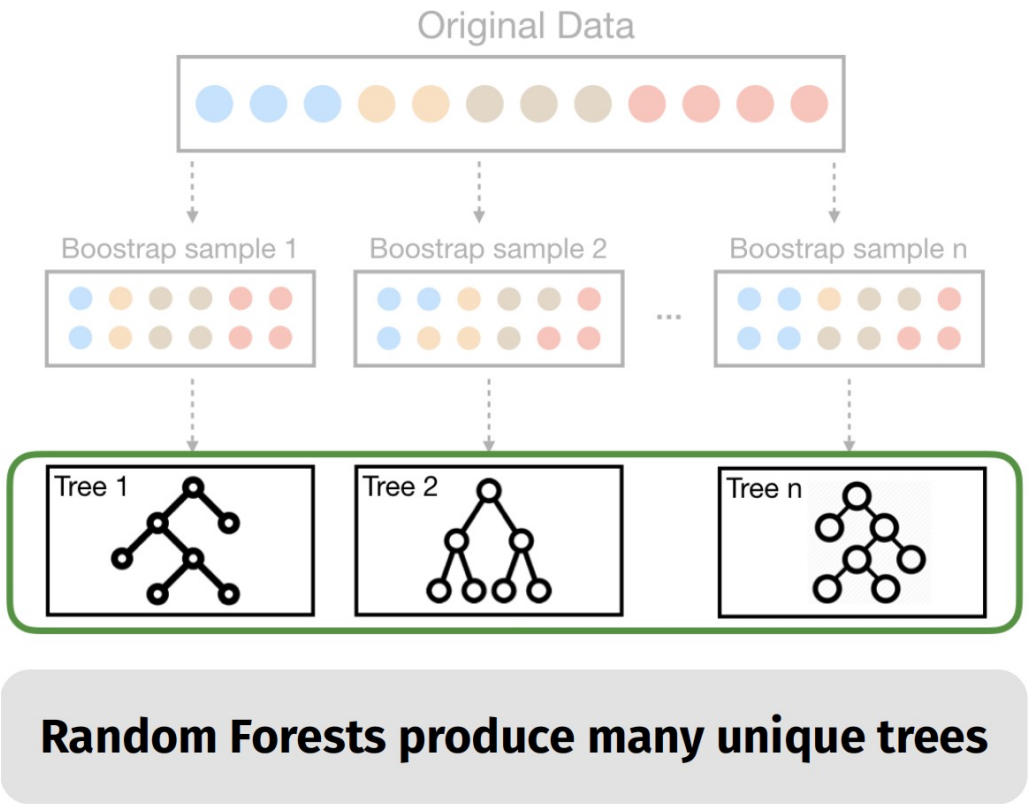
However, since there are many shared samples between two resampled training data sets, Bagging is likely to produce many correlated trees.
The diversity is not large enough.



Random Forest

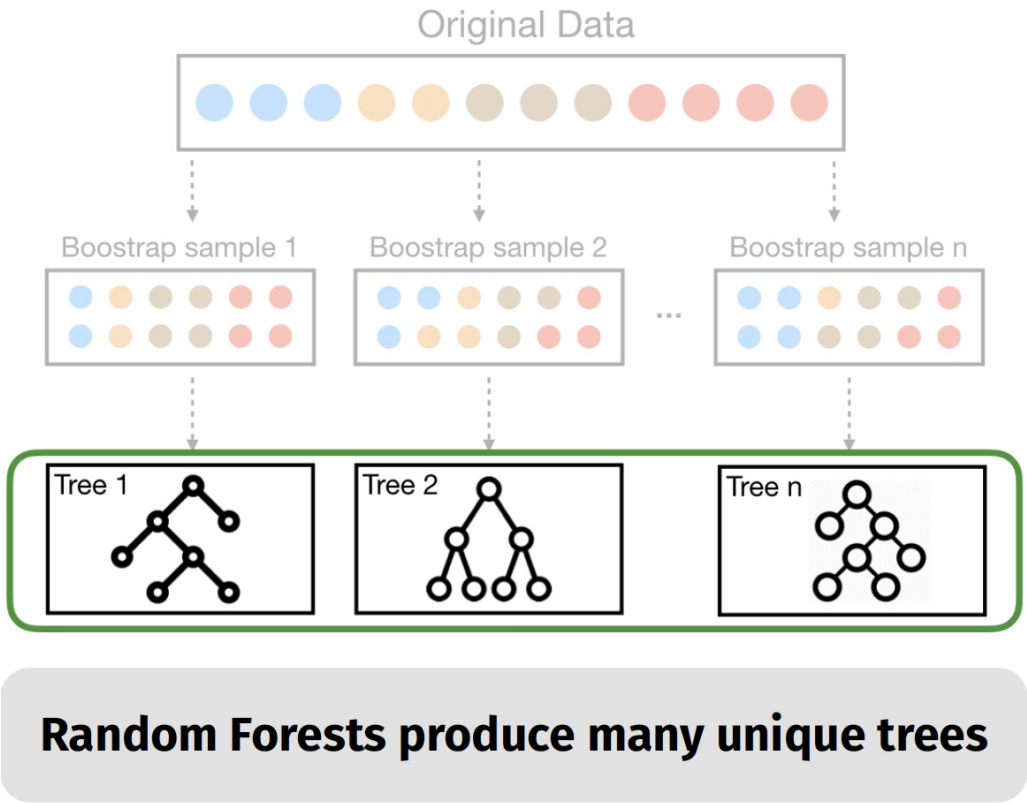
- To reduce the correlation among the trees produced by Bagging, we introduce split-attribute randomization into the model.
- It is called Random Forests. Many trees form a forest!

- Follow a similar bagging process but...
- Each time a split is to be performed, the search for the split attribute is limited to a random subset of m of the N attributes
  - For regression trees: $m = \frac{N}{3}$
  - For classification trees: $m = \sqrt{N}$



Random Forest

- Bagging introduces randomness into the data-level
- Random forests introduces randomness into both the data-level and attribute level
- Prediction error: Random forest $<$ Bagging $<$ single trees



The ensemble models can alleviate overfitting. A brief understanding is that each single decision tree in ensemble models overfits a different data set and attributes, to avoid the overfitting to the fixed original data set as did in single decision trees.

Random Forest v.s. Decision Tree

Aspect	Random Forest	Decision Tree
Nature	Ensemble of multiple decision trees	Single decision tree
Bias-Variance Trade-off	Lower variance, reduced overfitting	Higher variance, prone to overfitting
Predictive Accuracy	Generally higher due to ensemble	Prone to overfitting, may vary
Robustness	More robust to outliers and noise	Sensitive to outliers and noise
Training Time	Slower due to multiple tree construction	Faster as it builds a single tree
Interpretability	Less interpretable due to ensemble	More interpretable as a single tree
Feature Importance	Provides feature importance scores	Provides feature importance, but less reliable
Usage	Suitable for complex tasks, high-dimensional data	Simple tasks, easy interpretation

Further reading

- An Online dynamic slides about decision tree
- Function and demos of decision tree in sklearn
- Document of decision tree in sklearn