

Introduction to Machine Learning

Lecture 5: Decision trees and Random Forests

Harbour.Space University
February 2021

Nikolay Karpachev

Outline

1. Decision tree

- Definition
- Constructing decision tree
- Information criteria
- Pruning

2. Composition methods

- Bootstrap
- Random subspace method

3. Random Forest

Classification (regression) models so far

- Linear classification / linear regression
 - Very efficient
 - Achieve reasonably good quality
 - Can capture only linear dependencies
- Naive bayes model
 - Very simple
 - Feature independence assumption does not hold in real data

Decision Tree: intuition

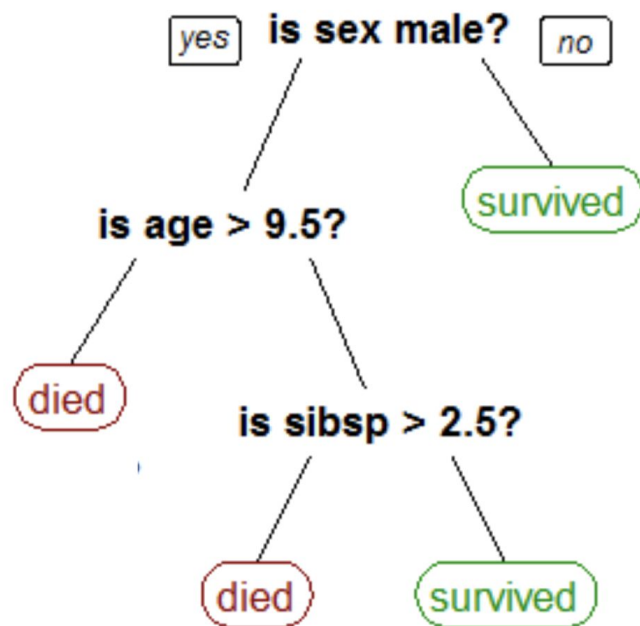
Decision tree

Decision tree is a logic-based classifier that is able to capture nonlinear dependencies

Decision tree

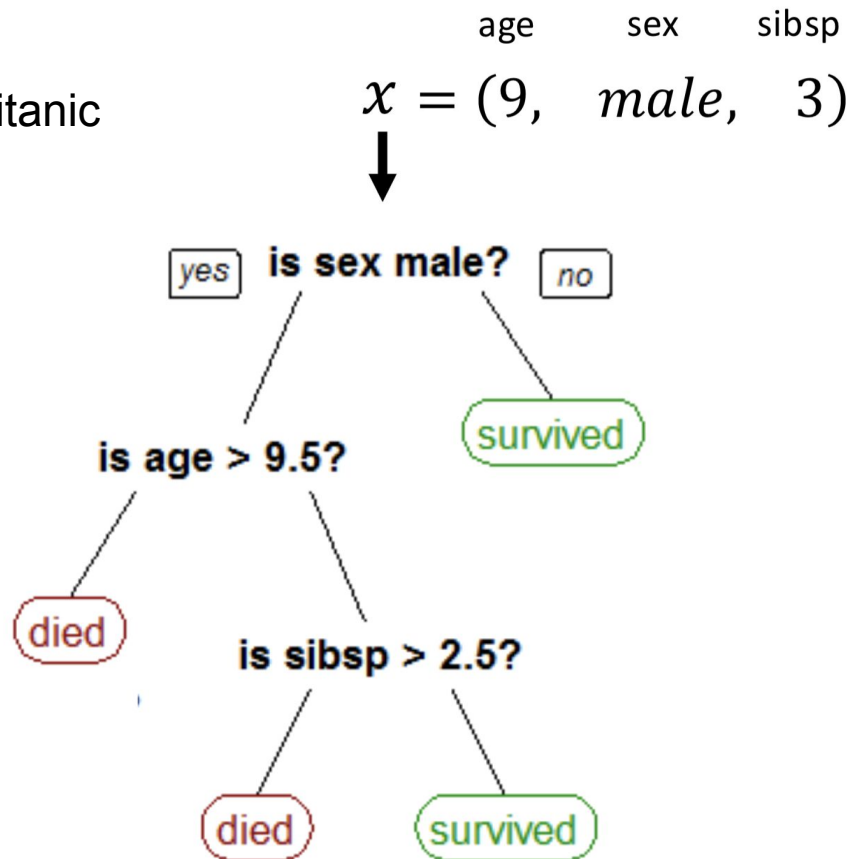
Example: Titanic dataset

$$x = (9, \text{male}, 3)$$



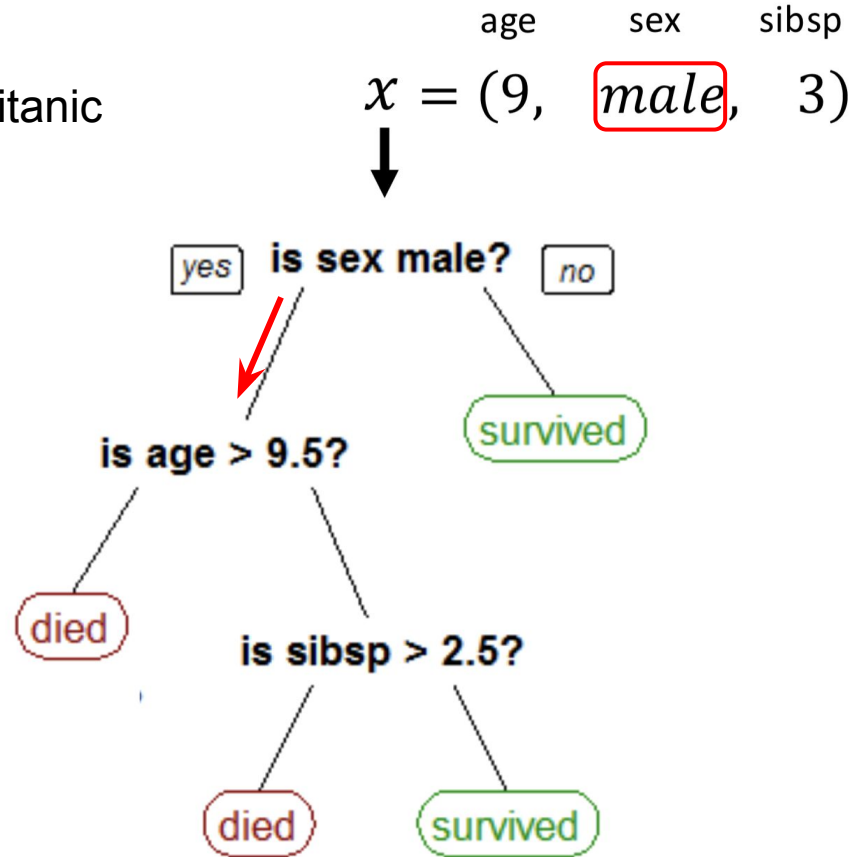
Decision tree

Example: Titanic dataset



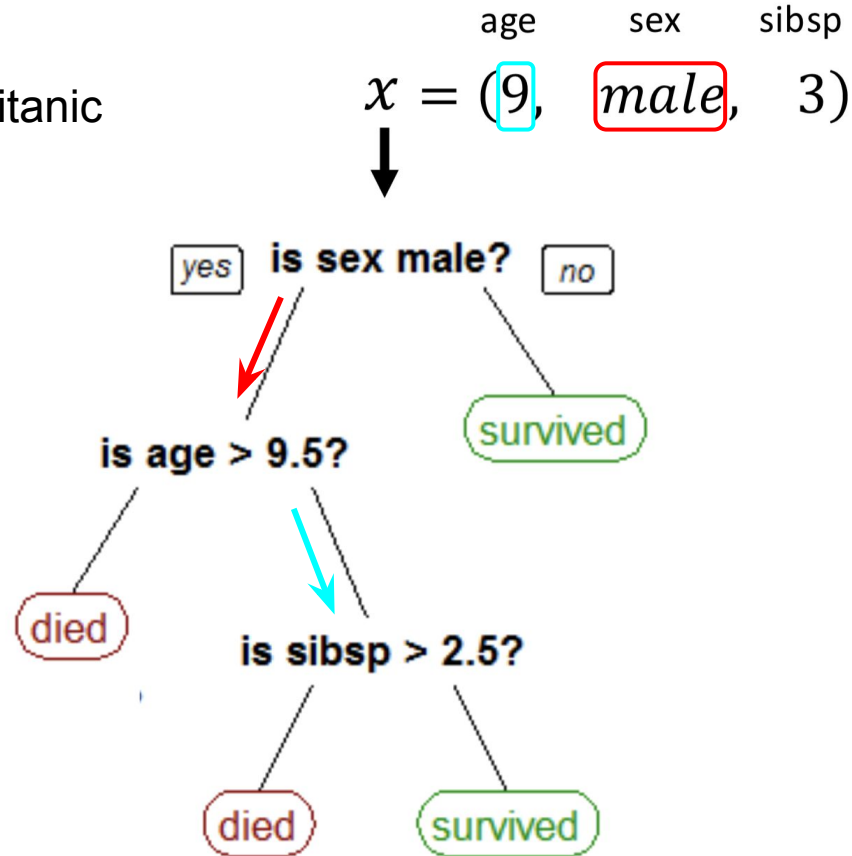
Decision tree

Example: Titanic dataset



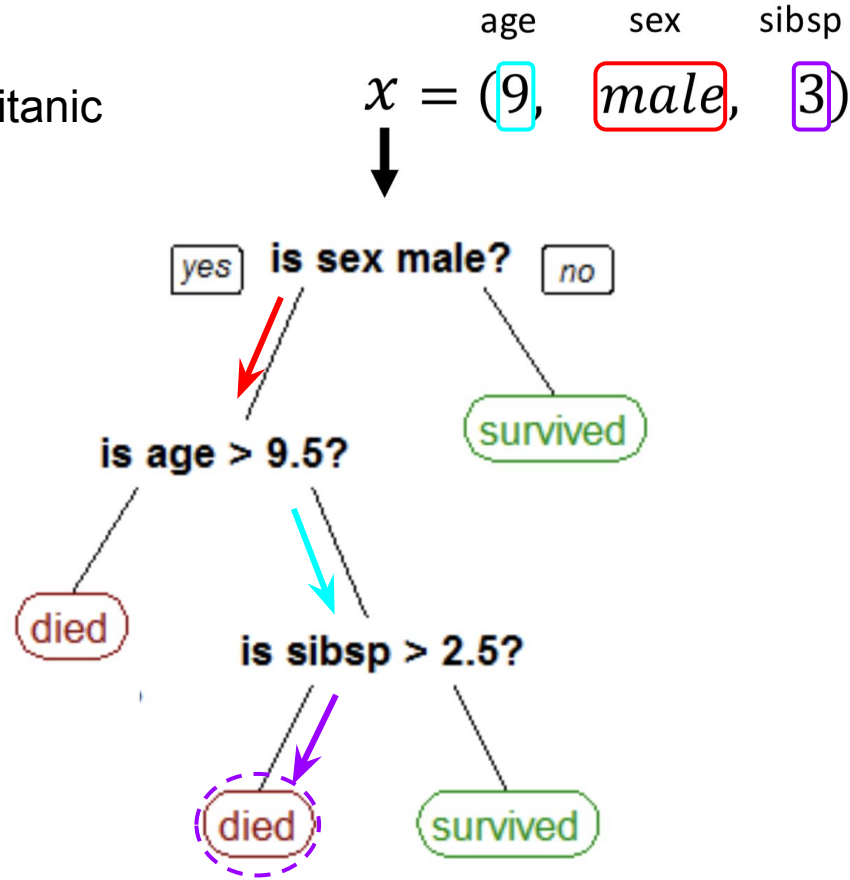
Decision tree

Example: Titanic dataset



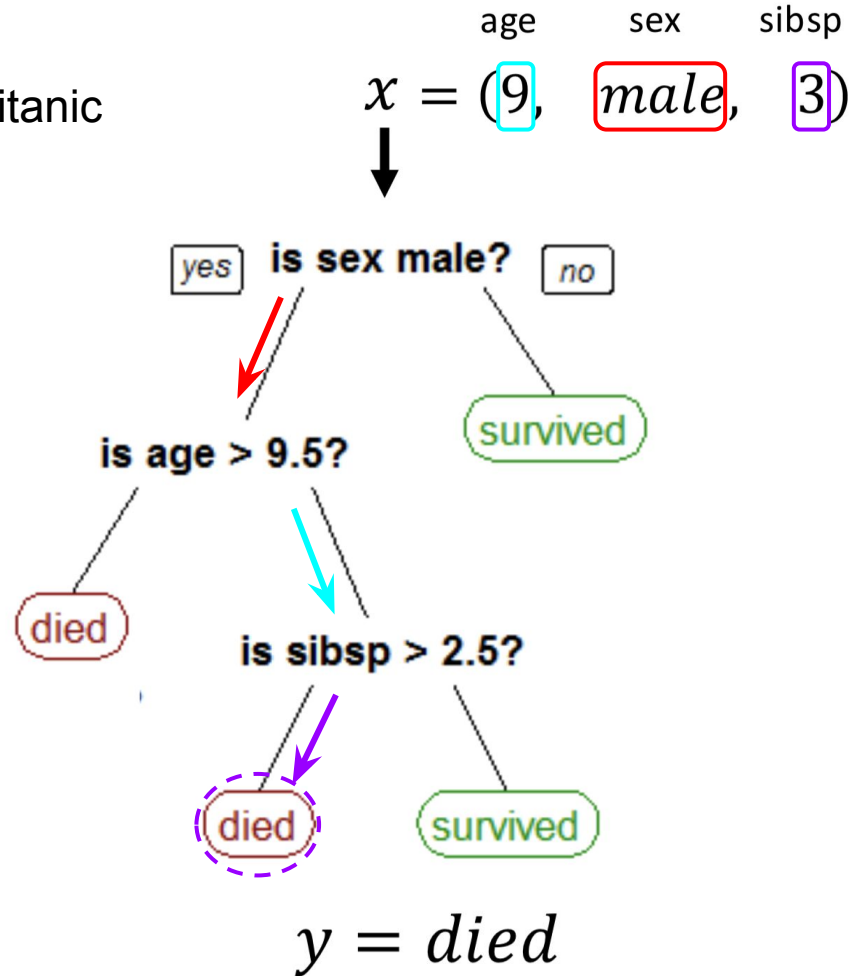
Decision tree

Example: Titanic dataset



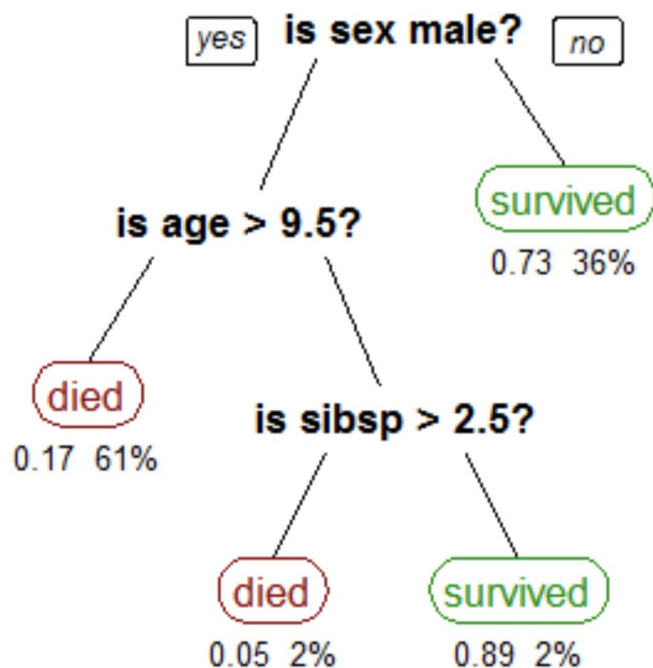
Decision tree

Example: Titanic dataset



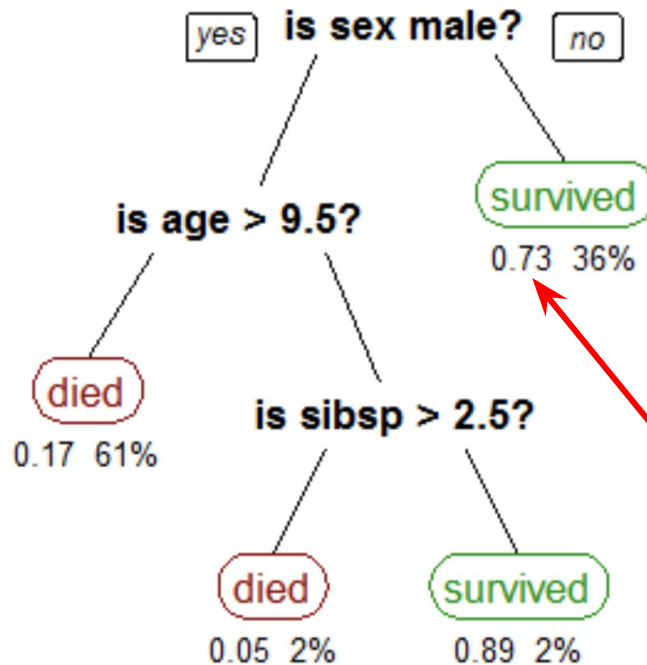
Decision tree in classification

Example: Titanic dataset



Decision tree in classification

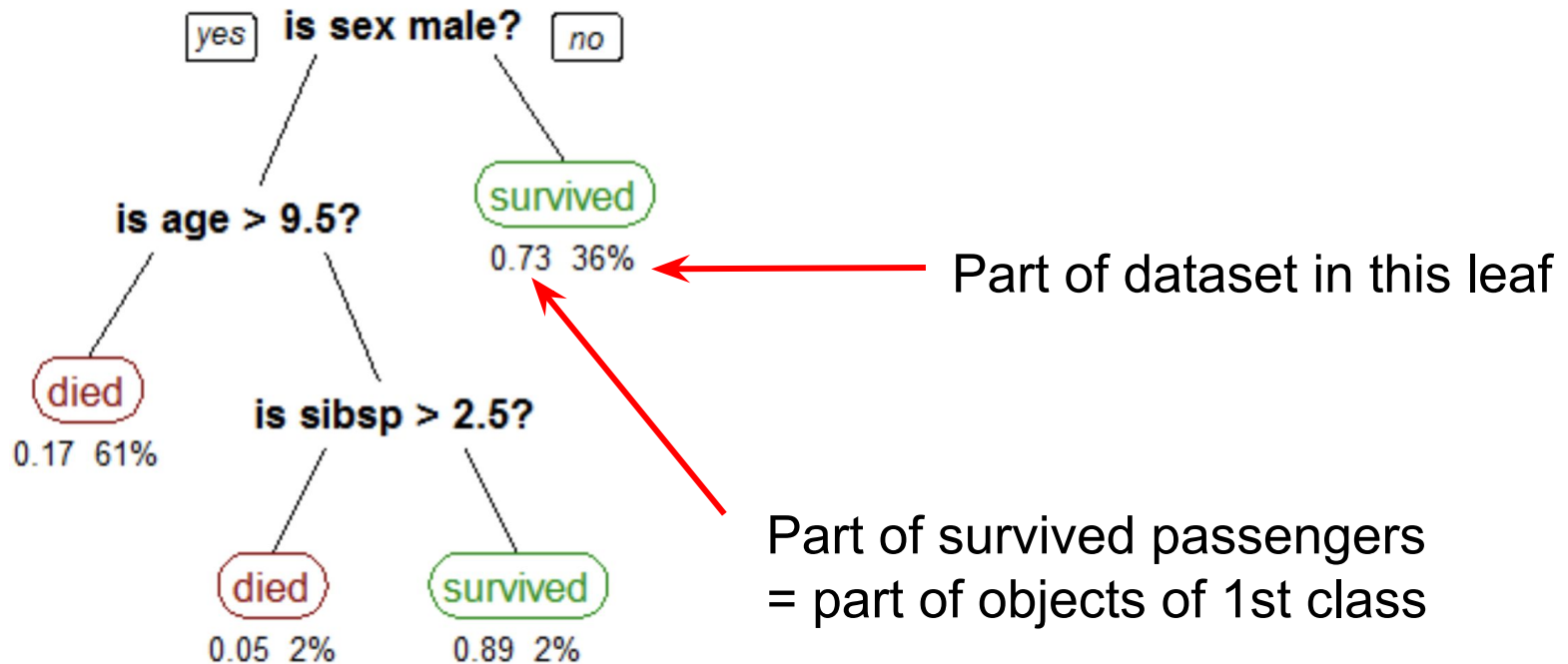
Example: Titanic dataset



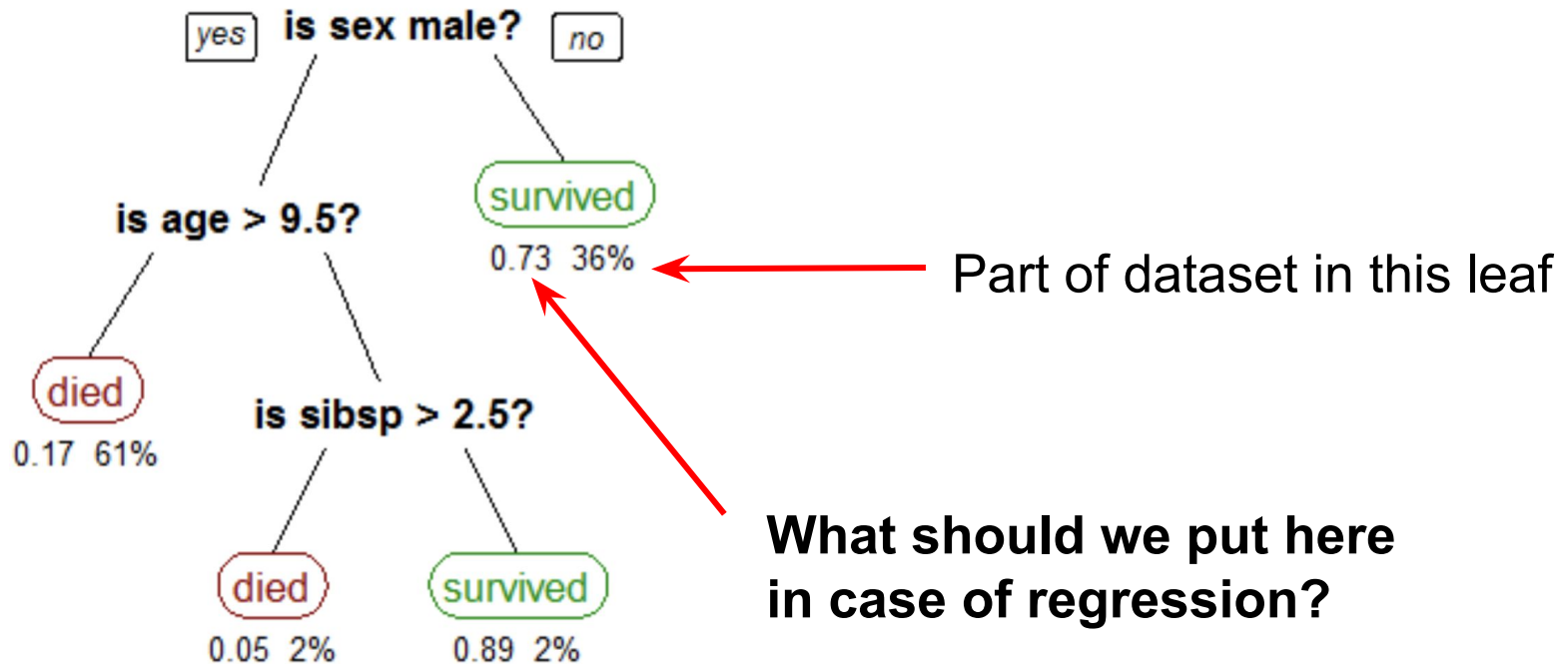
Part of survived passengers
= part of objects of 1st class

Decision tree in classification

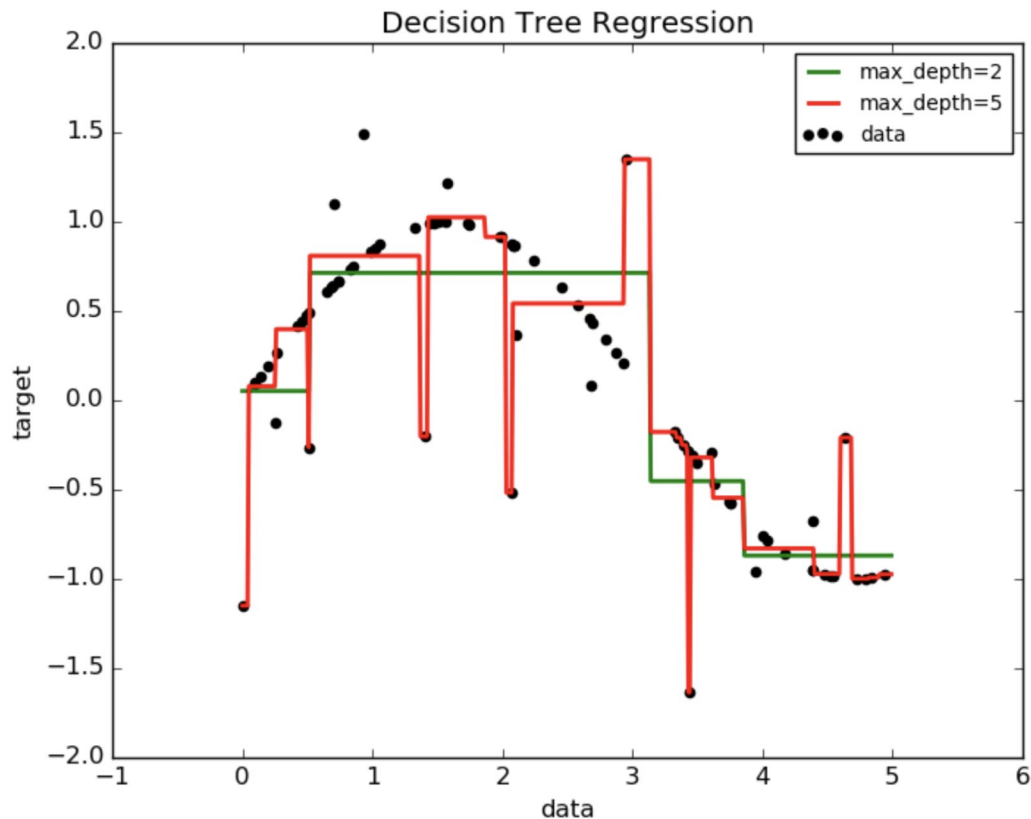
Example: Titanic dataset



Decision tree in regression



Decision tree in regression

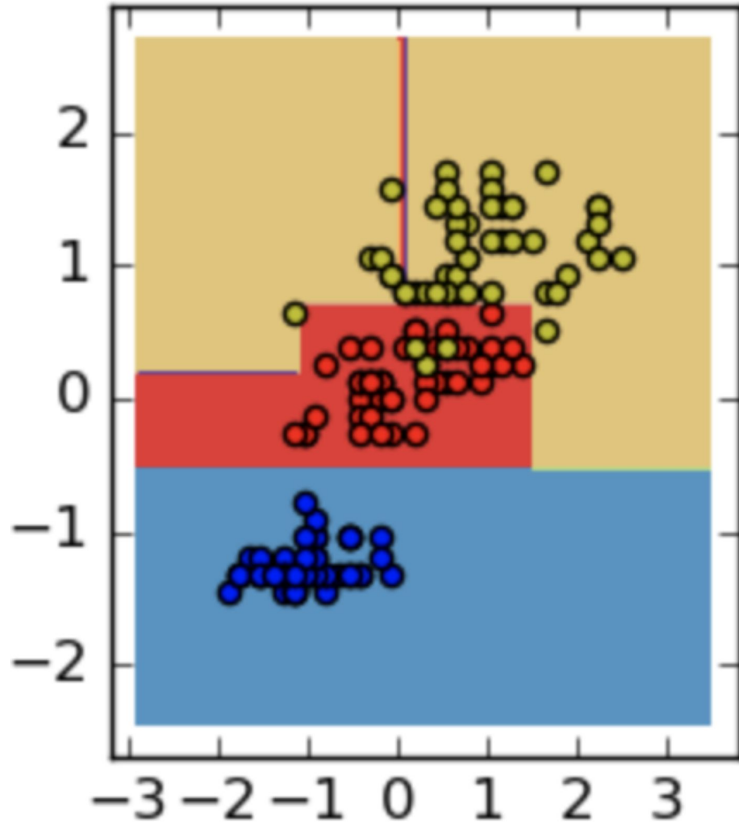


Green - decision tree of depth 2

Red - decision tree of depth 5

Every leaf corresponds to some constant.

Decision tree surface



Classification problem with 3 classes and 2 features.

Decision Tree

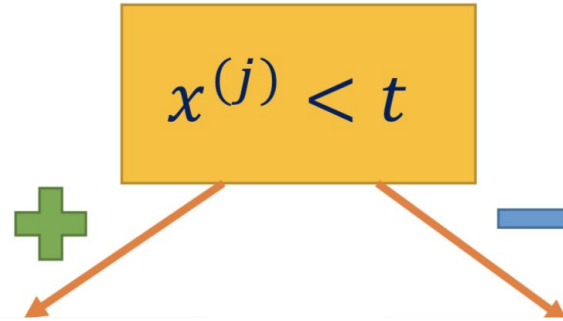
construction procedure

Constructing decision trees

$$x^{(j)} < t$$

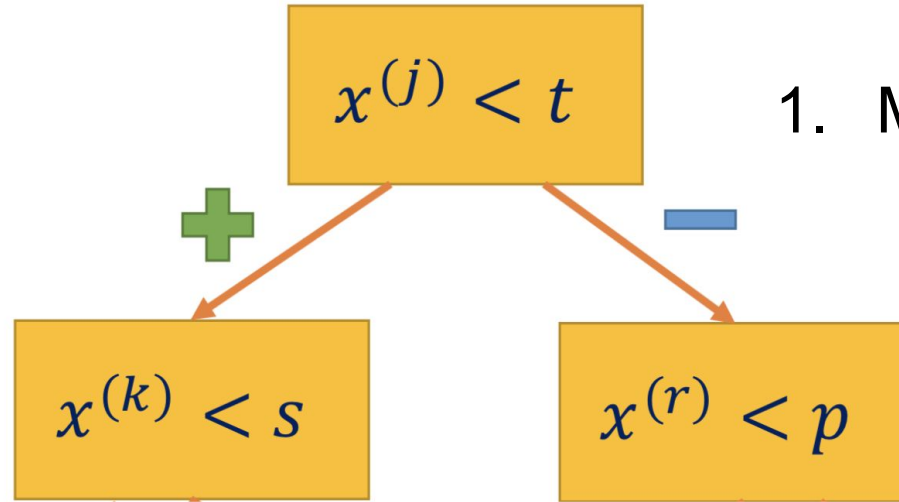
1. Make a split

Constructing decision trees



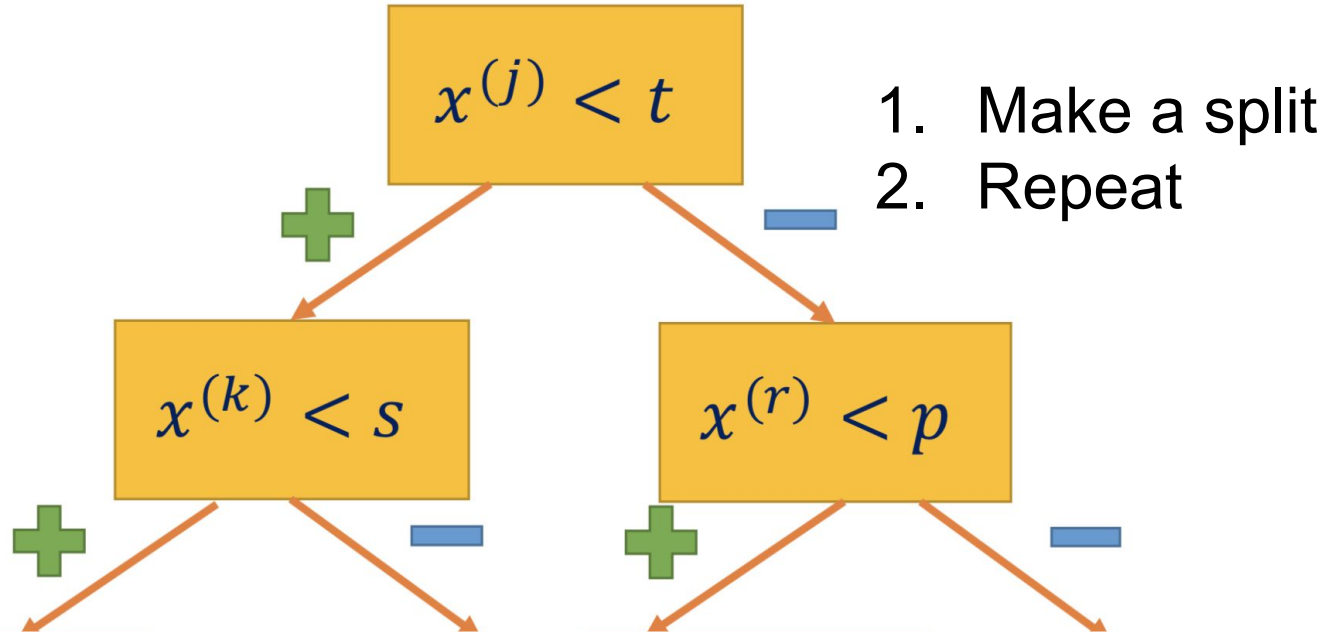
1. Make a split

Constructing decision trees

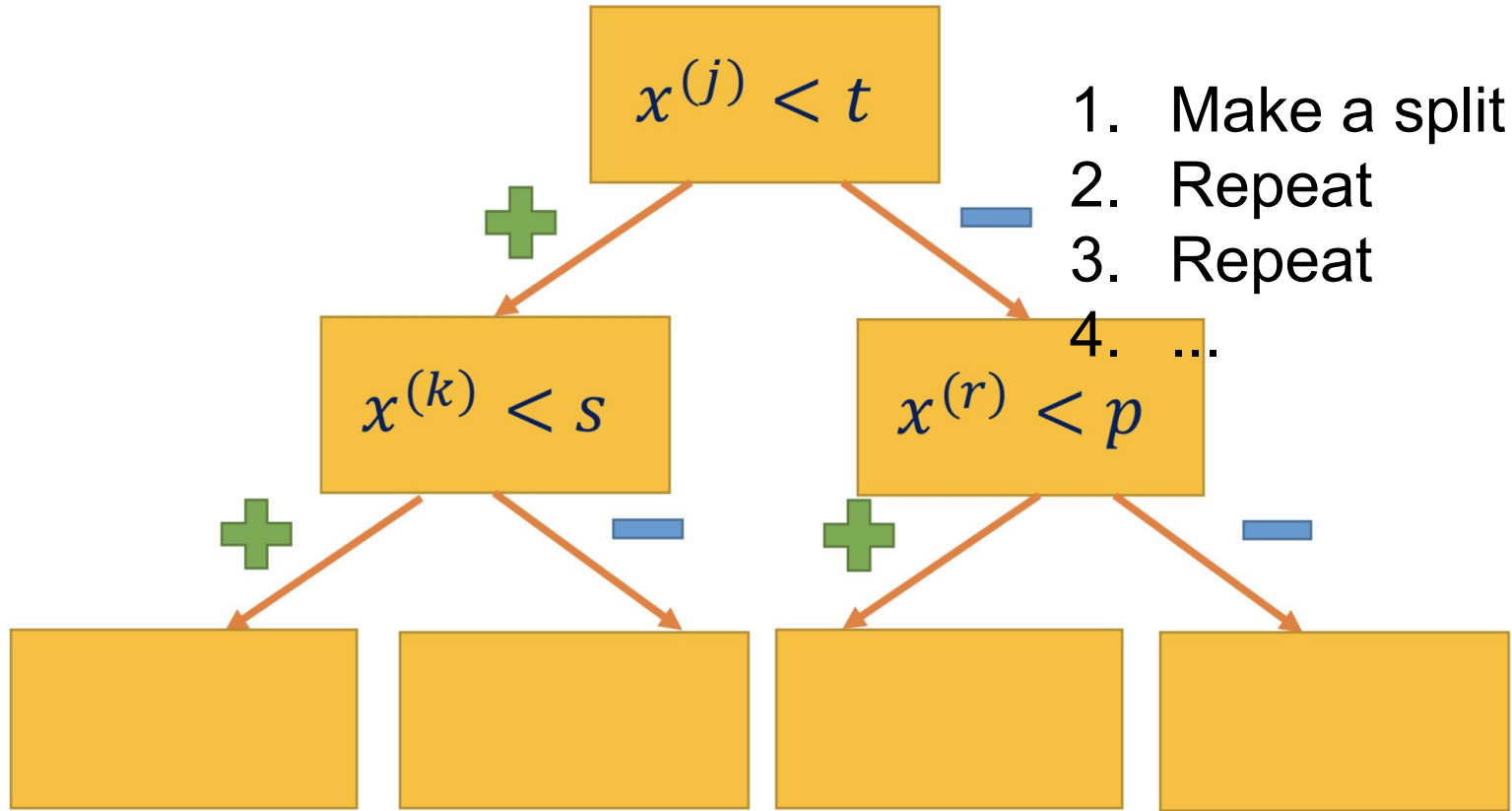


1. Make a split

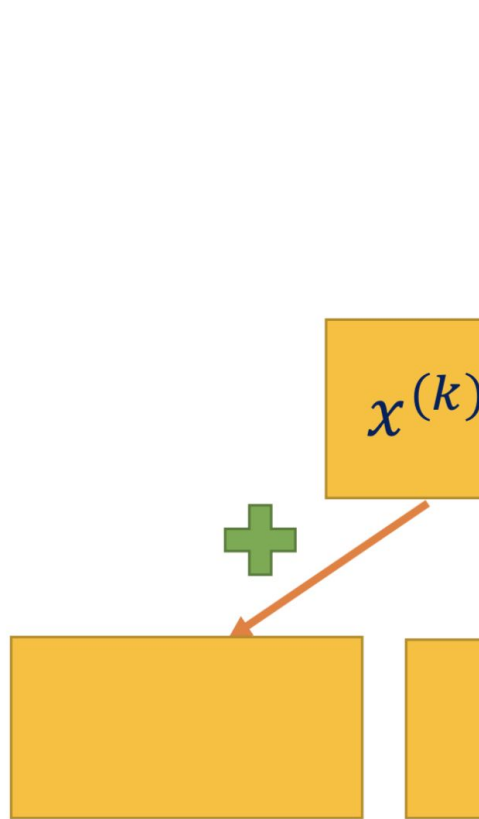
Constructing decision trees



Constructing decision trees

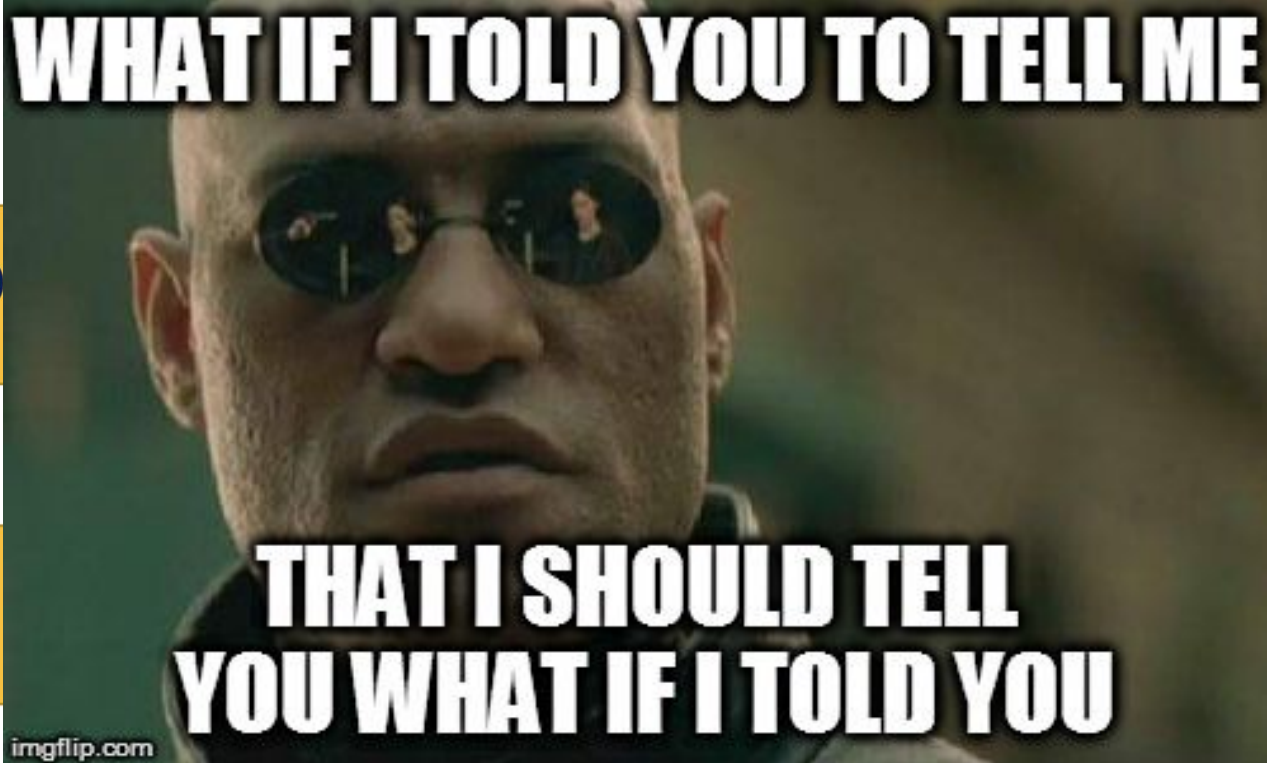


Constructing decision trees



$$x^{(j)} < t$$

1. Make a split



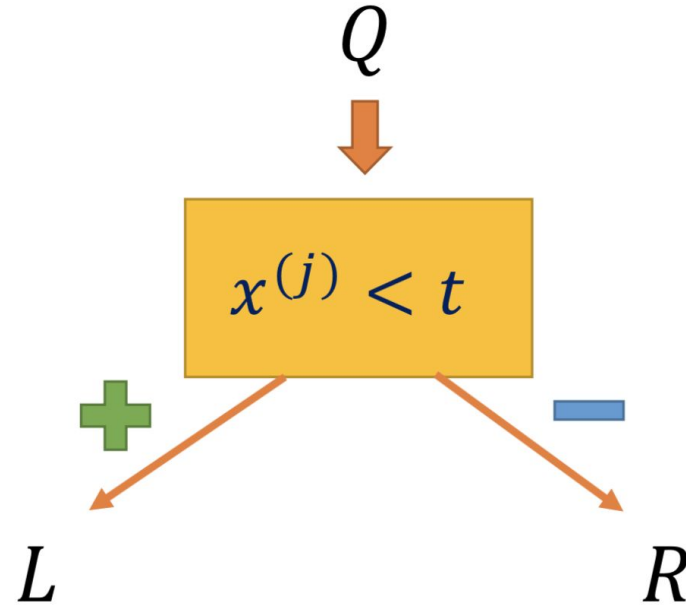
How to split data properly?

Q

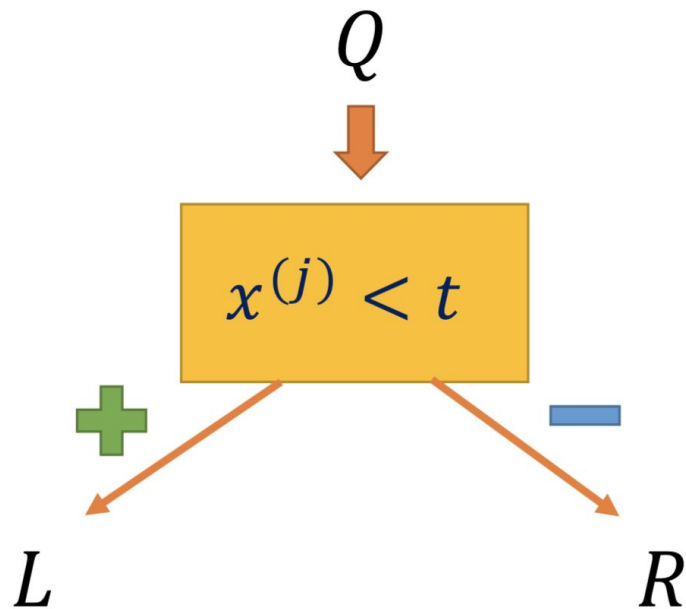


$$x^{(j)} < t$$

How to split data properly?

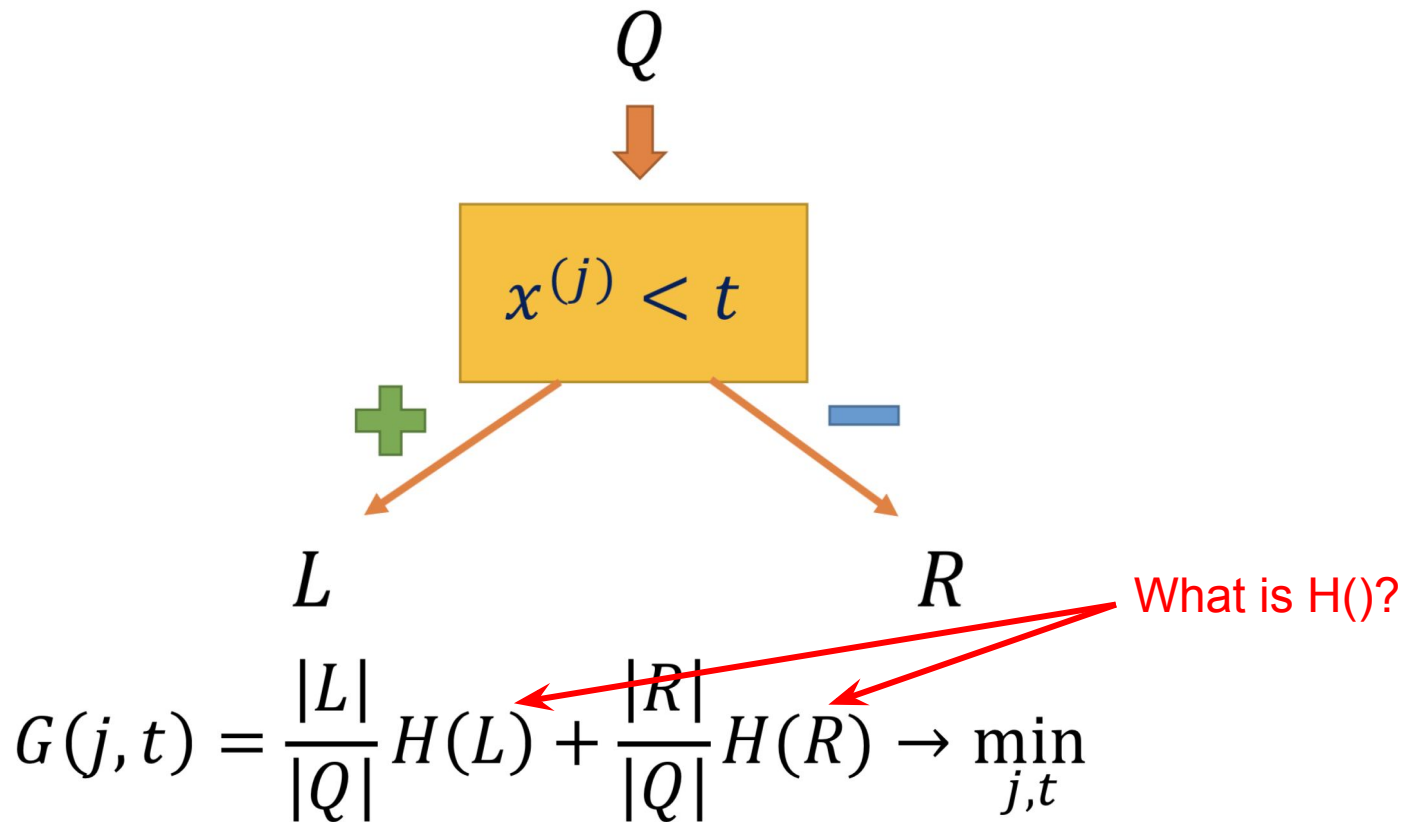


How to split data properly?



$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R)$$

How to split data properly?



Information criteria

$H(R)$ is measure of “heterogeneity” of our data.

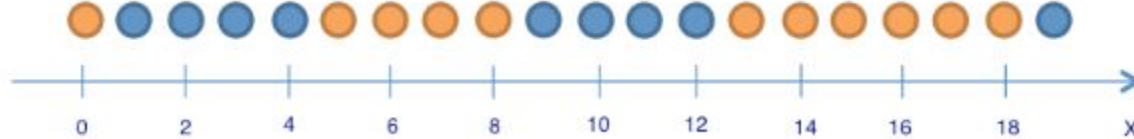
Consider **binary classification** problem:

1. Misclassification criteria: $H(R) = 1 - \max\{p_0, p_1\}$

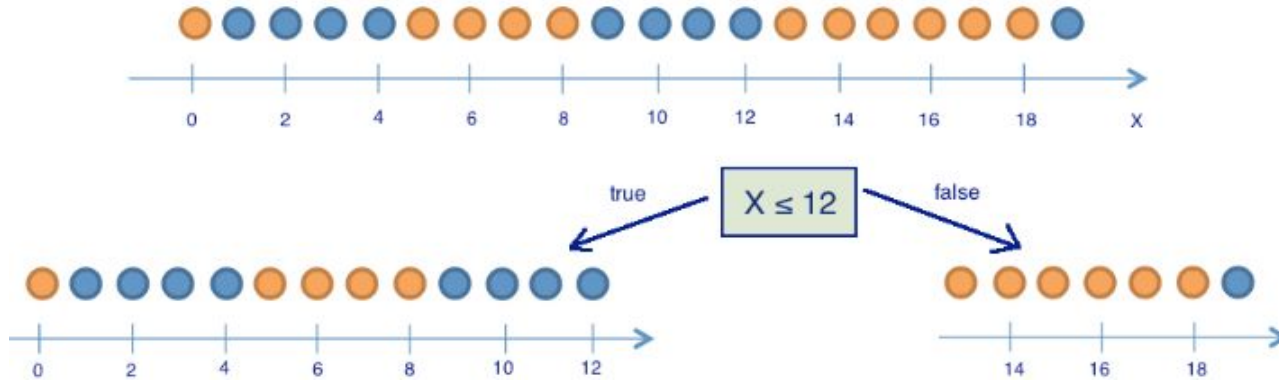
2. Entropy criteria: $H(R) = -p_0 \log_2 p_0 - p_1 \log_2 p_1$

3. Gini impurity: $H(R) = 1 - p_0^2 - p_1^2 = 1 - 2p_0p_1$

$H(R)$ is measure of “heterogeneity” of our data.
Consider binary classification problem:



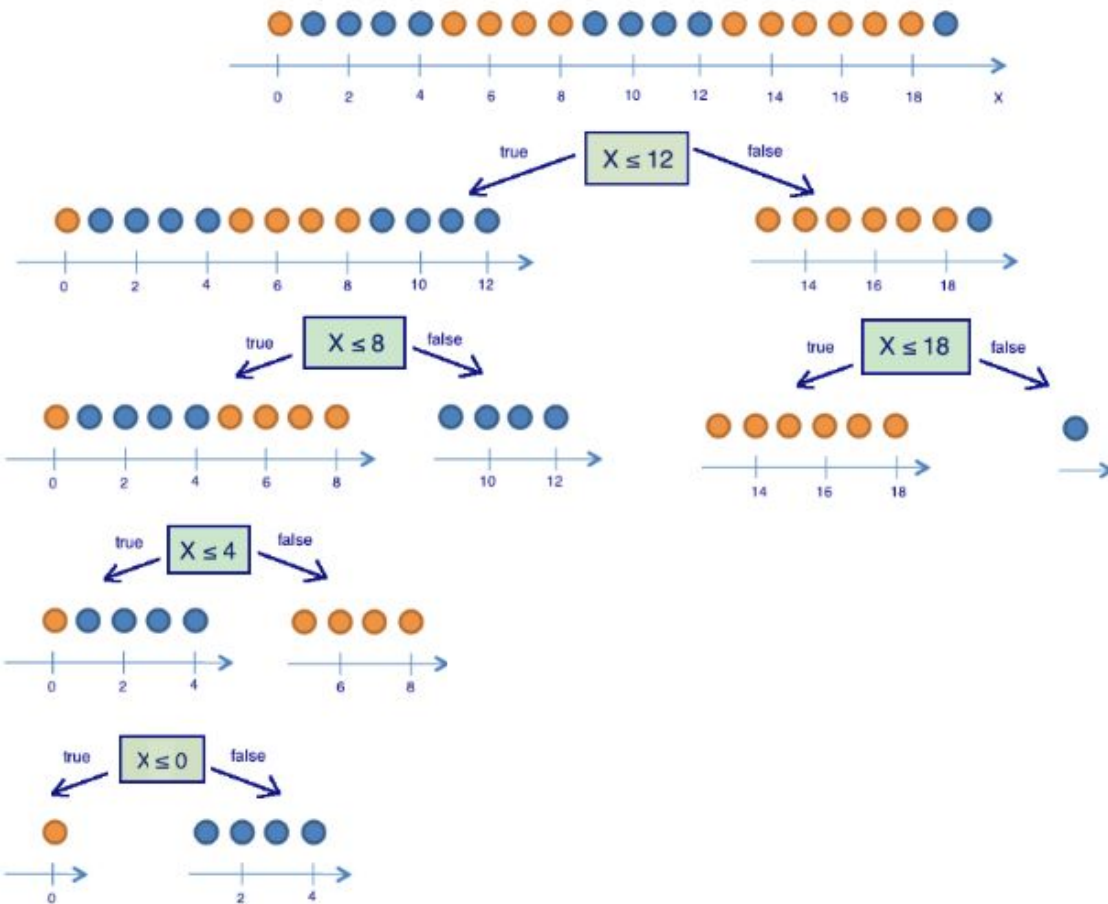
$H(R)$ is measure of “heterogeneity” of our data.
Consider binary classification problem:



Information criteria: Entropy

$$S = - \sum_k p_k \log_2 p_k$$

In binary case $N = 2$



$$S = -p_+ \log_2 p_+ - p_- \log_2 p_- = -p_+ \log_2 p_+ - (1 - p_+) \log_2 (1 - p_+)$$

Information criteria: Gini impurity

$$G = 1 - \sum_k (p_k)^2$$

In binary case $N = 2$

$$G = 1 - p_+^2 - p_-^2 = 1 - p_+^2 - (1 - p_+)^2 = 2p_+(1 - p_+)$$

$H(R)$ is measure of “heterogeneity” of our data.

Consider **multiclass classification** problem:

1. Misclassification criteria:
$$H(R) = 1 - \max_k \{p_k\}$$

2. Entropy criteria:
$$H(R) = - \sum_k p_k \log_2 p_k$$

3. Gini impurity:
$$H(R) = 1 - \sum_k (p_k)^2$$

$H(R)$ is measure of “heterogeneity” of our data.

Consider **regression** problem:

1. Mean squared error

$$H(R) = \min_c \frac{1}{|R|} \sum_{(x_i, y_i) \in R} (y_i - c)^2$$

What is the constant?

$$c^* = \frac{1}{|R|} \sum_{y_i \in R} y_i$$

Time for your questions and a coffee
break

Regularization for decision trees

Decision Tree can build a very complex dependency

In fact, it can perfectly fit the training data

Trivia: how can we construct such a tree?

Regularization for decision trees

Decision Tree can build a very complex dependency

In fact, it can perfectly fit the training data

Just create a very deep tree with one object per leaf.

Training error = 0

Thus, decision trees are very prone to overfitting!

Regularization for decision trees

General idea: let's make the decision rule more simple and smooth

Two main approaches:

- **Stopping criteria**
 - Create new rules until stopping criterion is satisfied
 - The criterion controls the complexity
- **Pruning**
 - Simplify constructed tree

Regularization for decision trees

Stopping criteria

- Maximum tree depth

Regularization for decision trees

Stopping criteria

- Maximum tree depth
- Minimal number of objects inside a leaf

Regularization for decision trees

Stopping criteria

- Maximum tree depth
- Minimal number of objects inside a leaf
- Maximum number of leaves

Regularization for decision trees

Stopping criteria

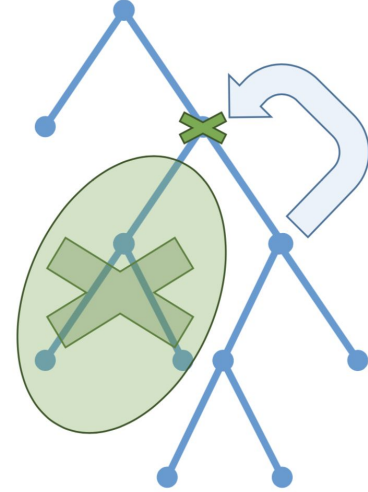
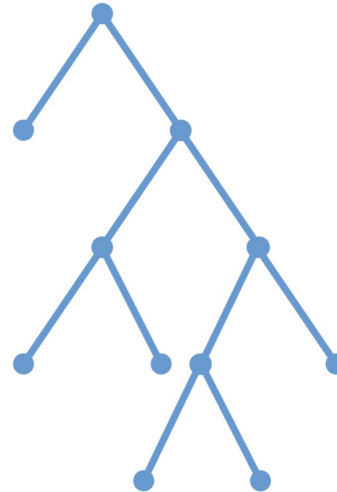
- Maximum tree depth
- Minimal number of objects inside a leaf
- Maximum number of leaves
- Stop if all the objects in the leaf share the same class

Regularization for decision trees

Stopping criteria

- Maximum tree depth
- Minimal number of objects inside a leaf
- Maximum number of leaves
- Stop if all the objects in the leaf share the same class
- Stop when quality metric stops increasing by $x\%$ after the split

- Post-pruning:
 - Simplify constructed tree.



Compositions of decision trees (ensembles)

Compositions

- Decision trees are prone to overfitting
- At the same time, they capture complex dependencies

Main idea:

- Let's train multiple decision trees that differ in some training properties
- Then their predictions' average will be the final answer

Bootstrap

Consider dataset X containing N objects.

Pick I objects with return from X and repeat in N times to get N datasets.

Error of model trained on X_j : $\varepsilon_j(x) = b_j(x) - y(x), \quad j = 1, \dots, N,$

Then $\mathbb{E}_x(b_j(x) - y(x))^2 = \mathbb{E}_x \varepsilon_j^2(x).$

The mean error of N models: $E_1 = \frac{1}{N} \sum_{j=1}^N \mathbb{E}_x \varepsilon_j^2(x).$

Bootstrap

Consider the errors unbiased and uncorrelated:

$$\mathbb{E}_x \varepsilon_j(x) = 0;$$

$$\mathbb{E}_x \varepsilon_i(x) \varepsilon_j(x) = 0, \quad i \neq j.$$

The final model averages all predictions:

$$a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x).$$

Error decreased by N times!

$$\begin{aligned} E_N &= \mathbb{E}_x \left(\frac{1}{N} \sum_{j=1}^n b_j(x) - y(x) \right)^2 = \\ &= \mathbb{E}_x \left(\frac{1}{N} \sum_{j=1}^N \varepsilon_j(x) \right)^2 = \\ &= \frac{1}{N^2} \mathbb{E}_x \left(\sum_{j=1}^N \varepsilon_j^2(x) + \underbrace{\sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x)}_{=0} \right) = \\ &= \frac{1}{N} E_1. \end{aligned}$$

Bootstrap

Consider the errors ~~unbiased and uncorrelated~~:

This is a lie

$$\mathbb{E}_x \varepsilon_j(x) = 0;$$

$$\mathbb{E}_x \varepsilon_i(x) \varepsilon_j(x) = 0, \quad i \neq j.$$

The final model averages all predictions:

$$a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x).$$

Error decreased by N times!

$$\begin{aligned} E_N &= \mathbb{E}_x \left(\frac{1}{N} \sum_{j=1}^n b_j(x) - y(x) \right)^2 = \\ &= \mathbb{E}_x \left(\frac{1}{N} \sum_{j=1}^N \varepsilon_j(x) \right)^2 = \\ &= \frac{1}{N^2} \mathbb{E}_x \left(\sum_{j=1}^N \varepsilon_j^2(x) + \underbrace{\sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x)}_{=0} \right) = \\ &= \frac{1}{N} E_1. \end{aligned}$$

Bagging = Bootstrap aggregating

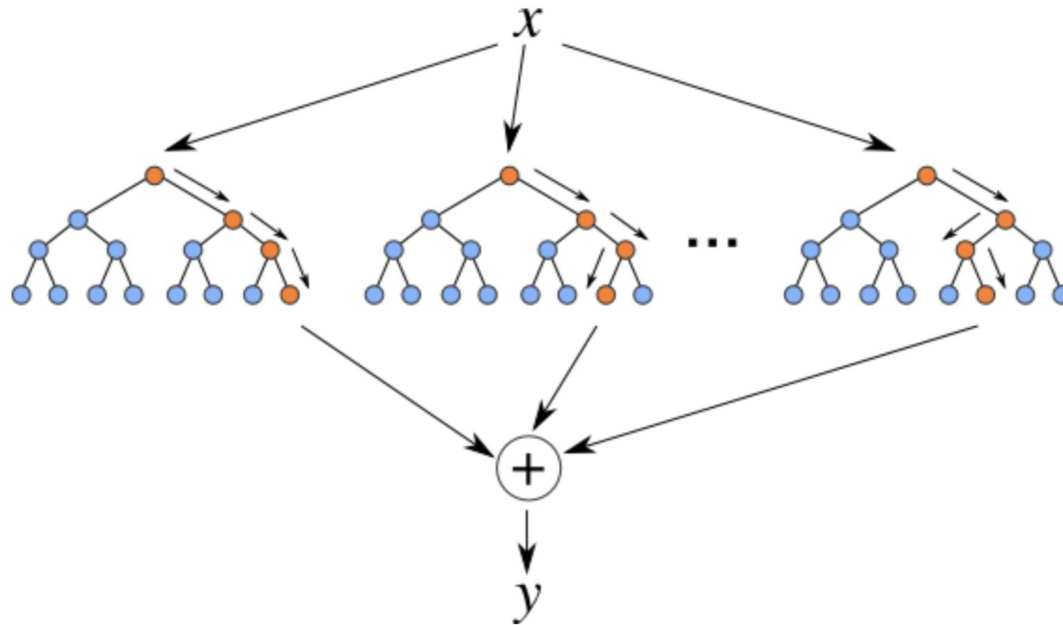
Decreases the error if the basic algorithms are not correlated.

RSM - Random Subspace Method

Same approach, but with features.

Random Forest

Bagging + RSM = Random Forest



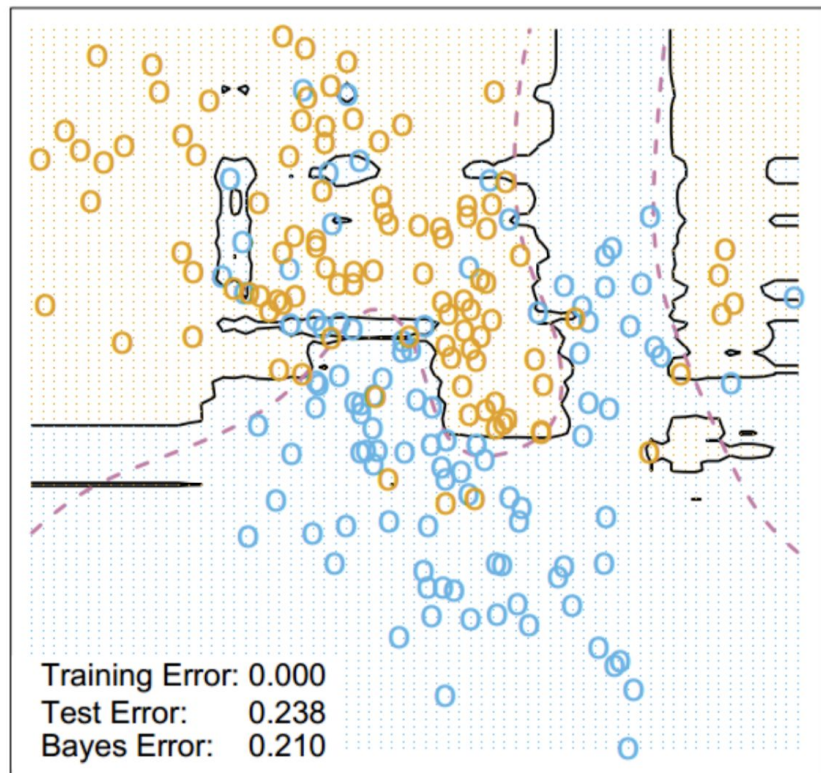
Training procedure:

- Create N bootstrap samples from the data
- For each sample, train a decision tree with RSM
- Average the trees' answers for the ensemble prediction

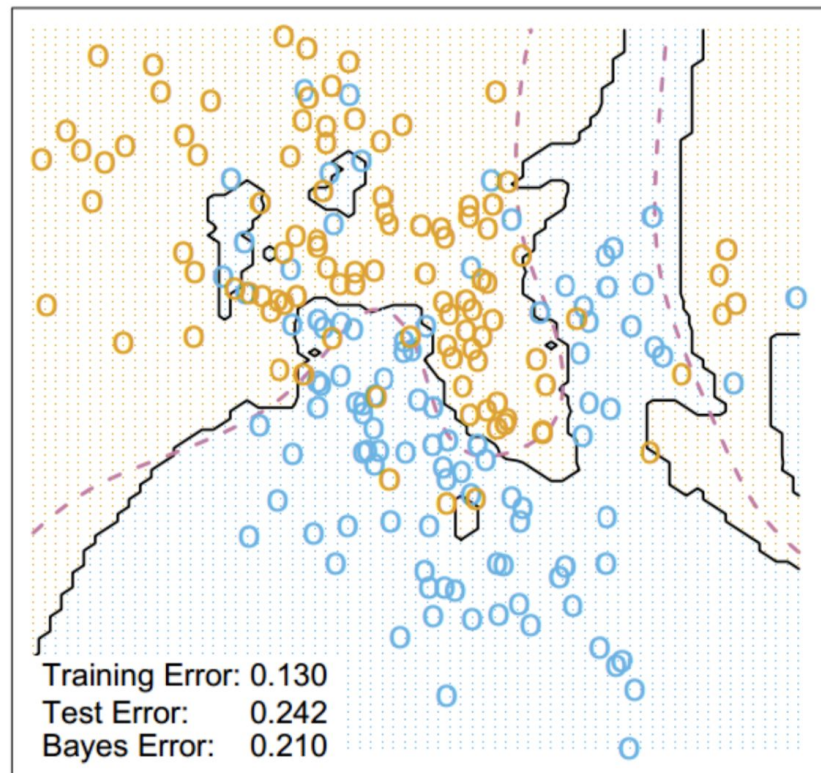
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

$$\text{OOB} = \sum_{i=1}^{\ell} L \left(y_i, \frac{1}{\sum_{n=1}^N [x_i \notin X_n]} \sum_{n=1}^N [x_i \notin X_n] b_n(x_i) \right)$$

Random Forest Classifier



3-Nearest Neighbors



Thanks for your attention