

# Decision Trees

Classification and Regression Trees, impurity functions, solution properties

Machine Learning and Data Mining, 2024

Majid Sohrabi

National Research University Higher School of Economics



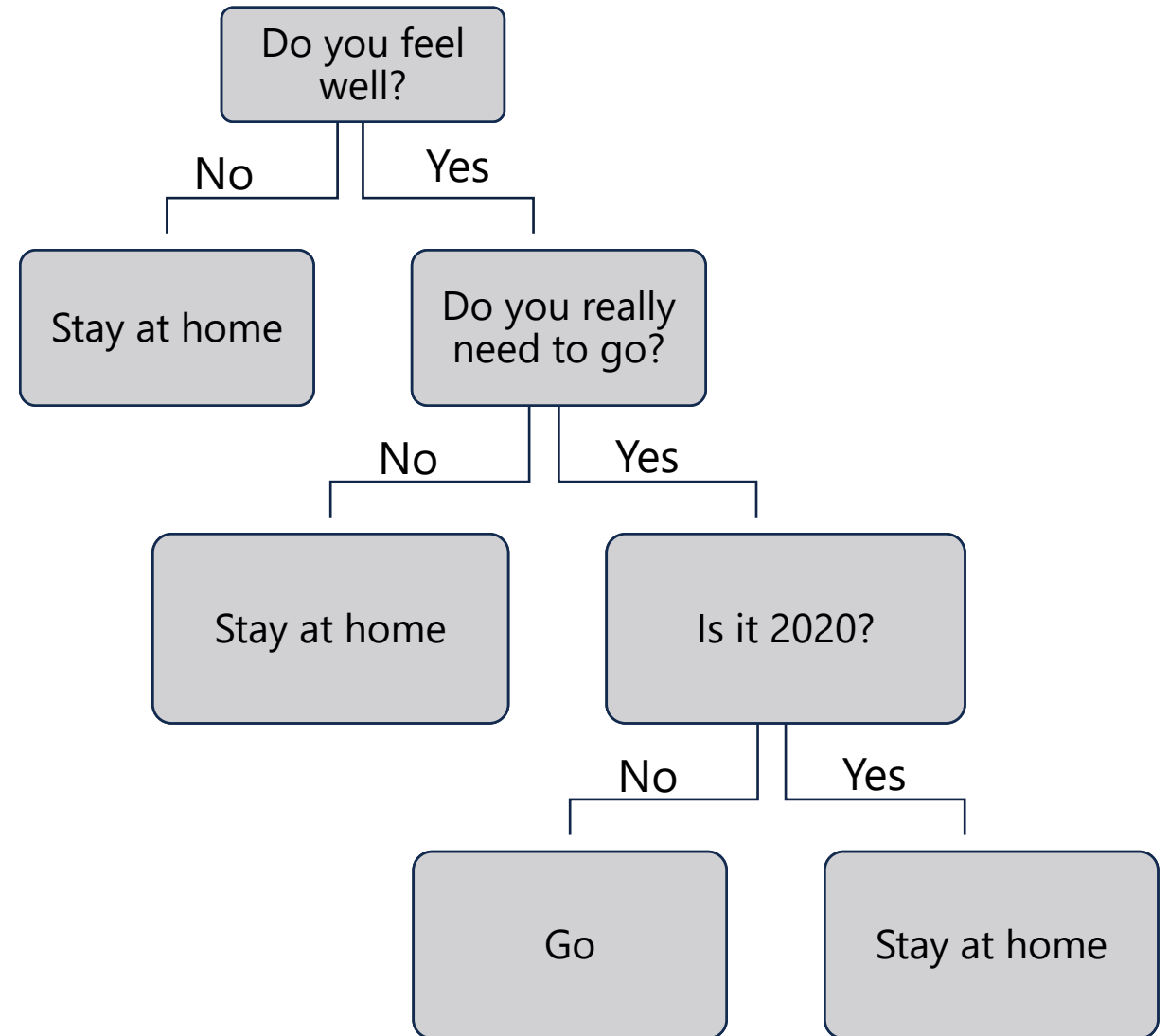
MMCP

November 06, 2024

# Basics

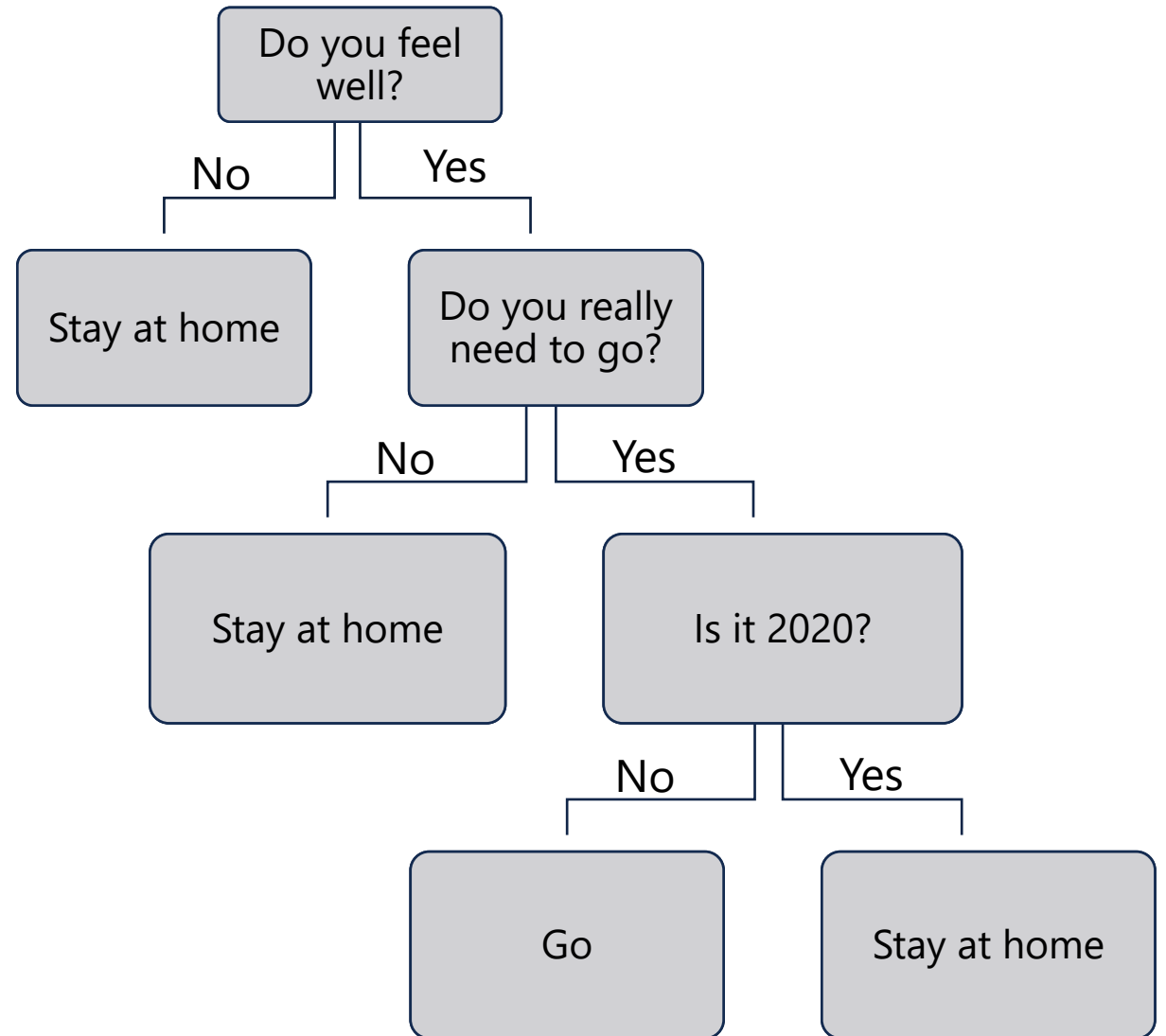


# "Should you go to work?" chart



# "Should you go to work?" chart

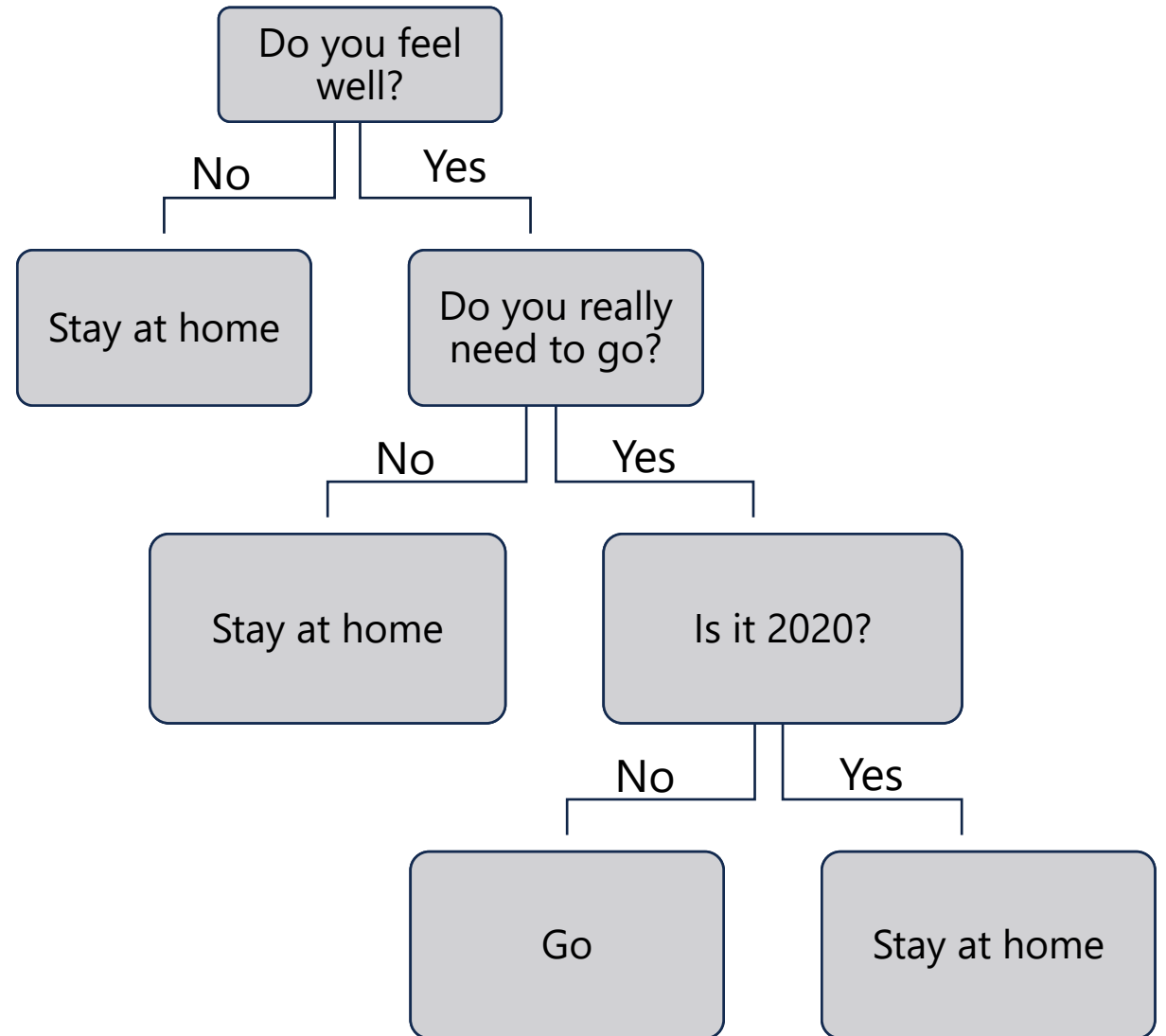
Directed graph



# "Should you go to work?" chart

Directed graph

No loops

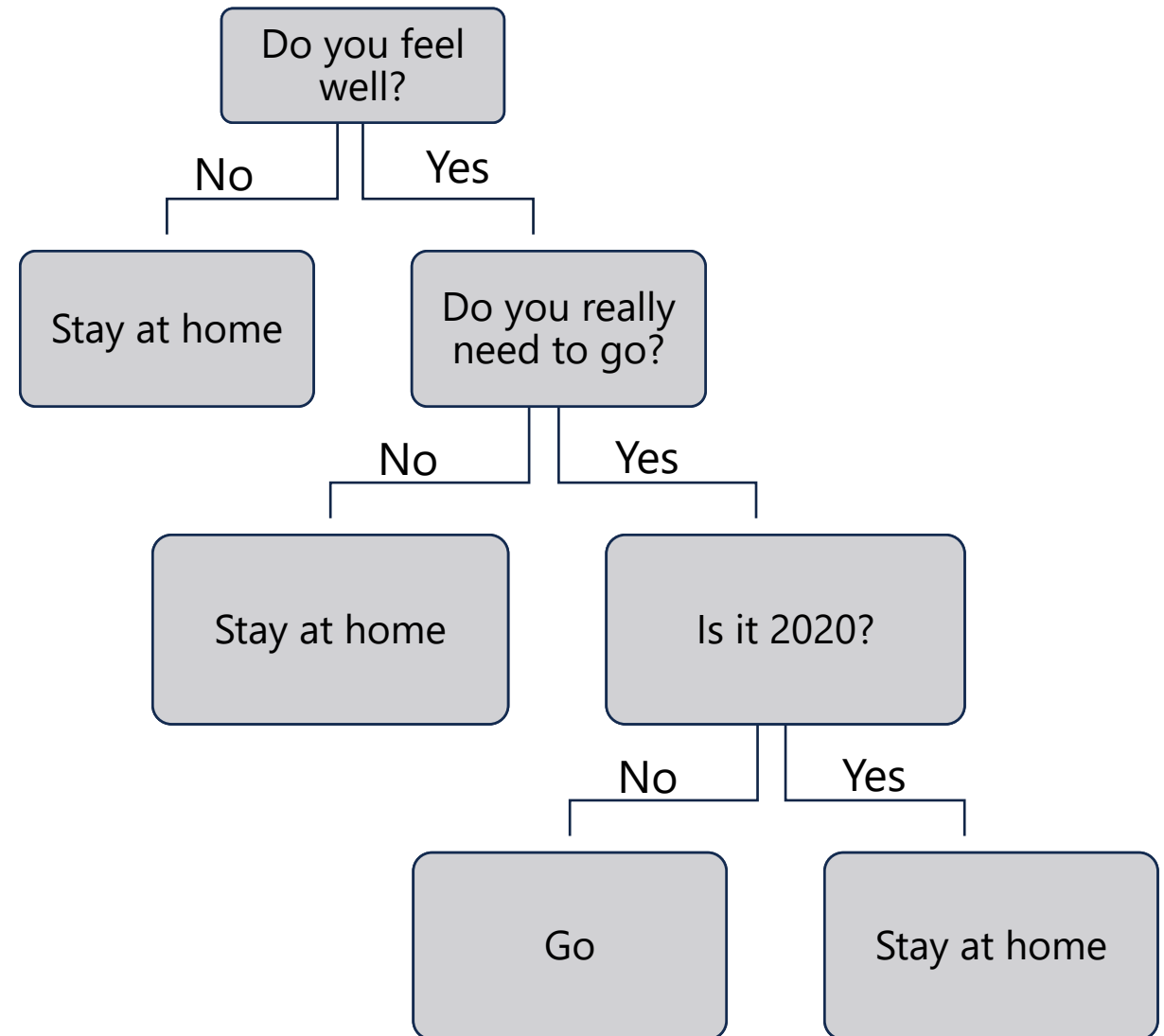


# "Should you go to work?" chart

Directed graph

No loops

Single root node



# "Should you go to work?" chart

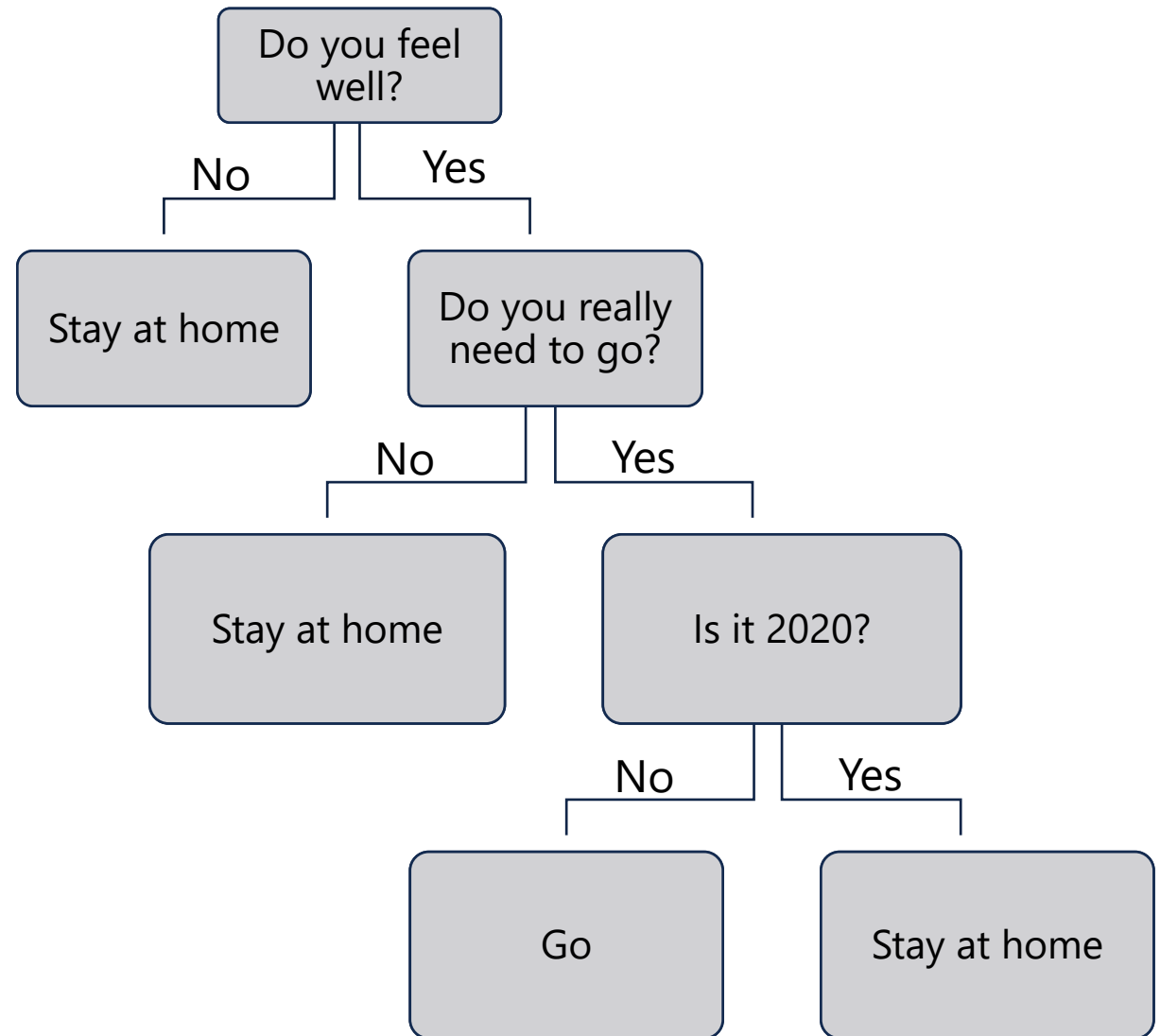
Directed graph

No loops

Single root node

Each node has:

- either 0 child nodes (**terminal node**, "leaf")



# "Should you go to work?" chart

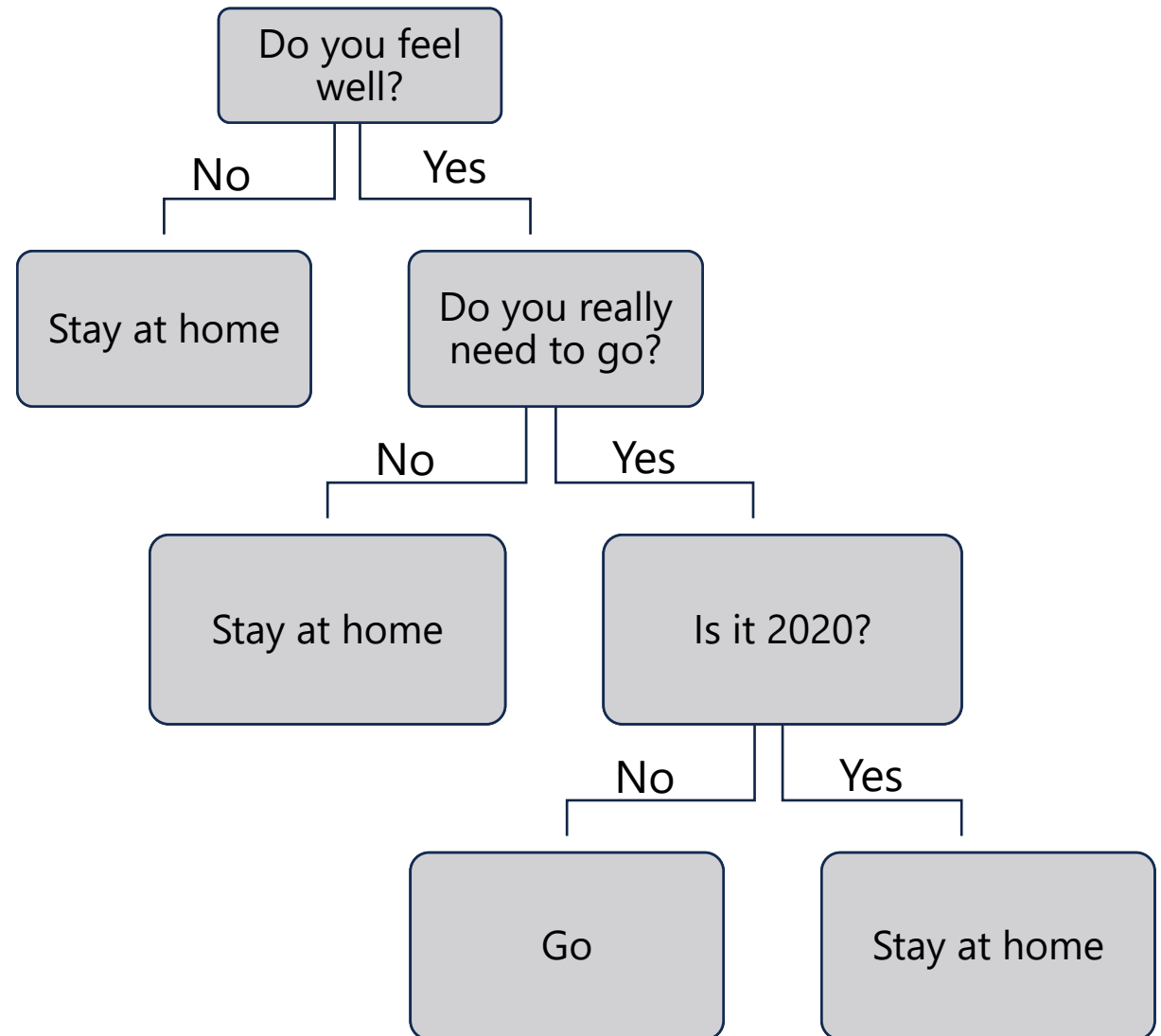
Directed graph

No loops

Single root node

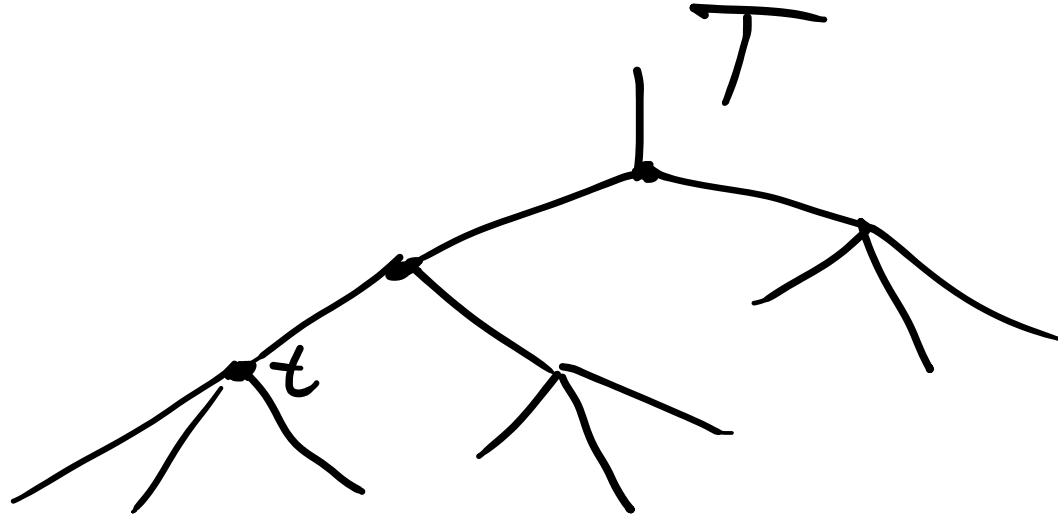
Each node has:

- either 0 child nodes (**terminal node**, "leaf")
- or  $\geq 2$  child nodes (**internal node**)
  - 2 nodes for binary trees

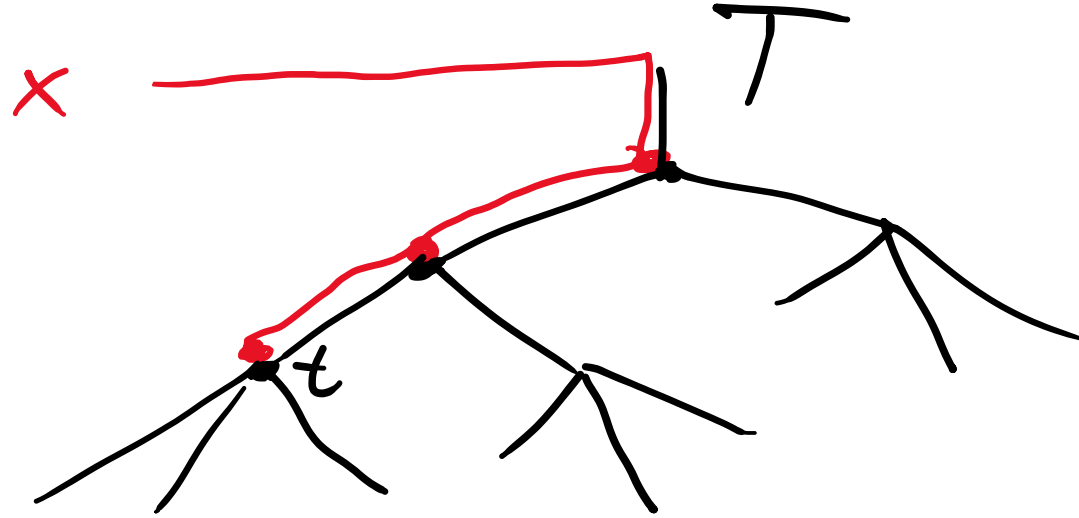




# Defining a tree (general approach)



# Defining a tree (general approach)

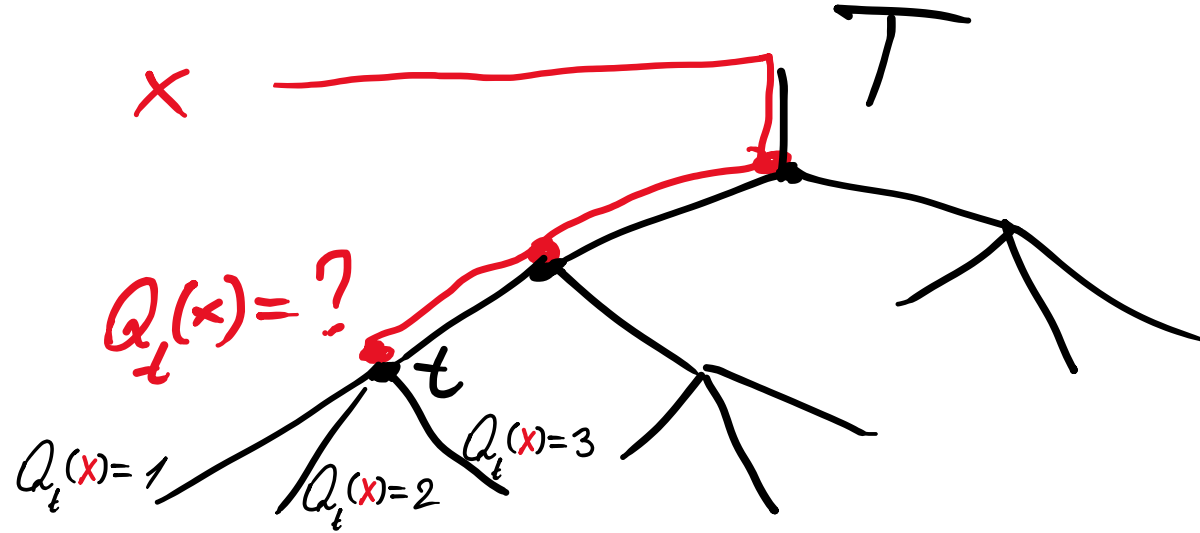


# Defining a tree (general approach)



For each node  $t \in T$  define a check function  $Q_t(x)$

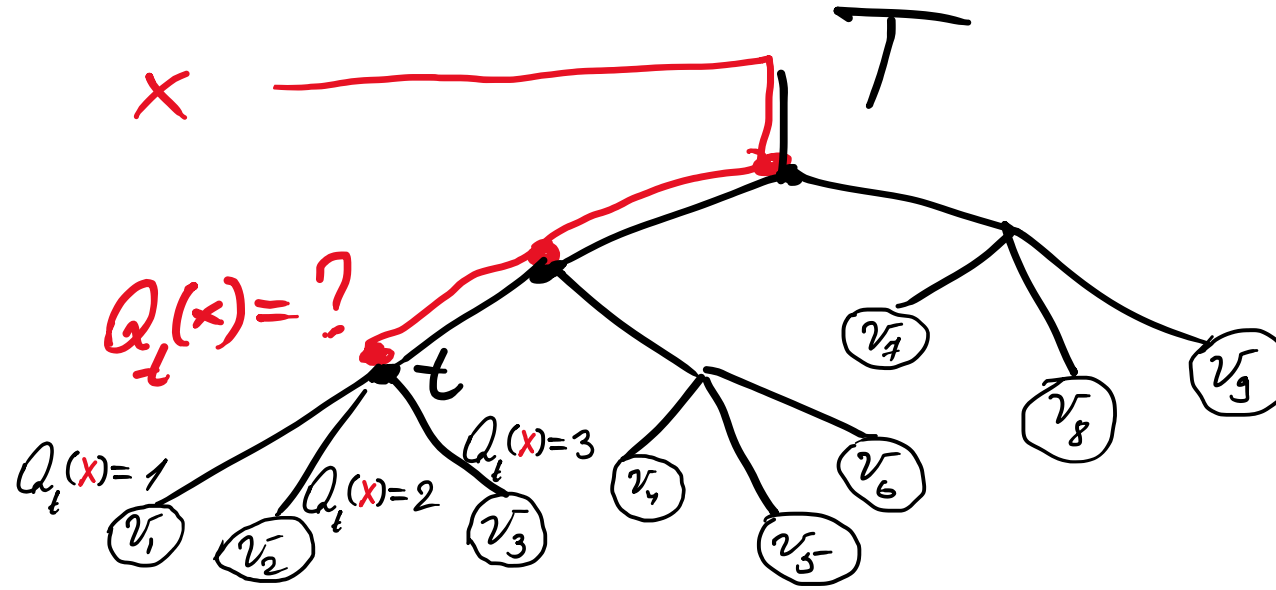
# Defining a tree (general approach)



For each node  $t \in T$  define a check function  $Q_t(x)$

For each child node of  $t$  assign a set of unique values of  $Q_t(x)$

# Defining a tree (general approach)



For each node  $t \in T$  define a check function  $Q_t(x)$

For each child node of  $t$  assign a set of unique values of  $Q_t(x)$

Assign each terminal node  $i$  a prediction value  $v_i$

# Classification and Regression Trees (CART)



# CART

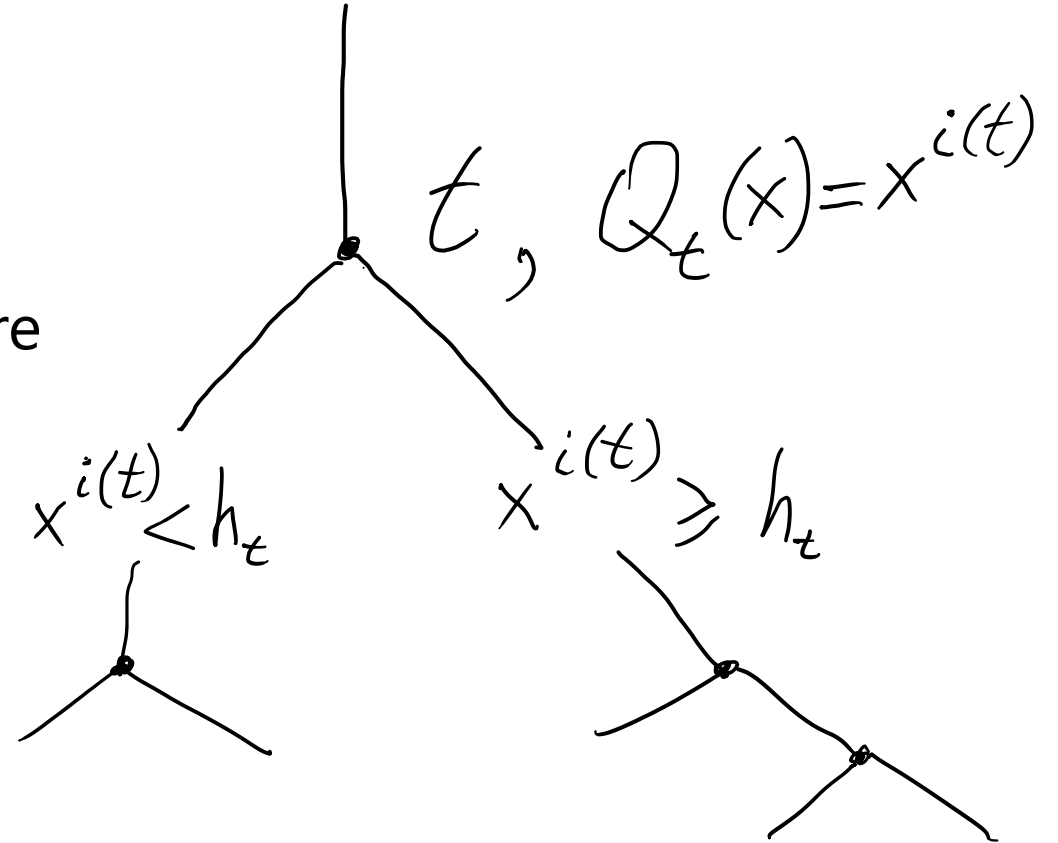
Binary trees

Check function:

$Q_t(x) = x^{i(t)}$  — pick a single ( $i$ -th) feature

Child nodes:

Left or right depending on whether  
 $Q_t(x) \geq h_t$



Finding the best tree is not trivial. In practice a **greedy algorithm** is used.

# Growing a tree

Given a dataset  $D = \{(x_1, y_1), \dots (x_N, y_N)\}$ , and **impurity function**  $I(D)$

Start from a single root node  $t_0$ , all data residing in it:  $D_{t_0} = D$



<https://pixabay.com/users/openclipart-vectors-30363/>



# Growing a tree

Given a dataset  $D = \{(x_1, y_1), \dots (x_N, y_N)\}$ , and **impurity function**  $I(D)$

Start from a single root node  $t_0$ , all data residing in it:  $D_{t_0} = D$

While stop condition not met, repeat for each leaf node  $t$ :



# Growing a tree

Given a dataset  $D = \{(x_1, y_1), \dots (x_N, y_N)\}$ , and **impurity function**  $I(D)$

Start from a single root node  $t_0$ , all data residing in it:  $D_{t_0} = D$

While stop condition not met, repeat for each leaf node  $t$ :

Find feature  $i$  and element  $(x_k, y_k) \in D_t$ , such that for the two subsets

$$D_{t\text{left}} = \{(x, y) \mid (x, y) \in D_t, x^i < x_k^i\},$$

$$D_{t\text{right}} = \{(x, y) \mid (x, y) \in D_t, x^i \geq x_k^i\}$$

the decrease of impurity:

$$|D_t| \cdot \Delta I_t = |D_t| \cdot I(D_t) - \left( |D_{t\text{right}}| \cdot I(D_{t\text{right}}) + |D_{t\text{left}}| \cdot I(D_{t\text{left}}) \right) > 0$$

is maximized (over  $k$  and  $i$ ).



# Growing a tree

Given a dataset  $D = \{(x_1, y_1), \dots (x_N, y_N)\}$ , and **impurity function**  $I(D)$

Start from a single root node  $t_0$ , all data residing in it:  $D_{t_0} = D$

While stop condition not met, repeat for each leaf node  $t$ :

Find feature  $i$  and element  $(x_k, y_k) \in D_t$ , such that for the two subsets

$$D_{t^{\text{left}}} = \{(x, y) \mid (x, y) \in D_t, x^i < x_k^i\},$$

$$D_{t^{\text{right}}} = \{(x, y) \mid (x, y) \in D_t, x^i \geq x_k^i\}$$

the decrease of impurity:

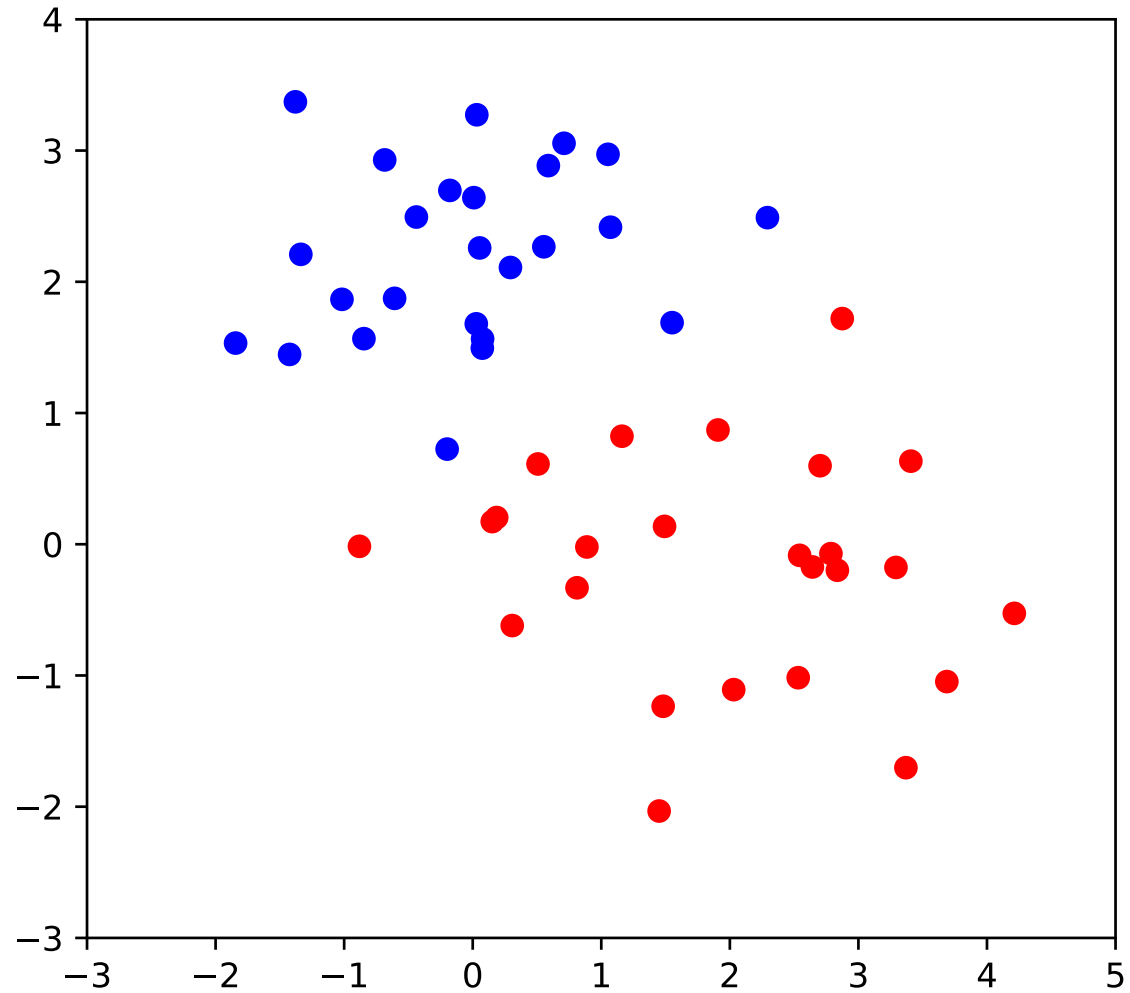
$$|D_t| \cdot \Delta I_t = |D_t| \cdot I(D_t) - \left( |D_{t^{\text{right}}}| \cdot I(D_{t^{\text{right}}}) + |D_{t^{\text{left}}}| \cdot I(D_{t^{\text{left}}}) \right) > 0$$

is maximized (over  $k$  and  $i$ ).

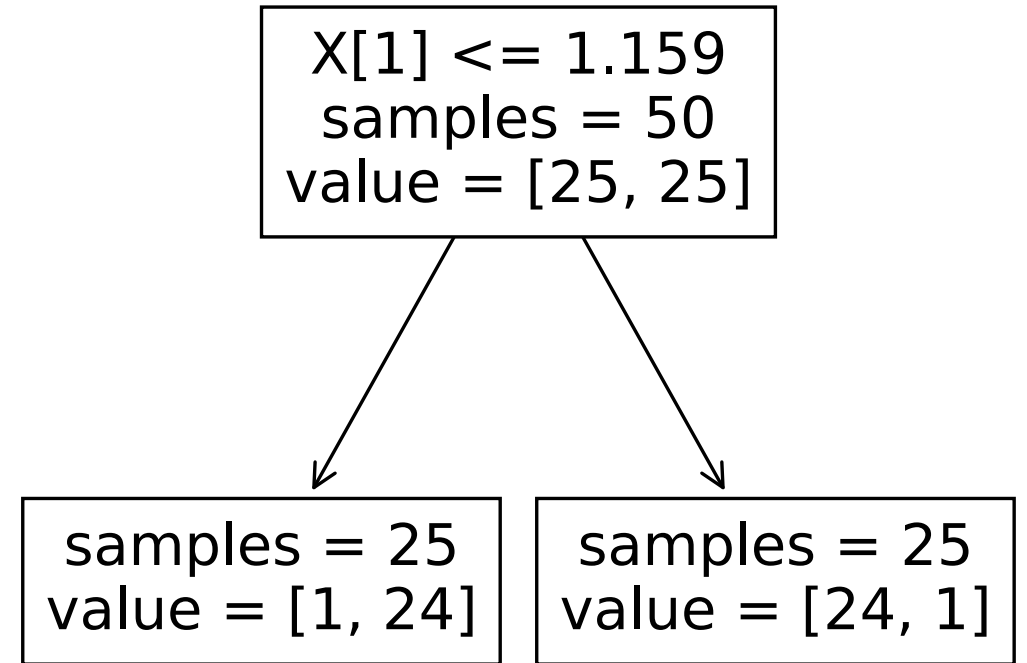
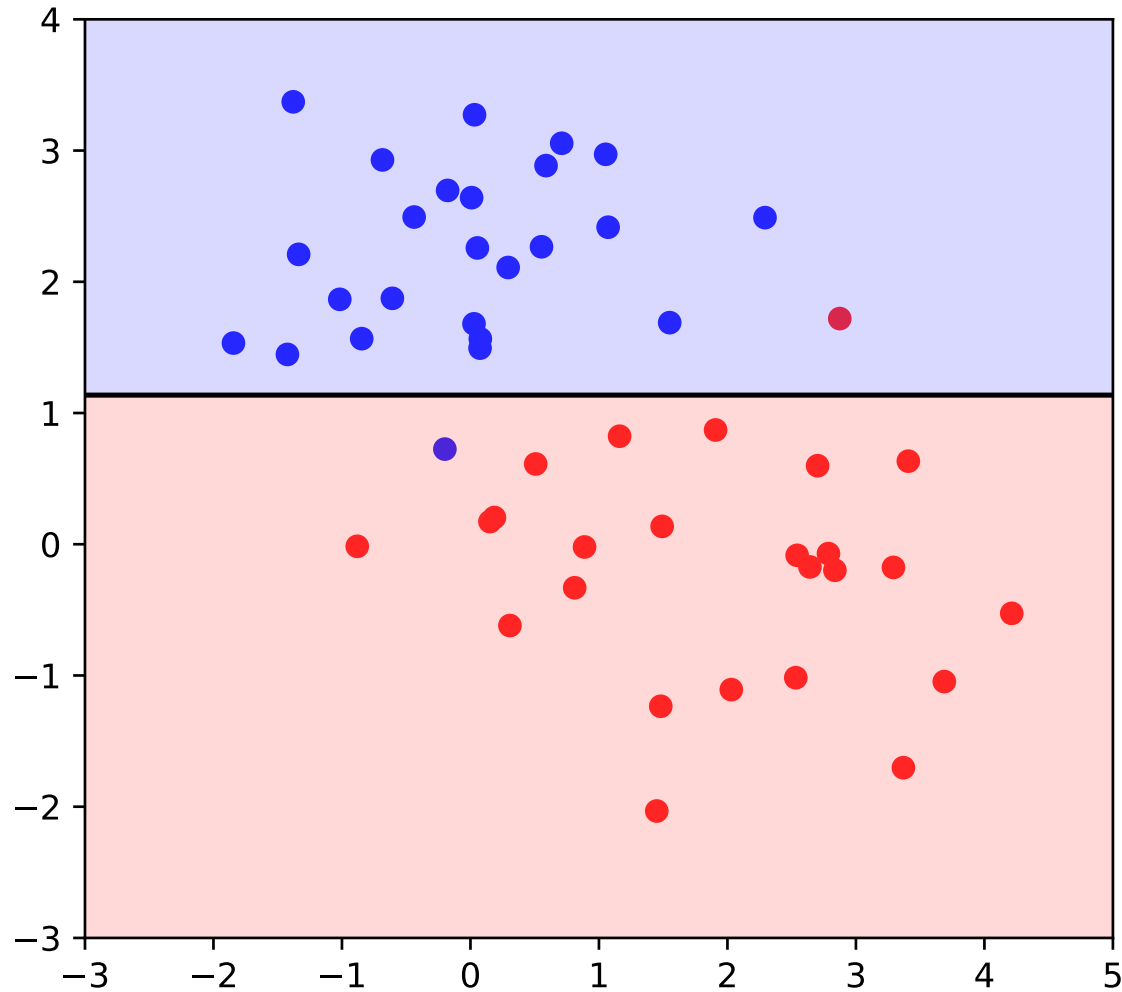
Set the check function  $Q_t(x) = x^i$ , and threshold  $h_t = x_k^i$ ,  
attach the two new corresponding child nodes  $t^{\text{left}}$  and  $t^{\text{right}}$  to  $t$ .



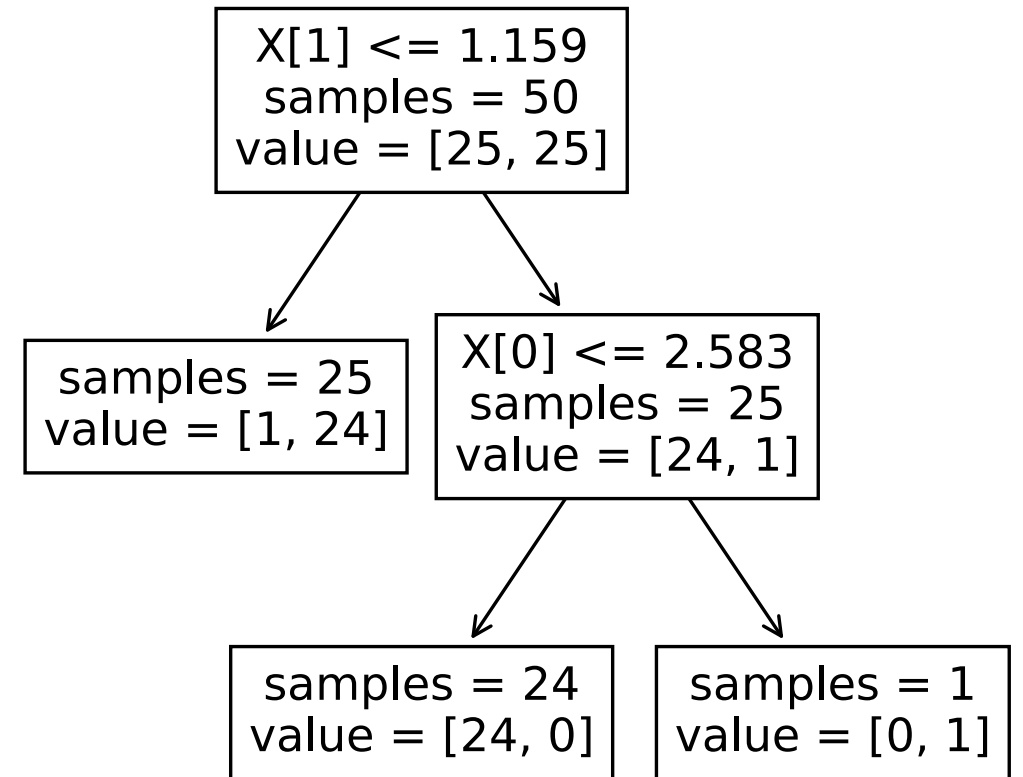
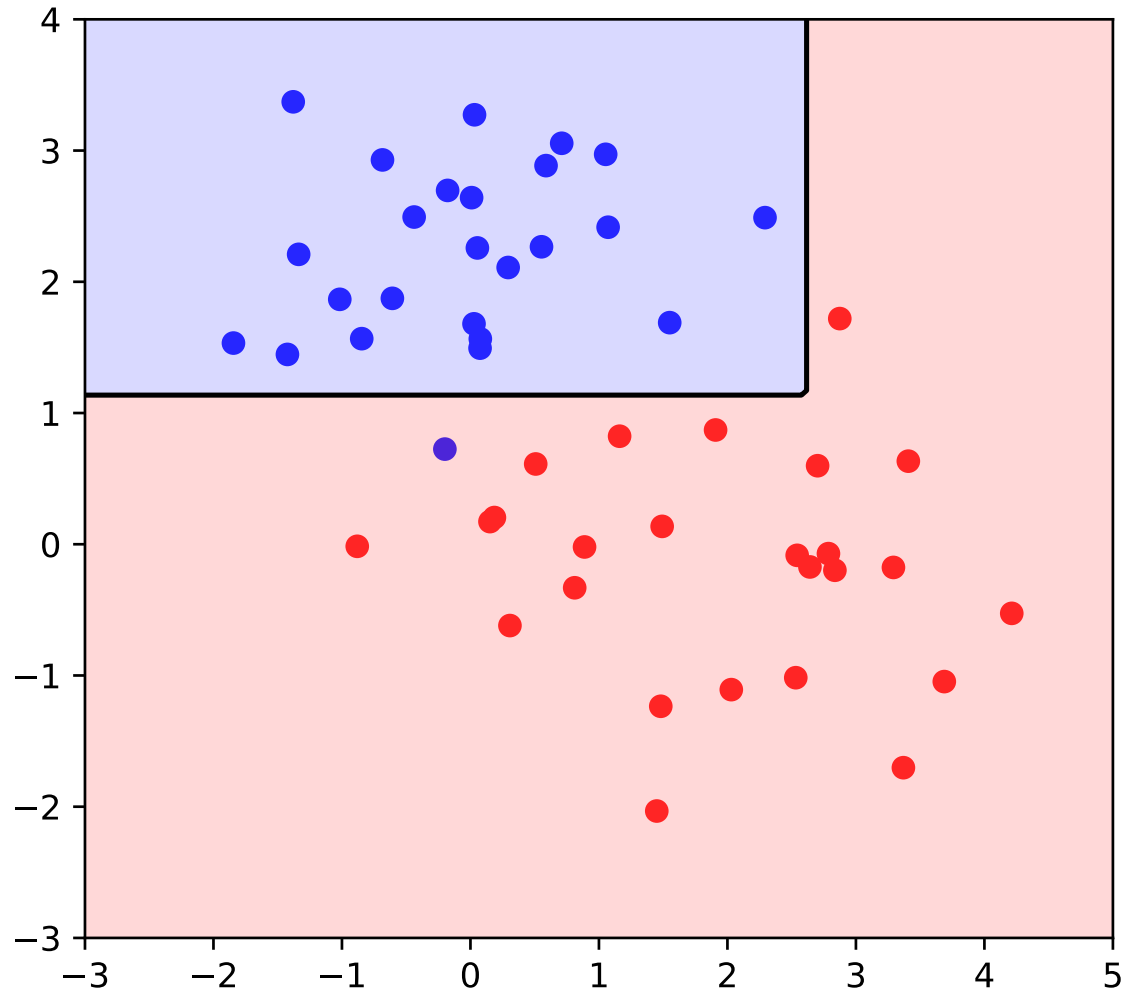
# Growing a tree



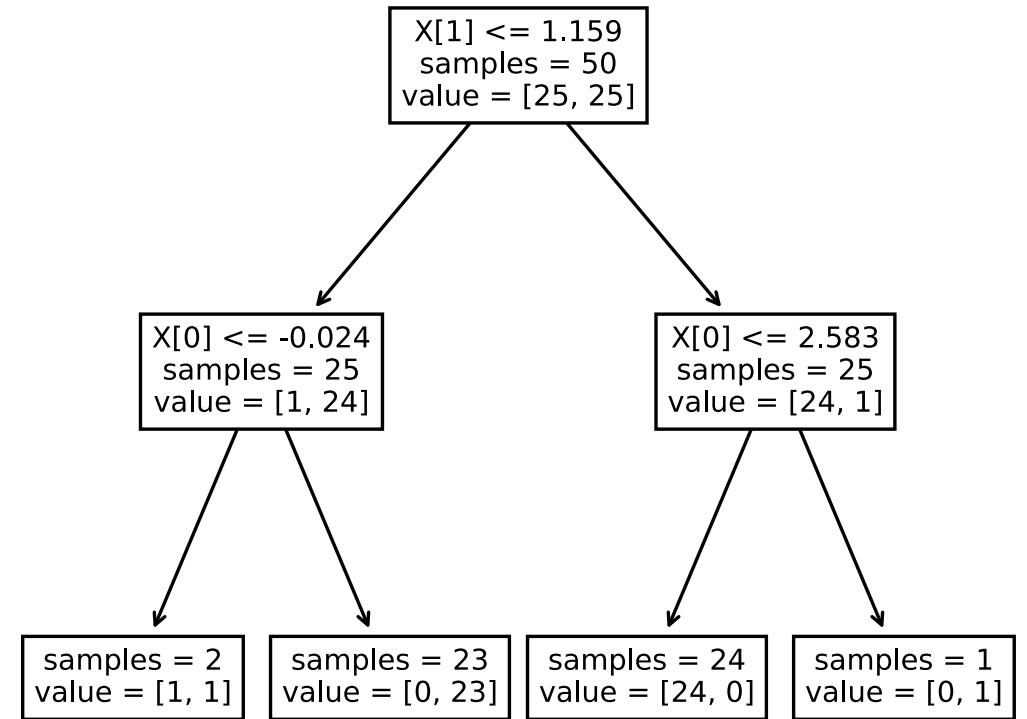
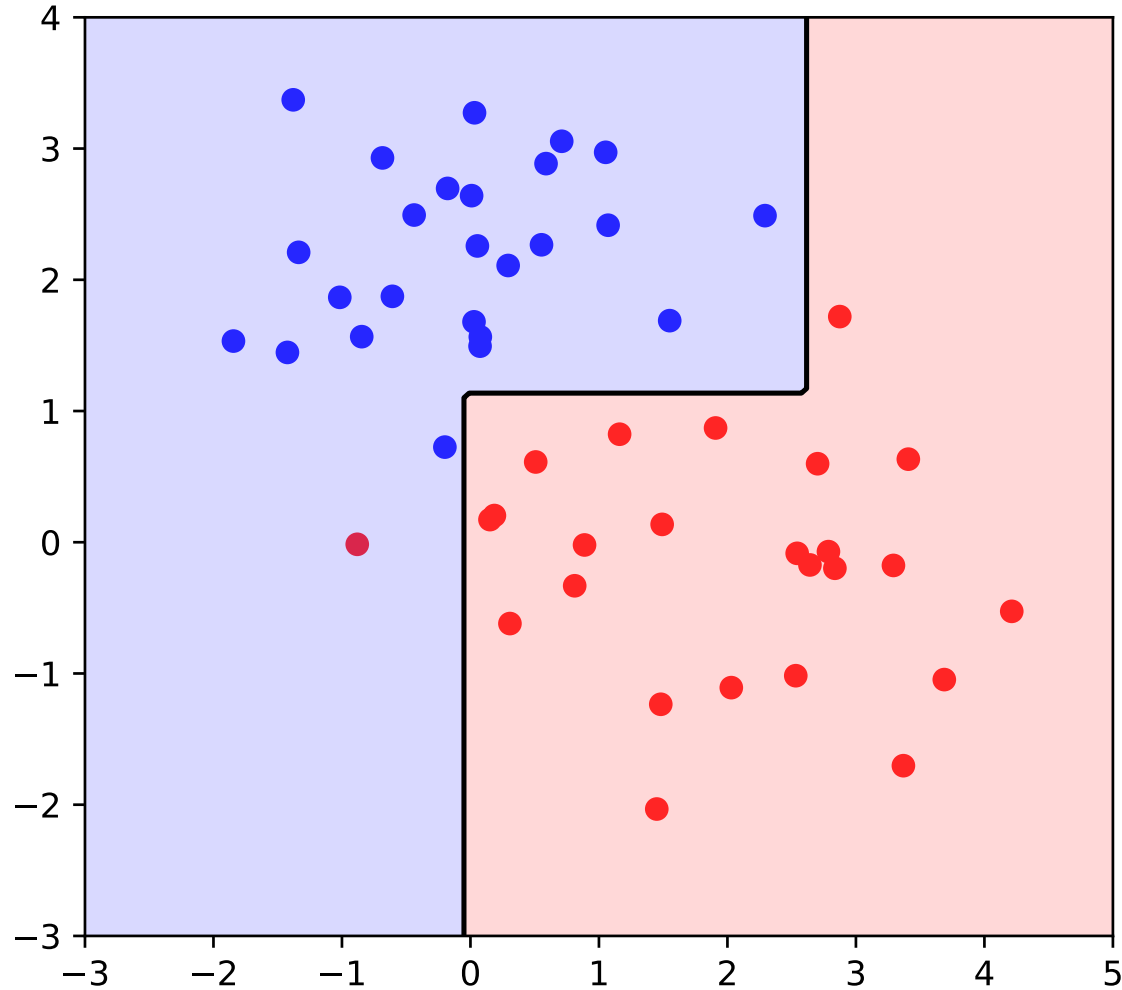
# Growing a tree



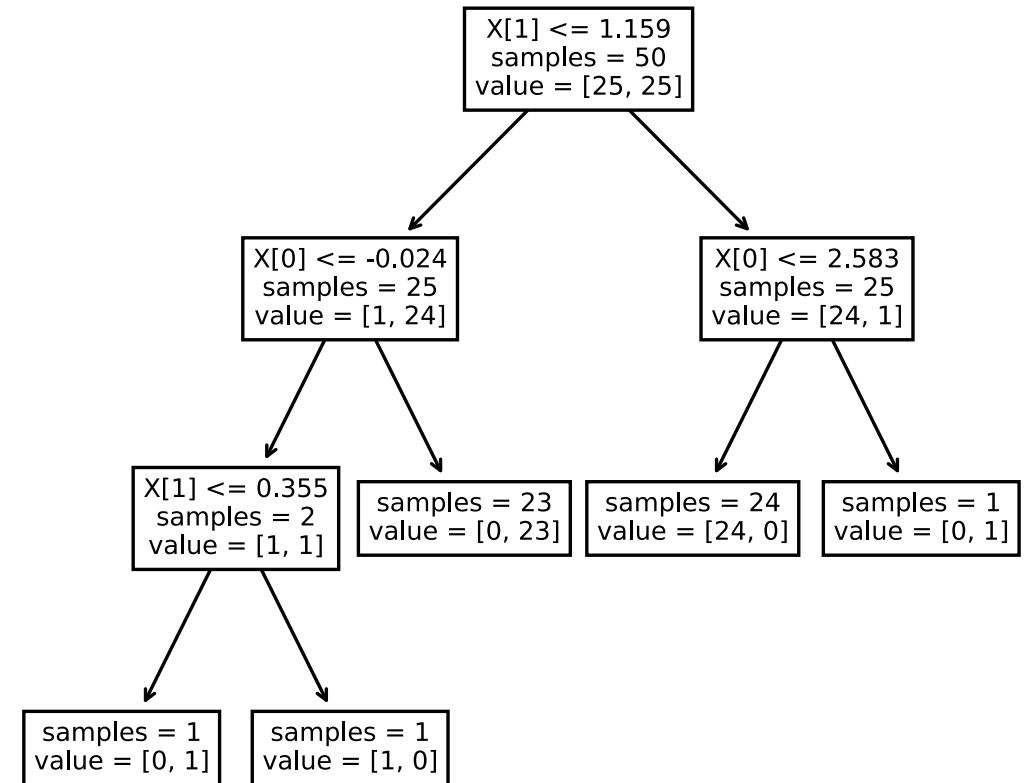
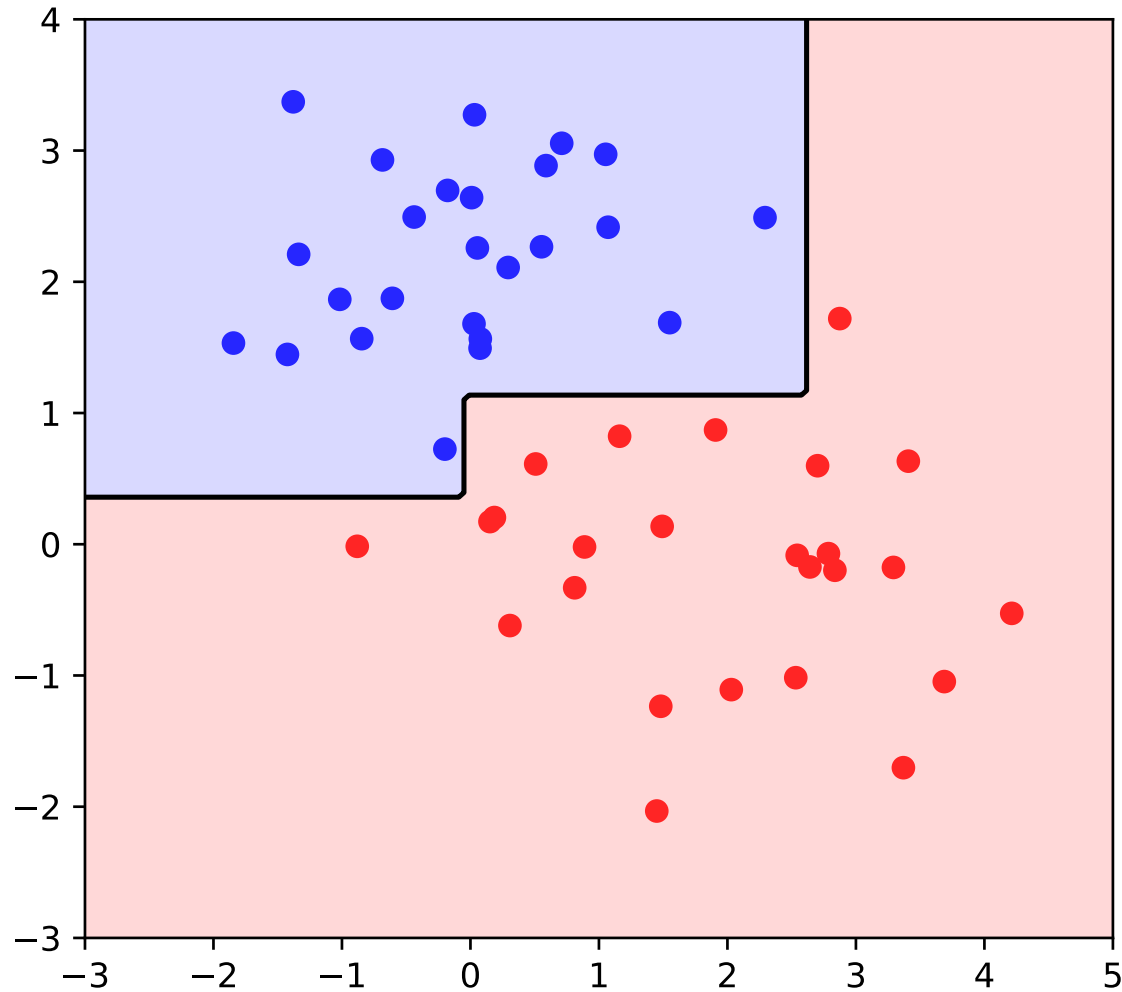
# Growing a tree



# Growing a tree



# Growing a tree





# Impurity measures

## Regression

MSE:

$$I(D_t) = \frac{1}{|D_t|} \sum_{(x,y) \in D_t} (y - \mu_{D_t})^2$$

mean target



MAE:

$$I(D_t) = \frac{1}{|D_t|} \sum_{(x,y) \in D_t} |y - m_{D_t}|$$

median target



# Impurity measures

## What about classification?

Define class probabilities:

$$p_j = \frac{1}{|D_t|} \sum_{(x,y) \in D_t} \mathbb{I}[y = j]$$

Then, impurity function  $\phi(D_t) = \phi(p_1, \dots, p_C)$  should satisfy:

# Impurity measures

## What about classification?

Define class probabilities:

$$p_j = \frac{1}{|D_t|} \sum_{(x,y) \in D_t} \mathbb{I}[y = j]$$

Then, impurity function  $\phi(D_t) = \phi(p_1, \dots, p_C)$  should satisfy:

- $\phi$  is defined for  $p_j \geq 0$  and  $\sum_j p_j = 1$
- $\phi$  is maximized when all  $p_j = 1/C$
- $\phi$  is minimized when a single  $p_j = 1$ , while others  $p_i = 0, i \neq j$
- $\phi$  is symmetric wrt its arguments

# Impurity measures

## Classification

Gini criterion:

$$I(D_t) = \sum_{i=1}^c p_i(1 - p_i) = 1 - \sum_{i=1}^c p_i^2$$

Probability of an error  
when predicting  
randomly with prior  
class probabilities  $p_i$

Entropy:

$$I(D_t) = - \sum_{i=1}^c p_i \log p_i$$

Shortest possible expected  
message length for the  
alphabet distributed under  
 $p_i$

# Stopping criteria

Maximum tree depth

Maximum number of leaves

Minimum number of samples in node to make a split

Minimum number of samples in a leaf

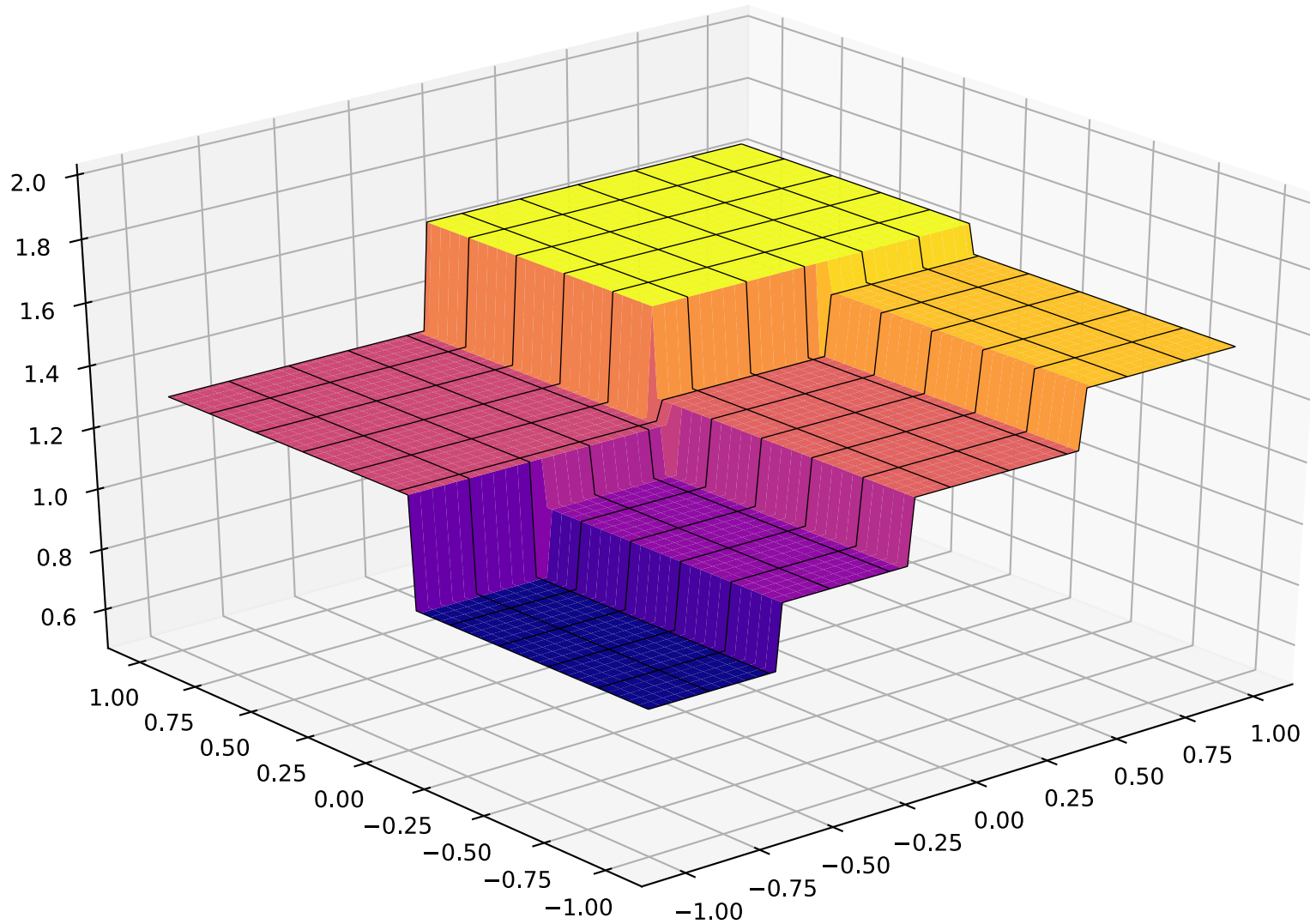
Minimum impurity gain

You name it...

# Solution properties



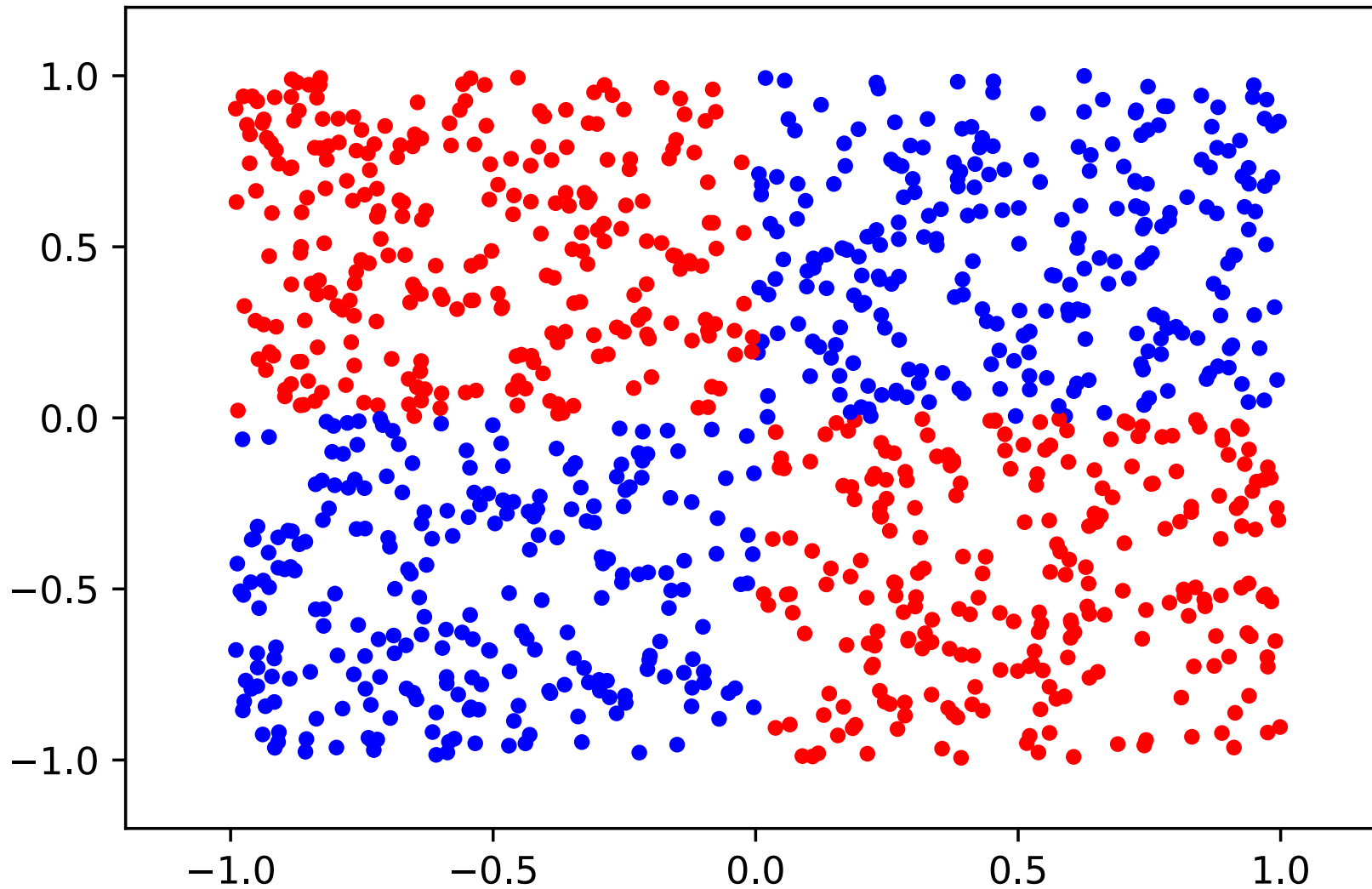
# Prediction function



Decision boundaries  
always **orthogonal to  
feature axes**

Resulting function is a  
**piecewise constant**

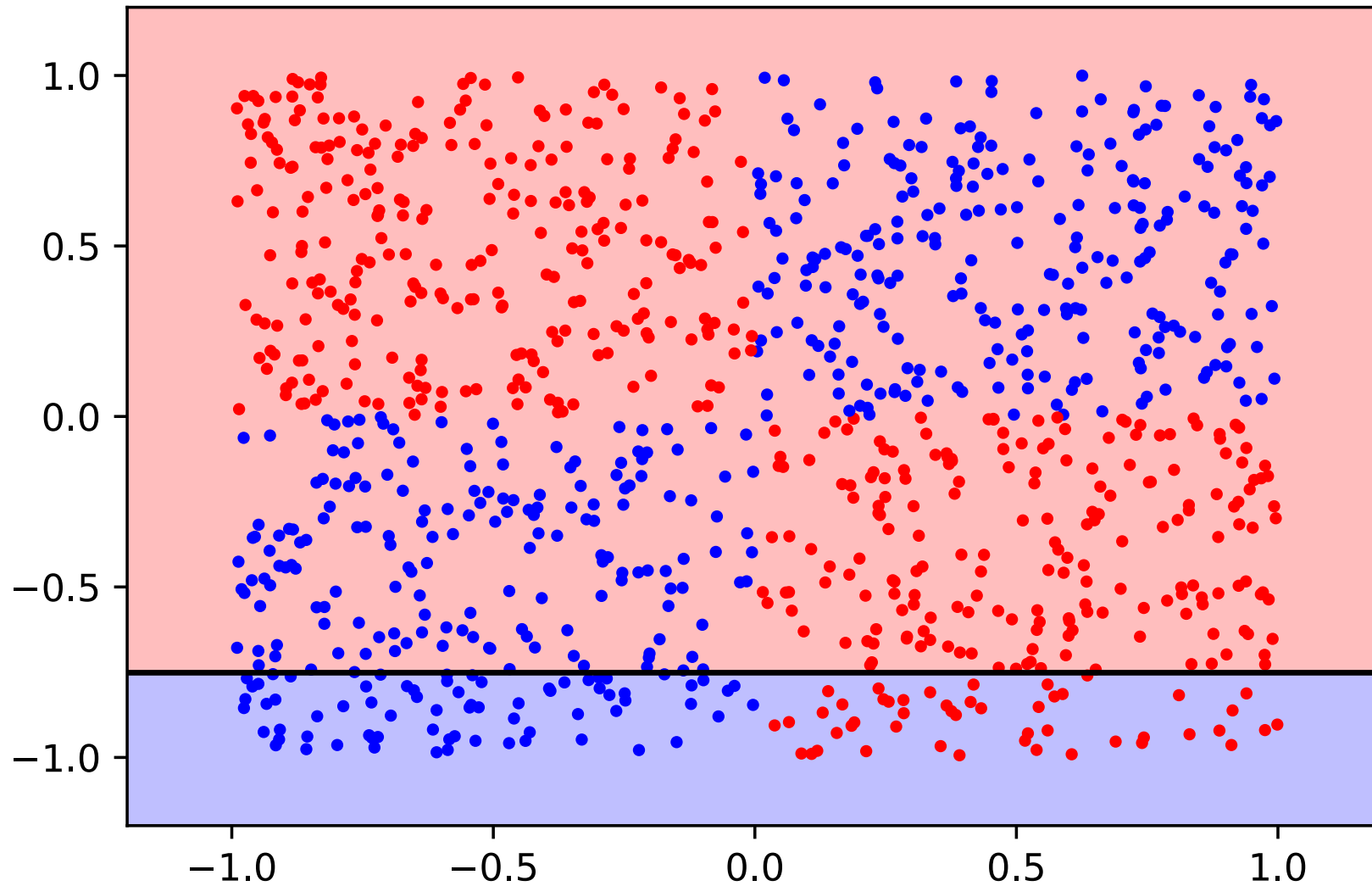
# XOR example



The greedy algorithm does not necessarily lead to the optimal solution!



# XOR example

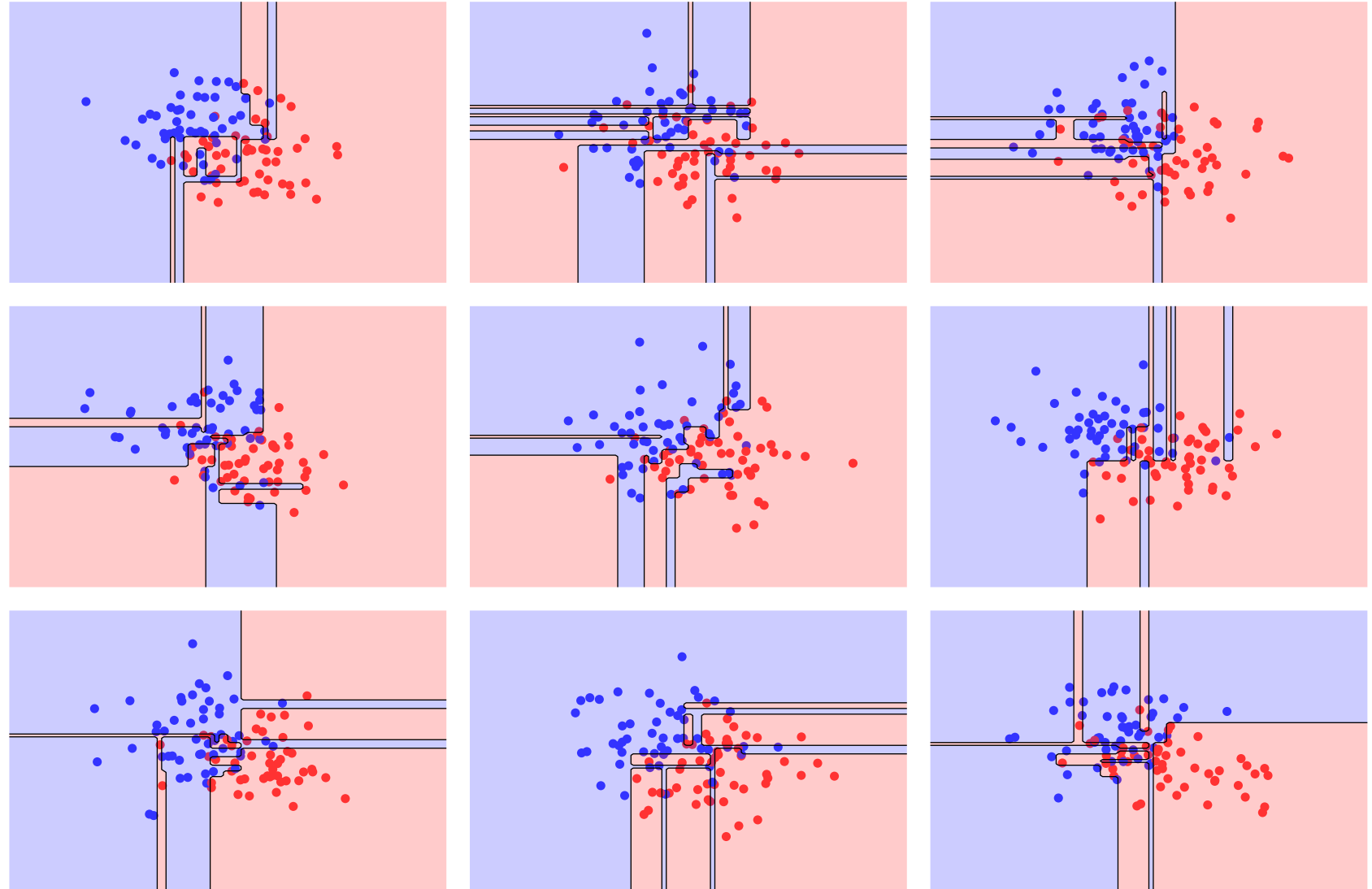


The greedy algorithm does not necessarily lead to the optimal solution!

# High Variance

Without a stopping criterion the tree will grow until every object is classified correctly

Can be regularized by a stopping criterion or with **pruning**



# Cost-Complexity Pruning

Original algorithm optimizes the sample-weighted impurity in the terminal nodes of the tree  $T$ :

$$R(T) = \sum_{t \in \text{leaves}(T)} |D_t| \cdot I(D_t)$$

Can modify this objective by adding a regularizer proportional to the **number of terminal nodes**  $|T|$ :

$$R_\alpha(T) = R(T) + \alpha|T|$$

Idea: build a full tree under  $R(T)$ , then remove some of the nodes to optimize  $R_\alpha(T)$ .

# Cost-Complexity Pruning

Let  $T_t$  be the subtree tree whose root node is  $t \in T$

$T_t$  will be pruned out if:

$$R(T_t) + \alpha|T_t| > R(t) + \alpha$$

or in other words if:

$$\alpha > \alpha_{\text{eff}}(t) = \frac{R(t) - R(T_t)}{|T_t| - 1}$$

# Categorical features

Ordinal → label encoding (preserving the order!)

Nominal → order the categories with:

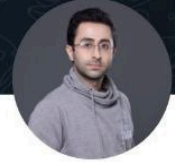
- positive class probability (binary classification)
- target mean/median (regression)
- (make sure the categories are **well populated** to avoid overfitting!)

# Thank you!

Majid Sohrabi



[msohrabi@hse.ru](mailto:msohrabi@hse.ru)



@MSOHRABI\_CS