

10. Decision Trees

**EECE454 Introduction to
Machine Learning Systems**

2023 Fall, Jaeho Lee

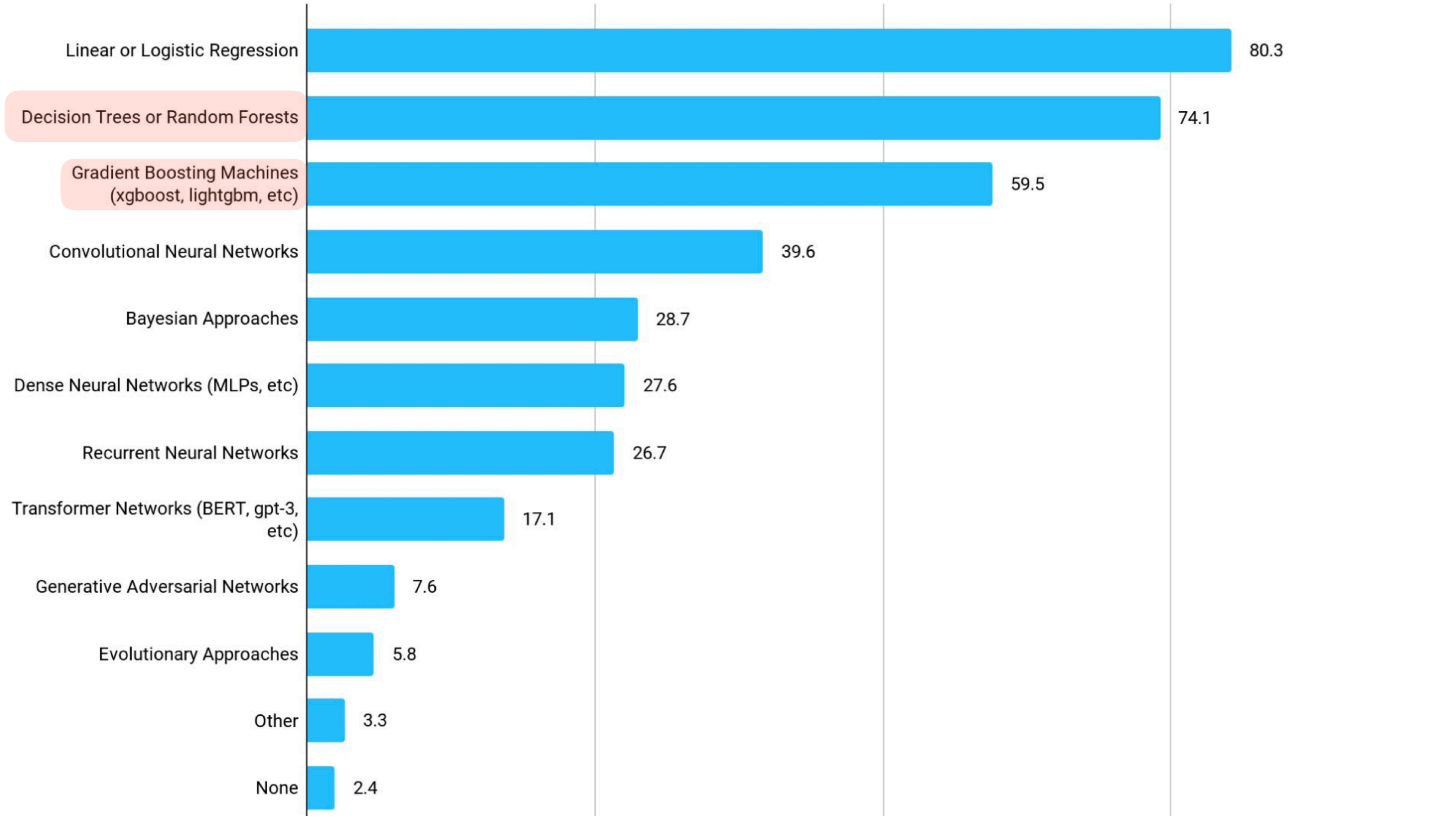
Kaggle

- A competition platform for ML
 - People upload data & put bounty on it. You solve it

The screenshot displays the Kaggle homepage. On the left is a navigation sidebar with the Kaggle logo and menu items: Create, Home, Competitions (highlighted), Datasets, Models, Code, Discussions, Learn, and More. The main content area features a search bar for competitions, a 'Filters' button, and a row of category tabs: All competitions, Featured, Getting Started, Research, Community, Playground, Simulations, and Analytics. Below this is the 'Active Competitions' section, sorted by 'Hotness'. It lists four competitions:

Competition Name	Bounty	Time Remaining
Open Problems – Single-Cell Perturbations	\$100,000	2 months to go
Stanford Ribonanza RNA Folding	\$100,000	2 months to go
Optiver - Trading at the Close	\$100,000	2 months to go
CommonLit - Evaluate Student Summaries	\$60,000	4 days to go

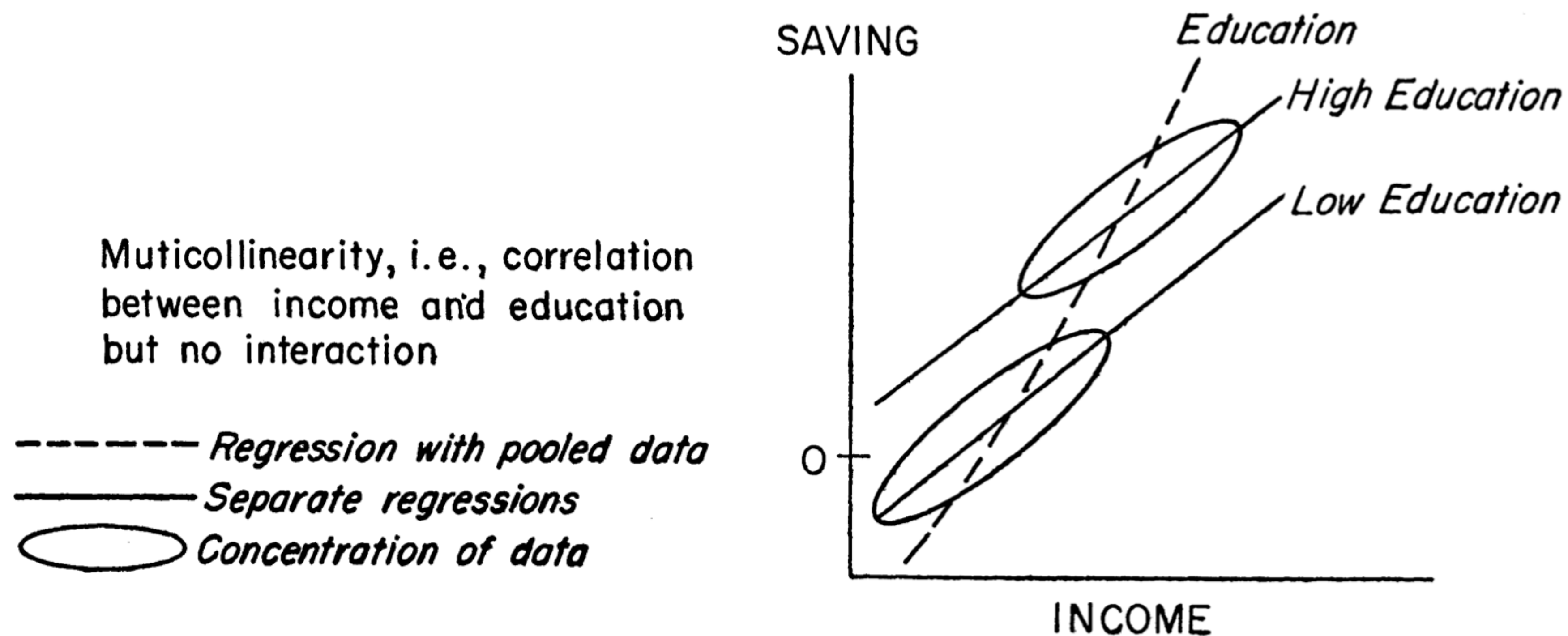
Kaggle Survey 2021



The Birth of Decision Trees

Motivation

- Some trace it back to Porphyry (234?—305?), a Greek philosopher.
- **Modern use.** Survey data analysis by Morgan & Sonquist (1963)
 - Some data demonstrated *multicollinearity*

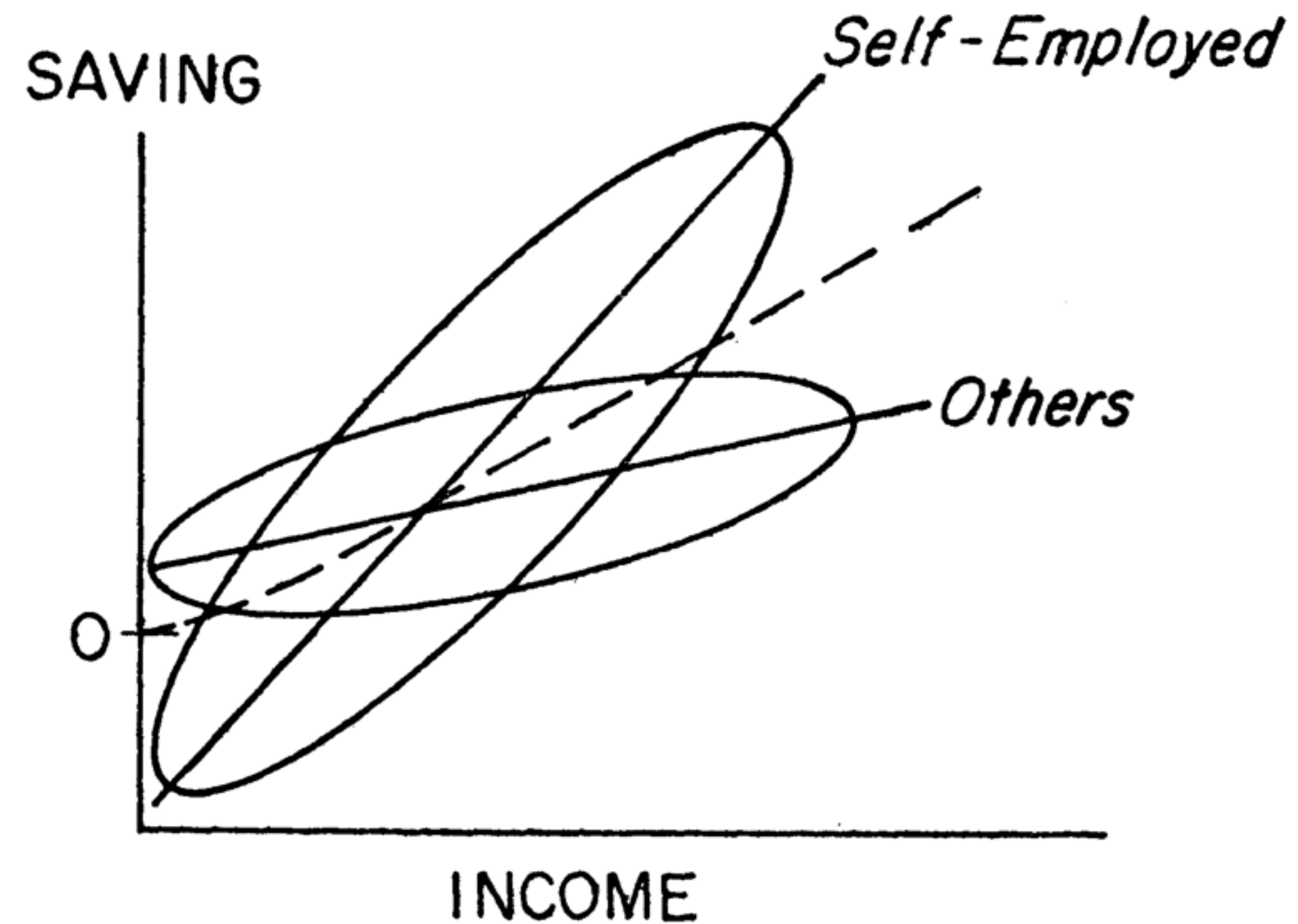


Motivation

- Some data demonstrated *interaction between features*

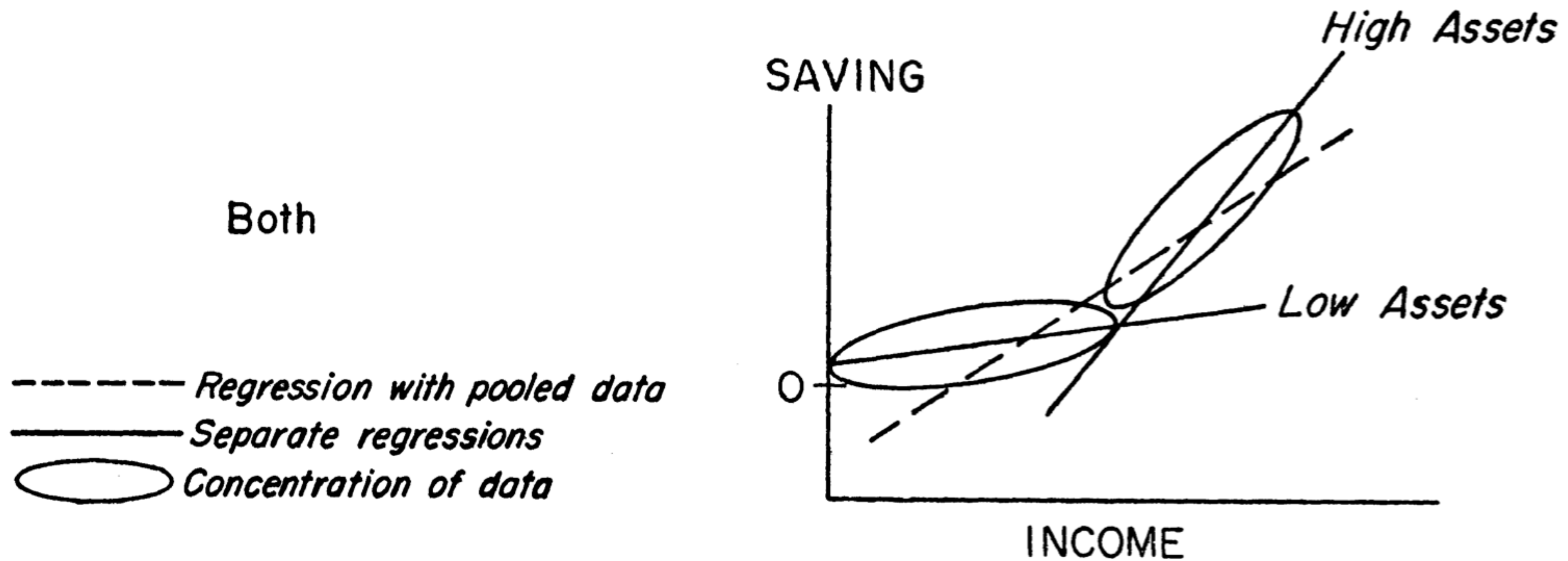
Interaction, but no multicollinearity
(no correlation between income and
self-employment)

- *Regression with pooled data*
- *Separate regressions*
- *Concentration of data*



Motivation

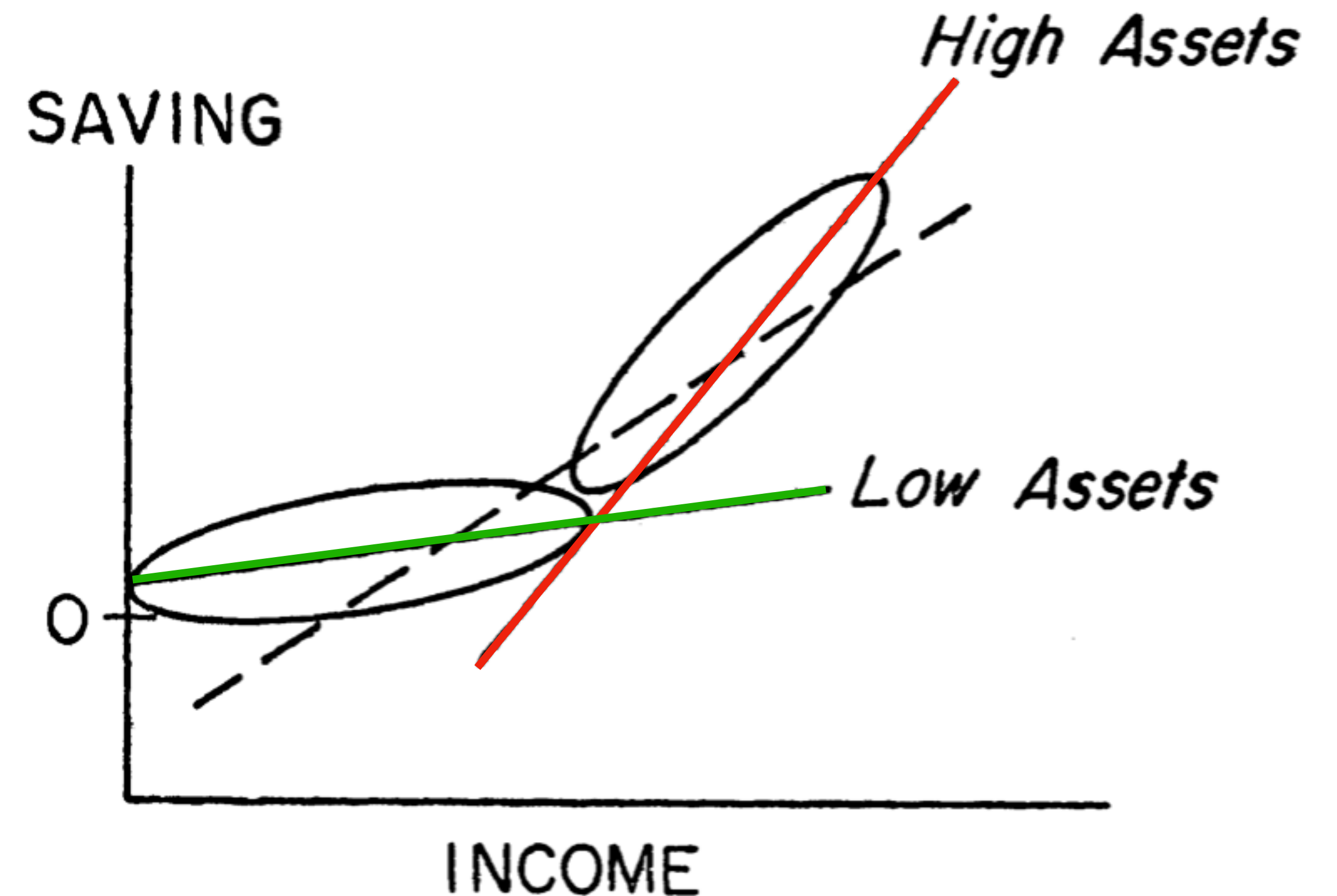
- Some data demonstrated **both**



Idea

- We may need some **sequential** approach
(instead of blindly assuming “additive” interactions)

- High asset?
 - Yes \Rightarrow Use **curve 1**
 - No \Rightarrow Use **curve 2**



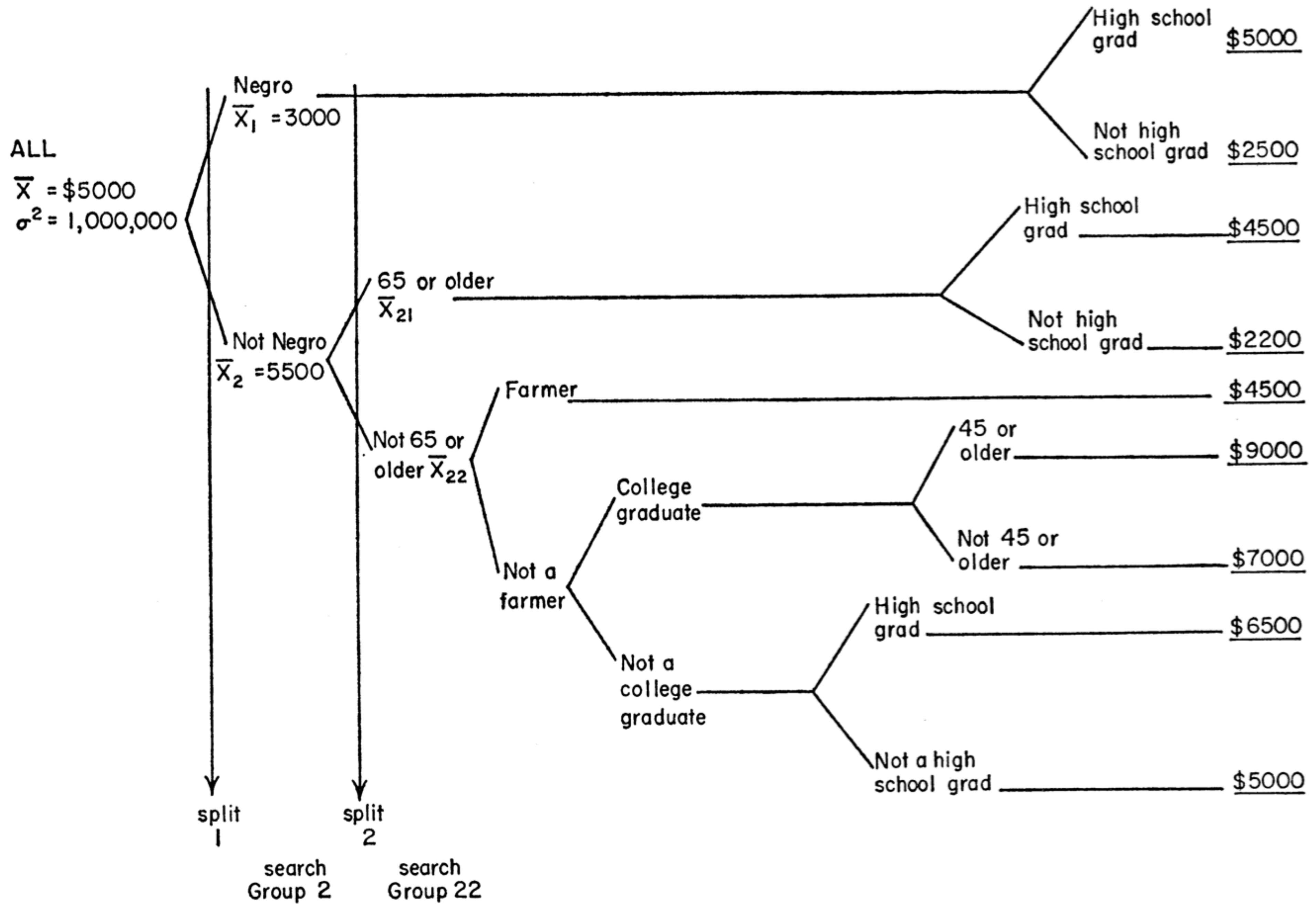


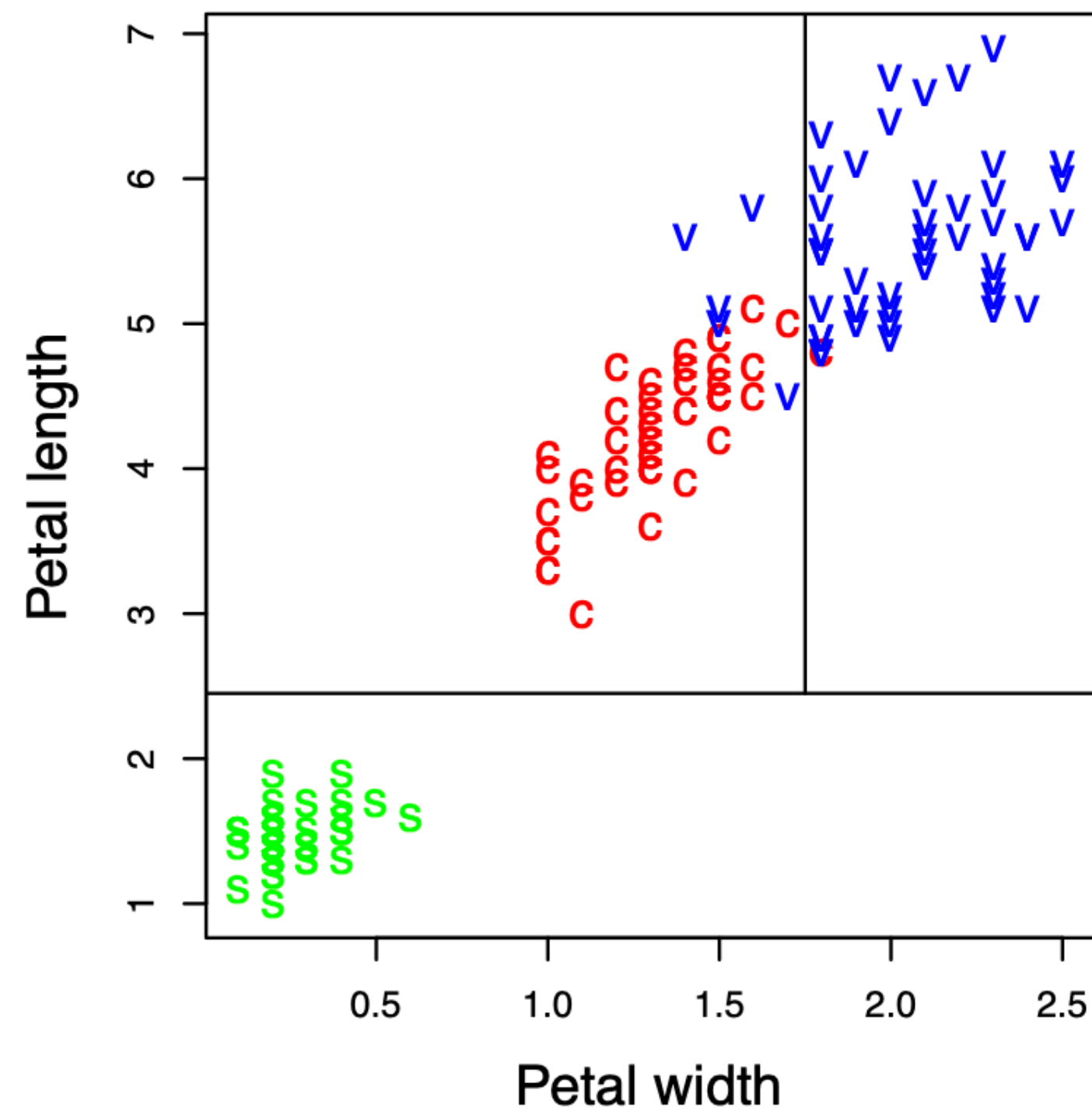
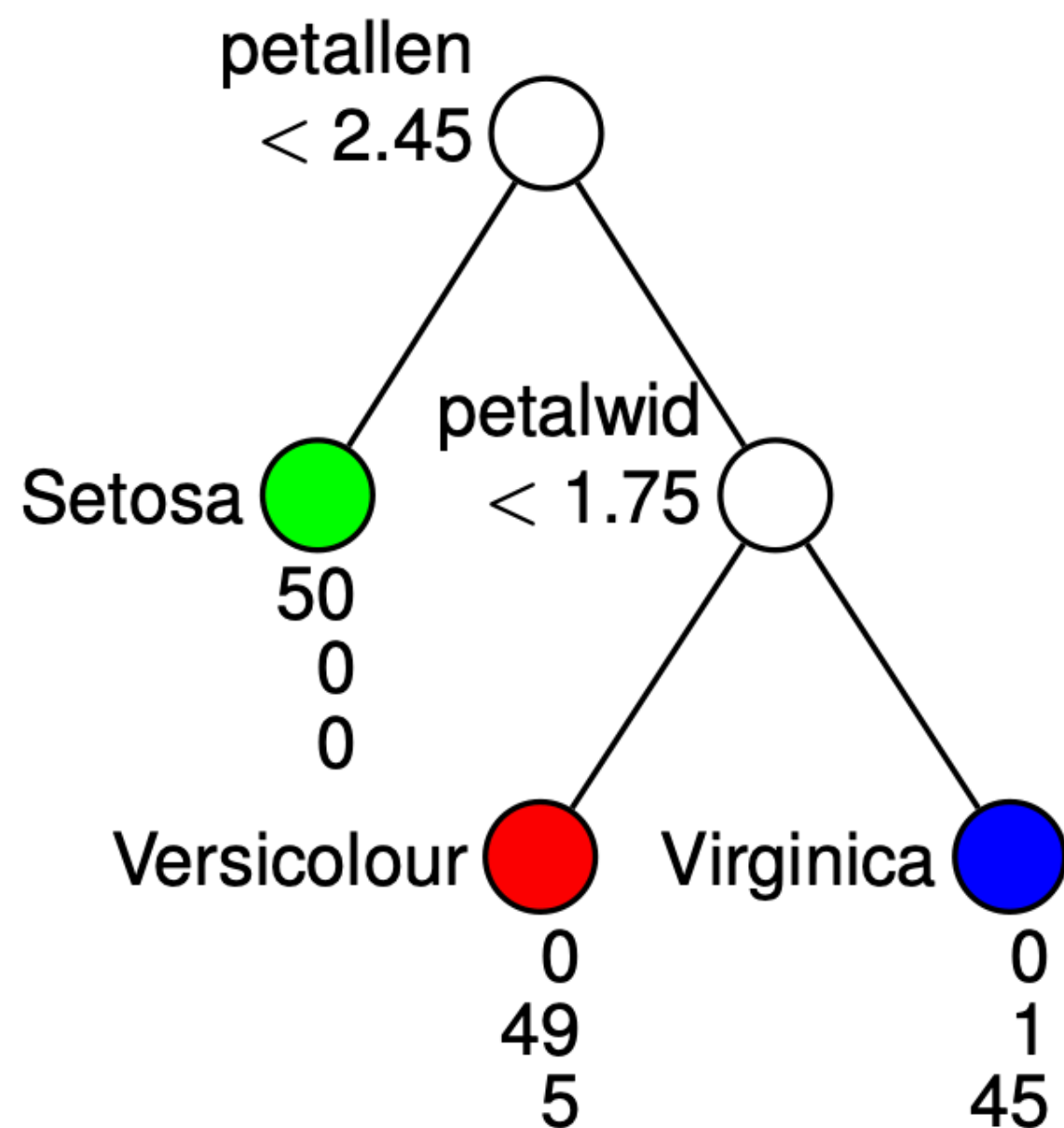
CHART II. Annual Earnings.

Decision Tree

Overview

- **What it is.** Nested if-then-else statements
- **Advantages.**
 - Relatively easy to interpret (when not too large)
 - Fast execution (when not too large)
 - Standard algorithm has nice properties.
- **vs. nearest neighbors.**
 - Both are nonparametric, based on *local regularity*

- A binary tree which recursively partitions/refines the input space.
 - Each **tree node** is associated with a splitting rule $g : \mathcal{X} \rightarrow \{0,1\}$.
 - Each **leaf node** is associated with a label \hat{y} .
 - **Prediction.** Given \mathbf{x} , recurse down the tree until a leaf is reached. Then, output its label.



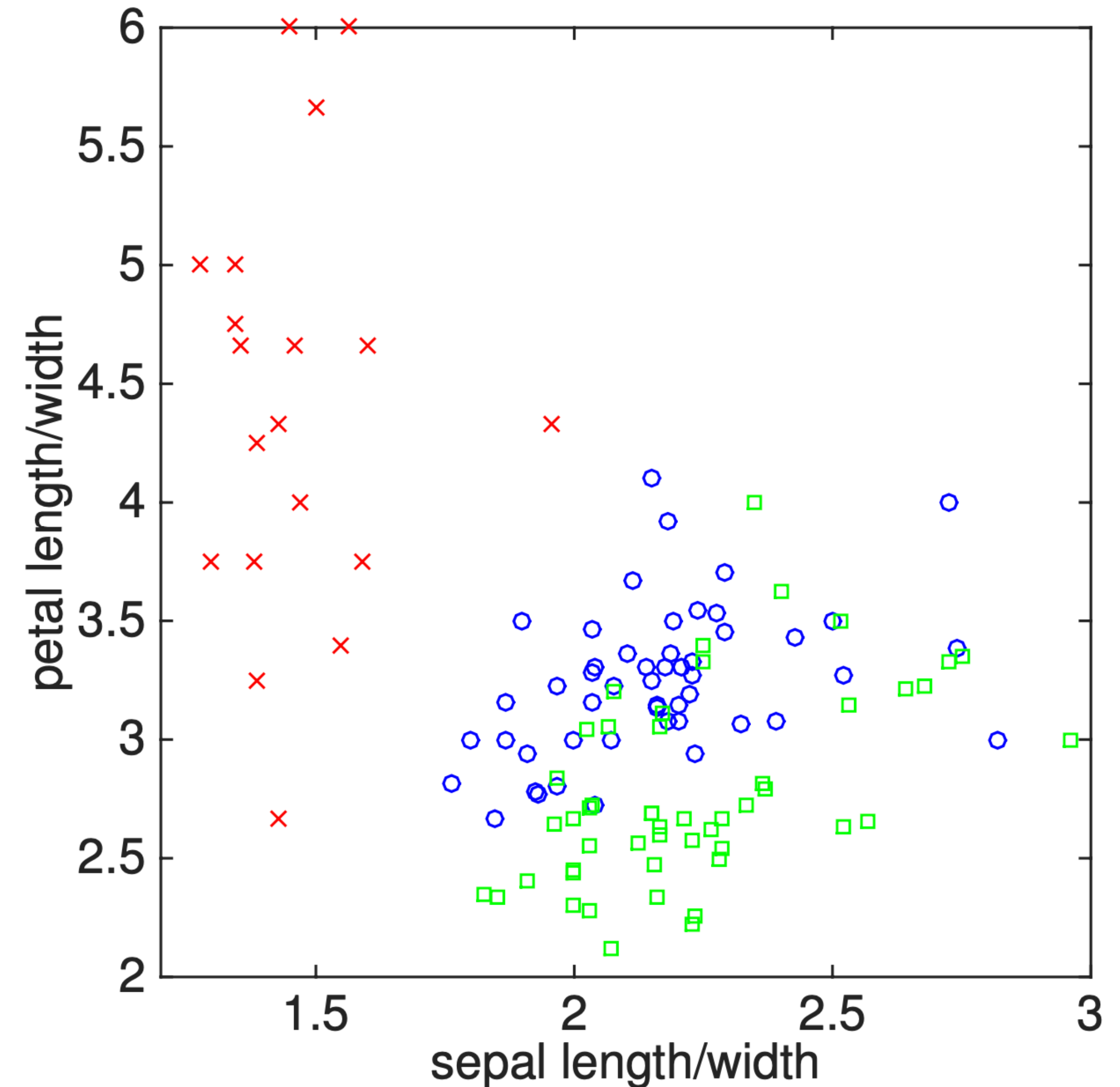
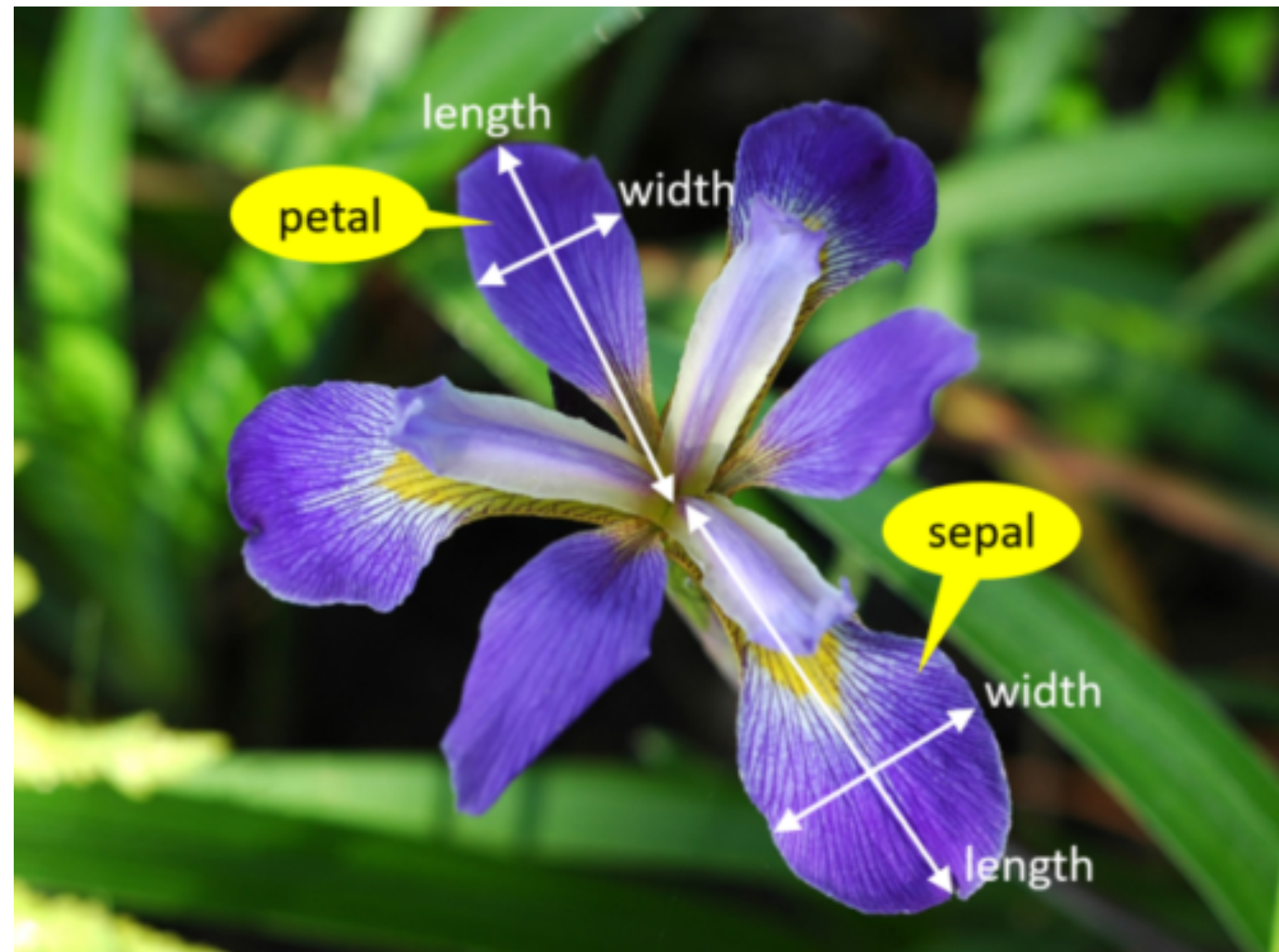
Typically, when $\mathcal{X} = \mathbb{R}^d$, only consider rules like

$$g(\mathbf{x}) = \mathbf{1}[x_i \geq t]$$

(called axis-aligned splits or coordinate splits) why?

Example: Iris Classification

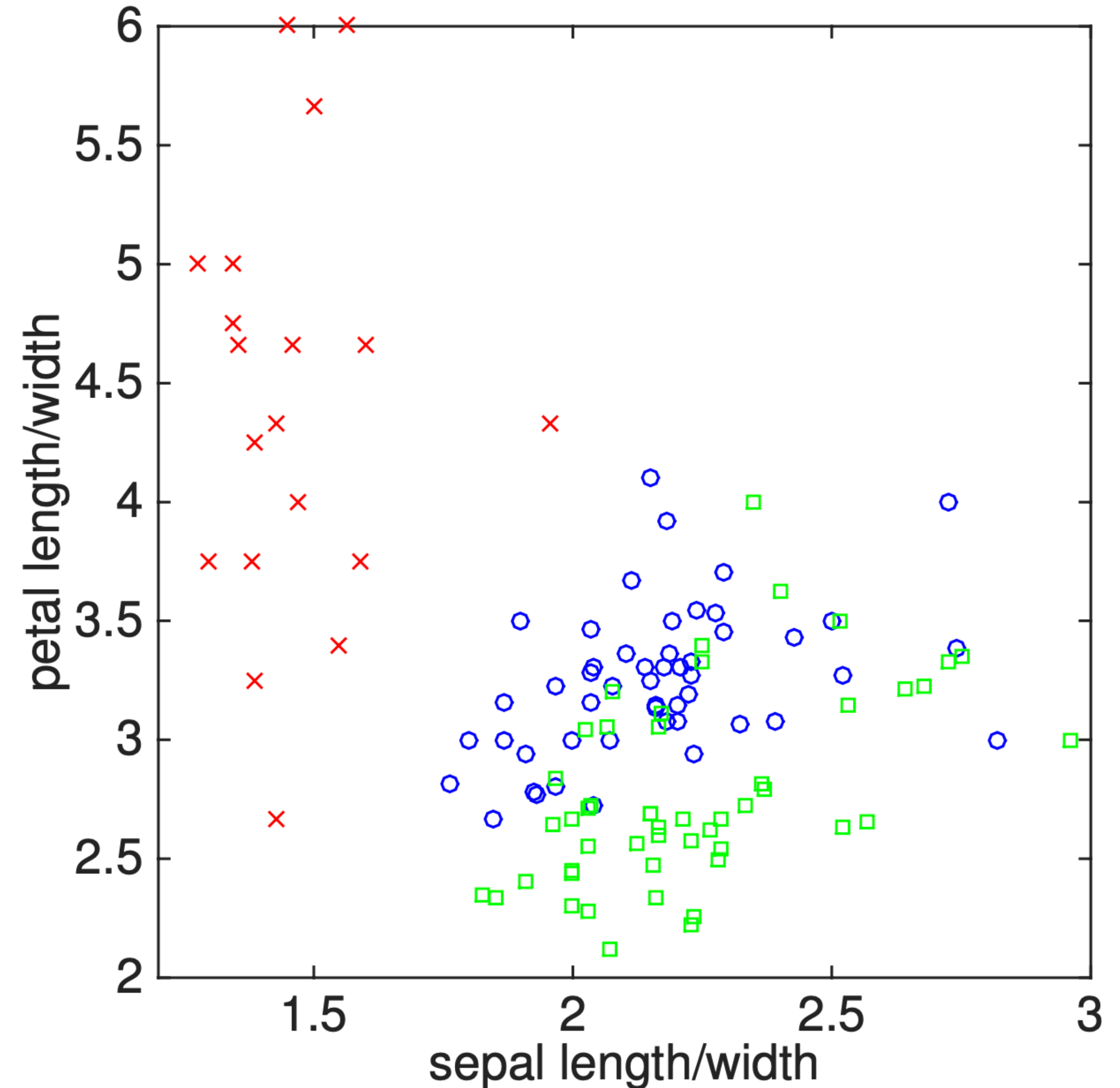
- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
 - x_1 = ratio of sepal length/width
 - x_2 = ratio of petal length/width



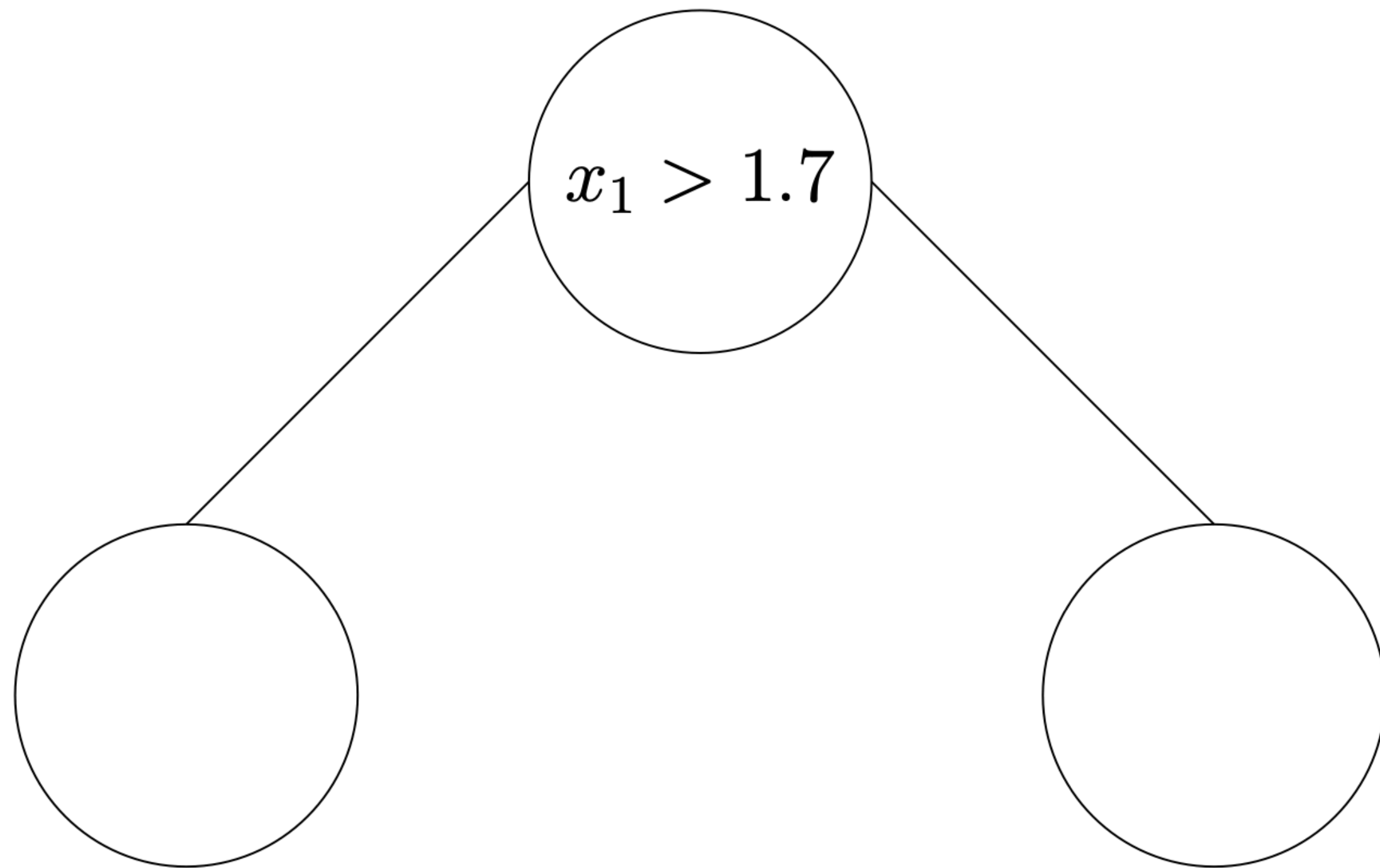
Example: Iris Classification

$$\hat{y} = 2$$

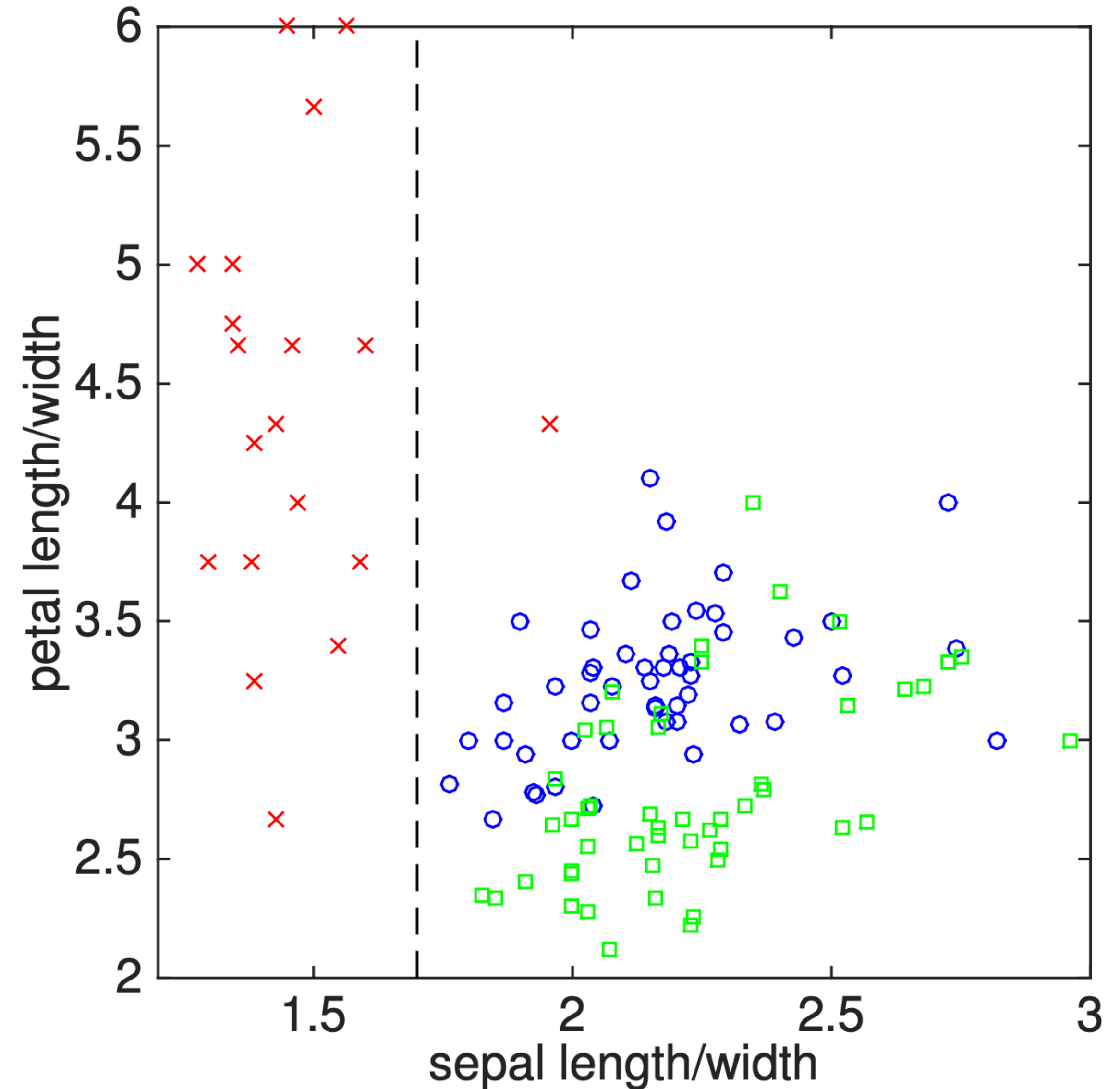
(according to some prediction rule)



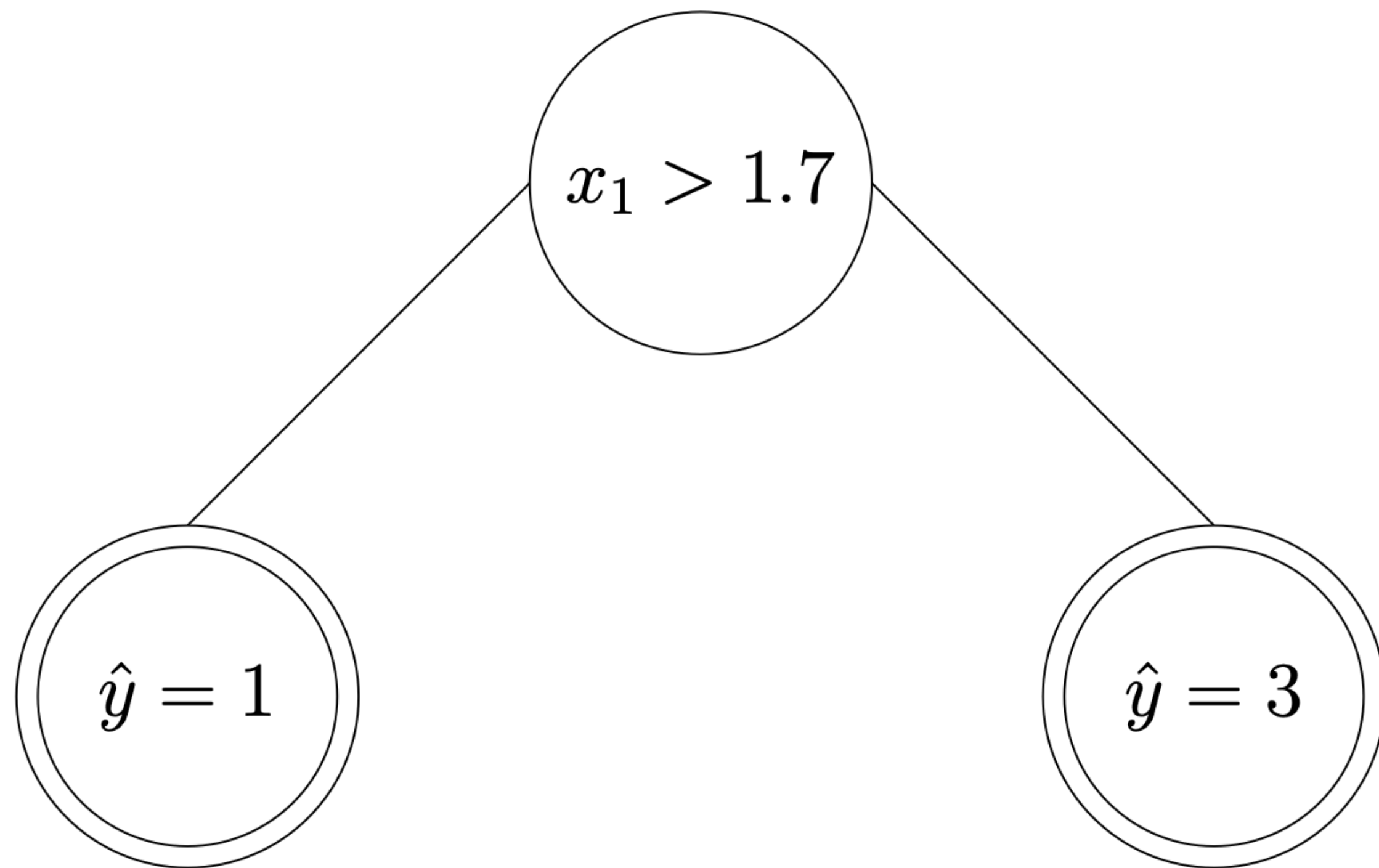
Example: Iris Classification



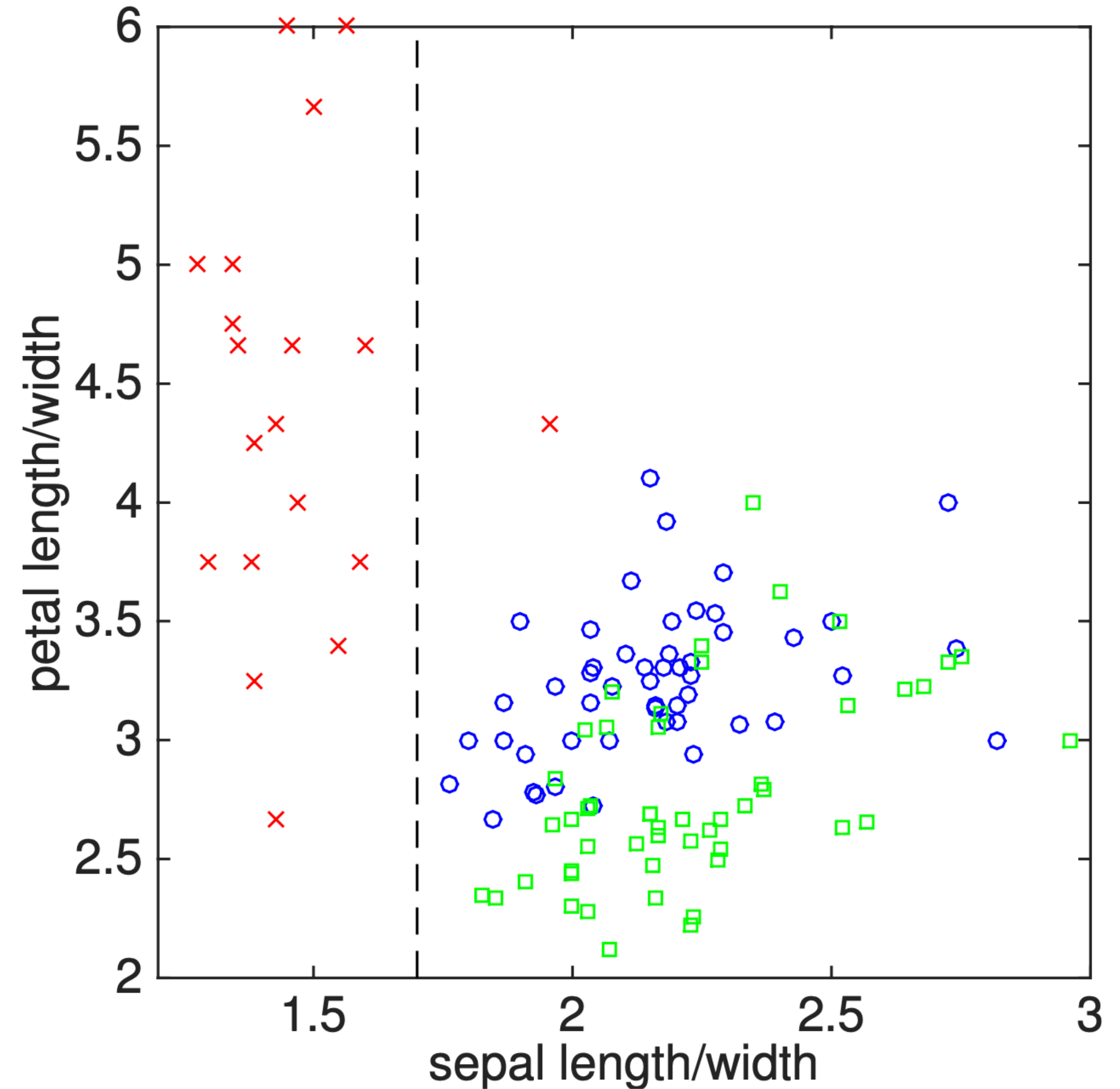
(according to some **splitting rule**)



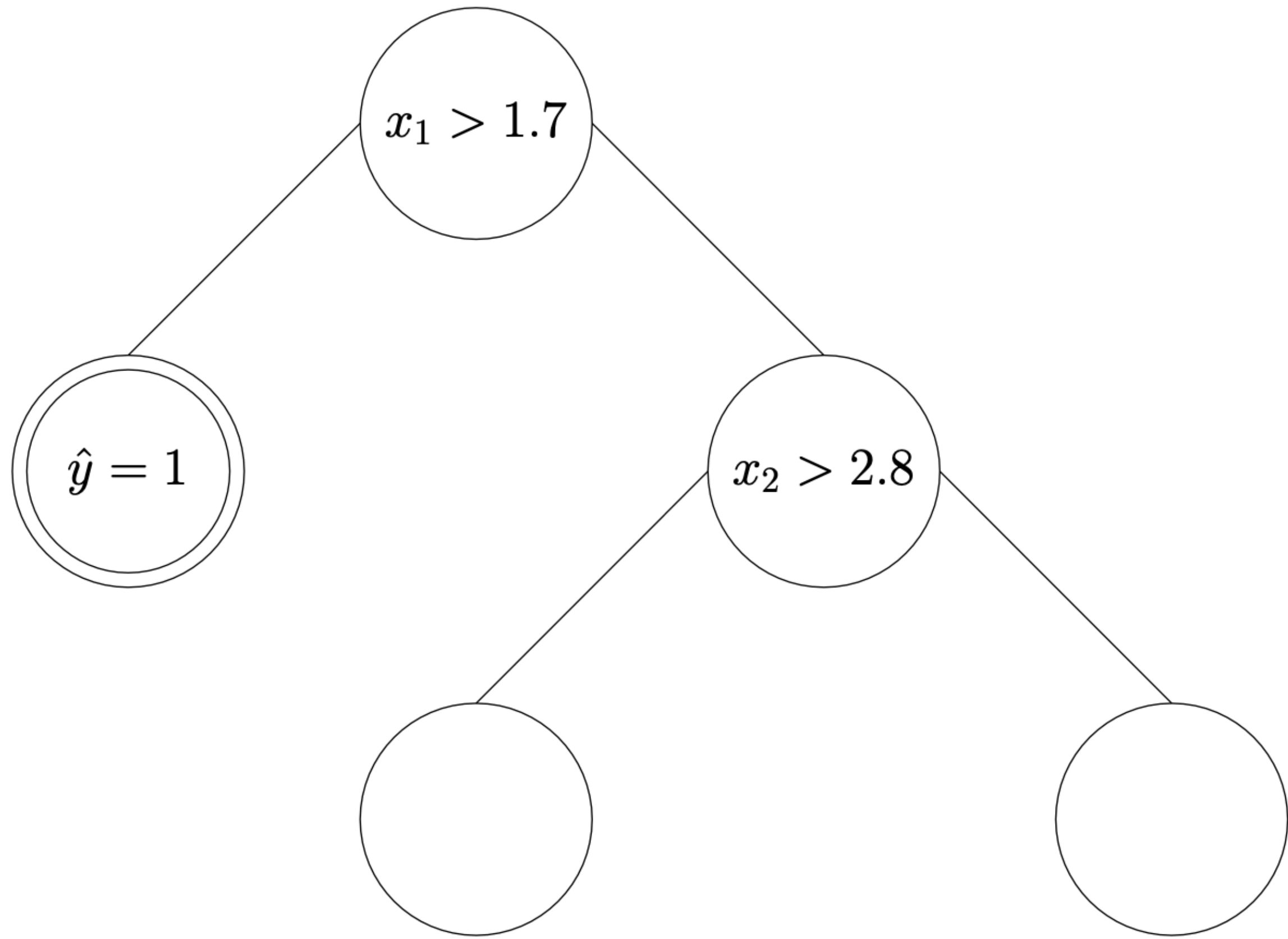
Example: Iris Classification



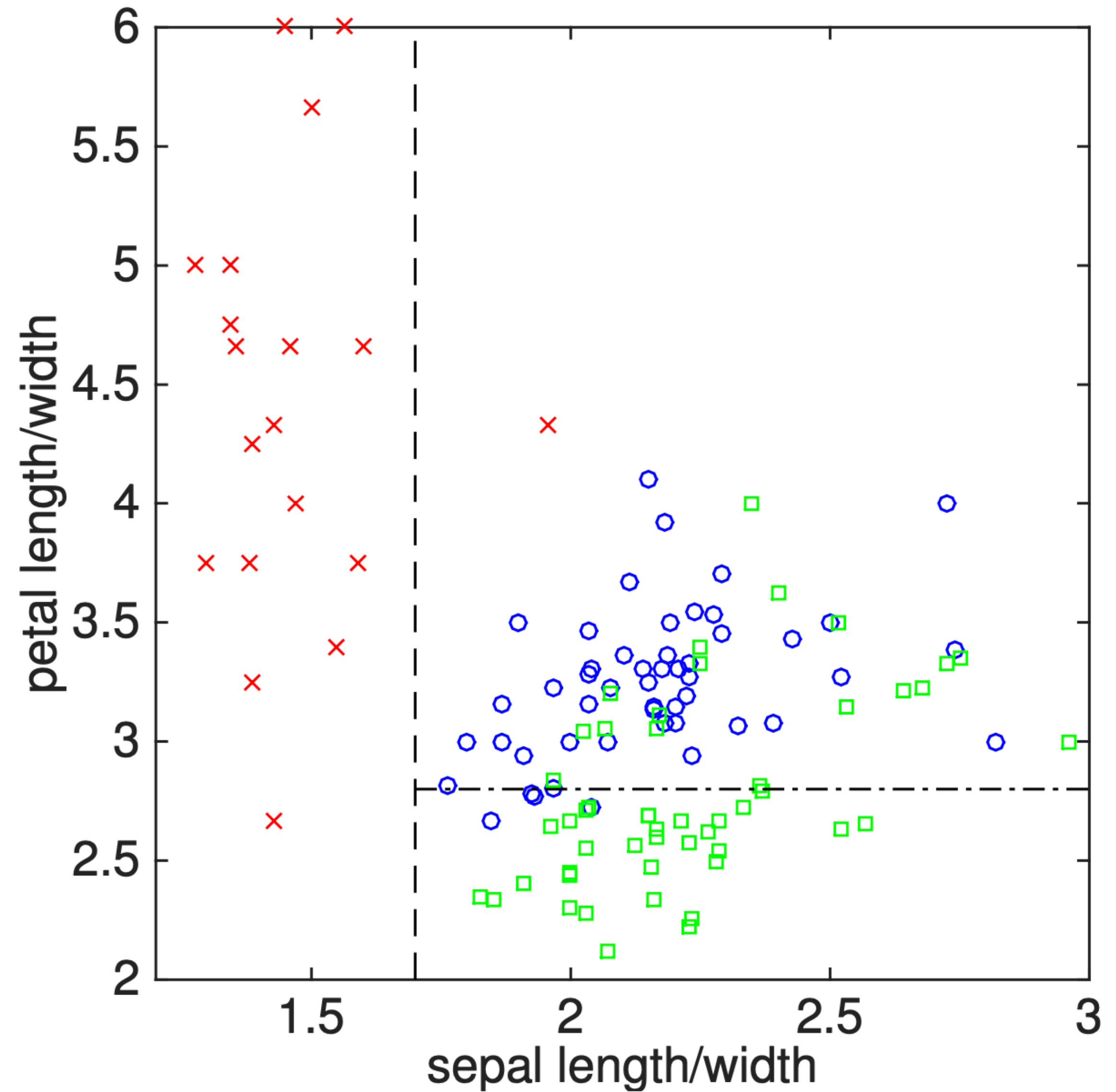
(according to some **prediction rule**)



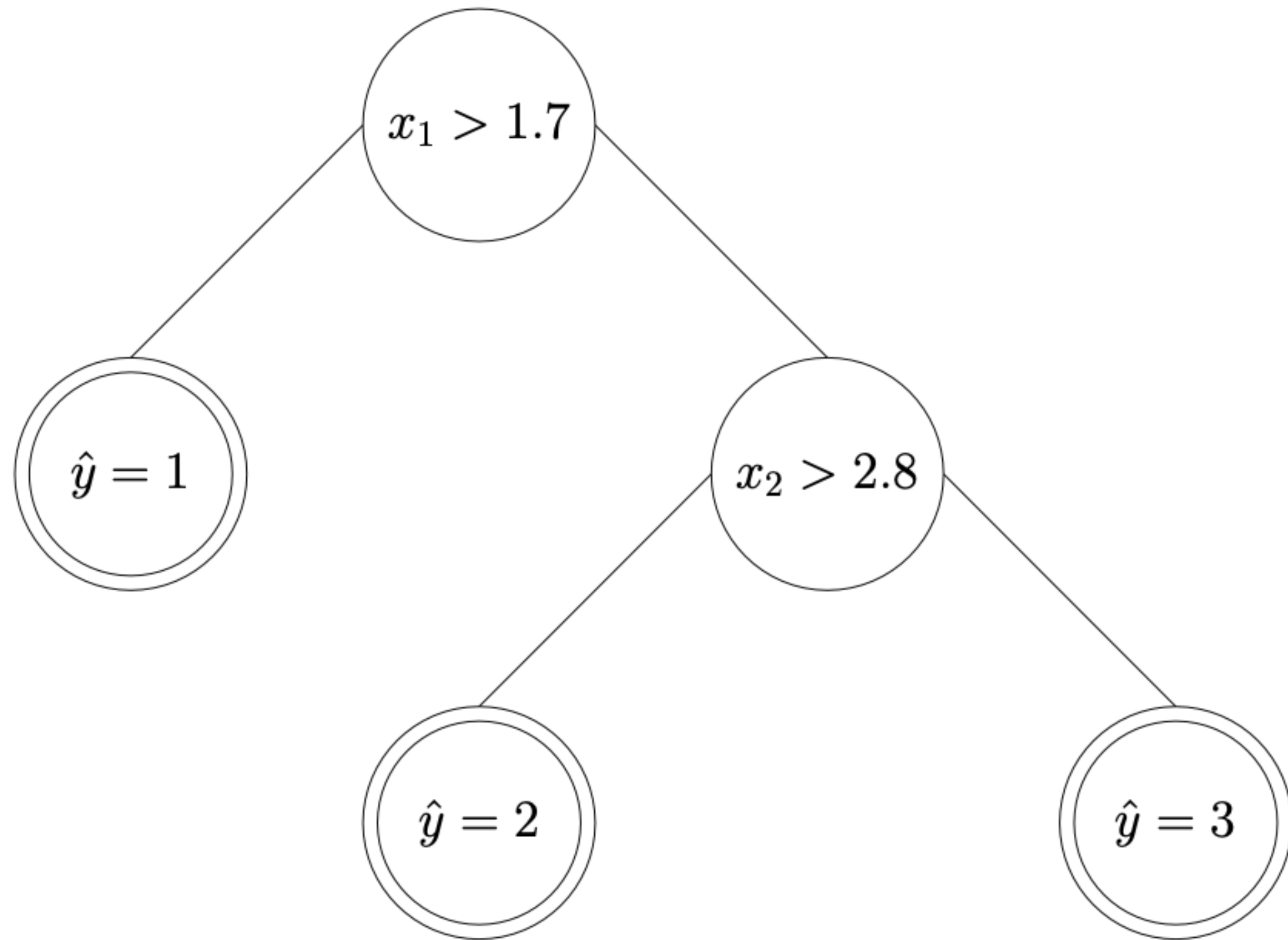
Example: Iris Classification



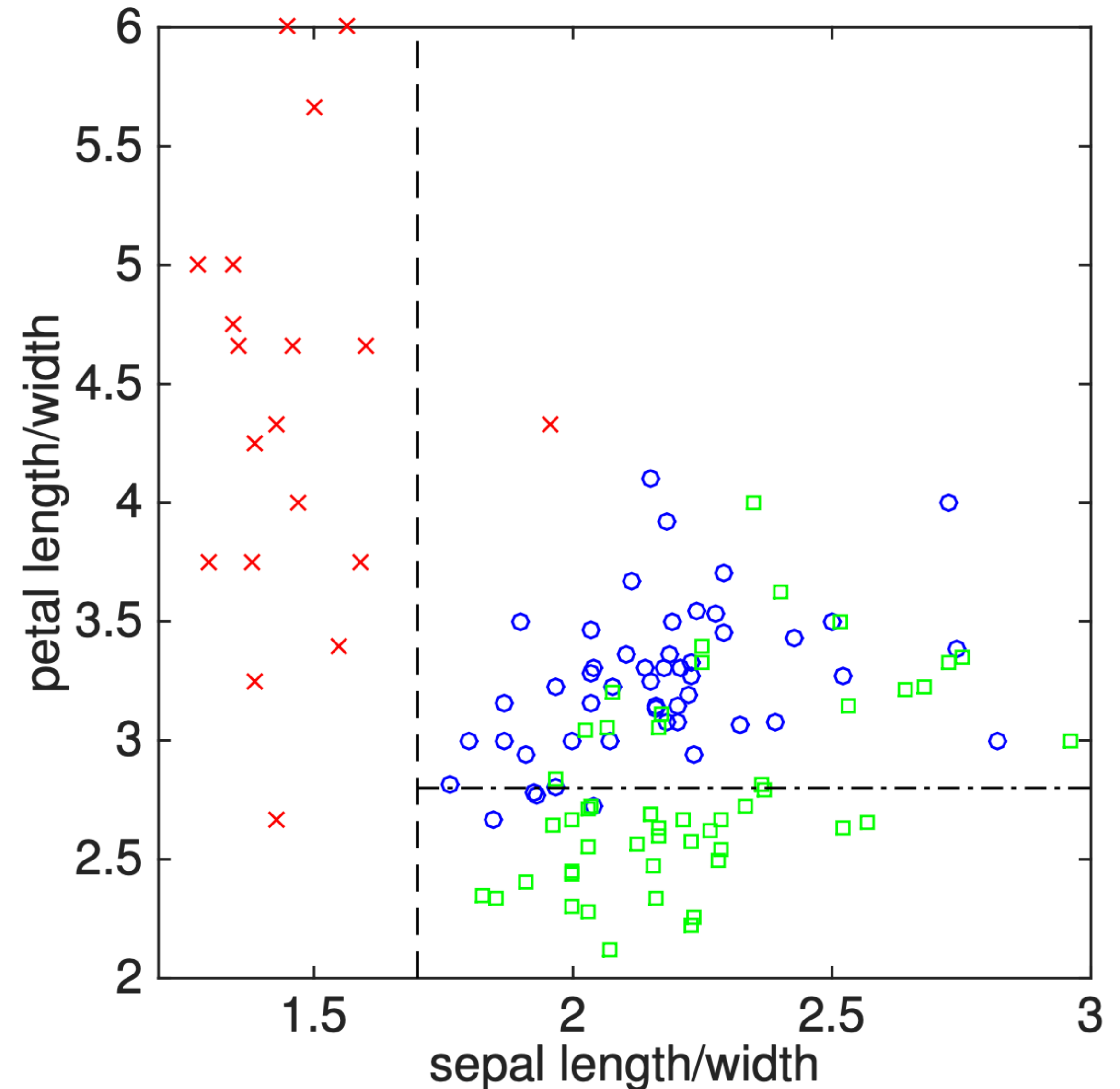
(according to some **stopping rule**)



Example: Iris Classification



(according to some **prediction rule**)



Algorithm

Elements of decision tree algorithm.

- We need three rules:
 - Prediction rule, Splitting rule, Stopping rule

until all leaf node is stopped:

visit a leaf node

if(stopping_rule(node) = True):

 apply prediction rule

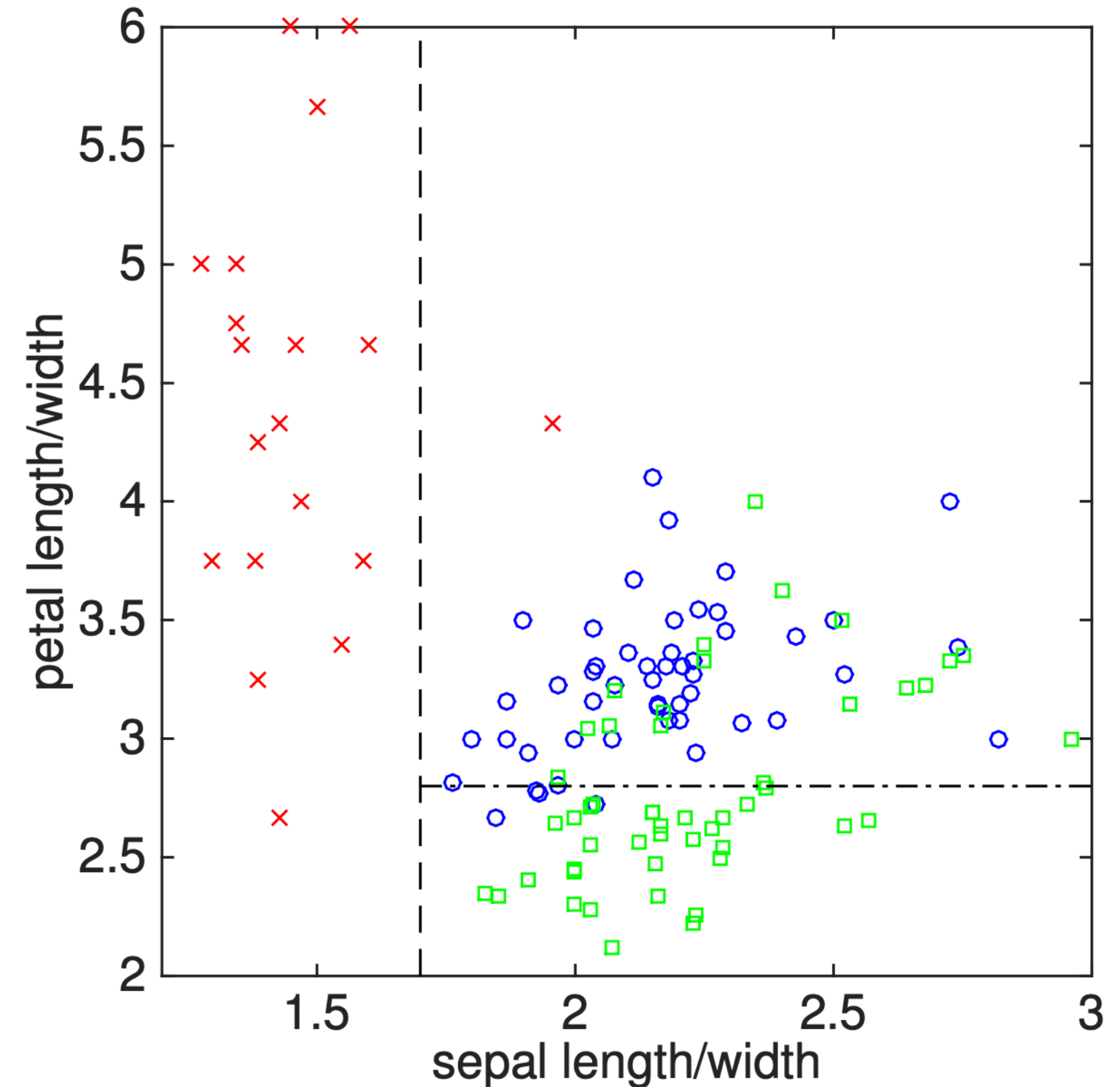
 stop the node

else:

 split the node, using the splitting rule

Prediction

- Usually very simple.
 - **Classification:** Majority
 - **Regression:** Average, median...



Splitting

- **Idea.** Partition the data to minimize the *uncertainty* for each cell.
- Example. Binary classification; if a set S has $p \cdot |S|$ labeled $+1$.

- **Classification error:**

$$u(S) = \min\{p, 1 - p\}$$

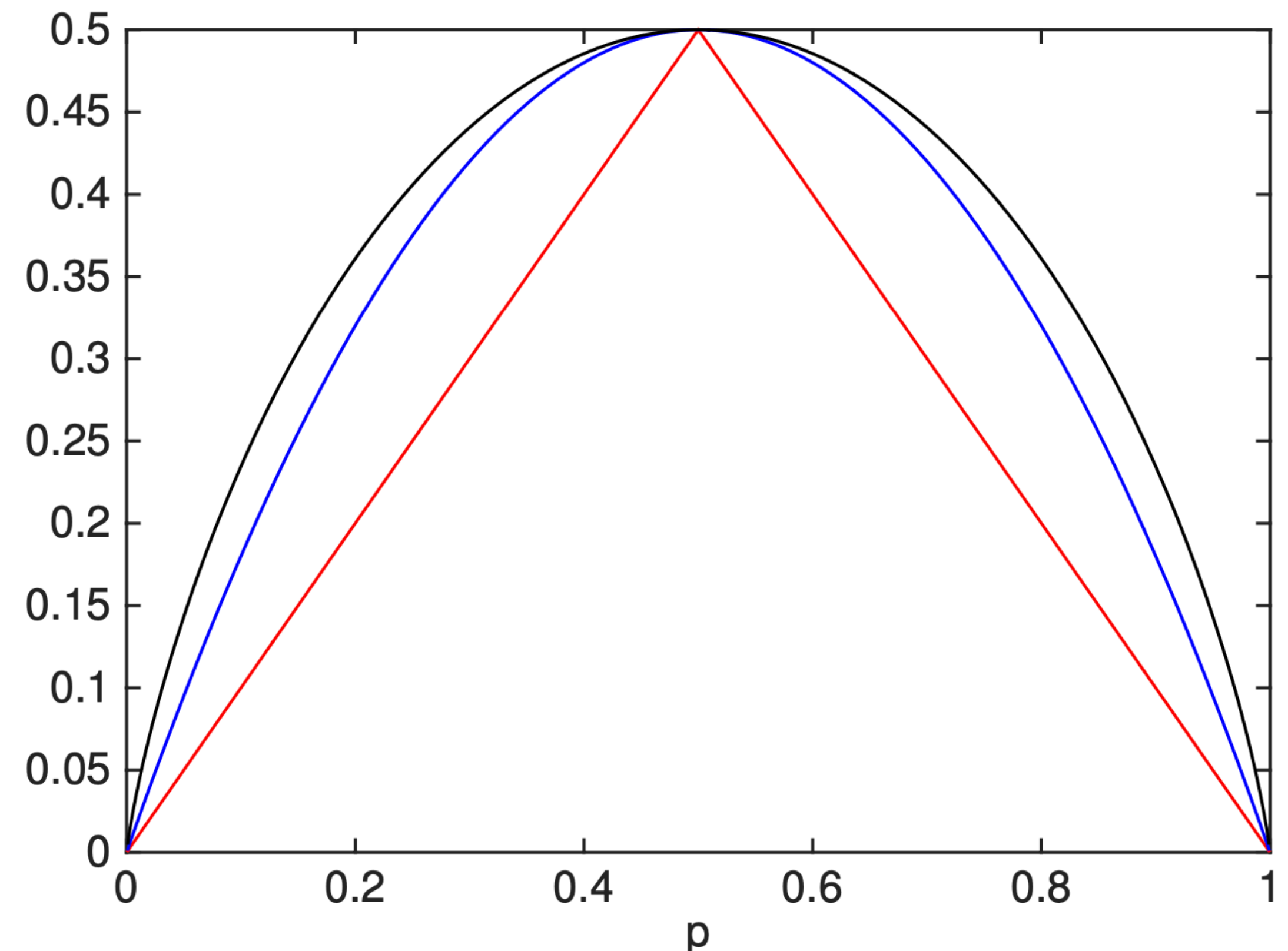
- **Gini Index:**

$$u(S) = 2p(1 - p)$$

- **Entropy:**

$$u(S) = p \log \frac{1}{p} + (1 - p) \log \frac{1}{1 - p}$$

(G,E are concave upper bounds on C)

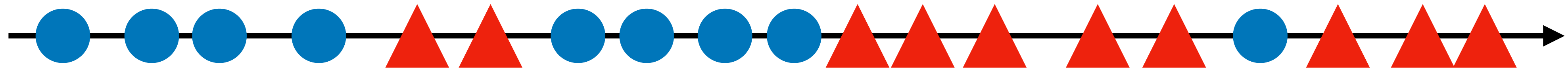


Splitting

- If we split a set S into S_1 and S_2 , we want to minimize

$$|S_1| \cdot u(S_1) + |S_2| \cdot u(S_2)$$

- **Question.** How to find such S_1 and S_2 ?
 - Depends on uncertainty measures
 - For classification, try the boundaries of the same-class clusters



Splitting

- The iterative algorithm is a “greedy” way to minimize

$$u(\mathcal{T}) := \frac{1}{n} \sum_{\text{leaf } S \in \mathcal{T}} |S| \cdot u(S)$$

- Greedy algorithms can fail in many cases, e.g., XOR
(mixing some random splits help)

until all leaf node is stopped:

visit a leaf node

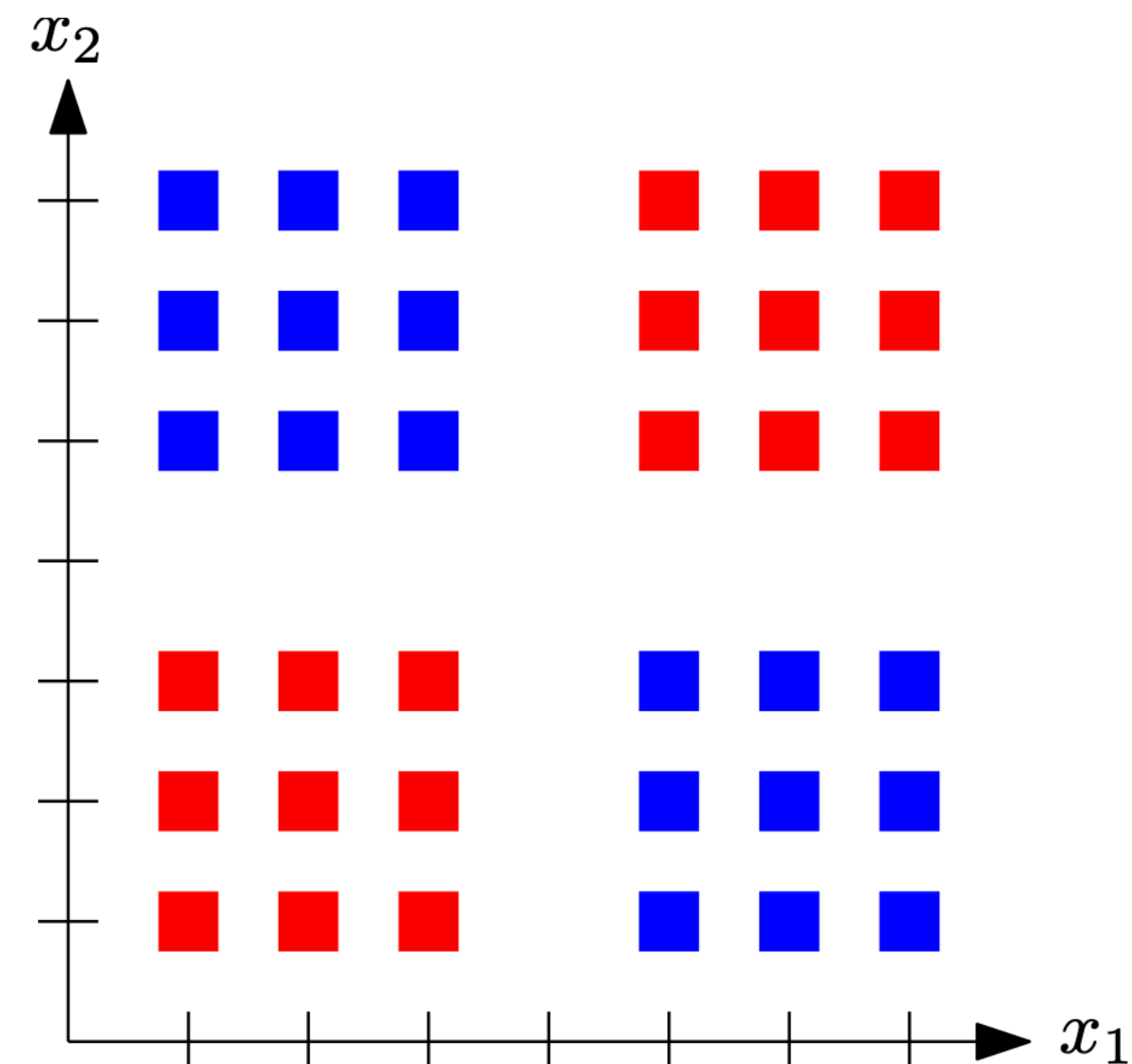
if(*stopping_rule*(node) = True):

 apply *prediction rule*

 stop the node

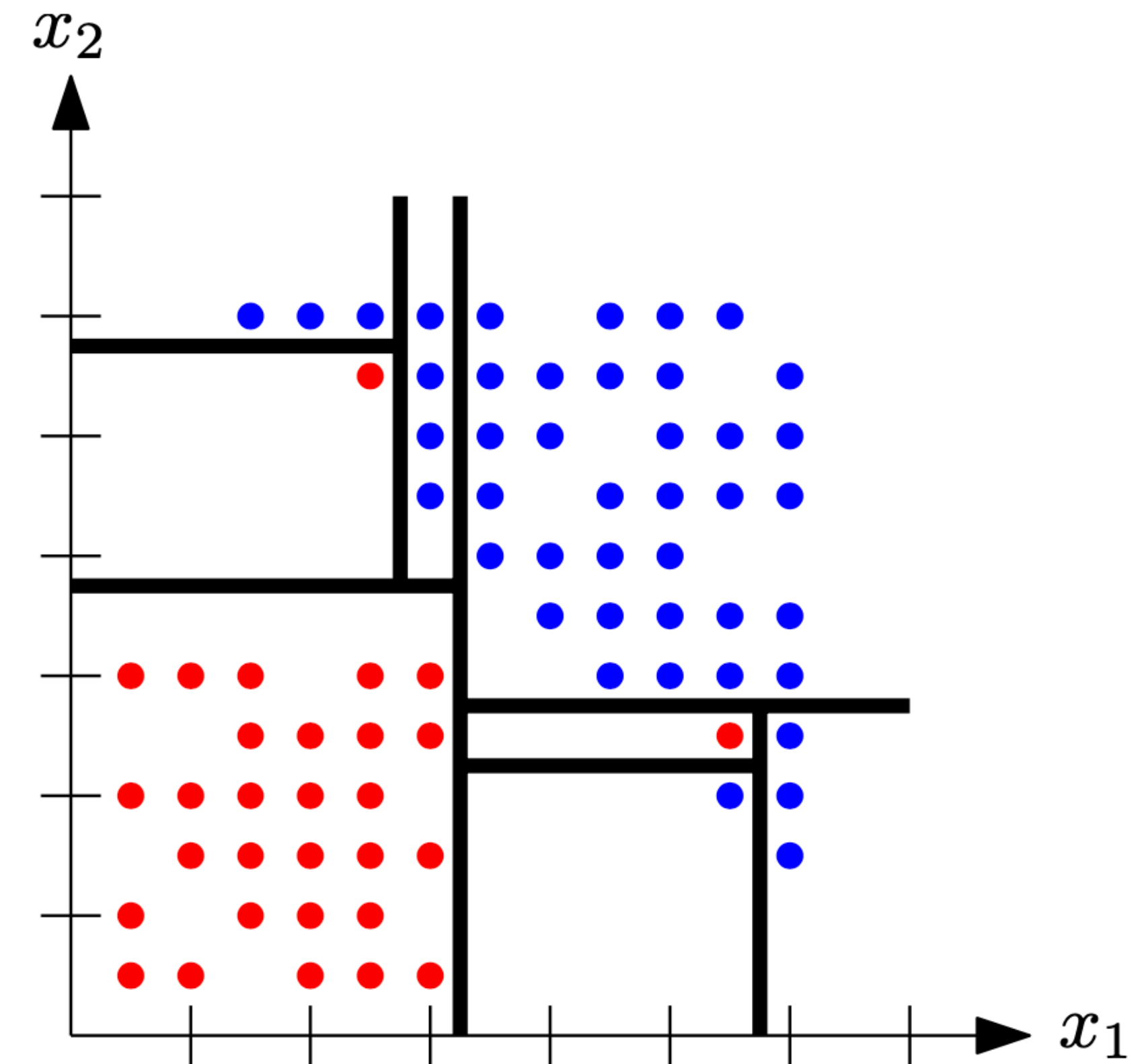
else:

 split the node, using the *splitting rule*



Stopping

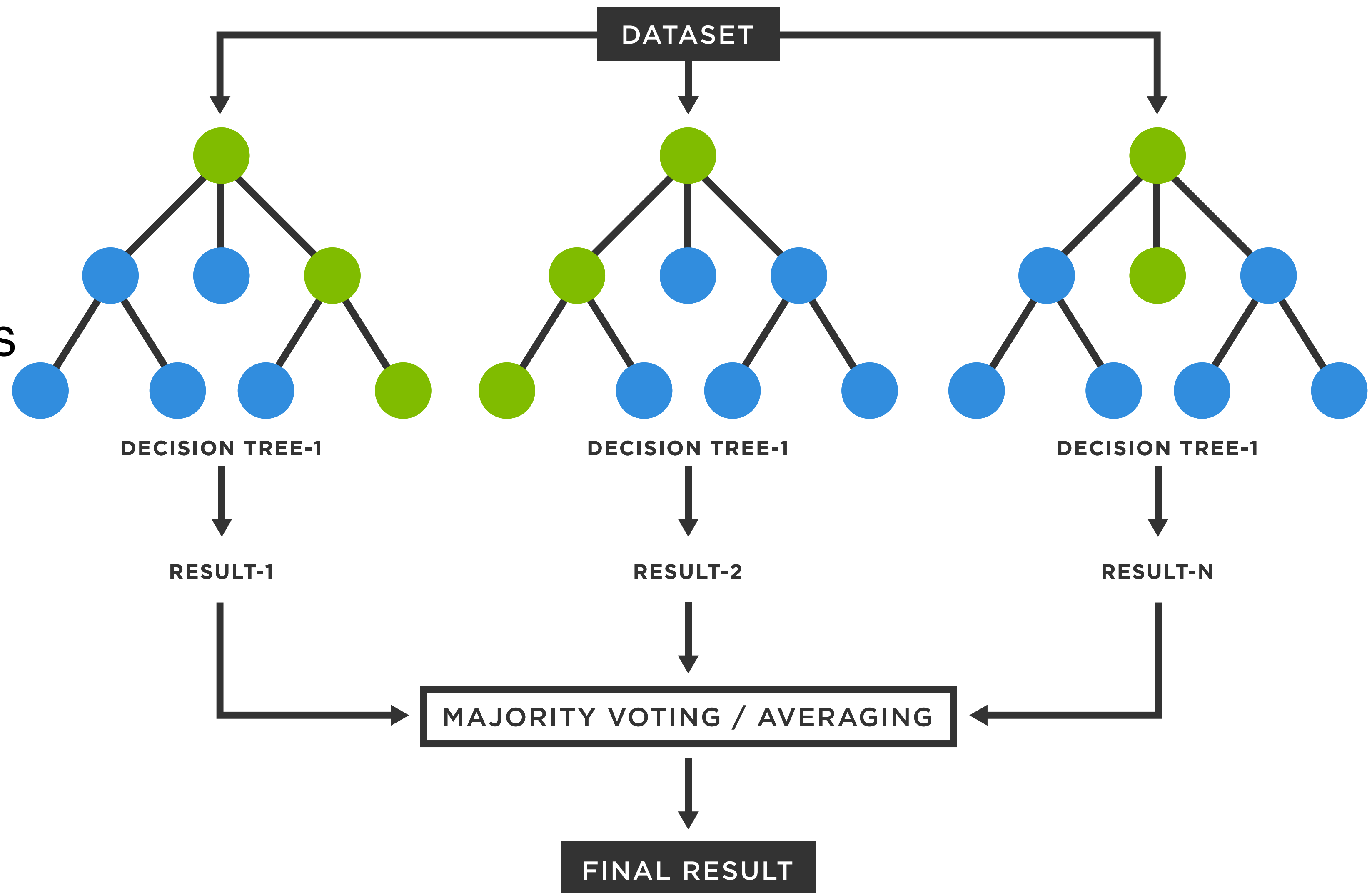
- Many criteria: Stop when
 - Splitting does not reduce the **uncertainty**.
 - Reaches pre-specified size.
 - Every leaf is “pure” (contain only one class)
 - Very prone to overfitting
 - Often resolved by “pruning” trees



Forests

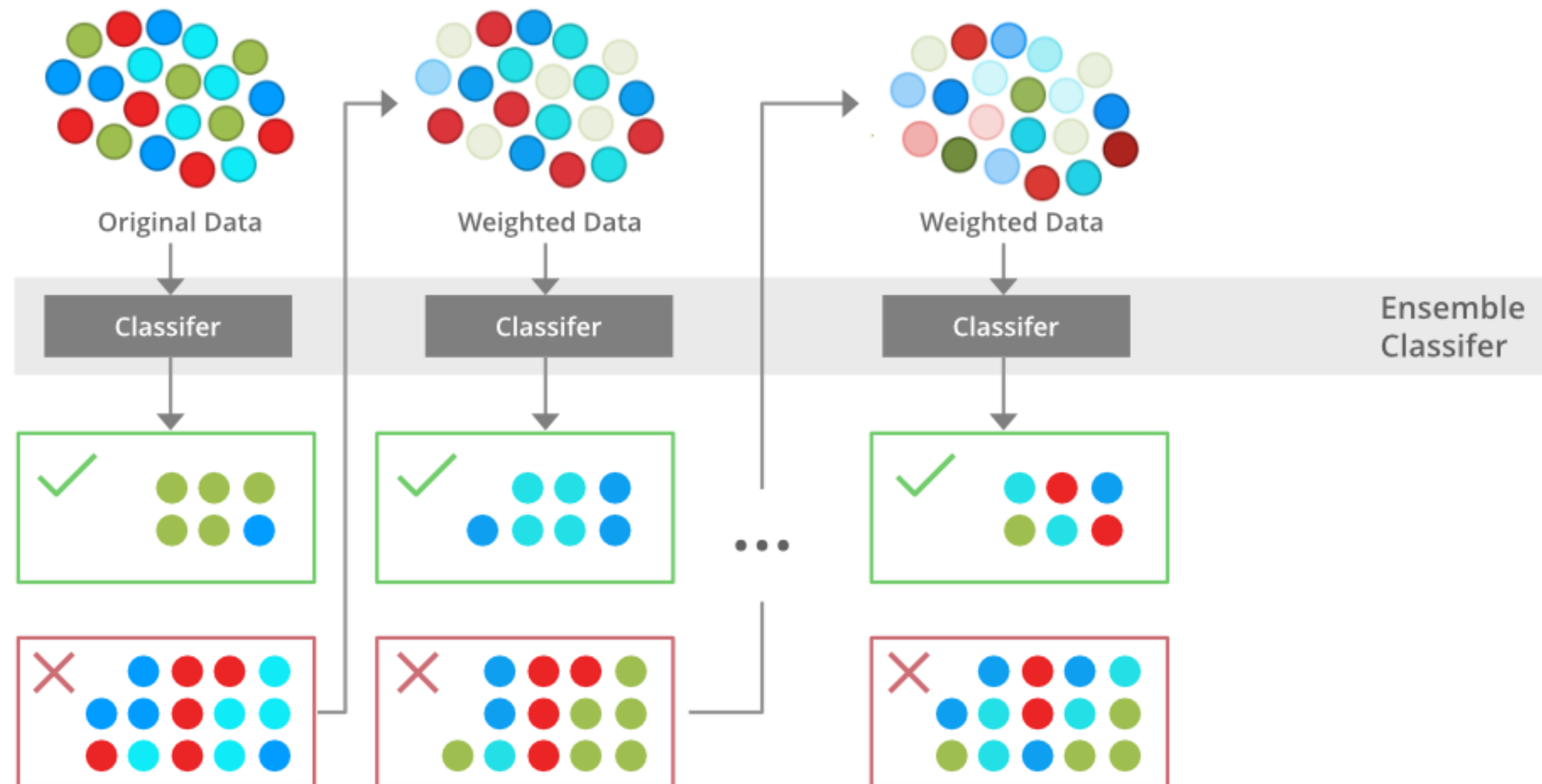
Random Forest

- Split the data (or split the features) to make many trees.
- Aggregate predictions by majority voting or averaging
 - called “bagging”



Boosting

- Sequentially make trees to diversify them.
 - Upweight the wrong classifications / learn to fit the residual



Cheers

- Next up. Dimensionality Reduction