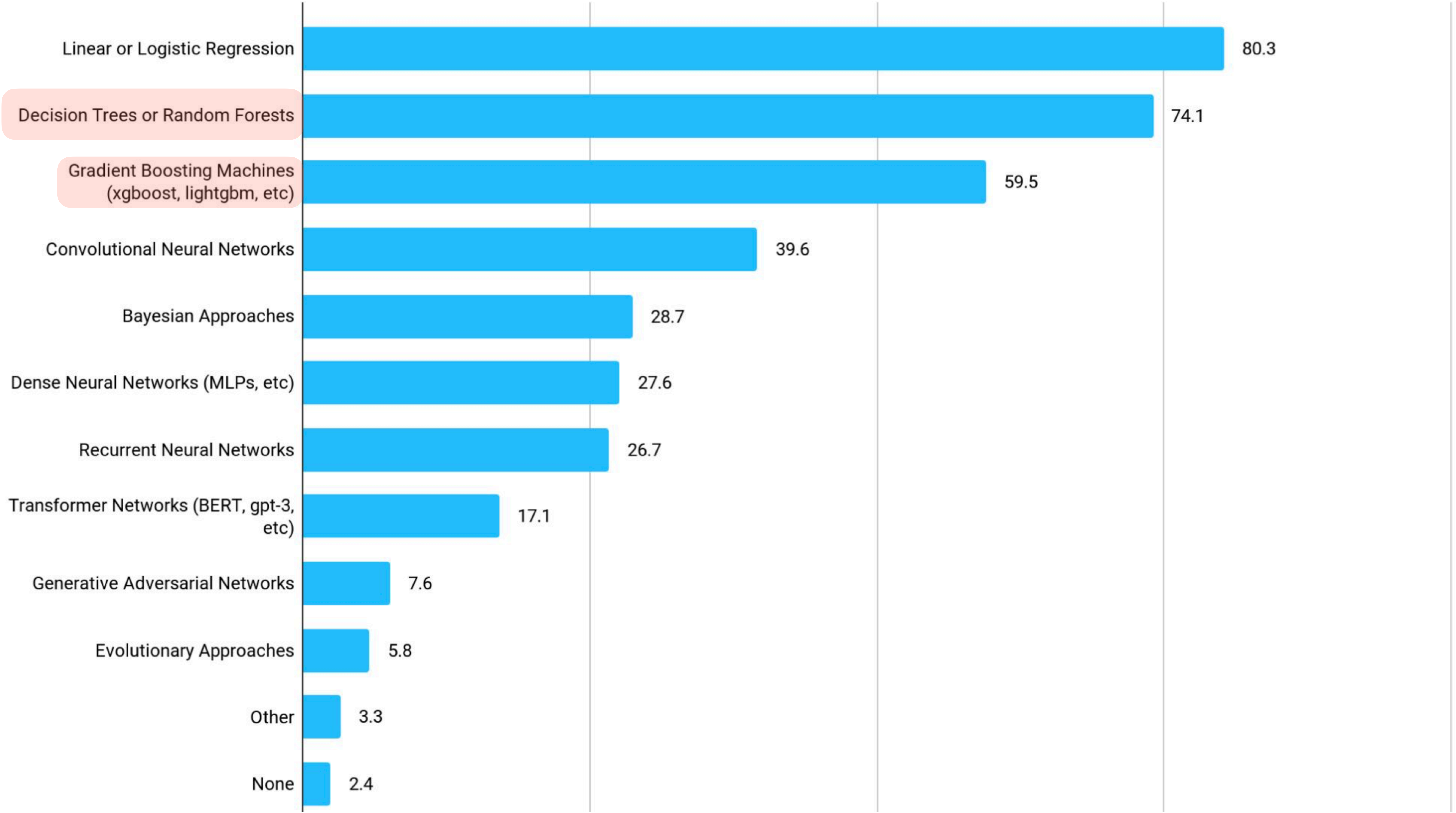
Decision Trees

Motivation

- **Kaggle.** A competition platform for ML and data science
  - People upload data and put bounty to it
  - You solve it

Kaggle Survey (2021)



Historical Bits

Historical bits

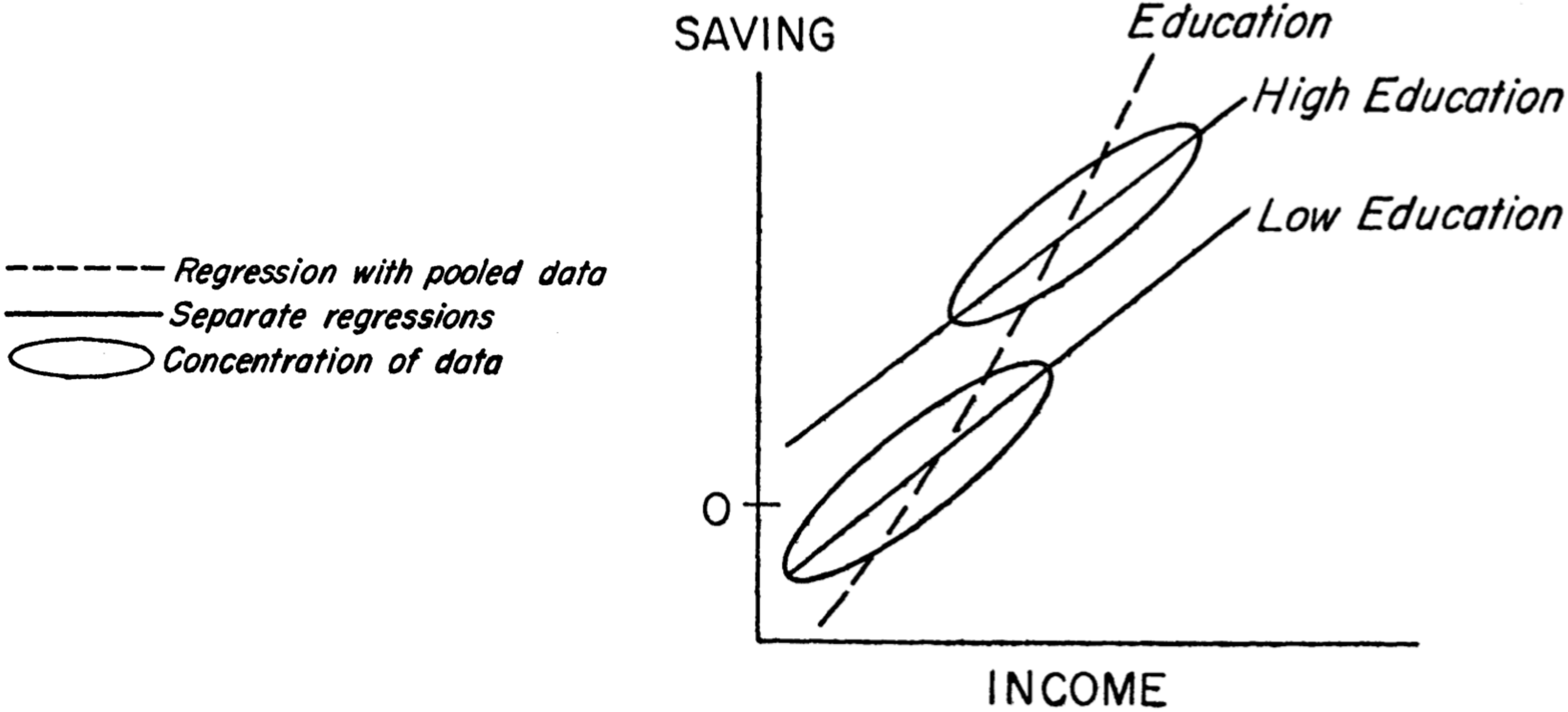
- Use in modern ML traces back to Morgan & Sonquist (1963)
  - Analyzing survey data on income & savings
    - Data included many demographic subgroups
  - Turned out that the trend was highly nonlinear

TABLE 1. SPENDING UNIT INCOME AND THE NUMBER IN THE UNIT WITHIN VARIOUS SUBGROUPS

Group	Spending unit average (1958) income	Number in unit	Number of cases
Nonwhite, did not finish high school	$ 2489	3.3	191
Nonwhite, did finish high school	5005	3.4	67
White, retired, did not finish high school	2217	1.7	272
White, retired, did finish high school	4520	1.7	72
White, nonretired farmers, did not finish high school	3950	3.6	87
White nonretired farmers, did finish high school	6750	3.6	24

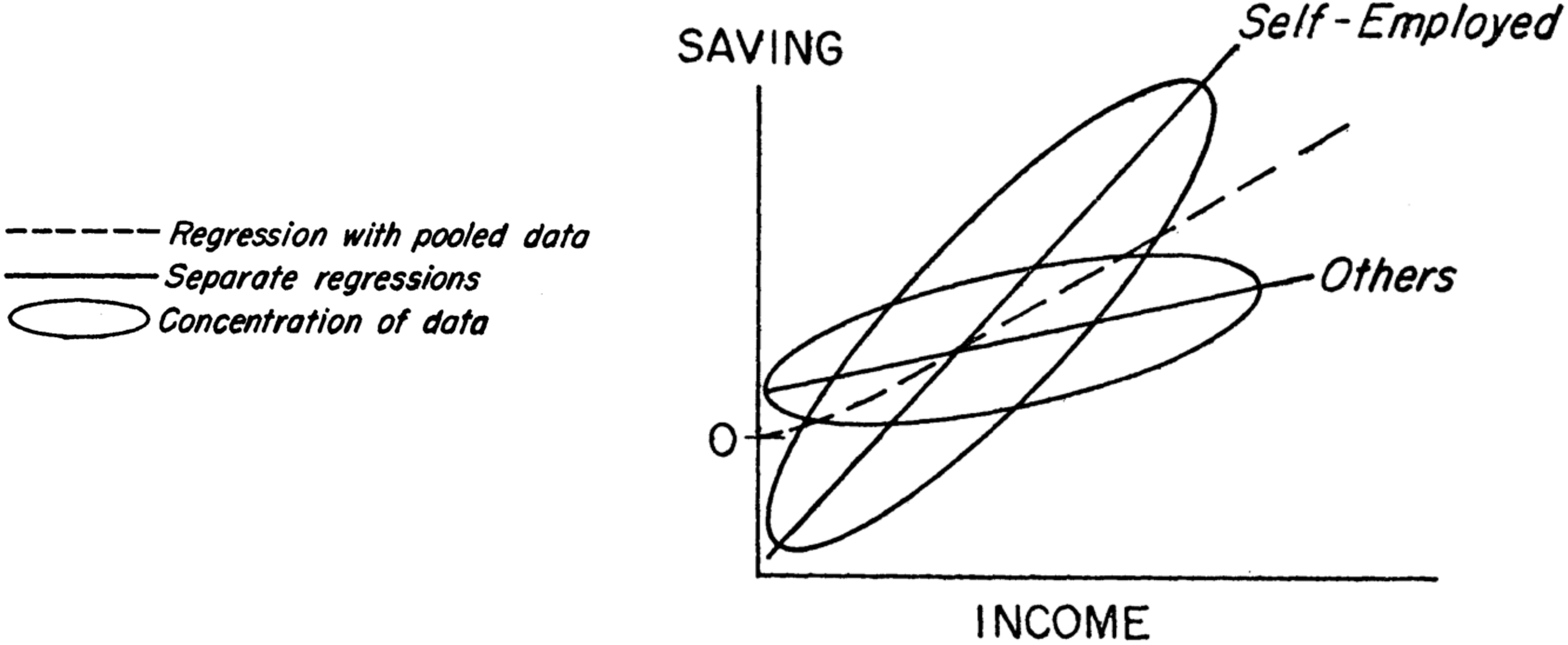
Historical bits

- **Case 1.** Multi-collinearity
  - Correlation between income & education, but no interaction



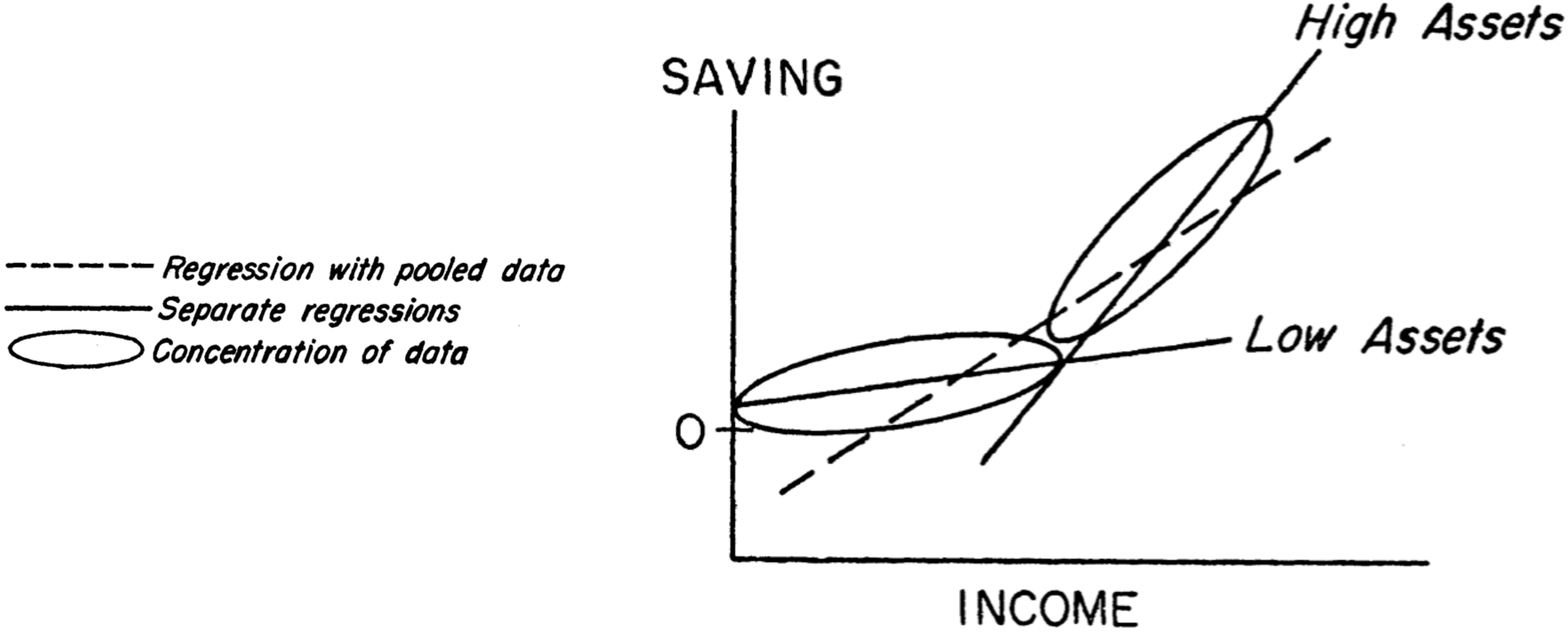
Historical bits

- **Case 2.** Interaction between features
  - No correlation between income & self-employment



Historical bits

- **Case 3.** Both

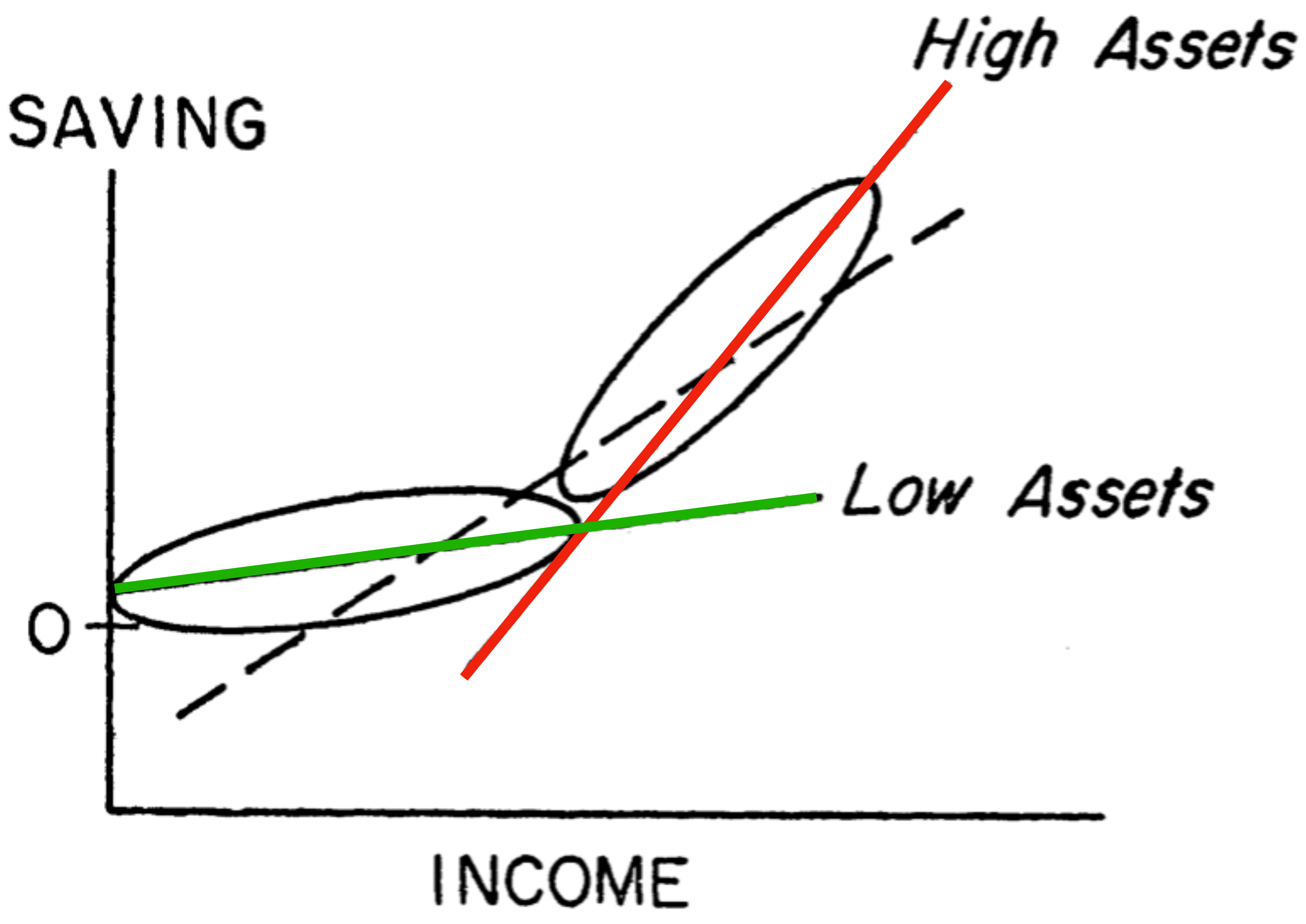


Historical bits

- In each of these examples, having a single linear model doesn't work well

- **Idea.** Take a sequential approach
  - Divide. Partition the data into many subgroups
  - Conquer. Have a simple model for each subgroup (e.g., linear)

- **Example.** High asset?
  - Yes → use curve 1
  - No → use curve 2



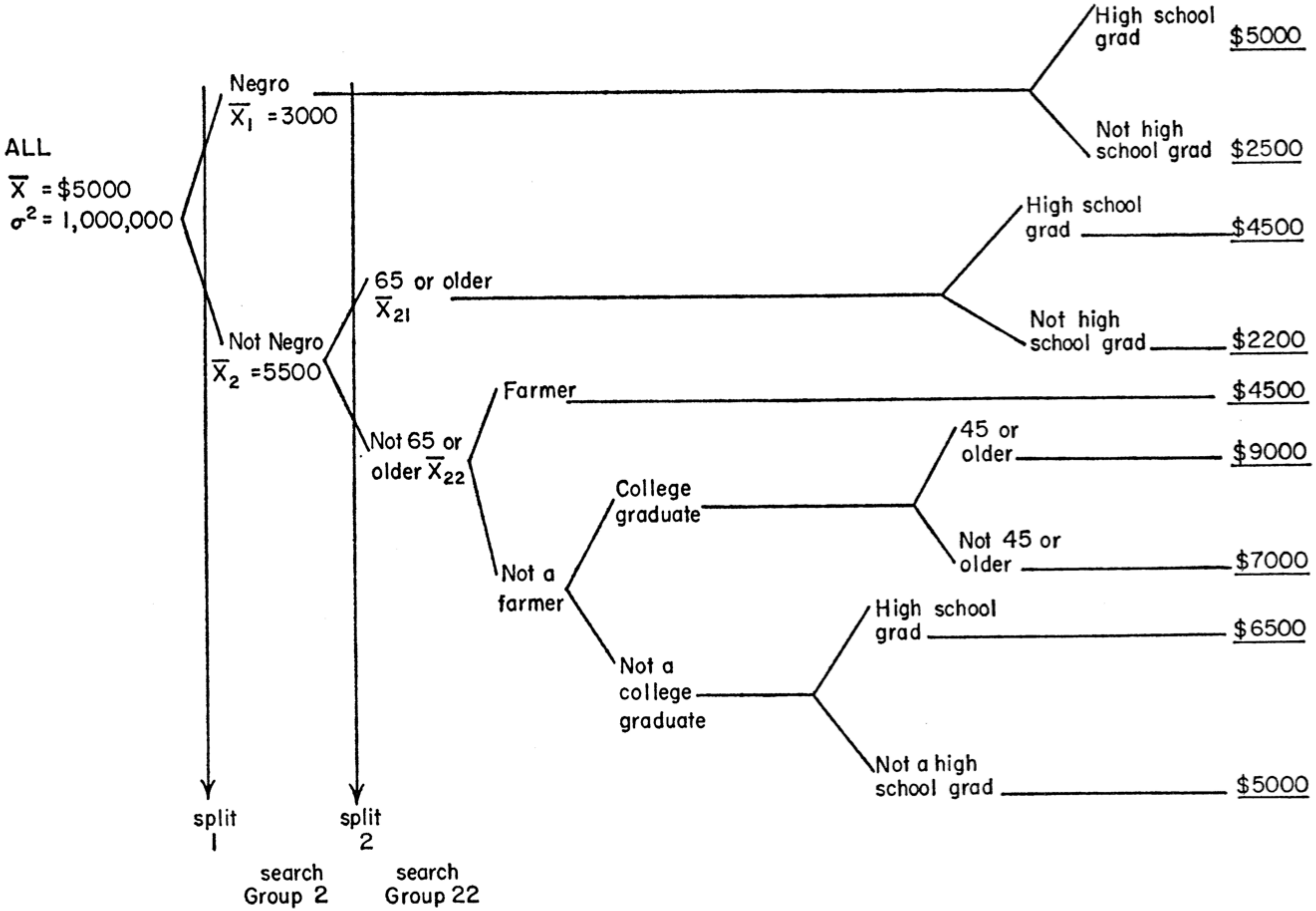
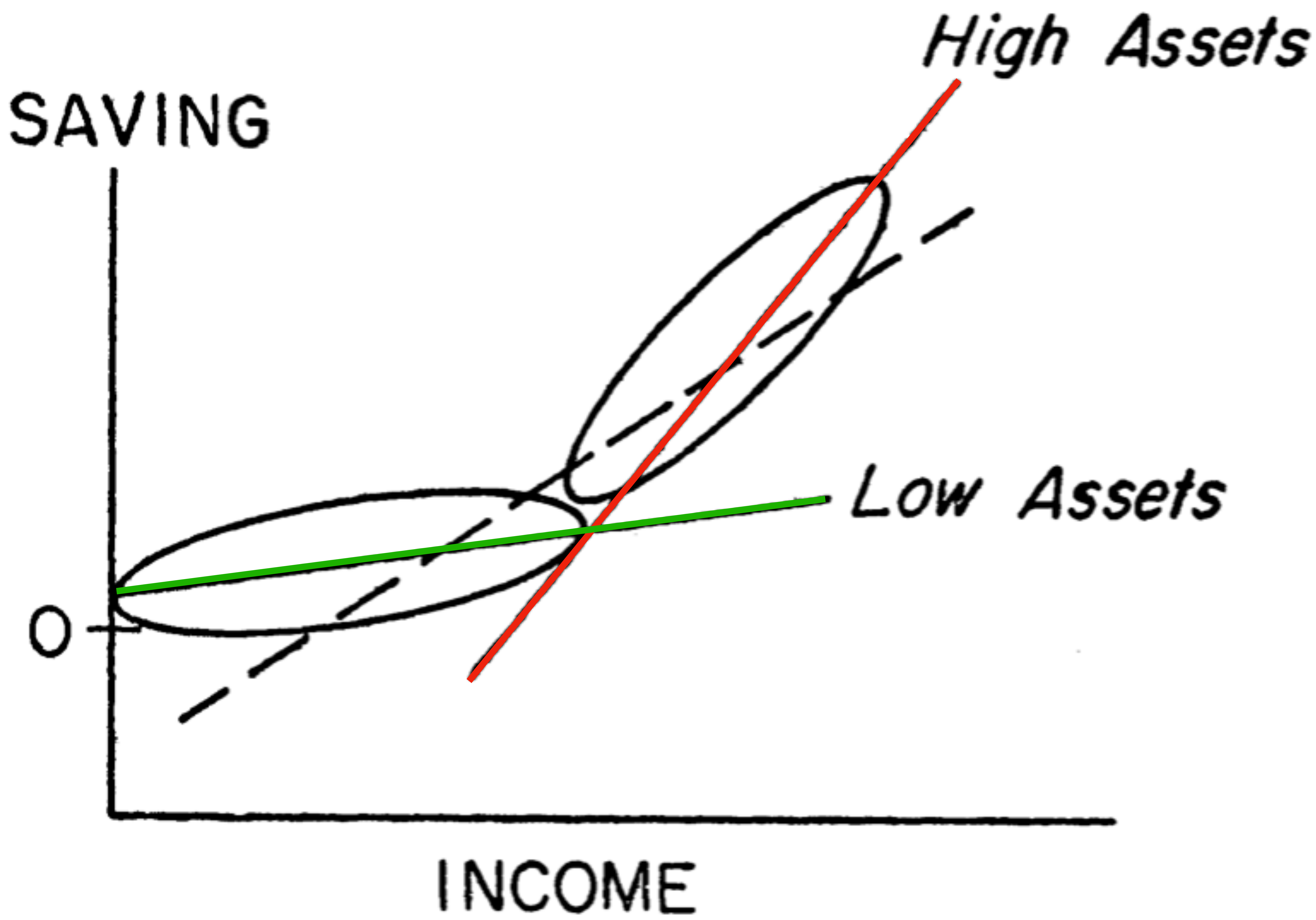


CHART II. Annual Earnings.

Key question

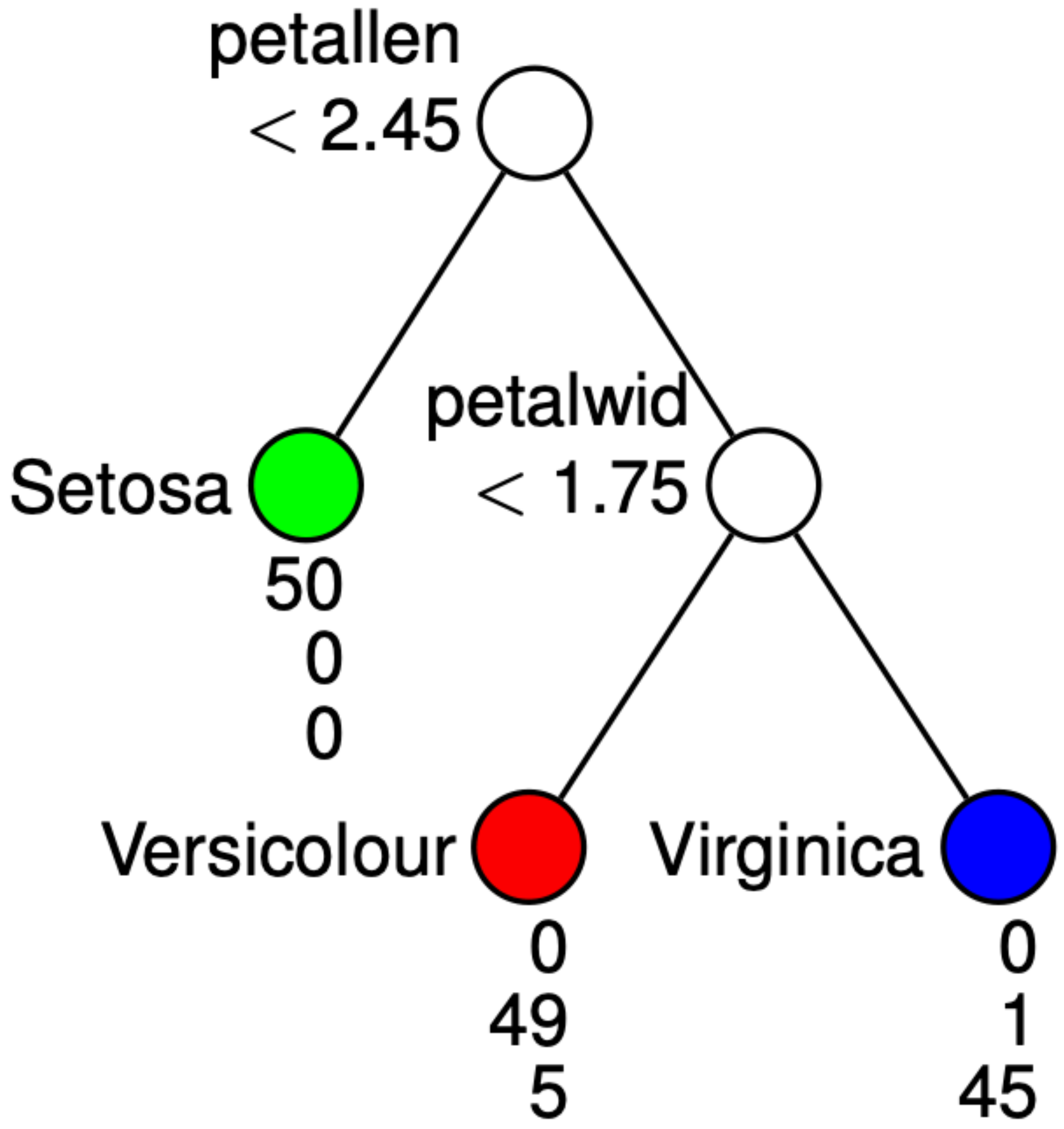
- How do we know if a subgroup needs division?
  - If we know, exactly how do we divide?



Decision Trees

Overview

- Basically a nested if-else statement
- A binary tree which recursively partitions and refines the input space
  - **Leaf.** Associated with some label $\hat{y}$
    - If discrete, classification
    - If continuous, regression
  - **Tree.** Associated with some splitting rule $g : \mathscr{X} \to \{0,1\}$



Inference

- Given $\mathbf{x}$, recurse down the tree until a leaf is reached
  - Then, output the label of the leaf

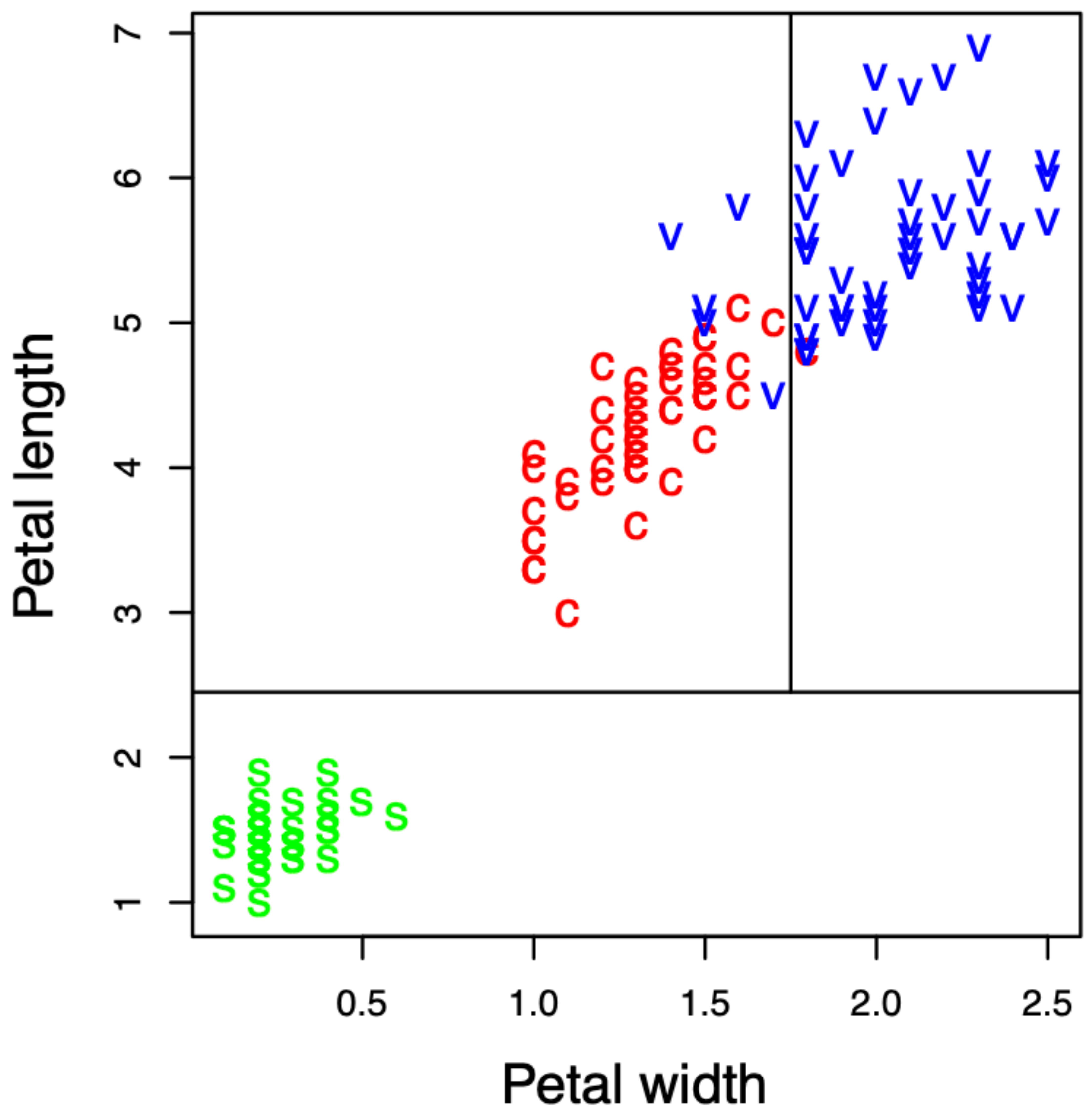
```
while(true):
    if(node == leaf): output label(node)
    else:
        if(condition == true): node = right_child(node)
        else: node = left_child(node)
```

Inference

- When $\mathscr{X} = \mathbb{R}^d$, it is typical to consider only the axis-aligned splits

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

  - Computationally efficient
    - Single index lookup
  - Human-interpretable decisions



Training

- Constructing a decision tree requires specifying three elements
  - Prediction rule
  - Stopping rule
  - Splitting rule

until all leaf node is stopped:

visit a leaf node

if(stopping_rule(node) = True):

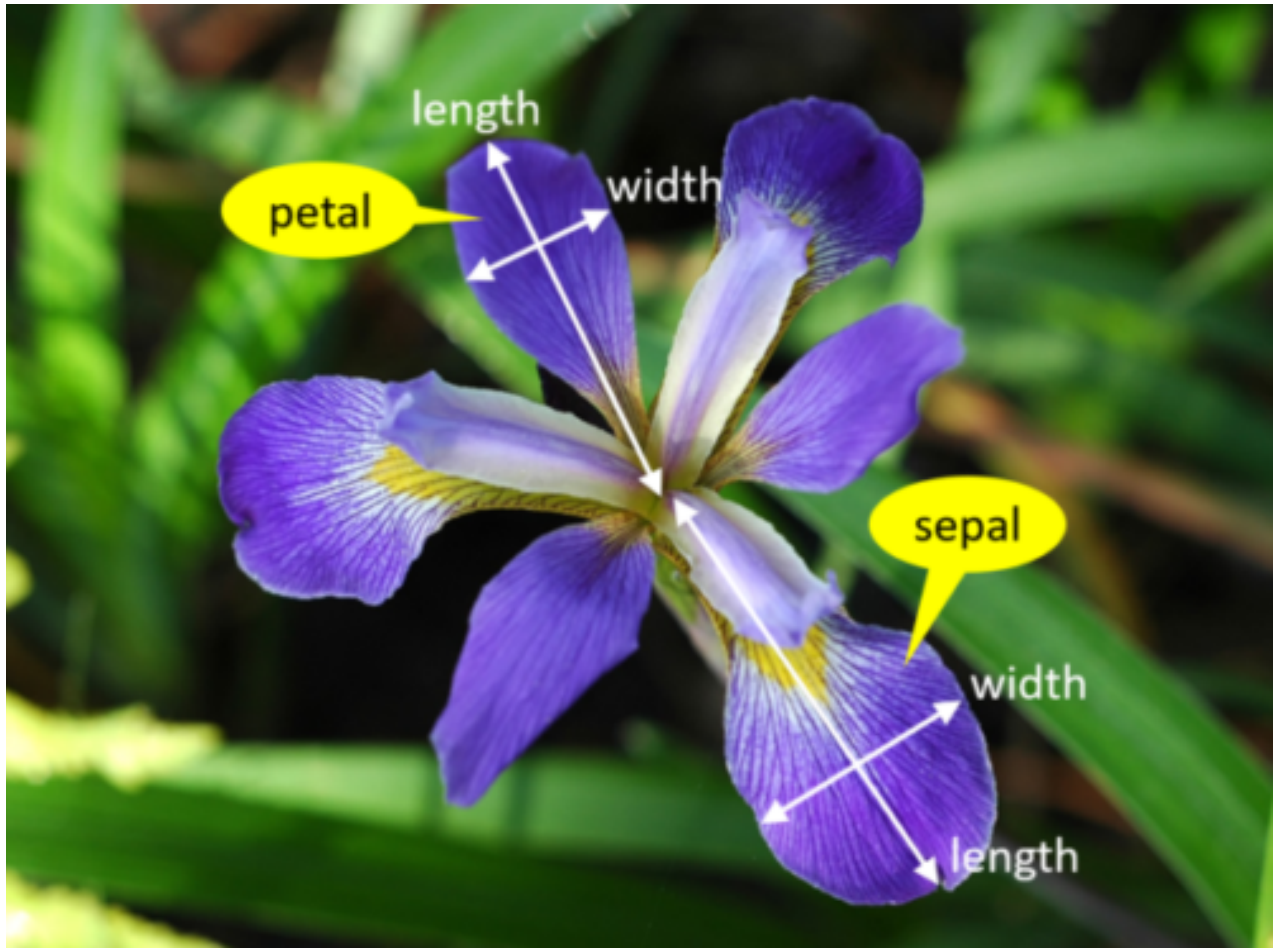
apply prediction rule to label the node
stop the node

else:

split the node, using the splitting rule

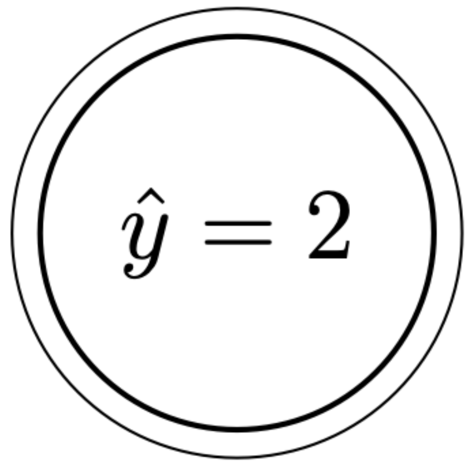
Example: Iris Classification

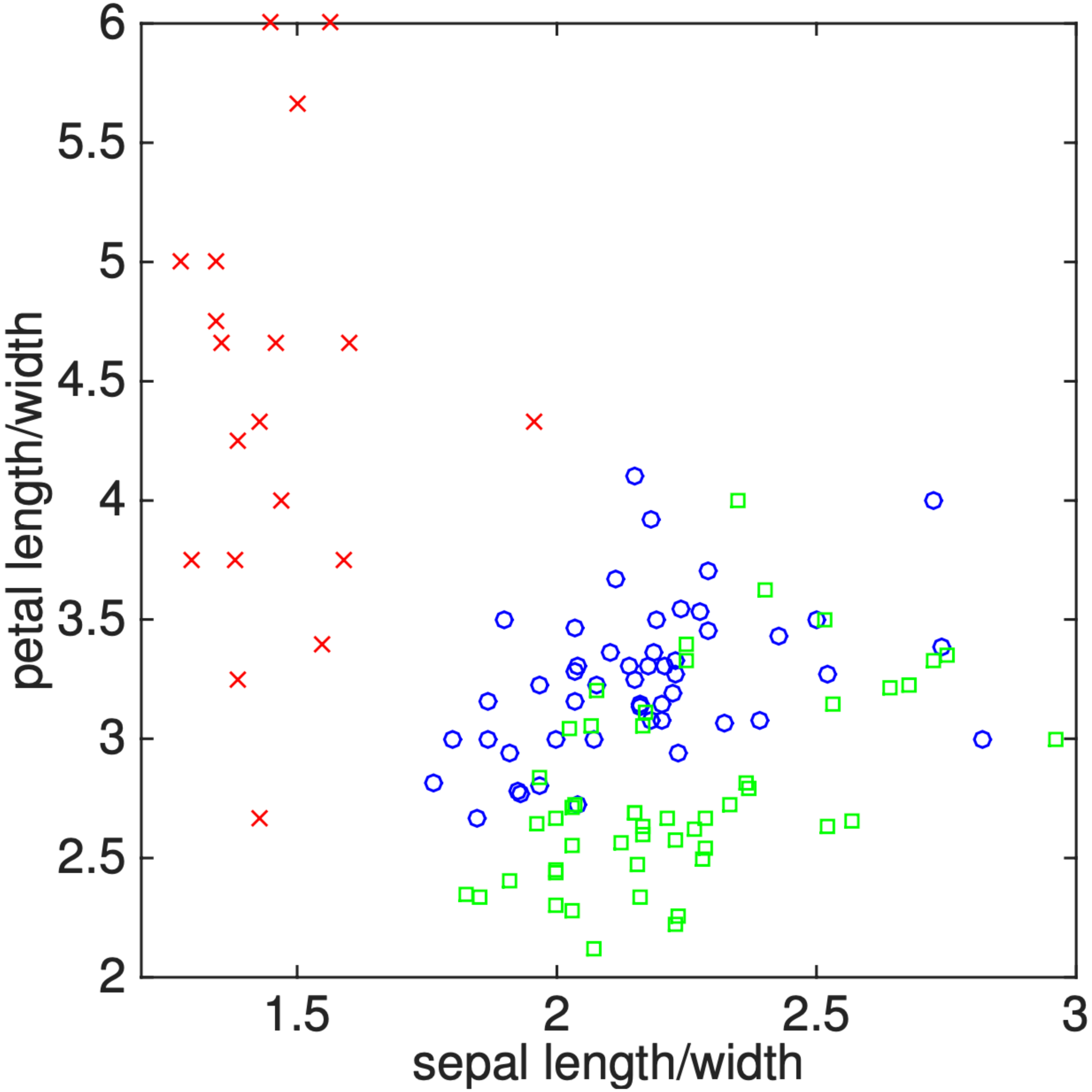
- For example, consider an iris classification task
  - Input features $\mathscr{X} = \mathbb{R}^2$
    - x_1: length-width ratio of sepal
    - x_2: length-width ratio of petal
  - Output labels $\mathscr{Y} = \{1, 2, 3\}$



Example: Iris Classification

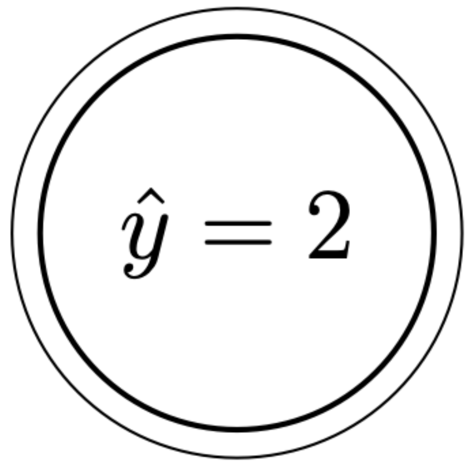
- First, construct a single leaf node, using a prediction rule that applies to the whole set

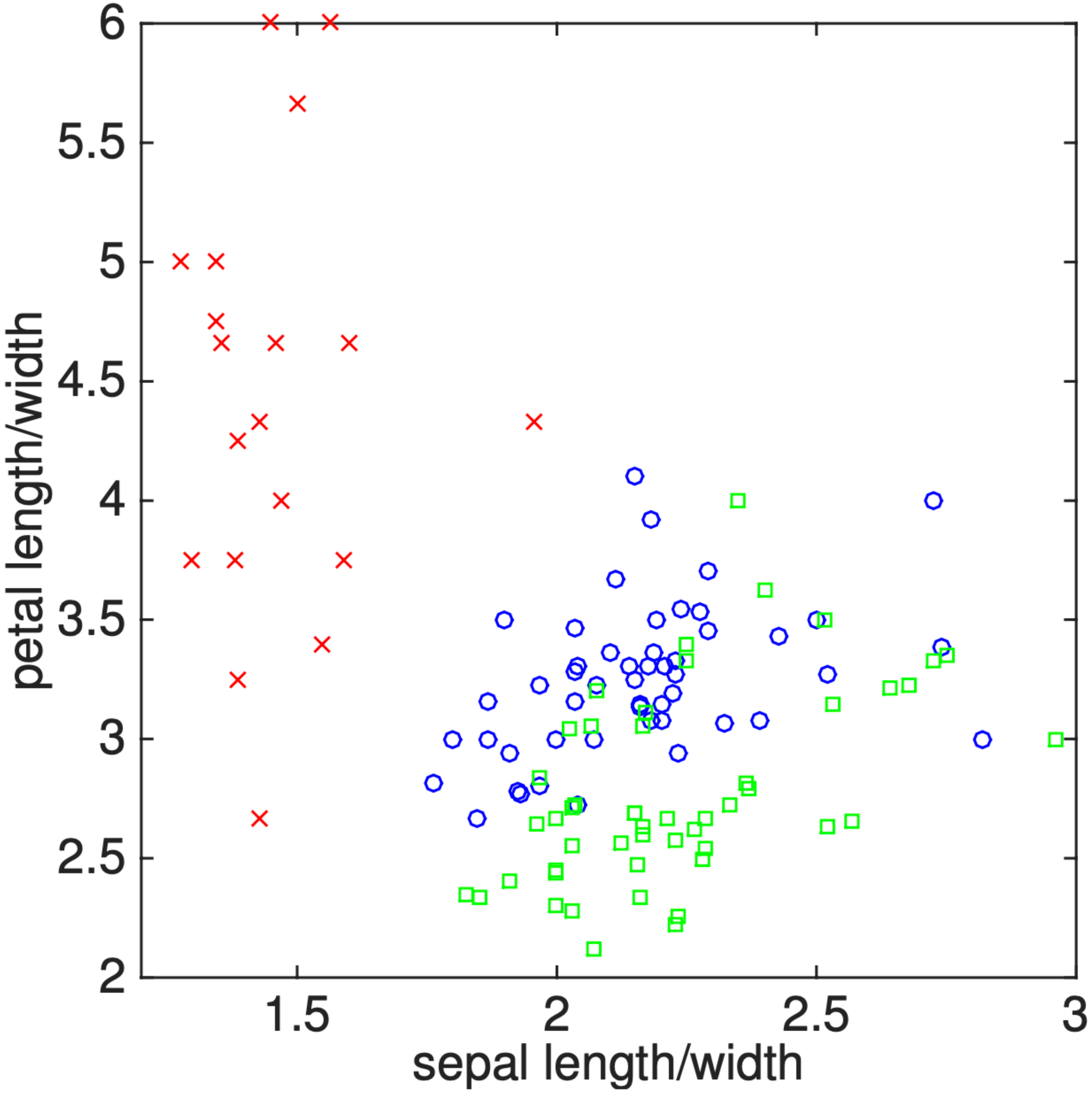




Example: Iris Classification

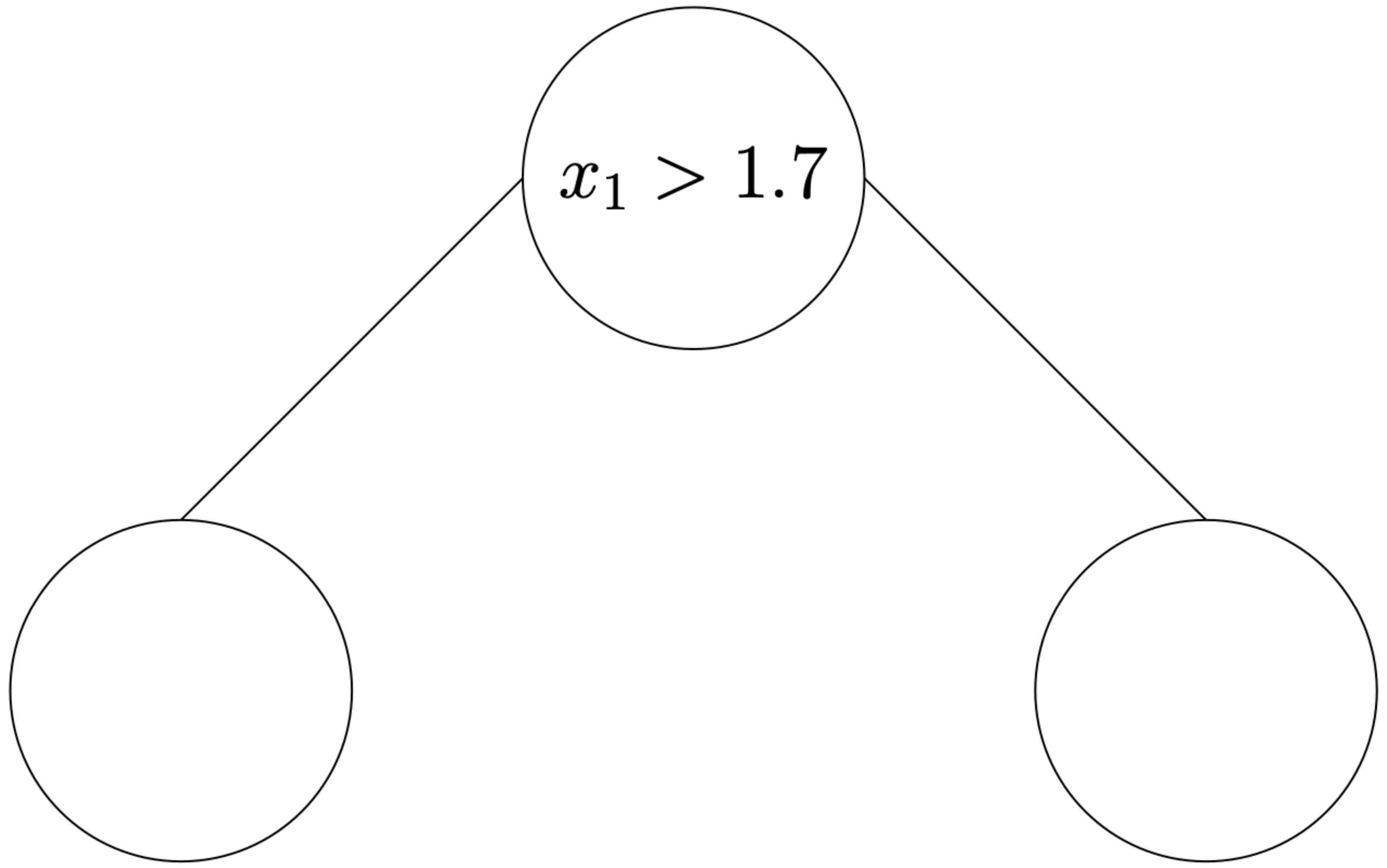
- See if the stopping rule is met
  - Very unhomogeneous; continue

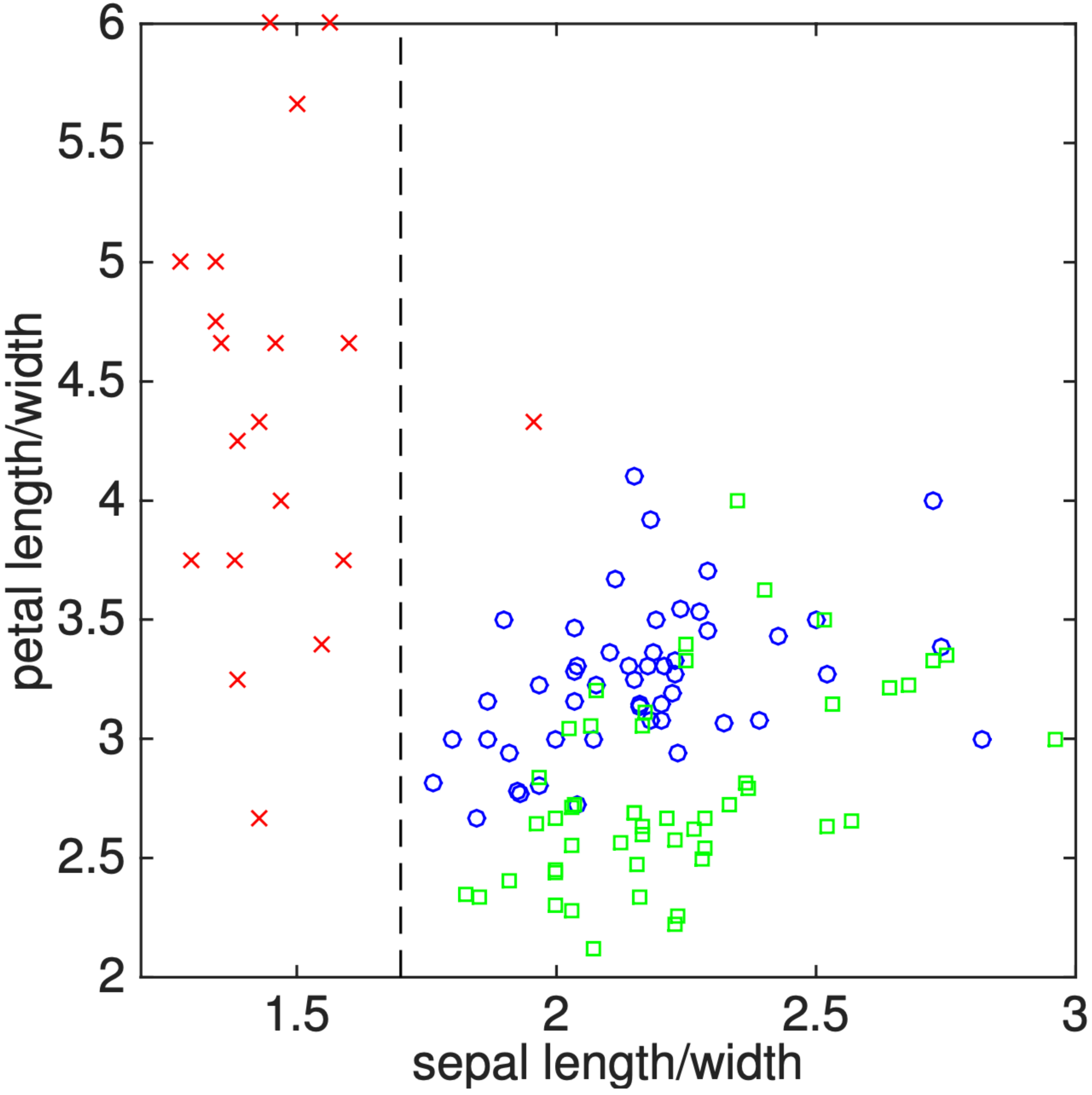




Example: Iris Classification

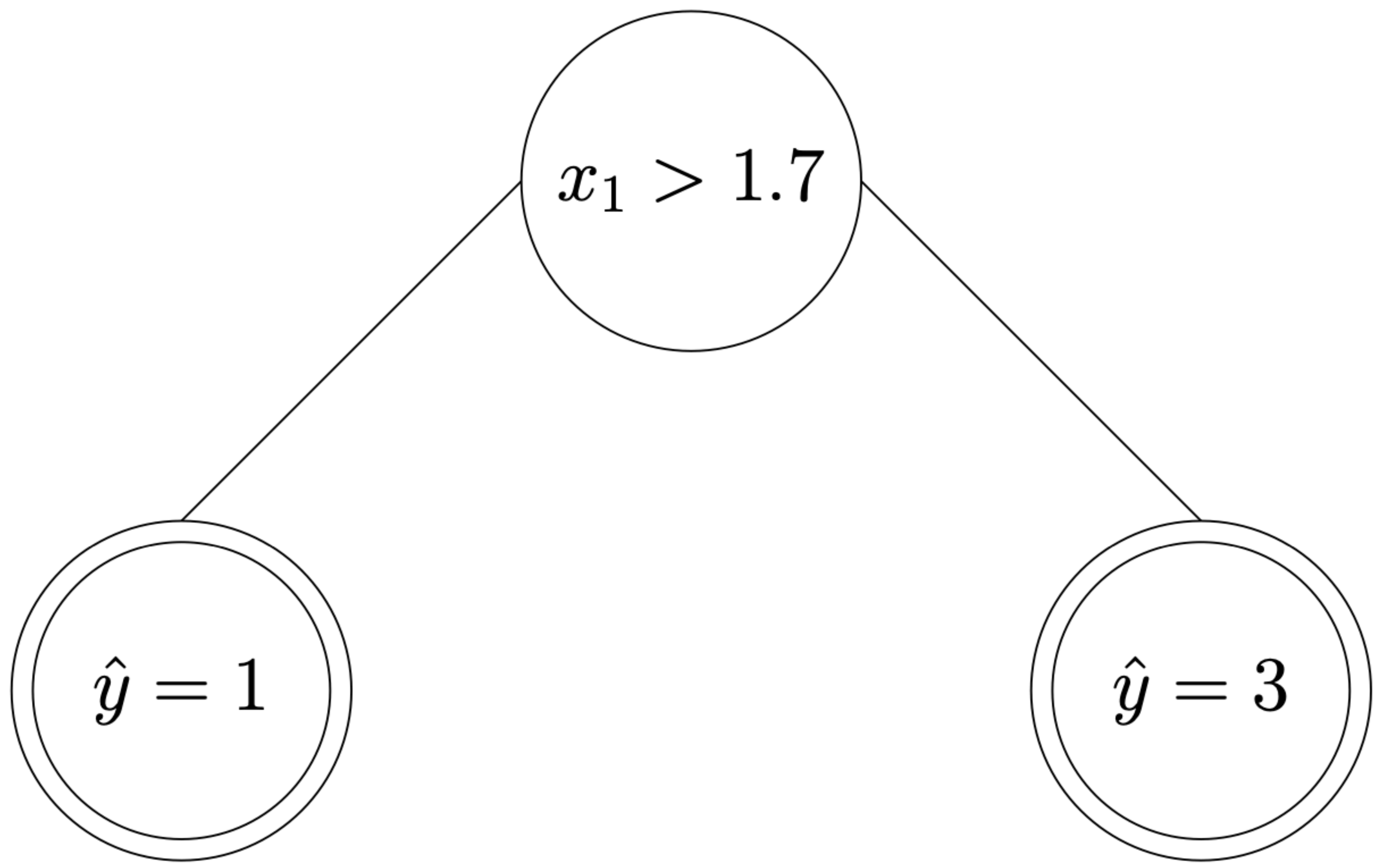
- According to the splitting rule, split the leaf to partition the input space for the node

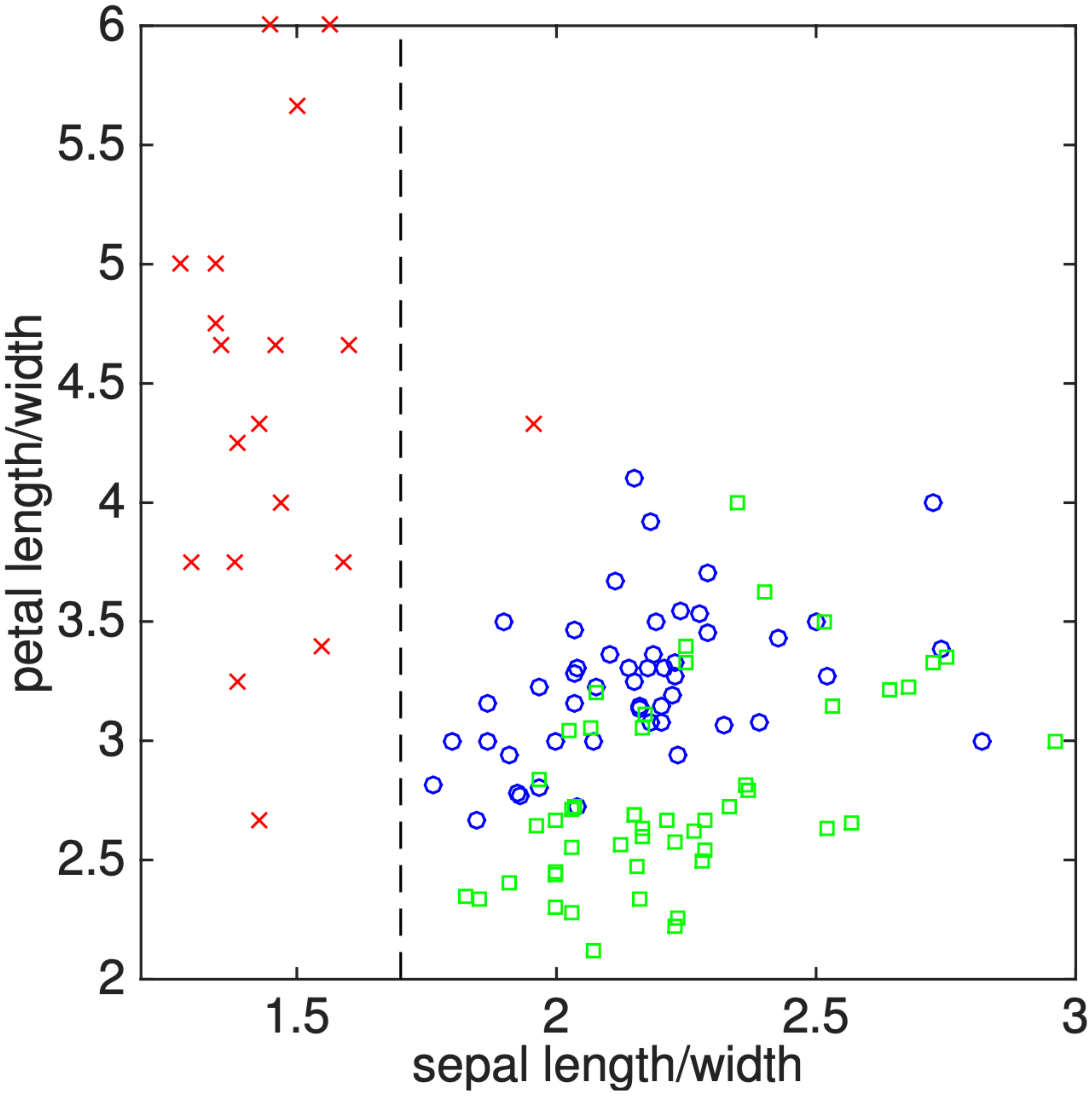




Example: Iris Classification

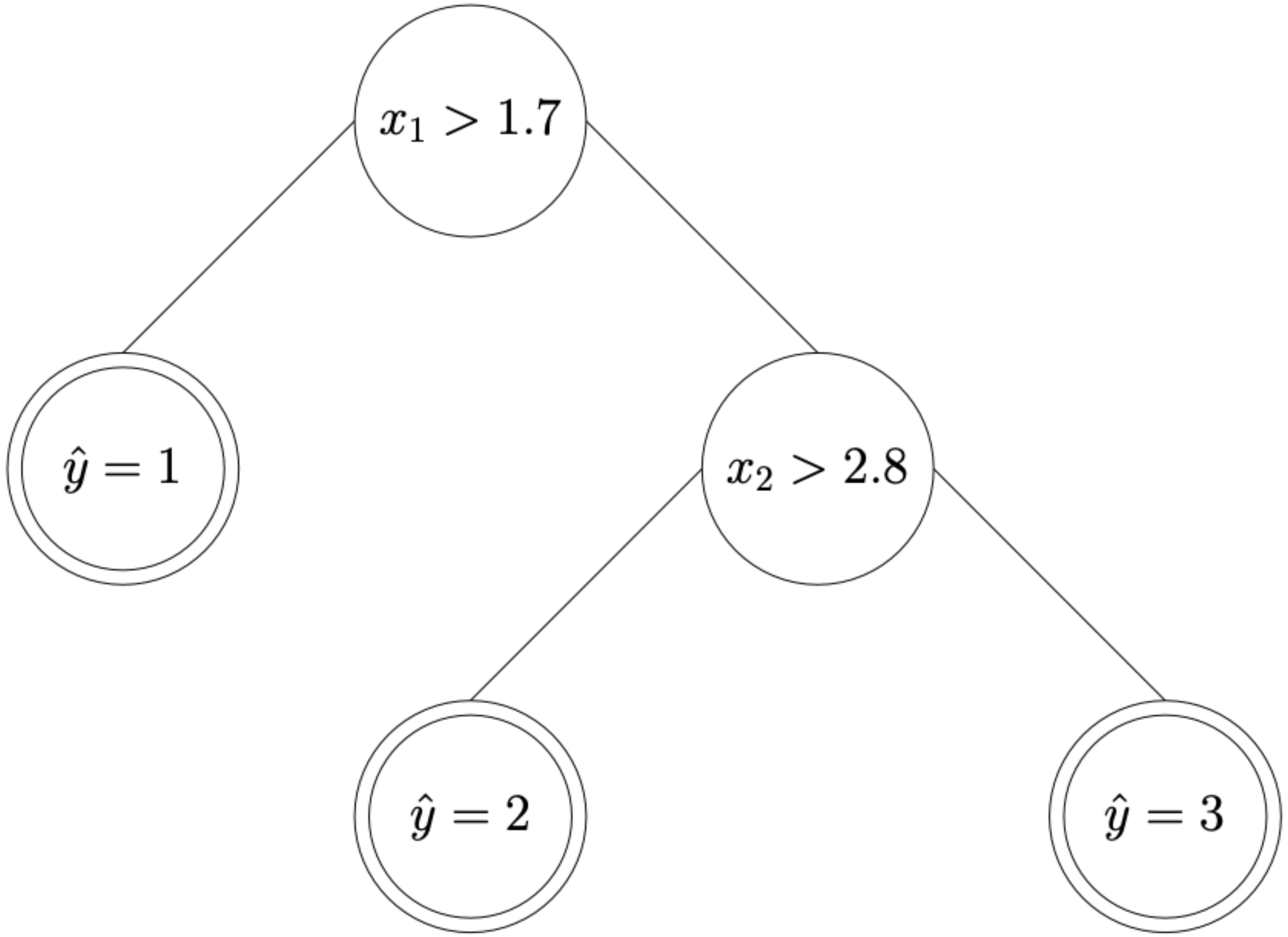
- According to the prediction rule, determine the prediction of new leaf nodes

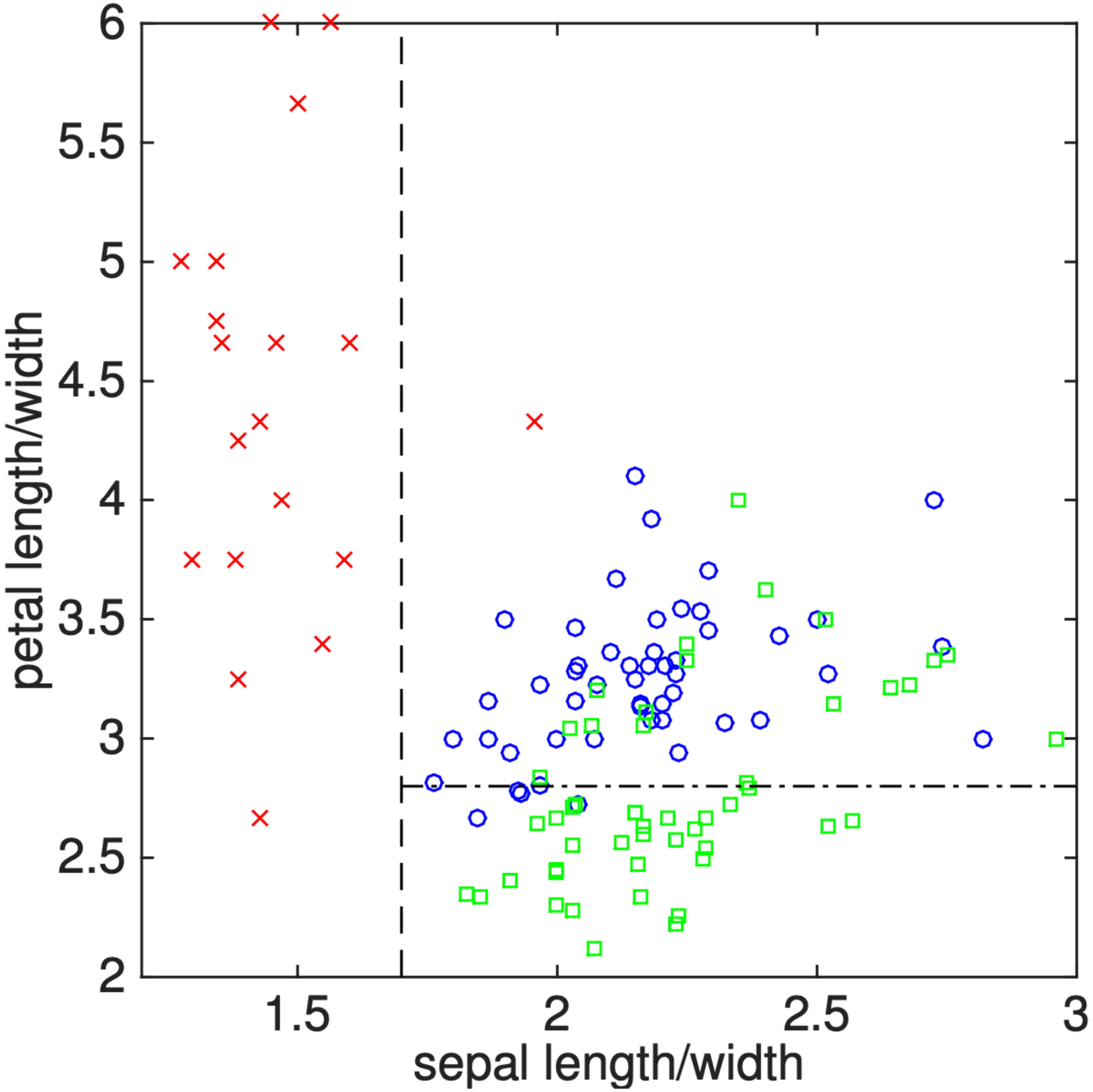




Example: Iris Classification

- Continue until some stopping rule is satisfied for all leaf nodes



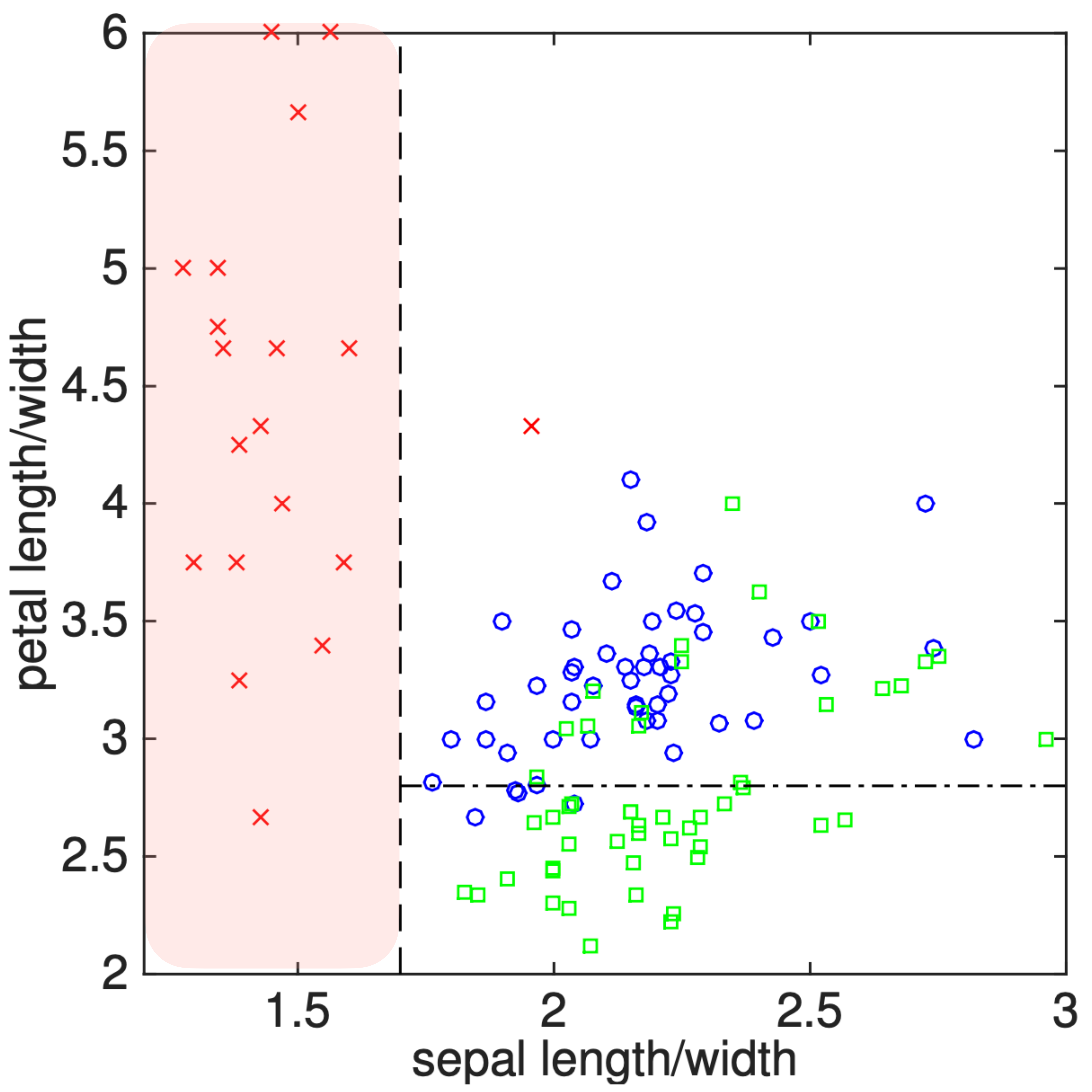


Elements

Training: Prediction Rule

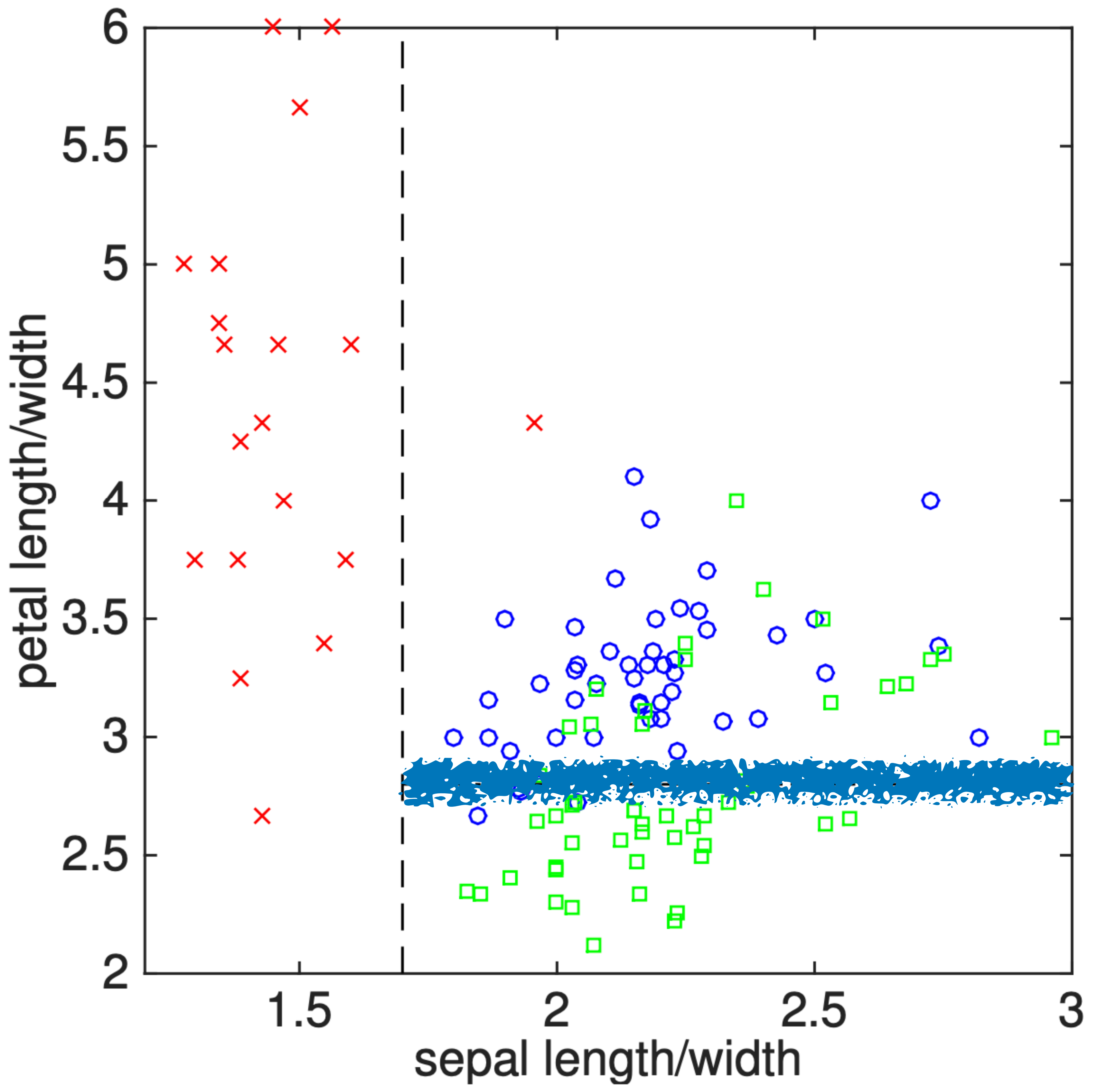
- Generating a label for a partitioned set

- Typically very simple
  - Classification. Majority voting
  - Regression. Average, Median, …



Training: Splitting Rule

- Generating how to partition a set
  - Which axis?
  - Which line?



Training: Splitting Rule

- **Idea.** Minimize some notion of uncertainty (a.k.a. impurities) after partitioning the set

- In other words, by dividing some set S into S_1, S_2, we want to solve:

$$\min_{S_1,S_2:\ S_1 \cup S_2 = S, S_1 \cap S_2 = \phi} \Big(|S_1| \cdot u(S_1) + |S_2| \cdot u(S_2) \Big)$$

  - Here, $u(\cdot)$ is some measure of uncertainty

Training: Splitting Rule

Example (Binary Classification)

- Suppose that we are given a set S, with $p|S|$ samples labeled as $+1$
  - Classification Error

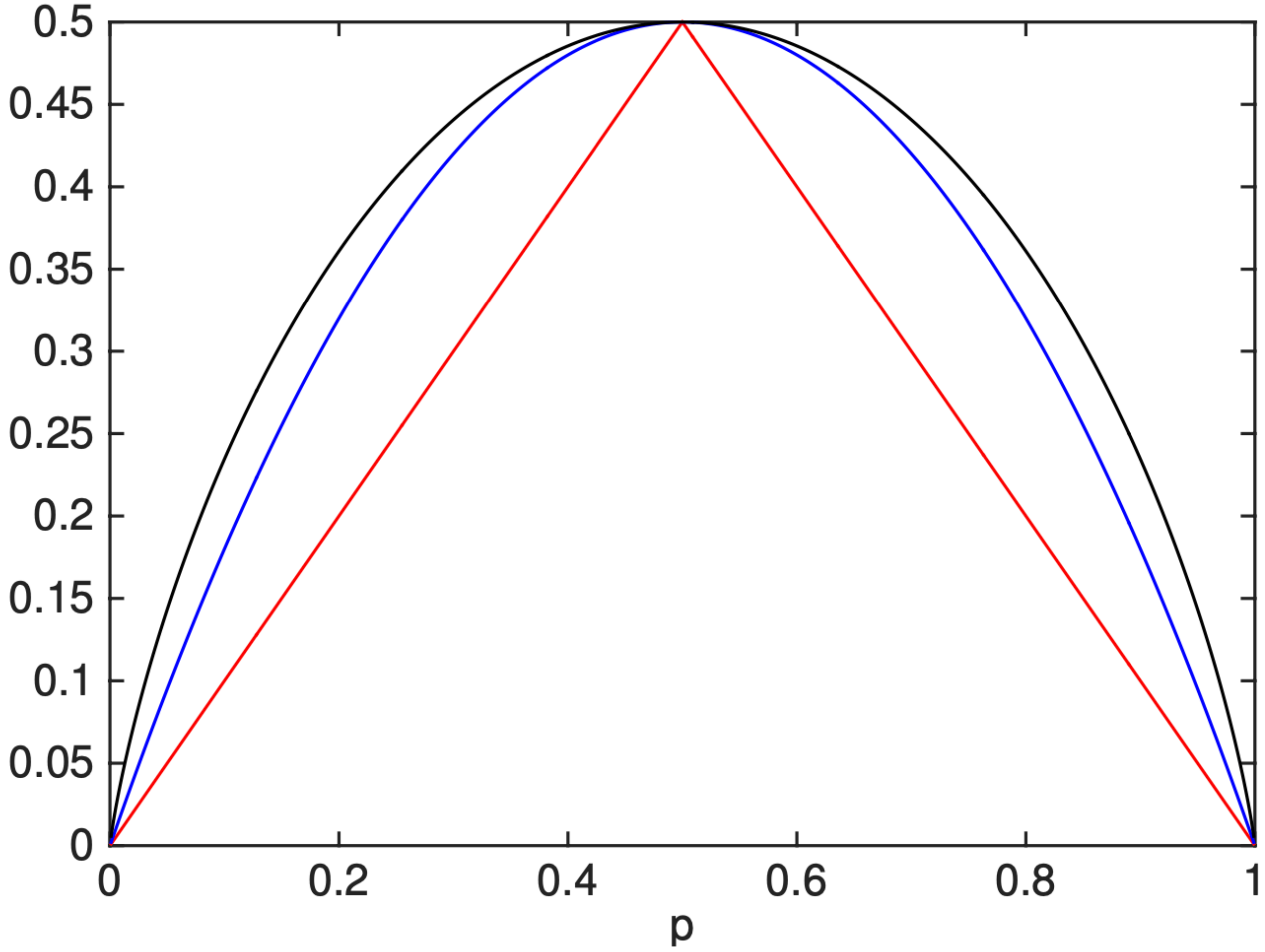
$$u(S) = \min\{p, 1-p\}$$

  - 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)

Training: Splitting Rule

Example (Regression)

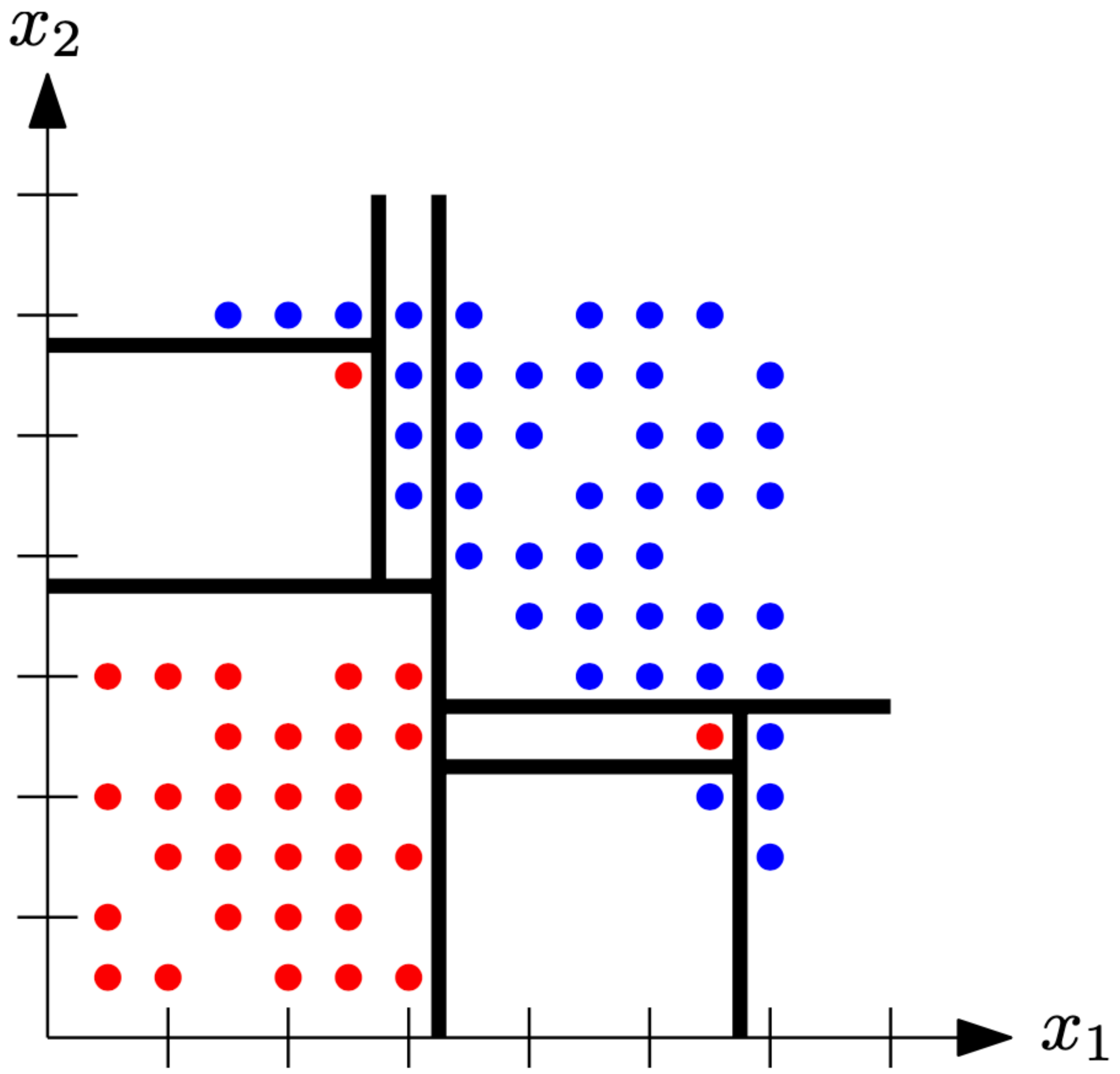
- We can simply use variance
    - the minimum mean squared error
    - i.e., the ℓ^2 error of the mean

- Similarly, we can use the minimum mean absolute error, …

Training: Stopping Rule

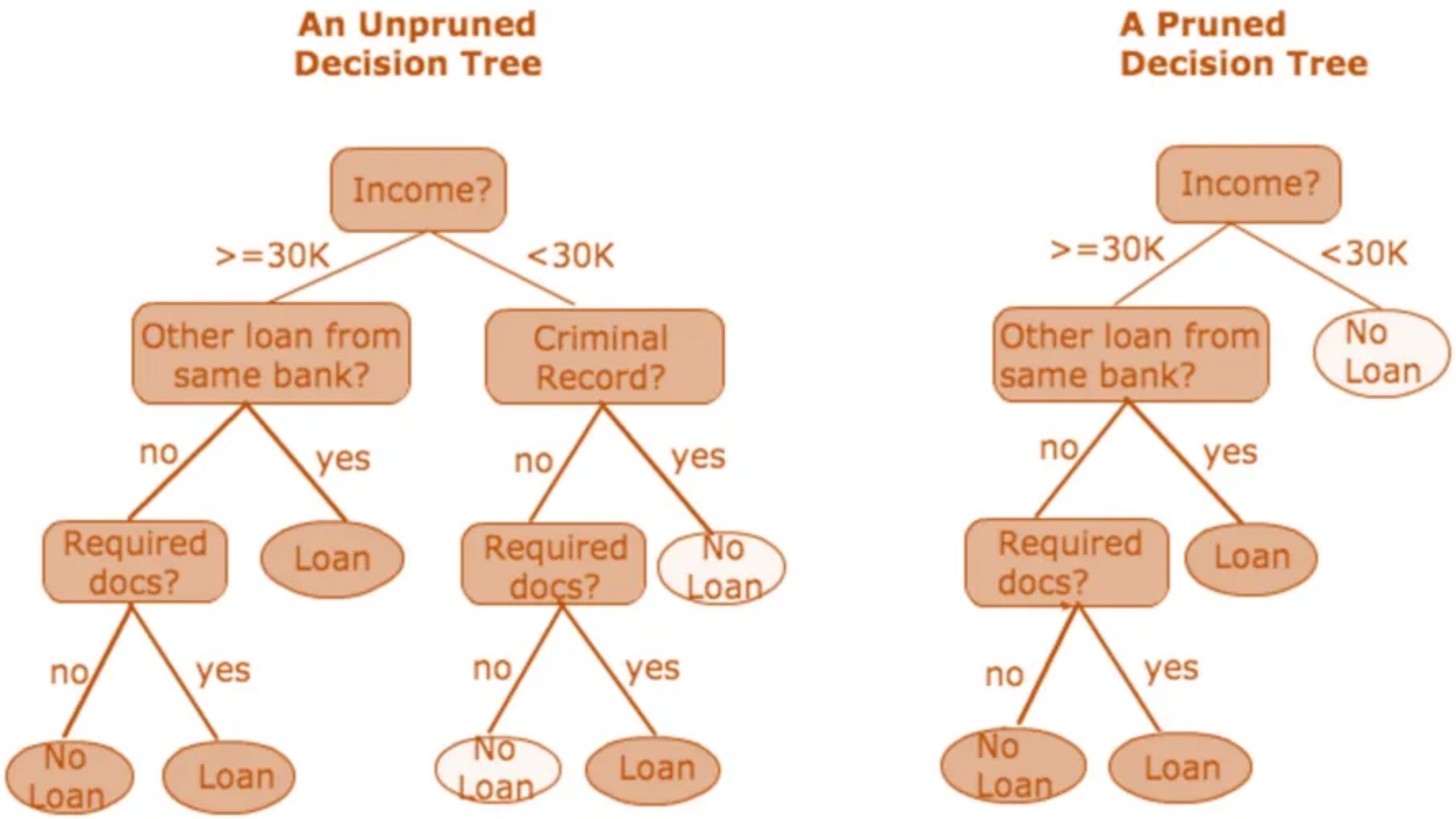
- Determining when to stop growing a tree

- Many criteria:
  - If splitting does not reduce the uncertainty
  - Reaches some pre-specified size of the tree
  - Every leaf is "pure"
    - Very prone to overfitting



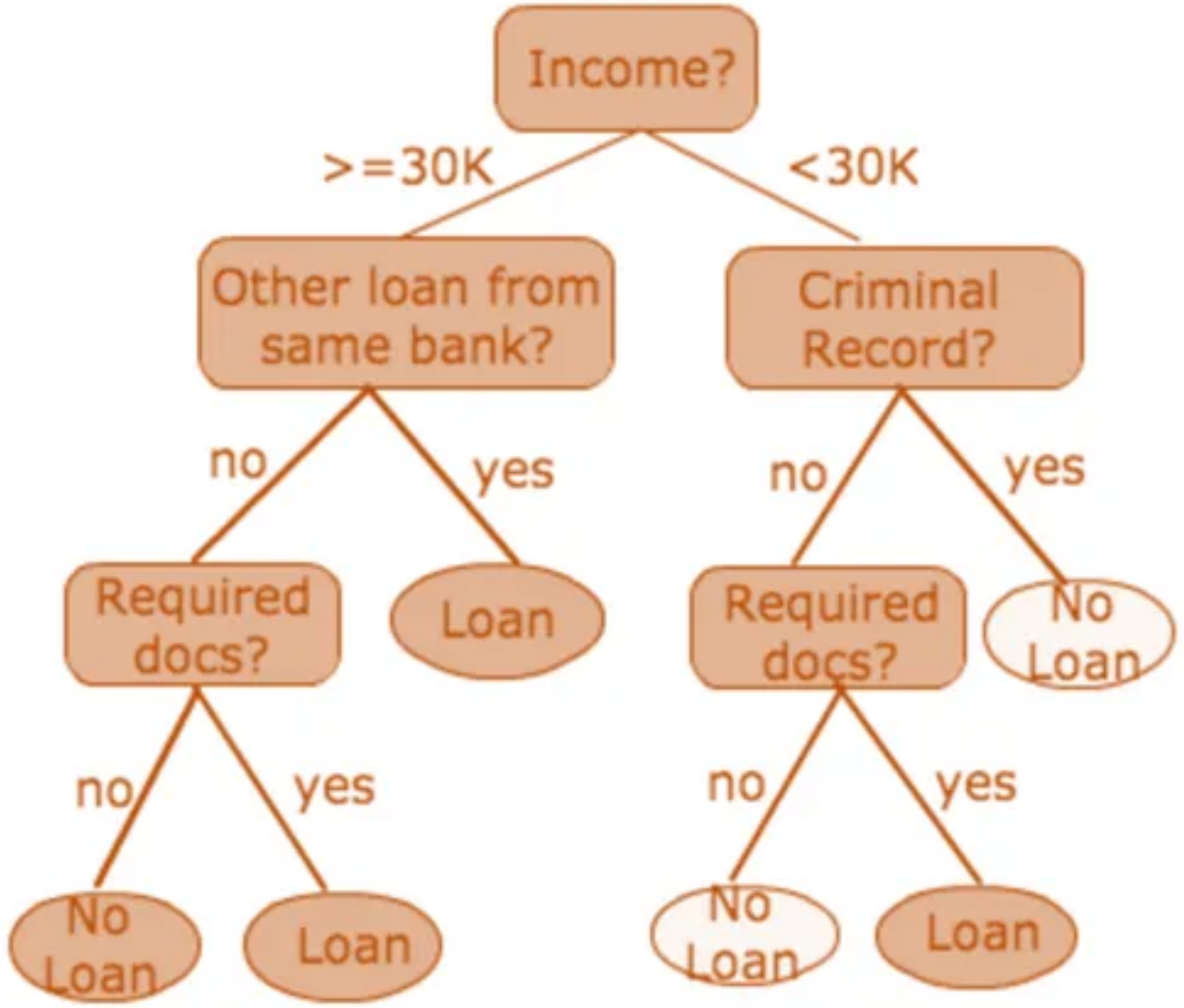
Pruning

- It is typical to **prune** the tree after growing
  - i.e., remove unnecessary split after training



Pruning

- **Algorithm.**
  - Pick a bottom-level split
  - Remove it
    - If the validation error is improved, leave it pruned
    - Else, restore the subtree
  - Repeat

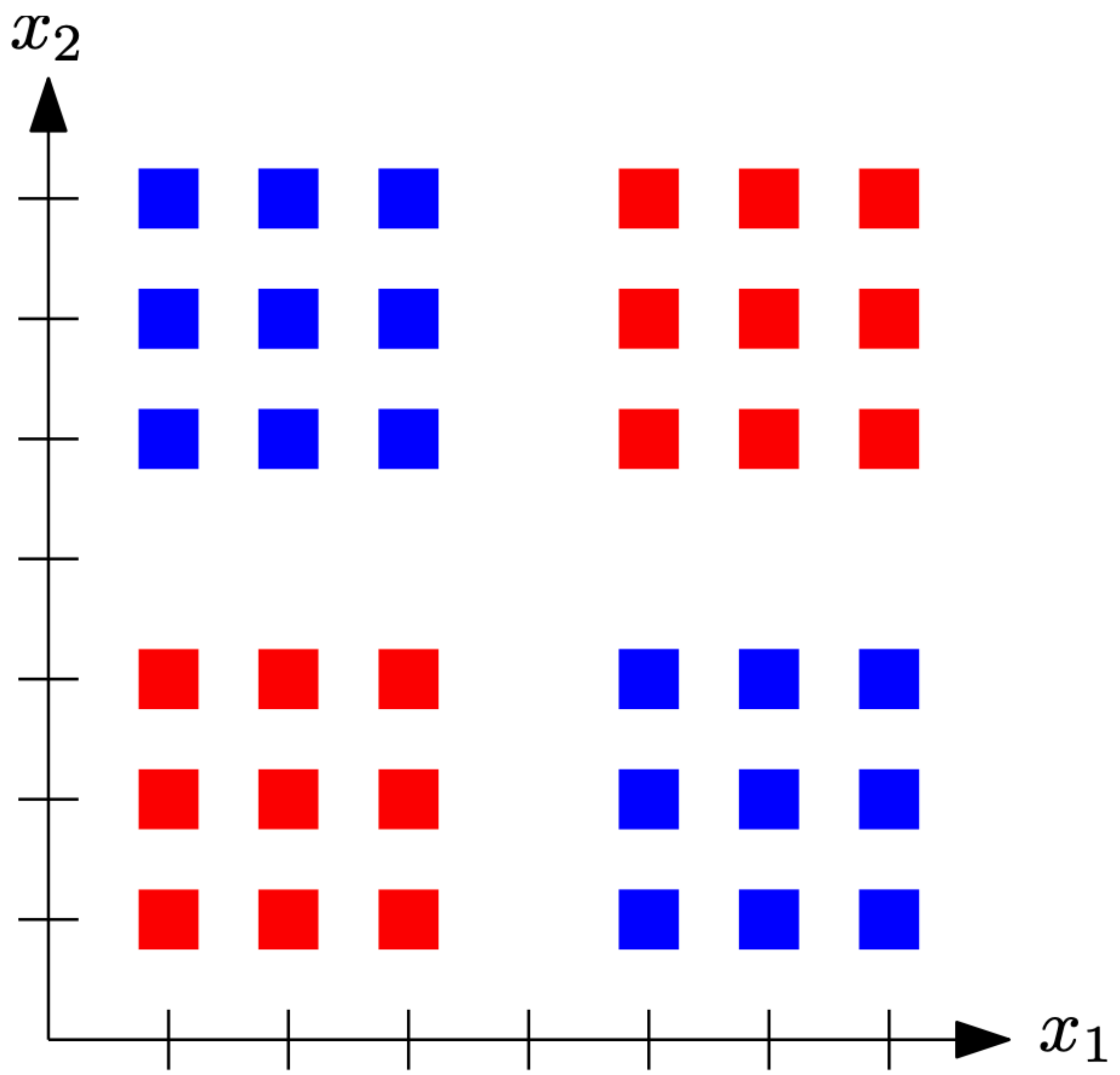


Pruning

- Note that the iterative algorithm is a "greedy" way to minimize the total uncertainty

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

- Prone to falling in local minima:
  - Fails on XOR (indifferent to splits)



- **Solution.**
  - Do "random" splits occasionally
  - Then, prune the unnecessary splits

Properties

- **Advantages.**
    - Easy to interpret
    - Fast to execute

- **Limitations.**
    - Difficult to scale up
        - Easy to overfit, if the tree is big

Properties

- Nonparametric
- Based on local regularity
    - Simple locally, complicated globally

- In these senses, similar with nearest neighbors

Forests

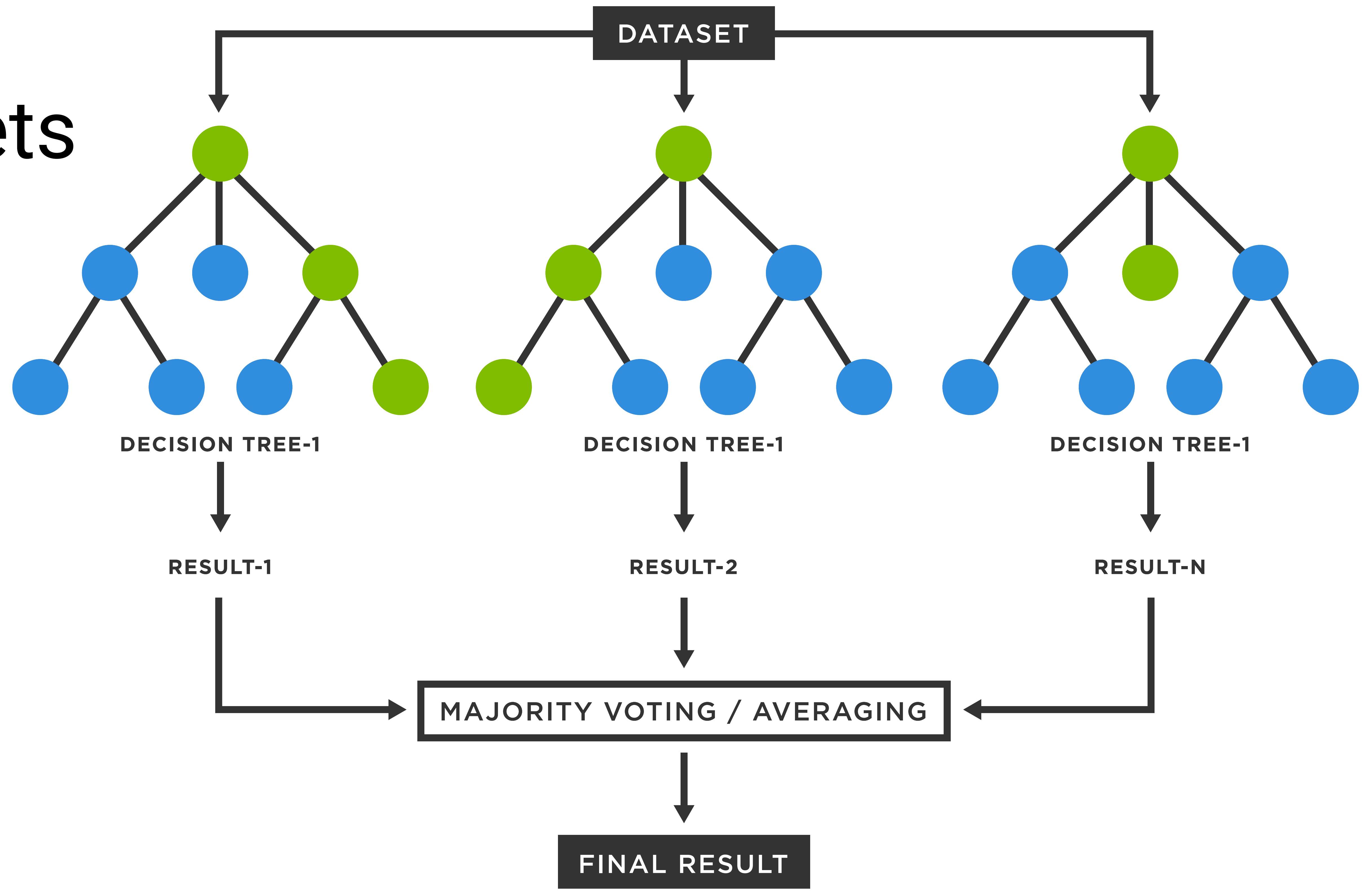
Forests

- Scaling up decision trees can be done by growing multiple trees

Bagging

- Stands for "Bootstrapped Aggregating"

- **Idea.** Split the data to multiple subsets
  - Generate a tree for each subset
  - Predictions of the trees are aggregate via
    - Majority voting
    - Averaging



Random Forest

- **Problem.** Bagging leads to highly correlated trees
  - That is, resulting trees look similar to each other

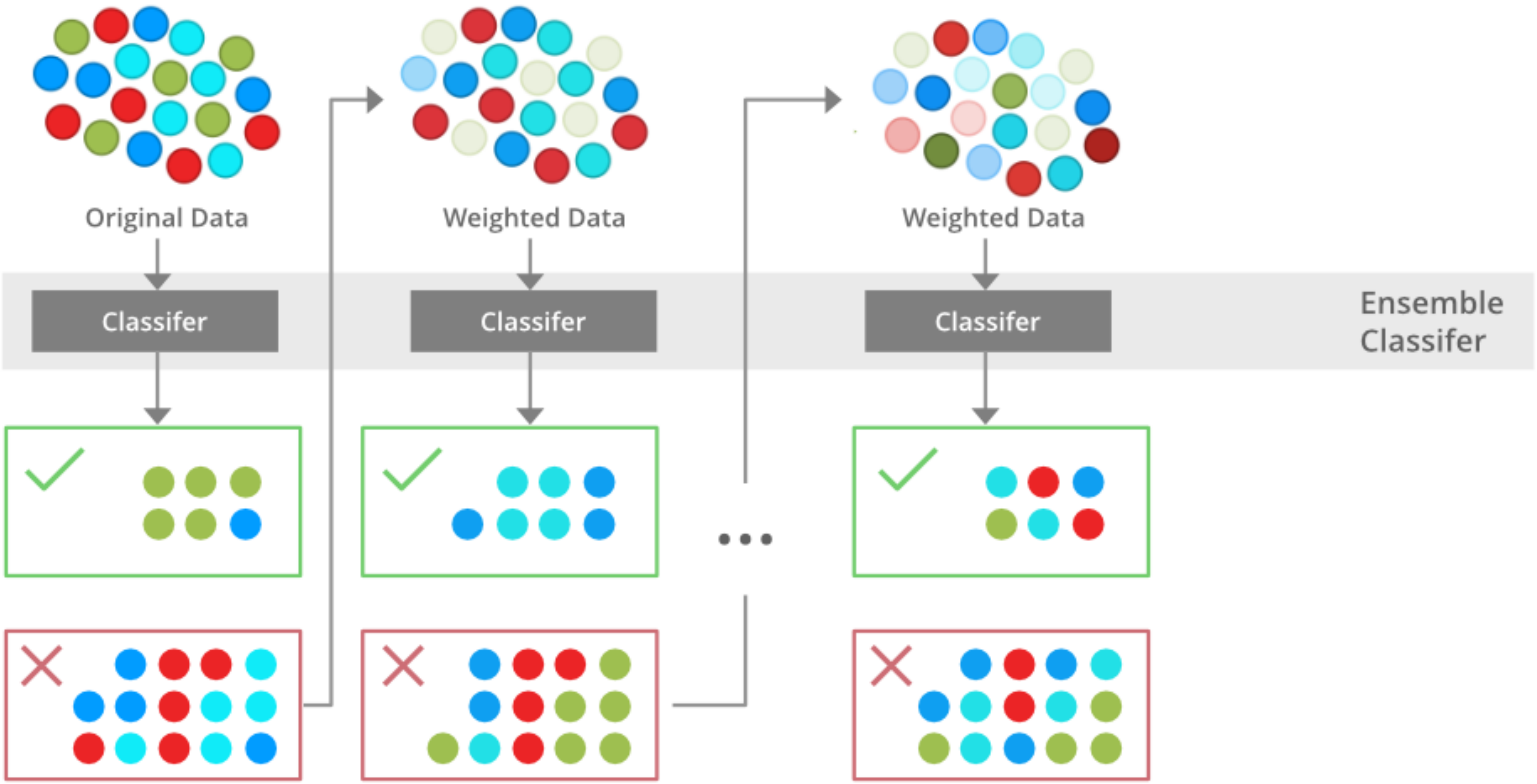
- **Idea.** Decorrelate the trees by using only a subset of features
  - To grow each node, randomly select a subset of features and choose the best one among this subset

RandomForest($\mathcal{D}; B, m, n$)

for $b = 1, \dots, B$
    Draw a bootstrap sample $\mathcal{D}_b$ of size n from $\mathcal{D}$
    Grow a tree T_b on data $\mathcal{D}_b$ by recursively:
        Select m variables at random from the d variables
        Pick the best variable and split point among the m variables
        Split the node
return tree T_b

Boosting

- **Idea.** Decorrelate by sequentially generating the trees
  - Assign higher weights to samples that other trees got wrong



</lecture 10>