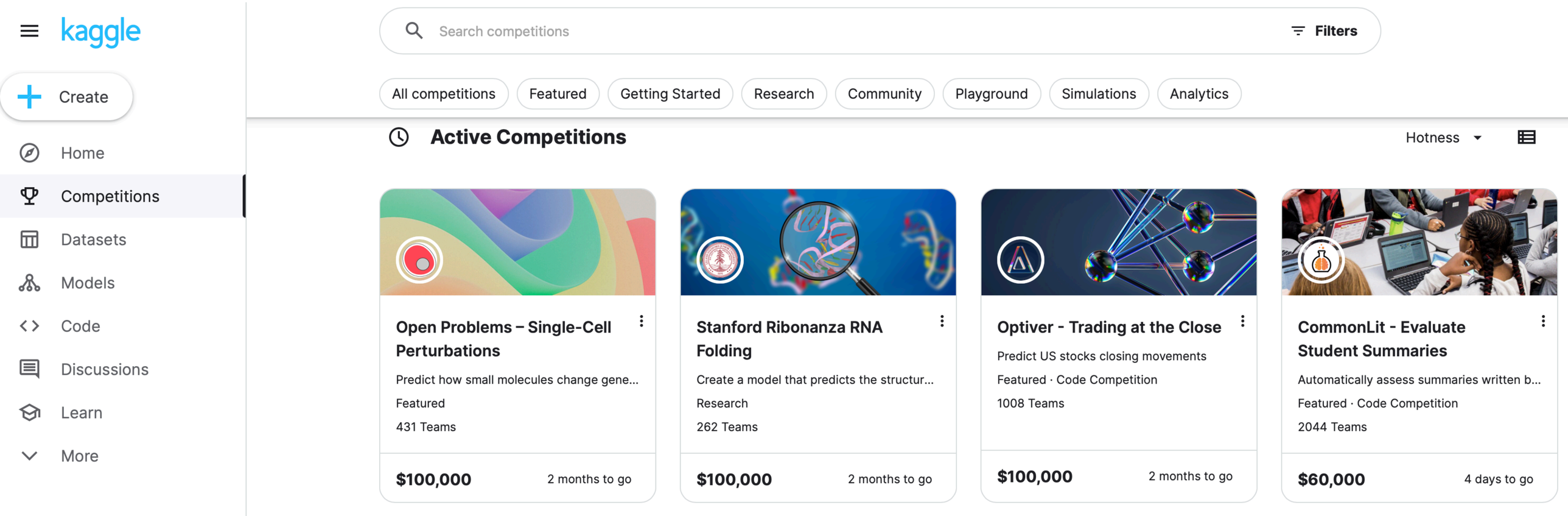
Decision Trees

EECE454 Intro. to Machine Learning Systems

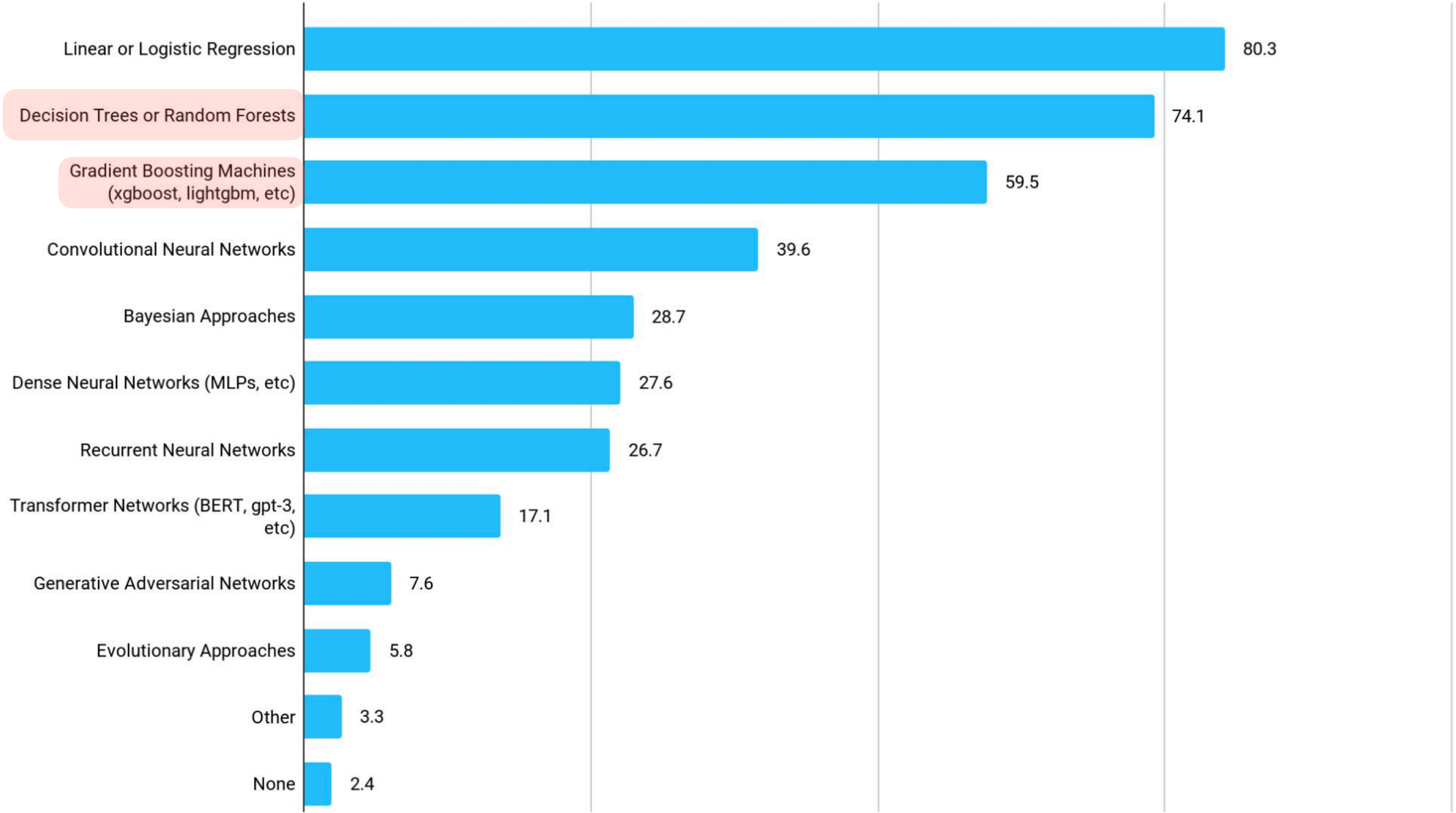
Fall 2024

Kaggle

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



Kaggle Survey 2021



Decision Trees: Historical Bits

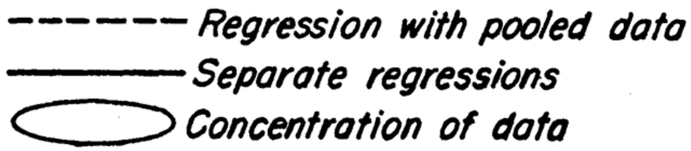
Origin

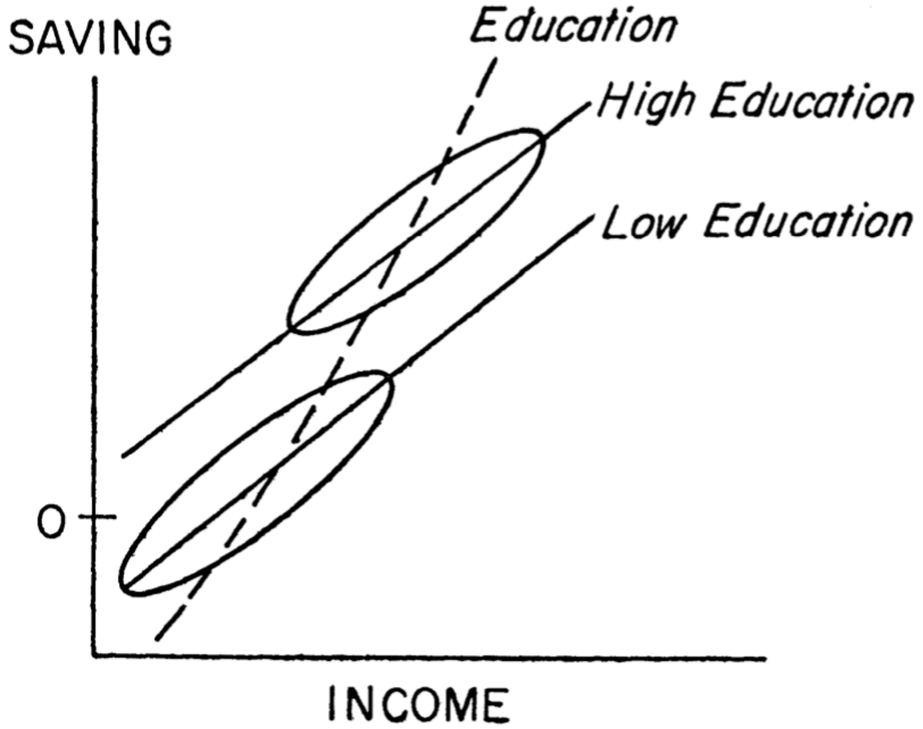
- Can be traced back to Porphyry (234? — 305?), a Greek Philosopher
    - First used “tree” to categorize living organisms (i.e., decision rules for classification)

Origin

- Can be traced back to Porphyry (234? — 305?), a Greek Philosopher
    - First used “tree” to categorize living organisms (i.e., decision rules for classification)

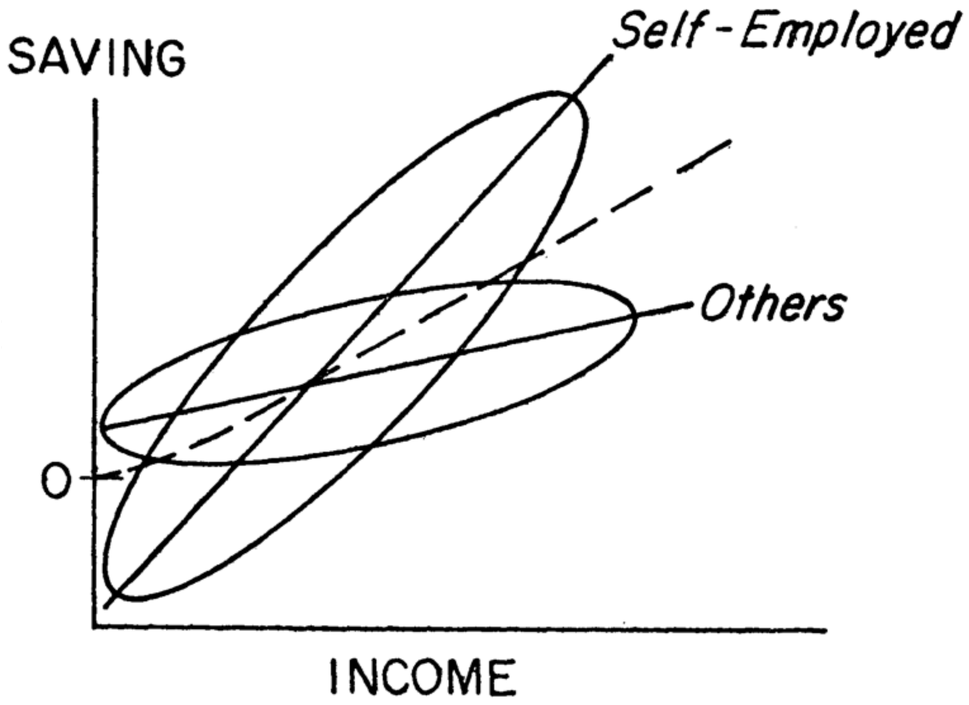
- **Modern use.** Survey data analysis by Morgan & Sonquist (1963)
    - Some data demonstrated *multicollinearity*
        - e.g., Correlation between income & education, but no interaction





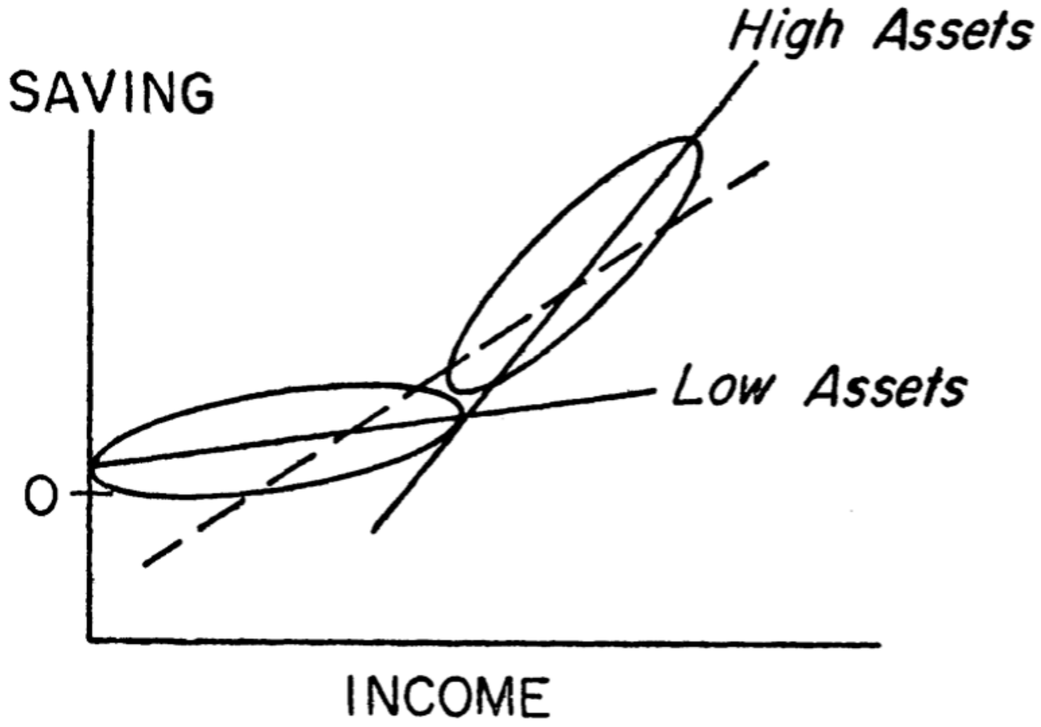
Origin

- Can be traced back to Porphyry (234? — 305?), a Greek Philosopher
  - First used “tree” to categorize living organisms (i.e., decision rules for classification)
- **Modern use.** Survey data analysis by Morgan & Sonquist (1963)
  - Some data demonstrated *multicollinearity*
  - Some demonstrated interaction between features
    - No correlation between income & self-employment



Origin

- Can be traced back to Porphyry (234? — 305?), a Greek Philosopher
  - First used “tree” to categorize living organisms (i.e., decision rules for classification)
- **Modern use.** Survey data analysis by Morgan & Sonquist (1963)
  - Some data demonstrated *multicollinearity*
  - Some demonstrated interaction between features
  - Some demonstrated both

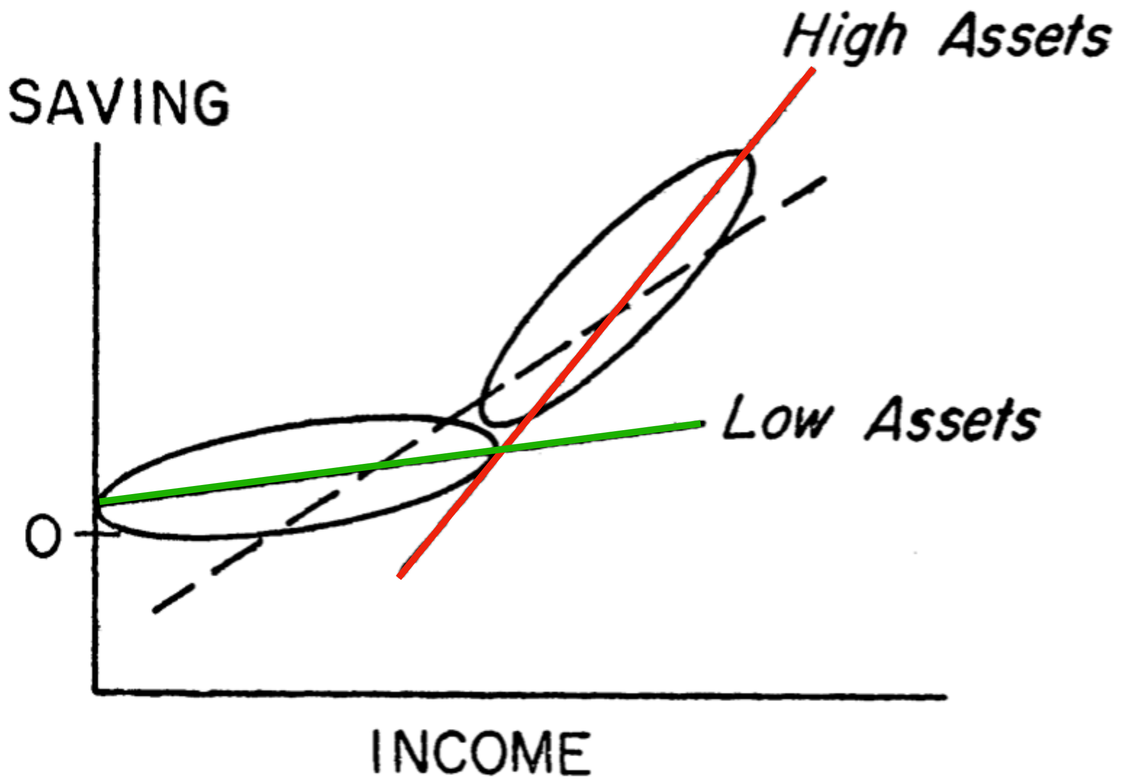


Decision trees

- **Idea.** We may need a sequential approach
  - Partition the data so that simple models work well in each partition
  - Opposed to blindly assuming that interactions are “additive”

Decision trees

- **Idea.** We may need a sequential approach
  - Partition the data so that simple models work well in each partition
  - Opposed to blindly assuming that interactions are "additive"
  - Example. High asset?
    - Yes ⇒ Use **curve 1**
    - No ⇒ Use **curve 2**



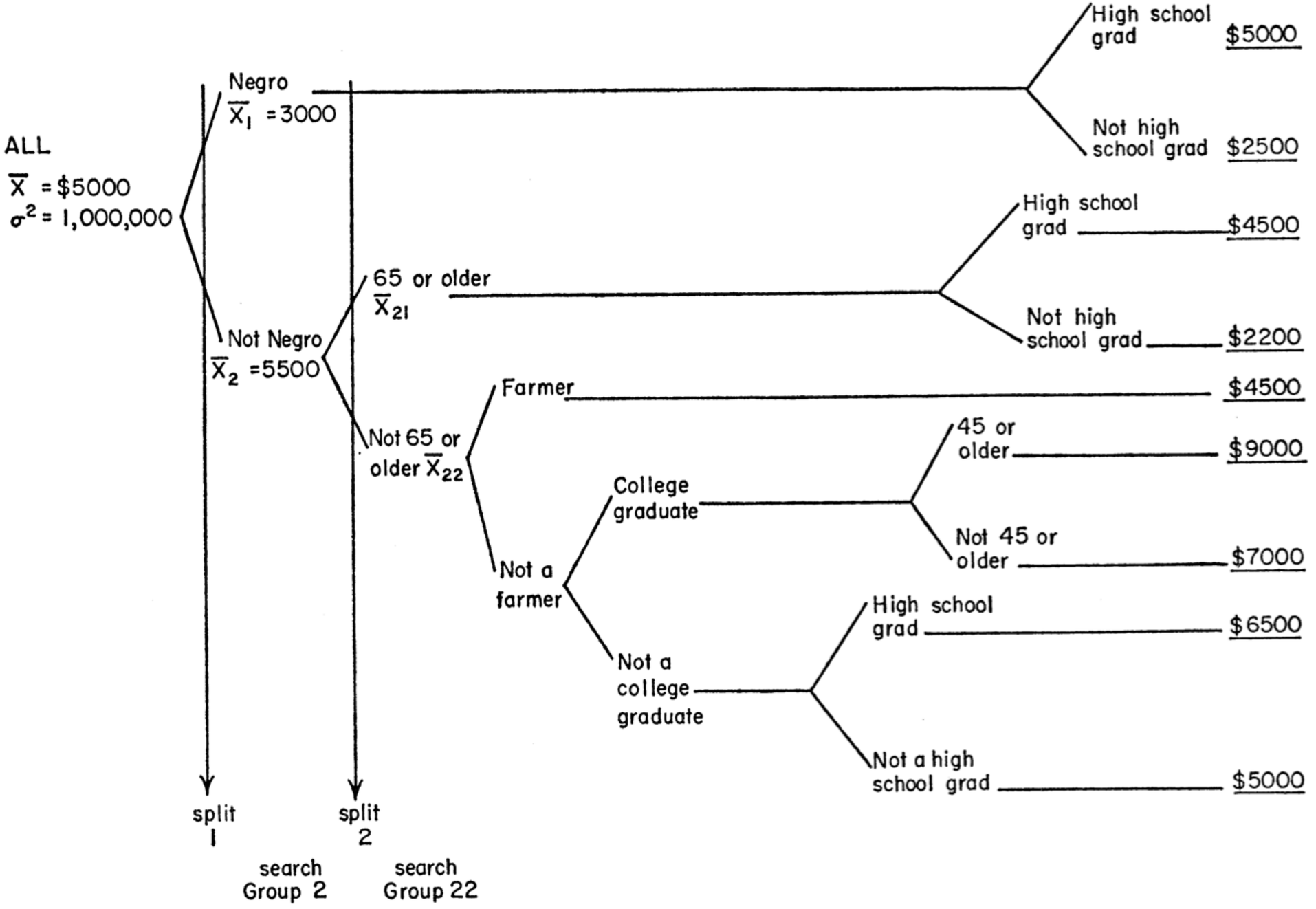
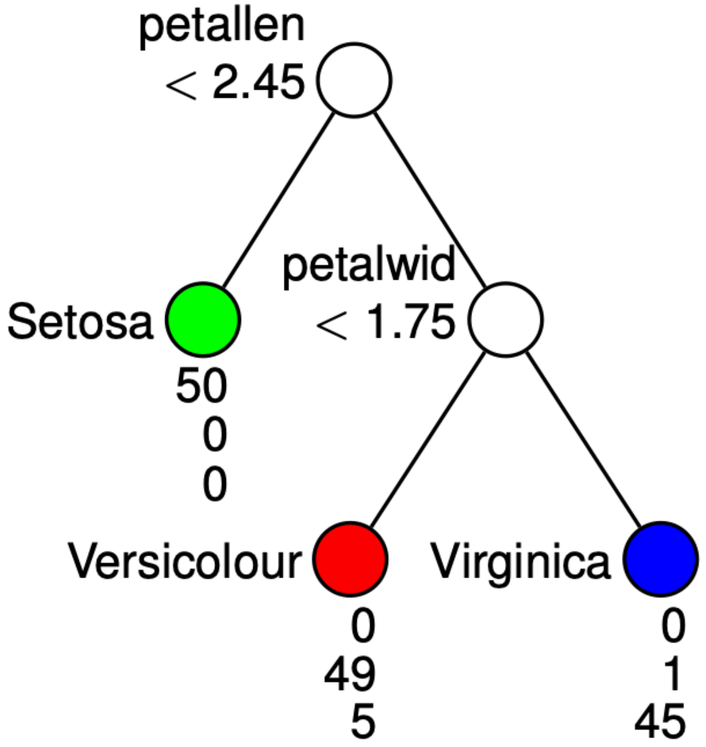


CHART II. Annual Earnings.

Decision Trees: An overview

Overview

- **What it is.** Nested if-else statements, for both classification & regression
  - A binary tree which recursively partitions and refines the input space
    - Leaf node. Assoc. with some label $\hat{y}$
    - Tree node. Assoc. with a splitting rule $g : \mathcal{X} \to \{0,1\}$

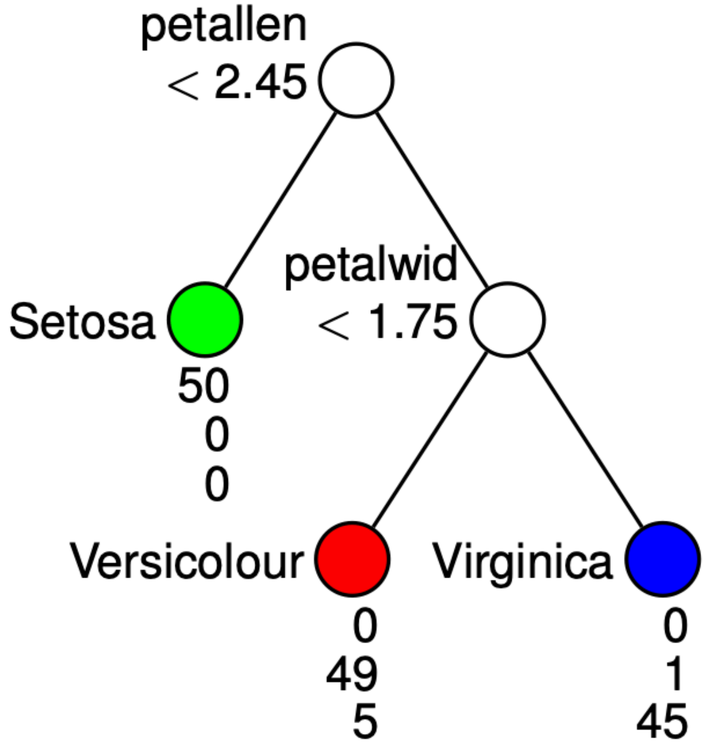


Overview

- **What it is.** Nested if-else statements
  - A binary tree which recursively partitions and refines the input space
    - Leaf node. Assoc. with some label $\hat{y}$
    - Tree node. Assoc. with a splitting rule $g : \mathcal{X} \to \{0,1\}$

- **Prediction.** Given $\mathbf{x}$, recurse down the tree until a leaf 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)
```

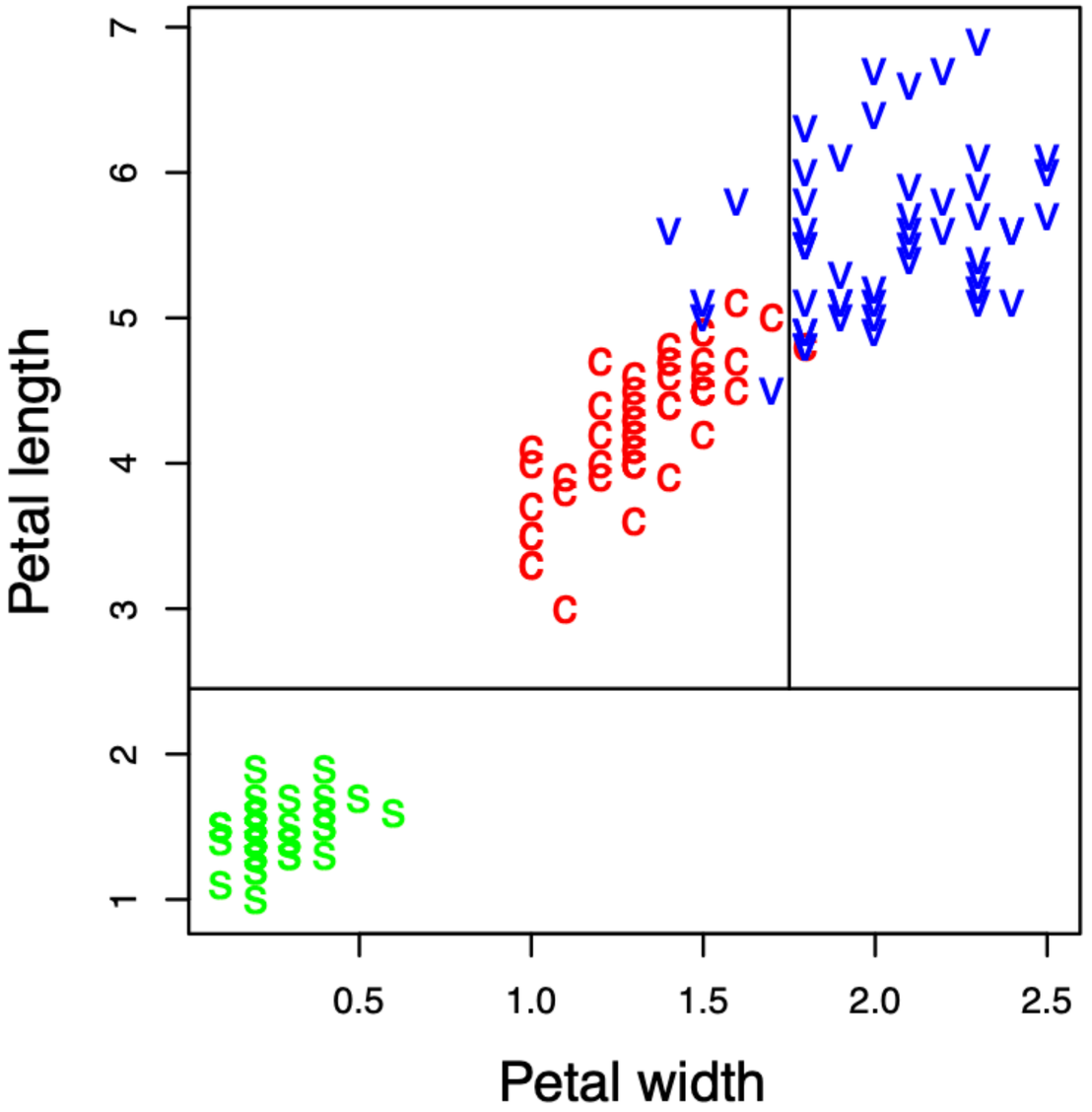


Overview

- **Note.** When $\mathcal{X} = \mathbb{R}^d$, it is typical to only consider axis-aligned splits (coordinate splits)

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

  - Computationally more efficient (Single index lookup)
  - Human-interpretable decisions



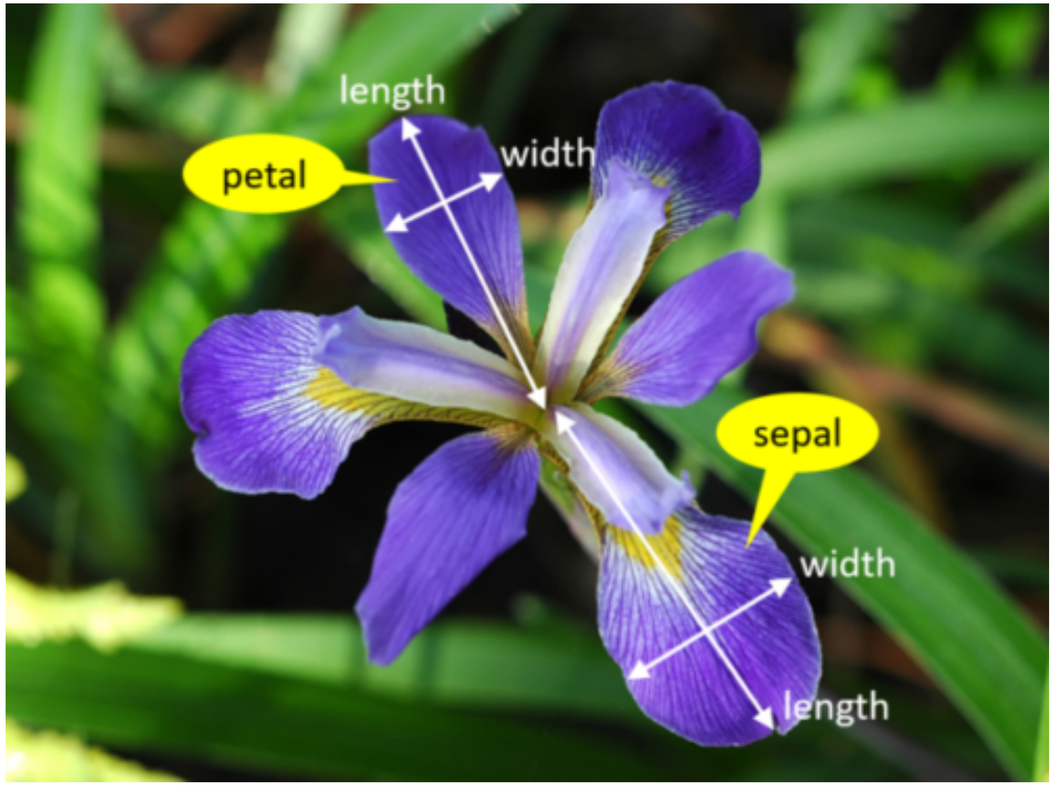
Overview

- **Training.** Constructing a decision tree requires designating three elements:
  - Prediction rule, Stopping rule, and 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
```

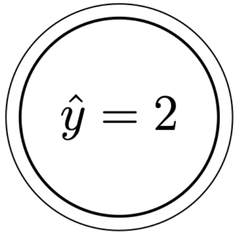
Example: Iris Classification

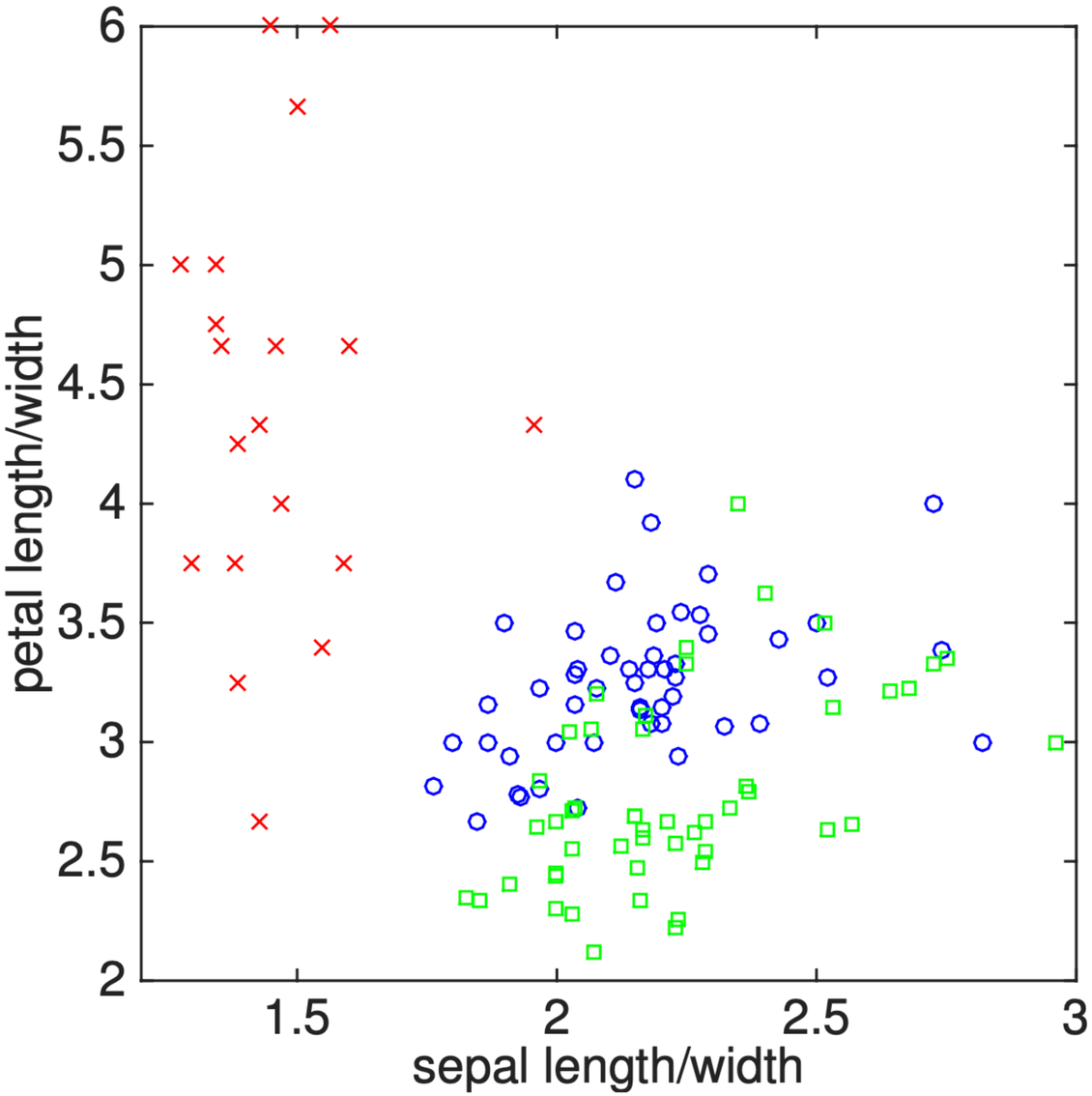
- **Example.** Consider an iris classification task
  - Input features $\mathcal{X} = \mathbb{R}^2$
    - x_1: Ratio of the sepal length and width
    - x_2: Ratio of the petal length / width
  - Output labels $\mathcal{Y} = \{1, 2, 3\}$



Example: Iris Classification

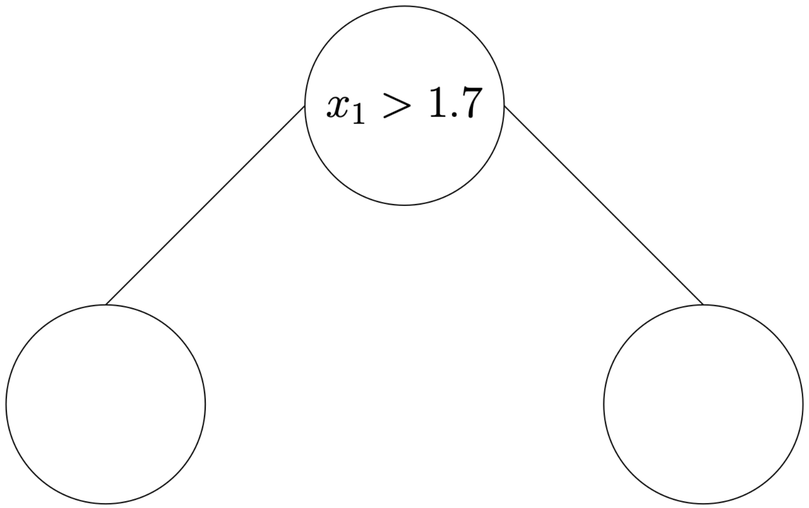
- We first construct a single leaf node, according to some prediction rule that applies for the whole set.

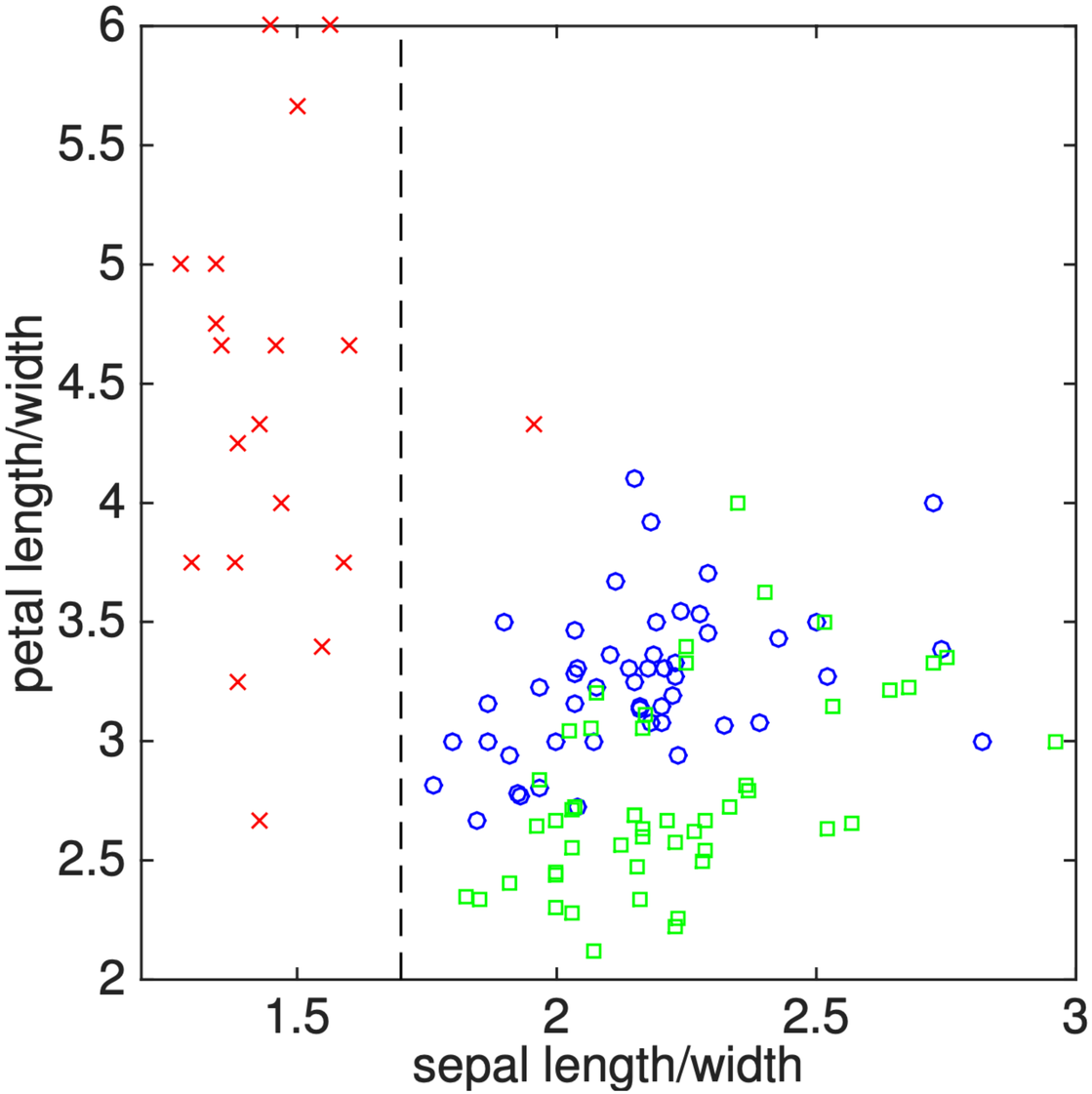




Example: Iris Classification

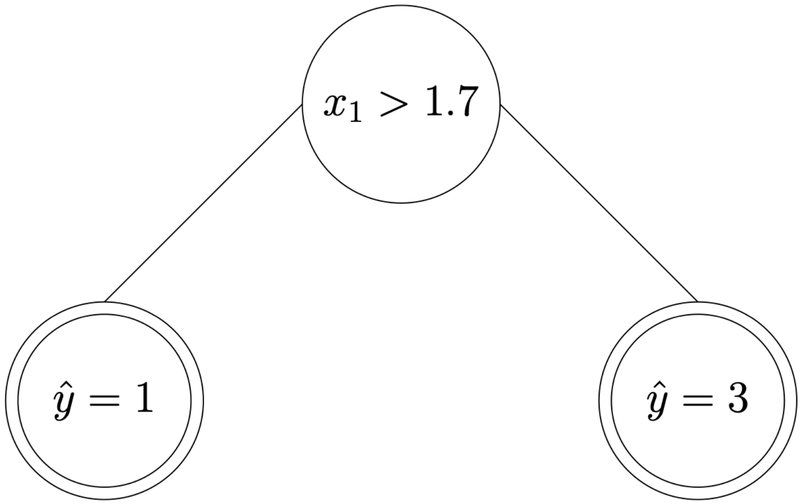
- We first construct a single leaf node, according to some prediction rule that applies for the whole set
- According to the splitting rule, split the leaf node to partition the input space for the node.

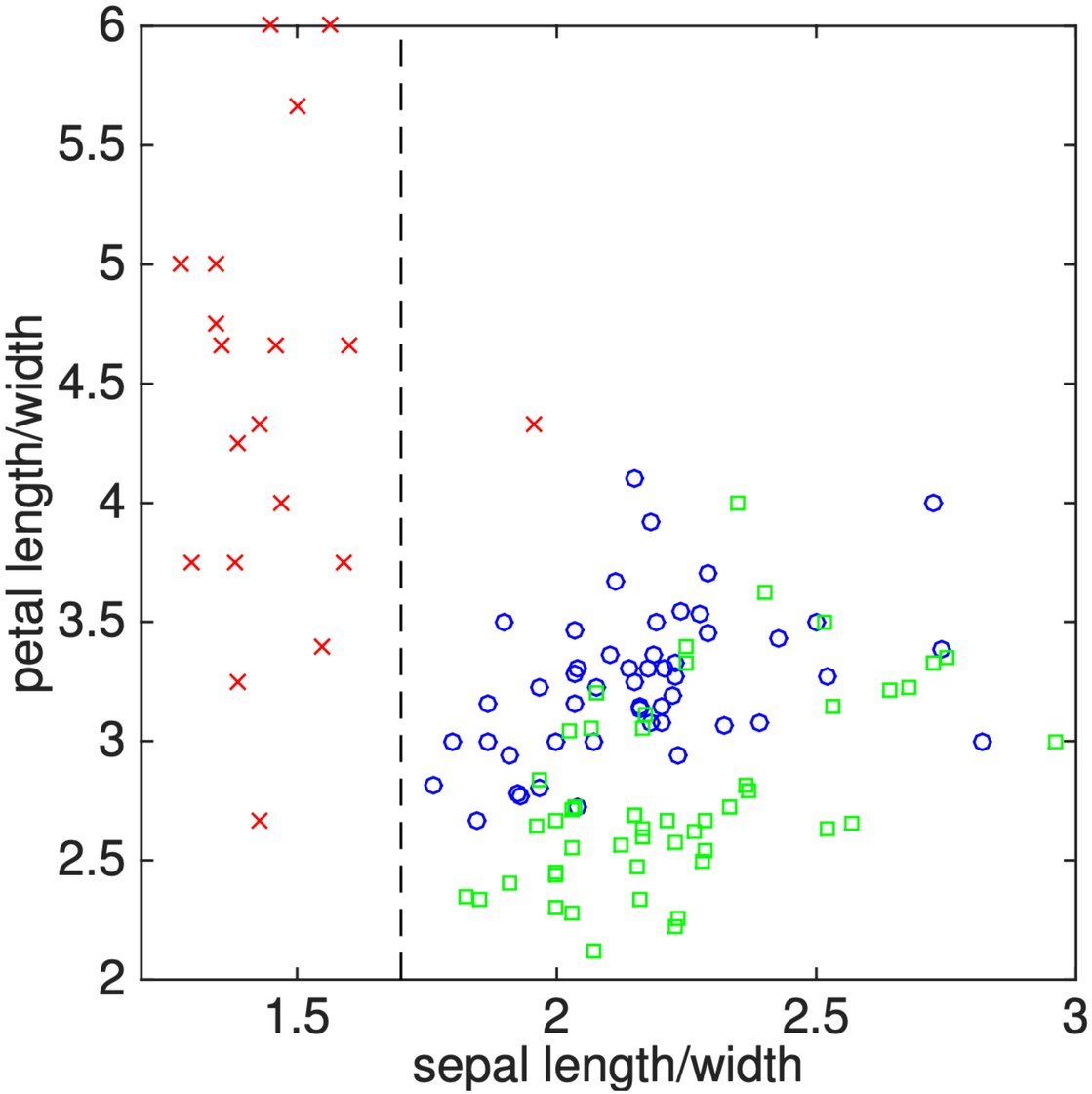




Example: Iris Classification

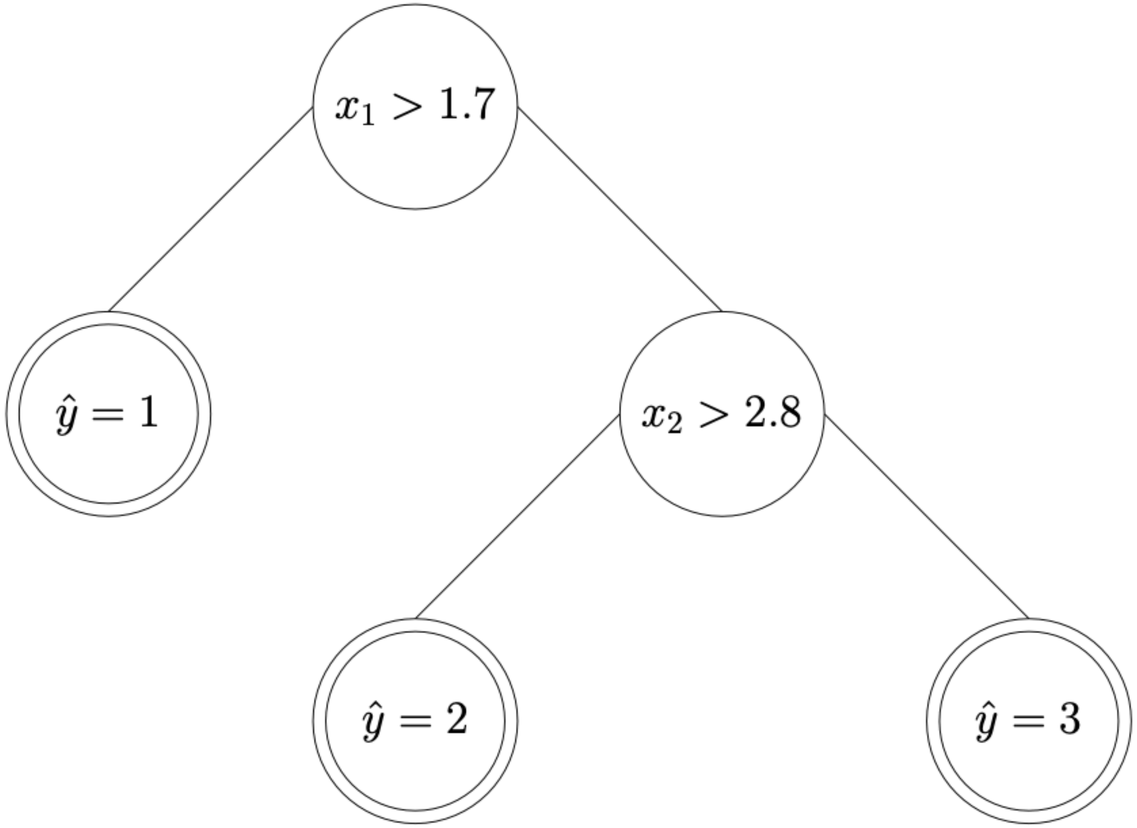
- We first construct a single leaf node, according to some prediction rule that applies for the whole set
- According to the splitting rule, split the leaf node to partition the input space for the node.
- According to the prediction rule, determine the prediction of each new leaf node.

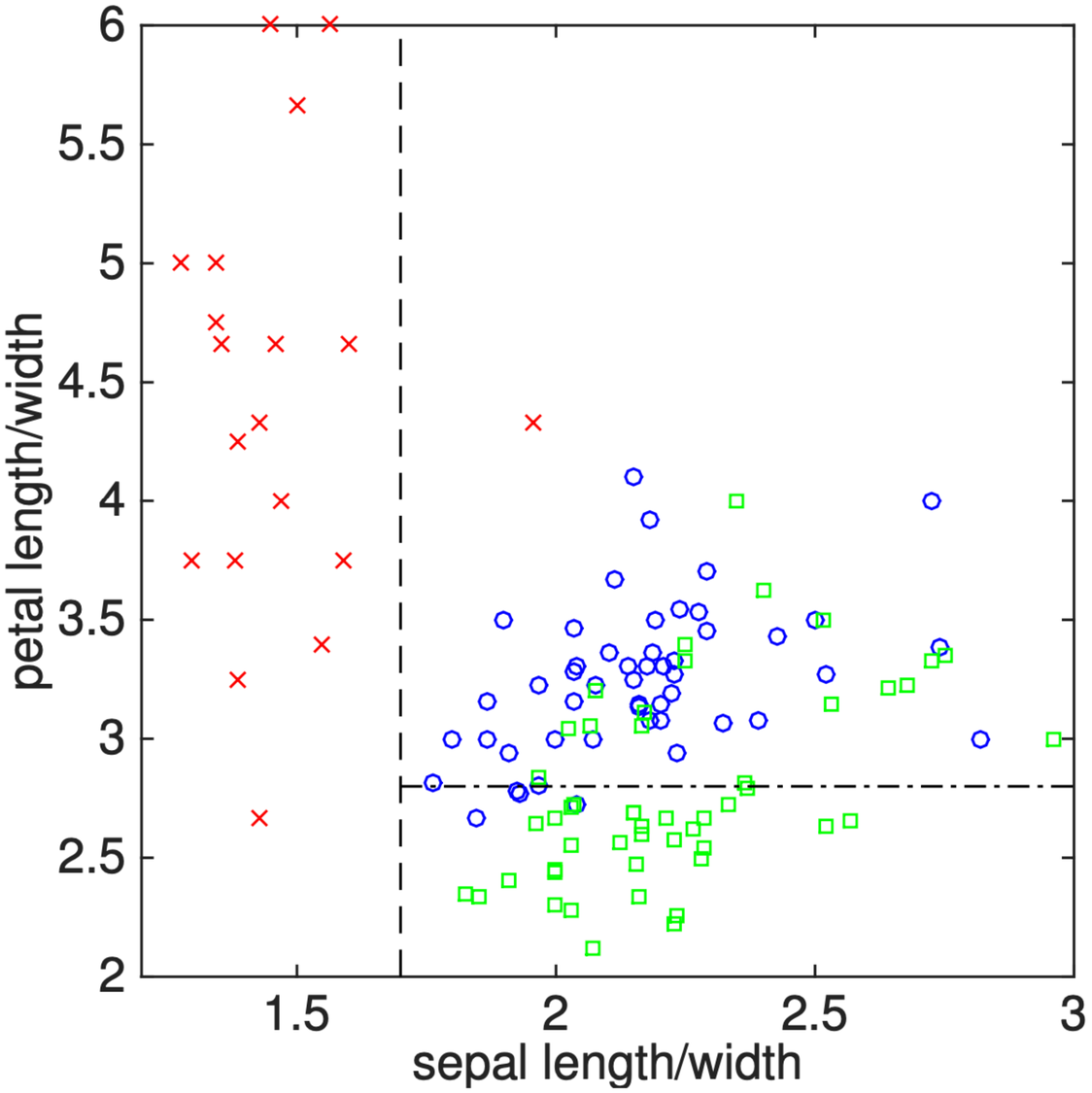




Example: Iris Classification

- Continue until some stopping rule is met.





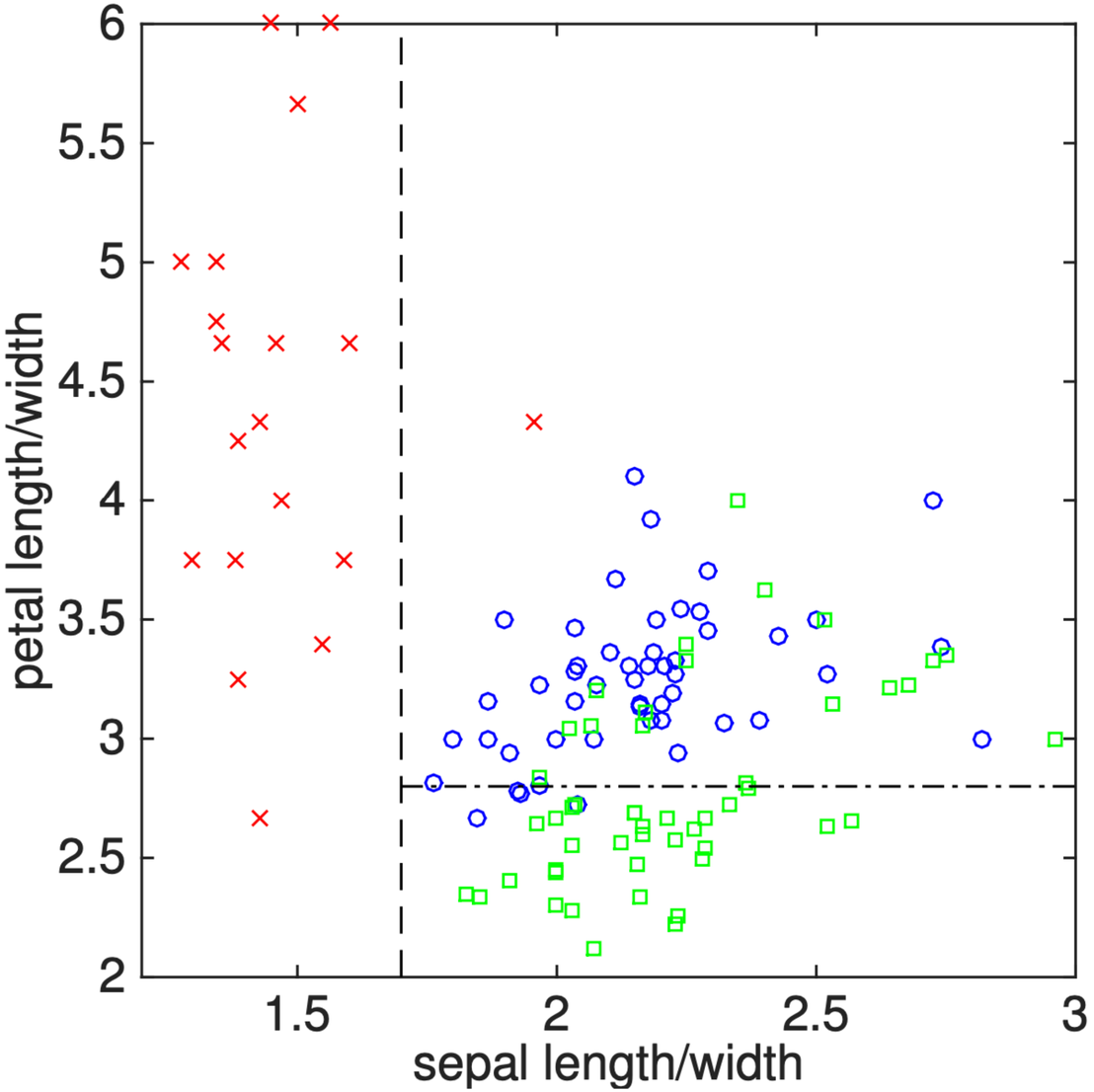
Elements of a Decision Tree

Elements

- Recall that we need rules for: Prediction, Splitting, Stopping
    - We'll look at these, one by one.

Prediction

- Generating a label for a partitioned set
  - Typically very simple
    - Classification. Majority voting
    - Regression. Average, Median, …



Splitting

- Determine how to partition the space into two, based on the data statistics
- **Idea.** Minimize some notion of uncertainty (called “impurities”) inside the partitioned sets.

Splitting

- Determine how to partition the space into two, based on the data statistics
- **Idea.** Minimize some notion of uncertainty inside the partitioned sets.
- **Example (binary classification).** Given a set S, the $p \cdot |S|$ samples are labeled $+1$
  - Classification error

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

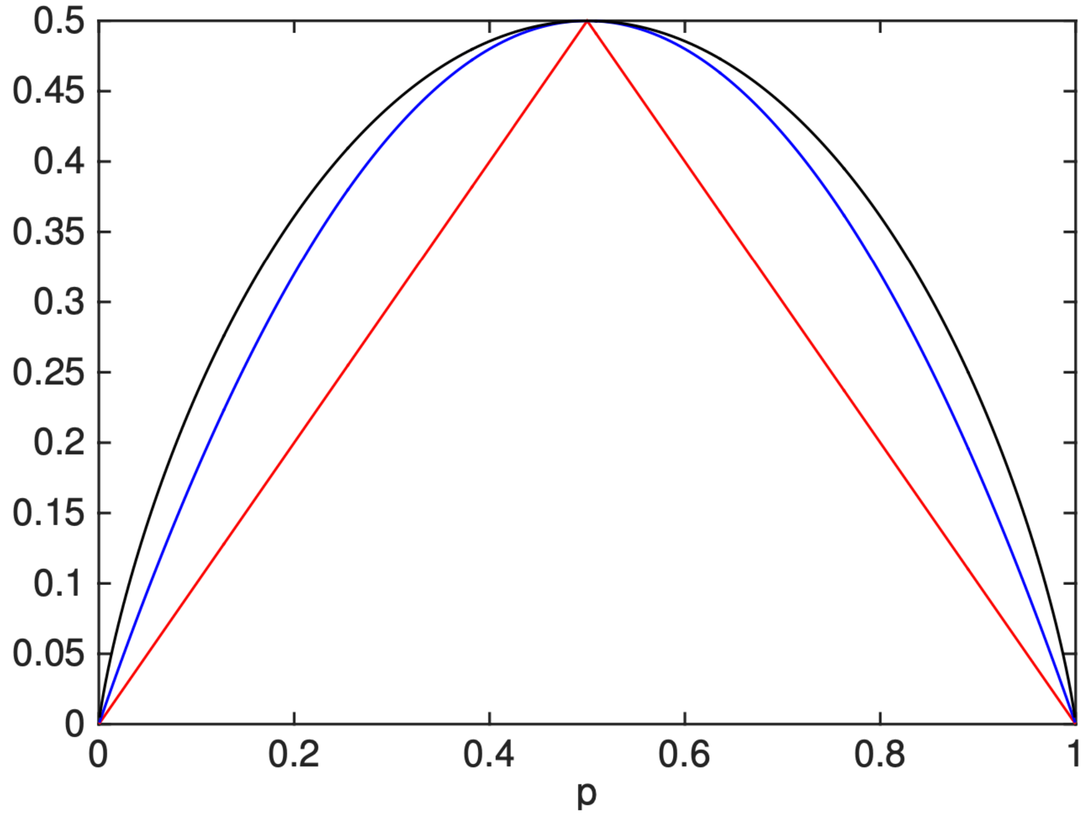
  - Gini Index

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

  - Entropy

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

    - G, E are concave upper bounds on C (easily optimizable surrogates)



Splitting

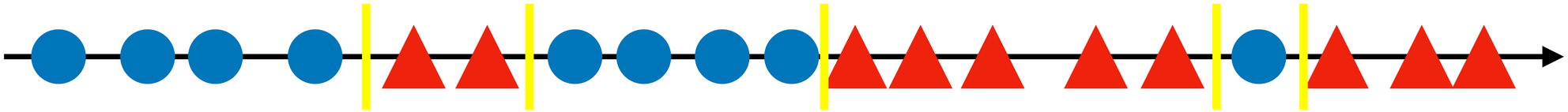
- Determine how to partition the space into two, based on the data statistics
- **Idea.** Minimize some notion of uncertainty inside the partitioned sets.
- **Example (binary classification).** Given a set S, the $p \cdot |S|$ samples are labeled $+1$
- **Example (regression).**
  - Variance. Corresponds to the average ℓ^2 error of the mean.
    - Similarly, using the average ℓ_1 error makes sense if we use median predictor.

Splitting

- If we split a set S into the sets S_1 and S_2, we want to minimize the average impurity

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

- **Want to do.** Find a nice axis & boundary for splitting.
  - Try each axis, and solve the minimization problem
    - For classification, it suffices to consider only the boundaries of the same-class clusters

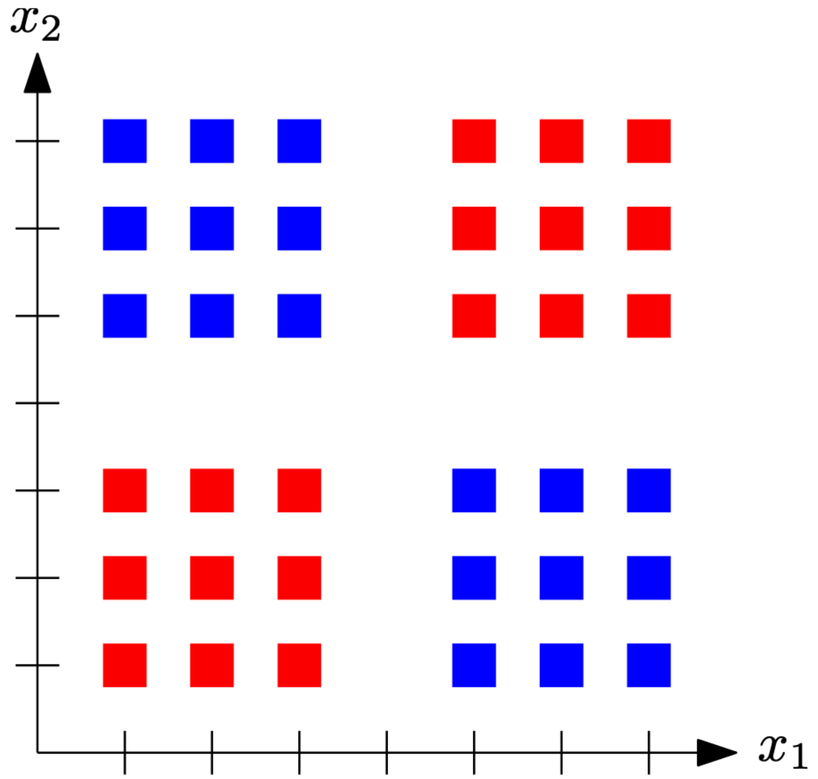


Splitting

- **Note.** The iterative algorithm is a "greedy" way to minimize the total impurity

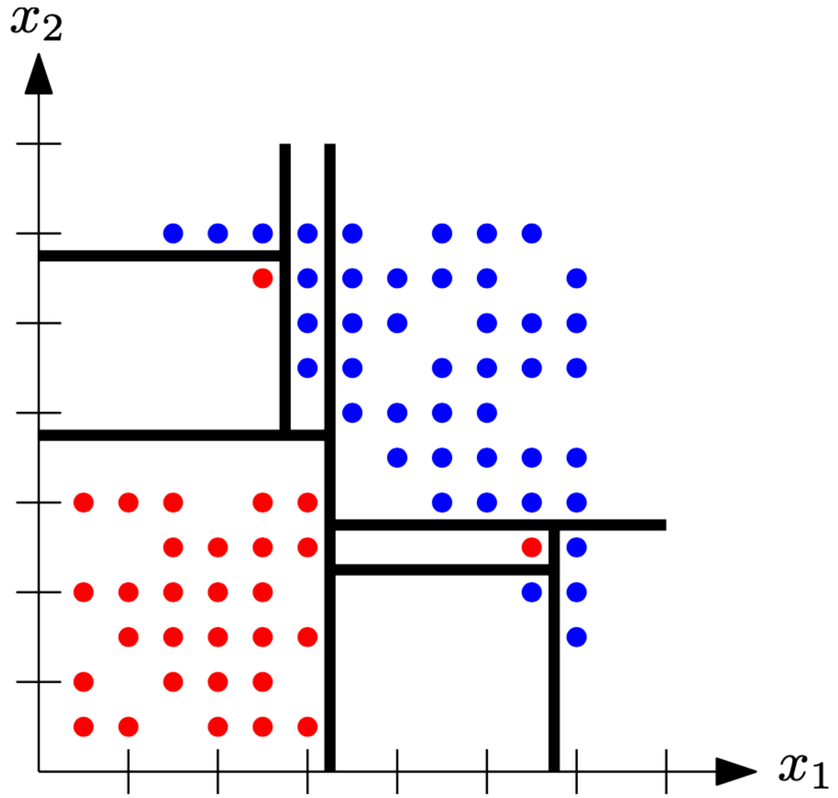
$$u(\mathcal{T}) := \frac{1}{n} \sum_{\text{leaf } S \in \mathcal{T}} |S| \cdot u(S)$$

  - Greedy algorithms fail in many cases.
    - Example. XOR — Indifferent to splitting!
    - Solution. Mix in some random splitting (and "prune out" unnecessary splits after training)



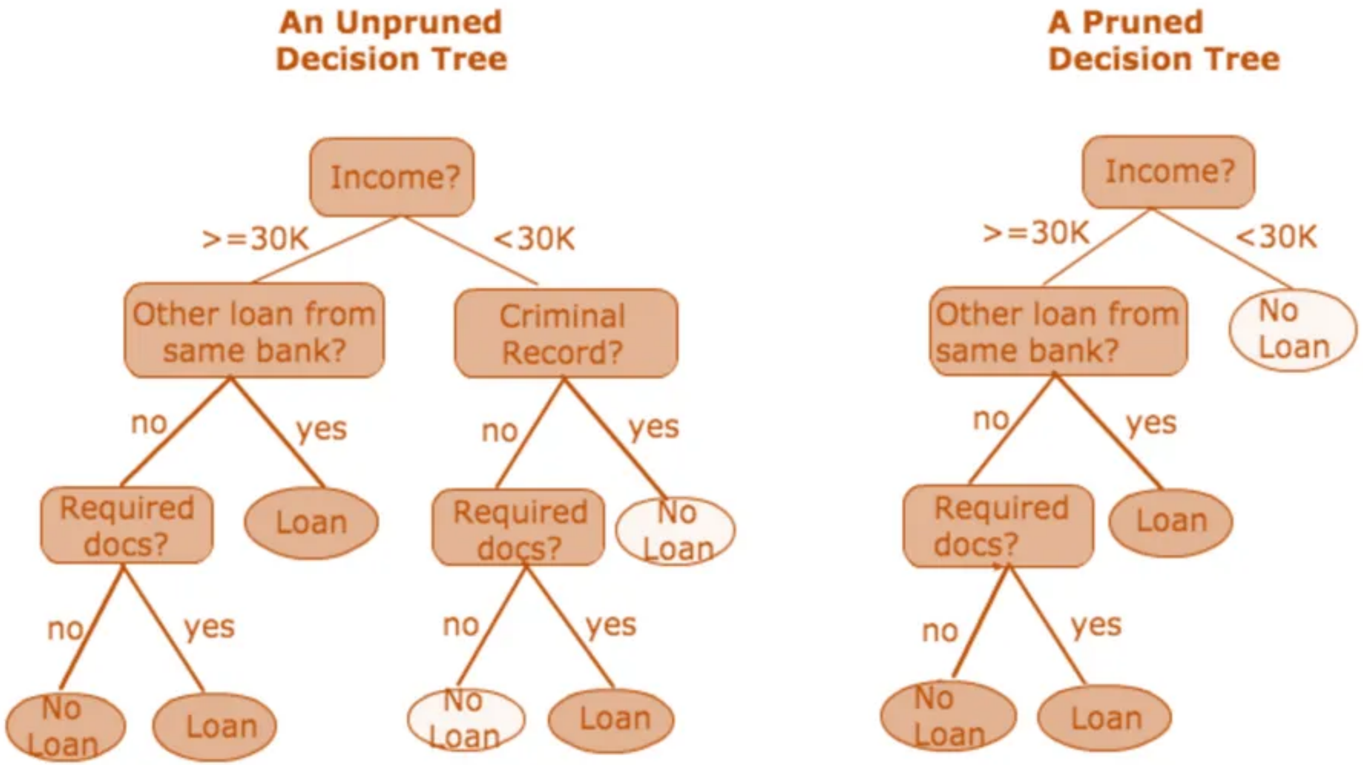
Stopping

- Determines when to stop growing a tree
  - Many criteria: Stop when ...
    - Splitting does not reduce the uncertainty
    - Reaches some pre-specified size
    - Every leaf is “pure” — contains only one class
      - Very prone to overfitting
      - Can be resolved by pruning trees after training.



“Pruning” the tree

- **Idea.** Remove the unnecessary splits, after training.
  - Pick a bottom-level split
    - Remove the split.
      - If validation error is improved, leave it pruned.
      - If not, restore the subtree.



Properties

- **Advantages**
    - Relatively easy to interpret
    - Fast to execute
    - Standard algorithms have nice properties
- **Limitations.**
    - Difficult to scale up
        - Easy to overfit, if the tree is big

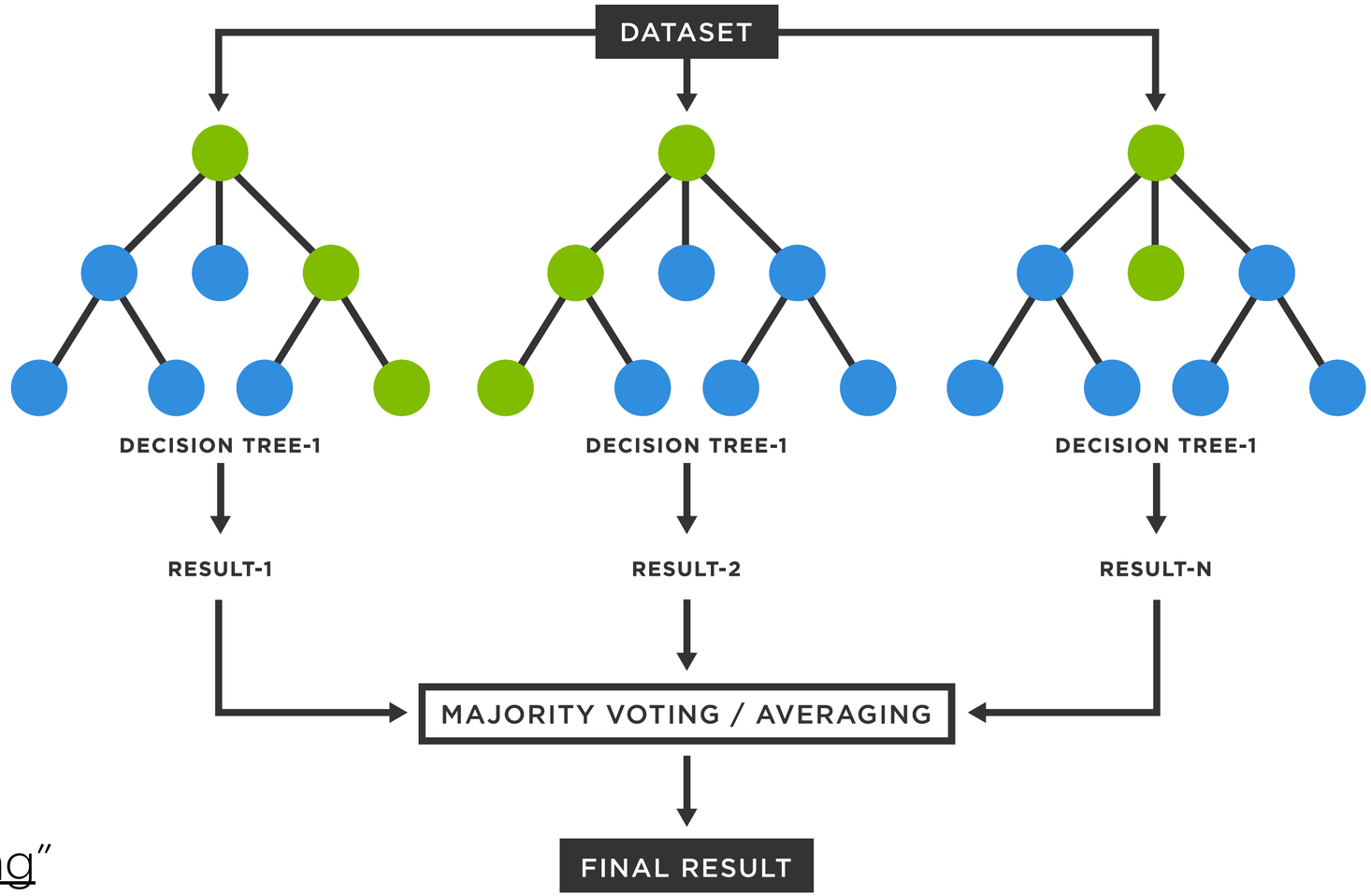
Properties

- **vs. Nearest Neighbor.** Similar, as both are
  - Nonparametric
  - Based on local regularity
    - Simple locally, despite being complicated globally

Forests

Bagging

- **Idea.** Split the data to multiple subsets, then generate multiple trees
  - One tree for each partition
  - Predictions are aggregated, via
    - Majority voting
    - Averaging
- This idea method of aggregation is called "bagging"
  - Stands for "Bootstrapped aggregating"



Random Forest

- **Problem.** Bagging often leads to highly correlated trees.
    - Most trees look similar...

Random Forest

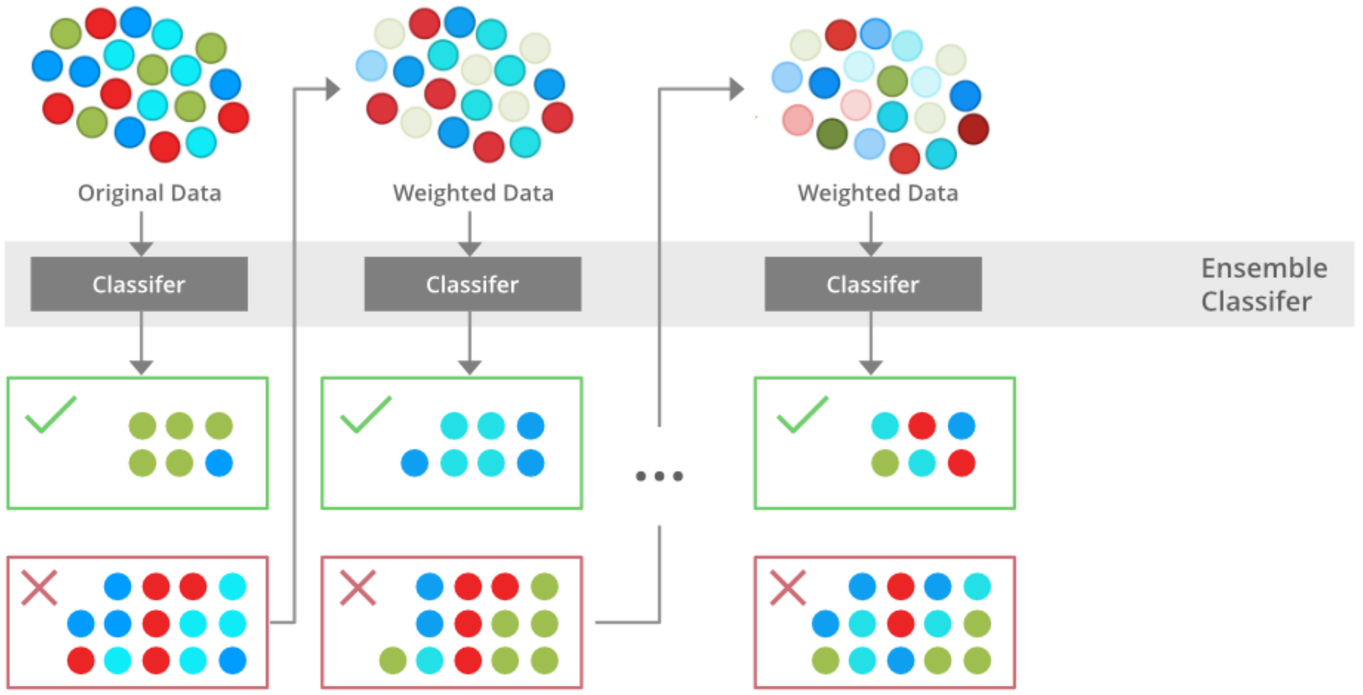
- **Problem.** Bagging often leads to highly correlated trees.
  - Most trees look similar...
- **Idea.** Decorrelate the trees by using only subset of features
  - To grow each node, select feature within a randomly selected subset of features.

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

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

Boosting

- **Idea.** Enforce the trees itself to be decorrelated, by sequentially generating them.
  - Given a tree, give higher weights to the samples it got wrong
    smaller weights to the samples it got right
  - Use reweighted data to train a new model
  - Examples.
    - AdaBoost
    - XGBoost



Cheers