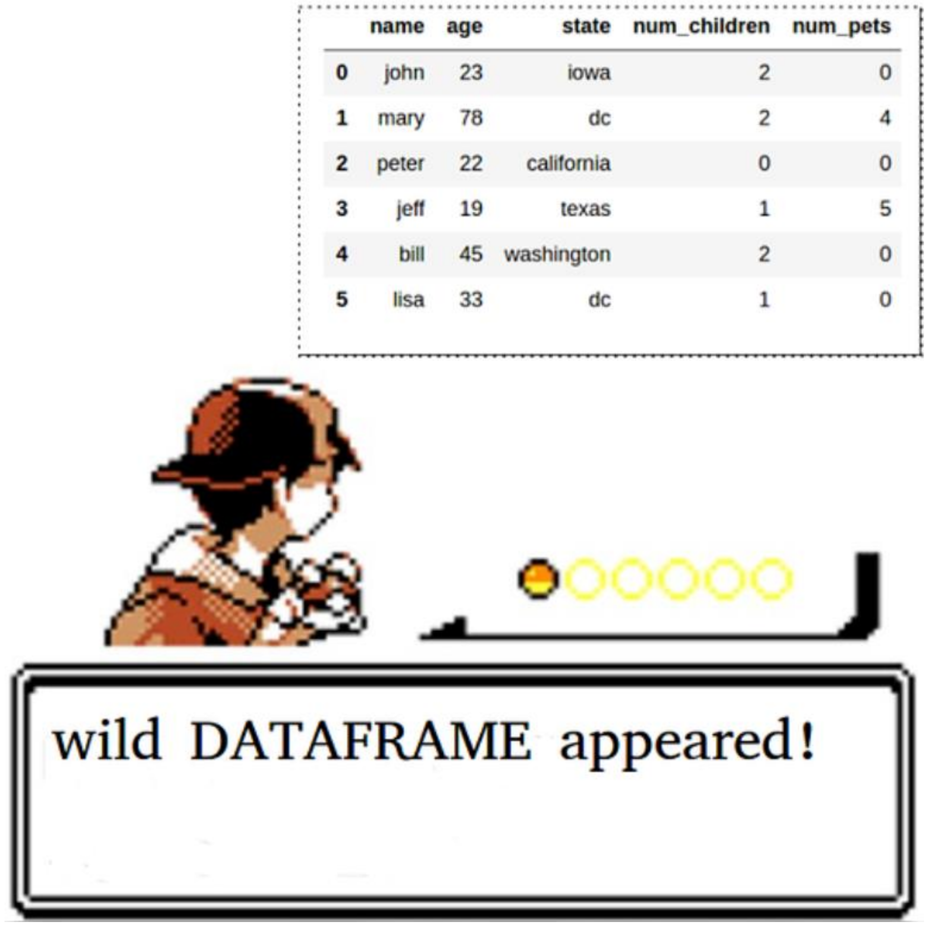
CISC 372
Advanced Data Analytics
L8 –Trees



	name	age	state	num_children	num_pets
0	john	23	iowa	2	0
1	mary	78	dc	2	4
2	peter	22	california	0	0
3	jeff	19	texas	1	5
4	bill	45	washington	2	0
5	lisa	33	dc	1	0

Monday

- Accuracy/Efficiency/Scalability/Interpretability
- Decision Tree
  - Top-down divide-and-conquer algorithm
  - Three equations:
    - Entropy

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

    - Conditional Entropy

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

    - Information Gain

$$Gain(A) = Info(D) - Info_A(D)$$

  - Interpretability
  - Overfitting & Pruning

Today

- Gain Ratio
- Gini Index
- ID3, CART, C4.5
- Feature Selection (is difficult)
- Random Forest

Information Gain

- Assume that the attributes below that have the **SAME** information gain

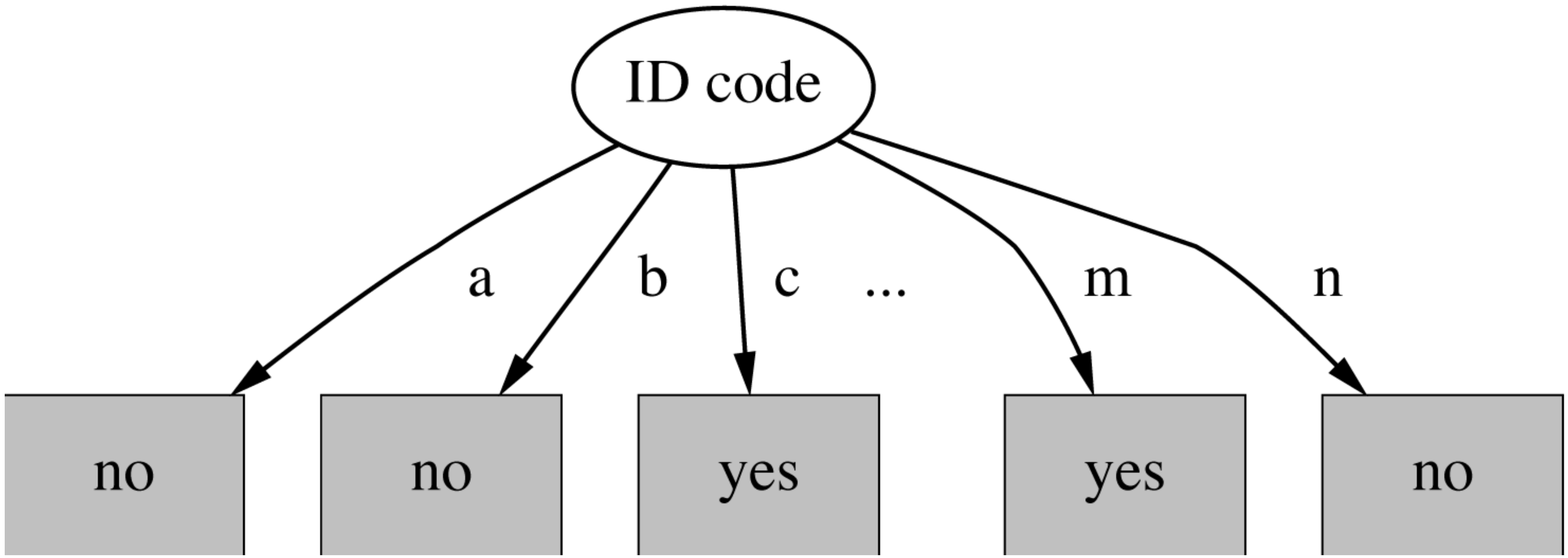
  - A: age

  - B: has_1000_driving_tickets

  - C: student_id

  - Which one would you pick?

Split for ID Code Attribute



Entropy of split = 0 (since each leaf node is “pure”, having only one case.

Information gain is maximal for ID code

Gain ratio

- *Gain ratio*: a modification of the information gain that reduces its bias towards high-branch attributes

- Gain ratio should encourage
  - Even distribution (of instances/samples)
  - Low distinct values

- How?
  - Low intrinsic information => the entropy of the attribute itself

Gain Ratio and Intrinsic Info.

- Intrinsic information: entropy of distribution of ***instances*** into branches

$$Intrinsic(A) = -\sum_{A}^{i} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

$$GainRatio(A) = \frac{Gain(A)}{Intrinsic\ (A)}$$

$$= \frac{Info(D) - Info_A(D)}{Intrinsic\ (A)}$$

More on the gain ratio

- Problem with gain ratio: it may overcompensate
  - May choose an attribute just because its intrinsic information is very low

  - Standard fix:
    - First, only consider attributes with greater than average information gain
    - Then, compare them on gain ratio

Gini Index - CART

Number of unique class labels

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

p_j is the relative frequency of class j in D

Gini Index - CART

Unique values for attribute A

$$gini_A(D) = \sum_{j}^{v_A} \frac{|D_j|}{|D|} gini(D_j)$$

Ratio of instance for A=j

gini on a subset of D where A=j

Gini Index - CART

- Algorithm for top-down induction of decision trees ("**ID3**") was developed by Ross Quinlan
  - **C4.5** (gain ratio), which can deal with numeric attributes, missing values, and noisy data

- Similar approach: **CART**
- There are many other attribute selection criteria!

Numeric attributes

- Standard method: discretization
  - E.g. temp < 45

- Unlike nominal attributes,
  every attribute has many possible split points

- Solution is straightforward extension:
  - Evaluate info gain (or other measure) for every possible split point of attribute
  - Choose "best" split point
  - Info gain for best split point is info gain for attribute

- Computationally more demanding

Numeric attributes

- Standard method: discretization
  - E.g. temp < 45

- Unlike nominal attributes,
  every attribute has many possible split points

- Solution is straightforward extension:
  - Evaluate info gain (or other measure) for every possible split point of attribute
  - Choose "best" split point
  - Info gain for best split point is info gain for attribute

- Computationally more demanding

Prepruning

- Based on statistical significance test
  - Stop growing the tree when there is ***no statistically significant*** association between any attribute and the class at a particular node
- Most popular test: ***chi-squared test***
- ID3 used chi-squared test in addition to information gain
  - Only statistically significant attributes were allowed to be selected by information gain procedure

Early stopping

	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

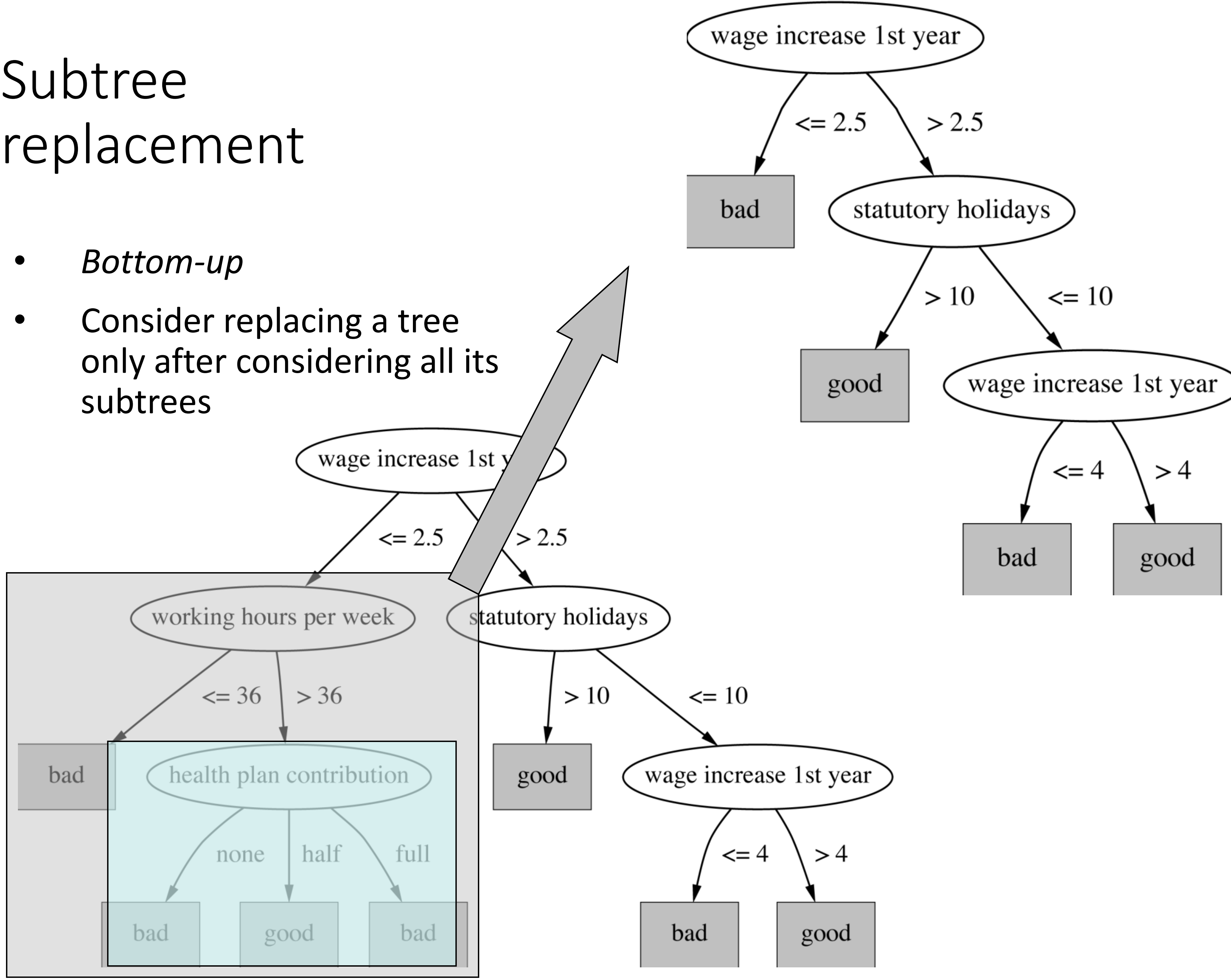
- Pre-pruning may stop the growth process prematurely: *early stopping*
- Classic example: XOR/Parity-problem
  - No *individual* attribute exhibits any significant association to the class
  - Structure is only visible in fully expanded tree
  - Pre-pruning won’t expand the root node
- But: XOR-type problems rare in practice
- And: pre-pruning faster than post-pruning

Post-pruning

- First, build full tree
- Then, prune it
  - Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:
  1. *Subtree replacement*
  2. *Subtree raising*
- Possible strategies:
  - error estimation
  - significance testing
  - Minimum Description Length principle

Subtree replacement

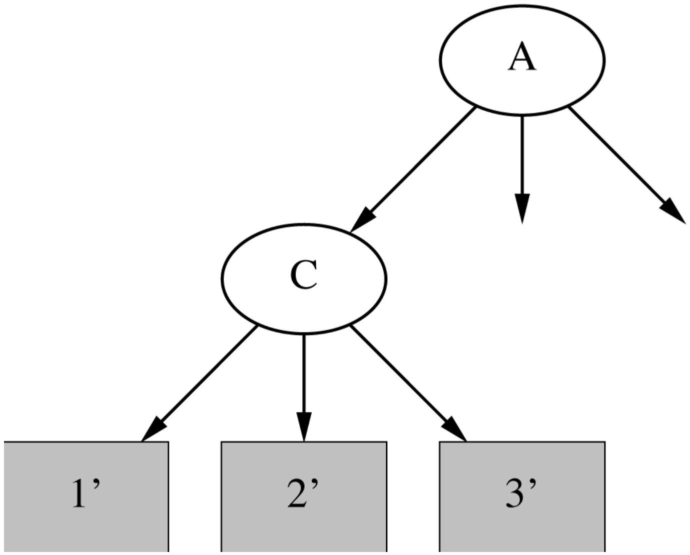
- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees

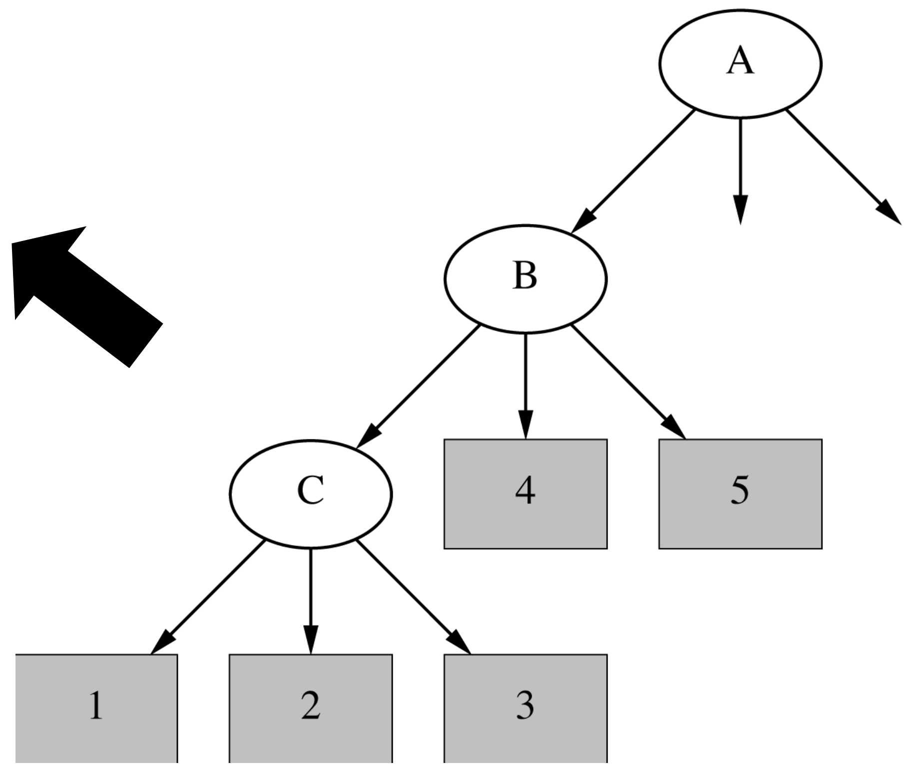


Subtree raising

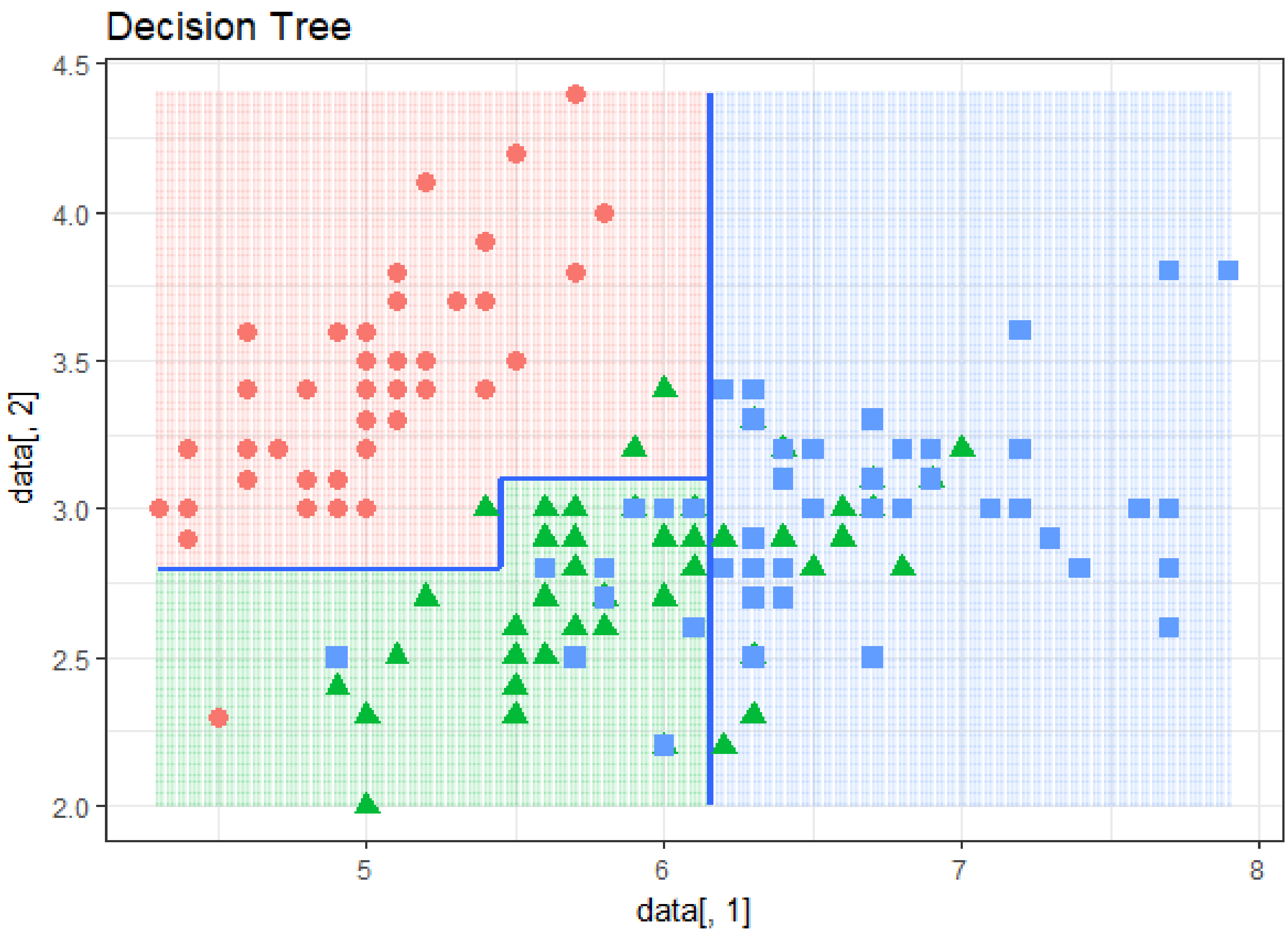
- Delete node
- Redistribute instances
- Slower than subtree replacement

  (Worthwhile?)

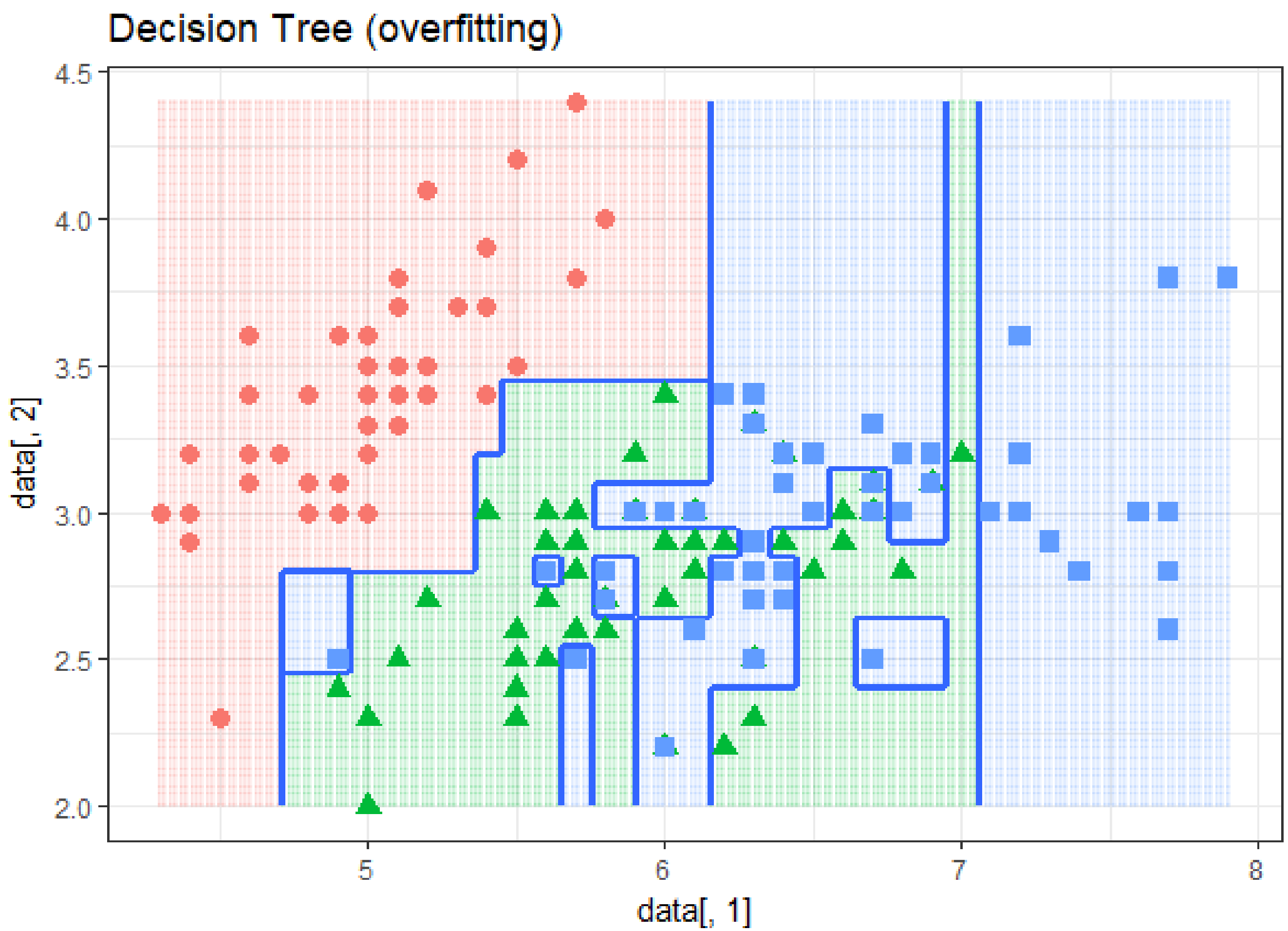




Decision Boundary



Decision Boundary – overfit (no pruning)



Feature Selection (is difficult)

- Why?
  - Given m features: high training complexity often m^2

- What (to remove)?
  - Values are not correlated with the target label
    - Correlation – no transitive property.
    - A, B can both correlate to D but A & B may/may not be correlated (so we can't just remove A/B)
    - A correlated to B => we can't just remove A
    - A,B correlates to D, and A,B correlates to each other => redundant
      - But keep A or B?
    - Theoretically we need to test every possible pair of m.

  - Low predictive power
    - A set of feature combined >= sum of each individual
    - Need to evaluate every possible subset of m.

Feature Selection (is difficult)

- Why?
  - Given m features: high training complexity often m^2
- What (to remove)?
  - Values are not correlated with the target label
    - Correlation – no transitive property.
    - A, B can both correlate to D but A & B may/may not be correlated (so we can't just remove A/B)
    - A correlated to B => we can't just remove A
    - A,B correlates to D, and A,B correlates to each other => redundant
      - But keep A or B?
    - Theoretically we need to test every possible pair of m.
  - Low predictive power
    - A set of feature combined >= sum of each individual
    - Need to evaluate every possible subset of m.

Feature Selection

Random Forests

The name comes from the fact that this was originally an extension of decision trees, but the idea applies to any predictor.

The basic idea: build a large number of decision trees, each using only some of the attributes of the dataset. Deploy the forest using voting (classification) or averaging (regression).

Recall: Bootstrapping

Recall: a bootstrap sample from n objects means draw n objects with replacement.

The sample will contain about 2n/3 unique objects, leaving n/3 objects that were never selected.

These unselected objects can be used as a test set with nice properties.

Growing one tree of the forest:

1. Select a bootstrap sample of the object**s**

2. If there are m attributes, choose some k < m (usually k is much smaller than m, maybe √m)

3. Grow a decision tree by randomly selecting k of the m attributes at each level, and choosing the best split based on these k attributes (in whatever standard way)

4. Grow the decision tree to full size (i.e. no pruning)

5. Use the out-of-bag set as a test sample and measure the test error for this tree (ongoing estimate of how well we're doing)

6. Run the entire dataset through the tree. Whenever two objects end up at the same leaf increase the proximity score for the pair.

Growing the forest:

1. Grow some number of individual trees – this is quite cheap so we might grow 100s or 1000s.

2. For each object, consider how many times it appeared in a test set, and count

$$\frac{\text{number of times prediction was wrong}}{\text{number of times it appeared in the test set}}$$

Average this to get an overall test error

3. Divide the proximities by the total number of trees

Advantages:

- handles multiple classes
- fast (worse than 1 decision tree, better than a neural network)
- low variance, low bias
- no separate test procedure needed (no cross-validation)
- high accuracy
- effective for large numbers of attributes
- helpful for attribute selection
- does not overfit no matter how many trees

Disadvantages:

- opaque predictor
- expensive to deploy