CIS 8695

Classification and Regression Trees (CART)

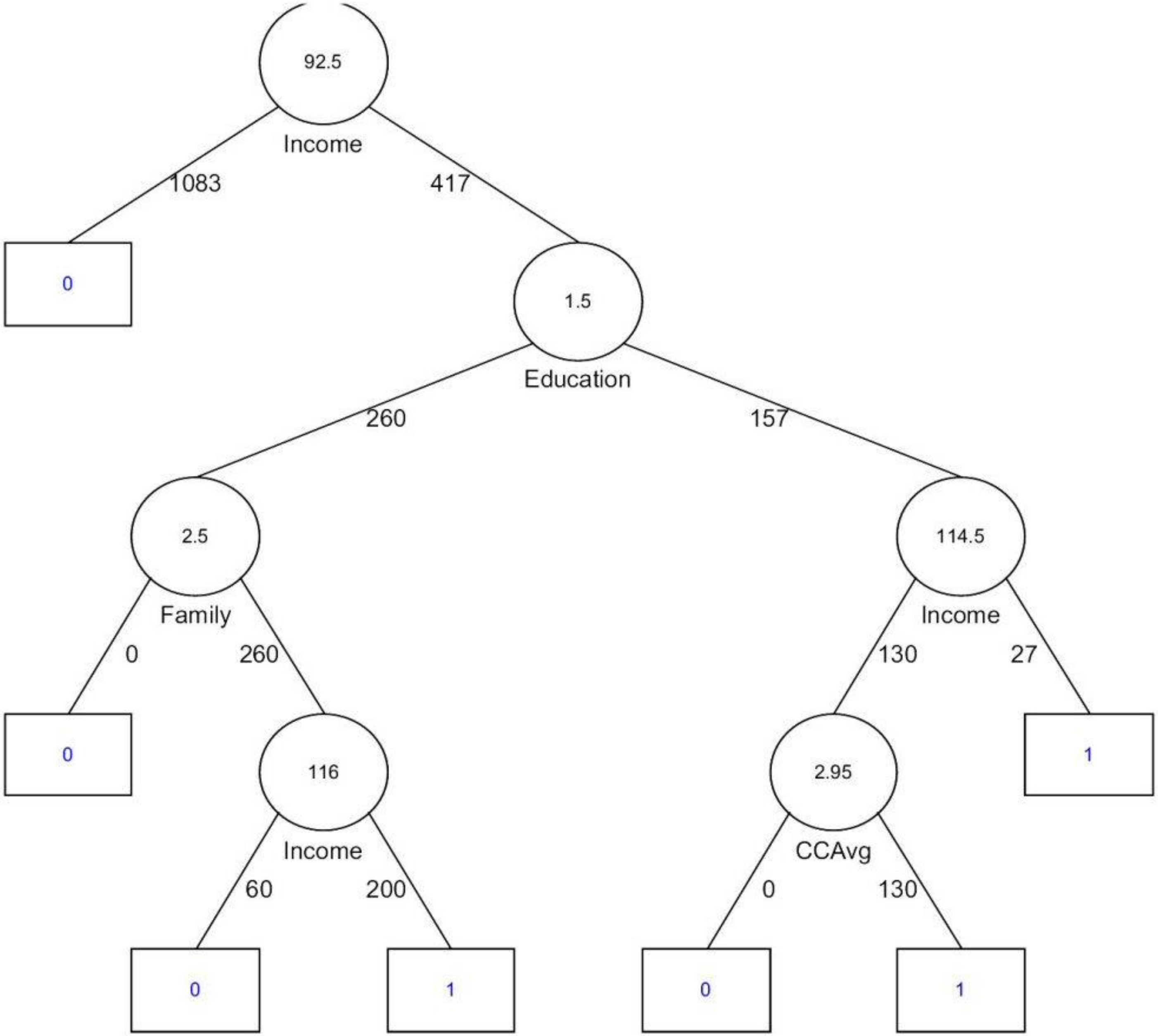
Ling Xue

CART

- **Goal:** Classify or predict an outcome based on a set of predictors
- The output is a set of **rules**

Example:

- Goal: classify a record as “will accept credit card offer” or “will not accept”
- Rule might be “IF (Income > 92.5) AND (Education < 1.5) AND (Family <= 2.5) THEN Class = 0 (nonacceptor)
- Also called CART, Decision Trees, or just Trees
- Rules are represented by tree diagrams

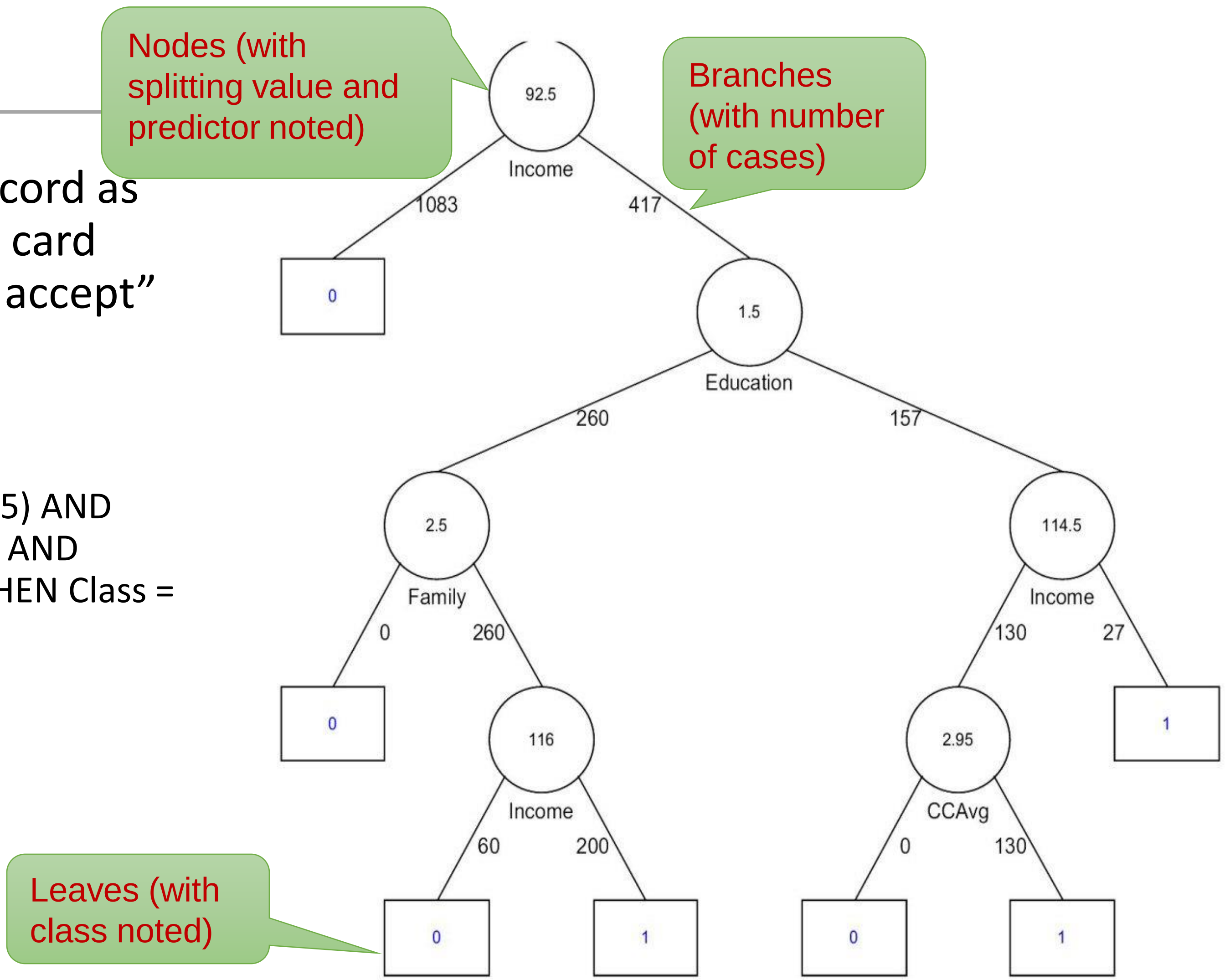

92.5
Income
1083
417
0
1.5
Education
260
157
2.5
Family
0
260
114.5
Income
130
27
0
116
Income
2.95
CCAvg
1
60
200
0
130
0
1
0
1

Example

- Goal: classify a record as "will accept credit card offer" or "will not accept"

- Rule example:
  - "IF (Income > 92.5) AND (Education < 1.5) AND (Family <= 2.5) THEN Class = 0 (non-acceptor)

Tree Representation



Key Ideas

- Goal: Classify or predict an outcome based on a set of predictors
  - The resulting subgroups should be more homogeneous in terms of the outcome variable.
- Two main ideas
  - **Recursive partitioning:** Repeatedly split the records into two parts so as to achieve maximum homogeneity within the new parts
  - **Pruning the tree:** Simplify the tree by pruning peripheral branches to avoid overfitting

Recursive Partitioning

Recursive partitioning

- Start from the root node
  - Represents all input data points

- **Main step**: If the node does not satisfy some "stopping" criteria:
  - Choose the "best" predictor to split the node, leading to two new nodes. Training data is partitioned accordingly.
  - Repeat the main step for each of the new nodes (*this is the recursive partitioning step*).

Choose the optimal split: maximize purity

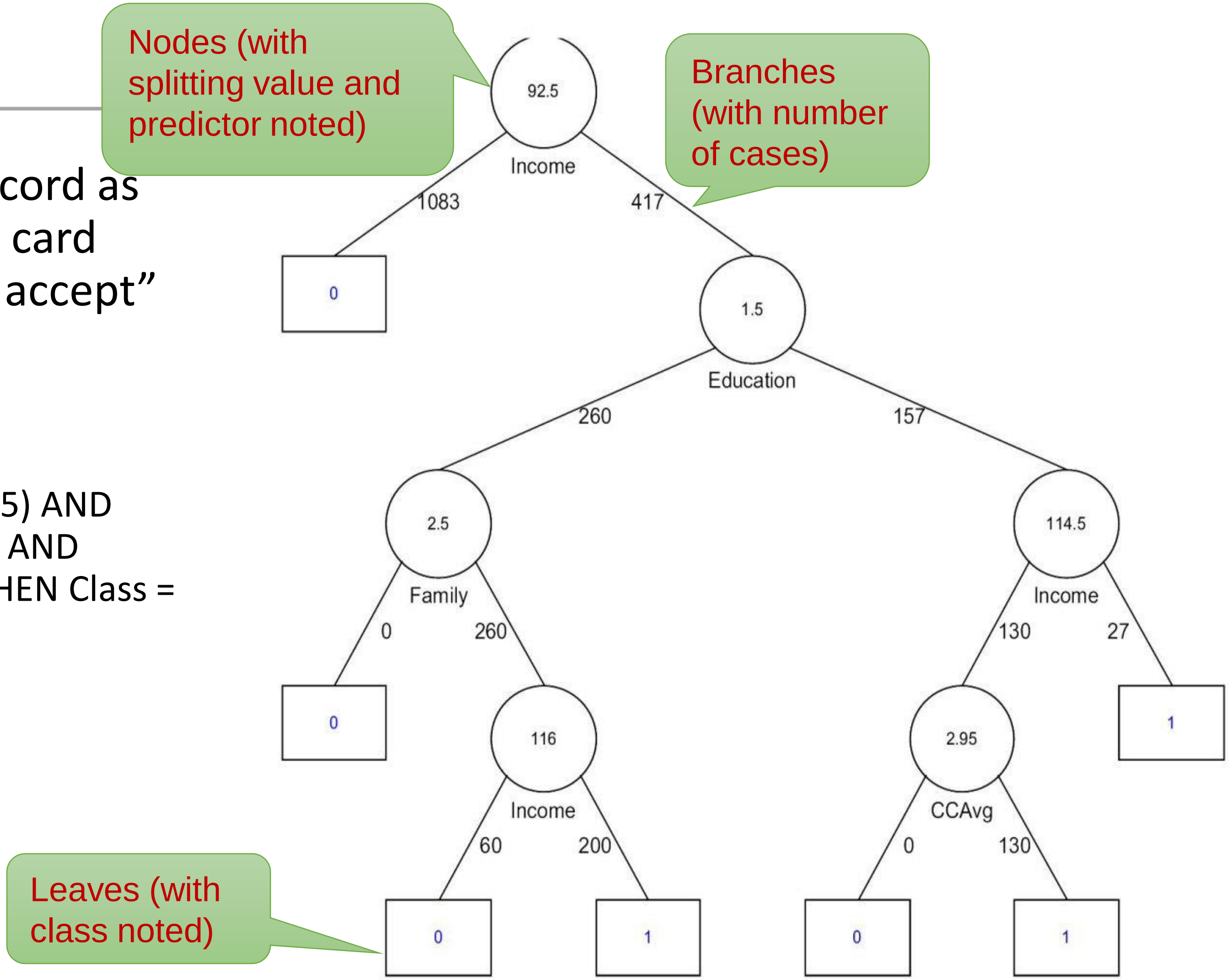
- Pick one of the predictor variables, x_i
- Pick a value of $x_{i,}$ say s_i, that divides the training data into two (not necessarily equal) portions
- Measure how "pure" or homogeneous each of the resulting portions are
  "Pure" = containing records of mostly one class
- Algorithm tries different values of $x_{i,}$ and s_i to maximize purity in initial split
- After you get a "maximum purity" split, repeat the process for a second split, and so on

Example

- Goal: classify a record as "will accept credit card offer" or "will not accept"

- Rule example:
  - "IF (Income > 92.5) AND (Education < 1.5) AND (Family <= 2.5) THEN Class = 0 (non-acceptor)

Tree Representation



Stopping Criteria and Leaf Labelling

- "Stopping" Criteria (when to stop splitting nodes?)
  - **Identical response**: All data points associated with the node are from the same class *c*
    - The node becomes a leaf node of class *c*
  - **Identical predictor**: There are no remaining attributes on which data points can further be partitioned
    - Use "majority voting" (of the class) to label this node
  - **No data**: no data point associated with the node
    - Use "majority voting" based on data points in the parent node to label this node
  - Tricky issue.

Example: Riding Mowers

- Goal: Classify 24 households as owning or not owning riding mowers
- Predictors = Income, Lot Size
- Target = ownership

Income	Lot_Size	Ownership
60.0	18.4	owner
85.5	16.8	owner
64.8	21.6	owner
61.5	20.8	owner
87.0	23.6	owner
110.1	19.2	owner
108.0	17.6	owner
82.8	22.4	owner
69.0	20.0	owner
93.0	20.8	owner
51.0	22.0	owner
81.0	20.0	owner
75.0	19.6	non-owner
52.8	20.8	non-owner
64.8	17.2	non-owner
43.2	20.4	non-owner
84.0	17.6	non-owner
49.2	17.6	non-owner
59.4	16.0	non-owner
66.0	18.4	non-owner
47.4	16.4	non-owner
33.0	18.8	non-owner
51.0	14.0	non-owner
63.0	14.8	non-owner

How to split

- Order records according to one variable, say lot size
- Find midpoints between successive values

  E.g. first midpoint is 14.4 (halfway between 14.0 and 14.8)
- Divide records into those with lotsize > 14.4 and those < 14.4
- After evaluating that split, try the next one, which is 15.4 (halfway between 14.8 and 16.0)

Note: Categorical Variables

- Examine all possible ways in which the categories can be split.

- E.g., categories A, B, C can be split 3 ways

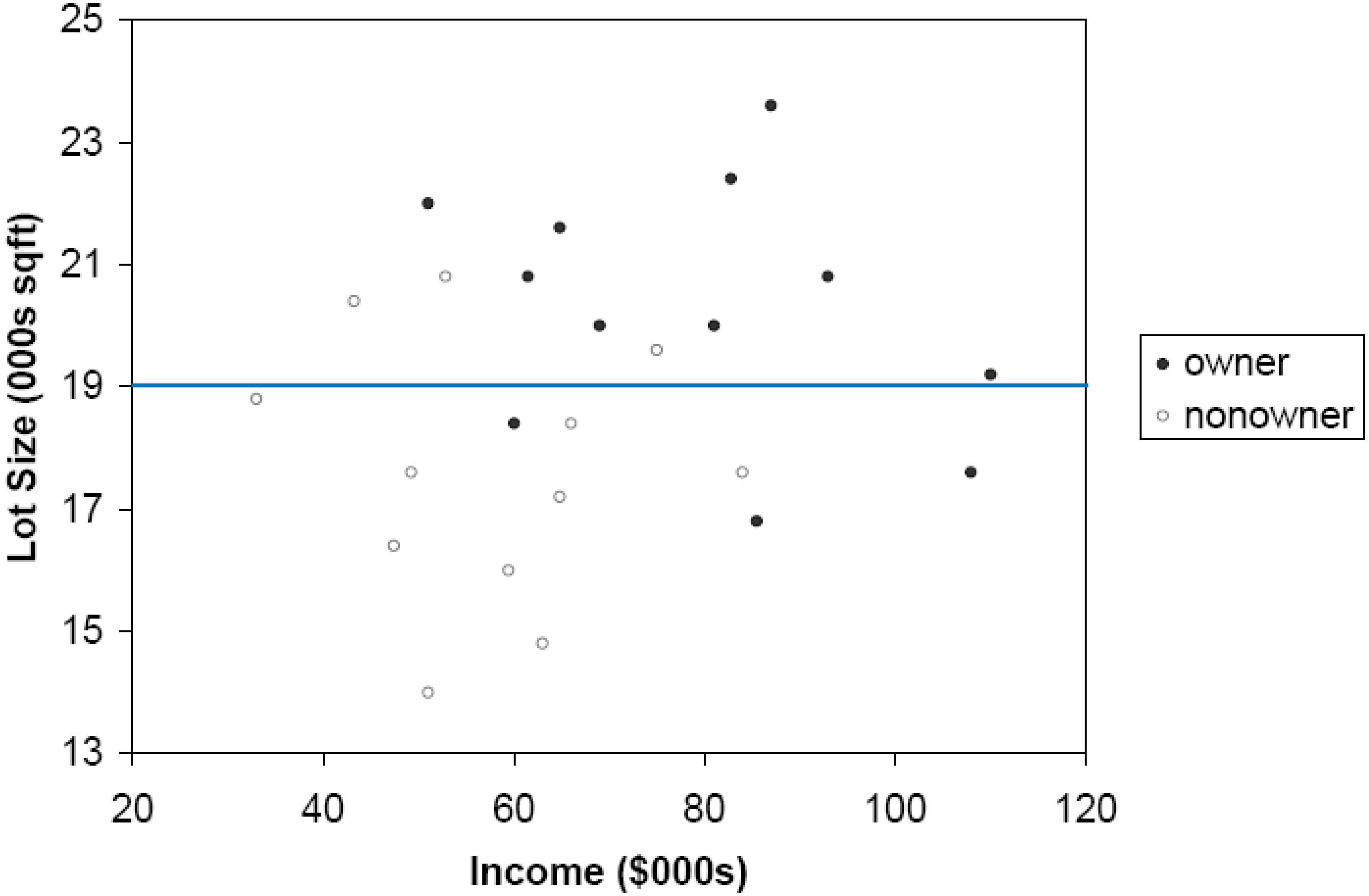
  {A} and {B, C}

  {B} and {A, C}

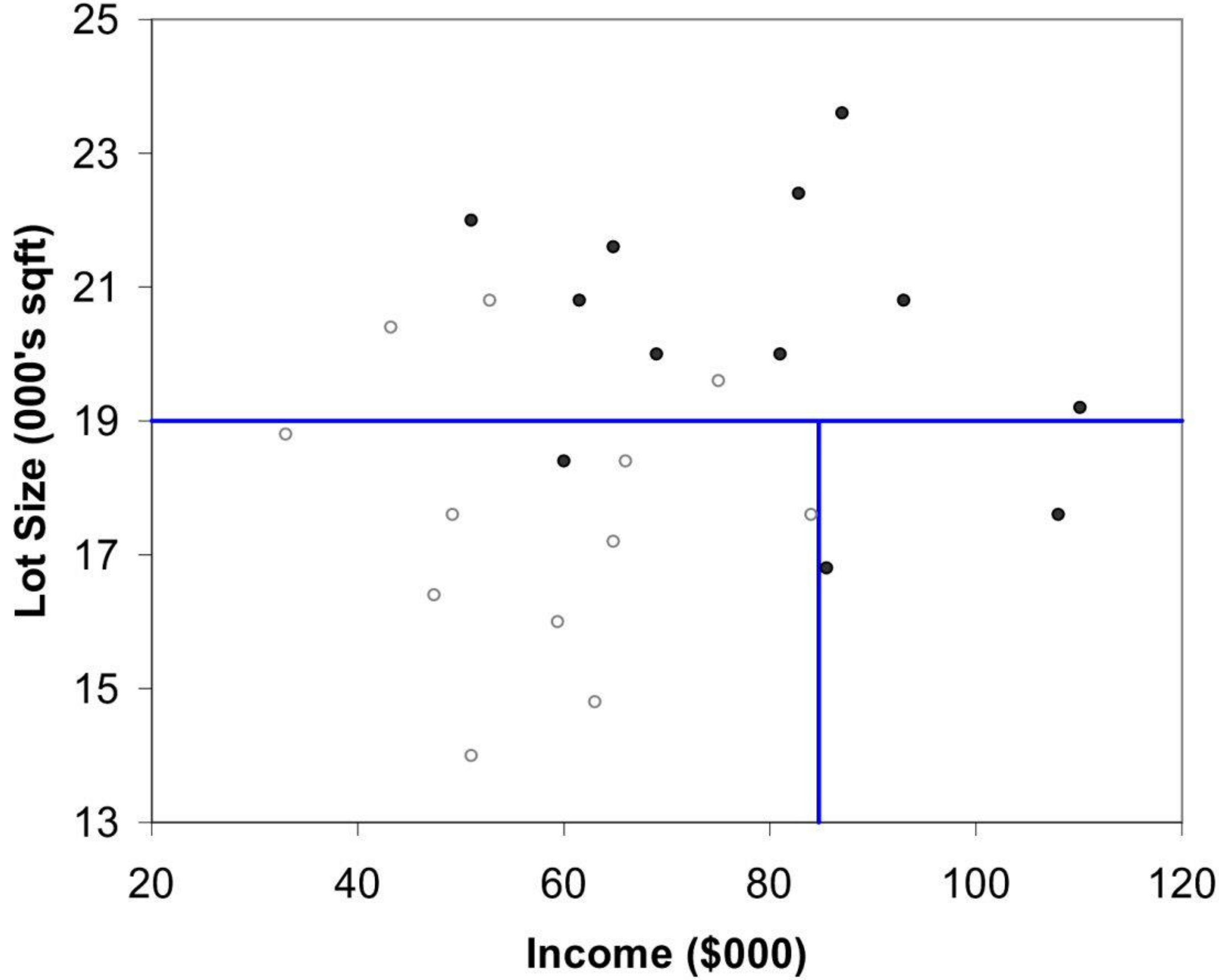
  {C} and {A, B}

- With many categories, # of splits becomes huge

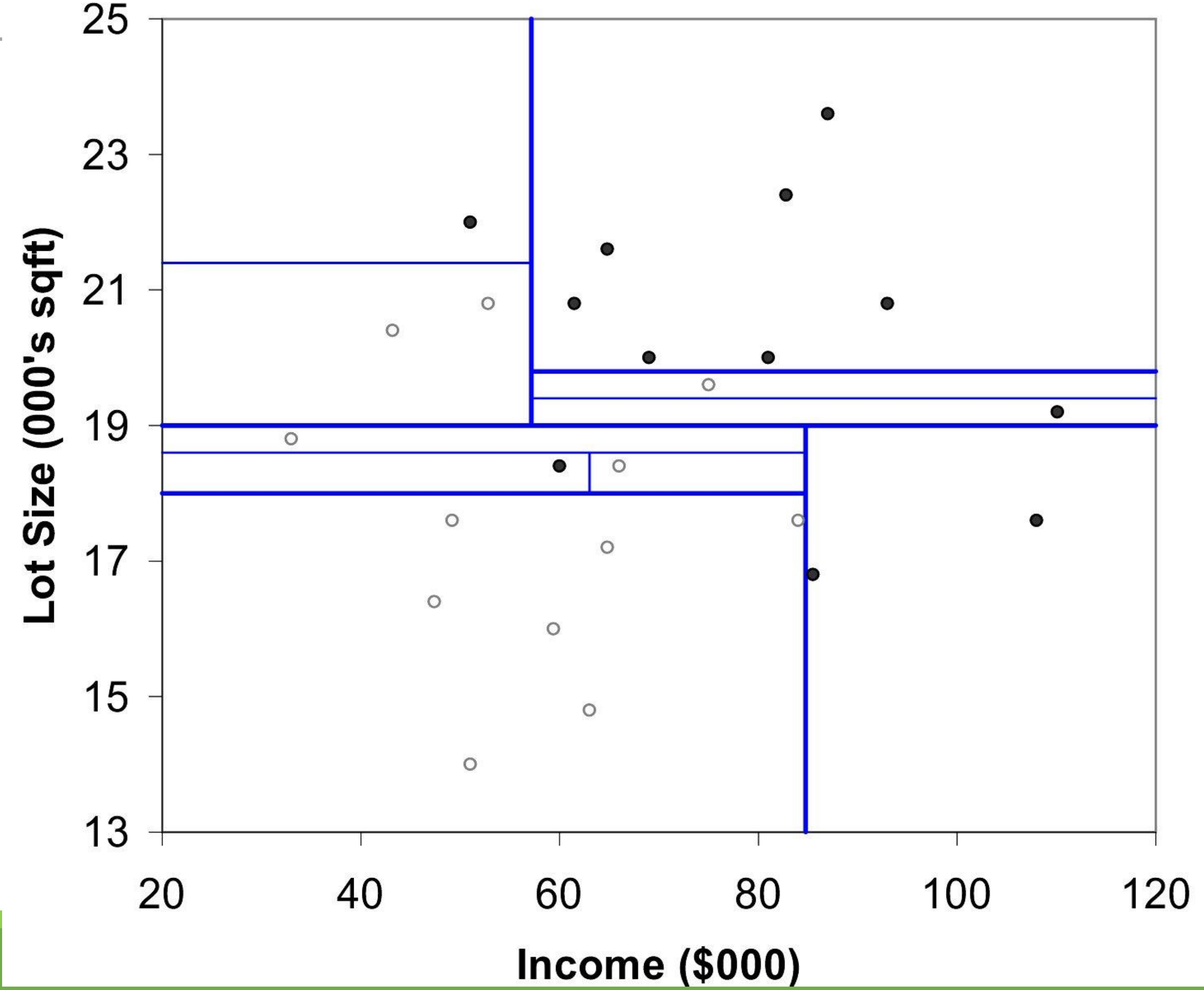
The first split: Lot Size = 19,000



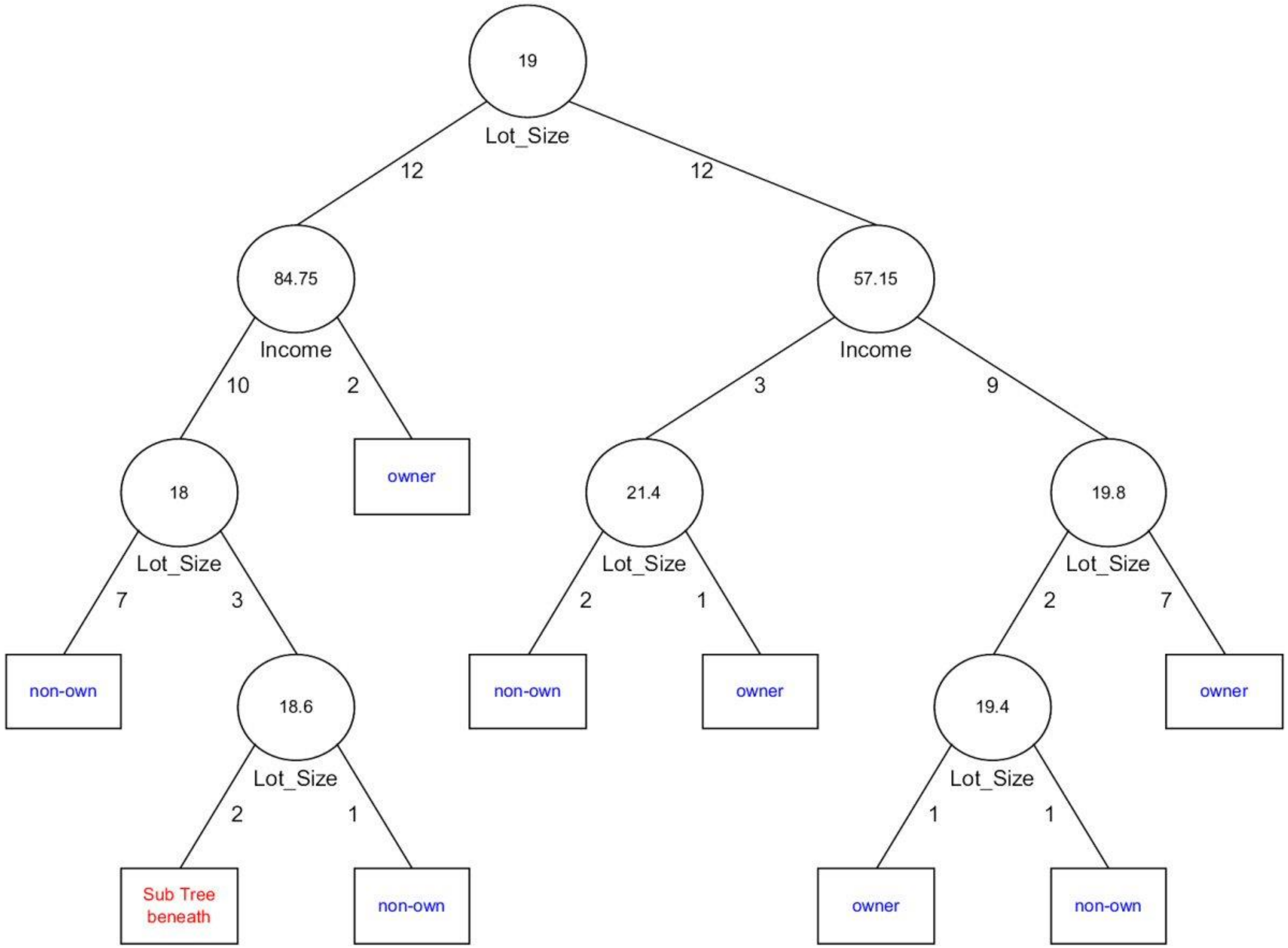
Second Split: Income = $84,000



After All Splits



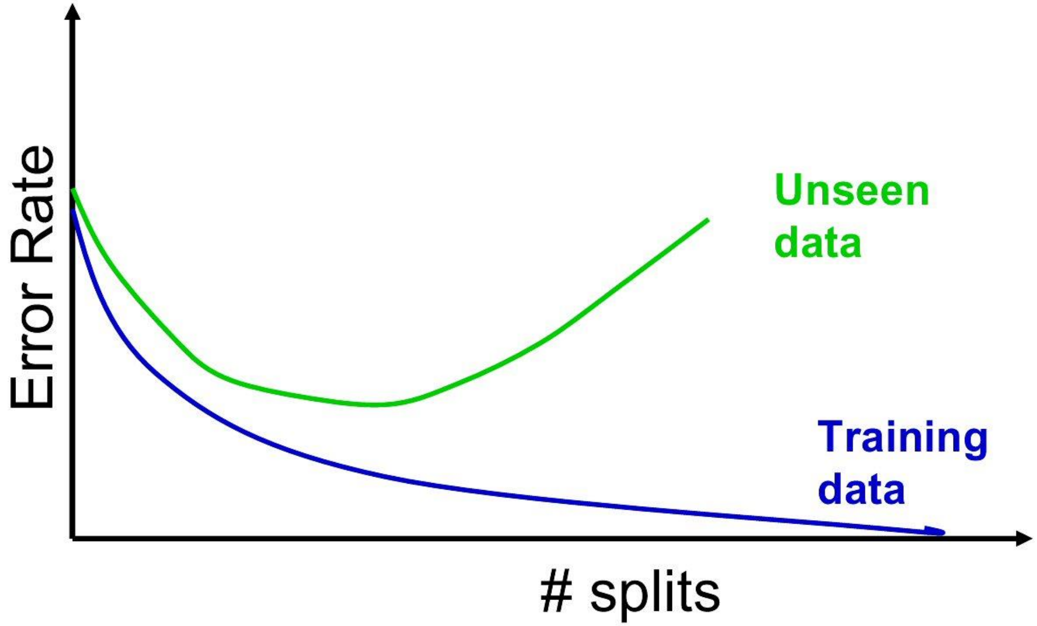
Tree after All Splits



Overfitting???

The Overfitting Problem

- Natural end of process is 100% purity in each leaf
- This overfits the data, which end up fitting noise in the data
- Overfitting leads to low predictive accuracy of new data



Tree Pruning

- Pruning is used to address the overfitting problem
  - Pre-pruning: halting the tree induction early
    - Using a threshold on attribute metrics
    - Using a threshold on the number of data points in each node
    - Using a threshold on the total number of tree nodes
  - Post-pruning
    - Performed after building the entire tree
    - Calculate expected error rates with and without a node, choose the better
- Pruning criteria:
  - Generally based on penalizing high misclassification rates and large trees.

Using Validation Error to Prune

- Pruning process yields a set of trees of different sizes and associated error rates
- Two trees of interest:
  - Minimum error tree
    - Has lowest error rate on validation data
  - Best pruned tree
    - Smallest tree within one std. error of min. error
    - This adds a bonus for simplicity/parsimony

α

Which branch to cut at each stage of pruning?

$$CC(T) = Err(T) + \alpha L(T)$$

CC(T) = cost complexity of a tree

Err(T) = proportion of misclassified records

α = penalty factor attached to tree size (set by user)

- Among trees of given size, choose the one with lowest CC
- Do this for each size of tree (stage of pruning)

Regression Trees

Regression Trees for Prediction

- Used with continuous outcome variable
- Procedure similar to classification tree
- Many splits attempted, choose the one that minimizes impurity

Differences from Classification Tree

- Prediction is computed as the **average** of numerical target variable in the rectangle (in Classification Tree it is majority vote)
- Impurity measured by **sum of squared deviations** from leaf mean
- Performance measured by RMSE (root mean squared error)

Advantages of trees

- Easy to use, understand
- Produce rules that are easy to interpret & implement
- Variable selection & reduction is automatic
- Do not require the assumptions of statistical models
- Can work without extensive handling of missing data

Disadvantages

- May not perform well where there is structure in the data that is not well captured by horizontal or vertical splits

- Since the process deals with one variable at a time, no way to capture interactions between variables

- The resulting trees may change drastically with small change in the data.

Random Forests and Boosted Trees

- Predictions from many trees are combined
- Very good predictive performance, better than single trees (often the top choice for predictive modeling)
- Cost: loss of rules you can explain implement (since you are dealing with many trees, not a single tree)
  - However, RF does produce “variable importance scores,” (using information about how predictors reduce Gini scores over all the trees in the forest)

Random Forests (library `randomForest`)

1. Draw multiple bootstrap resamples of cases from the data
2. For each resample, use a random subset of predictors and produce a tree
3. Combine the predictions/classifications from all the trees (the “forest”)
   - Voting for classification
   - Averaging for prediction

Boosted Trees

Random forests and boosted trees are really the same models; the difference arises from how we train them.

Boosted Trees:

1. Fit a single tree
2. Draw a bootstrap sample of records with higher selection probability for misclassified records
3. Fit a new tree to the bootstrap sample
4. Repeat steps 2 & 3 multiple times
5. Use weighted voting (classification) or averaging (prediction) with heavier weights for later trees