

HOMEWORK #1 REMINDER

- Out 8/28 on Piazza, Due **9/12 @ 11:59 PM ET** on Gradescope
- 4 questions
 - Q1-Q2: Get familiar with Python
 - Numerical programming (Numpy)
 - Dataset loading and visualization (Pandas and other libraries)
 - Q3-Q4: kNN
 - Implement kNN (use Numpy)
 - Evaluate kNN (use sklearn)



HOMEWORK #1: FAQ (FROM PIAZZA)

- Submit ancillary code for partial credit (e.g., those associated with the non-coding answers) [[Q14](#)]
- Q3: knn implementation
 - Should generalize to any number of features [[Q10](#)]
 - Does not need to be the most efficient optimization but should avoid nested for-loops [[Q11](#)]

COURSE LOGISTICS

- Vote on [Piazza](#) for Midterm Date (11/8 vs 11/15)
- [Python workshop material](#) posted

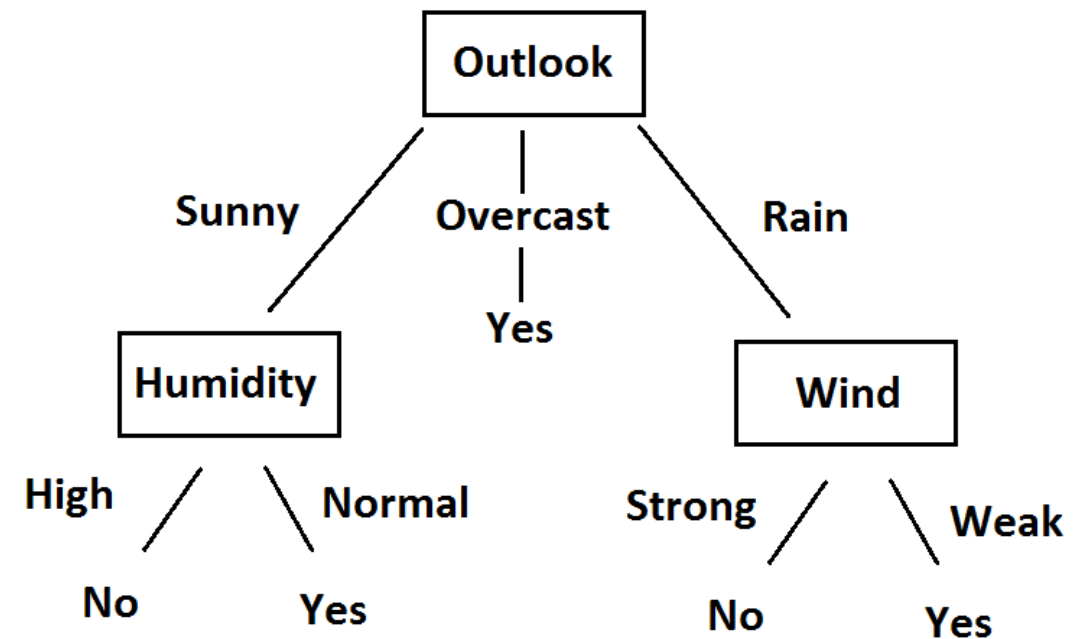
DECISION TREES (PART II)

CS 334: Machine Learning

REVIEW: DECISION TREE

A tree structure

- Each internal (decision) node represents a test on a feature, each branch represents a value
- Each leaf node represents a class label
- Each path represents a classification/decision rule following successive choices



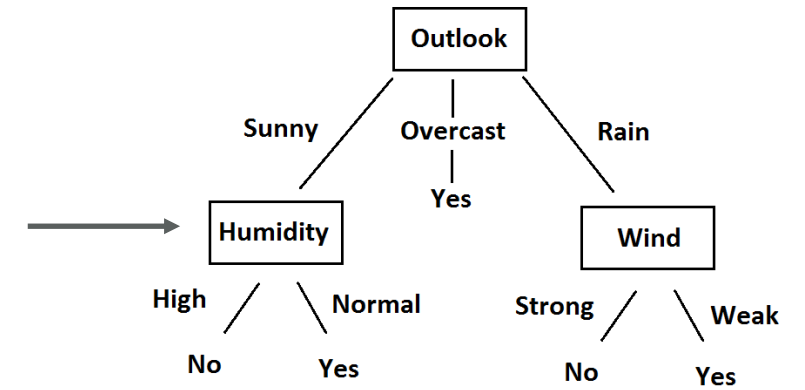
<https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96>

<https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/>

REVIEW: DECISION TREE

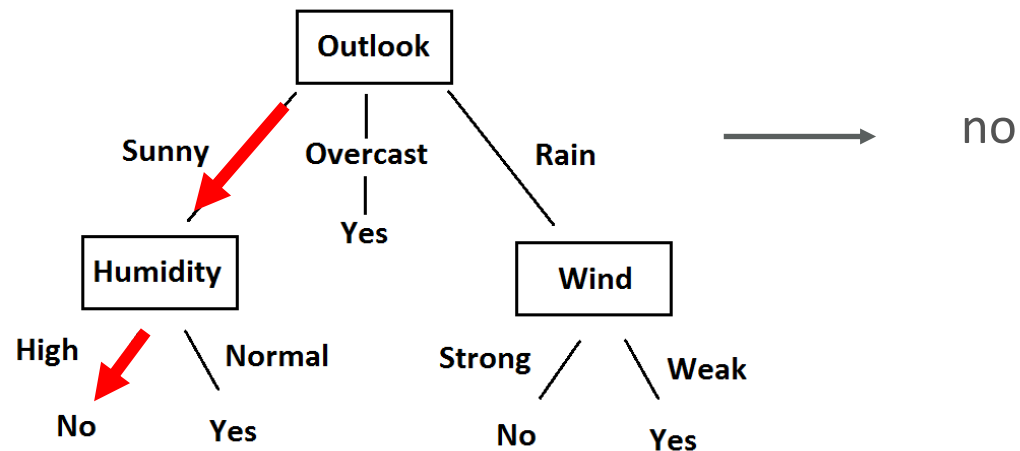
- **Training:** Build a decision tree from training data

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



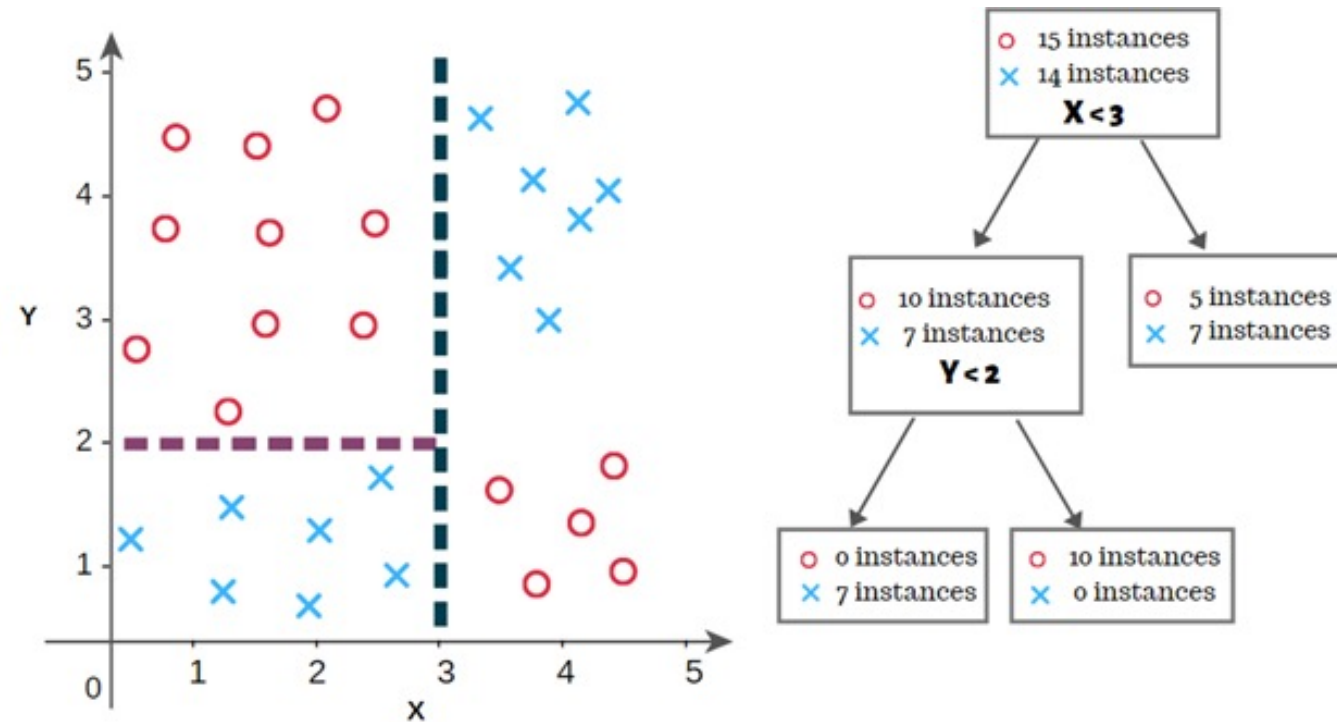
- **Prediction:** Given a new data point, find the path using its features and predict the label at the leaf node

(Sunny, Mild, High, Weak)



REVIEW: HOW TO LEARN THE TREE?

- Recursively create a tree node that splits the current data region into two subregions
- How to choose the node (splits)?
- When to stop the tree (how big to grow)?

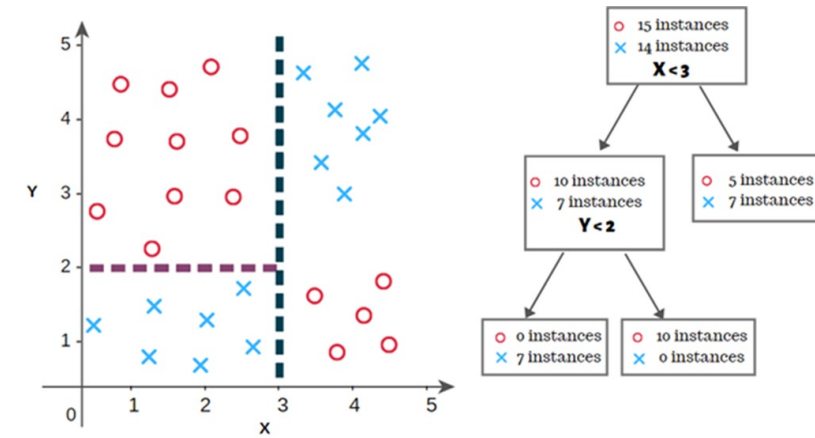


DECISION TREE: TRAINING (C4.5 ALGORITHM)




Algorithm 1.1 C4.5(D)

Input: an attribute-valued dataset D

```
1: Tree = {}
2: if  $D$  is "pure" OR other stopping criteria met then ← Stop criteria
3:   terminate
4: end if
5: for all attribute  $a \in D$  do ← For all possible splitting points
6:   Compute information-theoretic criteria if we split on  $a$  ← Scoring criteria
7: end for
8:  $a_{best}$  = Best attribute according to above computed criteria ← Select best split attribute
9: Tree = Create a decision node that tests  $a_{best}$  in the root
10:  $D_v$  = Induced sub-datasets from  $D$  based on  $a_{best}$ 
11: for all  $D_v$  do ← Recurse on subregions
12:   Tree $v$  = C4.5( $D_v$ )
13:   Attach Tree $v$  to the corresponding branch of Tree
14: end for
15: return Tree
```



REVIEW: CHOOSING A GOOD SPLIT

- Idea: Use label counts at the node to define probability distributions to measure uncertainty
- Deterministic — good (all positive or all negative) 
- Uniform — bad (all classes have equal probability) 
- What about in-between? 
- Common metrics
 - Information entropy and information gain (ID3/C4.5)
 - Gini index (CART)

A LITTLE BIT OF INFORMATION THEORY

Flip three different coins (4 sides):

- Sequence 1: A A A A A A A A A A A ... (deterministic) 0 bit/letter
- Sequence 2: A B C D B A D C B C D A ... (uniform: $P_A = \frac{1}{4}$ $P_B = \frac{1}{4}$ $P_C = \frac{1}{4}$ $P_D = \frac{1}{4}$)
000110110100111001101101 A:00 B:01 C:10 D:11 2 bits/letter
- Sequence 3: A A B C B C A A B A C A ... (biased: $P_A = \frac{1}{2}$ $P_B = \frac{1}{4}$ $P_C = \frac{1}{4}$ $P_D = 0$)

Can we use less than 2 bits for sequence 3?

A LITTLE BIT OF INFORMATION THEORY

Flip three different coins (4 sides):

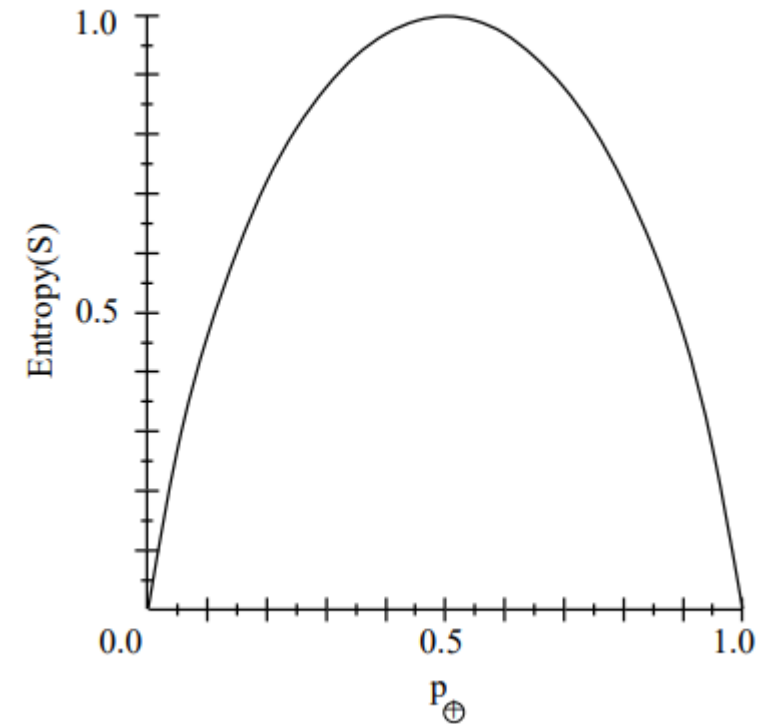
- Sequence 1: A A A A A A A A A A A ... (deterministic) 0 bit/letter
- Sequence 2: A B C D B A D C B C D A ... (uniform: $P_A = \frac{1}{4}$ $P_B = \frac{1}{4}$ $P_C = \frac{1}{4}$ $P_D = \frac{1}{4}$)
000110110100111001101101 A:00 B:01 C:10 D:11 2 bits/letter
- Sequence 3: A A B C B C A A B A C A ... (biased: $P_A = \frac{1}{2}$ $P_B = \frac{1}{4}$ $P_C = \frac{1}{4}$ $P_D = 0$)
0 0 101110110 0 10 0 11 0 A:0 B:10 C: 11 1.5bits/letter

ENTROPY

- Measure of the information content (uncertainty) of a random variable, i.e. expected number of bits needed to encode the variable

$$H(\mathbf{X}) = - \sum_{k=1}^K p(X = k) \log_2 p(X = k)$$

- High entropy \rightarrow uniform-like distribution / flat histogram
- Low entropy \rightarrow biased (peaks and valleys) with more predictable values

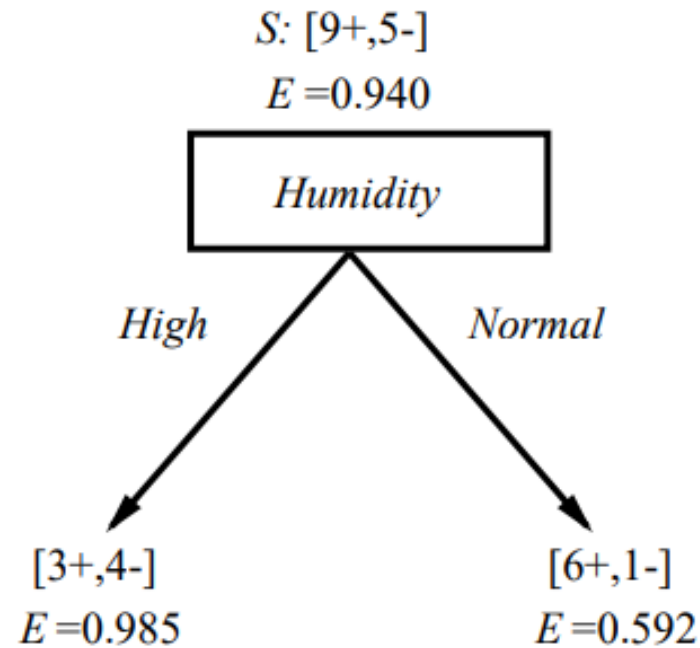


$$Entropy(S) \equiv -p_{+} \log_2 p_{+} - p_{-} \log_2 p_{-}$$

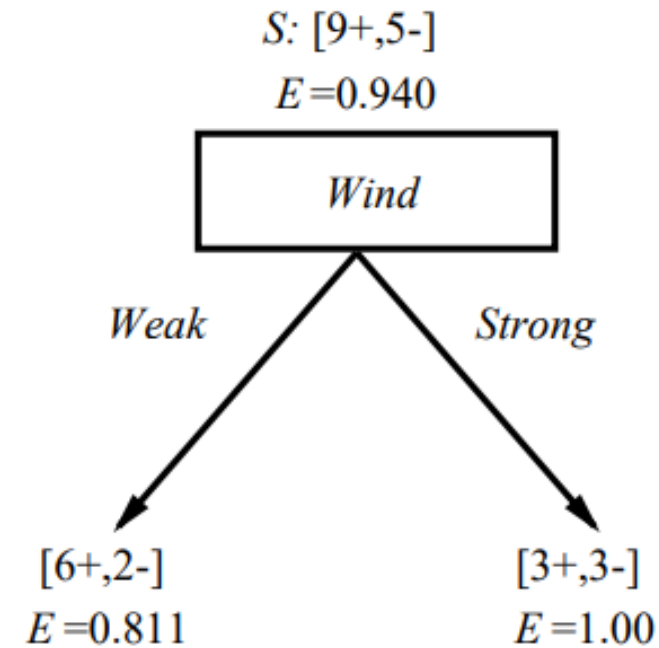
ATTRIBUTE SELECTION: INFORMATION GAIN

Difference of entropy before and after the split

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



$$\begin{aligned} Gain(S, Humidity) &= .940 - (7/14).985 - (7/14).592 \\ &= .151 \end{aligned}$$

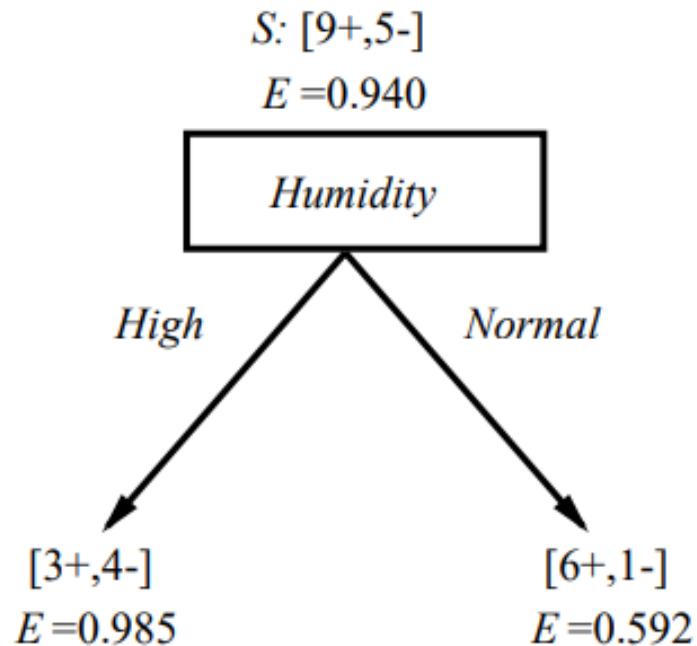


$$\begin{aligned} Gain(S, Wind) &= .940 - (8/14).811 - (6/14)1.0 \\ &= .048 \end{aligned}$$

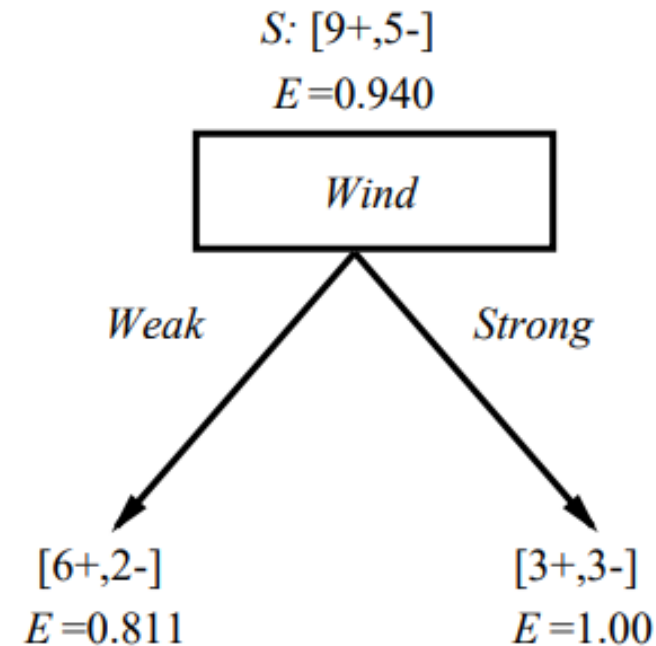
Which attribute offers a better split?

INFORMATION GAIN

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



$$\begin{aligned} Gain(S, Humidity) &= .940 - (7/14).985 - (7/14).592 \\ &= .151 \end{aligned}$$



$$\begin{aligned} Gain(S, Wind) &= .940 - (8/14).811 - (6/14)1.0 \\ &= .048 \end{aligned}$$

Choose split with larger information gain
Or choose split with lower entropy after split

CHOOSING A GOOD SPLIT

- Idea: Use counts at the node to define probability distributions to measure uncertainty

- Deterministic — good (all positive or all negative)



- Uniform — bad (all classes have equal probability)



- What about in-between?



- Common metrics

- Information entropy and information gain (ID3/C4.5)

- **Gini index (CART)**

GINI INDEX

- Measure of how often a randomly chosen element from set would be incorrectly labeled if it was randomly labeled based on the labels in subset

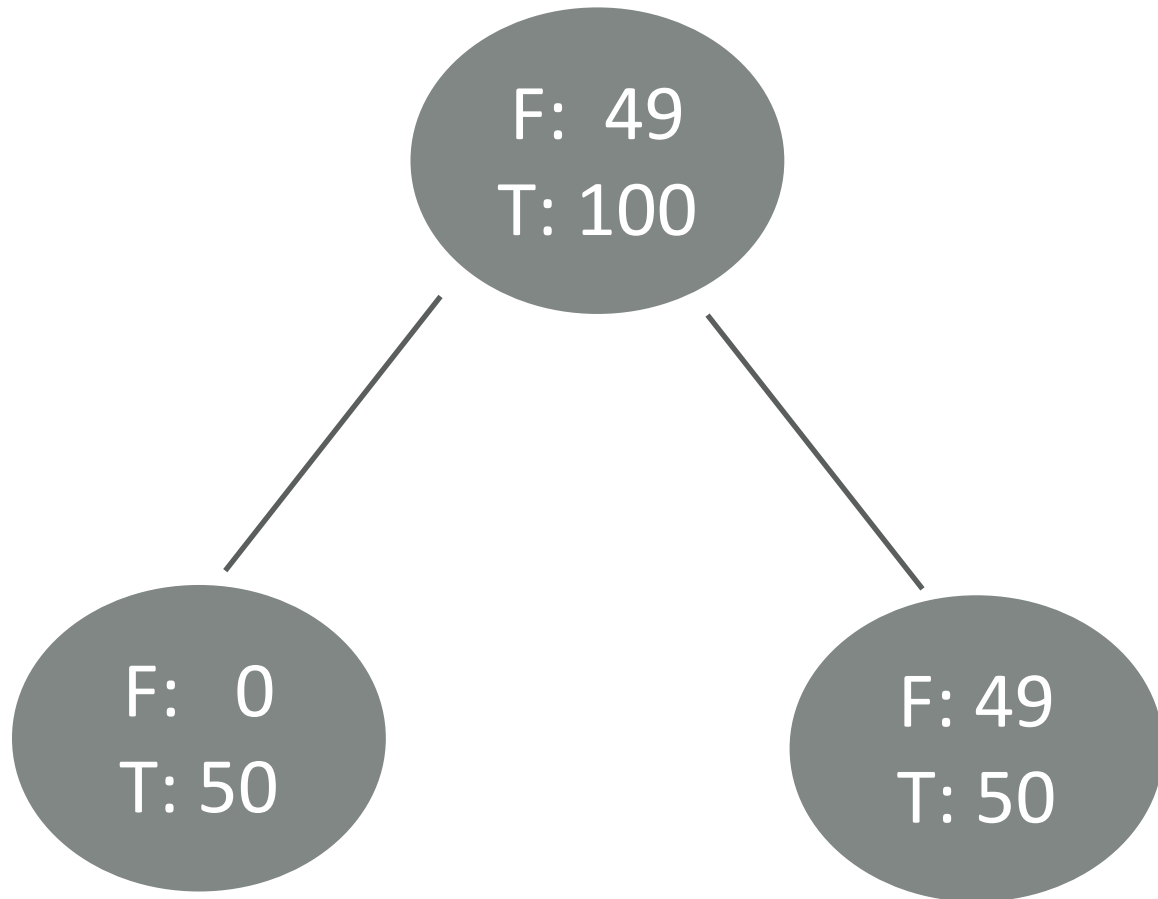
$$\sum_{k=1}^K p(X = k)(1 - p(X = k))$$



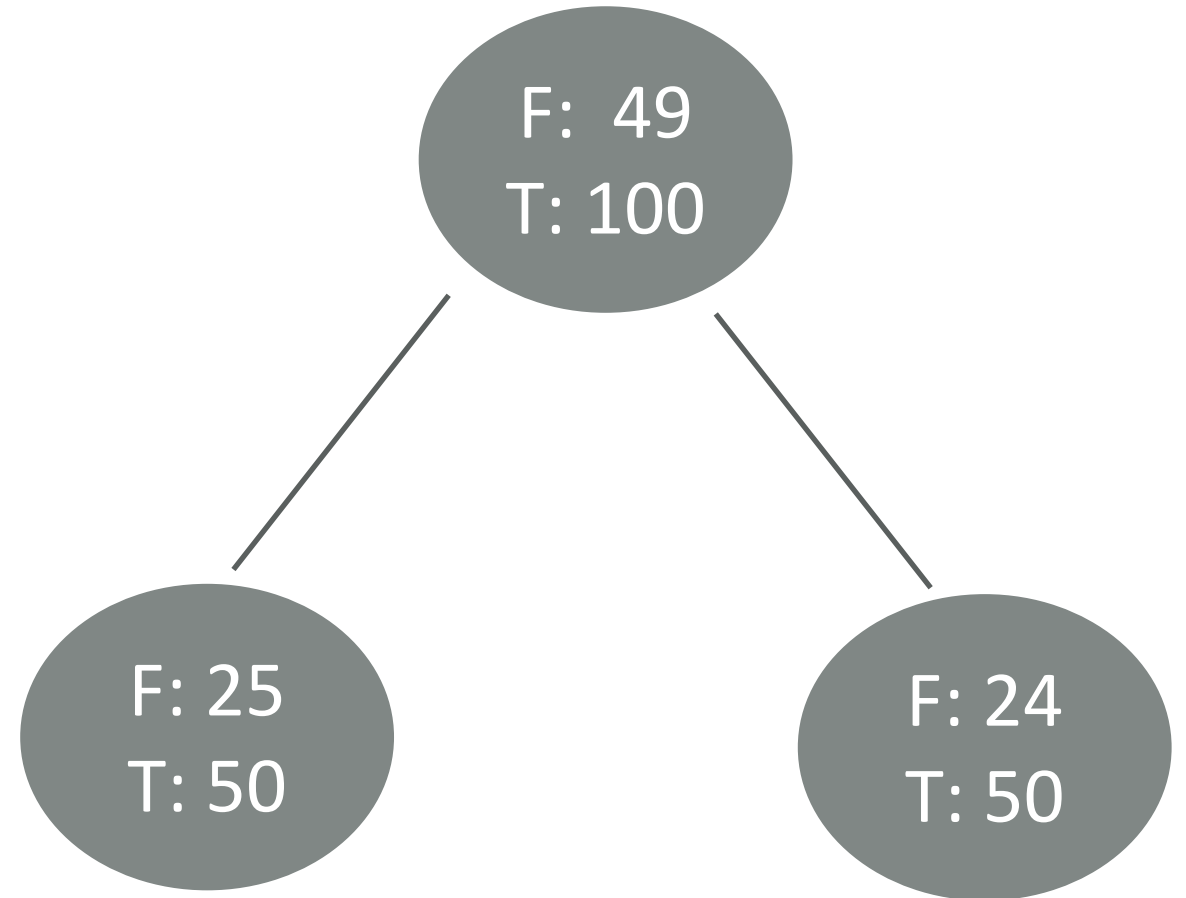
- Also known as measure of node impurity
- Used by the CART (classification and regression tree) algorithm

CAN WE USE CLASSIFICATION ERROR?

Split on X_1



Split on X_2



ENTROPY VS GINI VS CLASSIFICATION ERROR

Entropy (a way to measure impurity):

$$Entropy = - \sum_j p_j \log_2 p_j$$

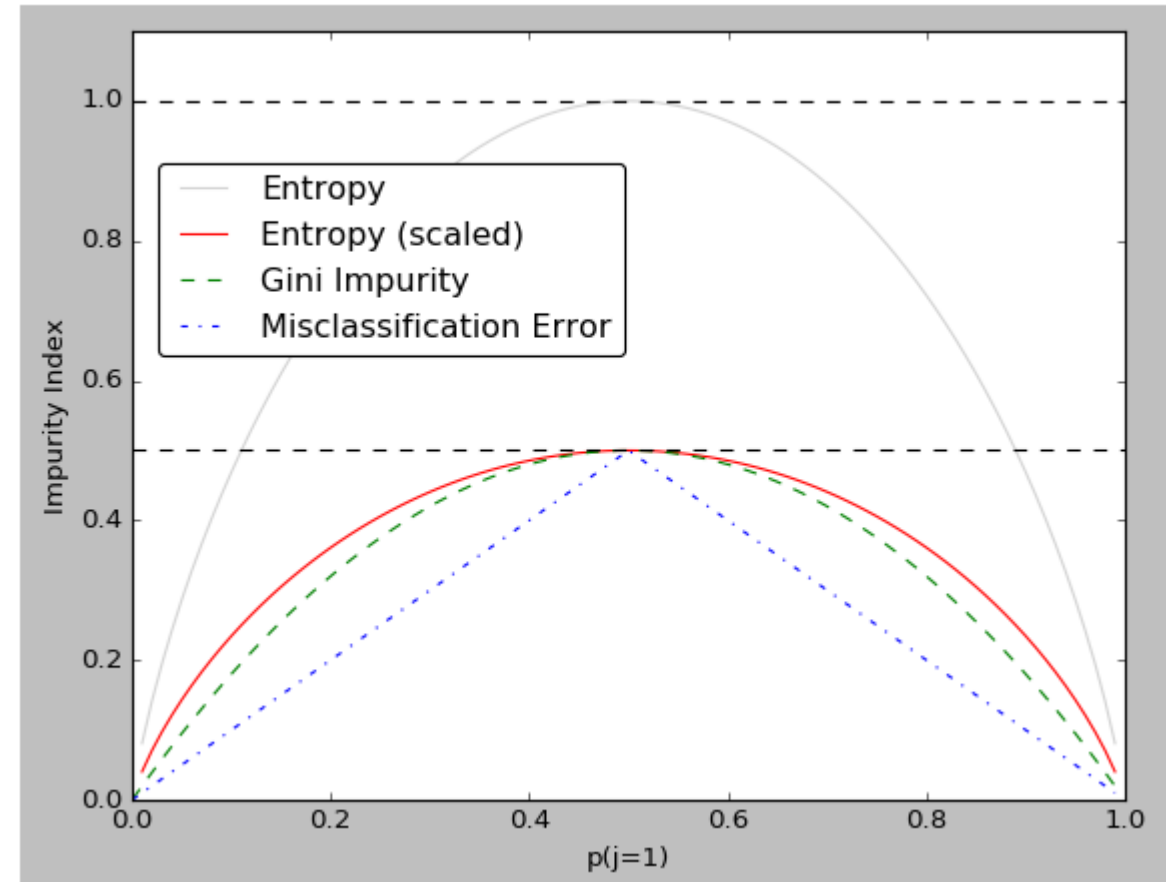
Gini index (a criterion to minimize the probability of misclassification):

$$Gini = 1 - \sum_j p_j^2$$

Classification Error:

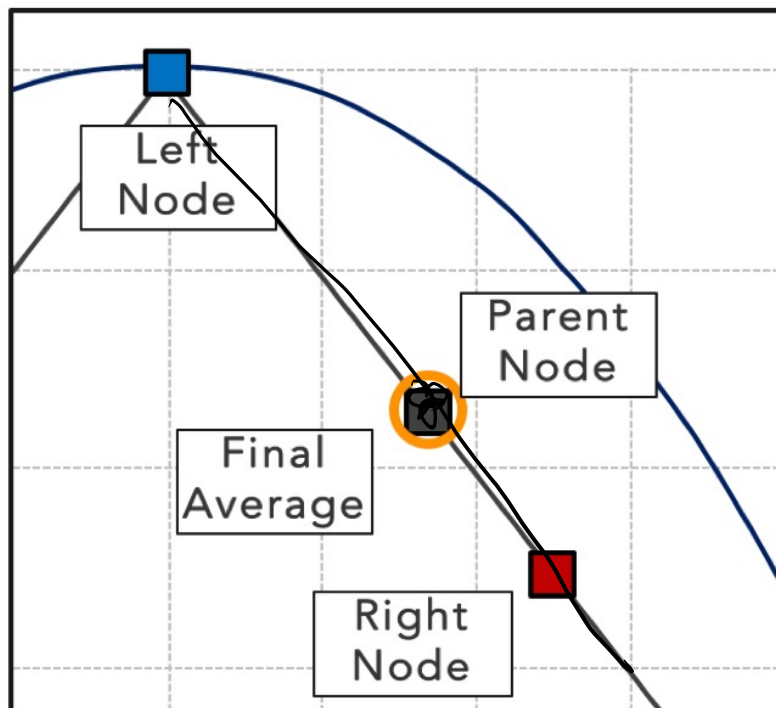
$$ClassificationError = 1 - \max p_j$$

where p_j is the probability of class j .

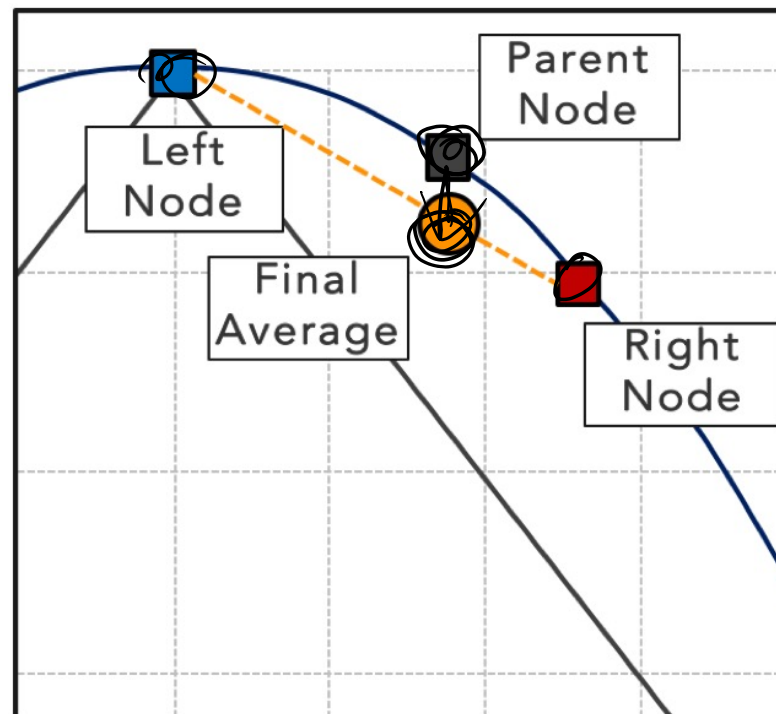


WHY DOES IT MATTER?

Classification Error



Information Gain

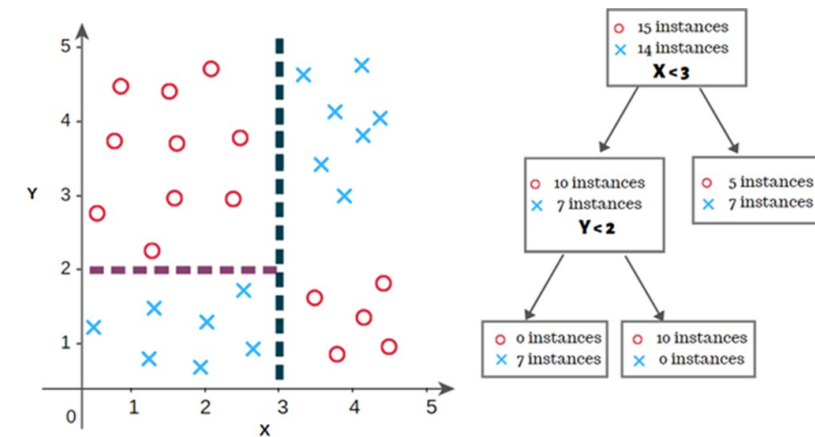


DECISION TREE: TRAINING (C4.5 ALGORITHM)

Algorithm 1.1 C4.5(D)

Input: an attribute-valued dataset D

```
1: Tree = {}
2: if  $D$  is "pure" OR other stopping criteria met then ← Stop criteria
3:   terminate
4: end if
5: for all attribute  $a \in D$  do ← For all possible splitting points
6:   Compute information-theoretic criteria if we split on  $a$  ← Scoring criteria
7: end for
8:  $a_{best}$  = Best attribute according to above computed criteria ← Select best split attribute
9: Tree = Create a decision node that tests  $a_{best}$  in the root
10:  $D_v$  = Induced sub-datasets from  $D$  based on  $a_{best}$ 
11: for all  $D_v$  do ← Recurse on subregions
12:   Tree $_v$  = C4.5( $D_v$ )
13:   Attach Tree $_v$  to the corresponding branch of Tree
14: end for
15: return Tree
```

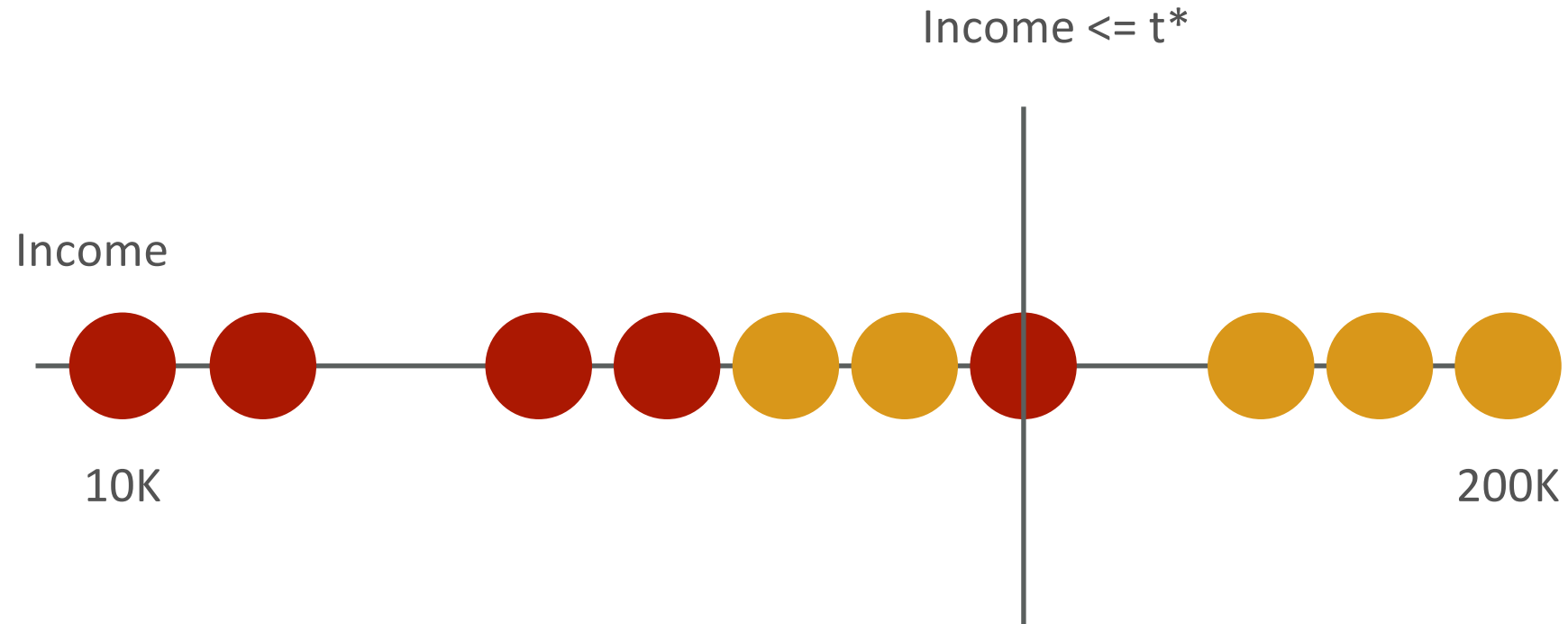


POSSIBLE SPLIT POINTS

- Real-valued features
- Categorical features

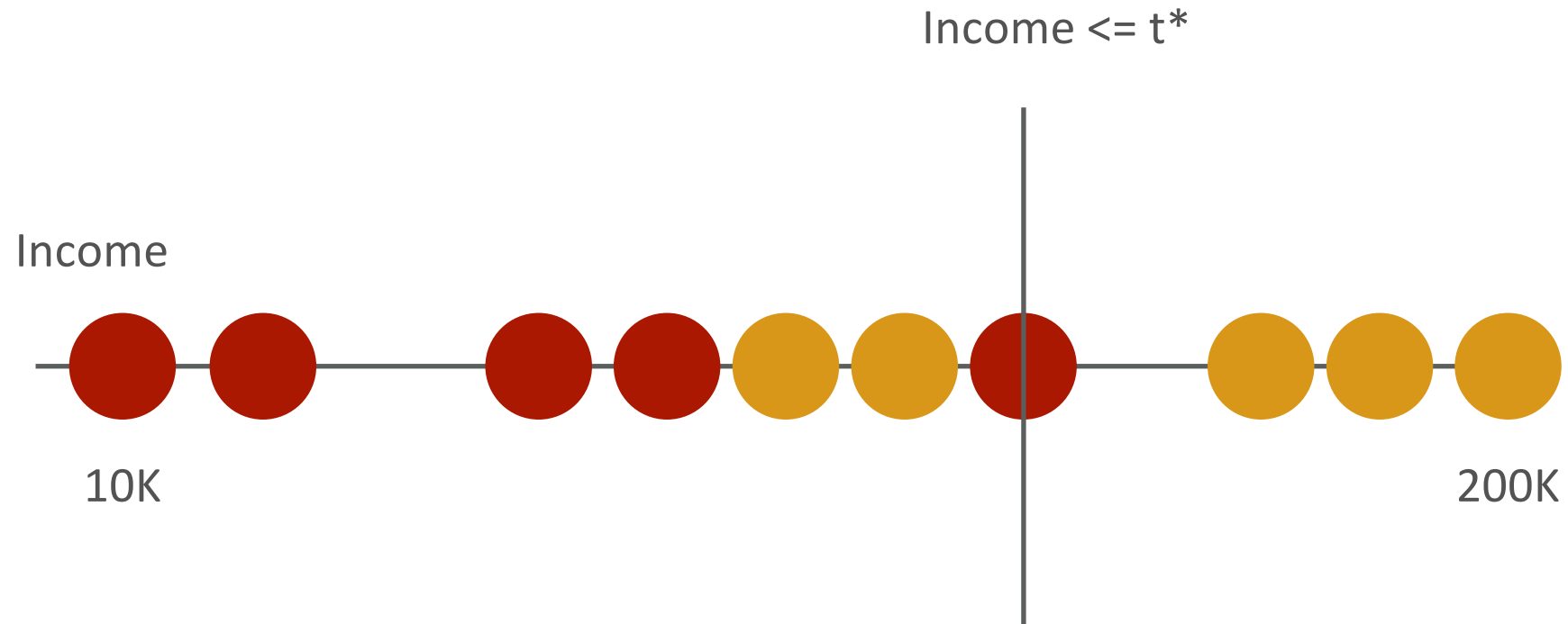
Income	Term	Credit	Y
105K	3	Excellent	Safe
112K	5	Good	Risky
73K	3	Fair	Safe
69K	5	Excellent	Safe
217K	3	Excellent	Risky
120K	5	Good	Safe
64K	3	Fair	Risky
340K	5	Excellent	Safe
60K	3	Good	Risky

REAL-VALUED FEATURES



Aren't there infinite possible values of t ?

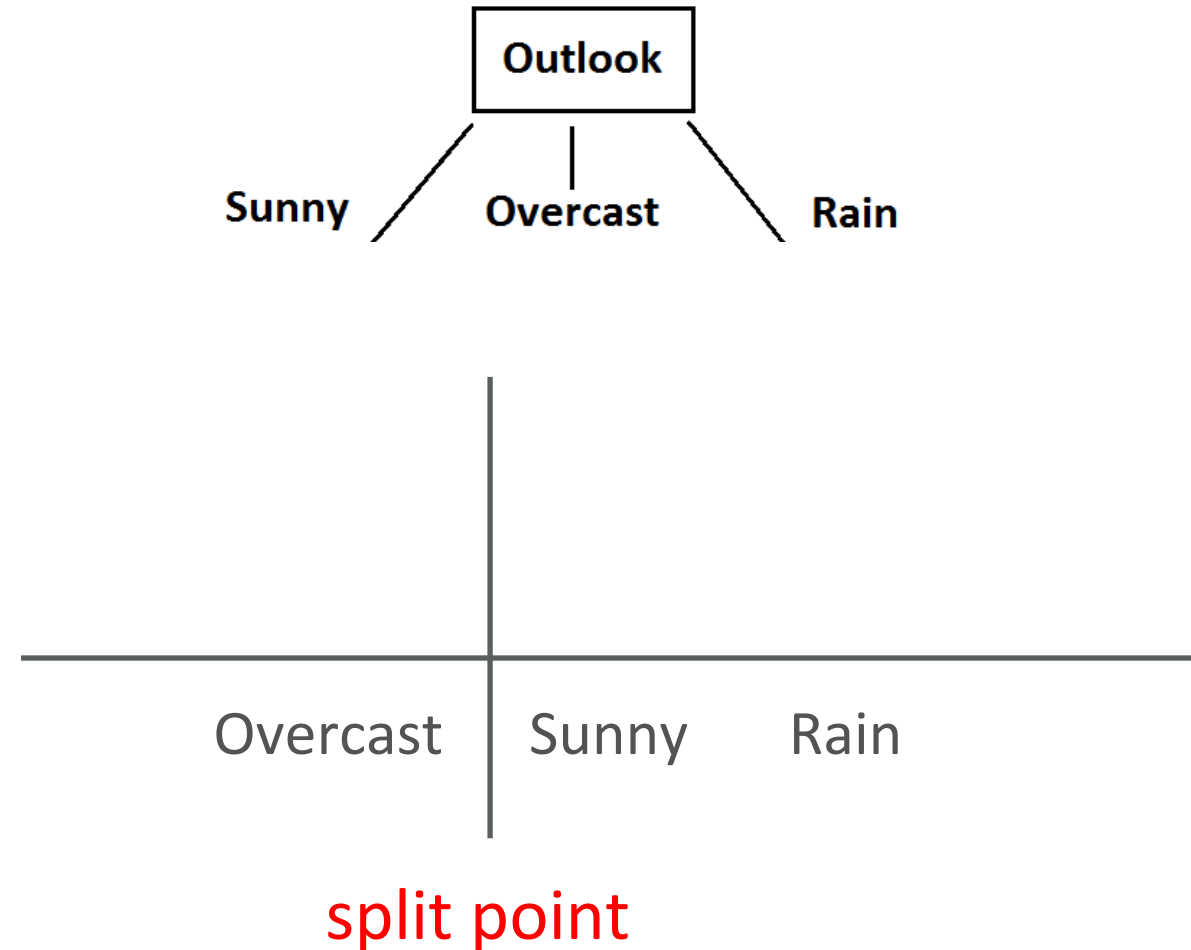
REAL-VALUED FEATURES



sort the training data points, use each value t as the threshold

CATEGORICAL FEATURES

- Multiway split:
 - not a good general strategy
- Binary split:
 - **Order** the attribute values based on proportion falling in an outcome class
 - Find the best **split point**
- Convert to numerical attributes



CATEGORICAL FEATURES: ENCODING

- Option 1: Assign an integer value
 - Example: Cold, mild, hot temperature => 1, 2, 3
- Option 2: One-hot encoding where each value becomes a binary variable
 - Example: Temp_cold, Temp_mild, Temp_hot

EXAMPLE: ONE-HOT ENCODING

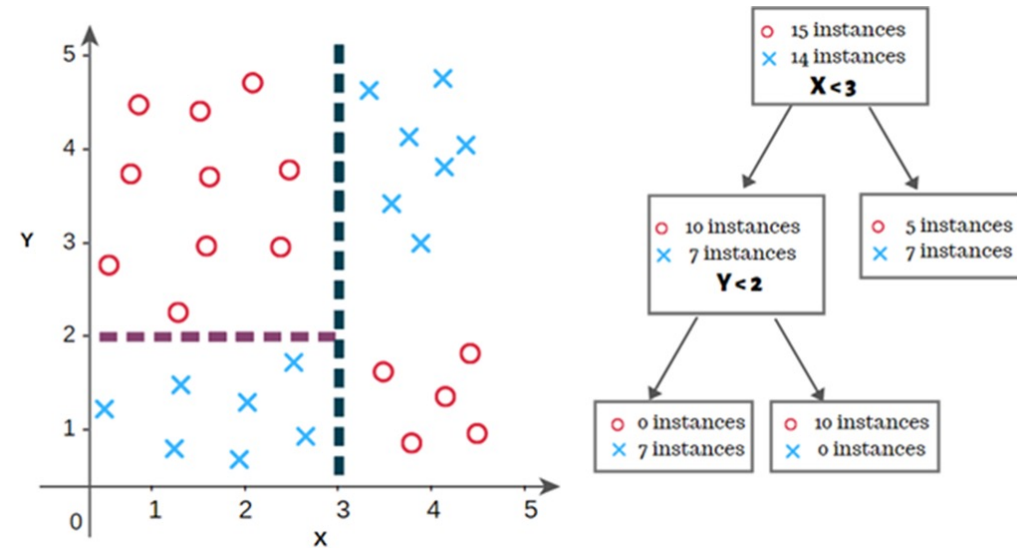
	Outlook	Temp	Humidity	Wind	Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes



	x0_Overcast	x0_Rain	x0_Sunny	x1_Cool	x1_Hot	x1_Mild	x2_High	x2_Normal	x3_Strong	x3_Weak
0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
1	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
2	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
3	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0
4	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0

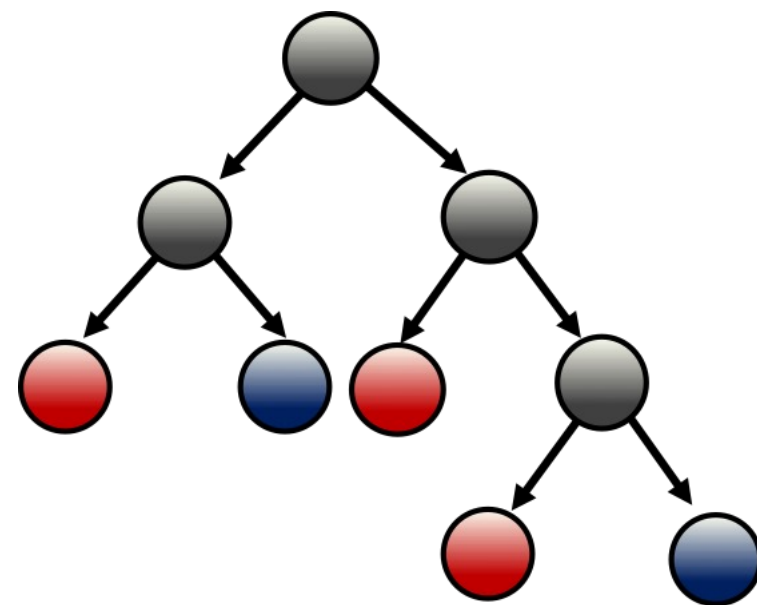
HOW TO GROW (BUILD) TREES?

- How to learn the best tree?
- How to choose the node (splits)?
- When to stop the tree (how big to grow)?

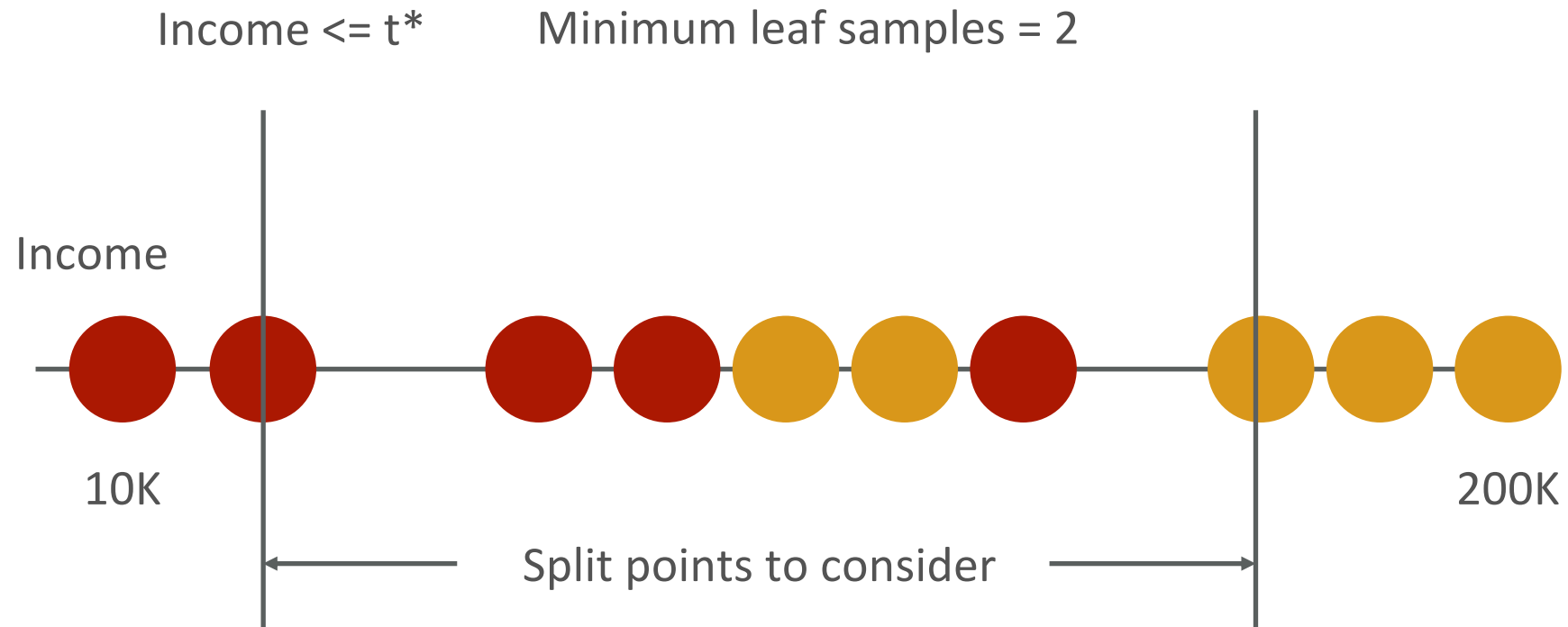


STOPPING CRITERION

- No more features
- 100% node purity or minimum impurity
- Minimum samples
- **Minimum leaf samples**
- **Maximum tree depth**
- Others: impurity decrease

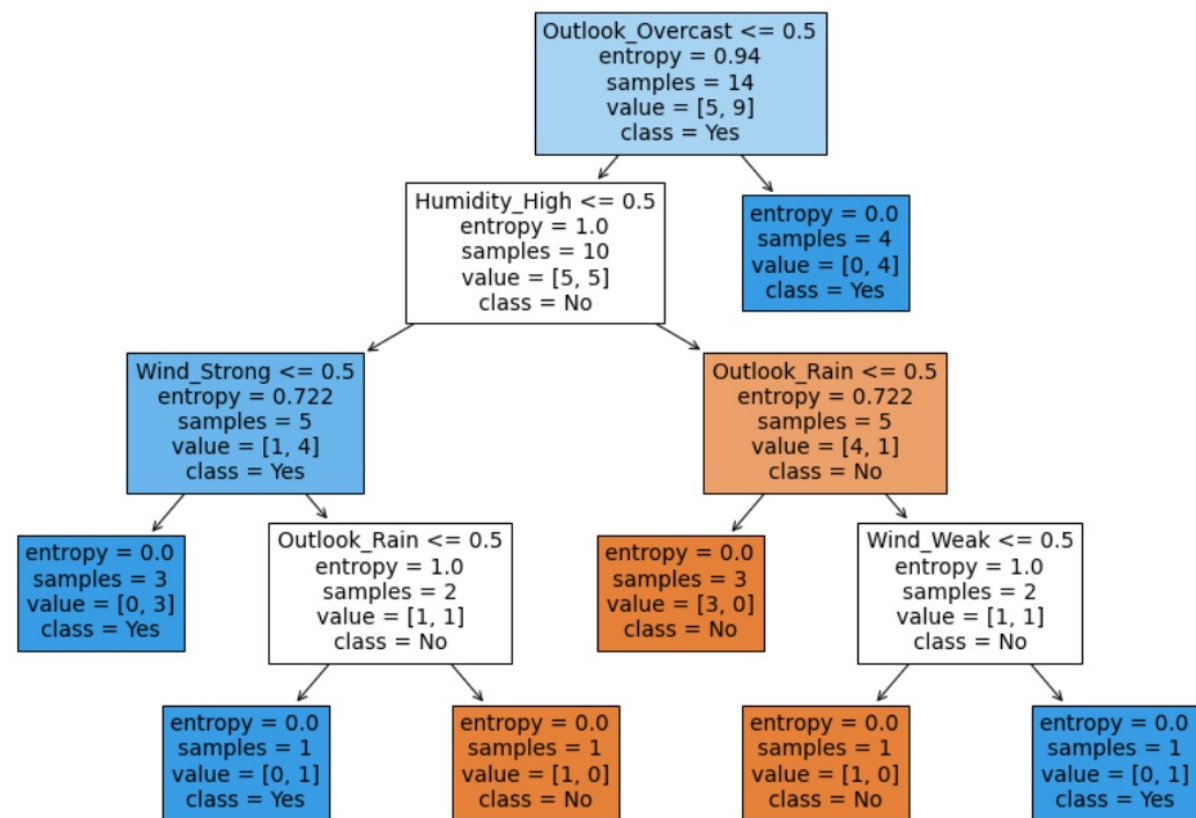
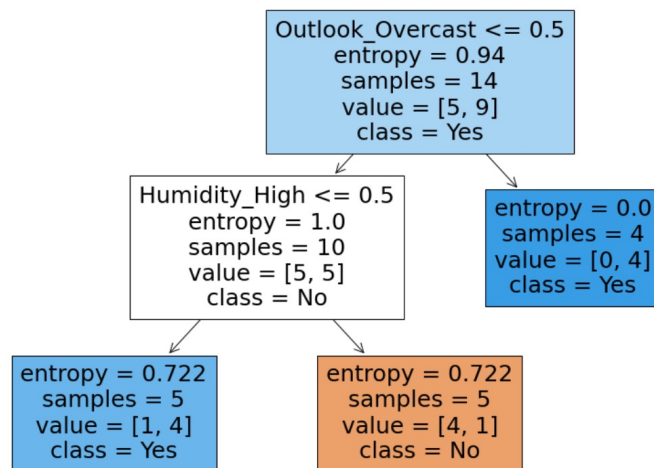
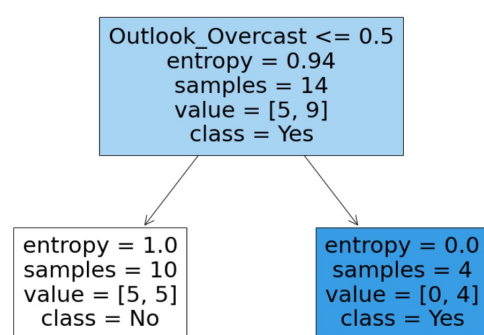


MINIMUM LEAF SAMPLES: IMPLEMENTATION



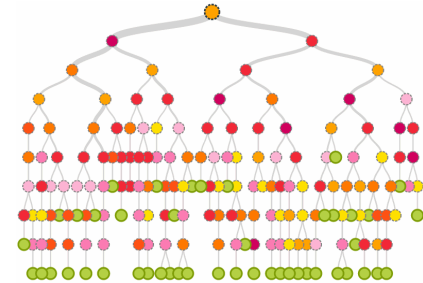
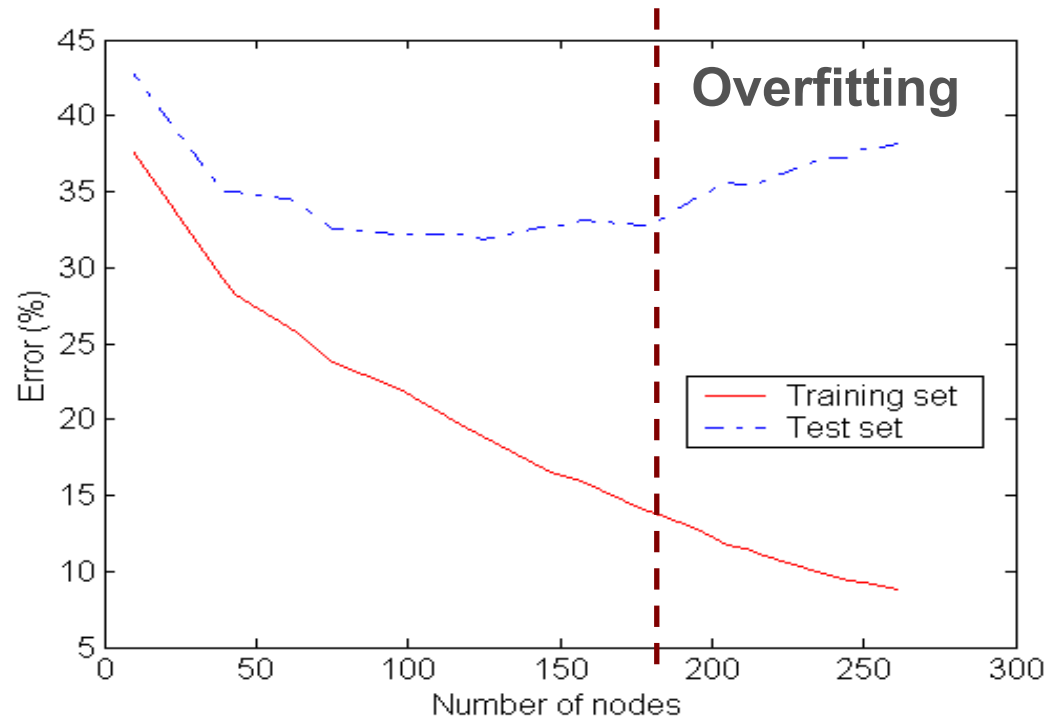
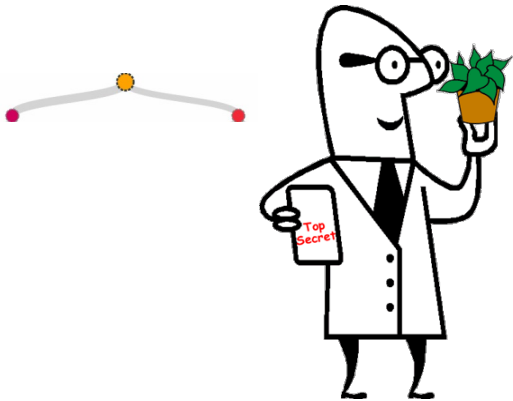
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

WHAT MAKES A GOOD TREE?



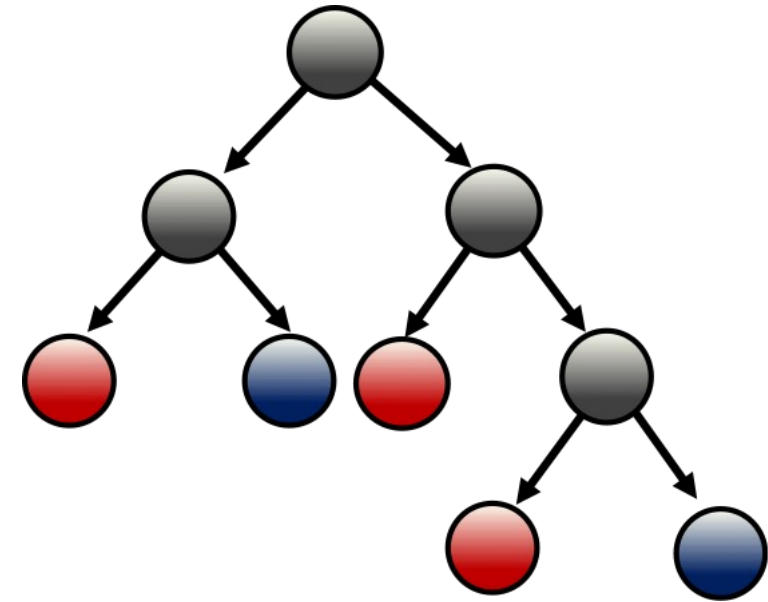
WHAT MAKES A GOOD TREE?

- Overfitting (too big): too many branches that “memorize” training data, may reflect anomalies and noises; poor accuracy for unseen sample
- Underfitting (too small): too simple, both training and test errors are large
- Bias-variance tradeoff (later)



WHAT MAKES A GOOD TREE?

- Big enough to handle important and possibly subtle distinctions in the data (avoid underfitting)
- Not too big for better interpretation and better performance on unseen data (avoid overfitting)
- Early stopping vs. post-pruning



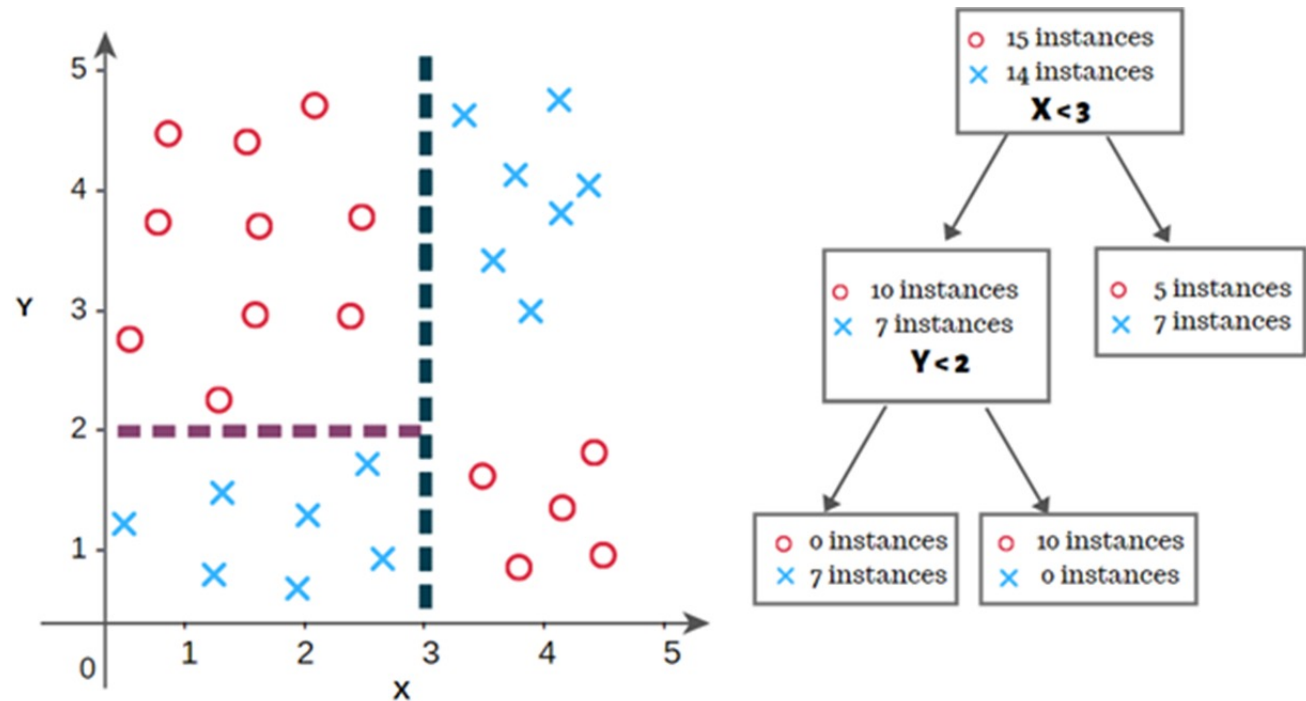
PREDICTED CLASS PROBABILITIES

- Each region R_j contains some subset of training data point
- Predicted probability is proportion of points in the region belong to class k

$$\hat{p}_g(R_j) = \frac{1}{n_j} \sum_{\mathbf{x}_i \in R_j} \mathbb{1}_{\{y_i=g\}}$$

- Predicted class is the most common class occurring amongst these points

$$g_j = \operatorname{argmax} \hat{p}_g(R_j)$$



PRUNING A TREE

- Grow a very large tree T_0 and prune it to get a subtree
- Cost complexity pruning (weakest link pruning) – find a subtree T in T_0 that minimizes both classification error and tree size:

$$C_\alpha(T) = \sum_{j=1}^{|T|} [1 - \hat{p}_{g_j}(R_j)] + \alpha|T|$$

- Prune the weakest leaf one at a time
- Use **cross validation** to choose alpha (**later**)
- Grow a tree using Info gain or Gini, prune a tree using classification error



PRUNING A TREE: EXAMPLE

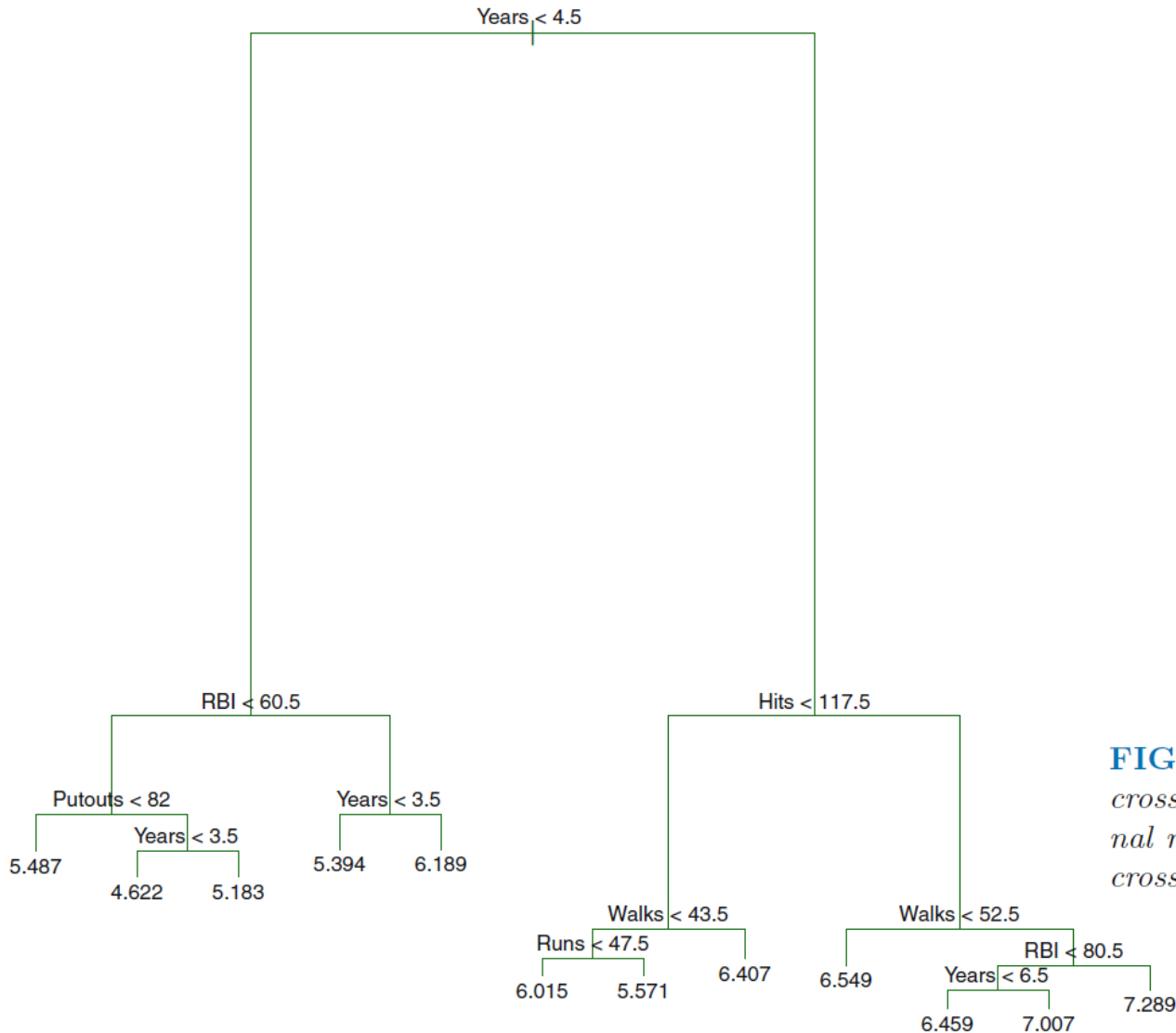


FIGURE 8.4. Regression tree analysis for the **Hitters** data. The unpruned tree that results from top-down greedy splitting on the training data is shown.

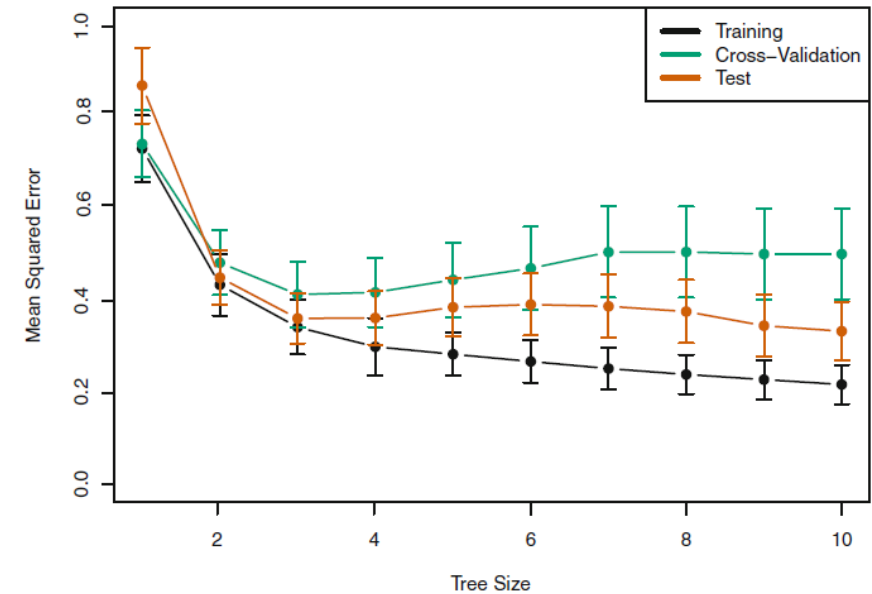


FIGURE 8.5. Regression tree analysis for the **Hitters** data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.

HOW TO GROW (BUILD) TREES?

- How to learn the best tree?
 - How to choose the node (splits)?
 - When to stop the tree (how big to grow)?
- Regression vs. classification
- Handling missing attribute values

