Decision Trees

aiphabet

Your AI Journey Starts Here

Tree Classifiers

- Popular classifiers
- Simple and interpretable
- No assumption about data distribution
- **History**
  - CART (classification and regression trees): Friedman 1977
  - ID3 and C4.5 family: Quilan, 1979 to 1983
  - Refinements in mid-1990s (e.g., pruning, numerical features)
- **Applications**
  - Medical research (e.g., disease classification)
  - Computational biology (e.g., interaction between genes)

Tree Classifiers

- The terminology **tree** is graphic
- However, a decision tree is grown from the root downward; the idea is to send the examples down the tree, using the concept of information entropy
- **General steps to build a tree**

1. Start with the root node that has all the examples
2. Greedy selection of the next best feature to build the branches; the splitting criterion is *node purity*
3. Class majority will be assigned to the leaves

Toy Example

Training data:

Mood	HW completed	Weather	Friend available	Movies?
Happy	Yes	Sunny	TRUE	yes
Bored	No	Rainy	FALSE	yes
Bored	Yes	Rainy	TRUE	yes
Excited	Yes	Sunny	TRUE	yes
Excited	No	Sunny	TRUE	no
Happy	On it	Sunny	TRUE	no
Happy	Yes	Rainy	FALSE	yes
Excited	No	Sunny	FALSE	no
Happy	On it	Rainy	FALSE	yes
Bored	On it	Sunny	TRUE	no
Bored	On it	Rainy	FALSE	yes
Excited	Yes	Rainy	FALSE	no
Bored	Yes	Rainy	TRUE	yes
Happy	No	Sunny	FALSE	no

Toy Example

Training data:

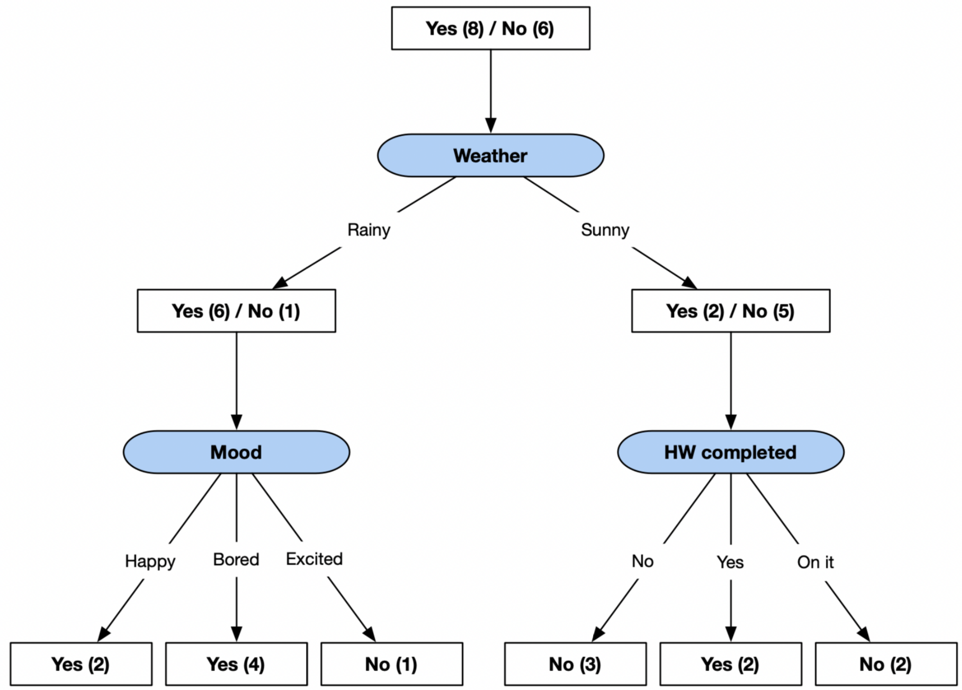
Mood	HW completed	Weather	Friend available	Movies?
Happy	Yes	Sunny	TRUE	yes
Bored	No	Rainy	FALSE	yes
Bored	Yes	Rainy	TRUE	yes
Excited	Yes	Sunny	TRUE	yes
Excited	No	Sunny	TRUE	no
Happy	On it	Sunny	TRUE	no
Happy	Yes	Rainy	FALSE	yes
Excited	No	Sunny	FALSE	no
Happy	On it	Rainy	FALSE	yes
Bored	On it	Sunny	TRUE	no
Bored	On it	Rainy	FALSE	yes
Excited	Yes	Rainy	FALSE	no
Bored	Yes	Rainy	TRUE	yes
Happy	No	Sunny	FALSE	no

Test data:

Mood	HW completed	Weather	Friend available	Movies?
Bored	On it	Rainy	TRUE	?

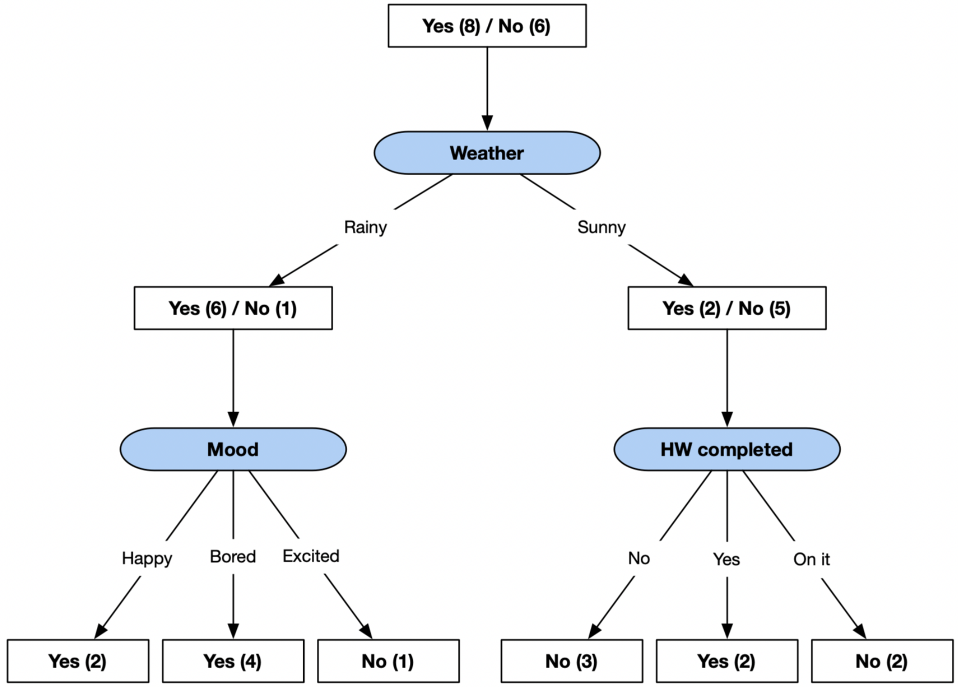
Toy Example

Using the **Training data**, we build the following **Decision Tree:**



Toy Example

Using the **Training data**, we build the following **Decision Tree:**



What is the prediction for the **Test data?**

Mood	HW completed	Weather	Friend available	Movies?
Bored	On it	Rainy	TRUE	?

Splitting Criteria

How did we build the tree?

1. The central choice is selecting the next attribute to split on.

2. We want some criteria that measure the homogeneity or impurity of examples in the nodes.

   (a) Quantify the mix of classes at each node

   (b) Maximum if equal number of examples from each class

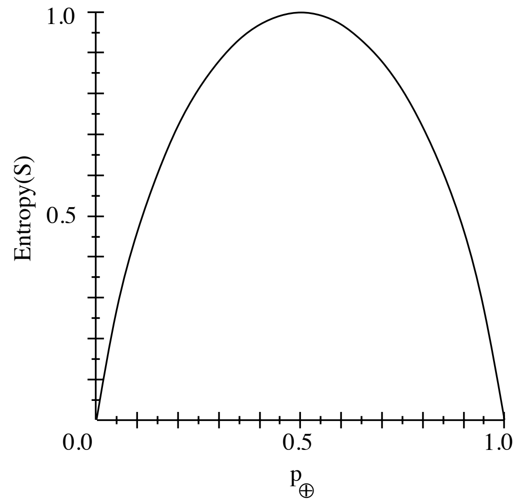
   (c) Minimum if the node is pure

Splitting Criteria

A perfect measure commonly used in *information theory*:

$$\boxed{Entropy(S) = -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}}$$

$p_{\oplus}$ is the proportion of positive examples.
$p_{\ominus}$ is the proportion of negative examples.



Now each node has some entropy that measures its homogeneity.

Splitting Criteria

- How do you decide on which attribute it is best to split based on entropy?

Splitting Criteria

- How do you decide on which attribute it is best to split based on entropy?
- We use **information gain** that measures the expected reduction in entropy caused by partitioning the examples according to the attributes:

  For a node, the gain of splitting at attribute A, is the entropy at that note minus the sum of entropies at the children:

$$Gain(node, A) = Entropy(node) - \sum_{child} \frac{|child|}{|node|} Entropy(child)$$

Splitting Criteria

- How do you decide on which attribute it is best to split based on entropy?
- We use **information gain** that measures the expected reduction in entropy caused by partitioning the examples according to the attributes:

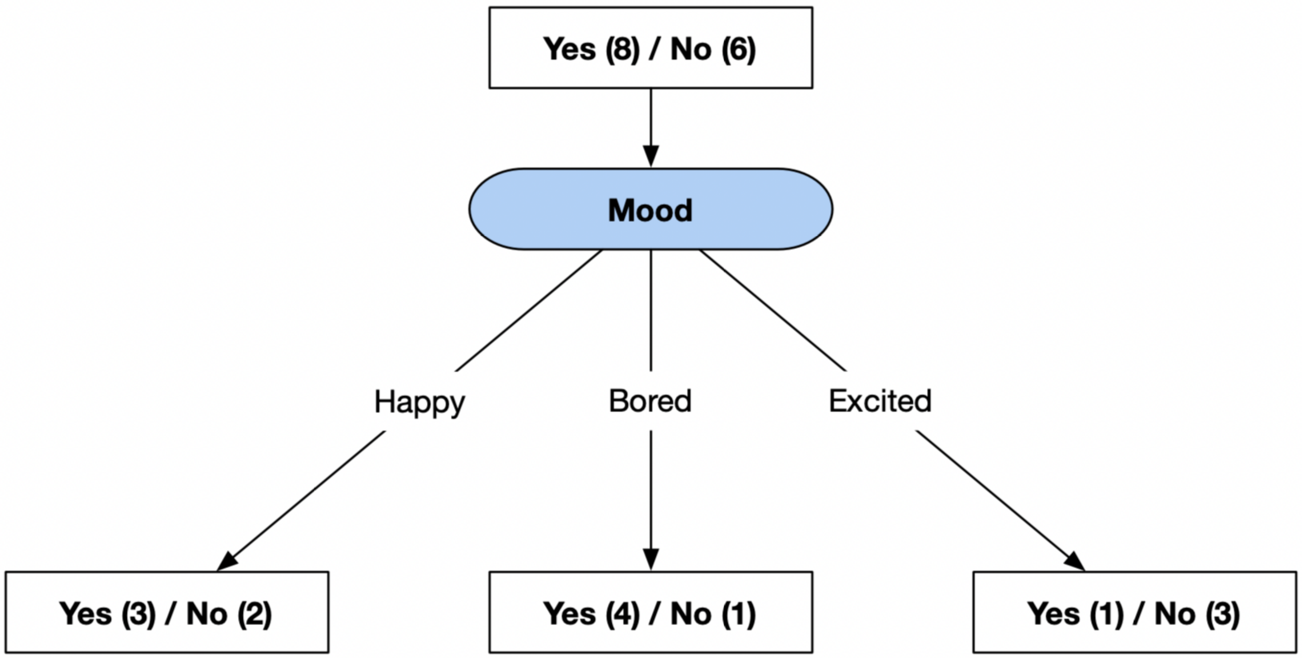
  For a node, the gain of splitting at attribute A, is the entropy at that note minus the sum of entropies at the children:

$$Gain(node, A) = Entropy(node) - \sum_{child} \frac{|child|}{|node|} Entropy(child)$$

Why did we multiply by $\frac{|child|}{|node|}$?

Toy Example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985

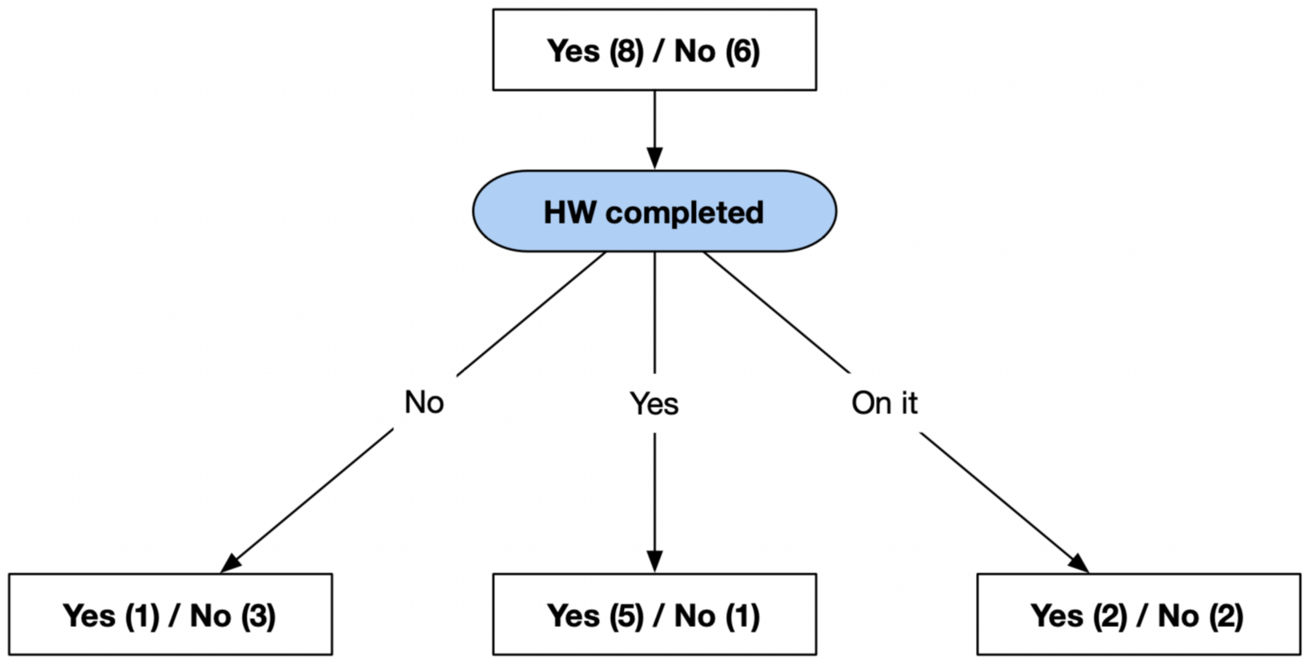


Entropy(3, 2) = 0.971 Entropy(4, 1) = 0.722 Entropy(1, 3) = 0.811

Gain(S, Mood) = 0.985 – (5/14) x 0.971 – (5/14) x 0.722 – (4/14) x 0.811 = 0.149

Toy Example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985

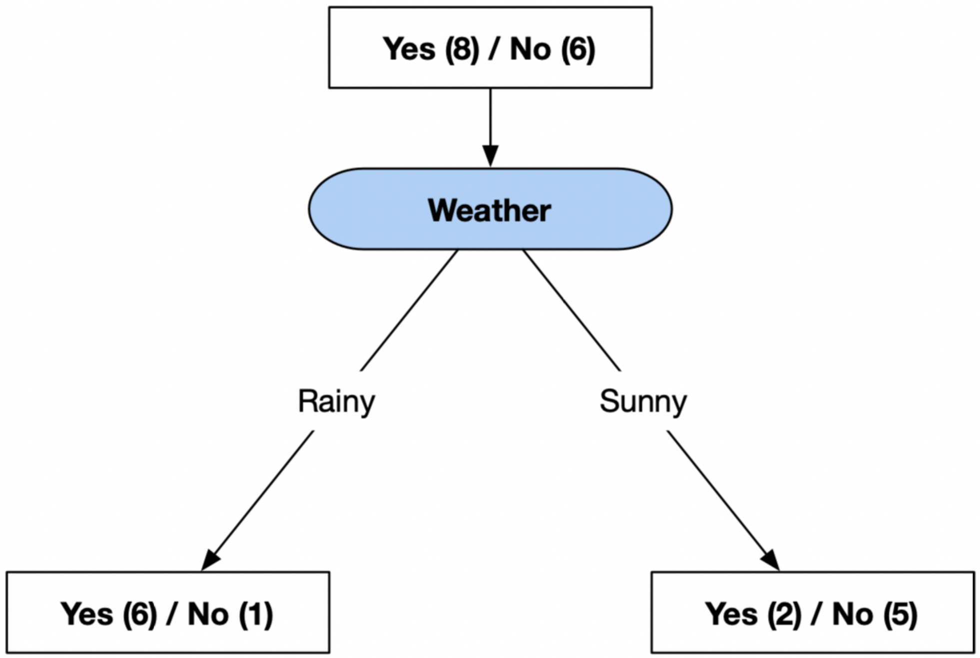


Entropy(1, 3) = 0.811 | Entropy(5, 1) = 0.650 | Entropy(2, 2) = 1.000

Gain(S, HW completed) = 0.985 – (4/14) x 0.811 – (6/14) x 0.650 – (4/14) x 1.000 = 0.189

Toy Example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



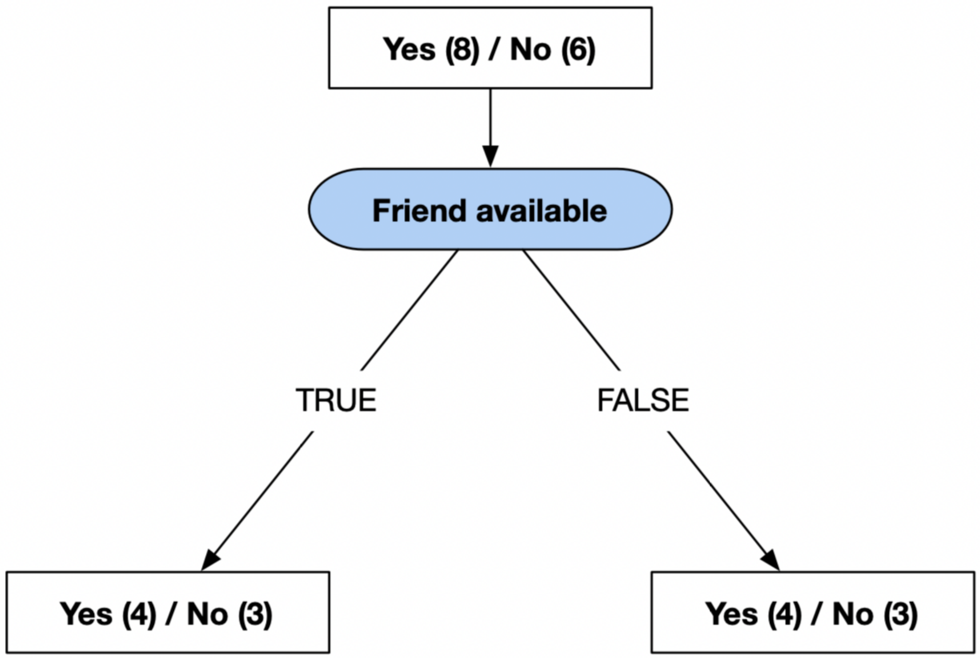
Entropy(6, 1) = 0.592

Entropy(2, 5) = 0.863

Gain(S, Weather) = 0.985 – (7/14) x 0.592 – (7/14) x 0.863 = 0.258

Toy Example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(4, 3) = 0.985

Entropy(4, 3) = 0.985

Gain(S, Friend available) = 0.985 – (7/14) x 0.985 – (7/14) x 0.985 = 0.000

Toy Example

Feature	Information Gain
Mood	0.149
HW completed	0.189
Weather	**0.258**
Friend available	0.000

At the first split starting from the root, we choose the attribute that has the max gain. **Here Weather is the winner!**

Then, we restart the same process at each of the children nodes.

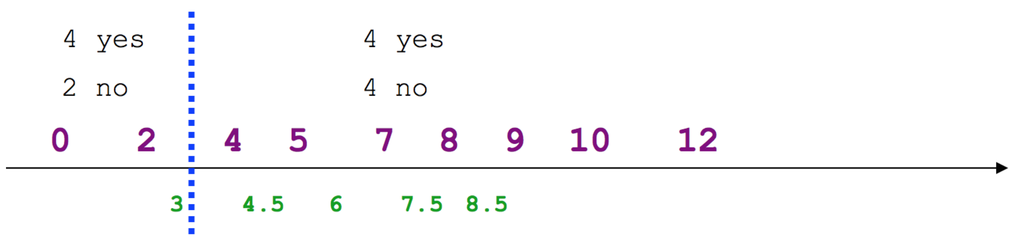
Numerical Features

Mood	#HW to complete	Weather	Friend available	Movies?
Happy	5	Sunny	TRUE	yes
Bored	1	Rainy	FALSE	yes
Bored	6	Rainy	TRUE	yes
Excited	4	Sunny	TRUE	yes
Excited	3	Sunny	TRUE	no
Happy	8	Sunny	TRUE	no
Happy	5	Rainy	FALSE	yes
Excited	3	Sunny	FALSE	no
Happy	7	Rainy	FALSE	yes
Bored	7	Sunny	TRUE	no
Bored	9	Rainy	FALSE	yes
Excited	4	Rainy	FALSE	no
Bored	4	Rainy	TRUE	yes
Happy	2	Sunny	FALSE	no

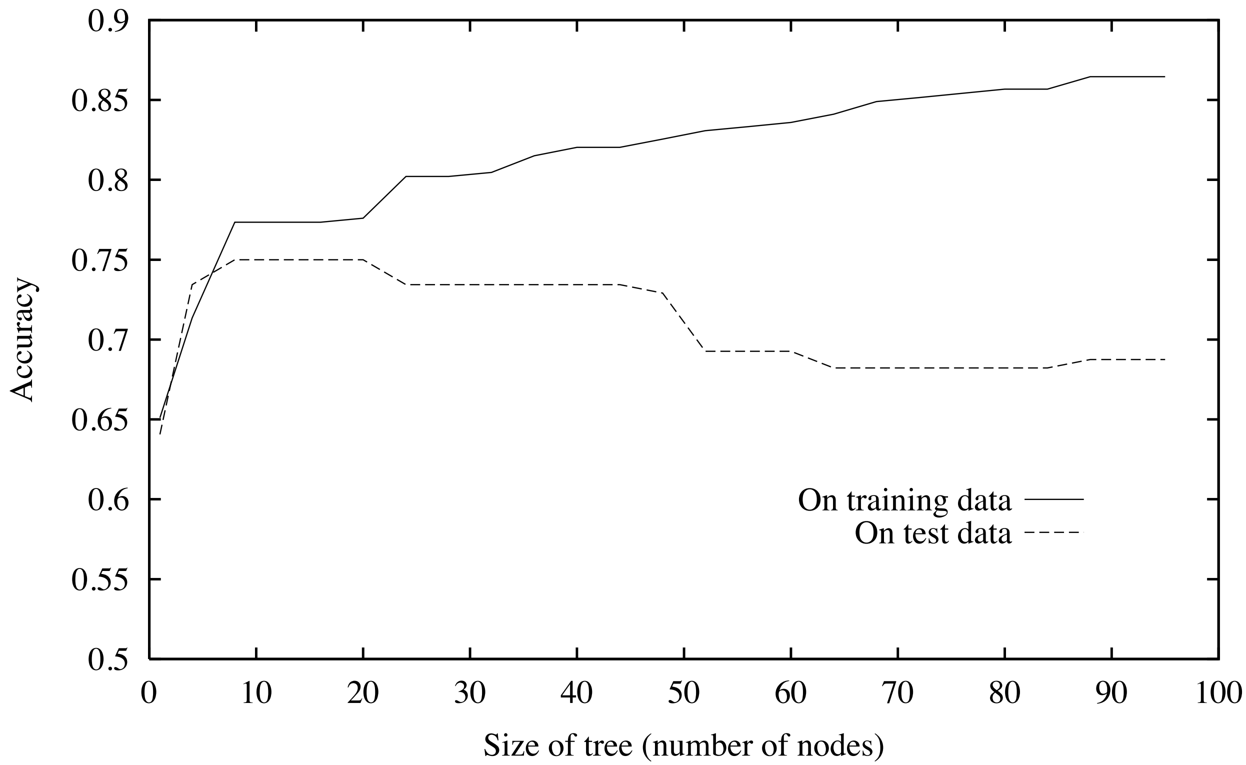
Mood	HW completed	Weather	Friend available	Movies?
Bored	2	Rainy	TRUE	?

Numerical Features

- Numerical features are explicitly discretized and used as categorical features (e.g., #HW to complete: $[0, 2)$, $[2, 4)$...)
- Numerical features are individually discretized on the fly.
  1. Order the k values of the numerical feature to discretize
  2. Determine the cutoff (threshold point that leads to the best bipartition of the examples at the node to split)
  3. The point is to pick among the $k - 1$ points in the middle of the intervals
  4. Test each discretization against the gain and keep the best cutoff point



Overfitting



Accuracy = 1-Error

Pruning Strategies

To get suitable tree sizes and avoid overfitting:

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training examples (difficult to know when to stop)
- Grow a complex tree and then prune it back (best strategy found)
    1. Use a validation set/cross-validation to evaluate the utility of post-pruning (remove a subtree if the performance of the new tree is no worse than the original tree)

Practical Considerations

1. Consider performing dimensionality reduction beforehand to keep the most discriminative features.

2. Use ensemble methods (e.g., random forest, have a great performance).

3. Balance the dataset before training to prevent the tree from creating a tree biased toward the classes that are dominant.

   - Undersampling: reduce the majority class
   - Oversampling: increase the minority class