Extending decision trees

- **Numeric (real-valued) attributes**
- Missing attribute values
- Discrete attributes with many values
- Attributes with costs
- Multivariate class variable
- Noise and overfitting

Numeric (real-valued) attributes

- Many real-world problems contain numeric attributes
- E.g.: Jeeves data with temperature recorded using real-valued attribute

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	29.4	High	Weak	No
2	Sunny	26.6	High	Strong	No
3	Overcast	28.3	High	Weak	Yes
4	Rain	21.1	High	Weak	Yes
5	Rain	20.0	Normal	Weak	Yes
6	Rain	18.3	Normal	Strong	No
7	Overcast	17.7	Normal	Strong	Yes
8	Sunny	22.2	High	Weak	No
9	Sunny	20.6	Normal	Weak	Yes
10	Rain	23.9	Normal	Weak	Yes
11	Sunny	23.9	Normal	Strong	Yes
12	Overcast	22.2	High	Strong	Yes
13	Overcast	27.2	Normal	Weak	Yes
14	Rain	21.7	High	Strong	No

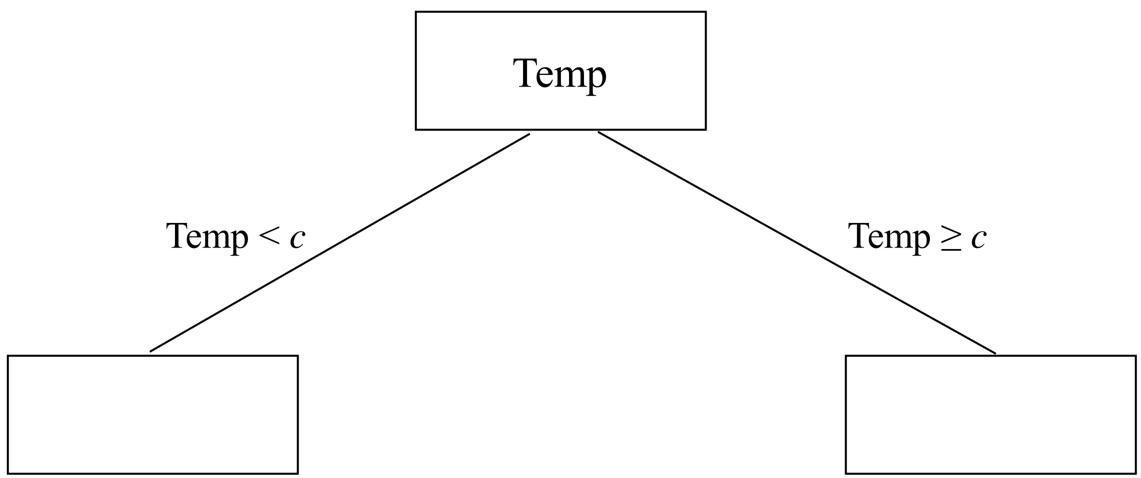
Solution (I)

- Discretize:

$Temp < 20.8$	$\rightarrow$	Cool
$20.8 \leq Temp < 25.0$	$\rightarrow$	Mild
$25.0 \leq Temp$	$\rightarrow$	Hot

Solution (II)

- Branch on real-valued attributes in decision tree
- Idea: dynamically choose a split point c



Solution (II)

- How to choose threshold *c*?
  1. sort the instances according to the real-valued attribute
  2. possible *c*’s are those that are midway between two values that differ in their classification
  3. determine the information gain for each of the possible *c*’s and choose the *c* with the largest gain

Jeeves data with temperature recorded using real-valued attribute

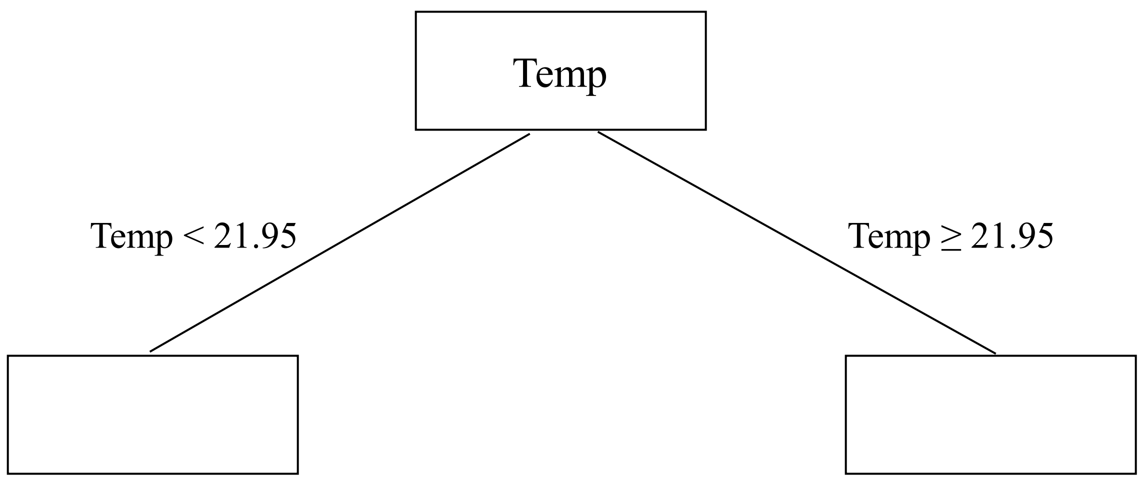
Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	29.4	High	Weak	No
2	Sunny	26.6	High	Strong	No
3	Overcast	28.3	High	Weak	Yes
4	Rain	21.1	High	Weak	Yes
5	Rain	20.0	Normal	Weak	Yes
6	Rain	18.3	Normal	Strong	No
7	Overcast	17.7	Normal	Strong	Yes
8	Sunny	22.2	High	Weak	No
9	Sunny	20.6	Normal	Weak	Yes
10	Rain	23.9	Normal	Weak	Yes
11	Sunny	23.9	Normal	Strong	Yes
12	Overcast	22.2	High	Strong	Yes
13	Overcast	27.2	Normal	Weak	Yes
14	Rain	21.7	High	Strong	No

Jeeves data sorted by temperature

Day	Outlook	Temp	Humidity	Wind	Tennis?
7	Overcast	17.7	Normal	Strong	Yes
6	Rain	18.3	Normal	Strong	No
5	Rain	20.0	Normal	Weak	Yes
9	Sunny	20.6	Normal	Weak	Yes
4	Rain	21.1	High	Weak	Yes
14	Rain	21.7	High	Strong	No
8	Sunny	22.2	High	Weak	No
12	Overcast	22.2	High	Strong	Yes
10	Rain	23.9	Normal	Weak	Yes
11	Sunny	23.9	Normal	Strong	Yes
2	Sunny	26.6	High	Strong	No
13	Overcast	27.2	Normal	Weak	Yes
3	Overcast	28.3	High	Weak	Yes
1	Sunny	29.4	High	Weak	No

Example information gain

- The split $c = \frac{(21.7+22.2)}{2} = 21.95$ gives:



Additional complication…

- On any path from the root to a leaf
  - discrete attribute: tested at most once
  - but real-valued attribute: can be tested *many* times
- Result:
  - *large* trees
  - trees that are difficult to understand

Extending decision trees

- Numeric (real-valued) attributes
- **Missing attribute values**
- Discrete attributes with many values
- Attributes with costs
- Multivariate class variable
- Noise and overfitting

Missing attribute values

- Real-world data will often have missing attribute values
  - E.g.: values not recorded or too expensive to obtain
- Two cases:
  1. when constructing decision tree
  2. when using decision tree

Solution: when constructing decision tree

1. Use other instances to estimate missing attribute (use majority), *or*
2. Divide example into fractional examples weighted according to frequency of value of attributes

Solution: when constructing decision tree

- E.g.: Suppose Outlook value for Day 1 missing from training data

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	???	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Using majority

- E.g.: Suppose Outlook value for Day 1 missing from training data

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	**Rain**	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Using fractional examples

- E.g.: Suppose Outlook value for Day 1 missing from training data

Day	Outlook	Temp	Humidity	Wind	Tennis?	Weight
1a	**Sunny**	Hot	High	Weak	No	4/13
1b	**Overcast**	Hot	High	Weak	No	5/13
1c	**Rain**	Hot	High	Weak	No	4/13
2	Sunny	Hot	High	Strong	No	1
3	Overcast	Hot	High	Weak	Yes	1
4	Rain	Mild	High	Weak	Yes	1
5	Rain	Cool	Normal	Weak	Yes	1
6	Rain	Cool	Normal	Strong	No	1
7	Overcast	Cool	Normal	Strong	Yes	1
8	Sunny	Mild	High	Weak	No	1
9	Sunny	Cool	Normal	Weak	Yes	1
10	Rain	Mild	Normal	Weak	Yes	1
11	Sunny	Mild	Normal	Strong	Yes	1
12	Overcast	Mild	High	Strong	Yes	1
13	Overcast	Hot	Normal	Weak	Yes	1
14	Rain	Mild	High	Strong	No	1

Solution: when using decision tree

- When using decision tree:
  - pretend example has all possible values of attribute
  - follow all possible branches
  - weight answer from a branch by the probability of that value (as estimated from training data)
  - return most probable classification

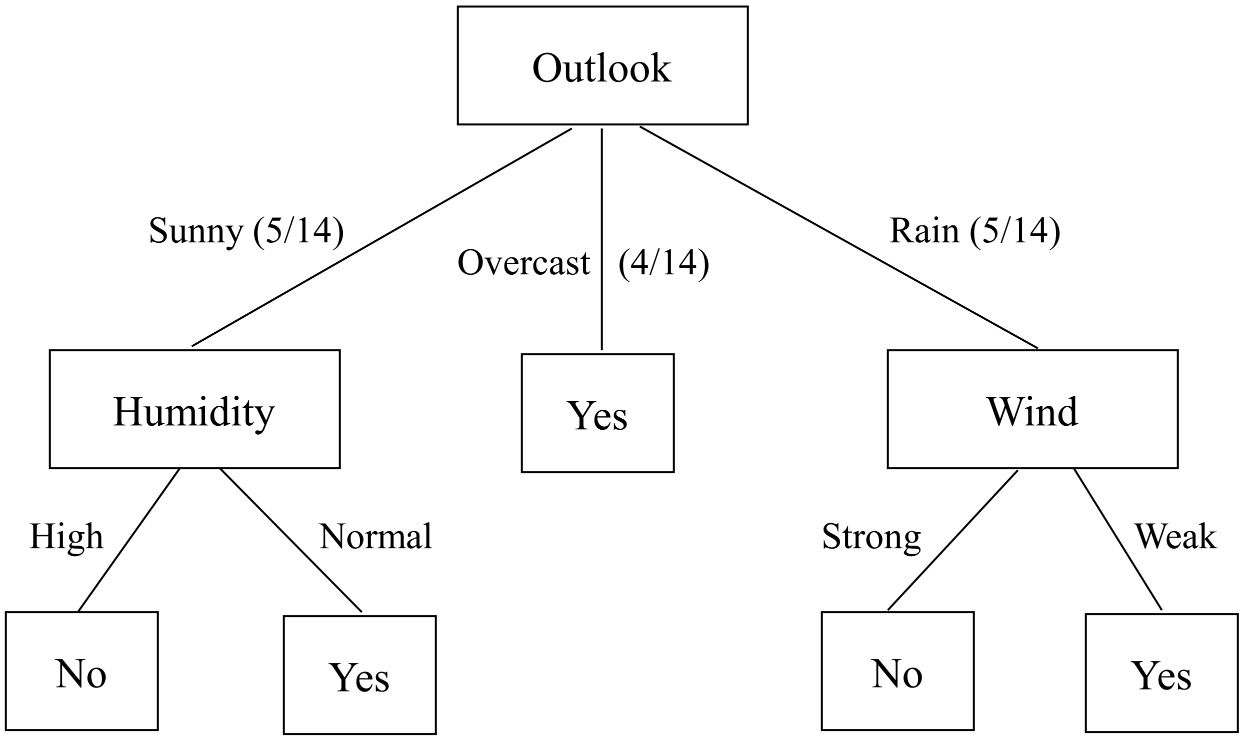
Solution: when using decision tree

- E.g.: Suppose Outlook value for Day 1 missing from test data

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	???	Mild	High	Strong	No
2	Rain	Hot	Normal	Strong	No
3	Rain	Cool	High	Strong	No
4	Overcast	Hot	High	Strong	Yes
5	Overcast	Cool	Normal	Weak	Yes
6	Rain	Hot	High	Weak	Yes
7	Overcast	Mild	Normal	Weak	Yes
8	Overcast	Cool	High	Weak	Yes
9	Rain	Cool	High	Weak	Yes
10	Rain	Mild	Normal	Strong	No
11	Overcast	Mild	High	Weak	Yes
12	Sunny	Mild	Normal	Weak	Yes
13	Sunny	Cool	High	Strong	No
14	Sunny	Cool	High	Weak	No

Solution: when using decision tree

- E.g.: Outlook = **???**, Temp = Mild, Humidity = High, Wind = Strong



Extending decision trees

- Numeric (real-valued) attributes
- Missing attribute values
- **Discrete attributes with many values**
- Attributes with costs
- Multivariate class variable
- Noise and overfitting

Discrete attributes with many values

- Recall: choose the attribute to split on that gives maximum information gain
- Problem: If an attribute has many values, gain will select it

Discrete attributes with many values

- E.g.: Imagine using Day in the training data as an attribute

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Solution

- Pick attribute that maximizes GainRatio
- Suppose that attribute A splits a set of examples S into k different subsets: $S_1, \ldots, S_k$

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{I\left(\frac{|S_1|}{|S|}, \ldots, \frac{|Sk|}{|S|}\right)}$$

Extending decision trees

- Numeric (real-valued) attributes
- Missing attribute values
- Discrete attributes with many values
- **Attributes with costs**
- Multivariate class variable
- Noise and overfitting

Attributes with costs

- In some learning tasks, attributes may have costs
  - E.g.: Medical setting

Temperature	← **less costly, non-invasive**
Pulse	← **less costly, non-invasive**
Biopsy	← **costly, invasive**
Blood test	← **costly, invasive**

- Want: high accuracy *and* low cost
- One solution: Pick attribute which maximizes

$$\text{GainCost}(A) = \frac{(\text{Gain}(A))^2}{Cost(A)}$$

Extending decision trees

- Numeric (real-valued) attributes
- Missing attribute values
- Discrete attributes with many values
- Attributes with costs
- **Multivariate class variable**
- Noise and overfitting

Multivariate class variable

- So far: class variable is binary (Tennis = Yes, Tennis = No)
- Suppose *class* in $\{c_1, \ldots, c_L\}$
- Changes to ID3:

ID3(F, S)

1. if S contains only positive examples, return "Yes"

2. if S contains only negative examples, return "No"

3. else

choose best feature $f \in F$

for each value v of f do

add arc to tree with label v

add subtree ID3(F - $\{f\}$, $\{ s \in S \mid f(s) = v \}$)

Extending decision trees

- Numeric (real-valued) attributes
- Missing attribute values
- Discrete attributes with many values
- Attributes with costs
- Multivariate class variable
- **Noise and overfitting**

Noise and avoiding overfitting

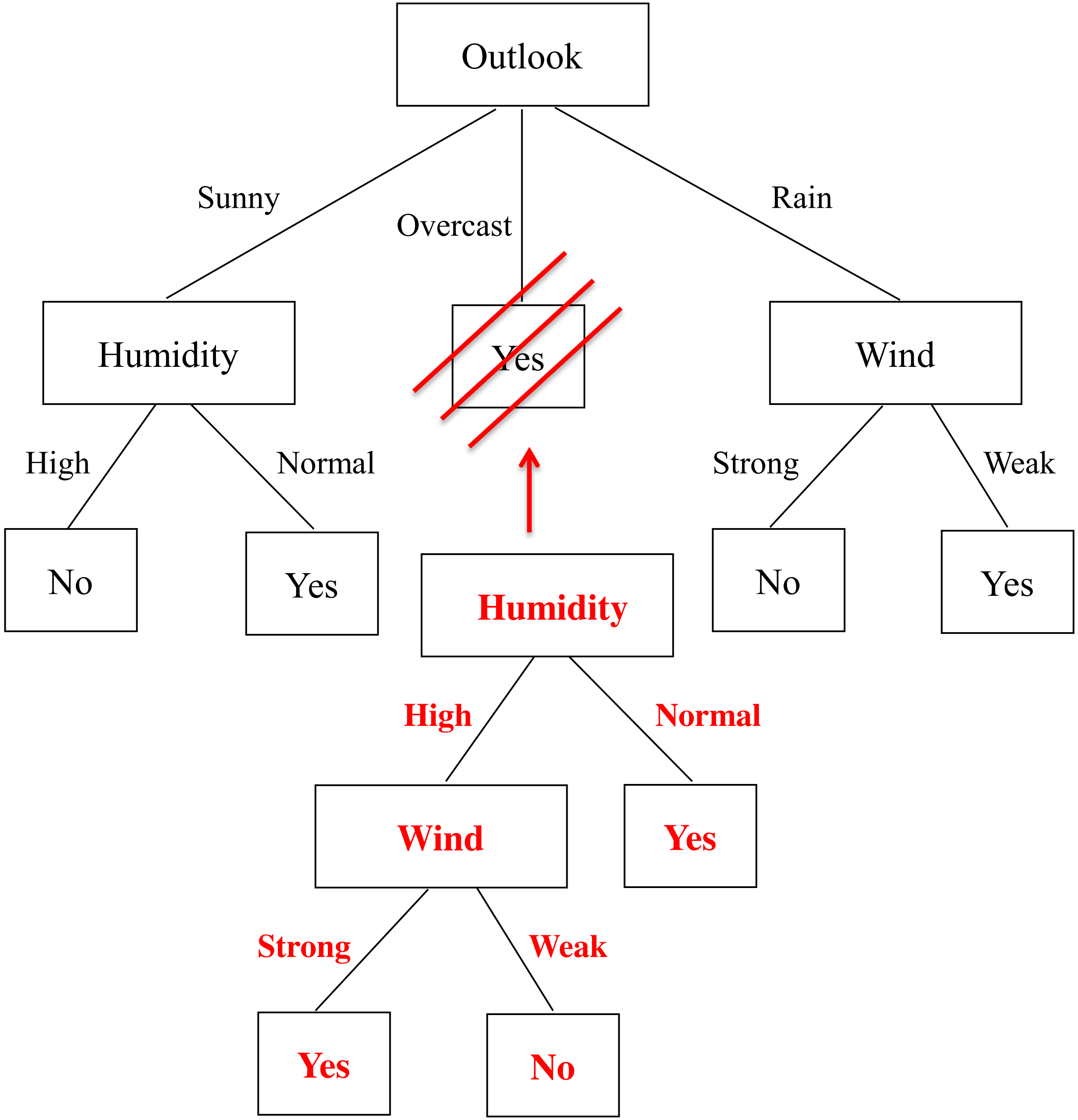
- Attributes may be based on measurements or subjective judgements
- E.g., Suppose Outlook for Day 1 incorrectly recorded as Overcast

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	**Overcast**	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Noise and avoiding overfitting

- Training examples may be misclassified
- E.g., Suppose class of Day 3 is misclassified as No

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
Humidity
No
Yes
High
Normal
Wind
Yes
Strong
Weak
Yes
No

Noise and avoiding overfitting

- Problem:

  ID3 algorithm grows each branch just deeply enough to perfectly classify the training examples

- Solutions:
  1. stop growing tree early (Chi-square statistical test)
  2. post-prune the tree (using a validation set)