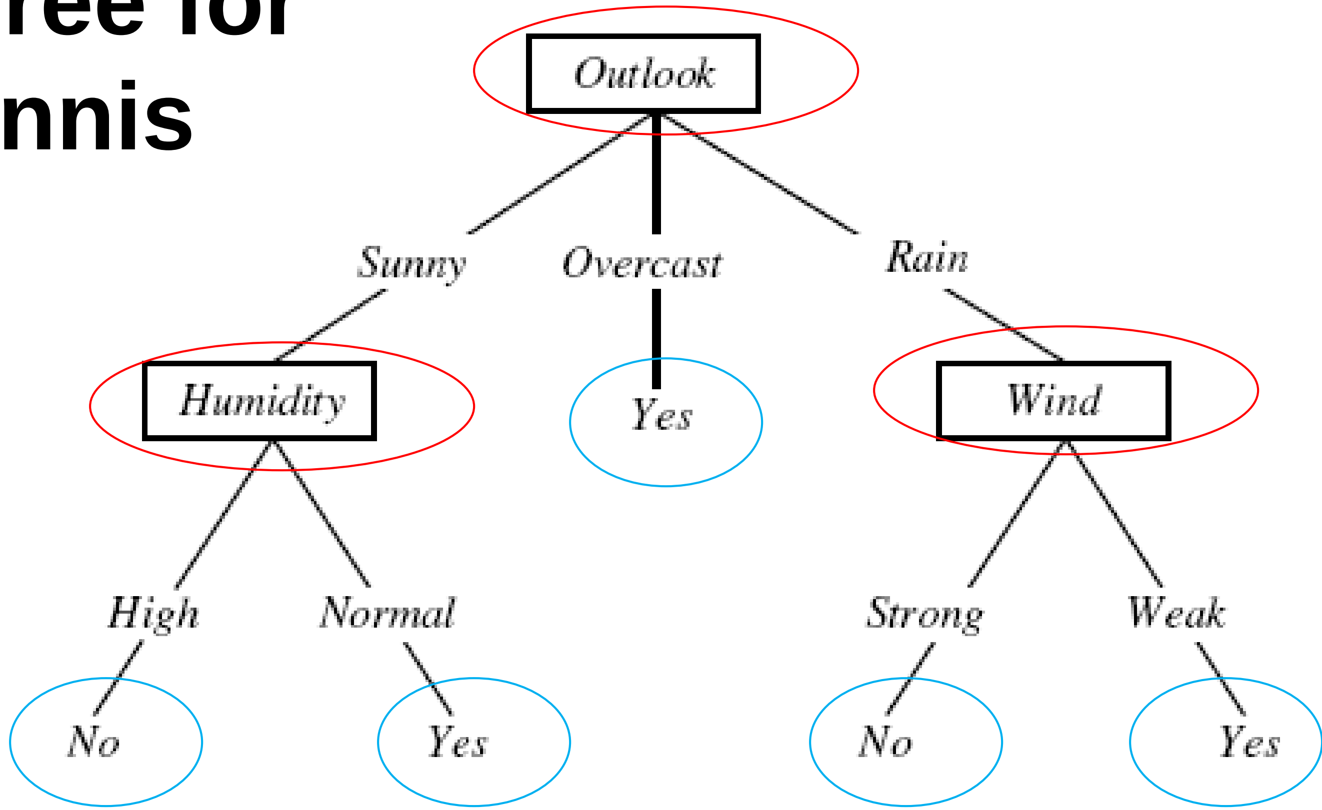
Decision Tree

Key concepts:

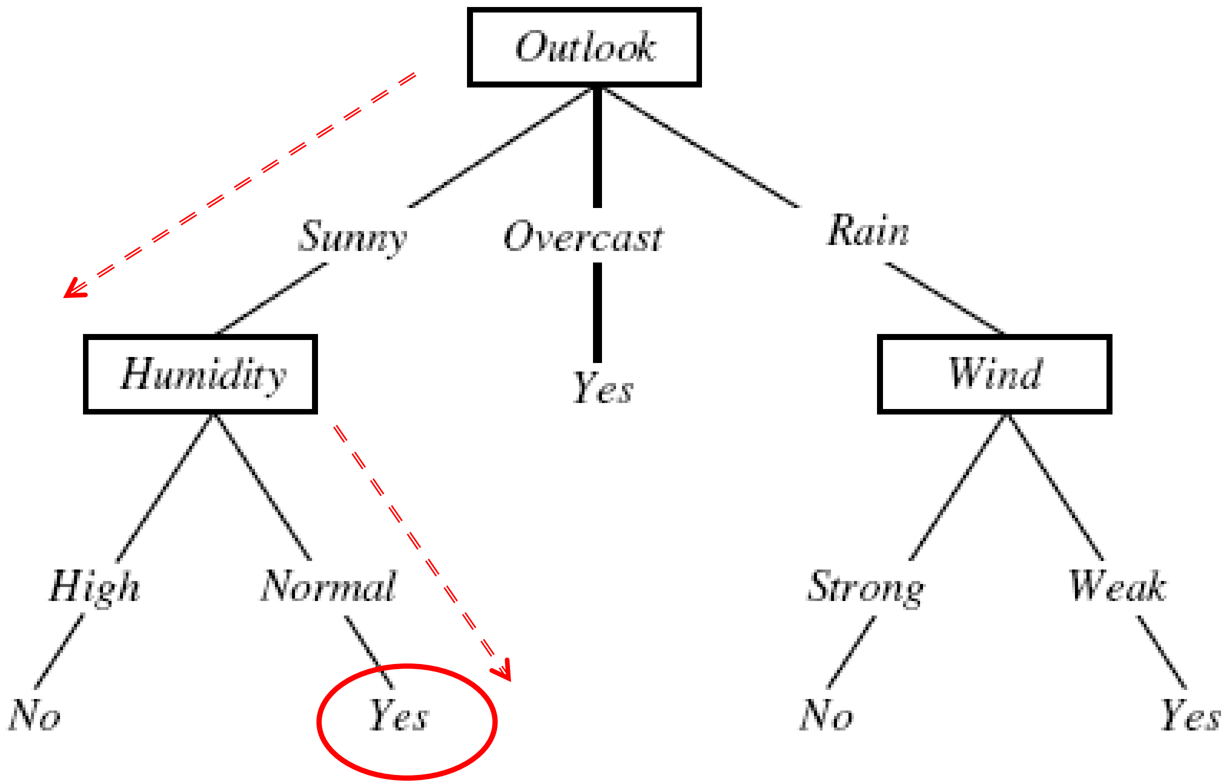
Decision tree learns axes parallel decision boundaries
Top-down greedy learning of decision trees
Entropy, conditional entropy
Mutual information, information gain
Building DT with multi-nomial and continuous features
Preventing Overfitting
Regression trees

Decision Tree for Playing Tennis



- Each internal node test on an attribute x_i
- Each branch from a node takes a particular value of x_i
- Each leaf node predicts a class label

(outlook=sunny, wind=strong, humidity=normal, ?)



DT for prediction C-section risks

Learned from medical records of 1000 women

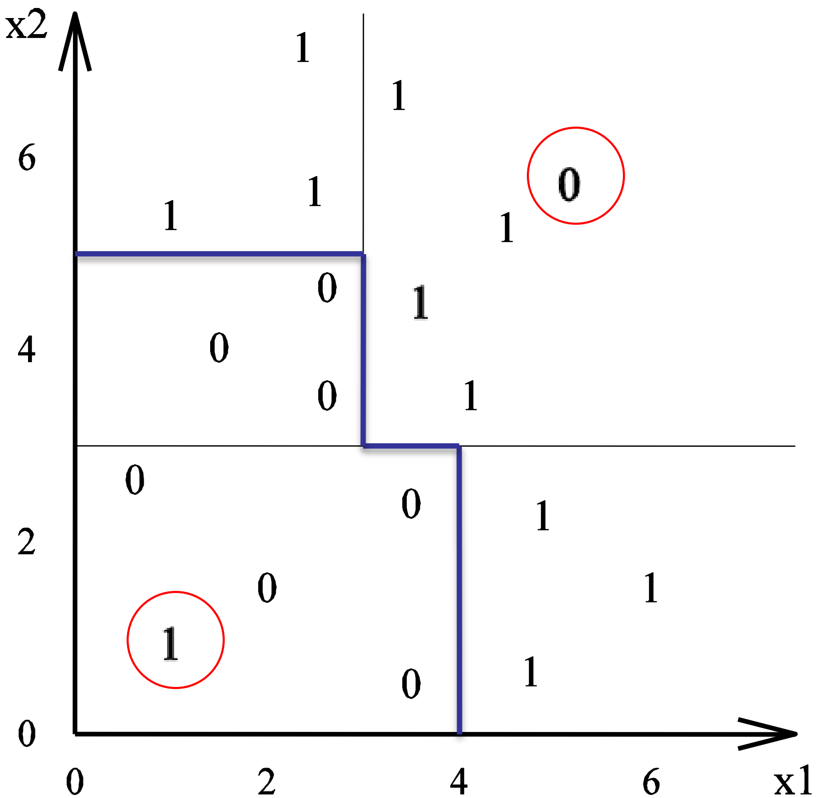
Negative examples are C-sections

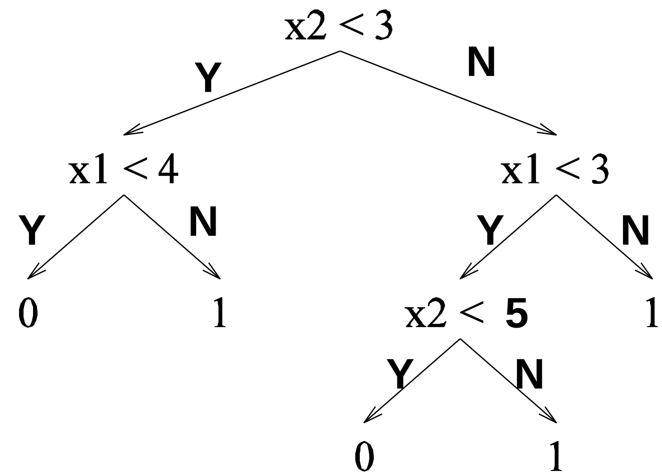
```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Characteristics of Decision Trees

- Decision trees have many appealing properties
  - Similar to human decision process, easy to understand
  - Deal with both **discrete and continuous** features without the need to normalize or similar preprocessing
  - Highly flexible ***hypothesis space*** *(the space of all possible solutions)***,** decision trees can represent increasingly complex decision boundaries as we increase the depth of the tree

DT can represent arbitrarily complex decision boundaries





If needed, the tree can keep on growing until all examples are correctly classified! Although it may not be the best idea

How to learn decision trees?

- Possible goal: find a decision tree *h* that achieves minimum error on training data
  – Trivially achievable – if use a large enough tree
- Another possibility: find the smallest decision tree that achieves the minimum training error
  – NP-hard

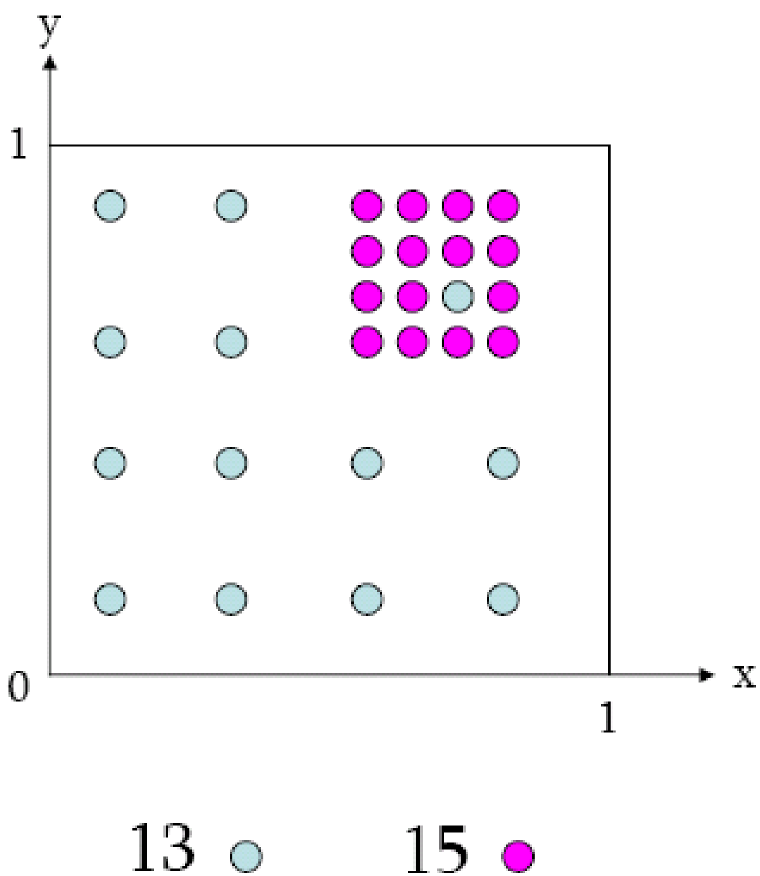
Greedy Learning For DT

We will study a top-down, greedy search approach. Instead of trying to optimize the whole tree together, we try to find one test at a time.

Basic idea: (assuming discrete features, relax later)

1. Choose the best attribute to test on at the root of the tree.
2. Create a descendant node for each possible outcome of the test
3. Training examples in training set S are sent to the appropriate descendent node
4. until some conditions are met Recursively apply the algorithm at each descendant node to select the best attribute to test using its associated training examples
   - If all examples in a node belong to the same class, turn it into a leaf node, label with the majority class

Building DT: start with an intuitive example



Training data contains

13 15

If you have to make a prediction without testing on any features, what would it be?

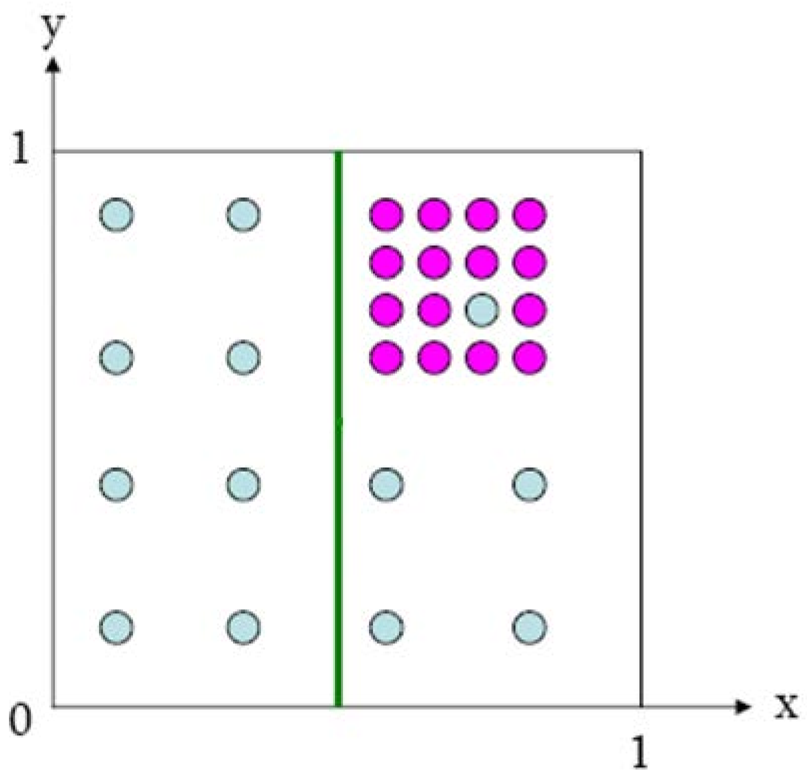
because it is the majority

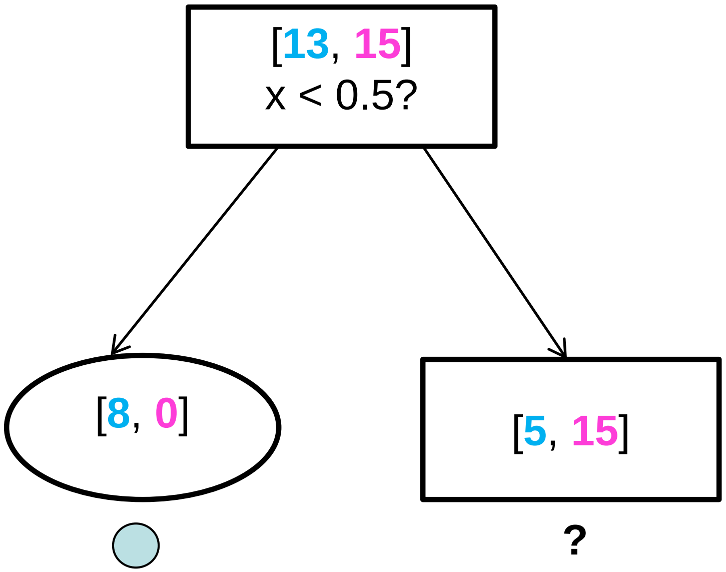
But this prediction is very uncertain

With 13/28 probability to be wrong

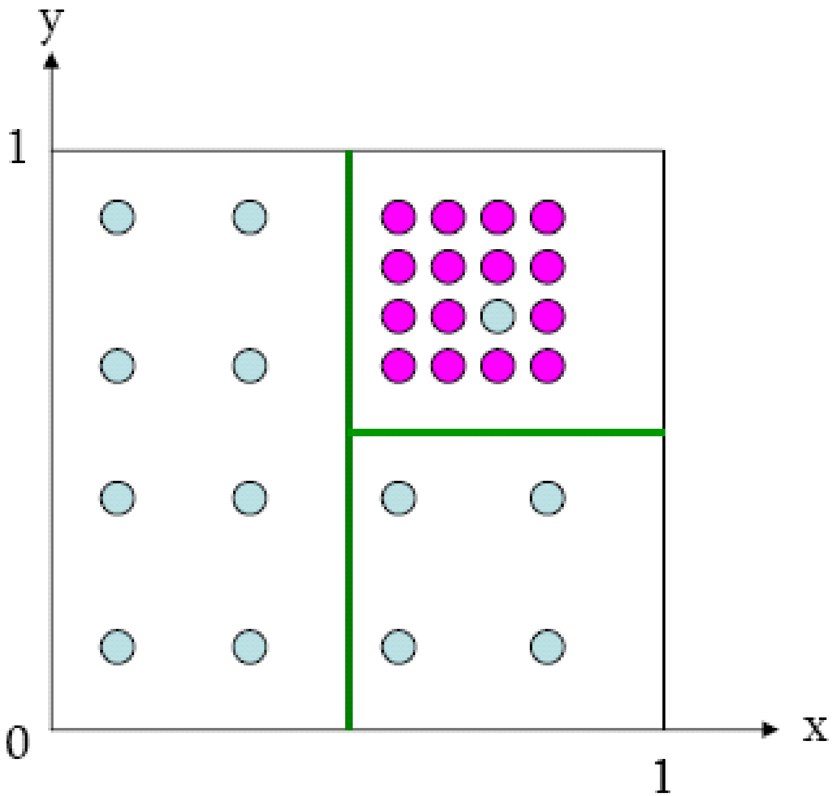
If you can ask one question to reduce uncertainty, what would that be?

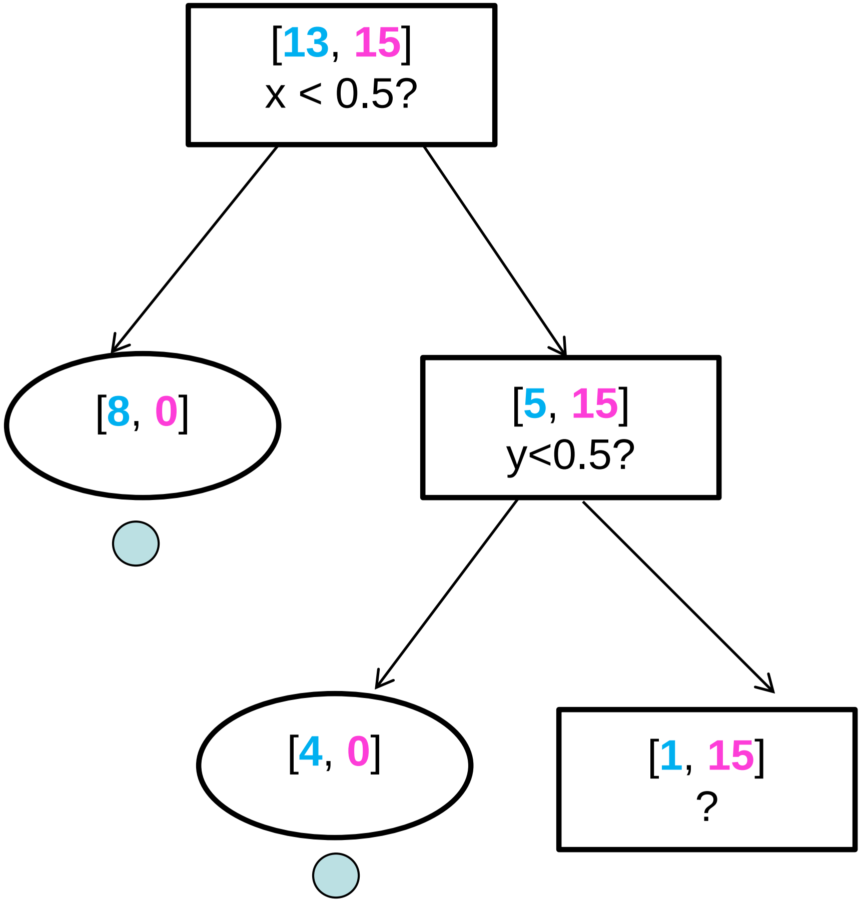
One possible question: is x <0.5?





Continue

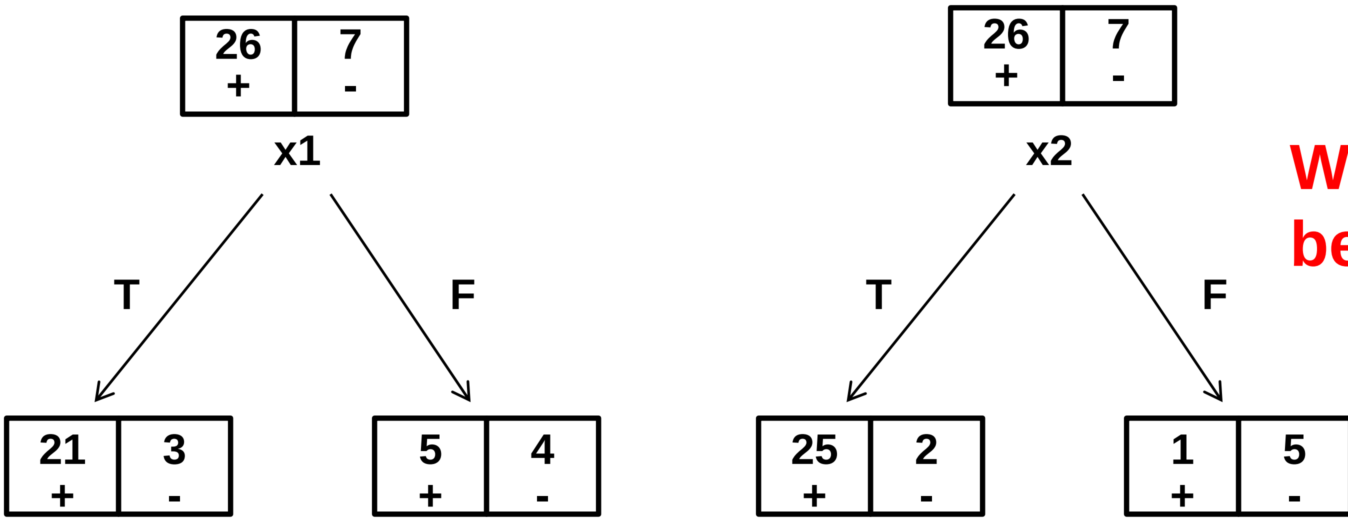




This could keep on going, until all examples are correctly classified.

How to choose the best test

Consider a (Hypothetical) data set:
25 + and 14 – examples
Consider two binary features x1 and x2 which splits the data in the following ways:



Which one is better?

A general recipe: choose the test that maximally reduce **<u>uncertainty about class label</u>**

Uncertainity가 measure의 기준이 된당

Measuring Uncertainty: Entropy

- In information theory, entropy measures the uncertainty of a random variable
- Let y be a random variable, it's entropy is defined as follows.
  - If y is a discrete random variable:

$$H(y) = -\sum_{i=1}^{k} P(y = v_i) \log_2 P(y = v_i)$$

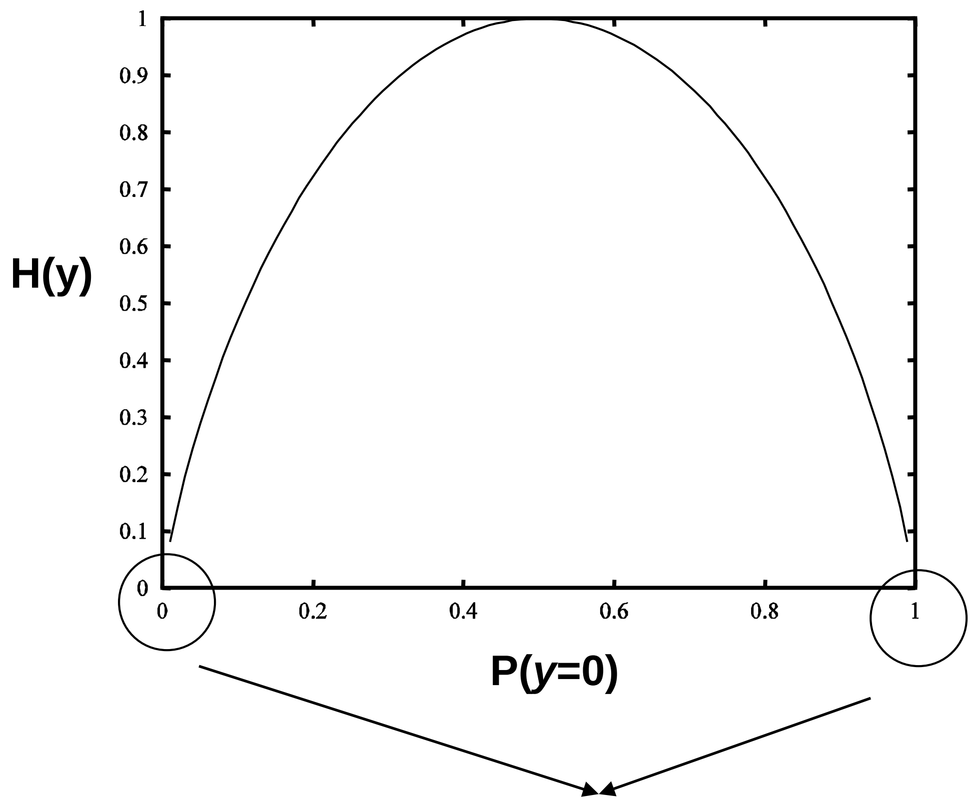
  - If y is a continuous random variable:

$$H(y) = -\int p_y(v) \log_2 p_y(v) \, dv$$

$$H(y) = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{k} p_i \log_2 p_i$$

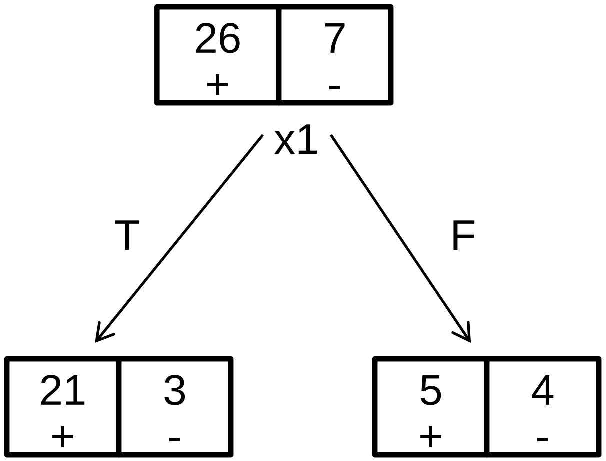
Entropy of a Binary y

- Entropy is a concave function downward



Minimum uncertainty occurs when p_0=0 or 1

Entropy reduction?



At the root:

$$P(y=1) = \frac{26}{33}, \qquad P(y=0) = \frac{7}{33}$$

$$H(y) = -\frac{26}{33}\log_2\frac{26}{33} - \frac{7}{33}\log_2\frac{7}{33} = .7455$$

Left branch:

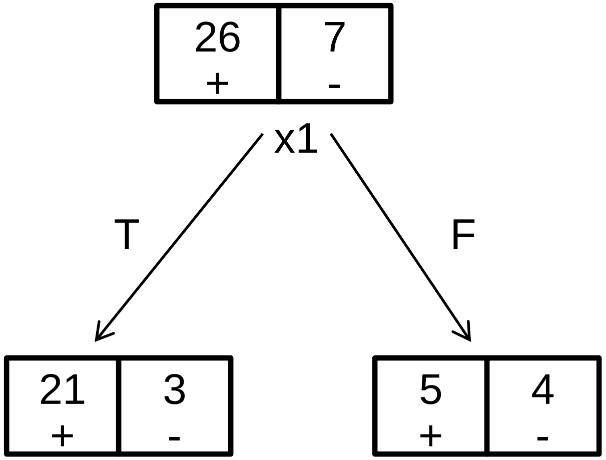
$$P(y=1) = \frac{21}{24};\ P(y=0) = \frac{3}{24};\ H(y) = -\frac{21}{24}\log_2\frac{21}{24} - \frac{3}{24}\log_2\frac{3}{24} = .5436$$

Right branch:

$$P(y=1) = \frac{5}{9};\ P(y=0) = \frac{4}{9};\ H(y) = -\frac{5}{9}\log_2\frac{5}{9} - \frac{4}{9}\log_2\frac{4}{9} = .9911$$

Uncertainty increase or decrease? How to combine the two branches?

Combining the branches



What is the probability of each branch?

$$P(x_1 = T) = \frac{24}{33}$$

$$P(x_1 = F) = \frac{9}{33}$$

- The combined uncertainty is simply the weighted entropy of all branches

$$P(x_1 = T)H(y|x_1 = T) + P(x_1 = F)H(y|x_1 = F)$$

Conditional entropy

- This is called **<u>conditional entropy</u>**

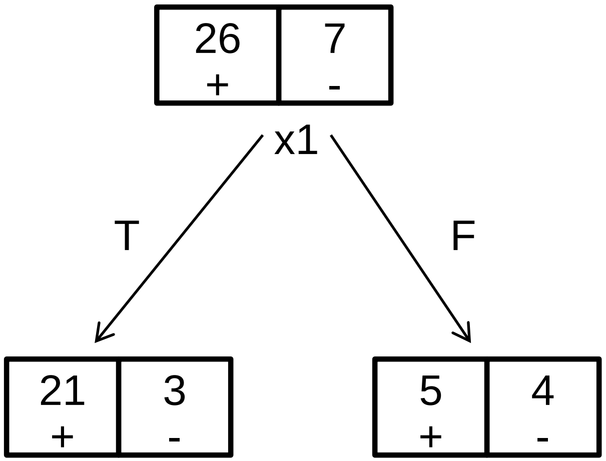
- More generally:

$$H(y|x) = \sum_{u} P(x = u) H(y|x = u)$$

where u denote the possible values for r.v. x

- Conditional entropy $H(y|x)$ measures the remaining <u>uncertainty of y after knowing the value of x</u>

Example: Conditional entropy $H(y|x_1)$



Original entropy:

$$H(y) = .746$$

Left branch:

$$P(x_1 = T) = \frac{24}{33};\ \ H(y|x_1 = T) = .544$$

Right branch:

$$P(x_1 = F) = \frac{9}{33};\ \ H(y|x_1 = F) = .991)$$

Conditional entropy:

$$H(y|x_1) = \frac{24}{33} * .544 + \frac{9}{33} * .991 = .6659$$

Mutual information

- By measuring the uncertainty with entropy, we are select the feature with the largest **mutual information** with the class label y
- Definition: the mutual information between two random variables x and y is defined as:

not common def. but def the most relevant to our class

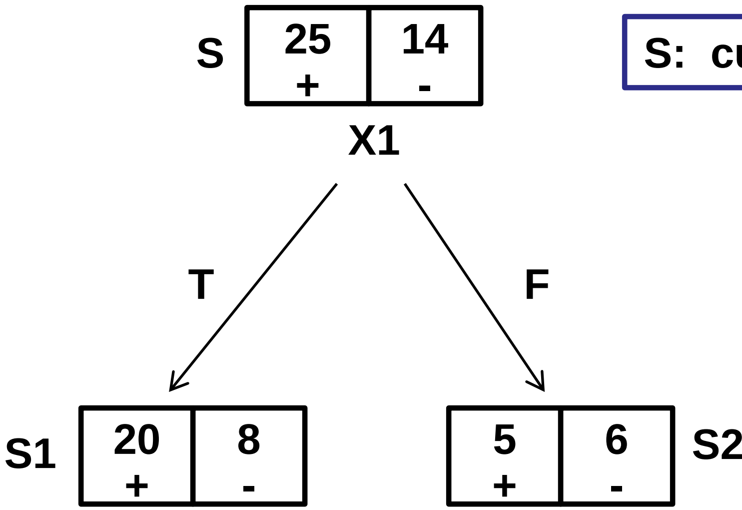
$$I(x, y) = H(y) - H(y|x)$$

how much info. x tells you about y

Mutual information is symmetric: $I(x, y) = I(y, x)$

- This is also referred to as the information gain criterion

More general measure of uncertainty reduction



S: current set of training examples

m branches, one for each possible outcome of the test

$S_1, S_2, \ldots S_m$: m subsets of training examples

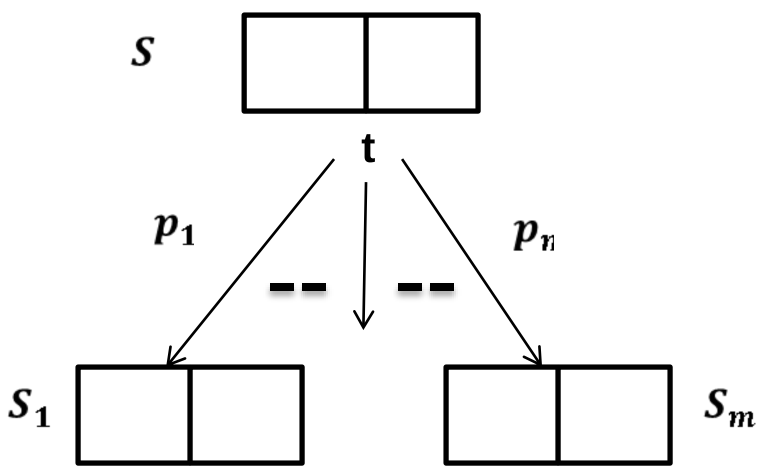
p_i: The portion of examples in S that takes branch i

$$\textbf{Benefit of split} = \underbrace{U(S)}_{\text{Uncertainty of the class label in S}} - \underbrace{\sum_i^m p_i U(S_i)}_{\text{Total Expected Remaining Uncertainty after the test}}$$

Uncertainty of the class label in S

Total Expected Remaining Uncertainty after the test

Choosing the Best Feature: Summary



Benefit of split = $\underbrace{U(S)}_{\text{Original uncertainty}} - \underbrace{\sum_i^m p_i U(S_i)}_{\text{Total Expected Remaining Uncertainty after the test}}$

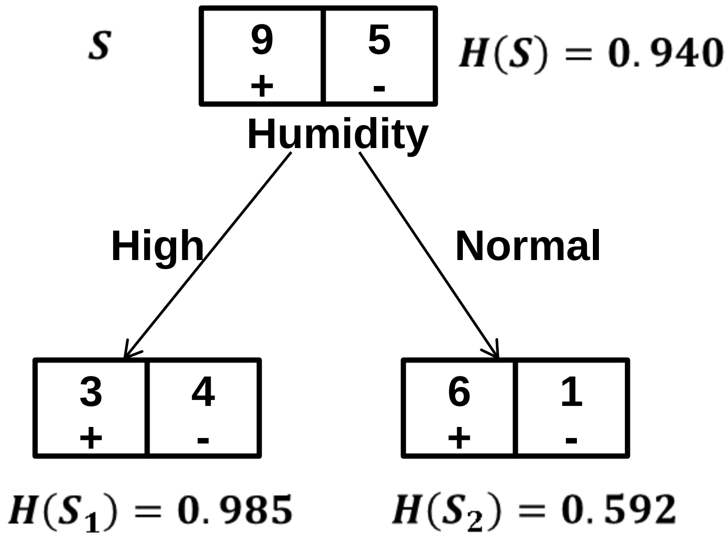
Measures of Uncertainty	
Error	$\min(p_+, p_-)$
Entropy	$-p_+ \log_2 p_+ - p_- \log_2 p_-$
Gini Index	$p_+ p_-$

Day is unique value, so we cannot apply
There’s no generalization for such feature

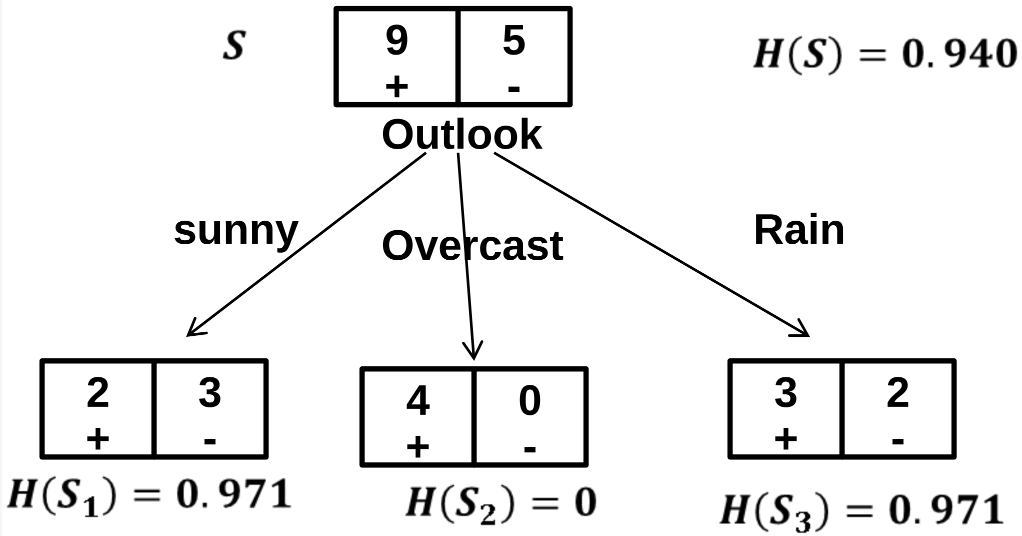
Example

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
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

Selecting the root test using information gain

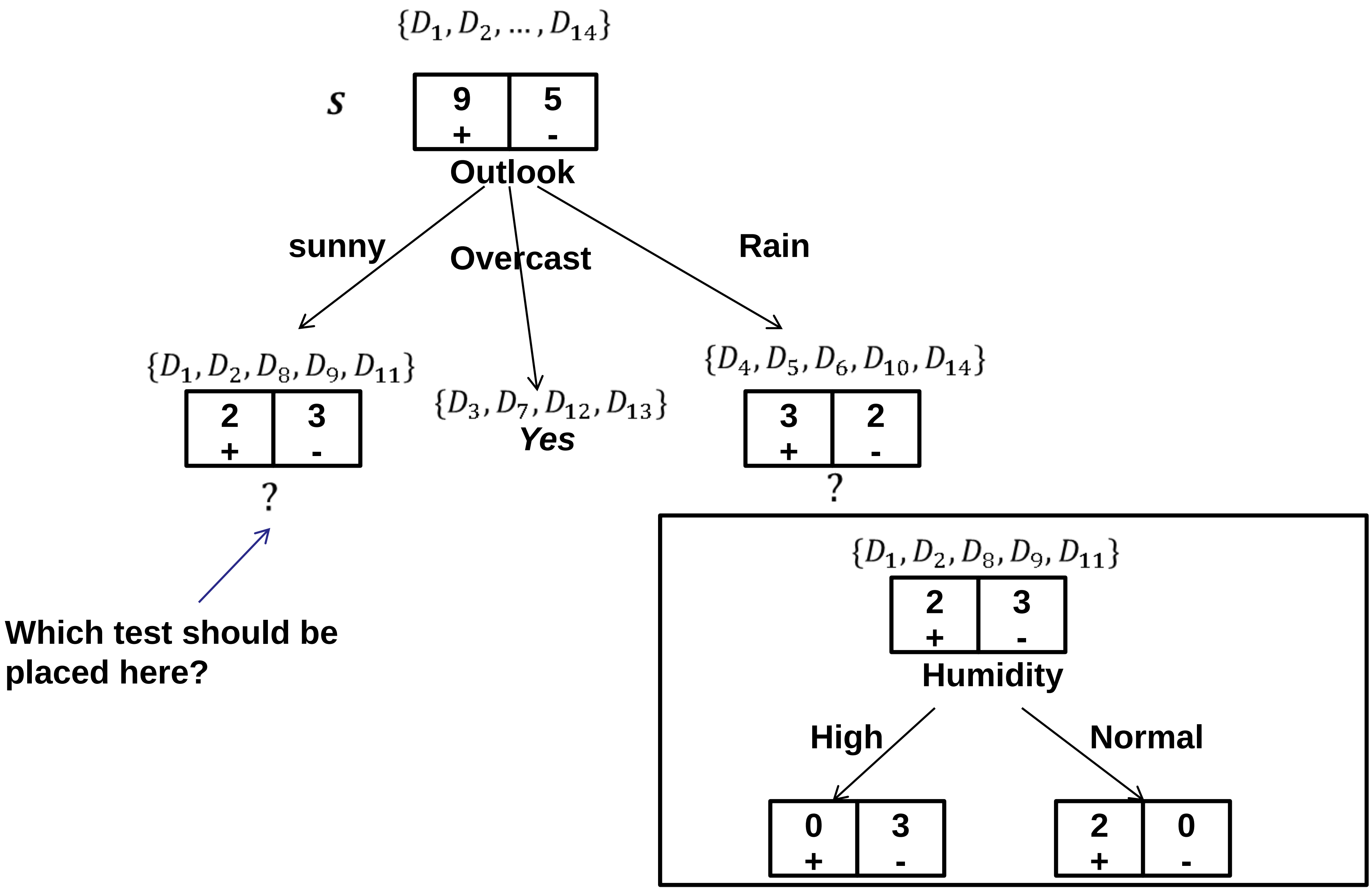


$$Gain(humidity)$$
$$= 0.940 - \frac{1}{2}0.985 - \frac{1}{2}0.592 = 0.151$$



$$Gain(Outlook)$$
$$= 0.940 - \frac{5}{14}0.971 - \frac{5}{14}0.971 = 0.2464$$

Continue building the tree



Issues with Multi-nomial Features

- Multi-nomial features: more than 2 possible values
- Consider two features, one is binary, the other has 100 possible values, which one you expect to have higher information gain?
- Conditional entropy of Y given the 100-valued feature will be low – why?
- This bias will prefer multinomial features to binary features

Method 1: To avoid this, we can rescale the information gain:

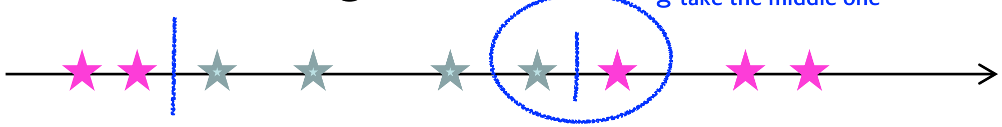
$$\arg\max_j \frac{H(y) - H(y \mid x_j)}{H(x_j)}$$

Method 2: Test for one value versus all of the others – commonly used

Dealing with Continuous Features

- Test against a threshold
- How to compute the best threshold θ_j for x_j?
  - Sort the examples according to x_j.
  - Move the threshold θ from the smallest to the largest value
  - Select θ that gives the best information gain
  - Trick: only need to compute information gain when class label changes

이 두개 사이에서만 움직일 수 밖에 없으니까
걍 take the middle one



요 두개만 consider하면 된다

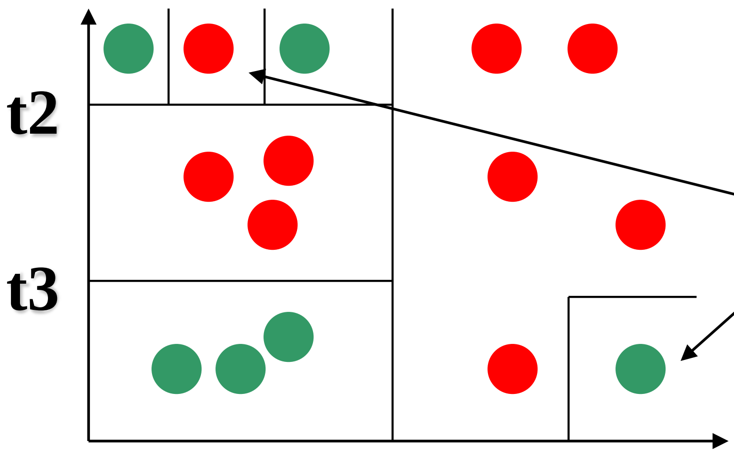
- Note that continuous features can be tested for multiple times on the same path in a DT

Considering both discrete and continuous features

- If a data set contains both types of features, do we need special handling?
- No, we simply consider all possibly splits in every step of the decision tree building process, and choose the one that gives the highest information gain
  - This include all possible (meaningful) thresholds

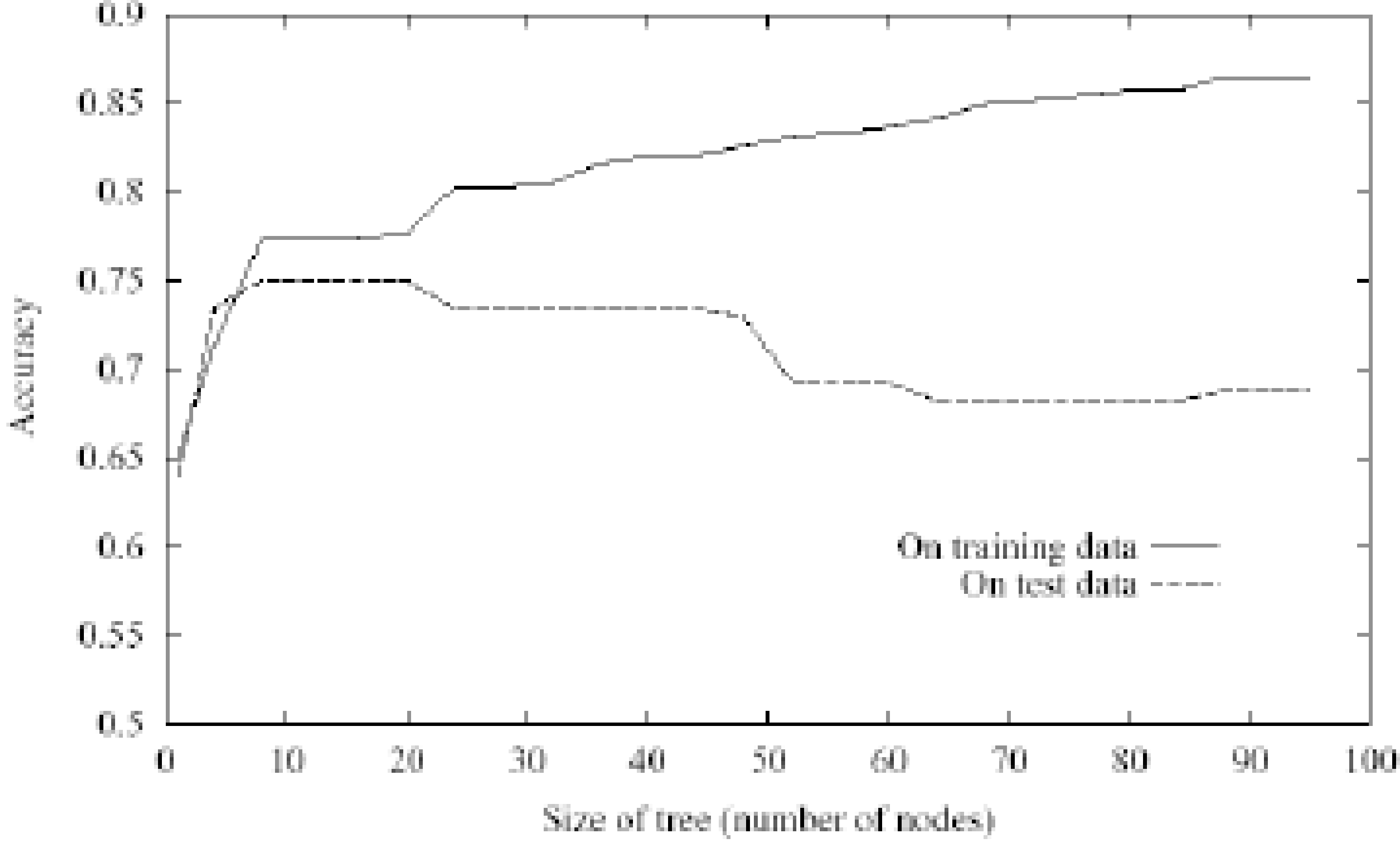
Issue of Over-fitting

- Decision tree has a very flexible hypothesis space
- As the nodes increase, we can represent arbitrarily complex decision boundaries
- This can lead to over-fitting



Possibly just noise, but the tree is grown larger to capture these examples

Over-fitting



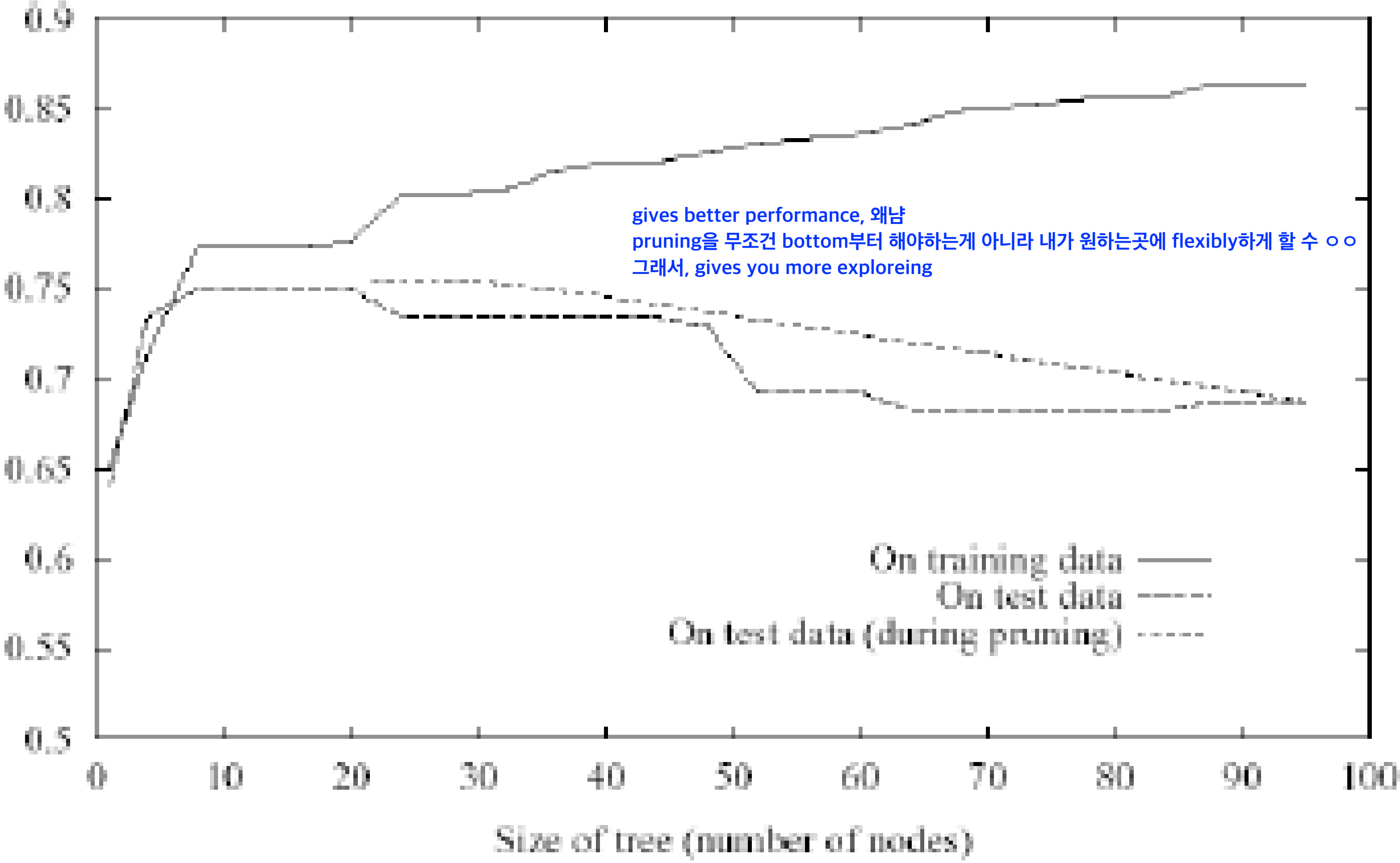
Avoid Overfitting

- Early stop
  - Stop growing the tree when data split does not offer large benefit (e.g., compare information gain to a threshold, or perform statistical testing to decide if the gain is significant)
- Post pruning

  random split 한다음에 이게 나은지, 이게나으면 걍 일캐하고
  아니면 뭐 greedy로하고

  - Separate training data into **training set** and **validating set**
  - Evaluate impact on validation set when pruning each possible node
  - Greedily prune the node that most improves the validation set performance

Effect of Pruning



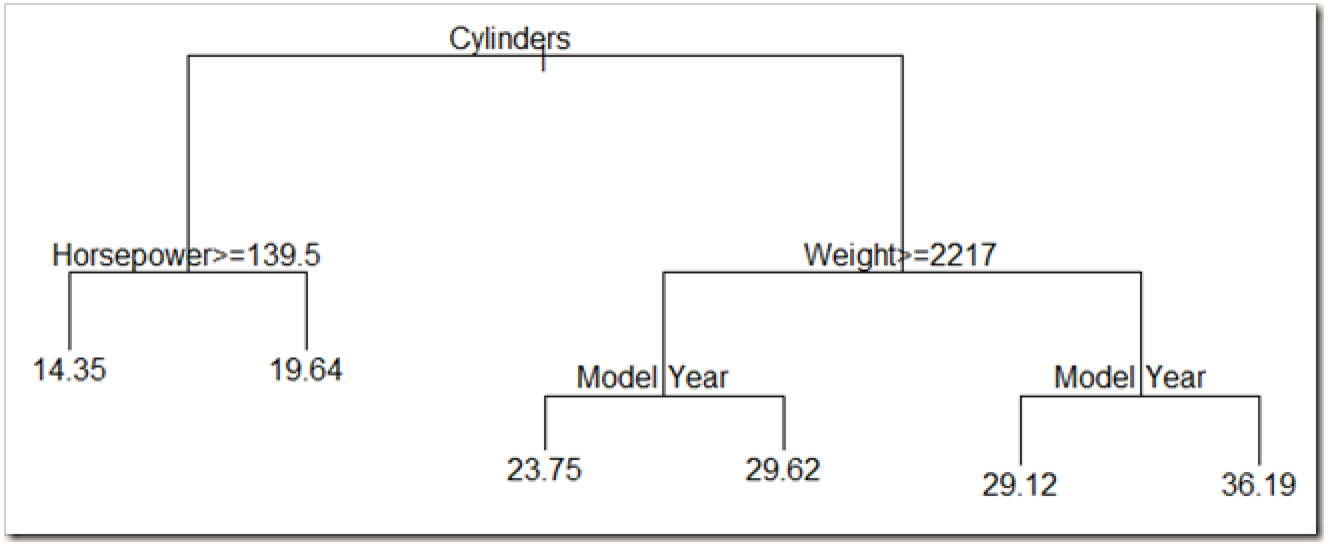
gives better performance, 왜냐
pruning을 무조건 bottom부터 해야하는게 아니라 내가 원하는곳에 flexibly하게 할 수 ㅇㅇ
그래서, gives you more exploreing

Pruning make you enter in different source space?

Regression Tree

- Similar ideas can be applied for regression problems
- Prediction is computed as the average of the target values of all examples in the leave node
- Uncertainty is measured by sum of squared errors

Example Regression Tree



Predicting MPG of a car given its # of cylinders, horsepower, weight, and model year

Summary

- Decision tree is a very flexible classifier
  - Can model arbitrarily complex decision boundaries
  - By changing the depth of the tree (or # of nodes in the tree), we can increase of decrease the model complexity
  - Handle both continuous and discrete features
  - Handle both classification and regression problems
- Learning of the decision tree
  - Greedy top-down induction
  - Not guaranteed to find an optimal decision tree
- DT can overfitting to noise and outliers
  - Can be controlled by early stopping or post pruning