# Decision Trees

Lecturer: Ji Liu

Thank Jerry Zhu for sharing his slides

[ Some slides from Andrew Moore http://www.cs.cmu.edu/~awm/tutorials and Chuck Dyer, with permission.]

# $x$

- The input
- These names are the same: example, point, instance, item, input
- Usually represented by a feature vector
  - These names are the same: attribute, feature
  - For decision trees, we will especially focus on *discrete* features (though continuous features are possible, see end of slides)

# $y$

- The output
- These names are the same: label, target, goal
- It can be
  - Continuous, as in our population prediction ➔ Regression
  - Discrete, e.g., is this mushroom x edible or poisonous? ➔ Classification

# Evaluating classifiers

- During training
  - Train a classifier from a training set $(x_1,y_1)$, $(x_2,y_2)$, …, $(x_n,y_n)$.
- During testing
  - For new test data $x_{n+1} \dots x_{n+m}$, your classifier generates predicted labels $y'_{n+1} \dots y'_{n+m}$
- Test set accuracy:
  - You need to know the true test labels $y_{n+1} \dots y_{n+m}$
  - Test set accuracy: $acc = \frac{1}{m} \sum_{i=n+1}^{n+m} 1_{y_i = y'_i}$
  - Test set error rate = 1 – acc

# Decision Trees

- One kind of classifier (supervised learning)
- Outline:
  - The tree
  - Algorithm
  - Mutual information of questions
  - Overfitting and Pruning
  - Extensions: real-valued features, tree→rules, pro/con

# Akinator: Decision Tree

- http://en.akinator.com/personnages/

# A Decision Tree

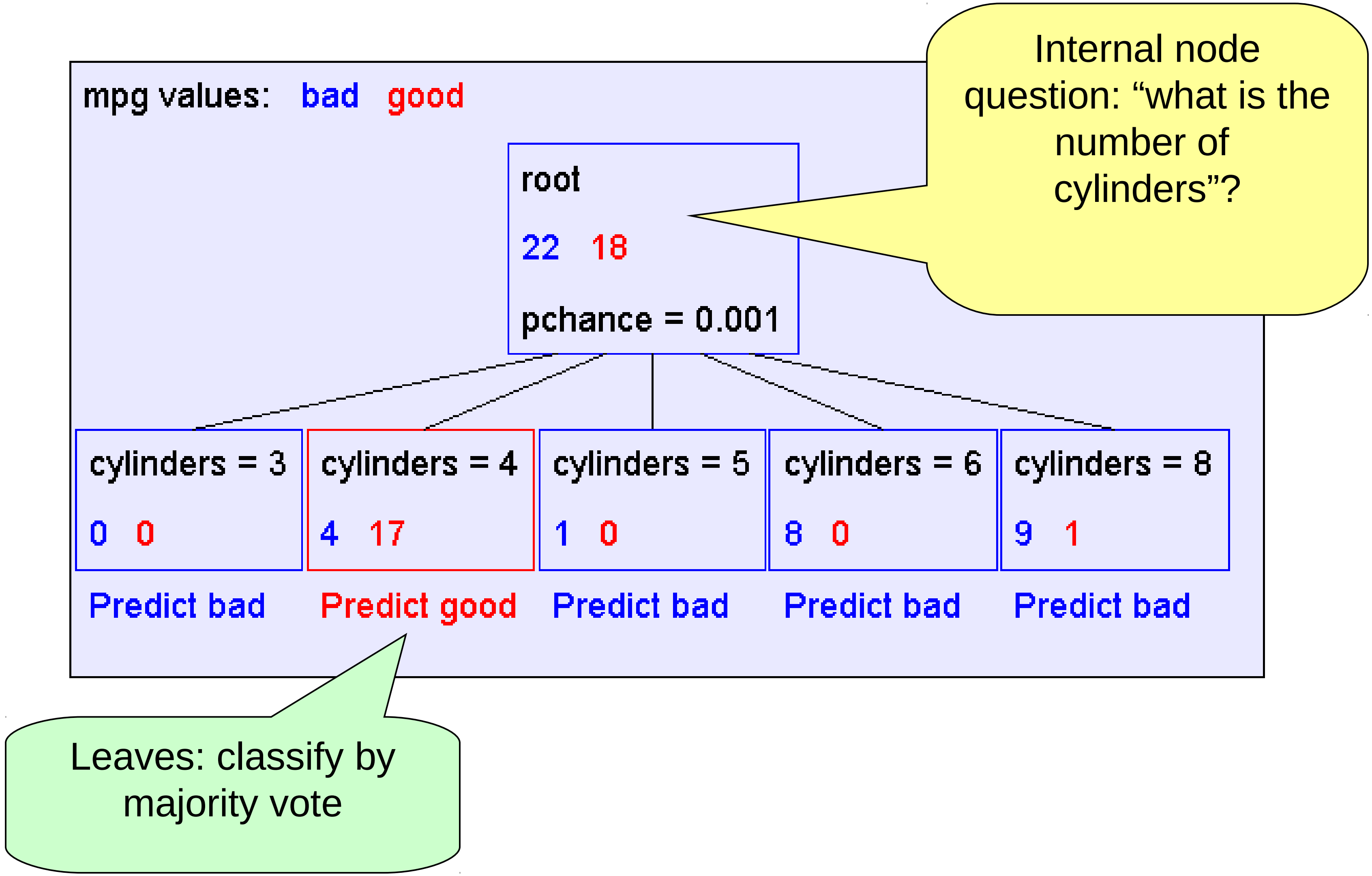
- A decision tree has 2 kinds of nodes

1. Each leaf node has a class label, determined by majority vote of training examples reaching that leaf.
2. Each internal node is a question on features.  It branches out according to the answers.

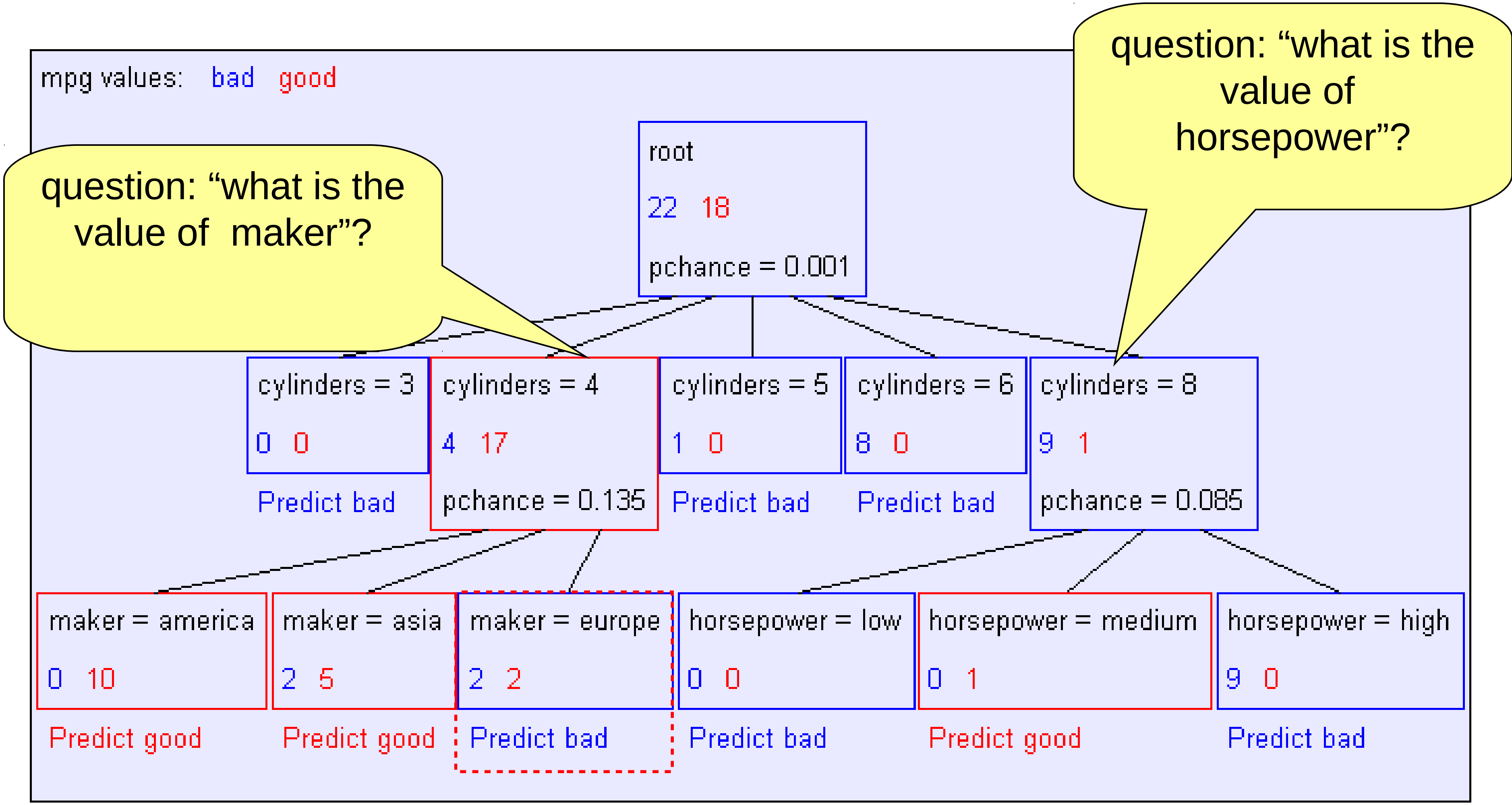
# Automobile Miles-per-gallon prediction


| mpg | cylinders | displacement | horsepower | weight | acceleration | modelyear | maker |
|---|---|---|---|---|---|---|---|
| good | 4 | low | low | low | high | 75to78 | asia |
| bad | 6 | medium | medium | medium | medium | 70to74 | america |
| bad | 4 | medium | medium | medium | low | 75to78 | europe |
| bad | 8 | high | high | high | low | 70to74 | america |
| bad | 6 | medium | medium | medium | medium | 70to74 | america |
| bad | 4 | low | medium | low | medium | 70to74 | asia |
| bad | 4 | low | medium | low | low | 70to74 | asia |
| bad | 8 | high | high | high | low | 75to78 | america |
| : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : |
| bad | 8 | high | high | high | low | 70to74 | america |
| good | 8 | high | medium | high | high | 79to83 | america |
| bad | 8 | high | high | high | low | 75to78 | america |
| good | 4 | low | low | low | low | 79to83 | america |
| bad | 6 | medium | medium | medium | high | 75to78 | america |
| good | 4 | medium | low | low | low | 79to83 | america |
| good | 4 | low | low | medium | high | 79to83 | america |
| bad | 8 | high | high | high | low | 70to74 | america |
| good | 4 | low | medium | low | medium | 75to78 | europe |
| bad | 5 | medium | medium | medium | medium | 75to78 | europe |

# A very small decision tree

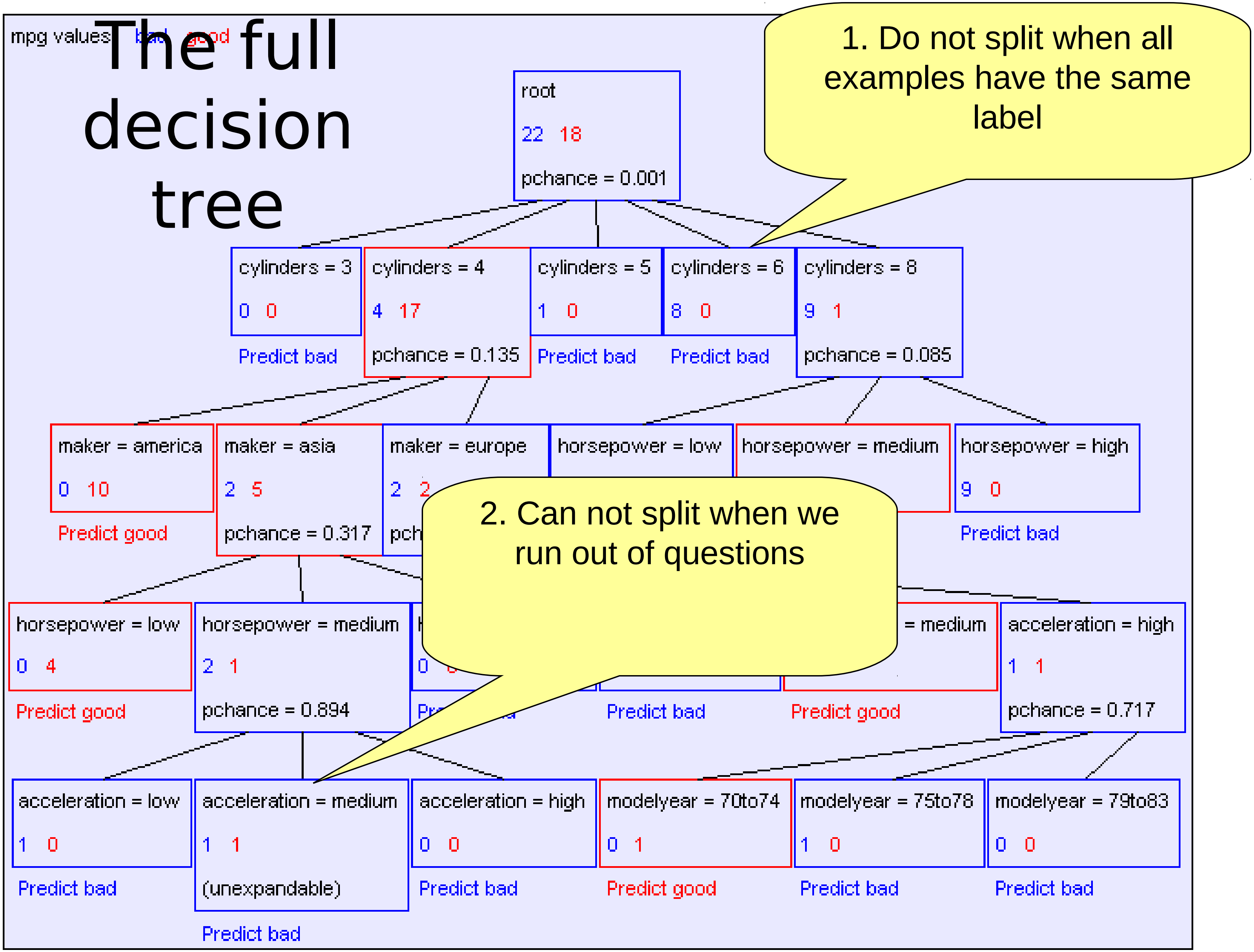


# A bigger decision tree




Predict "good" is also reasonable by following its parent node instead of the root node.

# The full decision tree



# Decision tree algorithm

**buildtree**(*examples*, *questions*, *default*)

/* examples: a list of training examples

questions: a set of candidate questions, e.g., "what's the value of feature $x_i$?"

default: default label prediction, e.g., over-all majority vote */

IF empty(*examples*) THEN return(*default*)

IF (*examples* have same label y) THEN return(y)

IF empty(*questions*) THEN return(majority vote in *examples*)

q = best_question(*examples, questions*)

Let there be n answers to q

- Create and return an internal node with n children
- The $i^{th}$ child is built by calling

  **buildtree**({*example*|q=$i^{th}$ answer}, *questions*\{q}, *default*)

# The best question

- What do we want: pure leaf nodes, i.e. all examples having (almost) the same y.
- A good question → a split that results in pure child nodes
- How do we measure the degree of purity induced by a question?  Here's one possibility (Max-Gain in book):

mutual information
(a.k.a. information gain)
A quantity from information theory

# Entropy (Impurity Measure)

- At the current node, there are $n=n_1+\ldots+n_k$ examples
  - $n_1$ examples have label $y_1$
  - $n_2$ examples have label $y_2$
  - …
  - $n_k$ examples have label $y_k$
- What's the impurity of the node?
- Turn it into a game: if I put these examples in a bag, and grab one at random, what is the probability the example has label $y_i$?

# Entropy (Impurity Measure)

- Probability estimated from samples:
  - with probability $p_1=n_1/n$ the example has label $y_1$
  - with probability $p_2=n_2/n$ the example has label $y_2$
  - …
  - with probability $p_k=n_k/n$ the example has label $y_k$
- $p_1+p_2+\ldots+p_k=1$
- The “outcome” of the draw is a random variable y with probability ($p_1$, $p_2$, …, $p_k$)
- What’s the impurity of the node ➔ what’s the uncertainty of y in a random drawing?

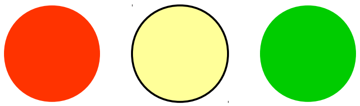
# Entropy (Impurity Measure)

$$H(Y) = \sum_{i=1}^{k} -\Pr(Y = y_i) \log_2 \Pr(Y = y_i)$$
$$= \sum_{i=1}^{k} -p_i \log_2 p_i.$$

- Interpretation: The number of yes/no questions (bits) needed on average to pin down the value of y in a random drawing


H(y)=


H(y)=



H(y)=

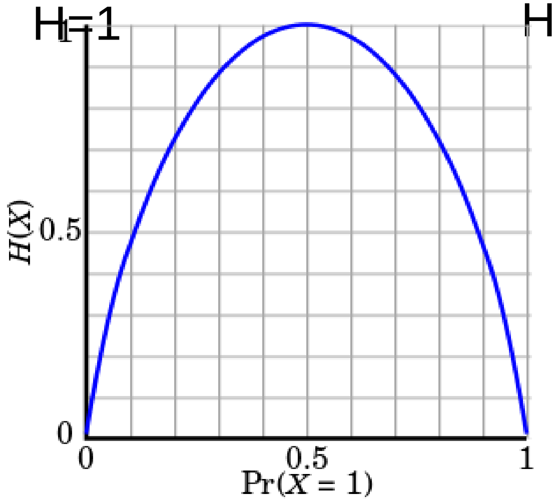
# Entropy (Impurity Measure)


**Jerry's coin**

p(head)=0.5
p(tail)=0.5
H=1

p(head)=0.51
p(tail)=0.49
H=0.9997

p(head)=1
p(tail)=0
H=0 (Why?)



# Excellent Video for Entropy

https://www.youtube.com/watch?v=R4OlXb9aTvQ

- Entropy roughly measures the average number of yes/no questions we need to ask to figure out the class label of an object without any additional attribute information.

# Conditional entropy

$$H(Y|X=v) = \sum_{i=1}^{k} -\Pr(Y=y_i|X=v)\log_2 \Pr(Y=y_i|X=v)$$

$$H(Y|X) = \sum_{v:\text{values of } X} \Pr(X=v)\,H(Y|X=v)$$

- Y: label. X: a question (e.g., a feature). v: an answer to the question
- Pr(Y|X=v): conditional probability
- H(Y|X) estimates the average number of y/n questions required after know the attribute information X

# Information gain

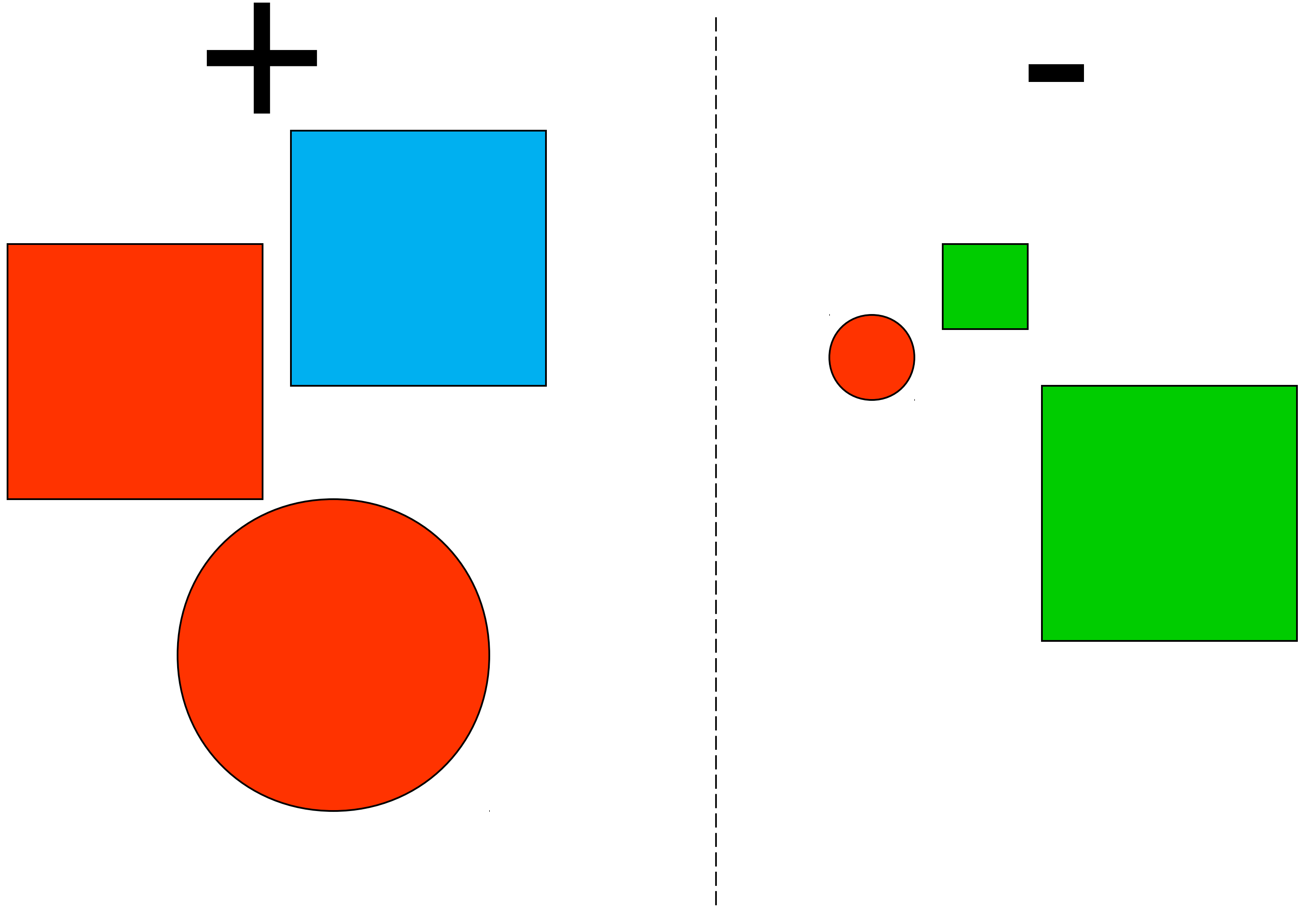
- Information gain, or mutual information

$$I(Y;X) = H(Y) - H(Y|X)$$

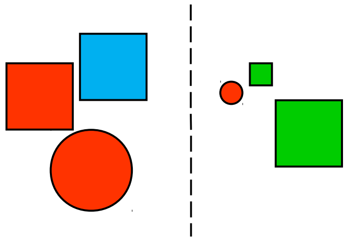
- Choose question (feature) X which maximizes I(Y;X).

# Example

- Features: color, shape, size
- What's the best question at root?



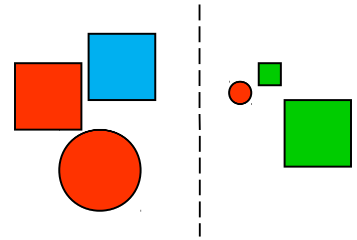
# The training set



| Example | Color | Shape | Size | Class |
|---|---|---|---|---|
| 1 | Red | Square | Big | + |
| 2 | Blue | Square | Big | + |
| 3 | Red | Circle | Big | + |
| 4 | Red | Circle | Small | - |
| 5 | Green | Square | Small | - |
| 6 | Green | Square | Big | - |

H(class)=
H(class | color)=

| Example | Color | Shape | Size | Class |
|---|---|---|---|---|
| 1 | Red | Square | Big | + |
| 2 | Blue | Square | Big | + |
| 3 | Red | Circle | Big | + |
| 4 | Red | Circle | Small | - |
| 5 | Green | Square | Small | - |
| 6 | Green | Square | Big | - |



$H(\text{class}) = H(3/6, 3/6) = 1$

$H(\text{class} \mid \text{color}) = 3/6 * H(2/3, 1/3) + 1/6 * H(1, 0) + 2/6 * H(0, 1)$

3 out of 6 are red

2 of the red are +

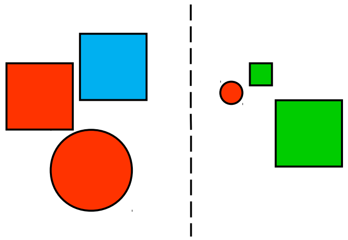
1 out of 6 is blue

blue is +

2 out of 6 are green

green is -

| Example | Color | Shape | Size | Class |
| --- | --- | --- | --- | --- |
| 1 | Red | Square | Big | + |
| 2 | Blue | Square | Big | + |
| 3 | Red | Circle | Big | + |
| 4 | Red | Circle | Small | - |
| 5 | Green | Square | Small | - |
| 6 | Green | Square | Big | - |

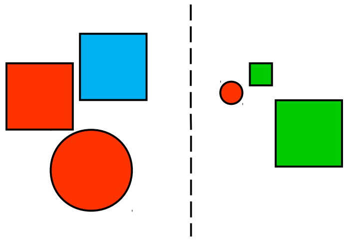


H(class)= H(3/6,3/6) = 1

H(class | color)= 3/6 * H(2/3,1/3) + 1/6 * H(1,0) + 2/6 * H(0,1)

I(class; color) = H(class) – H(class | color) = 0.54 bits

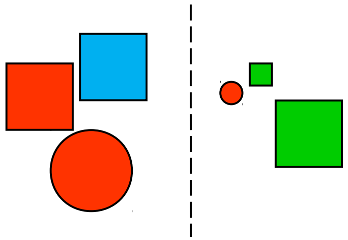
| Example | Color | Shape | Size | Class |
|---|---|---|---|---|
| 1 | Red | Square | Big | + |
| 2 | Blue | Square | Big | + |
| 3 | Red | Circle | Big | + |
| 4 | Red | Circle | Small | - |
| 5 | Green | Square | Small | - |
| 6 | Green | Square | Big | - |



H(class)= H(3/6,3/6) = 1
H(class | shape)= 4/6 * H(1/2, 1/2) + 2/6 * H(1/2,1/2)
I(class; shape) = H(class) – H(class | shape) = 0 bits

Shape tells us nothing about the class!

| Example | Color | Shape | Size | Class |
|---|---|---|---|---|
| 1 | Red | Square | Big | + |
| 2 | Blue | Square | Big | + |
| 3 | Red | Circle | Big | + |
| 4 | Red | Circle | Small | - |
| 5 | Green | Square | Small | - |
| 6 | Green | Square | Big | - |

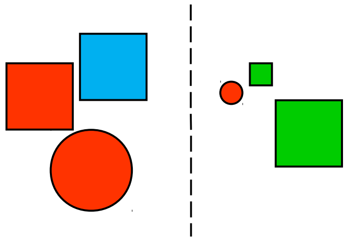


H(class)= H(3/6,3/6) = 1

H(class | size)= 4/6 * H(3/4, 1/4) + 2/6 * H(0,1)

I(class; size) = H(class) – H(class | size) = 0.46 bits

| Example | Color | Shape | Size | Class |
| --- | --- | --- | --- | --- |
| 1 | Red | Square | Big | + |
| 2 | Blue | Square | Big | + |
| 3 | Red | Circle | Big | + |
| 4 | Red | Circle | Small | - |
| 5 | Green | Square | Small | - |
| 6 | Green | Square | Big | - |



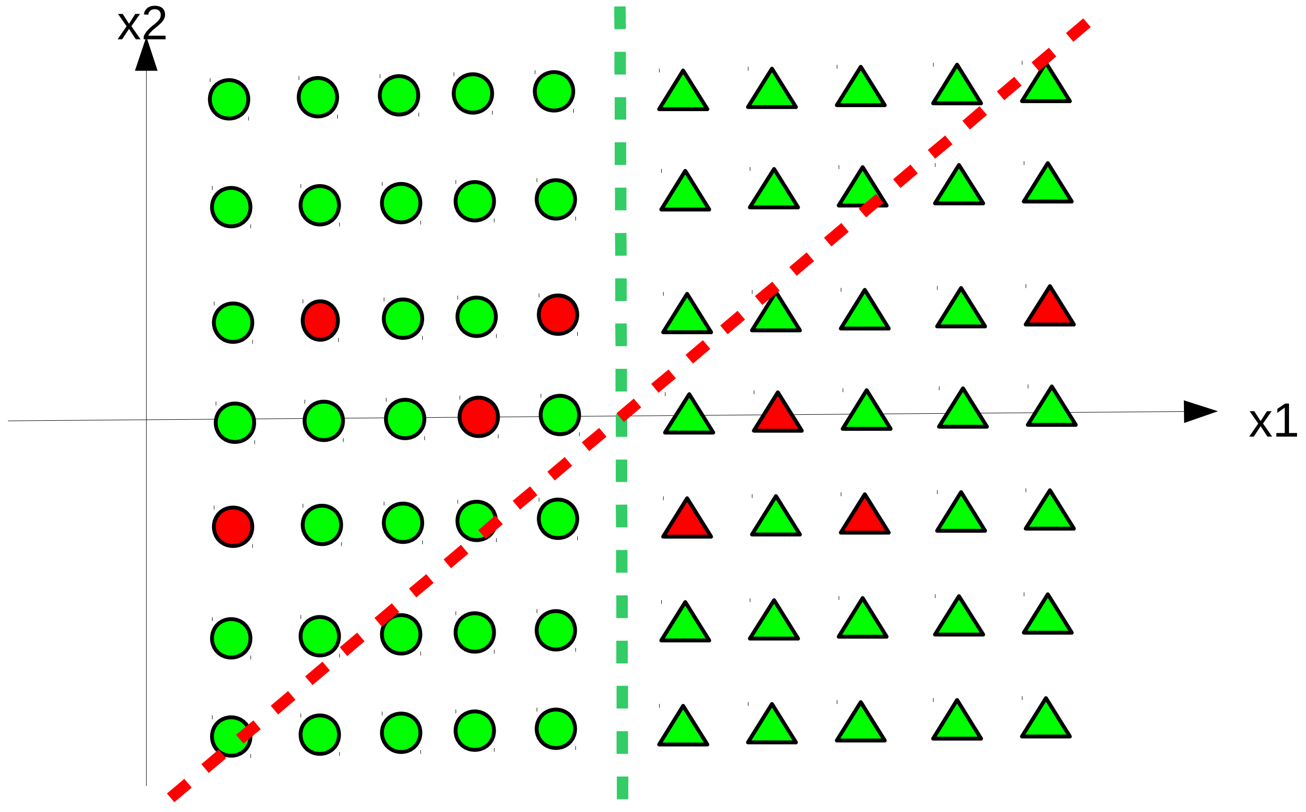
I(class; color) = H(class) – H(class | color) = 0.54 bits
I(class; shape) = H(class) – H(class | shape) = 0 bits
I(class; size) = H(class) – H(class | size) = 0.46 bits

➔ We select color as the question at root

# Overfitting

- Overfitting happens if the prediction model is overcomplicated while the training data is few.
- Another perspective to say overfitting is the model fits the training data perfectly.
- https://www.youtube.com/watch?v=iILj9g8xObc

# Example: Overfitting in SVM



# Example: Overfitting in regression: Predicting US Population

| $x$=Year | $y$=Million |
|---|---|
| 1900 | 75.995 |
| 1910 | 91.972 |
| 1920 | 105.71 |
| 1930 | 123.2 |
| 1940 | 131.67 |
| 1950 | 150.7 |
| 1960 | 179.32 |
| 1970 | 203.21 |
| 1980 | 226.51 |
| 1990 | 249.63 |
| 2000 | 281.42 |

- We have some training data ($n$=11)
- What will the population be in 2020?

# Regression: Polynomial fit

- The degree *d* (complexity of the model) is important

$$f(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$$

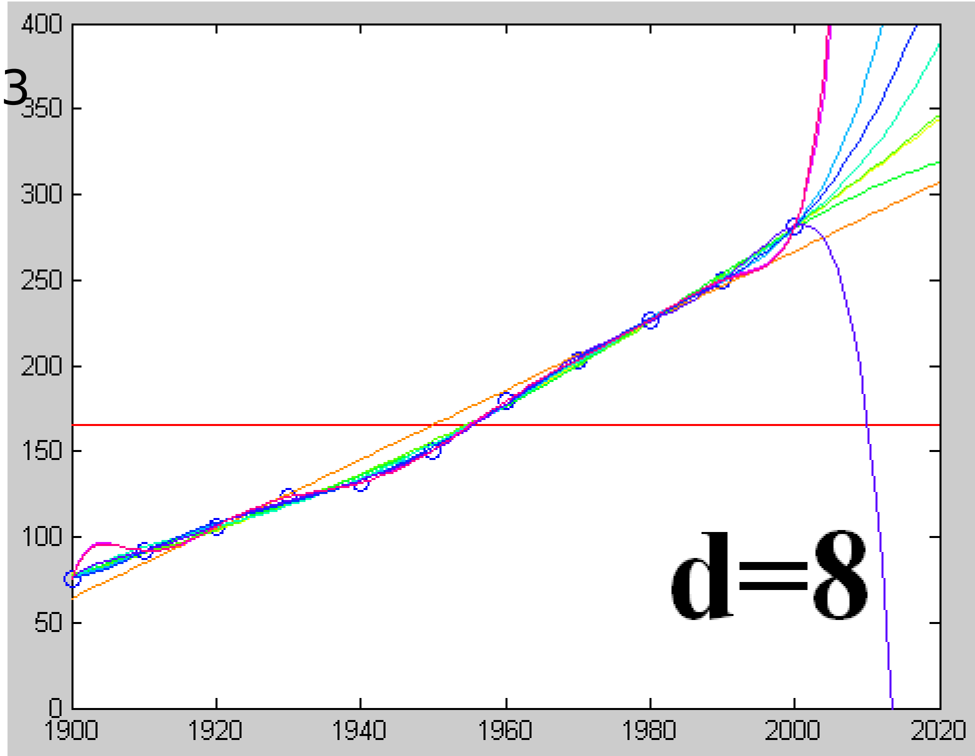
- Fit (=learn) coefficients $c_d$, ... $c_0$ to minimize Mean Squared Error (MSE) on training data

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - f(x_i) \right)^2$$

# Overfitting

- As *d* increases, MSE on training data improves, but prediction outside training data worsens

degree=0 MSE=4181.451643
degree=1 MSE=79.600506
degree=2 MSE=9.346899
degree=3 MSE=9.289570
degree=4 MSE=7.420147
degree=5 MSE=5.310130
degree=6 MSE=2.493168
degree=7 MSE=2.278311
degree=8 MSE=1.257978
degree=9 MSE=0.001433
degree=10 MSE=0.000000

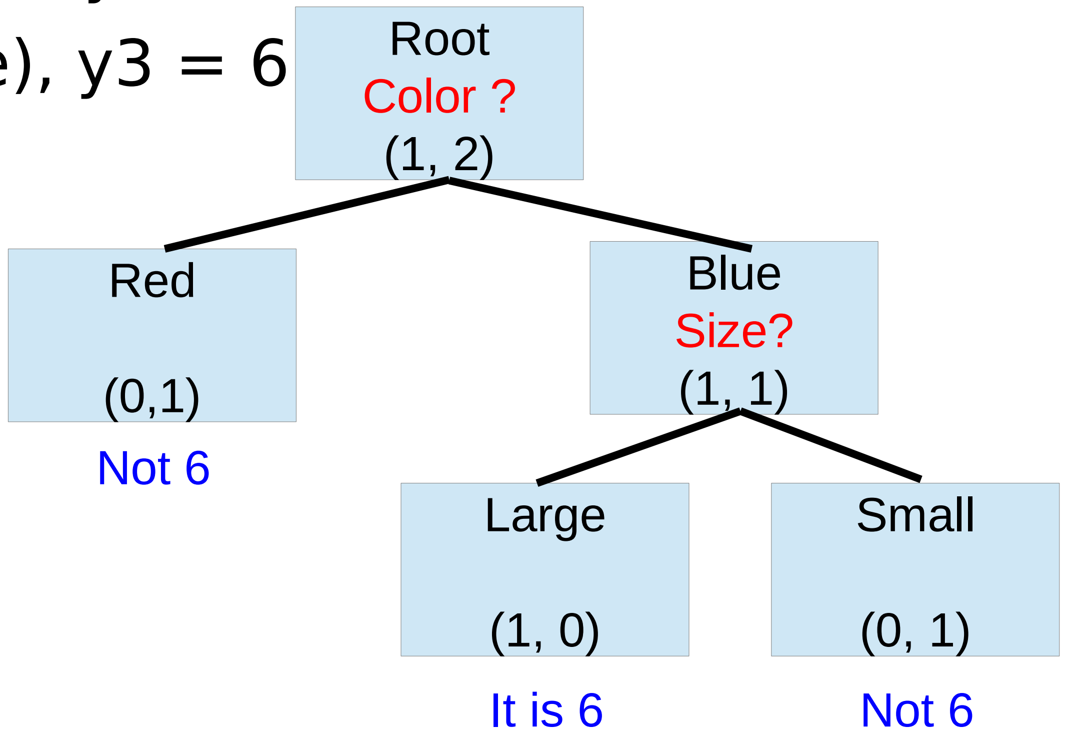


# Overfitting: Toy Example

- Predict if the outcome of throwing a die is “6” from its (color, size)
- Color = {red, blue}, Size={small, large}
- Three training samples:
  - X1 = (red, large), y1 = not 6
  - X2 = (blue, small), y2 = not 6
  - X3 = (blue, large), y3 = 6

# Overfitting: Example for Decision Tree

- Three training samples:
  - X1 = (red, large), y1 = not 6
  - X2 = (blue, small), y2 = not 6
  - X3 = (blue, large), y3 = 6

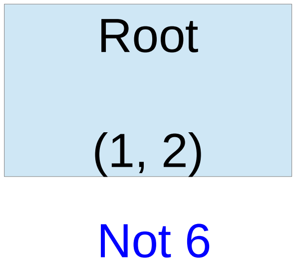


# Toy Example

- Assume “color” and “size” are independent attributes for any die
- Assume P(red)=P(blue)=1/2, P(large)=P(small)=1/2
- The prediction accuracy for this decision tree is 1-(1/2*1/6+1/4*5/6 + 1/4*1/6)=2/3

# Toy Example

- If the decision tree only has the root node, we predict all new instances as "Not 6".
- The accuracy is 5/6 > 2/3



# Overfit a decision tree

Five inputs, all bits, are generated in all 32 possible combinations

Output y = copy of e,
Except a random 25% of the records have y set to the opposite of e

32 records

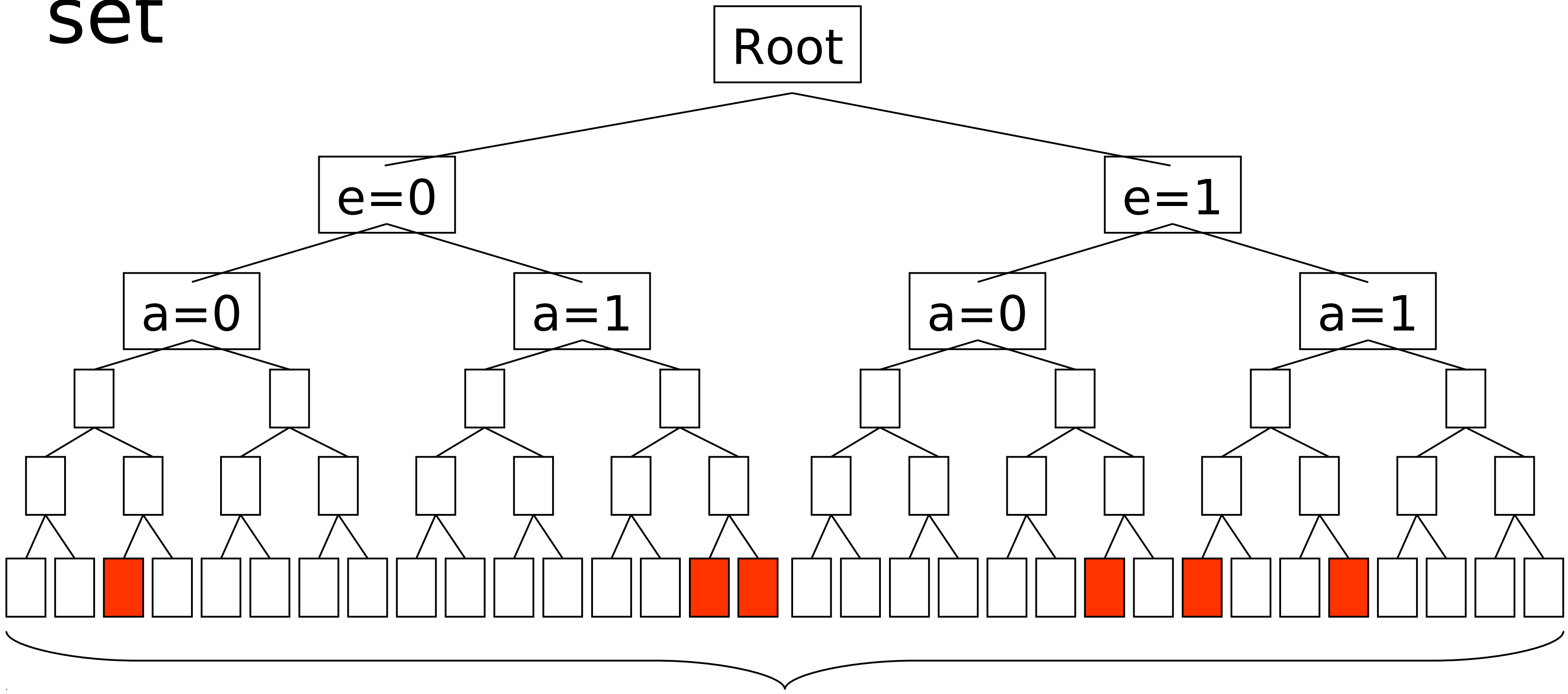
| a | b | c | d | e | y |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 1 |

# Overfit a decision tree

- The test set is constructed similarly
  - $y=e$, but 25% the time we corrupt it by $y=\neg e$
  - The corruptions in training and test sets are independent
- The training and test sets are the same, except
  - Some y's are corrupted in training, but not in test
  - Some y's are corrupted in test, but not in training

# Overfit a decision tree

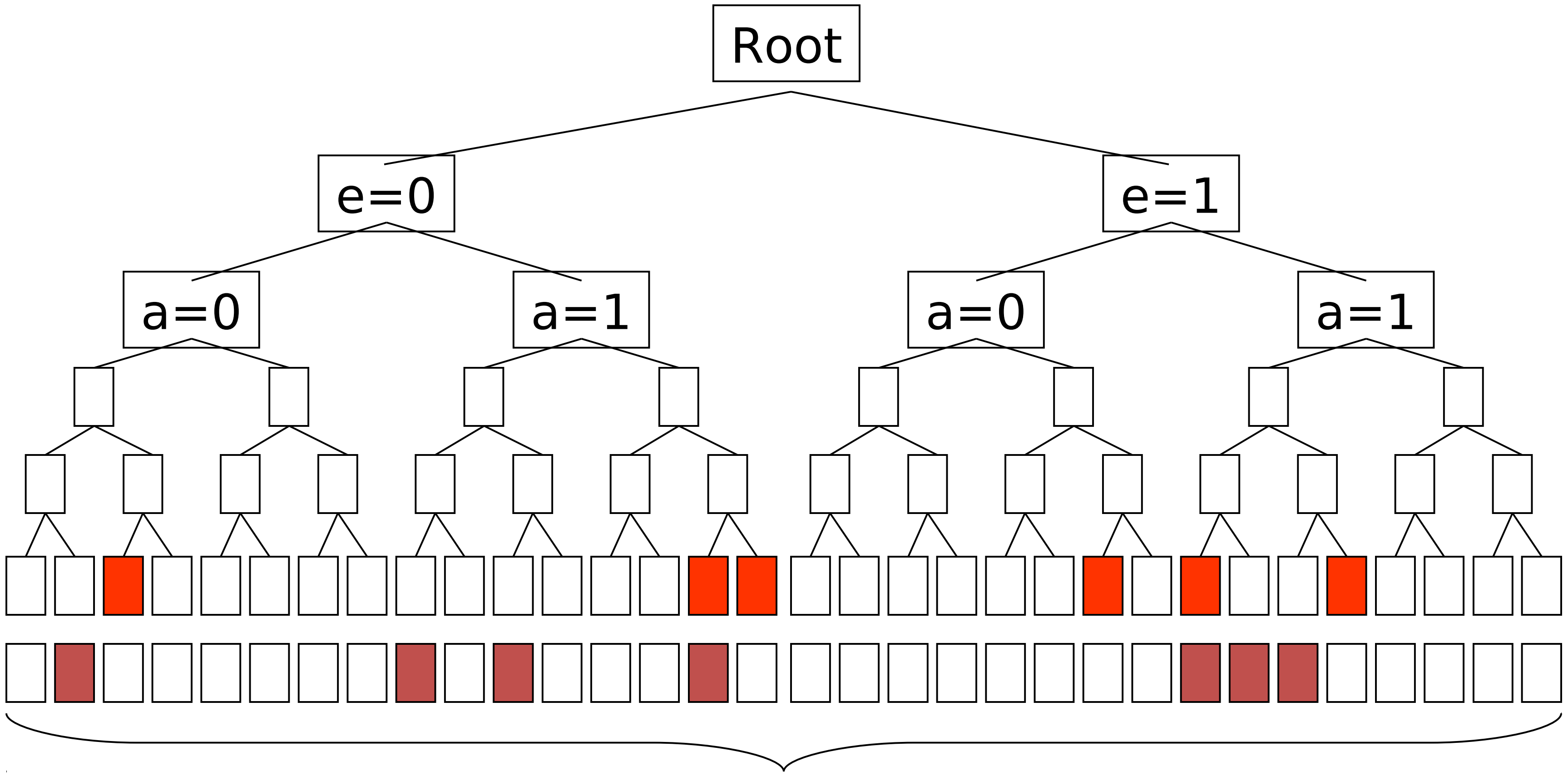
- We build a full tree on the training set




Training set accuracy = 100%
25% of these training leaf node labels will be corrupted (≠e)

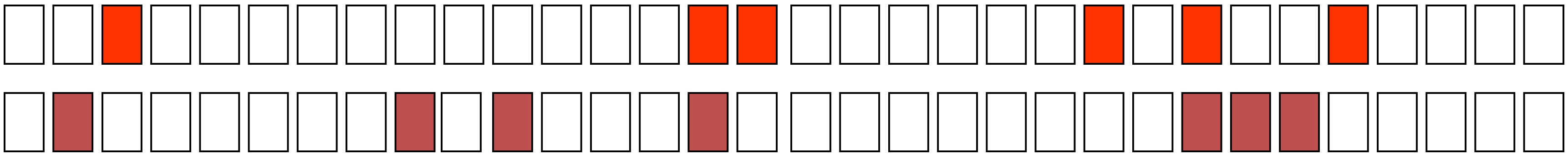
# Overfit a decision tree

- And classify the test data with the tree




25% of the test examples are corrupted – independent of training data

# Overfit a decision tree



On average:

- ¾ training data uncorrupted
  - ¾ of these are uncorrupted in test – correct labels
  - ¼ of these are corrupted in test – wrong
- ¼ training data corrupted
  - ¾ of these are uncorrupted in test – wrong
  - ¼ of these are also corrupted in test – correct labels
- Test accuracy = ¾ * ¾ + ¼ * ¼ = 5/8 = 62.5%

# Overfit a decision tree

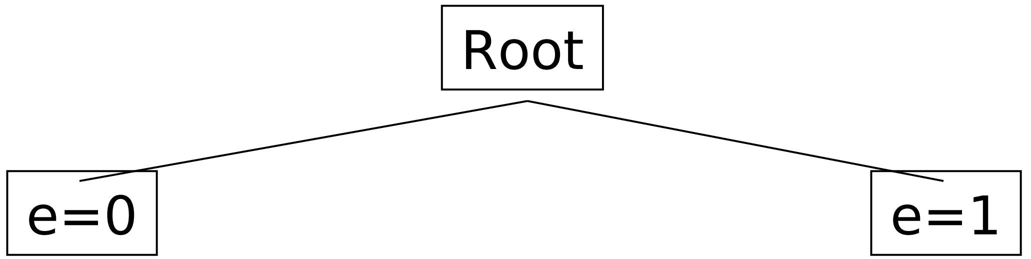
- But if we knew a,b,c,d are irrelevant features and don't use them in the tree…

Pretend they don't exist

| a | b | c | d | e | y |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 1 |

# Overfit a decision tree

- The tree would be



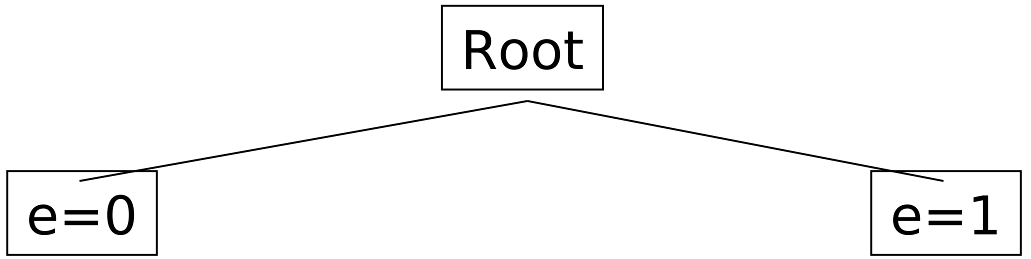

In training data, about ¾ y's are 0 here. Majority vote predicts y=0

In training data, about ¾ y's are 1 here. Majority vote predicts y=1

In test data, ¼ y's are different from e.
test accuracy = ?

# Overfit a decision tree

- The tree would be




In training data, about ¾ y's are 0 here. Majority vote predicts y=0

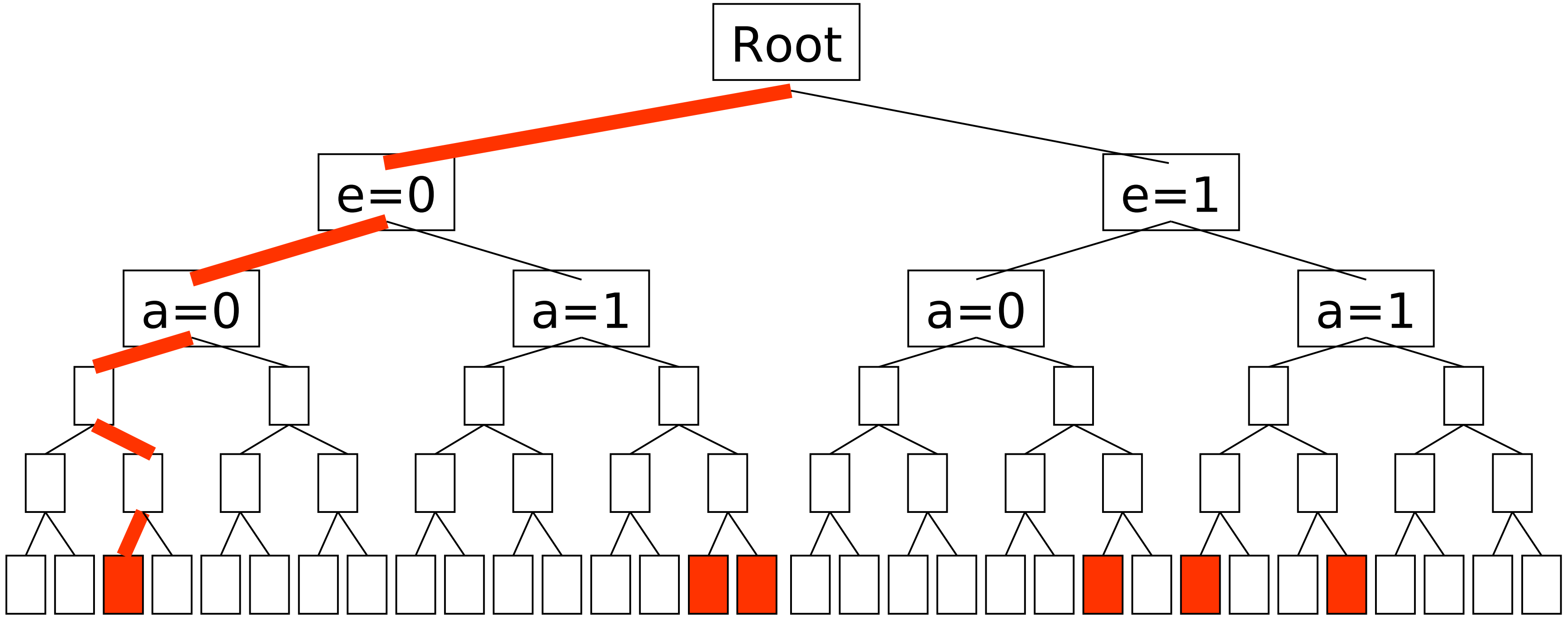
In training data, about ¾ y's are 1 here. Majority vote predicts y=1

In test data, ¼ y's are different from e.
test accuracy = ¾ = 75% (better!)

Full tree test accuracy = ¾ * ¾ + ¼ * ¼ = 5/8 = 62.5%

# Overfit a decision tree

- In the full tree, we overfit by learning non-existent relations (noise)



# Avoid overfitting: pruning

Pruning with a tuning set

1. Randomly split data into TRAIN and TUNE, say 70% and 30%
2. Build a full tree using only TRAIN
3. Prune the tree down on the TUNE set. On the next page you'll see a greedy version.

# Pruning

Prune(tree T, TUNE set)

1. Compute T's accuracy on TUNE, call it A(T)
2. For every internal node N in T:
   a) New tree $T_N$ = copy of T, but prune (delete) the subtree under N.
   b) N becomes a leaf node in $T_N$. The label is the majority vote of TRAIN examples reaching N.
   c) $A(T_N)$ = $T_N$'s accuracy on TUNE
3. Let T* be the tree (among the $T_N$'s and T) with the largest A(). Set $T \leftarrow T^*$ /* prune */
4. Repeat from step 1 until no more improvement available. Return T.

# Real-valued features

- What if some (or all) of the features x1, x2, …, xk are real-valued?
- Example: x1=height (in inches)
- Idea 1: branch on each possible numerical value.

# Real-valued features

- What if some (or all) of the features x1, x2, …, xk are real-valued?
- Example: x1=height (in inches)
- Idea 1: branch on each possible numerical value. (fragments the training data and prone to overfitting)
- Idea 2: use questions in the form of (x1>t?), where t is a threshold. There are fast ways to try all(?) t.

$$H(y|x_i>t?) = p(x_i>t)H(y|x_i>t) + p(x_i \leq t)H(y|x_i \leq t)$$

$$I(y|x_i>t?) = H(y) - H(y|x_i>t?)$$

# What does the feature space look like?

Axis-parallel cuts

# Conclusions

- Decision trees are popular tools for data mining
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Overfitting might happen
- We used decision trees for classification (predicting a categorical output from categorical or real inputs)

# What you should know

- Trees for classification
- Top-down tree construction algorithm
- Information gain
- Overfitting
- Pruning
- Real-valued features