DS 4400

Machine Learning and Data Mining I
Spring 2022

Alina Oprea

Associate Professor

Khoury College of Computer Science

Northeastern University

March 21 2022

Announcements

- Final Project
  - Feedback for proposal will be posted on Gradescope
  - Project milestone due on April 13
    - Report your progress and receive feedback
  - Project video recording (5 minute presentation) due on May 2
  - Project report due on May 2 (6-8 pages)
- Homework 4
  - Release on March 28, due on April 6
- Final exam: April 20

Outline

- Generative classifiers
- Naïve Bayes classifiers
  - Naïve Bayes assumption
  - Laplace smoothing
  - Comparison to LDA
- Decision trees
  - Information gain / entropy measures
  - Training algorithm

Recap: ML Concepts

- Supervised vs unsupervised learning
- Classification vs regression
- Linear vs non-linear classifiers
- Generative vs discriminative classifiers
- Loss functions
- Metrics for evaluation

Generative vs Discriminative

- Generative model
  - Given X and Y, learns the joint probability $P(X, Y)$
  - Can generate more examples from distribution
  - Examples: LDA, Naïve Bayes, language models (GPT-2, GPT-3, BERT)

- Discriminative model
  - Given X and Y, learns a decision function for classification

Generative classifiers based on Bayes Theorem

LDA Training and Testing

Given training data $(x_i, y_i), i = 1, \ldots, n, y_i \in \{1, \ldots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k/n.$$

Given testing point x, predict k that maximizes:

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

Naïve Bayes Classifier

Idea: Use the training data to estimate

$$P(X \mid Y) \quad \text{and} \quad P(Y) \, .$$

Then, use Bayes rule to infer $P(Y|X_{\text{new}})$ for new data

Easy to estimate
from data

Impractical, but necessary

$$P[Y = k|X = x] \quad = \quad \frac{P[Y = k]\, P[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k]}{P[X_1 = x_1 \wedge \cdots \wedge X_d = x_d]}$$

Unnecessary, as it turns out

Learning Joint Distributions

Step 1:

Build a JD table for your attributes in which the probabilities are unspecified

A	B	C	Prob
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Step 2:

Then, fill in each row with:

$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

Fraction of all records in which A and B are true but C is false

Example – Learning Joint Probability Distribution

This Joint PD was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	
		rich	0.0245895	
	v1:40.5+	poor	0.0421768	
		rich	0.0116293	
Male	v0:40.5-	poor	0.331313	
		rich	0.0971295	
	v1:40.5+	poor	0.134106	
		rich	0.105933	

Naïve Bayes Classifier

Problem: estimating the joint PD or CPD isn't practical
- Severely overfits, as we saw before

However, if we make the assumption that the attributes are independent given the class label, estimation is easy!

$$\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k] = \prod_{j=1}^{d} P[X_j = x_j | Y = k]$$

- In other words, we assume all attributes are *conditionally independent* given Y
- Often this assumption is violated in practice, but more on that later...

Using the Naïve Bayes Classifier

- Now, we have

$$P[Y = k | X = x] \quad = \quad \frac{P[Y = k] P[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k]}{\boxed{P[X_1 = x_1 \wedge \cdots \wedge X_d = x_d]}}$$

This is constant for a given instance, and so irrelevant to our prediction

Using the Naïve Bayes Classifier

- Now, we have

$$P[Y = k | X = x] \quad = \quad \frac{P[Y = k]P[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k]}{\boxed{P[X_1 = x_1 \wedge \cdots \wedge X_d = x_d]}}$$

This is constant for a given instance, and so irrelevant to our prediction

  - In practice, we use log-probabilities to prevent underflow

- To classify a new point $\mathbf{x}$,

$$h(\mathbf{x}) = \underset{y_k}{\arg\max} \; P(Y = k) \prod_{j=1}^{d} P(X_j = x_j | Y = k)$$

x_j = j^{th} attribute value of $\mathbf{x}$

d

Naïve Bayes Classifier

TRAIN

- For each class label k
  1. Estimate prior $\pi_k = P[Y = k]$ from the data
  2. For each value v of attribute X_j
     - Estimate $P[X_j = v | Y = k]$

TEST on INPUT $x = (x_1, \dots, x_d)$

- For every k, compute the probabilities
- $p_k = P[Y = k] \prod_{j=1}^{d} P[X_j = x_j | Y = k]$
- Classify x to the class k that maximizes p_k

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ directly from the training data by counting!

Sky	Temp	Humid
sunny	warm	normal
sunny	cold	high
rainy	cold	high
sunny	warm	high

Play?
yes
yes
no
yes

Prior: P(Play = Yes) =   P(Play = No) =

Conditional feature distributions
P(Sky = sunny| Play = Yes)=   P(Sky = sunny| Play = No)=
P(Temp = warm|Play = Yes) =   P(Temp = warm|Play = No) =
P(Humid = high|Play = Yes) =   P(Humid = high|Play = No) =

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ directly from the training data by counting!

Sky	Temp	Humid
sunny	warm	normal
sunny	cold	high
rainy	cold	high
sunny	warm	high

Play?
yes
yes
no
yes

Classify new point:

X: Sky = sunny, Temp = cold, Humid = high

Laplace Smoothing

- Notice that some probabilities estimated by counting might be zero
  - Possible overfitting!

Laplace Smoothing

- Notice that some probabilities estimated by counting might be zero
  - Possible overfitting!

- Fix by using Laplace smoothing:
  - Adds 1 to each count

$$P(X_j = v \mid Y = k\ ) = \frac{c_v + 1}{\displaystyle\sum_{v' \in \mathrm{values}(X_j)} c_{v'} + |\mathrm{values}(X_j)|}$$

where

  - c_v is the count of training instances with a value of v for attribute j and class label k
  - $|\mathrm{values}(X_j)|$ is the number of values X_j can take on

Naïve Bayes Classifier

TRAIN

- For each class label k
  1. Estimate prior $\pi_k = P[Y = k]$ from the data
  2. For each value v of attribute X_j

     - Estimate $P[X_j = v | Y = k]$ with Laplace smoothing

TEST on INPUT $x = (x_1, \ldots, x_d)$

- For every k, compute the probabilities
- $P[Y = k] \prod_{j=1}^{d} P[X_j = x_j | Y = k]$
- Classify x to the class k that maximizes the above product

Continuous Features

- Naïve Bayes can be extended to continuous features

- Gaussian Naïve Bayes

  - Here an additional assumption is that each distribution $P[X_j|Y = k]$ is Gaussian $N(\mu_j, \sigma_j)$

  - It estimates the mean and standard deviation from training data

- This leads to a linear classifier

Comparison to LDA

- ## Similarity to LDA
  - Both are generative models
  - They both estimate:
  $$P[X = x \text{ and } Y = k] = P[X = x | Y = k]P[Y = k]$$
  Using Bayes Theorem
- ## Difference from LDA
  - Naïve Bayes can handle discrete data
  - LDA uses multi-variate normal
  - LDA assumes same variances for all classes
  - Naïve Bayes make the conditional independence assumption
  - LDA is linear, while Naïve Bayes is usually not linear

Naïve Bayes Summary

Advantages:

- Fast to train (single scan through data)
- Fast to classify
- Not sensitive to irrelevant features
- Handles real and discrete data
- Handles streaming data well

Disadvantages:

- Assumes independence of features

Outline

- Generative classifiers
- Naïve Bayes classifiers
  - Naïve Bayes assumption
  - Laplace smoothing
  - Comparison to LDA
- Decision trees
  - Information gain / entropy measures
  - Training algorithm

Sample Dataset

- Columns denote features X_i
- Rows denote labeled instances $< x_i, y_i >$
- Class label denotes whether a tennis game was played

$< x_i, y_i >$

Categorical
data

| Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision Tree

- A possible decision tree for the data:
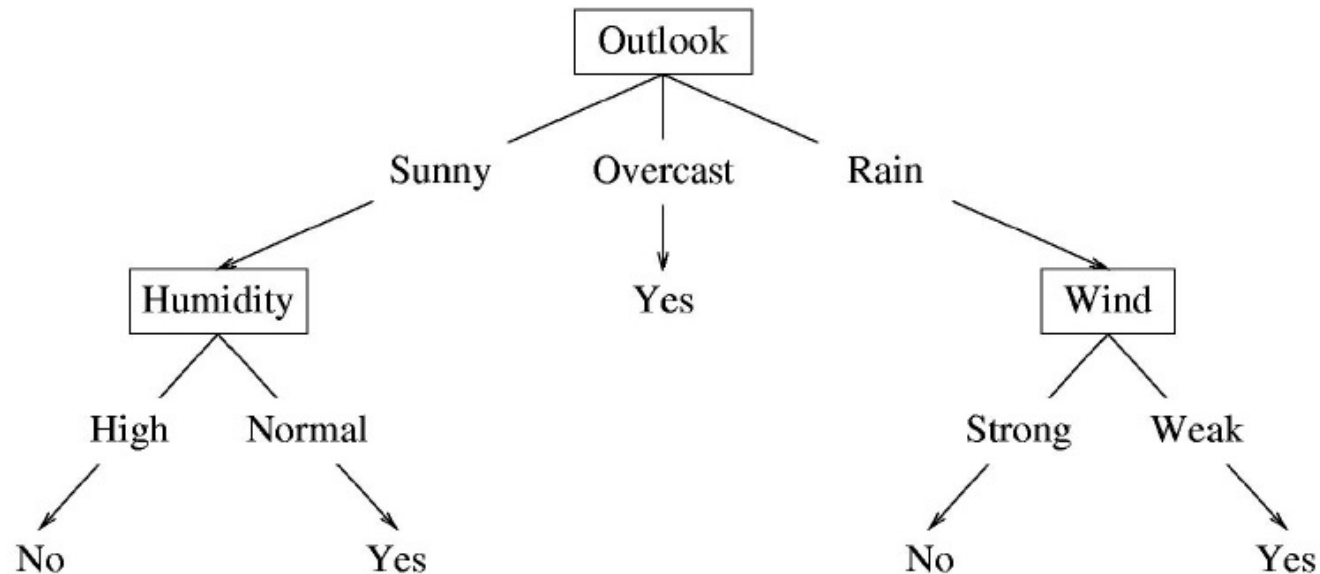
- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y \mid x \in \text{leaf})$)

Decision Tree

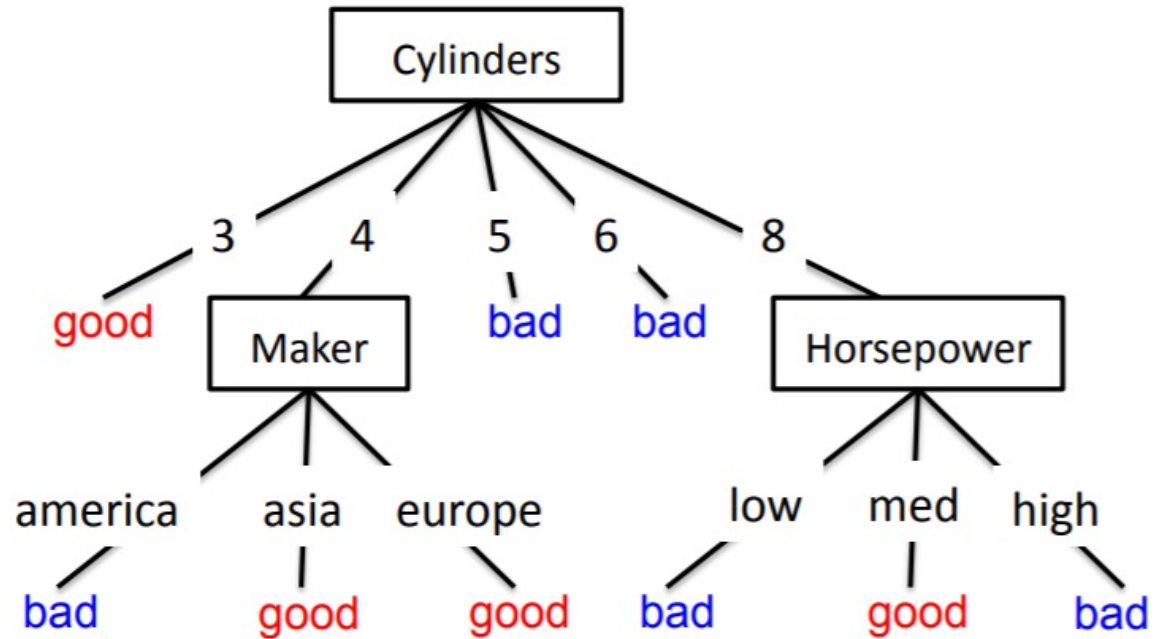- A possible decision tree for the data:
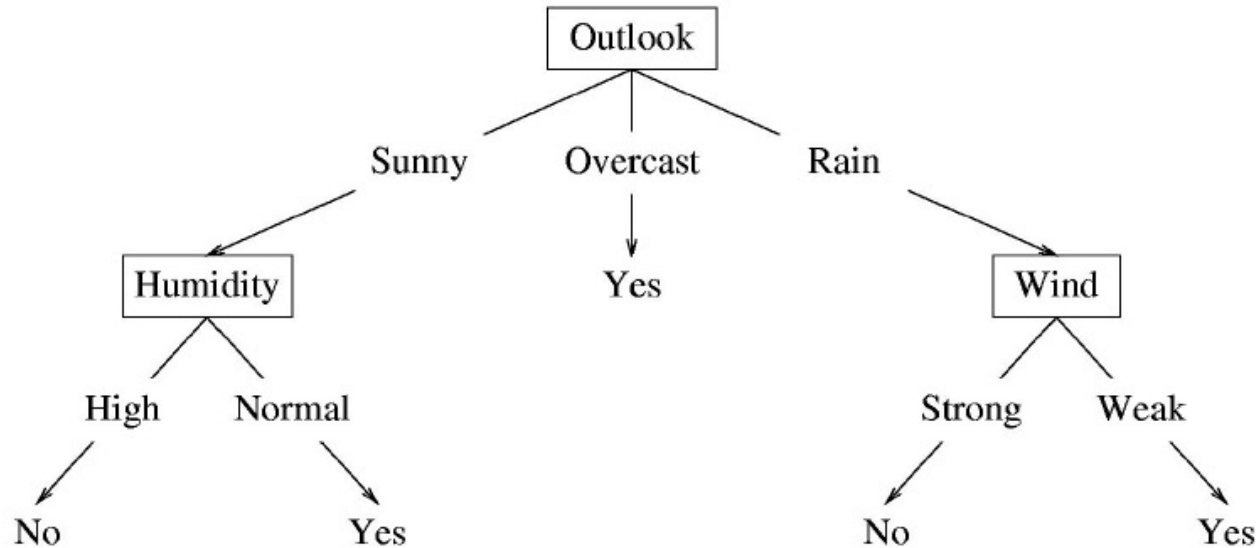
- What prediction would we make for

  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Interpretability

- Each internal node tests an attribute x_i

- One branch for each possible attribute value $x_i=v$

- Each leaf assigns a class y

- To classify input x: traverse the tree from root to leaf, output the labeled y
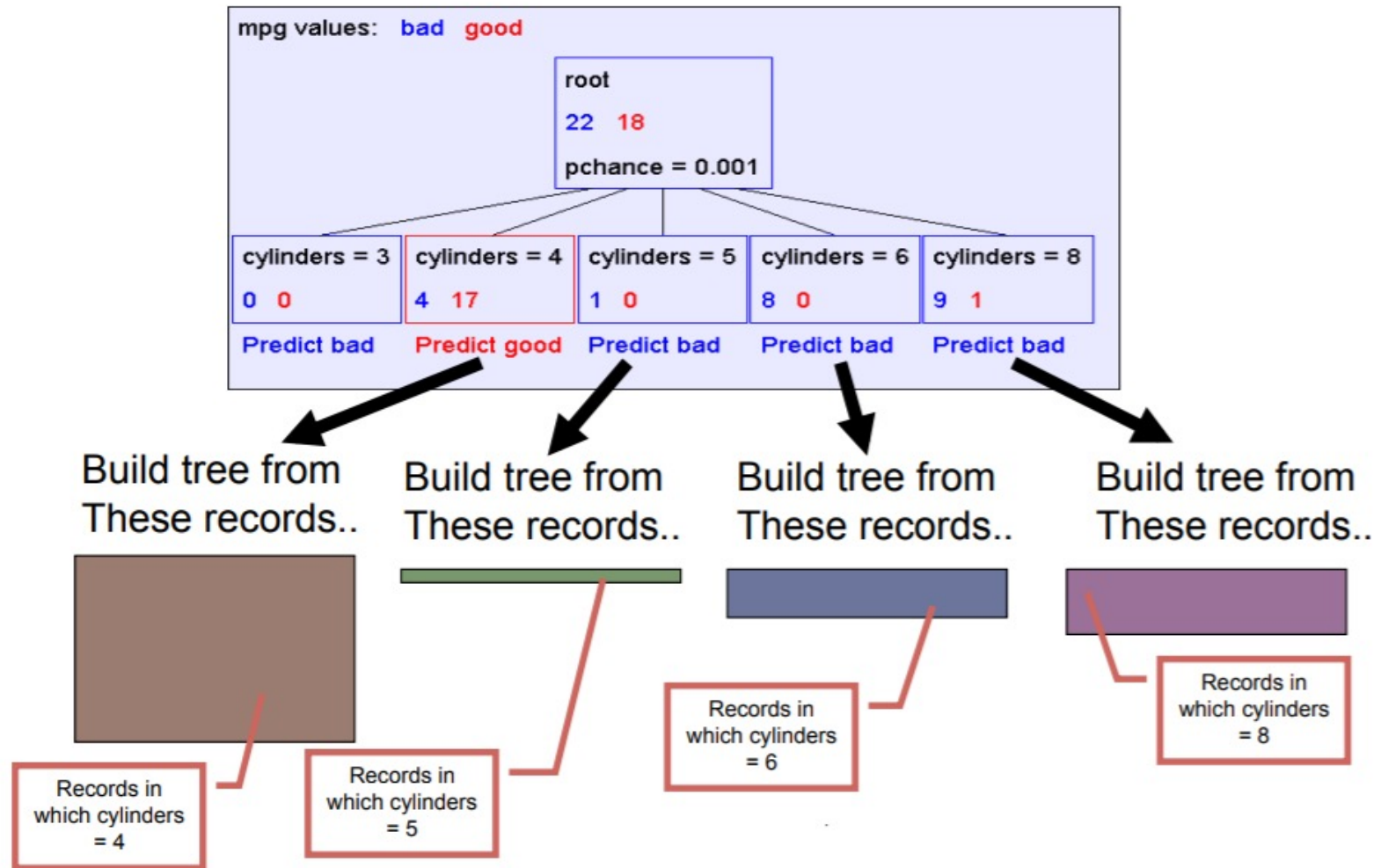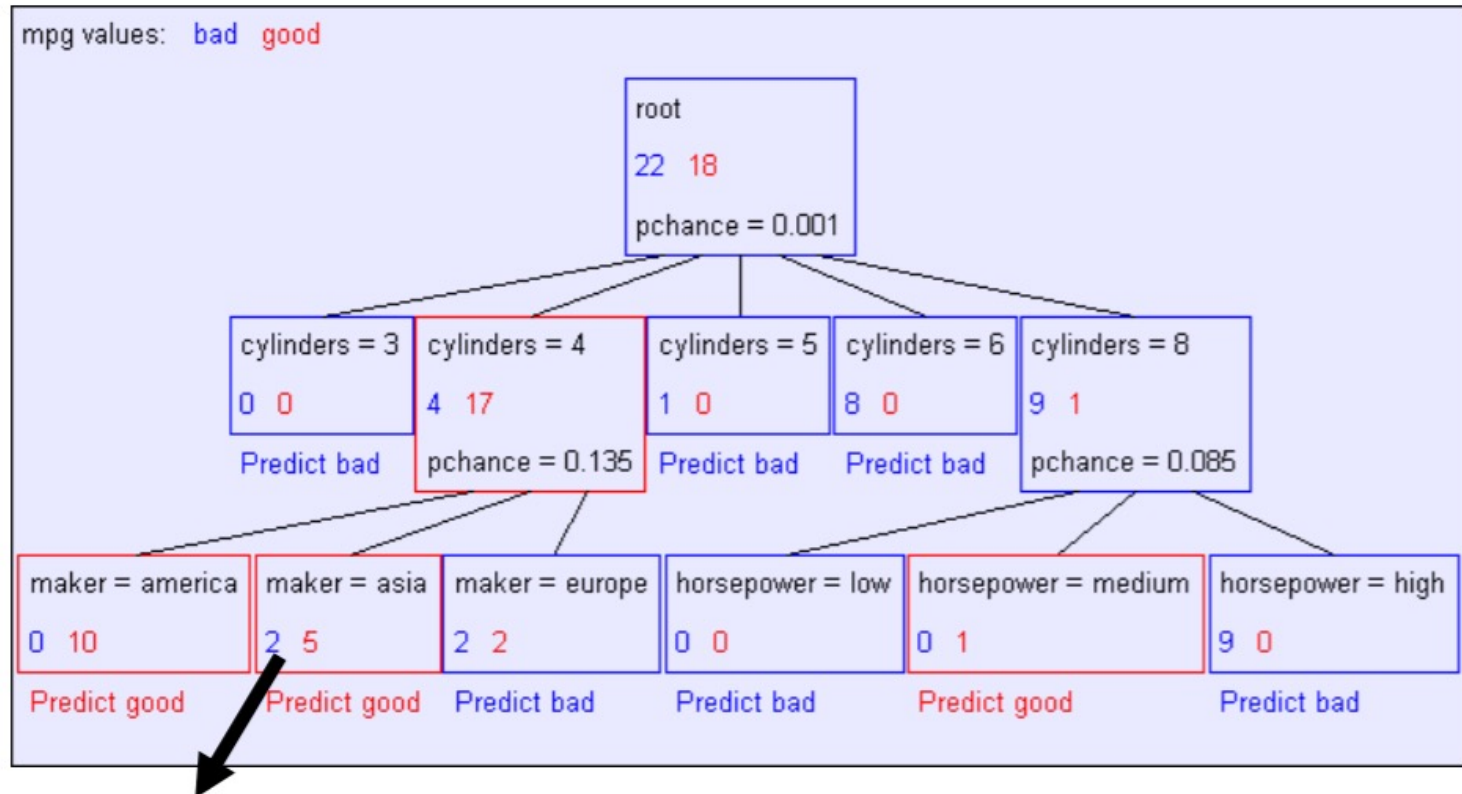

Human interpretable!

Learning Decision Trees

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse

Key Idea: Use Recursion Greedily

Second Level

mpg values: bad   good

root
22 18
pchance = 0.001

cylinders = 3	cylinders = 4	cylinders = 5	cylinders = 6	cylinders = 8
0 0	4 17	1 0	8 0	9 1
Predict bad	pchance = 0.135	Predict bad	Predict bad	pchance = 0.085

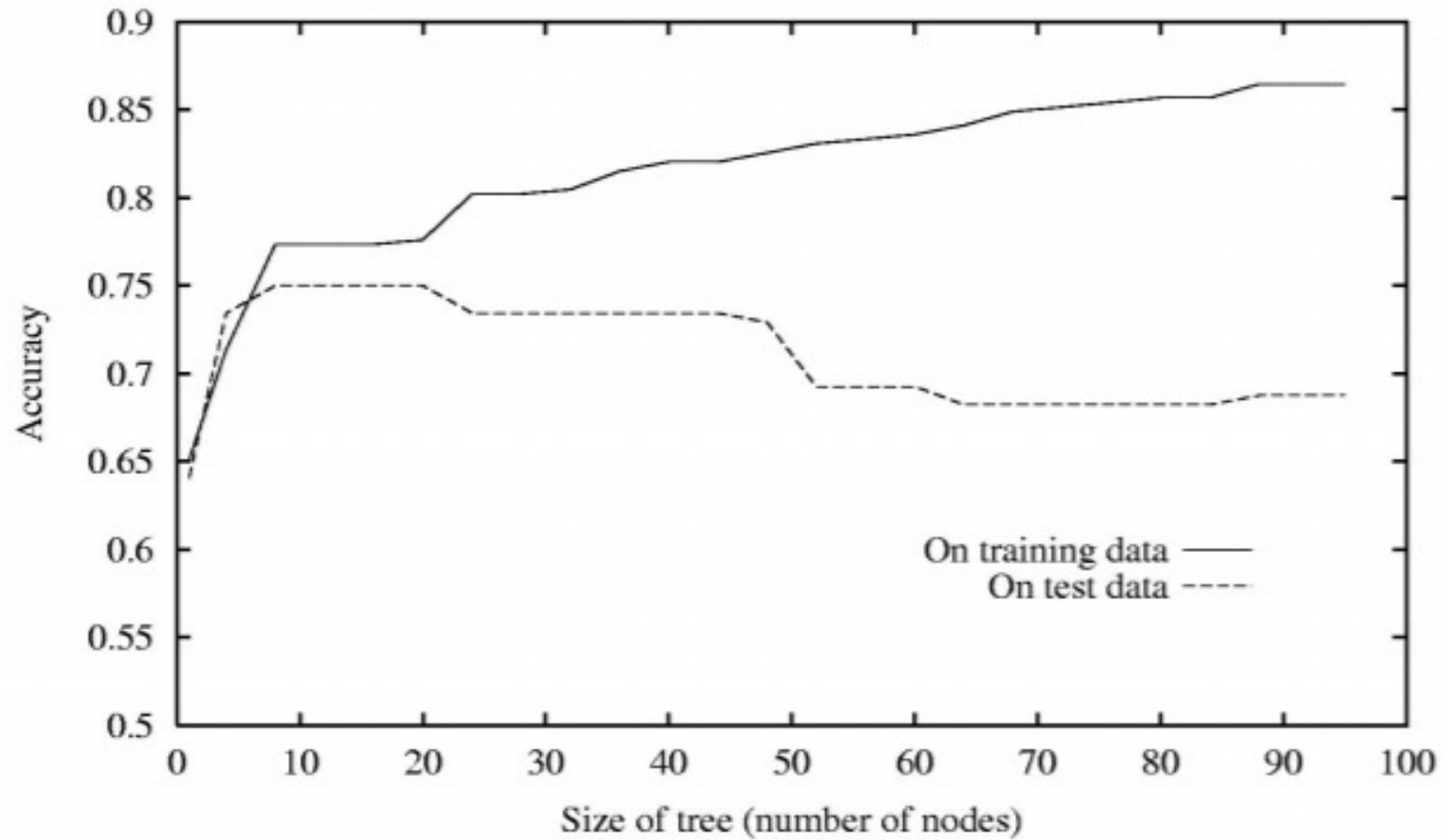| maker = america | maker = asia | maker = europe | horsepower = low | horsepower = medium | horsepower = high |
| 0 10 | 2 5 | 2 2 | 0 0 | 0 1 | 9 0 |
| Predict good | Predict good | Predict bad | Predict bad | Predict good | Predict bad |

Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

Full Tree
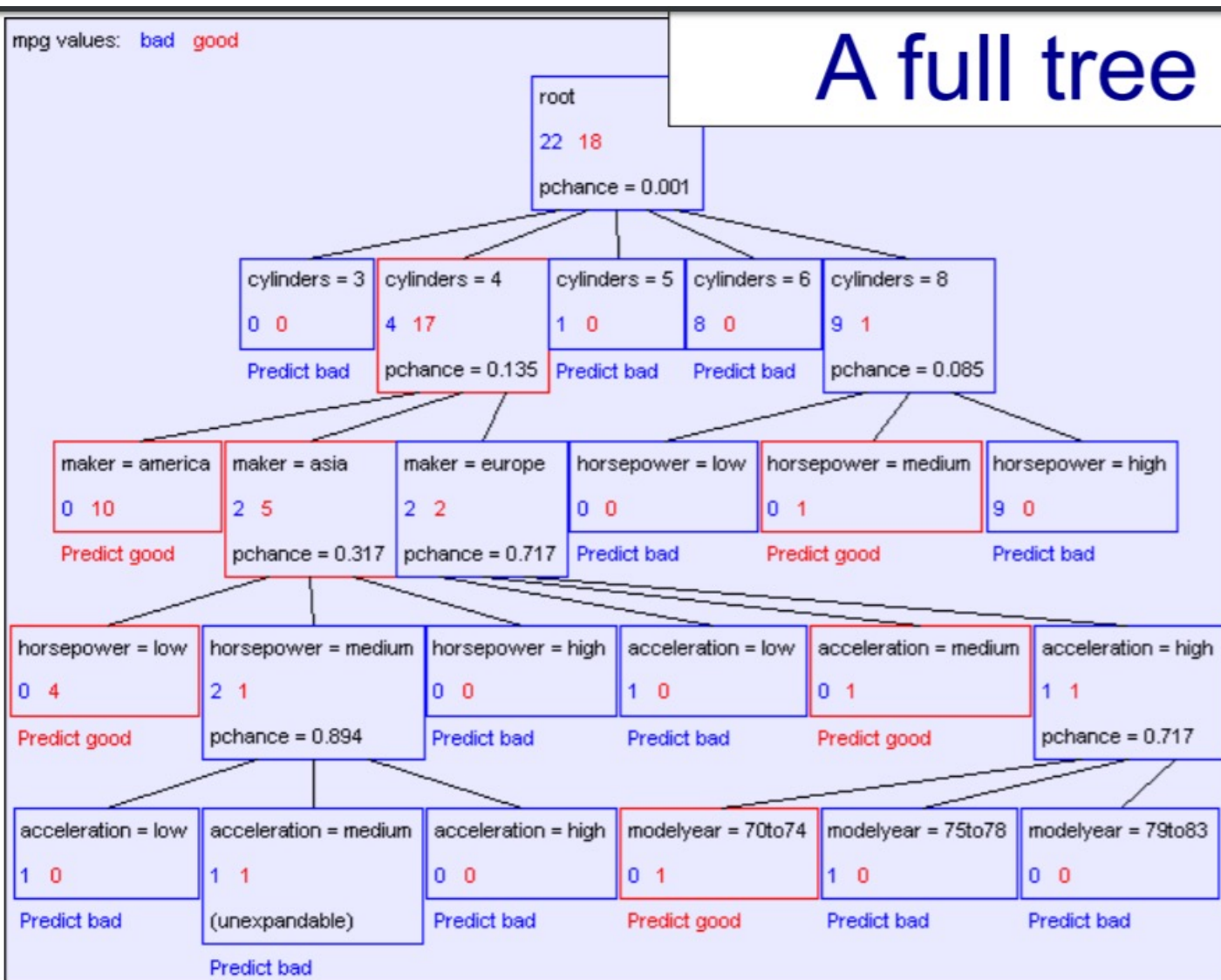
Overfitting

Solutions against Overfitting

- Standard decision trees have no learning bias
  - Training set error is always zero!
    - (If there is no label noise)
  - Lots of variance
  - Must introduce some bias towards simpler trees
- Many strategies for picking simpler trees
  - Fixed depth
  - Minimum number of samples per leaf
- Pruning
  - Remove branches of the tree that increase error using cross-validation