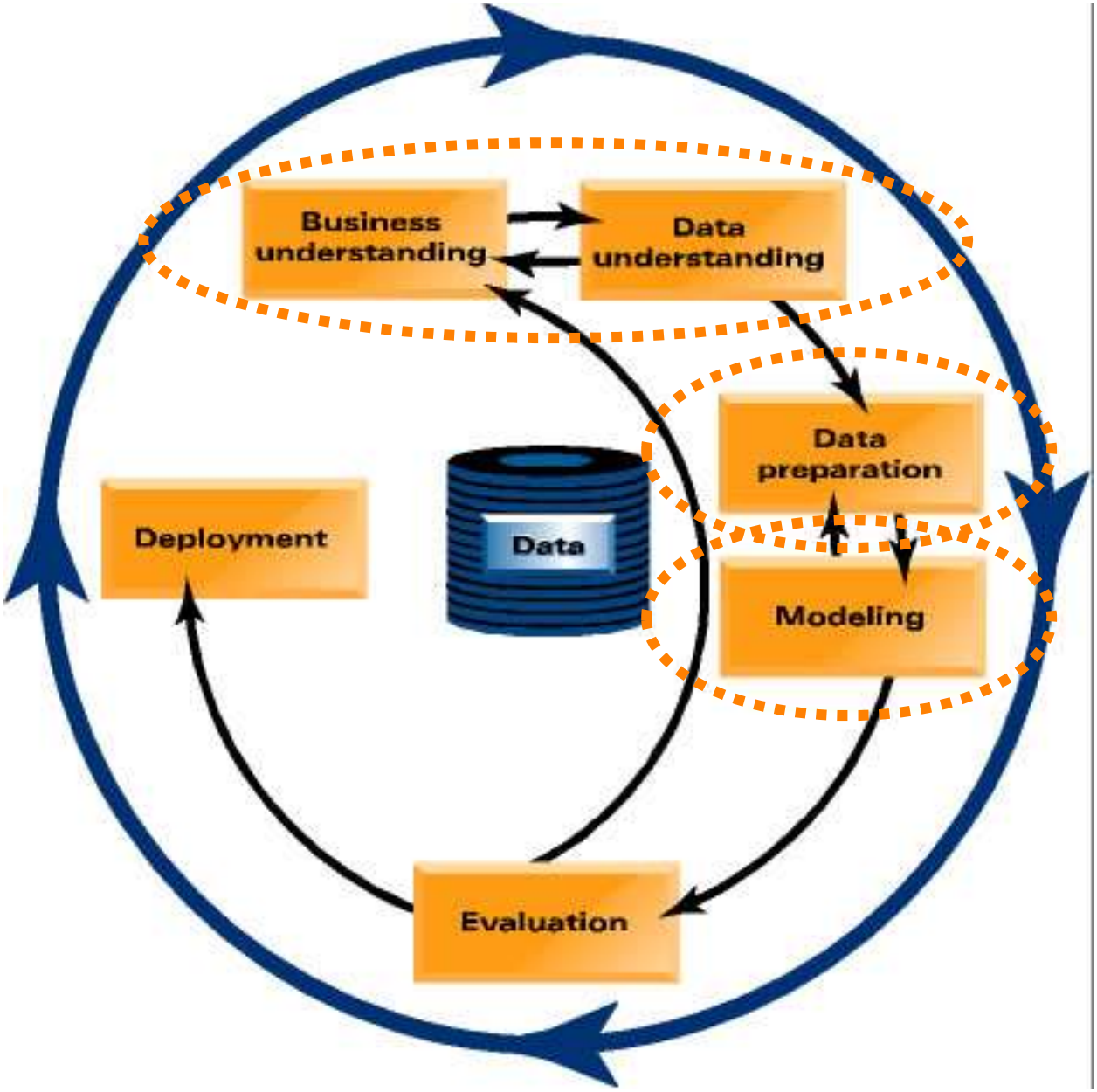
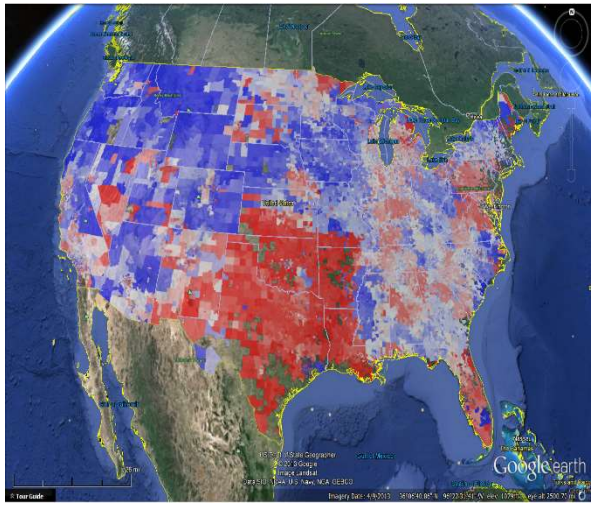
Predictive Modeling with Decision Trees: From Correlation to Supervised Segmentation

Source: Provost and Fawcett (2013), T. Mitchell (1997) and D. Jurafsky
Thanks to Maytal Saar-Tsechansky and Claudia Perlich.

Recall our cases so far…





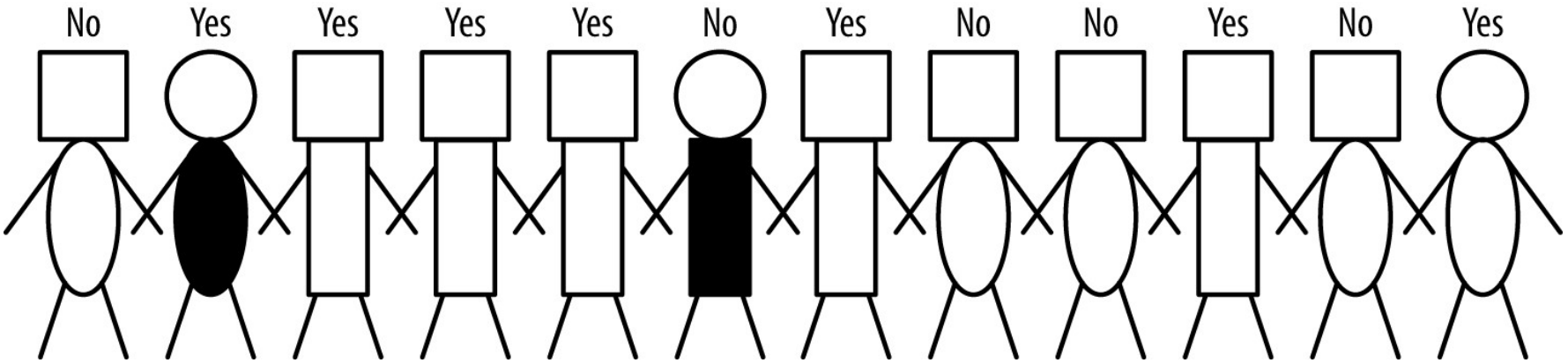
TARGET

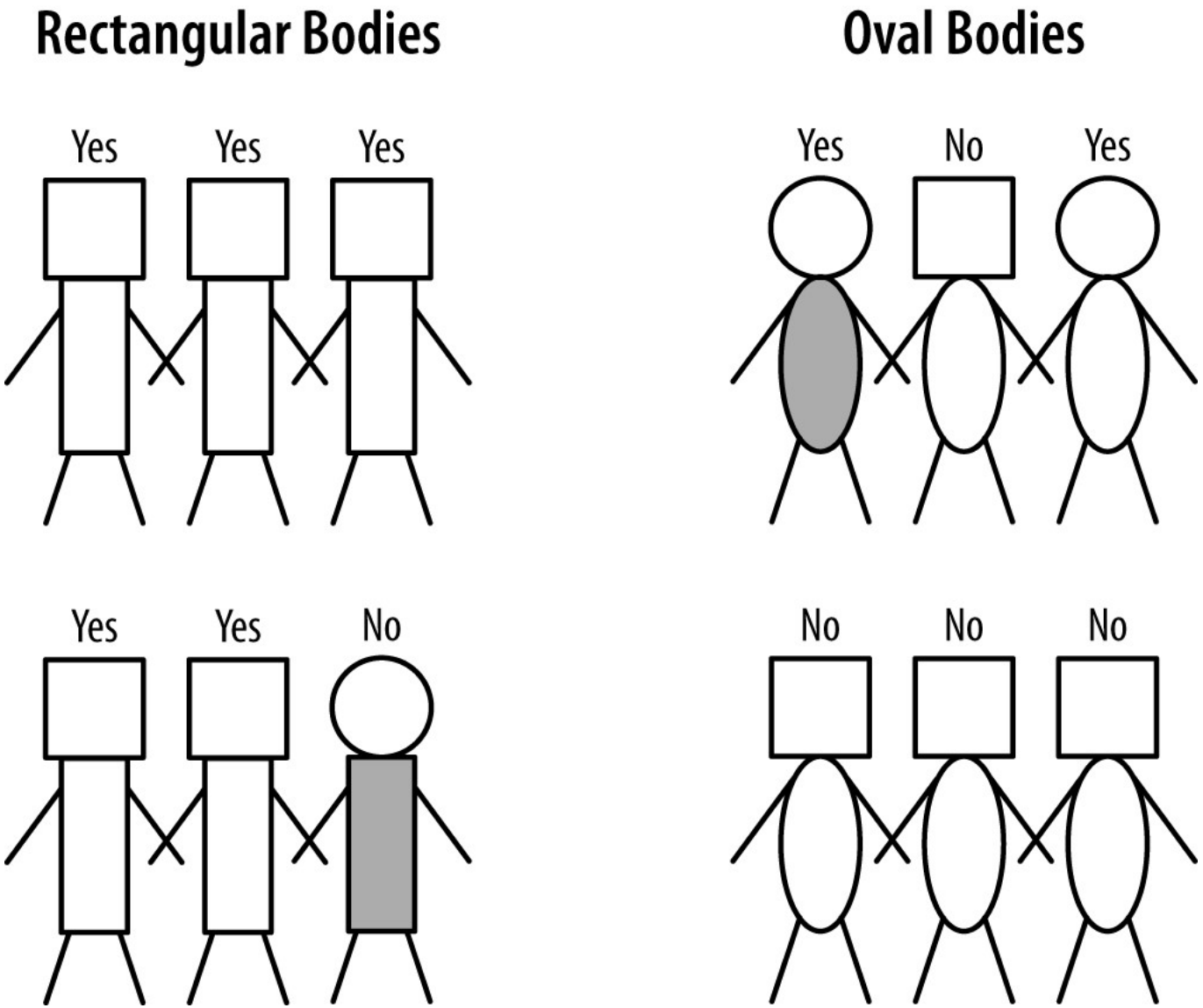
Fundamental concept in data science: selecting informative variables/attributes

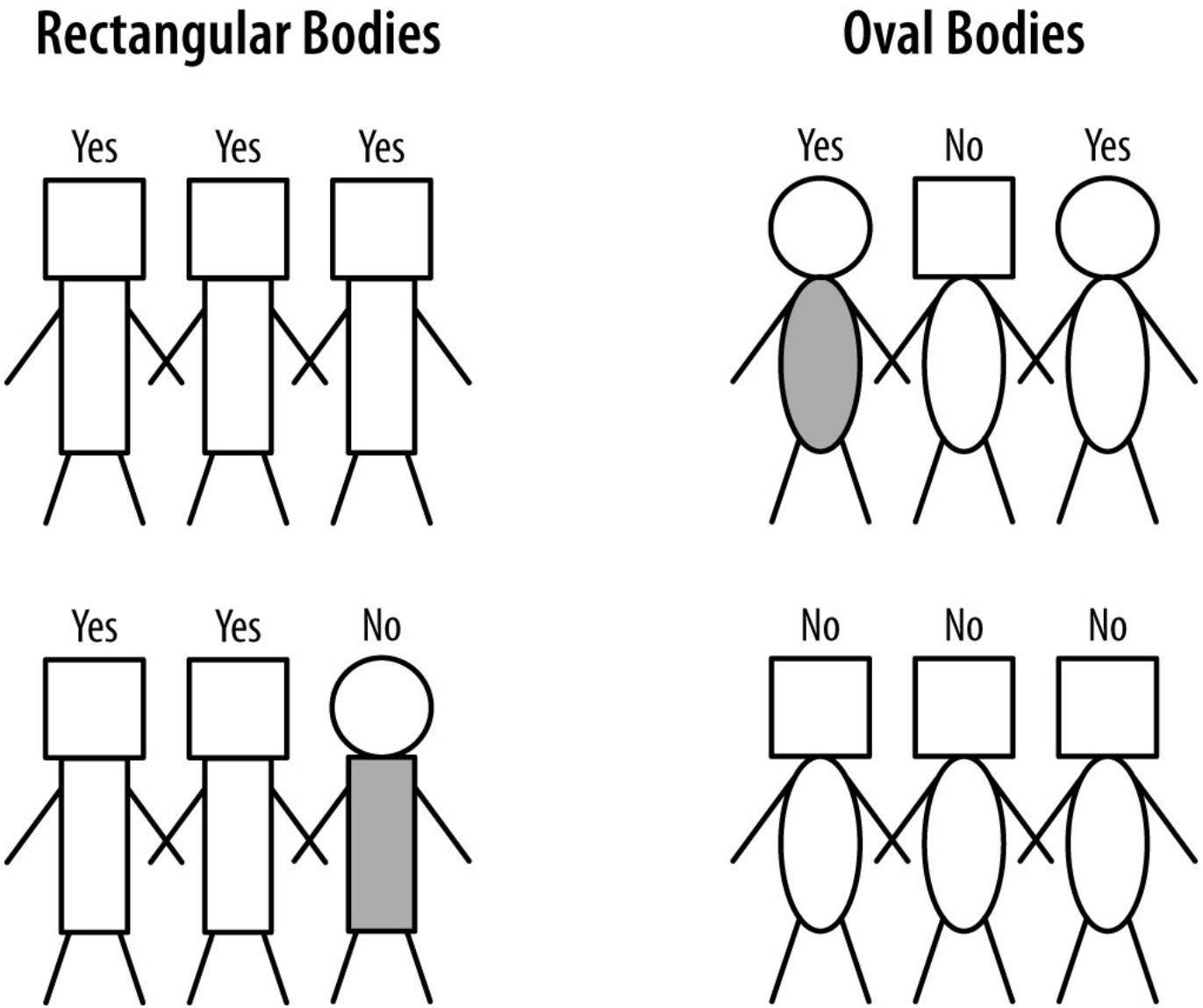
- How best to segment my customers, *with respect to differences in the target variable*

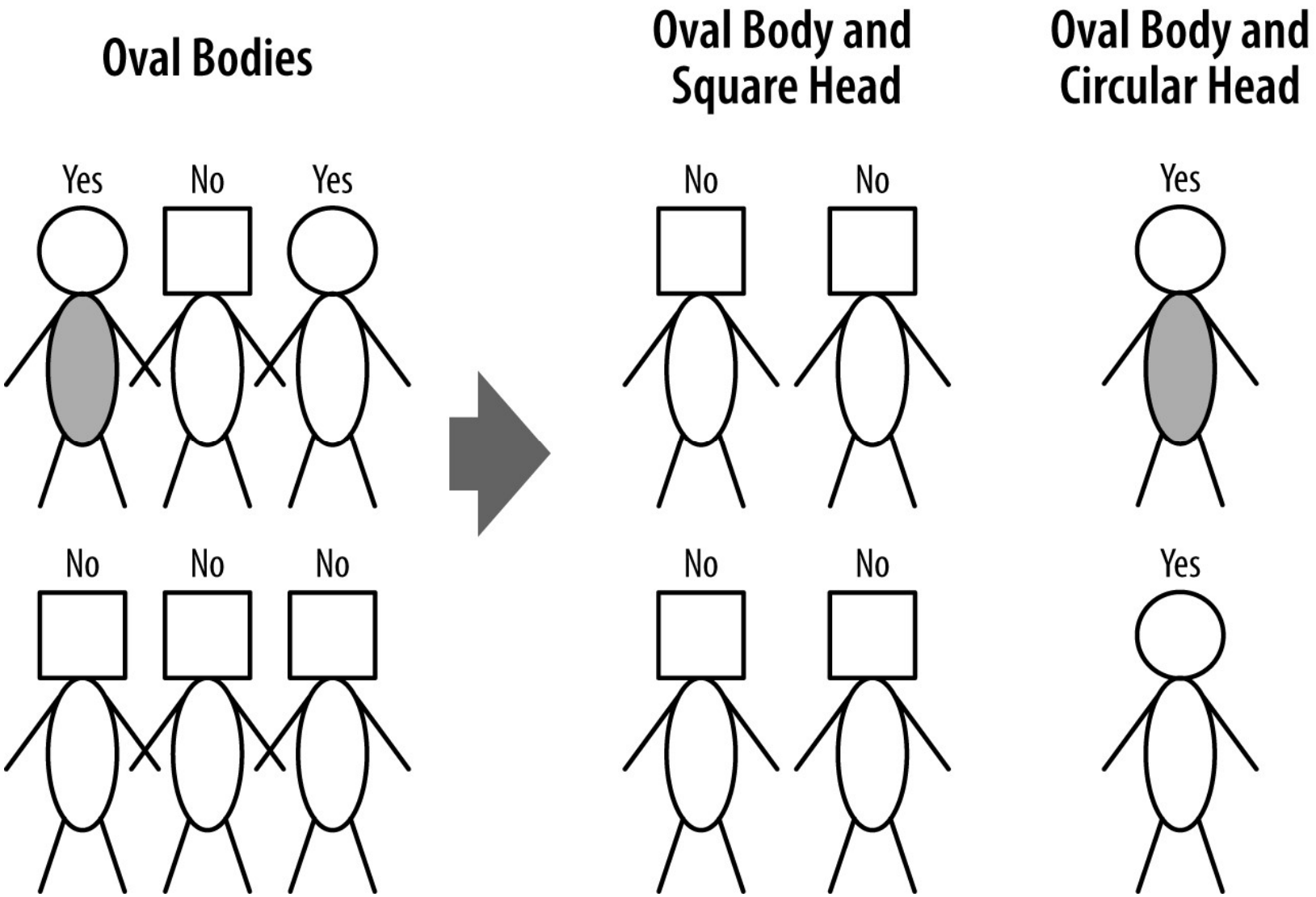
- The most basic predictive modeling/data mining technique
- Can be used as preprocess to many other data mining techniques
- The basis for one of the most popular, more-complex data mining techniques

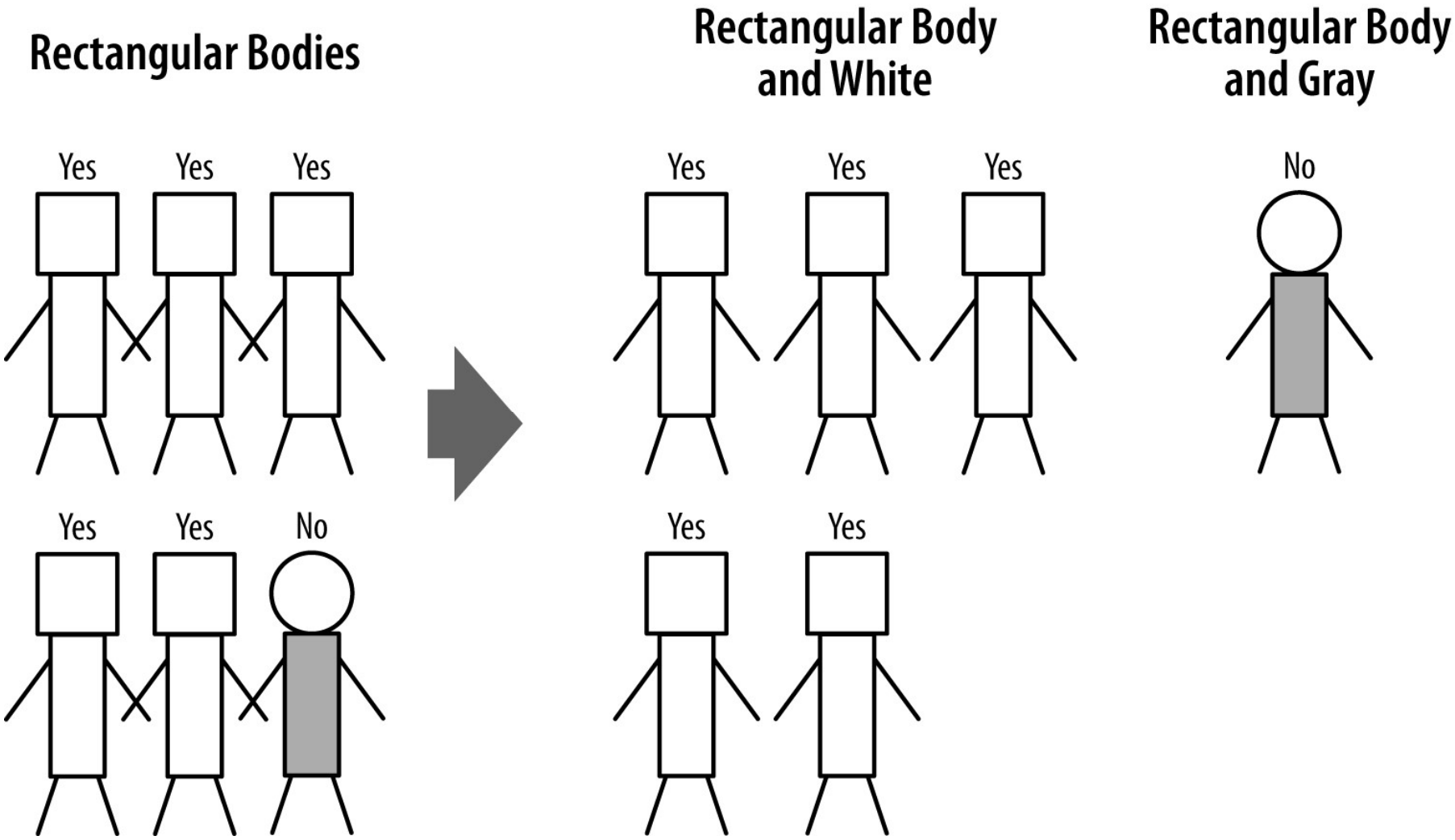
Selecting informative attributes

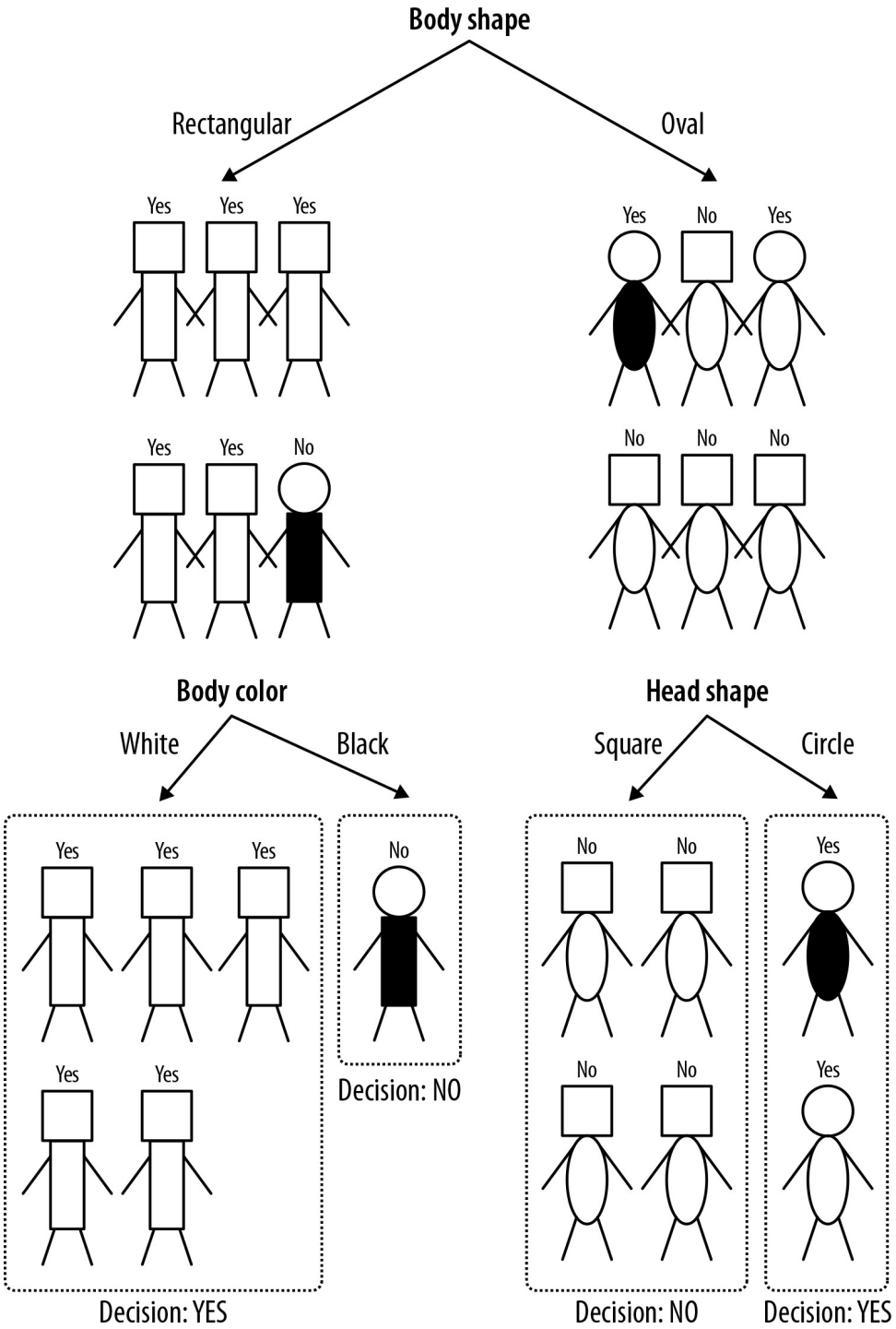



Rectangular Bodies
Yes
Yes
Yes
Yes
Yes
No
Oval Bodies
Yes
No
Yes
No
No
No

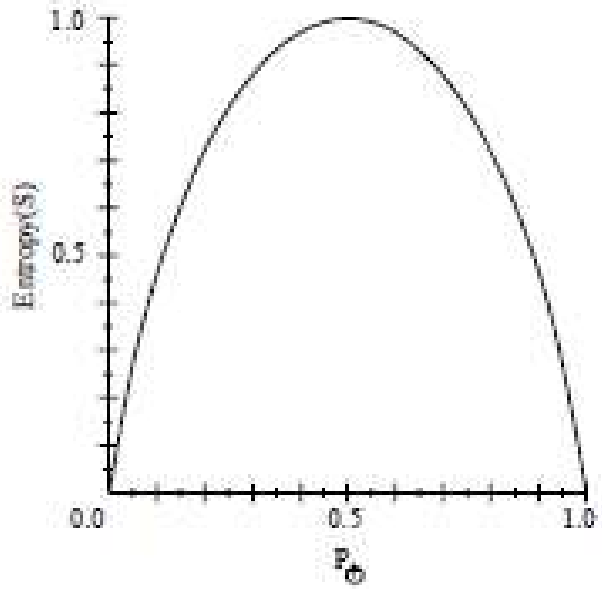

Rectangular Bodies
Yes
Yes
Yes
Yes
Yes
No
Oval Bodies
Yes
No
Yes
No
No
No


Oval Bodies
Oval Body and Square Head
Oval Body and Circular Head
Yes
No
Yes
No
No
No
No
No
No
Yes
No
No
Yes


Rectangular Bodies
Rectangular Body and White
Rectangular Body and Gray
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
No
Yes
Yes
No


Body shape
Rectangular
Oval
Yes Yes Yes
Yes No Yes
Yes Yes No
No No No
Body color
White
Black
Yes Yes Yes
Yes Yes
No
Decision: NO
Decision: YES
Head shape
Square
Circle
No No
No No
Yes
Yes
Decision: NO
Decision: YES

Entropy



- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

The late Claude Shannon of Bell Labs was the father of Information Theory, a theoretical basis for coding, compression, etc.

*Why "**Entropy**"? The story goes that Shannon didn't know what to call his new information measure, so he asked **von Neumann**, who said `You should call it **entropy** ... [since] ... no one knows what entropy really is, so in a debate you will always have the advantage' (Tribus 1971)*

Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of S:

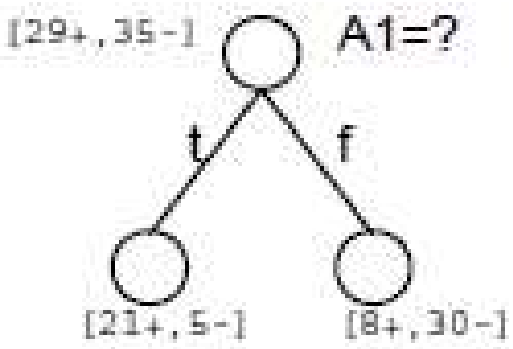
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

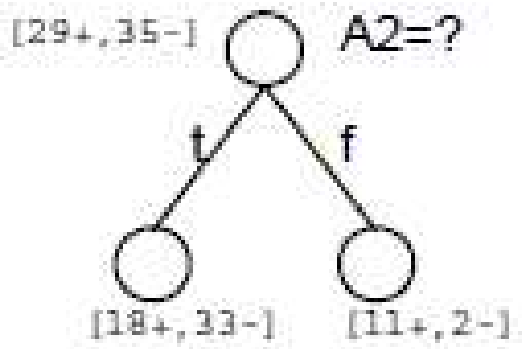
$$Entropy(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

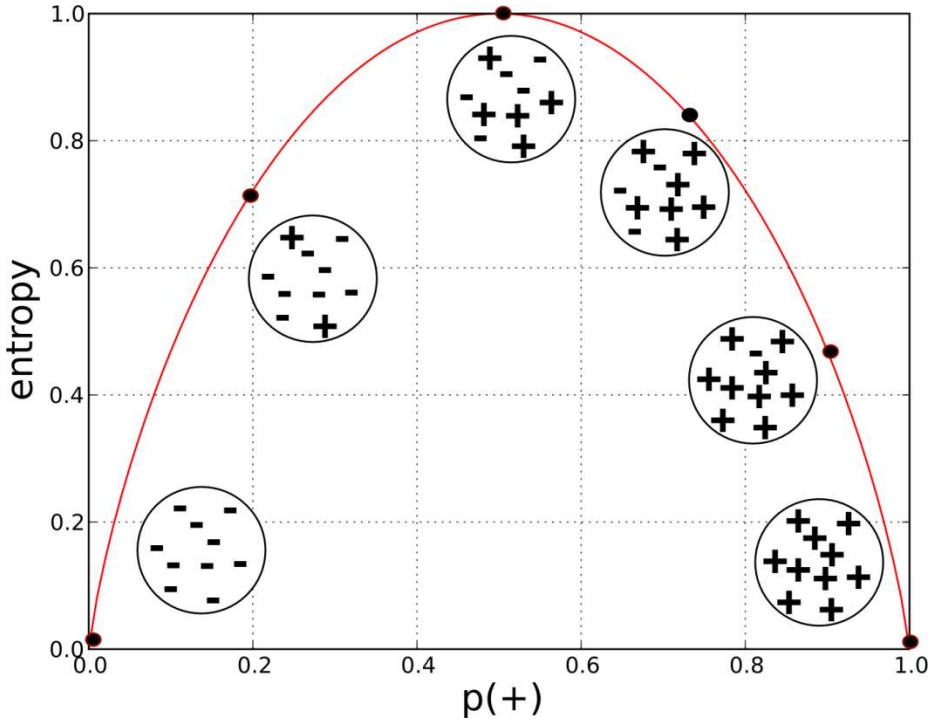
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$





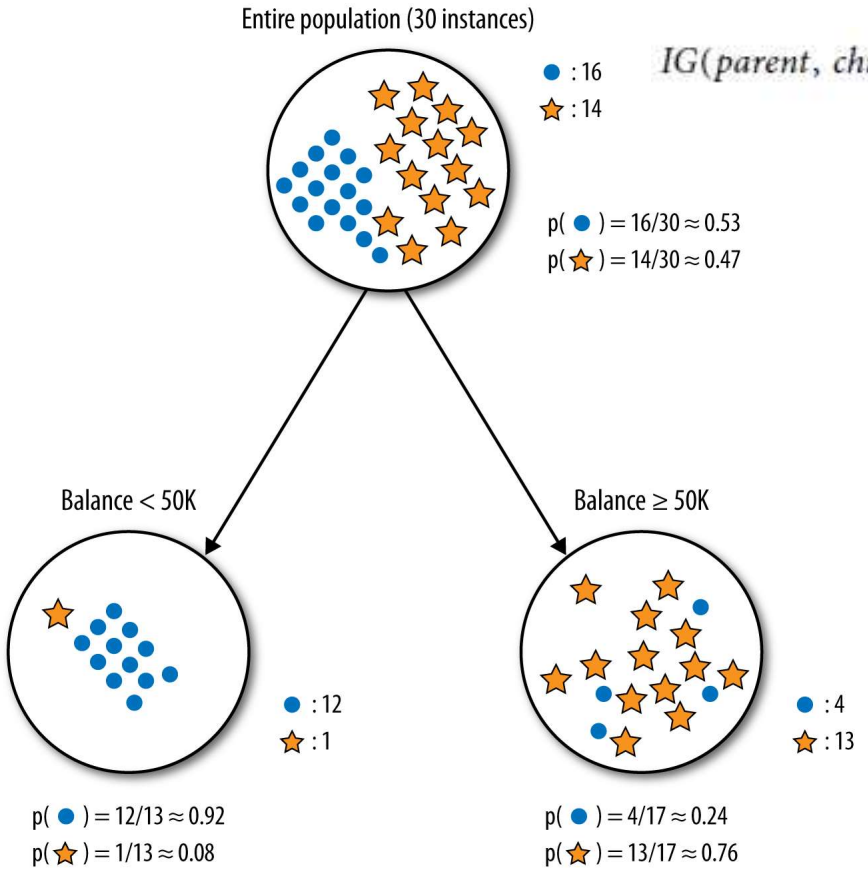
Selecting Informative Attributes

- The most common splitting criterion is called **information gain** (IG)
  - It is based on a **purity measure** called **entropy**
    - $entropy = -\mathrm{p}_1 \log_2(p_1) - p_2 \log_2(p_2) - ..$
    - Measures the general disorder of a set



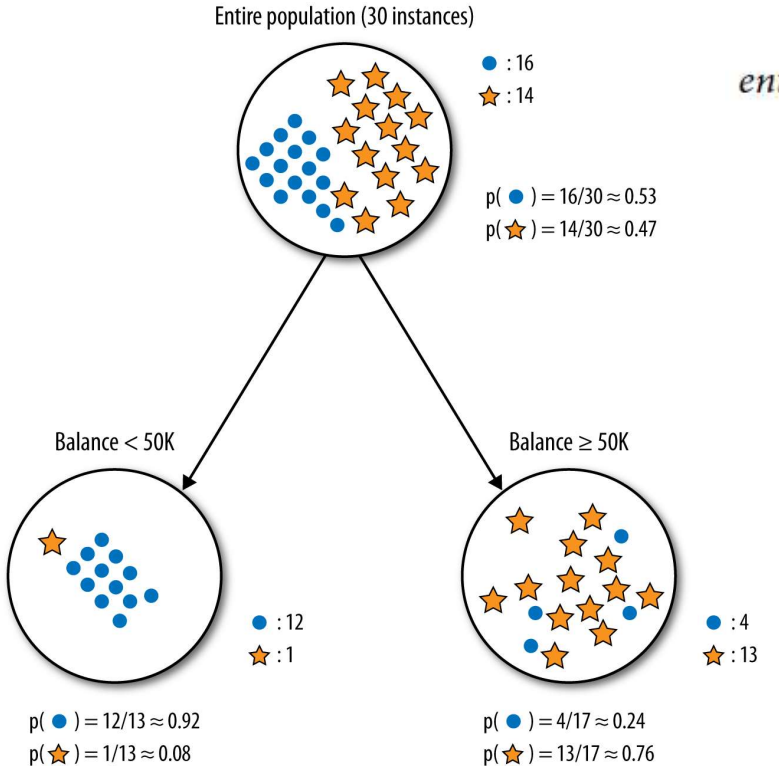
Information Gain

- Information gain measures the *change* in entropy due to any amount of new information being added



$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots]$$

Information Gain



$$entropy(parent) = -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)]$$
$$\approx -[0.53 \times -0.9 + 0.47 \times -1.1]$$
$$\approx 0.99 \quad \text{(very impure)}$$

The entropy of the *left* child is:

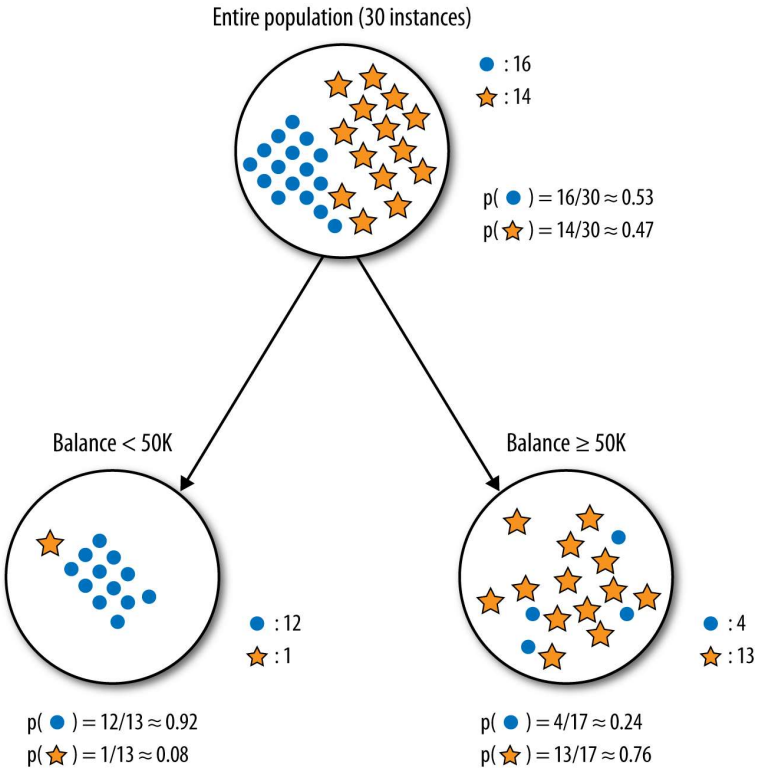
$$entropy(Balance < 50K) = -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)]$$
$$\approx -[0.92 \times (-0.12) + 0.08 \times (-3.7)]$$
$$\approx 0.40$$

The entropy of the *right* child is:

$$entropy(Balance \geq 50K) = -[p(\bullet) \times \log_2 p(\bullet) + p(\star) \times \log_2 p(\star)]$$
$$\approx -[0.24 \times (-2.1) + 0.76 \times (-0.39)]$$

0.79

Information Gain



$$
\begin{aligned}
IG &= entropy(parent) - [p(\text{Balance} < 50\text{K}) \times entropy(\text{Balance} < 50\text{K}) \\
&\qquad + p(\text{Balance} \geq 50\text{K}) \times entropy(\text{Balance} \geq 50\text{K})] \\
&\approx 0.99 - [0.43 \times 0.39 + 0.57 \times 0.79] \\
&\approx 0.37
\end{aligned}
$$

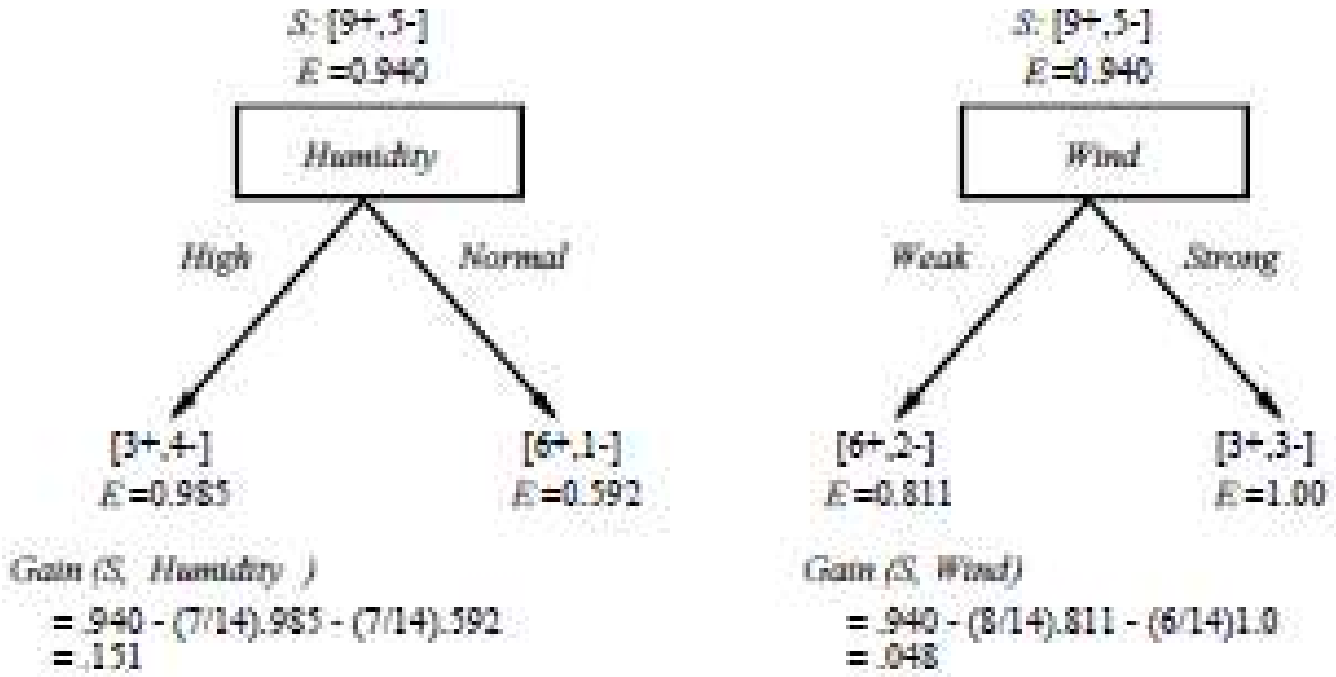
Attribute Selection

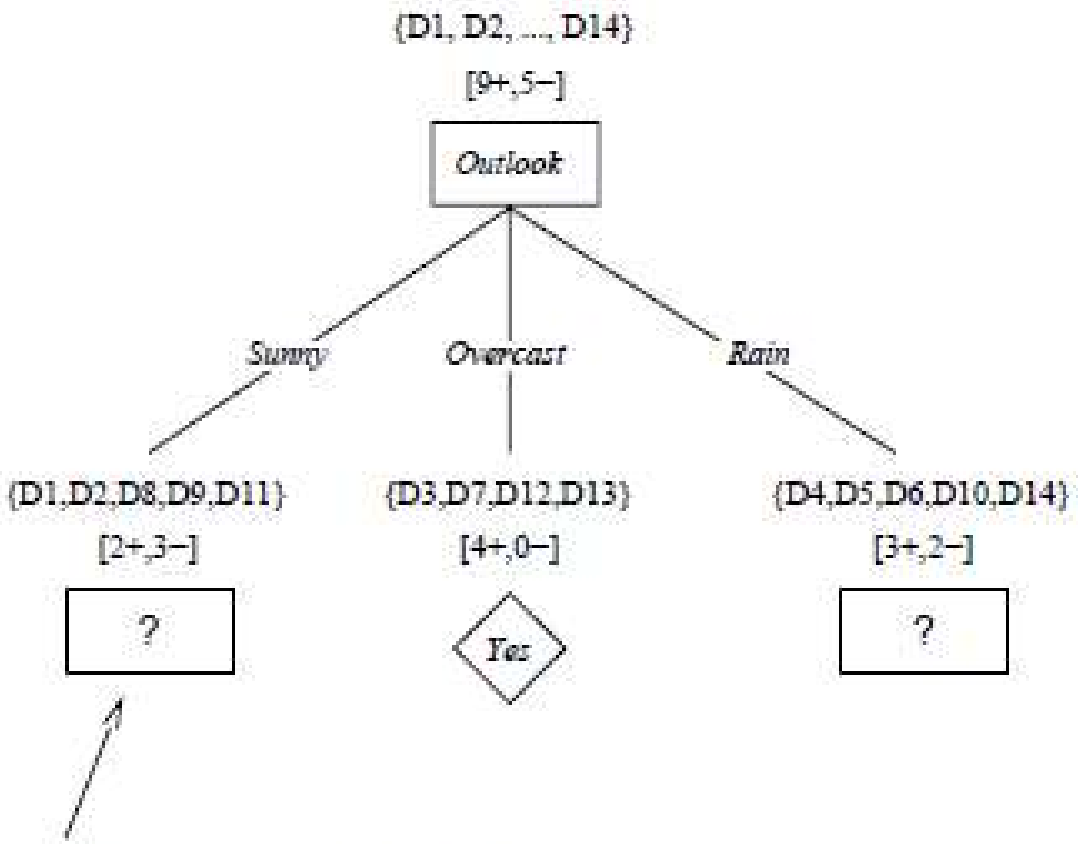
- Reasons for selecting only a subset of attributes:
- Better insights and business understanding
- Better explanations and more tractable models
- Reduced cost
- Faster predictions
- Better predictions!
    - Over-fitting (*to be continued*..)

and also determining the most informative attributes..

Selecting the Next Attribute

Which attribute is the best classifier?





S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny} , *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny} , *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny} , *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

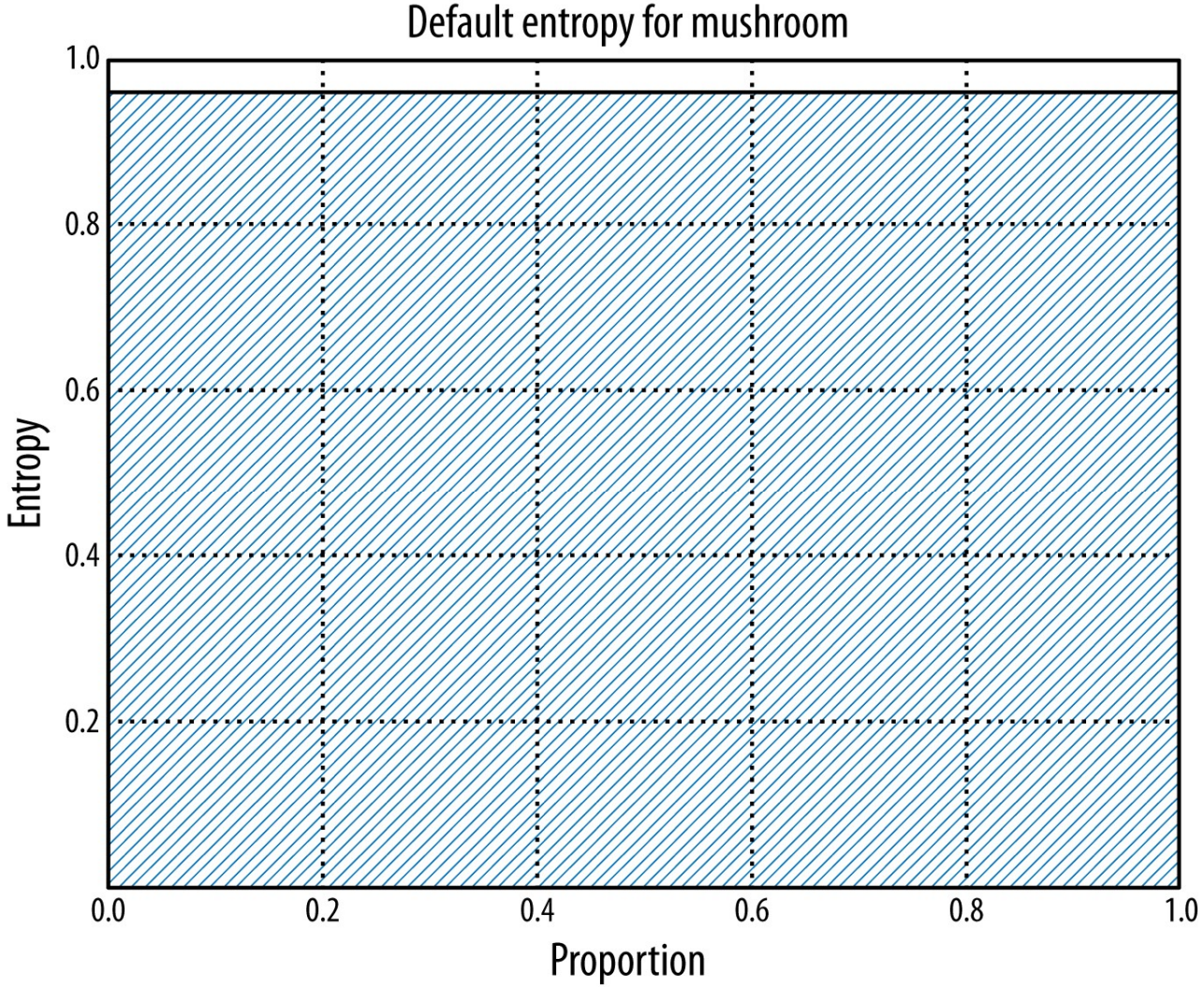
Example: Attribution Selection with Information Gain

- This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family
- Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended
    - This latter class was combined with the poisonous one
- The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like “leaflets three, let it be” for Poisonous Oak and Ivy

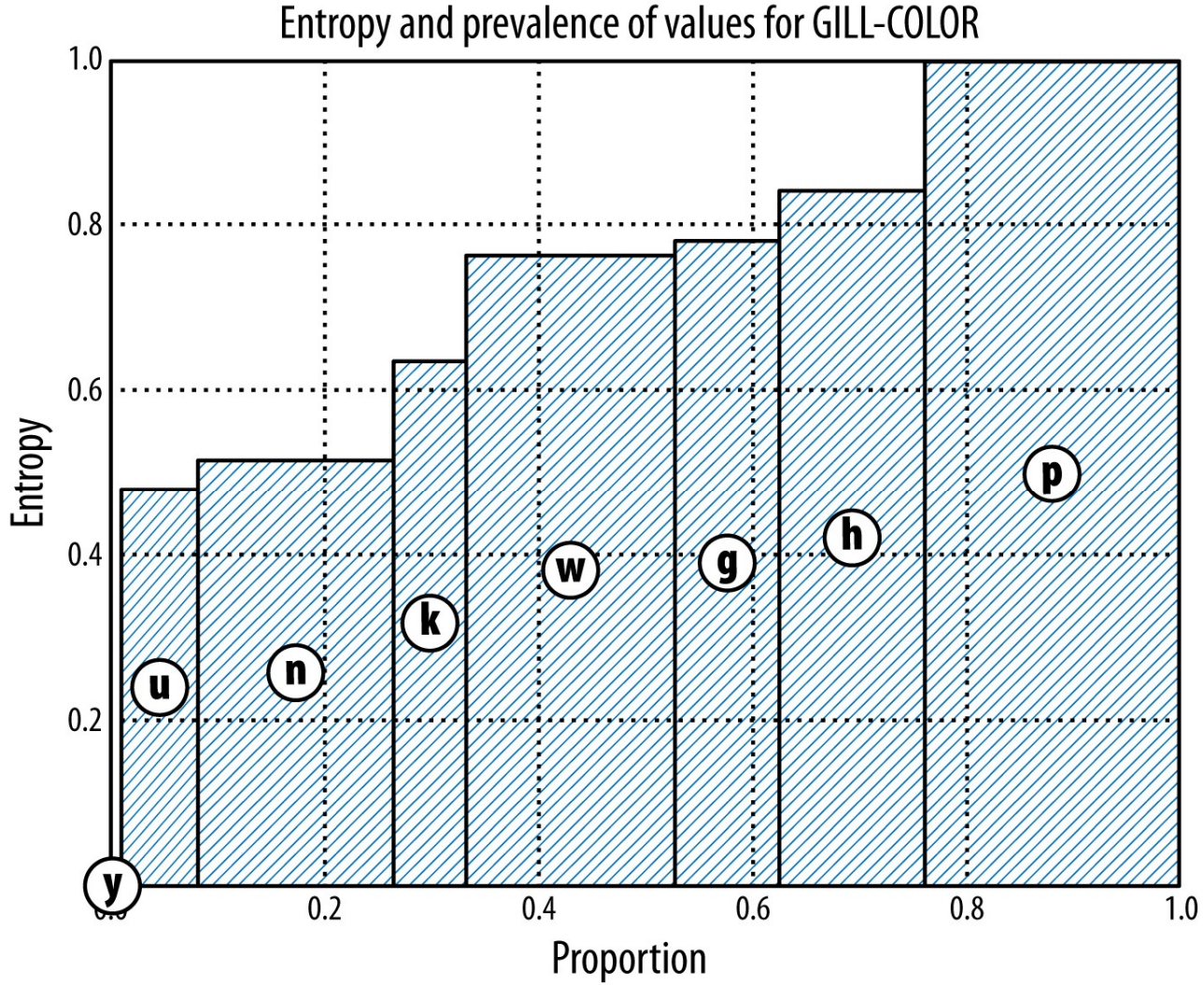
Example: Attribution Selection with Information Gain

Attribute name	Possible values
CAP-SHAPE	bell, conical, convex, flat, knobbed, sunken
CAP-SURFACE	fibrous, grooves, scaly, smooth
CAP-COLOR	brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
BRUISES?	yes, no
ODOR	almond, anise, creosote, fishy, foul, musty, none, pungent, spicy
GILL-ATTACHMENT	attached, descending, free, notched
GILL-SPACING	close, crowded, distant
GILL-SIZE	broad, narrow
GILL-COLOR	black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow
STALK-SHAPE	enlarging, tapering
STALK-ROOT	bulbous, club, cup, equal, rhizomorphs, rooted, missing
STALK-SURFACE-ABOVE-RING	fibrous, scaly, silky, smooth
STALK-SURFACE-BELOW-RING	fibrous, scaly, silky, smooth

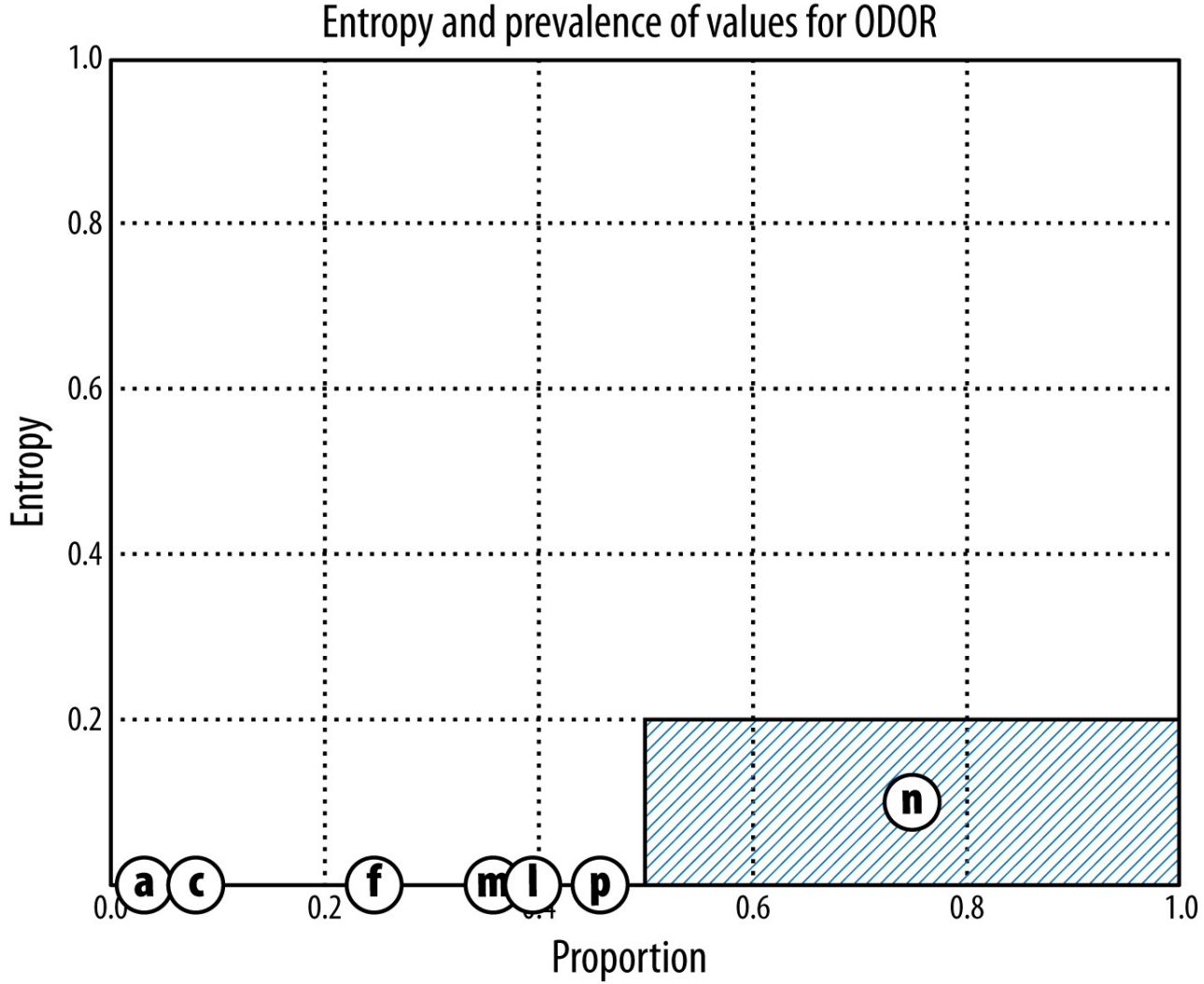
Example: Attribution Selection with Information Gain



Example: Attribution Selection with Information Gain



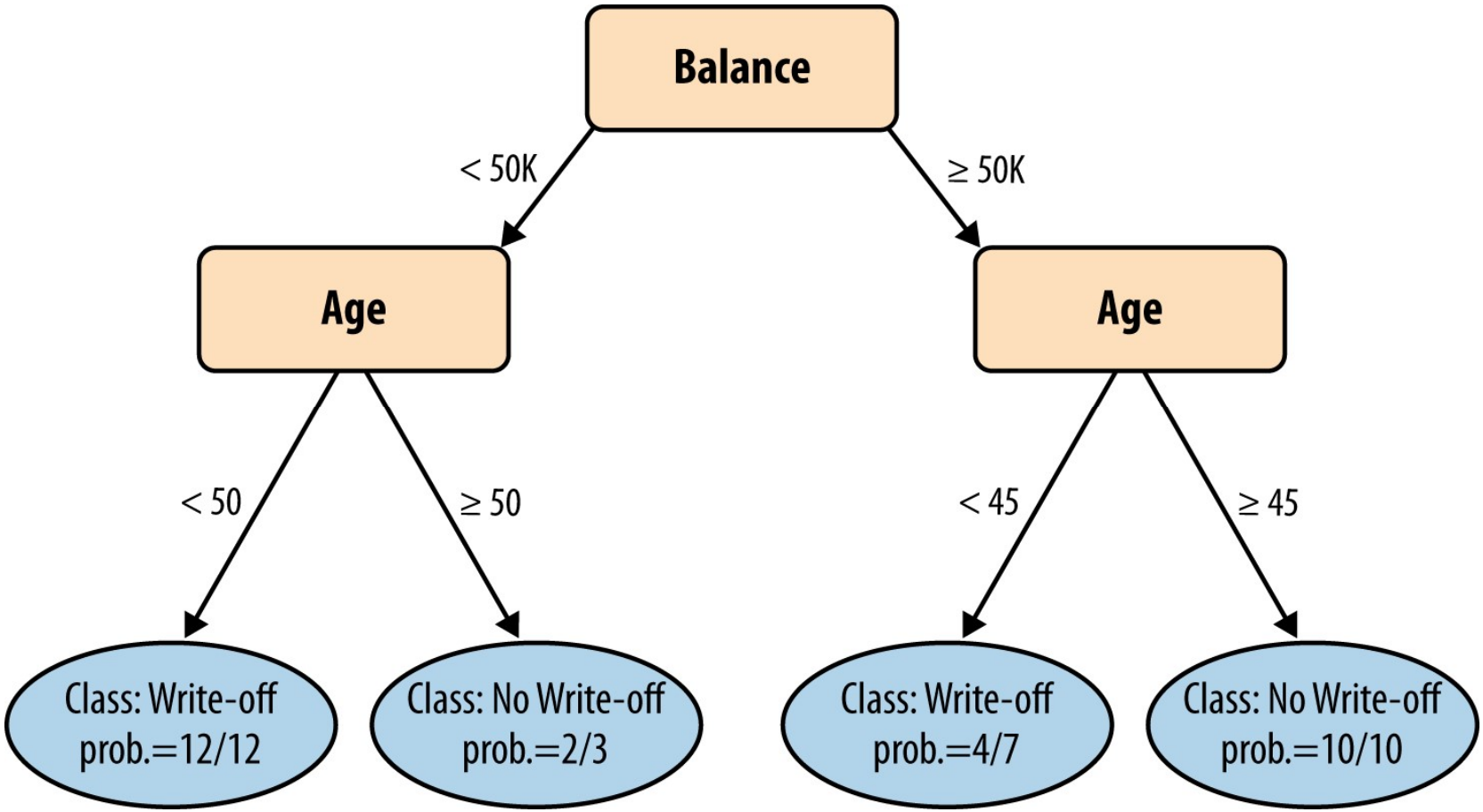
Example: Attribution Selection with Information Gain



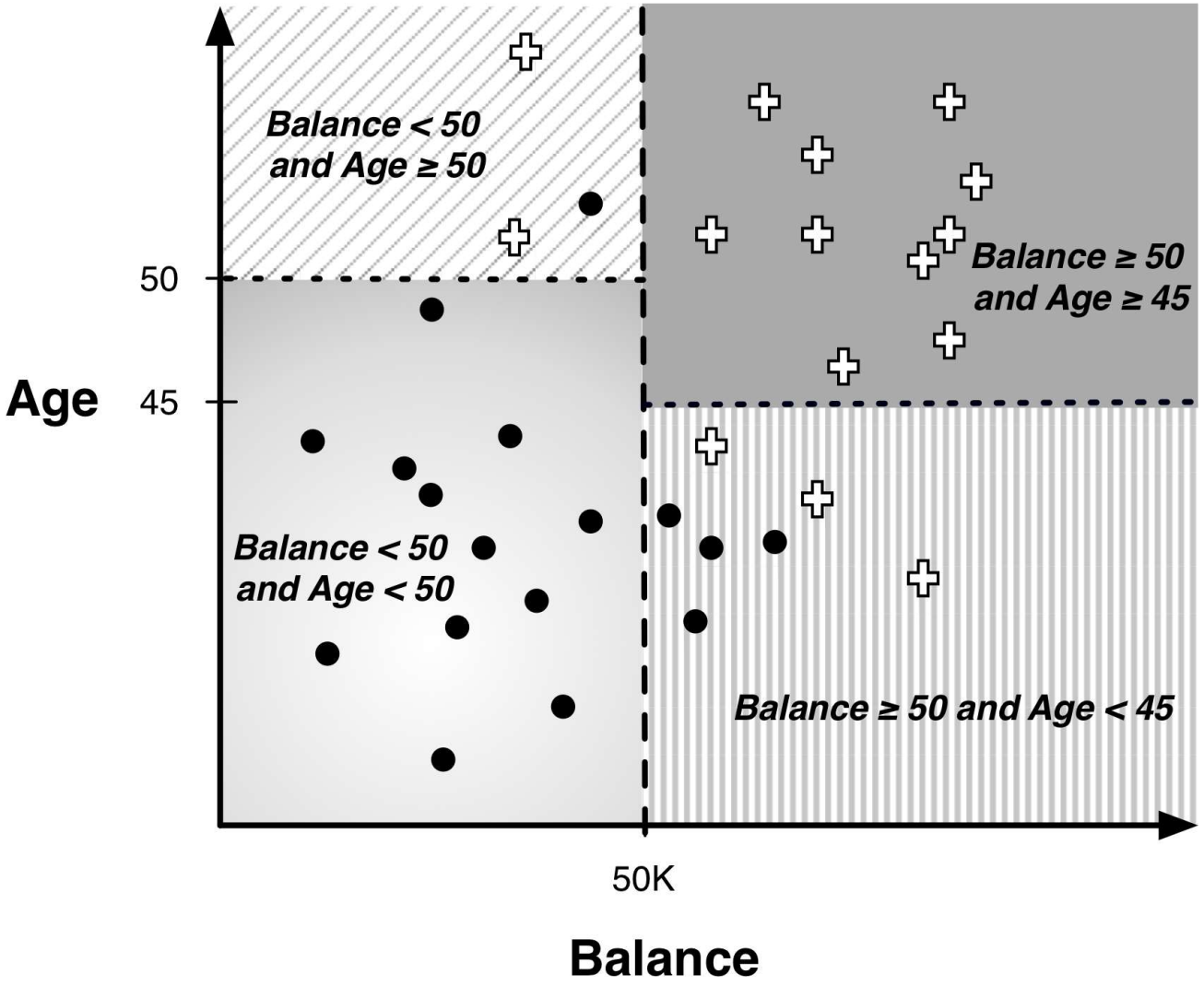
Multivariate Supervised Segmentation

- If we select the *single* variable that gives the most information gain, we create a very *simple* segmentation
- If we select multiple attributes each giving some information gain, how do we put them together?

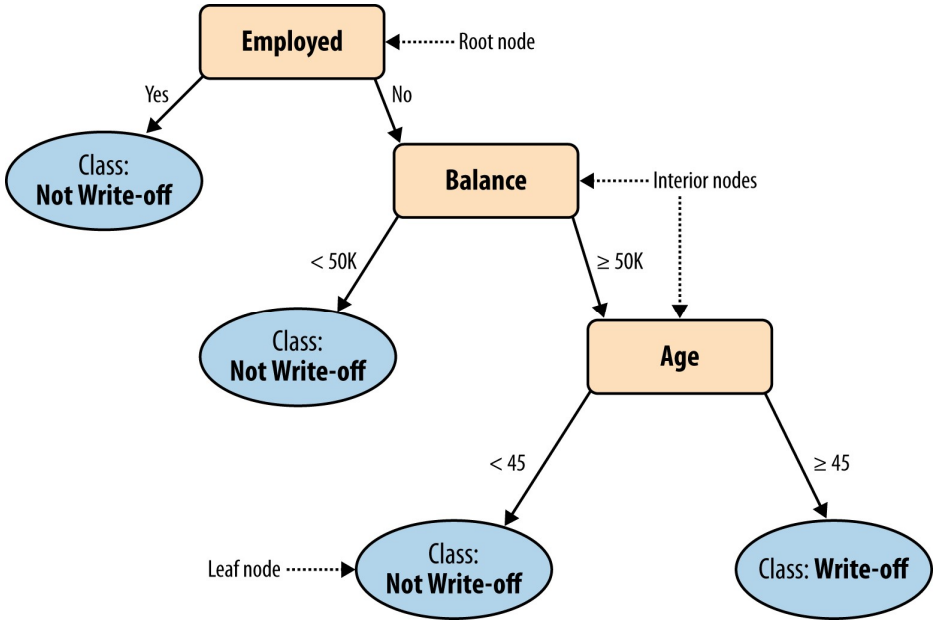
Visualizing Segmentations



Visualizing Segmentations

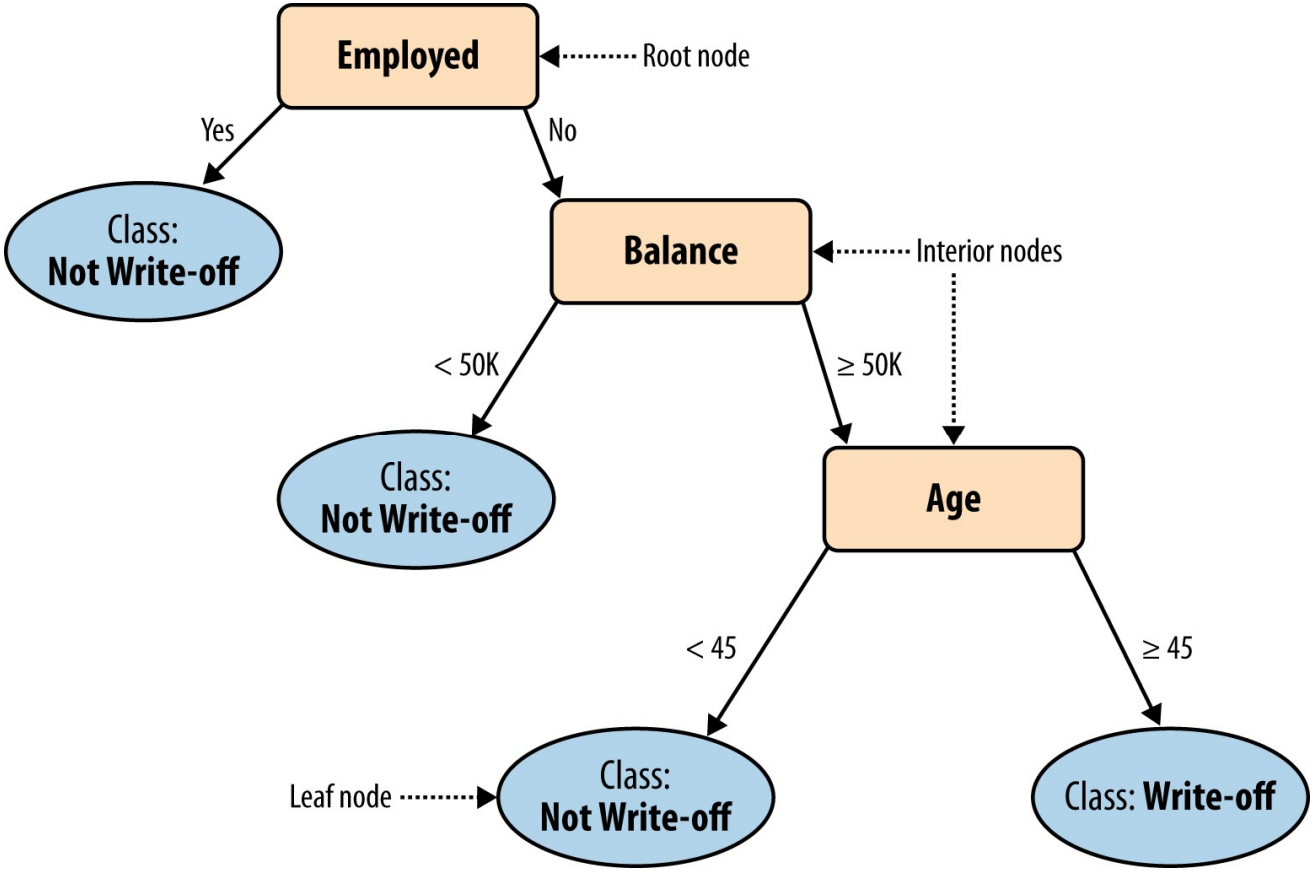


Tree-Structured Models



Tree-Structured Models

- Classify ‘John Doe’
  - Balance=115K, Employed=No, and Age=40



Why trees?

- Decision trees (DTs), or classification trees, are one of the most popular data mining tools
    - (along with linear and logistic regression)
- They are:
    - Easy to understand
    - Easy to implement
    - Easy to use
    - Computationally cheap
- Almost all data mining packages include DTs
- They have advantages for model comprehensibility, which is important for:
    - model evaluation
    - communication to non-DM-savvy stakeholders

Tree-Structured Models: “Rules”

- No two parents share descendants
- There are no cycles
- The branches always “point downwards”
- Every example always ends up at a leaf node with some specific class determination
    - Probability estimation trees, regression trees (*to be continued*..)

Tree Induction

- How do we create a classification tree from data?
  - **divide-and-conquer** approach
  - take each data subset and ***recursively*** apply attribute selection to find the best attribute to partition it
- When do we stop?
  - The nodes are pure,
  - there are no more variables, or
  - even earlier (over-fitting – *to be continued*..)

Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge$, $\vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

When to Consider Decision Trees

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

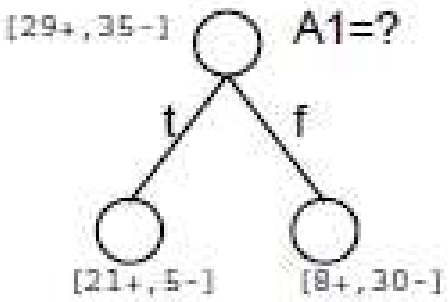
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

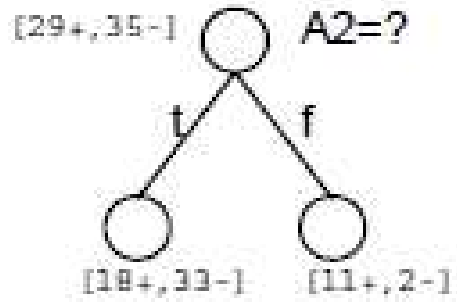
Top-Down Induction of Decision Trees

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?

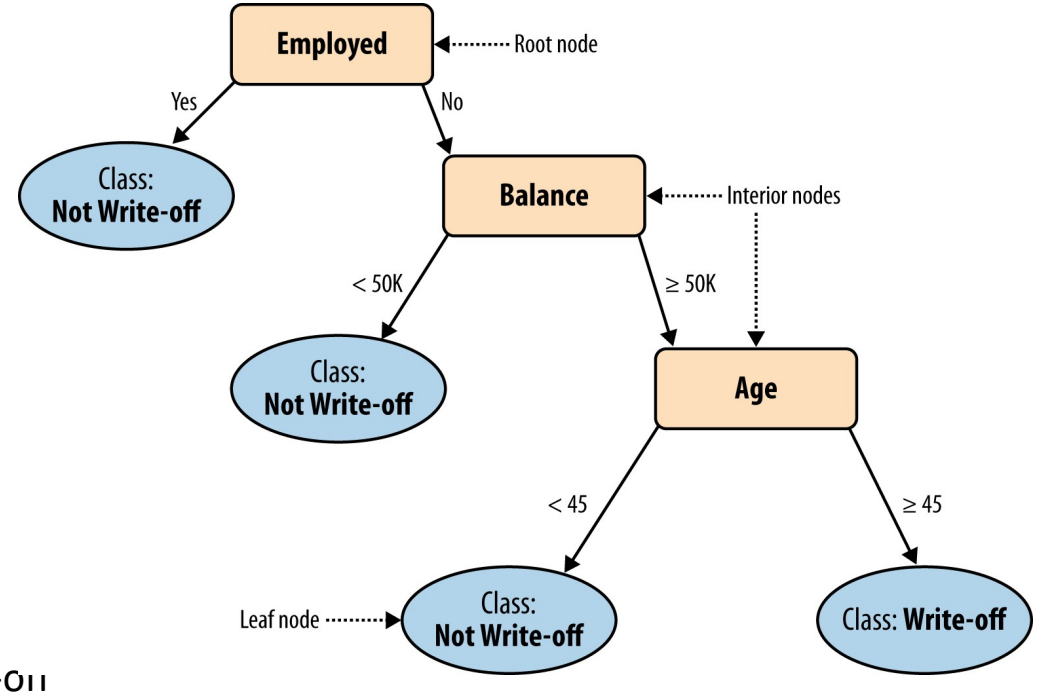




Trees as Sets of Rules

- The classification tree is equivalent to this rule set
- Each rule consists of the attribute tests along the path connected with **AND**

Trees as Sets of Rules



- IF (Employed = Yes) THEN Class=No Write-off
- IF (Employed = No) AND (Balance < 50k) THEN Class=No Write-off
- IF (Employed = No) AND (Balance ≥ 50k) AND (Age < 45) THEN Class=No Write-off
- IF (Employed = No) AND (Balance ≥ 50k) AND (Age ≥ 45) THEN Class=Write-off

MegaTelCo: Predicting Customer Churn

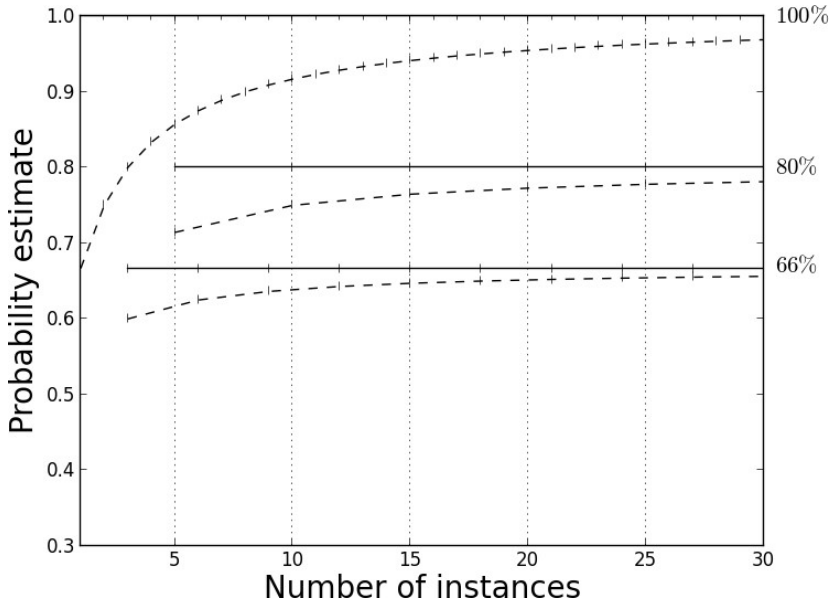
- Why would MegaTelCo want an estimate of the **probability** that a customer will leave the company within 90 days of contract expiration rather than simply predicting whether a person will leave the company within that time?
    - You might want to rank customers their probability of leaving

From Classification Trees to Probability Estimation Trees

- **Frequency-based estimate**
  - **Basic assumption**: Each member of a segment corresponding to a tree leaf has the same probability to belong in the corresponding class
  - If a leaf contains n positive instances and m negative instances (binary classification), the probability of any new instance being positive may be estimated as $\frac{n}{n+m}$
- Prone to **over-fitting**..

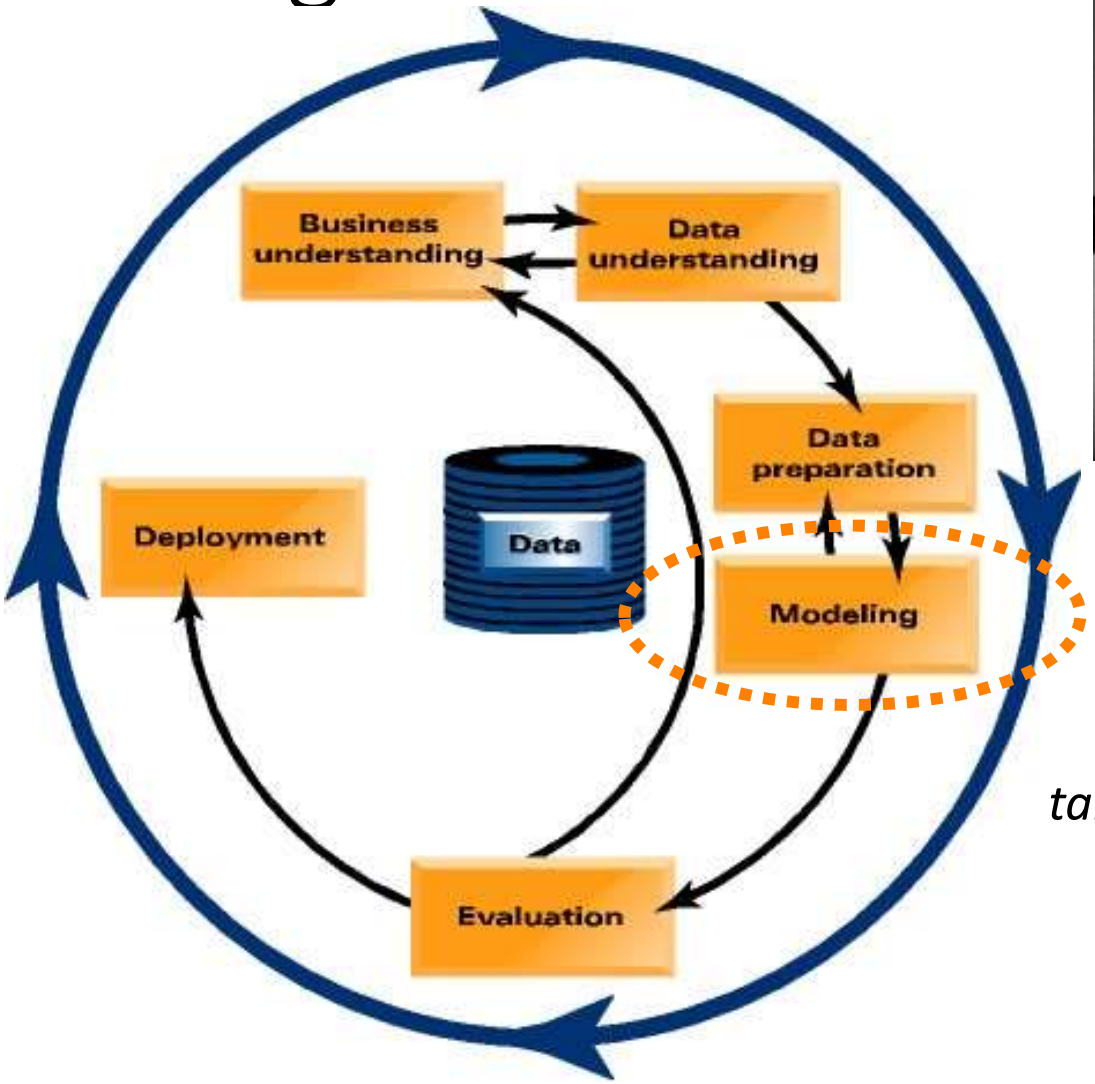
Laplace Correction: very few records in several leaves of a decision tree

- $p(c) = \frac{n+1}{n+m+2}$,
  - where n is the number of examples in the leaf belonging to class c, and m is the number of examples not belonging to class c



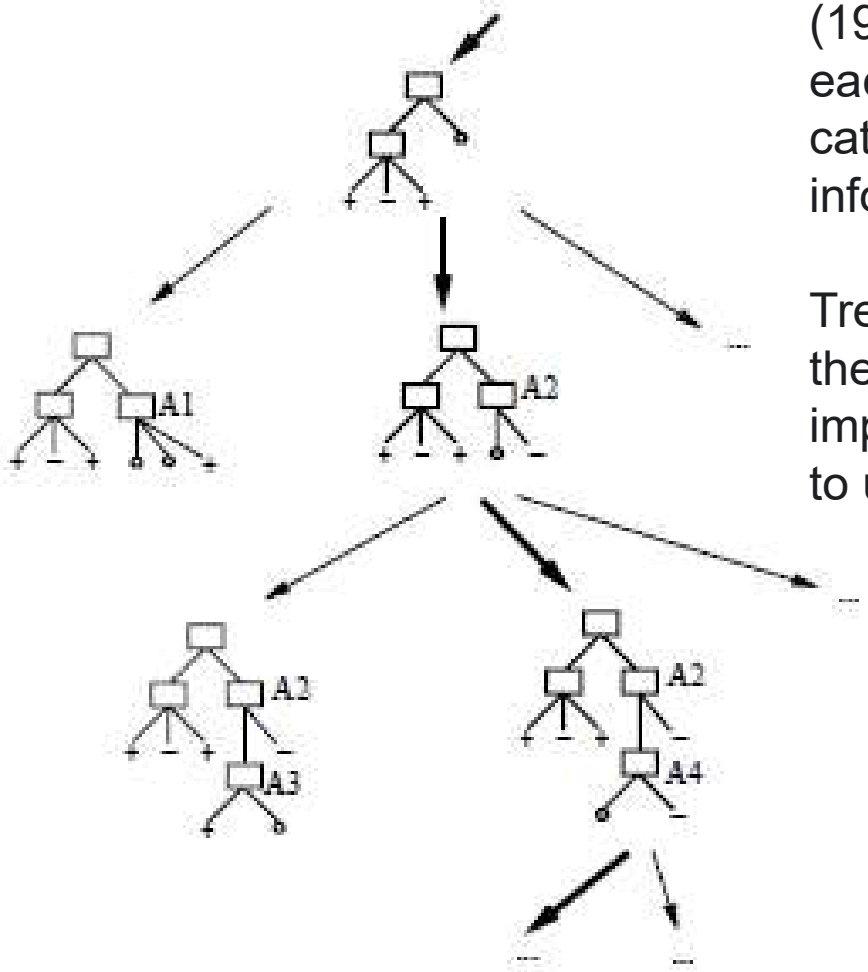
- As a result of this adjustment, when a leave has very few records the possibility that a particular class is true is reduced.
- Example: leaf with 2 positive records and 0 negative records, p = 0.75
- leaf with 20 positive records and 0 negative records, p = 0.95

Let's focus back in on actually mining the data..



Which customers should TelCo target with a special offer, prior to contract expiration?

Hypothesis Space Search by ID3



ID3 (Iterative Dichotomiser 3) Quinlan (1986): creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets.

Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalize to unseen data.

Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statisically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

Inductive Bias in ID3

Note H is the power set of instances X

$\rightarrow$ Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space H
- Occam's razor: prefer the shortest hypothesis that fits the data

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hyps. than long hyps.

→ a short hyp that fits data unlikely to be coincidence

→ a long hyp that fits data might be coincidence

Argument opposed:

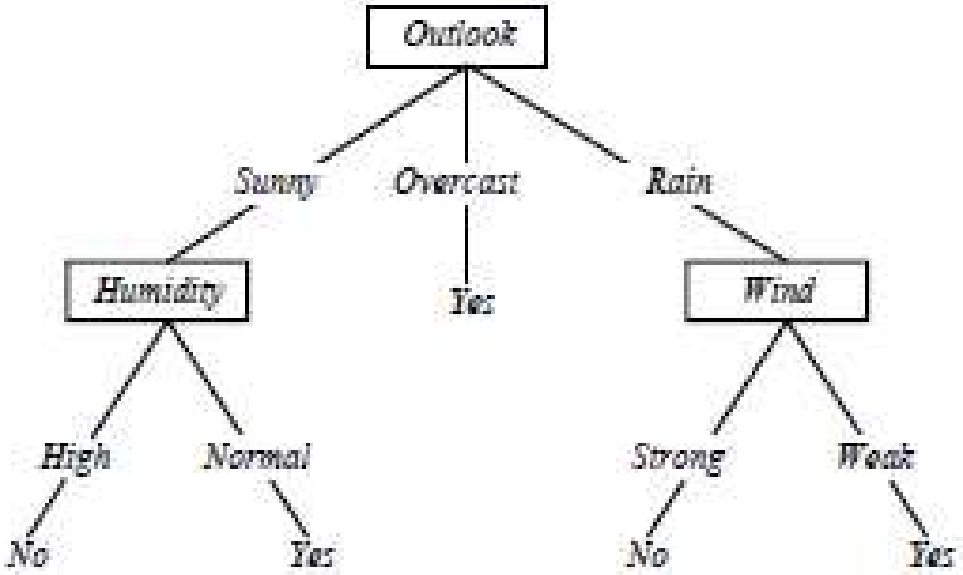
- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on *size* of hypothesis??

Overfitting in Decision Trees

Consider adding noisy training example #15:

$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$

What effect on earlier tree?



Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

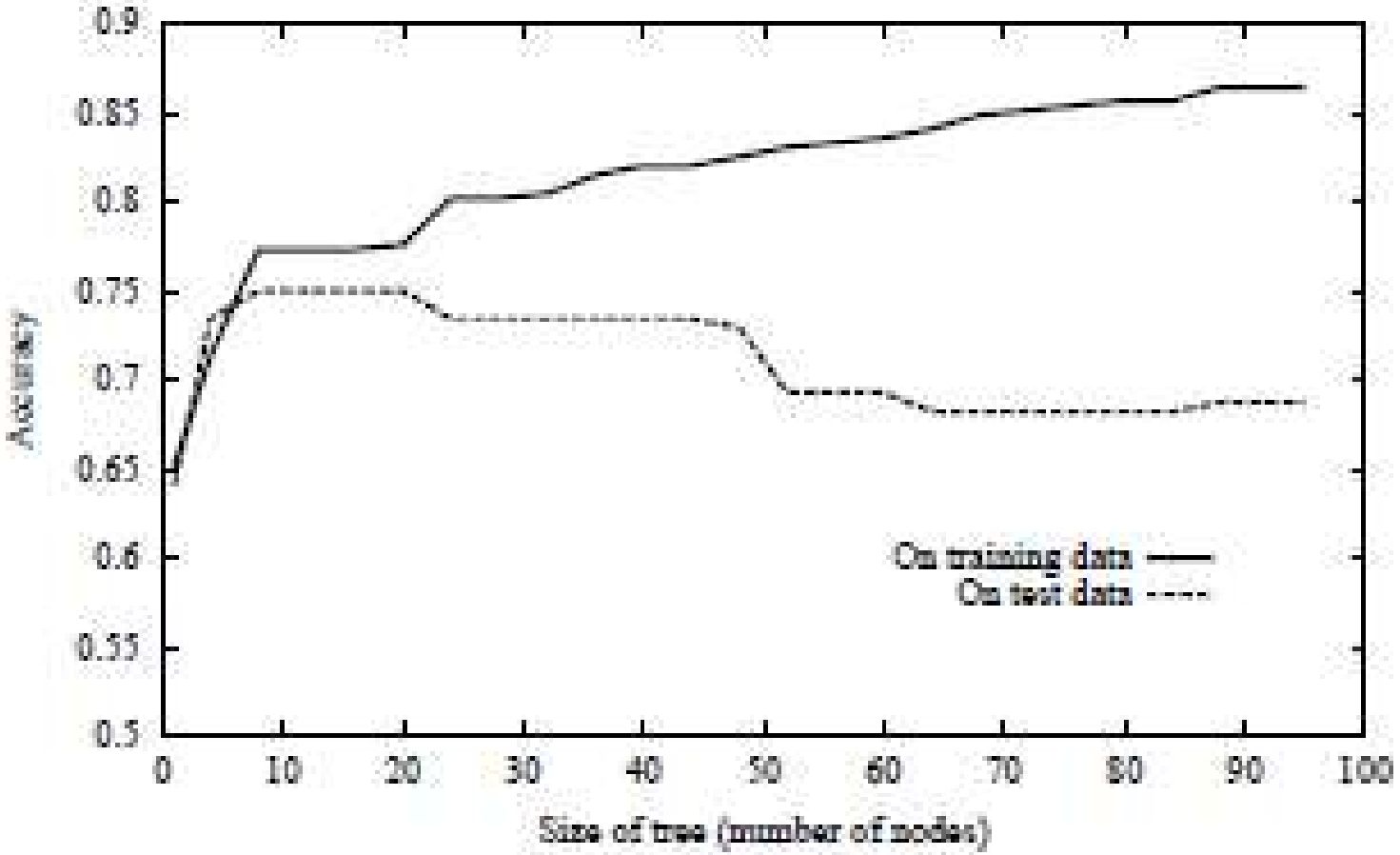
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize
  $size(tree) + size(misclassifications(tree))$

MDL: minimum description length

C4.5 algorithm

Improvement of ID3 correcting the following issues:

- Pruning trees after creation
- Handles continuous and discrete attributes
- Handles attributes with differing costs
- Handles training data with missing values

We explore these issues in the following slides.

C5.0: Quinlan's latest version release under a proprietary license.

- Uses less memory and builds smaller rules than C4.5 while being more accurate.

Pruning

- Pruning simplifies a decision tree to prevent over-fitting to noise in the data
- **Post-pruning**:
    - takes a fully-grown decision tree and discards unreliable parts
- **Pre-pruning**:
    - stops growing a branch when information becomes unreliable
- Post-pruning preferred in practice

Reduced-Error Pruning

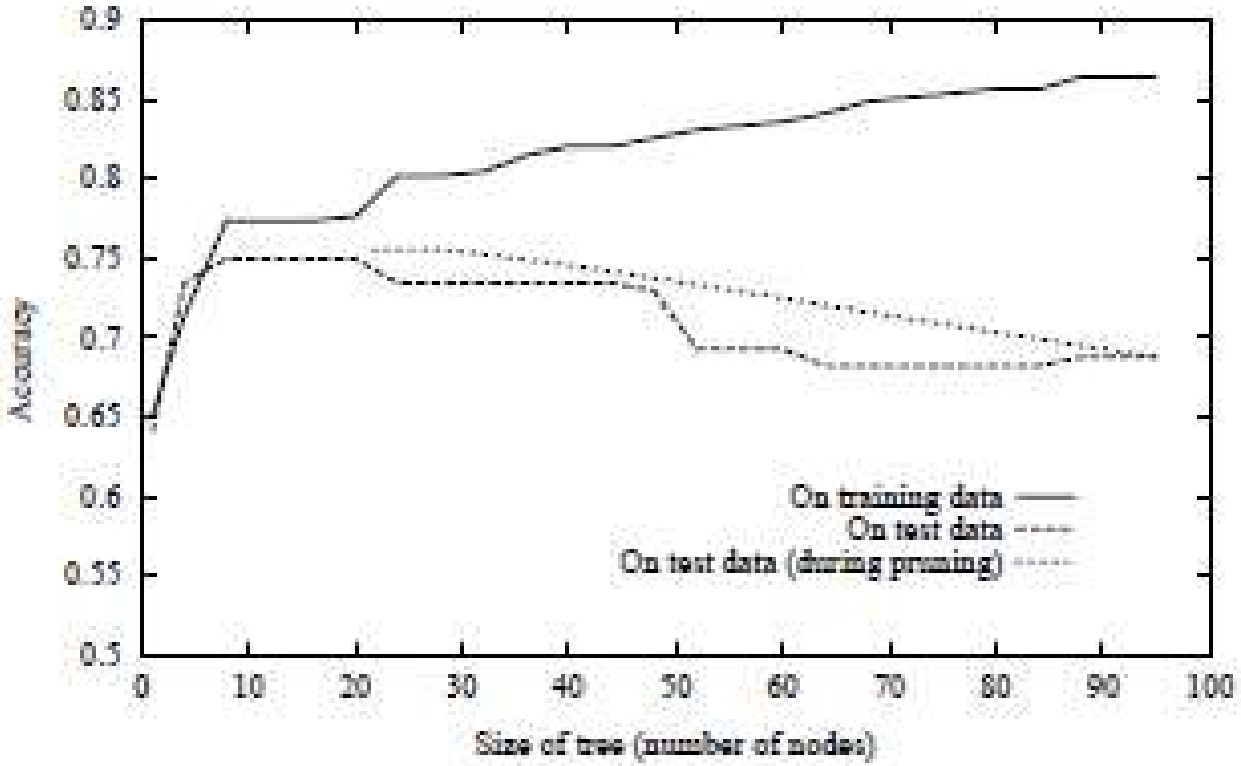
Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

Effect of Reduced-Error Pruning

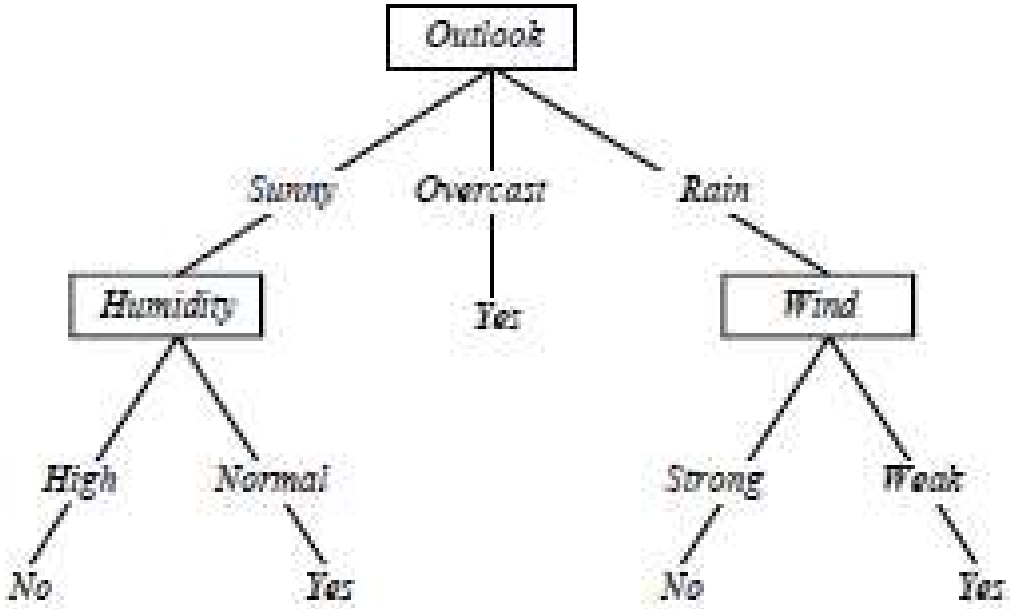


Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

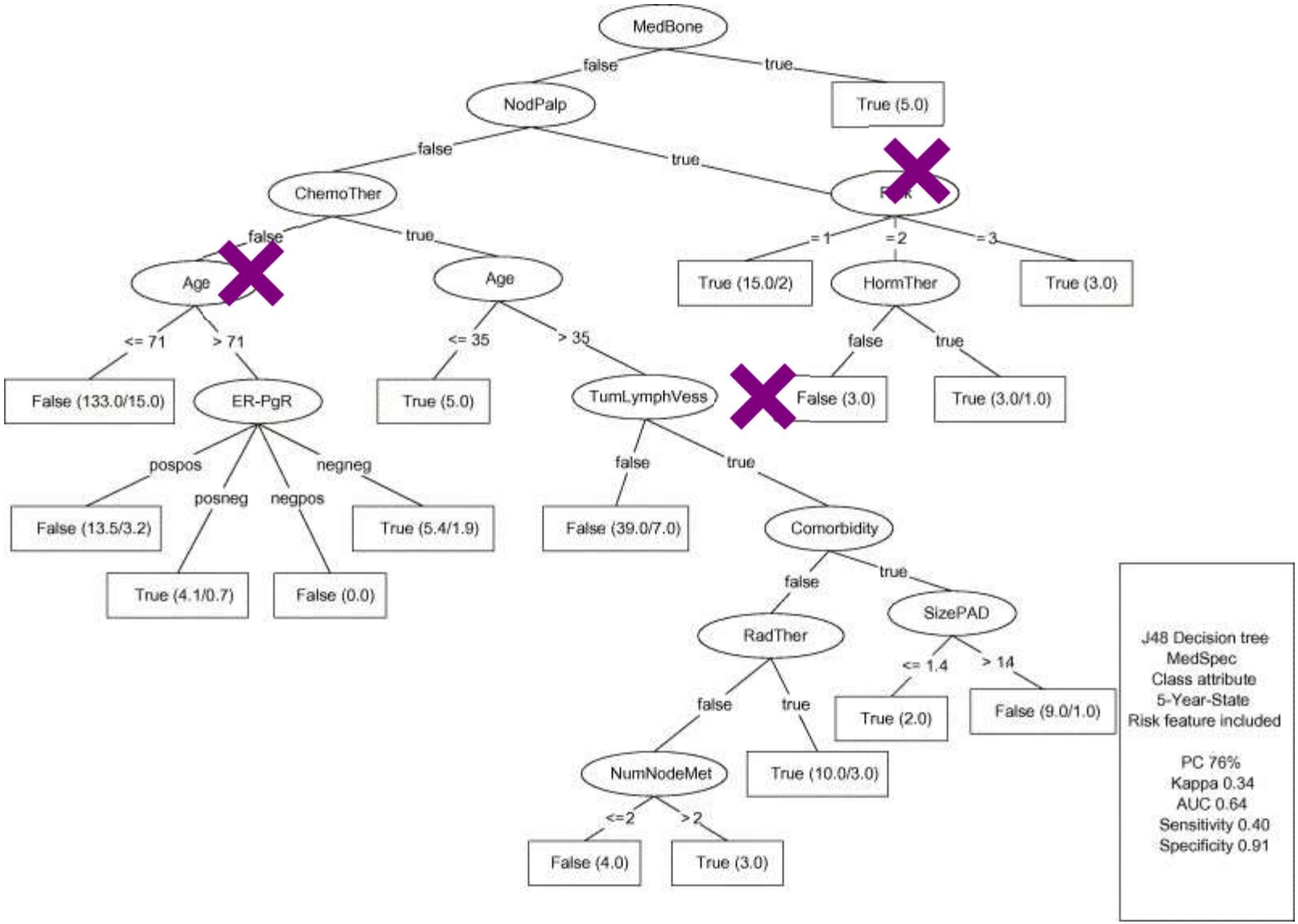
Converting A Tree to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

Post-pruning a tree



Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

Temperature:	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Attributes with Costs

Consider

- medical diagnosis, $BloodTest$ has cost \$150
- robotics, $Width\_from\_1ft$ has cost 23 sec.

How to learn a consistent tree with low expected cost?
One approach: replace gain by

- Tan and Schlimmer (1990)

$$\frac{Gain^2(S,A)}{Cost(A)}.$$

- Numez (1988)

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

where $w \in [0,1]$ determines importance of cost

Unknown Attribute Values

What if some examples missing values of A?
Use training example anyway, sort through tree

- If node n tests A, assign most common value of A among other examples sorted to node n
- assign most common value of A among other examples with same target value
- assign probability p_i to each possible value v_i of A
  - assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion

CART
(Classification and Regression Trees)

- Very similar to C4.5

Differences:

- Support numerical target variables (regression)
- Does not compute rule sets.
- Grow large tree and then prune to minimize error rate using cross-validation.
- Construct binary trees using the feature and threshold that reduces Gini impurity at each node:

Gini impurity: $\sum_{i=1}^{K}(p_i(1-p_i))$ with K classes

p_i is the probability of an element with label i to be selected.

  - Takes value zero when all elements are labeled with one category

Decision tree application to text normalization

- Analysis of raw text into pronounceable words:

He said the increase in credit limits helped B.C. Hydro achieve record net income of about $1 billion during the year ending March 31. This figure does not include any write-downs that may occur if Powerex determines that any of its customer accounts are not collectible. Cousins, however, was insistent that all debts will be collected: "We continue to pursue monies owing and we expect to be paid for electricity we have sold."

- Sentence Tokenization
- Text Normalization
  - Identify tokens in text
  - Chunk tokens into reasonably sized sections
  - Map tokens to words
  - Identify types for words

Following slides from D. Jurafsky

I. Text Processing

- He stole $100 million from the bank
- It's 13 St. Andrews St.
- The home page is http://www.stanford.edu
- Yes, see you the following tues, that's 11/12/01
- IV: four, fourth, I.V.
- IRA: I.R.A. or Ira
- 1750: seventeen fifty (date, address) or one thousand seven… (dollars)

I.1 Text Normalization Steps

- Identify tokens in text
- Chunk tokens
- Identify types of tokens
- Convert tokens to words

Step 1: identify tokens and chunk

- Whitespace can be viewed as separators
- Punctuation can be separated from the raw tokens
- Converts text into
  - ordered list of tokens
  - each with features:
    - its own preceding whitespace
    - its own succeeding punctuation

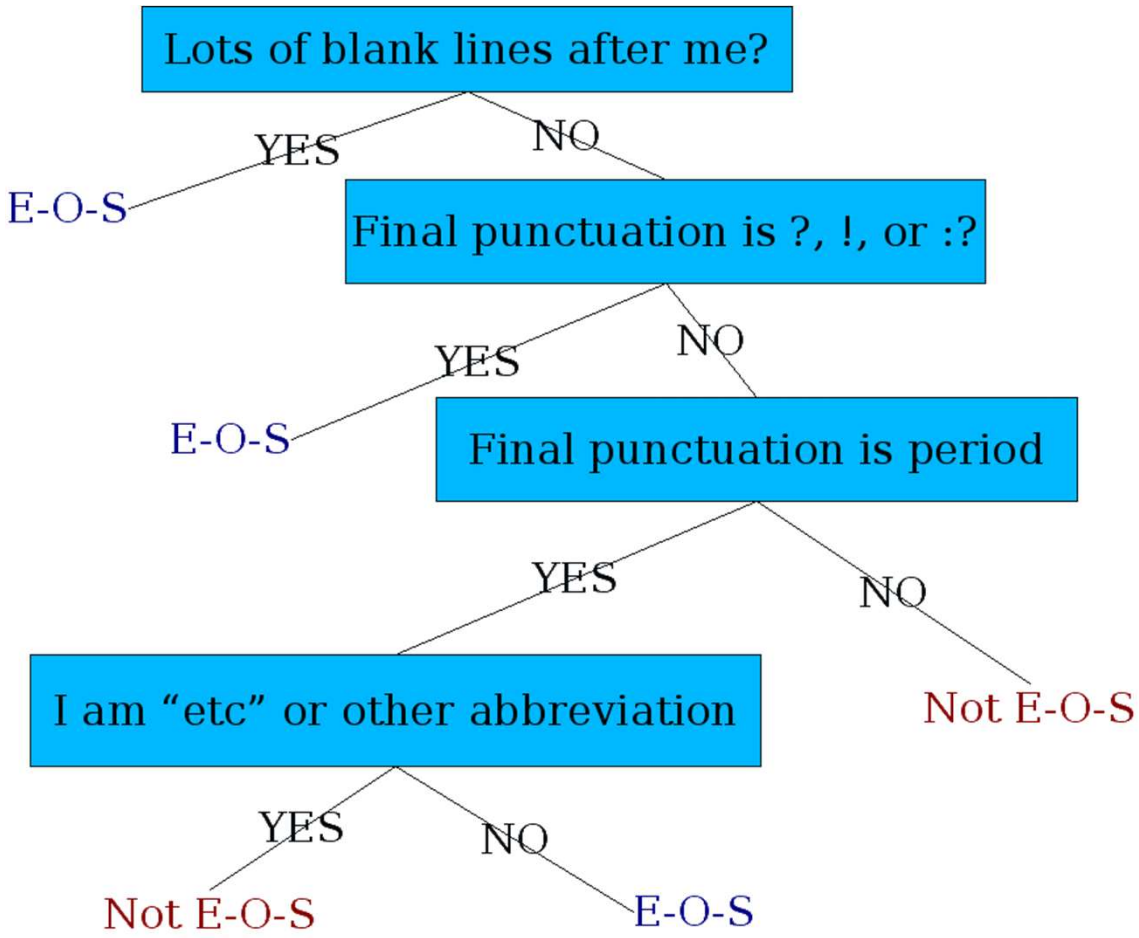
Important issue in tokenization: end-of-utterance detection

- Relatively simple if utterance ends in ?!
- But what about ambiguity of “.”
- Ambiguous between end-of-utterance and end-of-abbreviation
  - My place on Forest Ave. is around the corner.
  - I live at 360 Forest Ave.
  - (Not “I live at 360 Forest Ave..”)
- How to solve this period-disambiguation task?

How about rules for end-of-utterance detection?

- A dot with one or two letters is an abbreviation
- A dot with 3 cap letters is an abbreviation
- An abbreviation followed by 2 spaces and a capital letter is an end-of-utterance
- Non-abbreviations followed by capitalized word are breaks

Determining if a word is end-of-utterance: a Decision Tree (CART, C4.5)



EOS: end of sentence

More sophisticated decision tree features

- Prob(word with “.” occurs at end-of-s)
- Prob(word after “.” occurs at begin-of-s)
- Length of word with “.”
- Length of word after “.”
- Case of word with “.”: Upper, Lower, Cap, Number
- Case of word after “.”: Upper, Lower, Cap, Number
- Punctuation after “.” (if any)
- Abbreviation class of word with “.” (month name, unit-of-measure, title, address name, etc)

From Richard Sproat slides

Tree Induction Summary

- For classification modeling, tree induction is one of the most popular data mining tools
- Also can be used for regression
- It is:
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- Works remarkably well
  - (not the most accurate, by the way)
- Almost all data mining packages include tree algorithms
- Has advantages for model comprehensibility, which is important for:
  - model evaluation
  - communication to non-DM-savvy stakeholders