Decision tree learning

Sunita Sarawagi
IIT Bombay

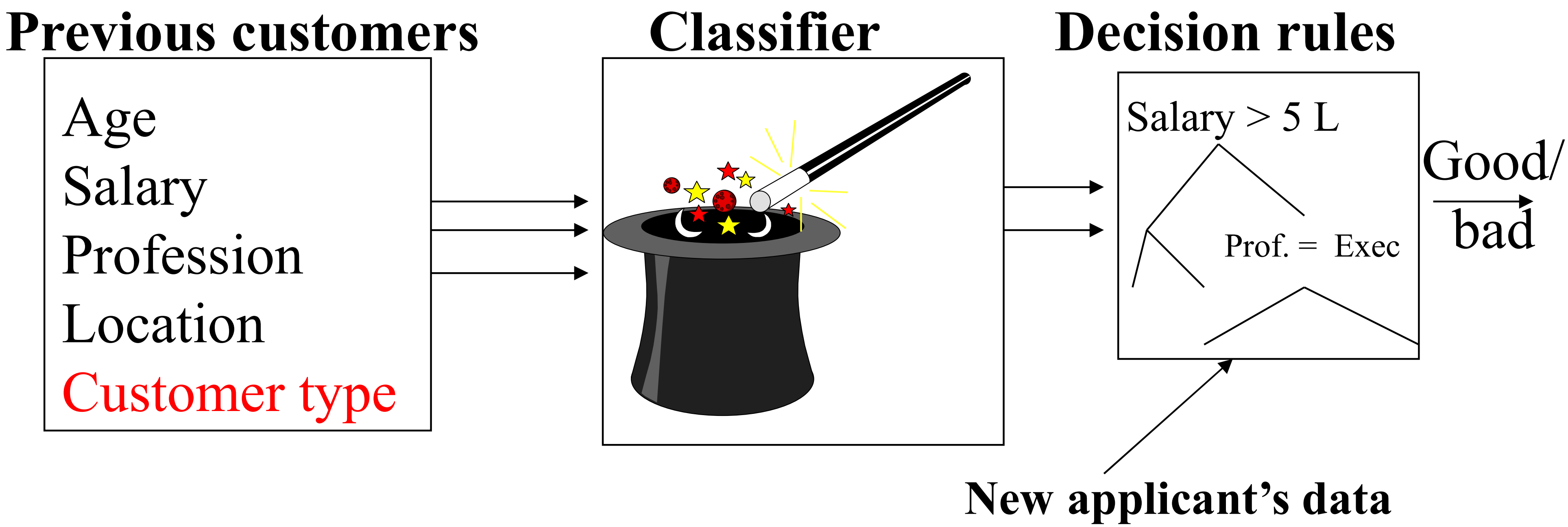
http://www.cse.iitb.ac.in/~sunita

Decision tree classifiers

- Widely used learning method
- Easy to interpret: can be re-represented as if-then-else rules
- Approximates function by piece wise constant regions
- Does not require any prior knowledge of data distribution, works well on noisy data.
- Has been applied to:
  - classify medical patients based on the disease,
  - equipment malfunction by cause,
  - loan applicant by likelihood of payment.
  - lots and lots of other applications..

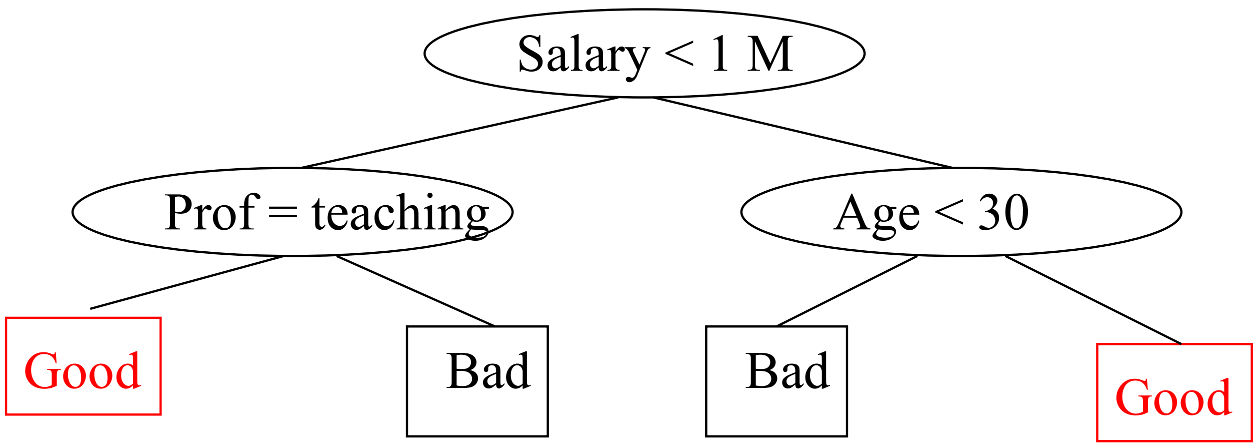
Setting

- Given old data about customers and payments, predict new applicant's loan eligibility.



Decision trees

- Tree where internal nodes are simple decision rules on one or more attributes and leaf nodes are predicted class labels.

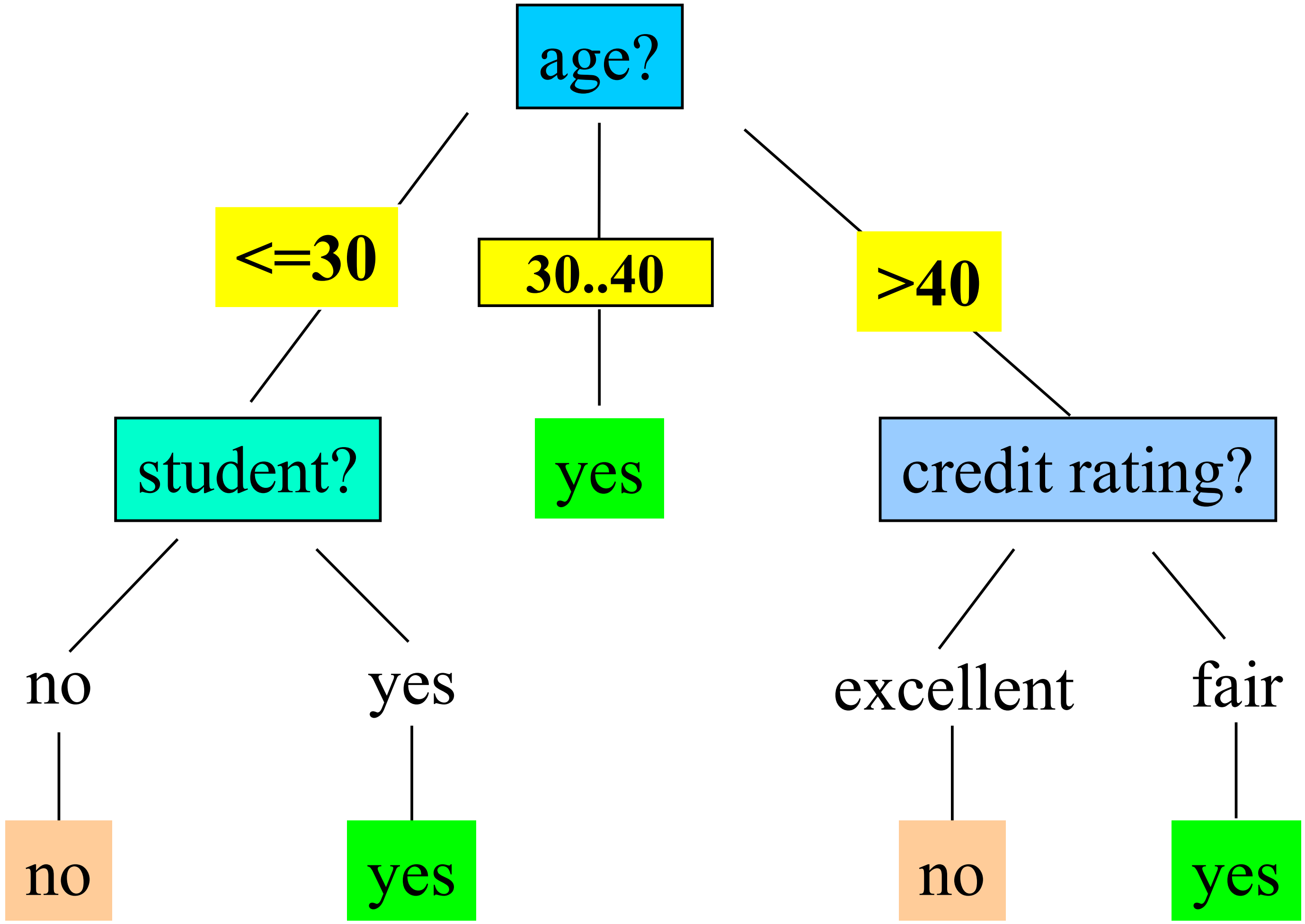


Training Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"

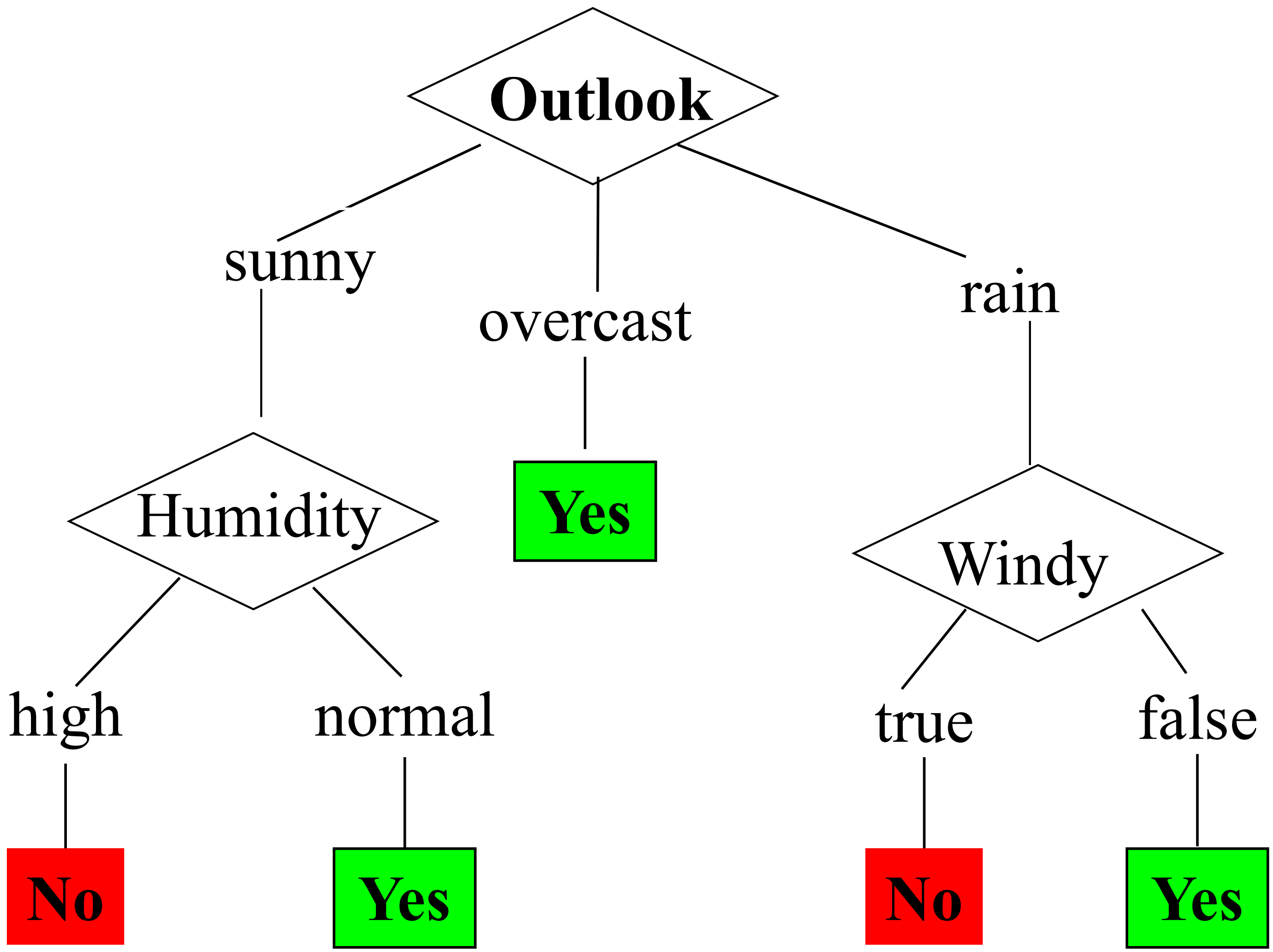


Weather Data: Play or not Play?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Note:
Outlook is the Forecast,
no relation to Microsoft email program

Example Tree for "Play?"



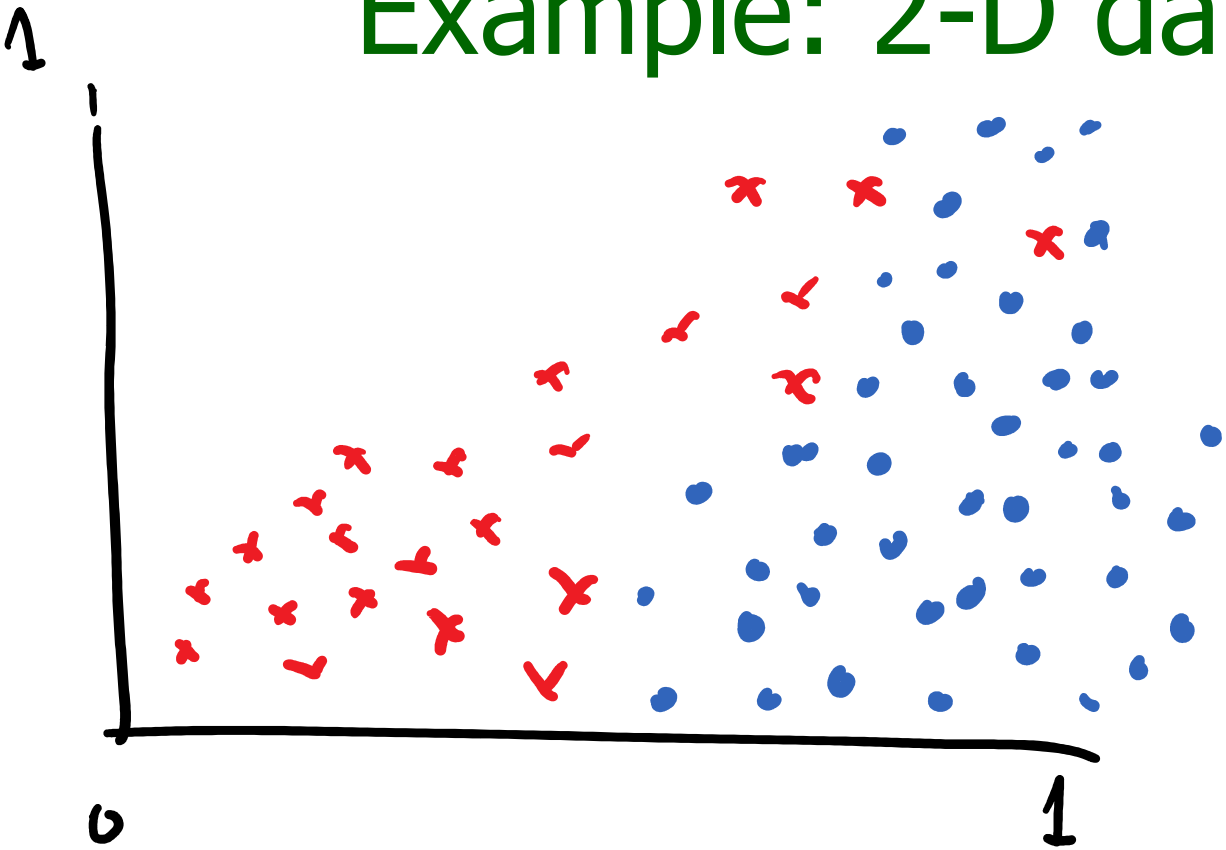
Topics to be covered

- Tree construction:
  - Basic tree learning algorithm
  - Measures of predictive ability
  - High performance decision tree construction: Sprint
- Tree pruning:
  - Why prune
  - Methods of pruning
- Other issues:
  - Handling missing data
  - Continuous class labels
  - Effect of training size

Tree learning algorithms

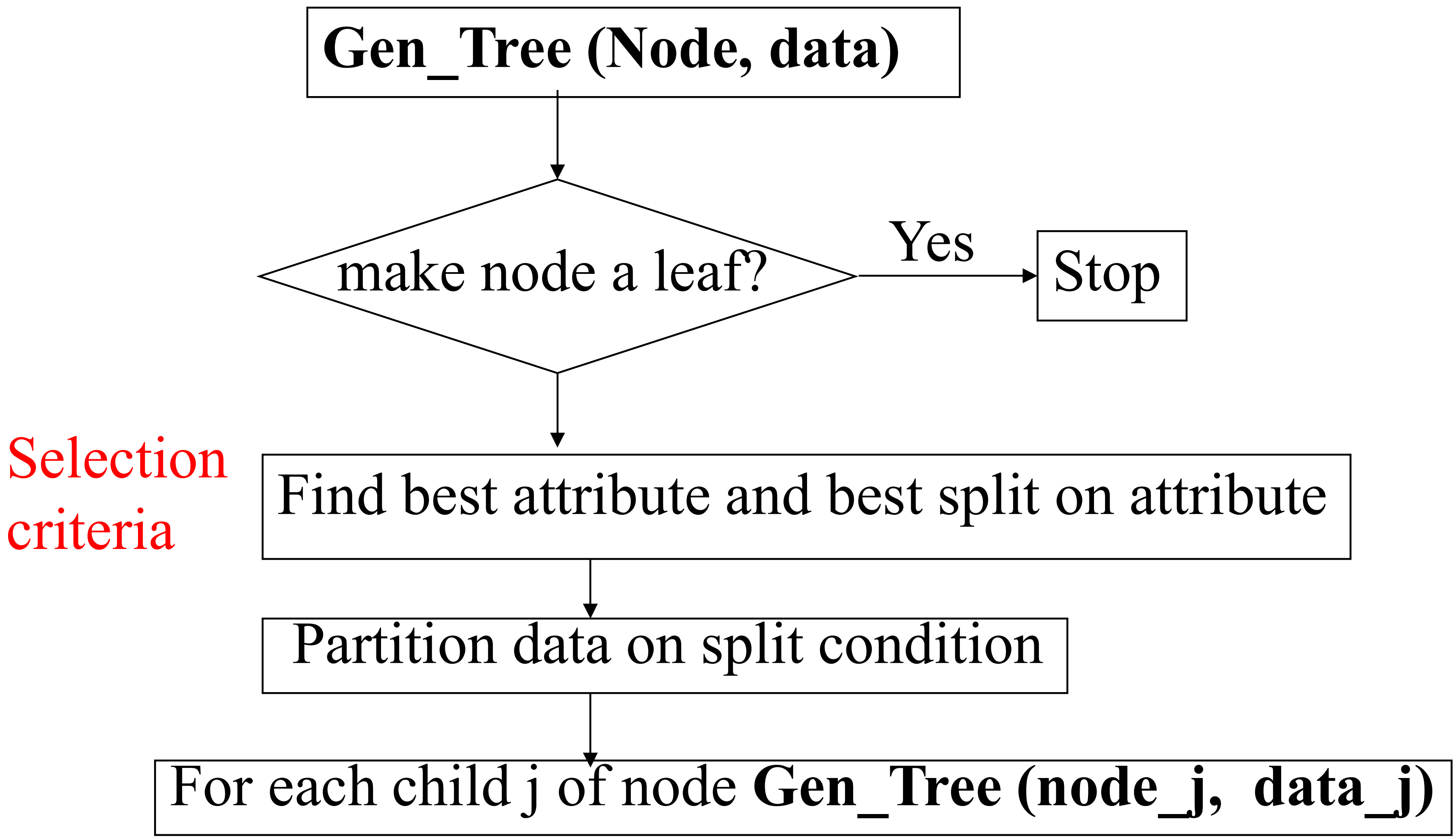
- ID3 (Quinlan 1986)
- Successor C4.5 (Quinlan 1993)
- CART
- SLIQ (Mehta et al)
- SPRINT (Shafer et al)

Example: 2-D data



Basic algorithm for tree building

- Greedy top-down construction.



Split criteria

- Select the attribute that is best for classification.
- Intuitively pick one that best separates instances of different classes.
- Quantifying the intuitive: measuring separability:
- First define *impurity* of an arbitrary set S consisting of K classes
- Smallest when consisting of only one class, highest when all classes in equal number.
- Should allow computations in multiple stages.

Measures of impurity

- Entropy

$$Entropy(S) = -\sum_{i=1}^{k} p_i \log p_i$$

$p_1, p_2 \ldots p_k$ denote the fraction of examples in classes $1, 2, \ldots k$ respectively in set S.

- Gini

$$Gini(S) = 1 - \sum_{i=1}^{k} p_i^2$$

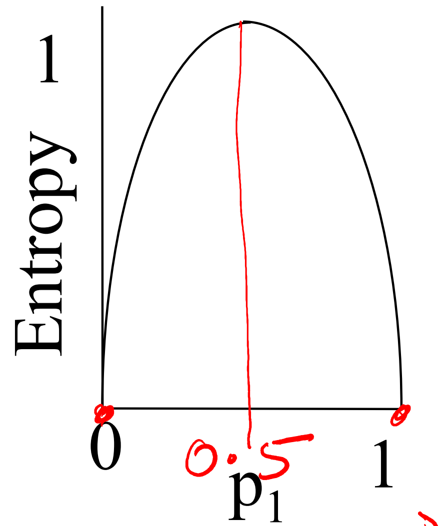
Information gain

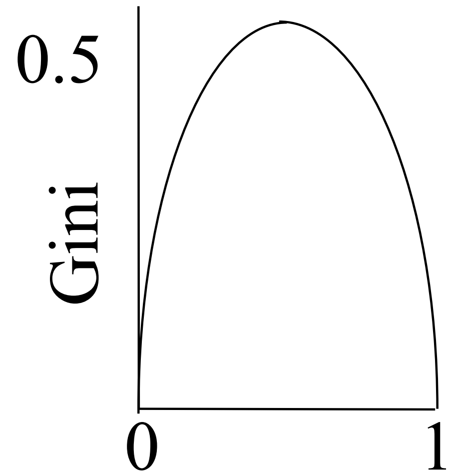
$-p_1 \log p_1 - p_2 \log p_2$

$k = 2$

p_1

$p_2 = 1 - p_1$





- Information gain on partitioning S into r subsets
- Impurity (S) - sum of weighted impurity of each subset

of information

$$Gain(S, S_1..S_r) = Entropy(S) - \sum_{j=1}^{r} \frac{S_j}{S} Entropy(S_j)$$

Average entropy after splitting S into $S_1, S_2 \ldots S_k$

*Properties of the entropy

- The multistage property:

$$\text{entropy}(p,q,r) = \text{entropy}(p,q+r) + (q+r)\times\text{entropy}(\frac{q}{q+r},\frac{r}{q+r})$$

- Simplification of computation:

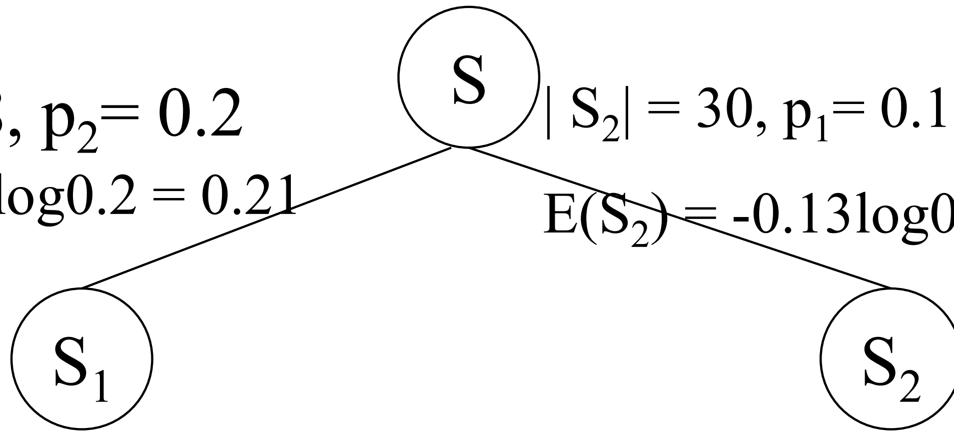
$$\text{info}([2,3,4]) = -2/9\times\log(2/9) - 3/9\times\log(3/9) - 4/9\times\log(4/9)$$

Information gain: example

$$K = 2,\ |S| = 100,\ p_1 = 0.6,\ p_2 = 0.4$$
$$E(S) = -0.6 \log(0.6) - 0.4 \log(0.4) = 0.29$$

$$|S_1| = 70,\ p_1 = 0.8,\ p_2 = 0.2$$
$$E(S_1) = -0.8 \log 0.8 - 0.2 \log 0.2 = 0.21$$

$$|S_2| = 30,\ p_1 = 0.13,\ p_2 = 0.87$$
$$E(S_2) = -0.13 \log 0.13 - 0.87 \log 0.87 = .16$$

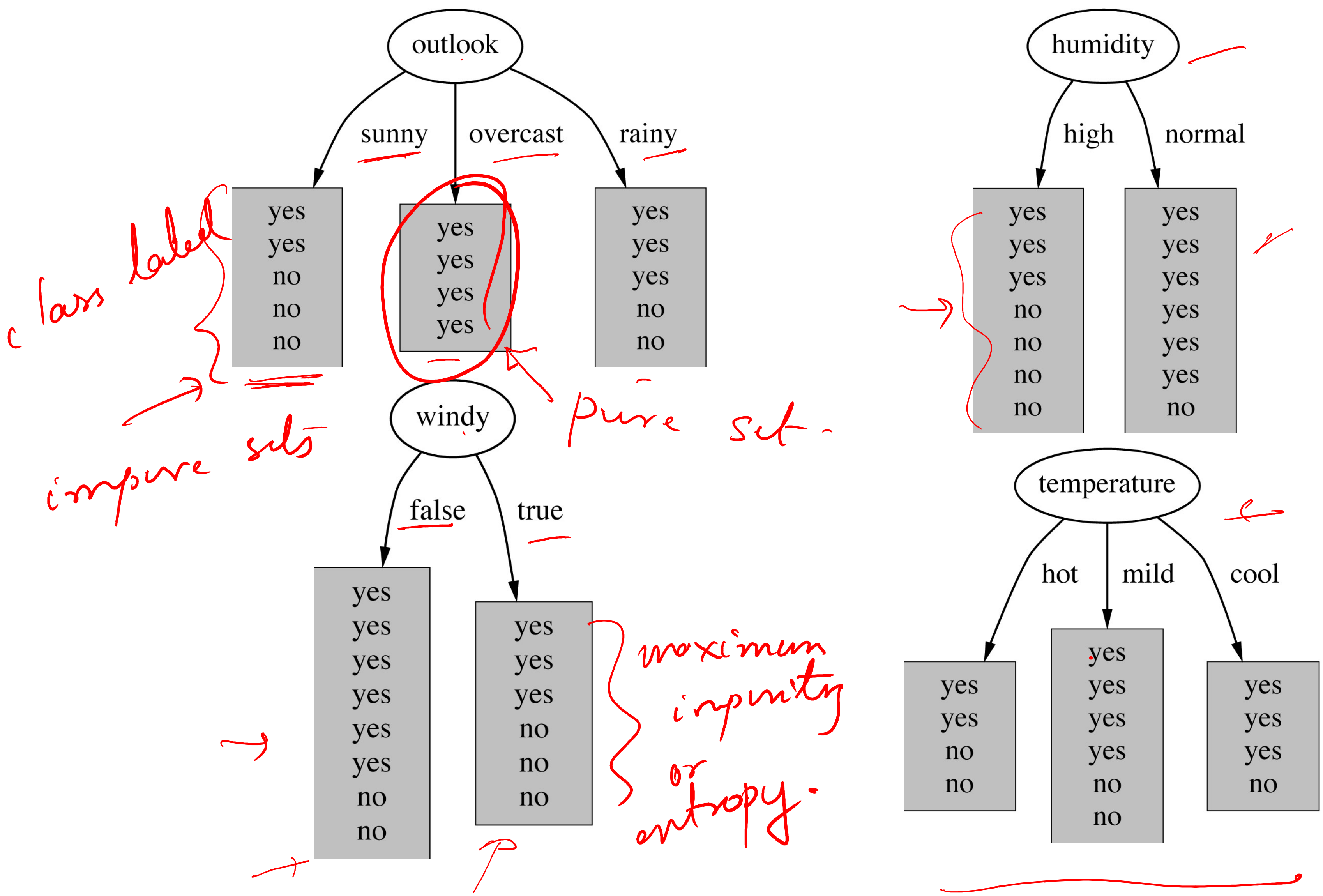


Information gain: $E(S) - (0.7\,E(S_1) + 0.3\,E(S_2)) = 0.1$

Weather Data: Play or not Play?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Which attribute to select?



Example: attribute "Outlook"

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\ \text{bits}$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0\ \text{bits}$$

Note: log(0) is not defined, but we evaluate 0*log(0) as zero

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\ \text{bits}$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693\ \text{bits}$$

Computing the information gain

- Information gain:

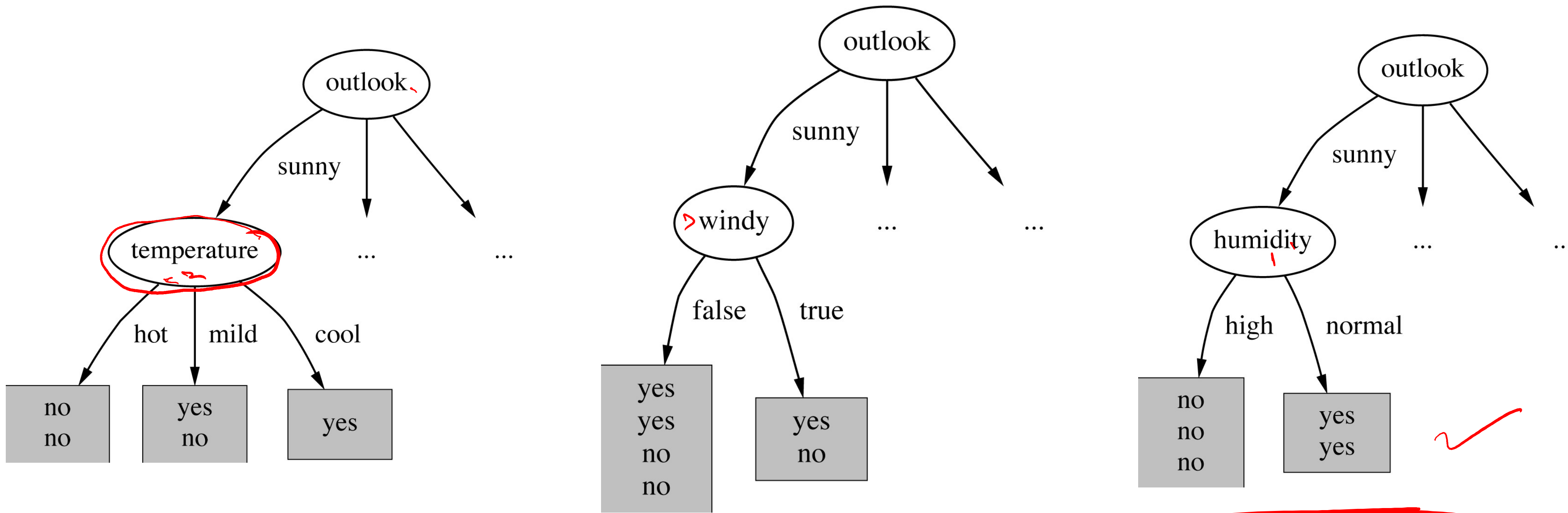
(information before split) – (information after split)

$$\text{gain}(\text{"Outlook"}) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

- Information gain for attributes from weather data:

$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$
$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$
$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$
$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

Continuing to split

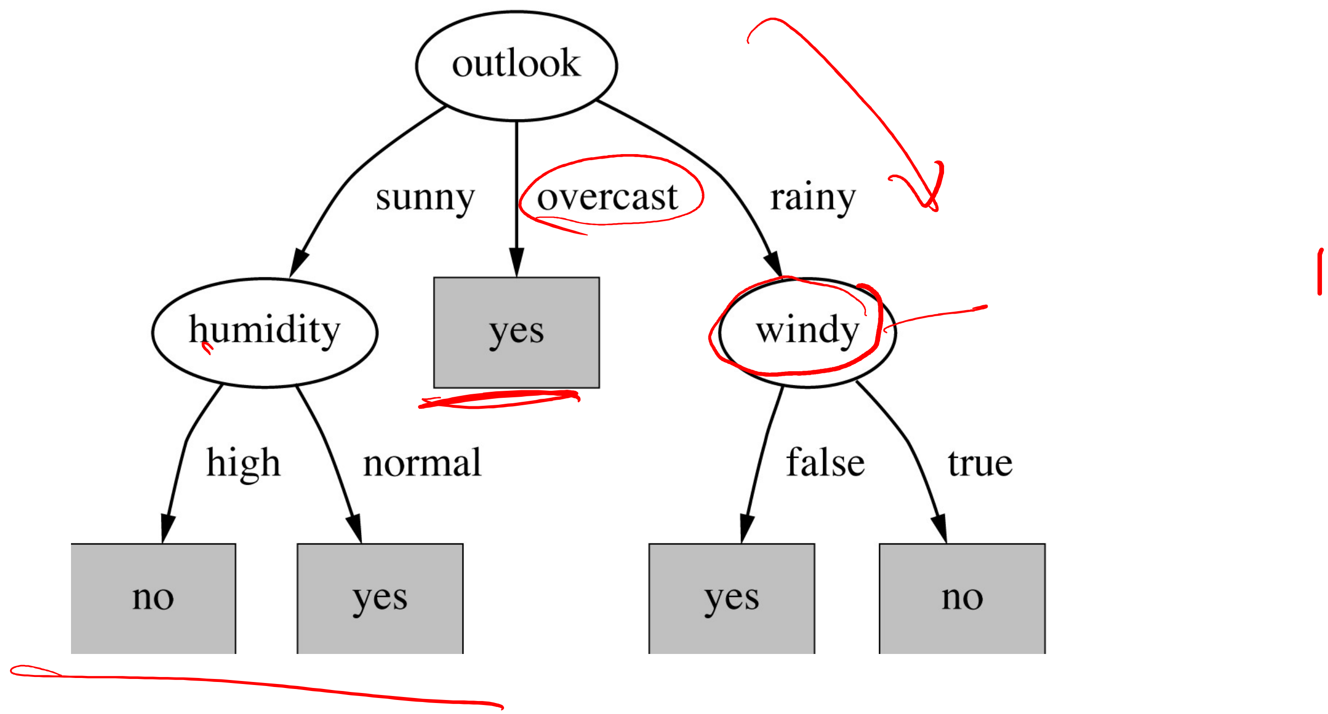


$$\text{gain}(\text{"Humidity"}) = 0.971 \text{ bits}$$

$$\text{gain}(\text{"Temperature"}) = 0.571 \text{ bits}$$

$$\text{gain}(\text{"Windy"}) = 0.020 \text{ bits}$$

The final decision tree



- Note: not all leaves need to be pure; sometimes identical instances have different classes

  ⇒ Splitting stops when data can't be split any further

Highly-branching attributes

- Problematic: attributes with a large number of values (extreme case: ID code)
- Subsets are more likely to be pure if there is a large number of values
  - ⇒Information gain is biased towards choosing attributes with a large number of values
  - ⇒This may result in *overfitting* (selection of an attribute that is non-optimal for prediction)

Data Setup: Attribute Lists

- One list for each attribute
- Entries in an Attribute List consist of:
  - attribute value
  - class value
  - record id

Example list:

Age	Risk	RID
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

- Lists for continuous attributes are in sorted order
- Lists may be disk-resident
- Each leaf-node has its own set of attribute lists representing the training examples belonging to that leaf

Attribute Lists: Example

attribute class label record id .

Age	Car Type	Risk
23	family	High
17	sports	High
43	sports	High
68	family	Low
32	truck	Low
20	family	High

⇨

Age	Risk	RID
23	High	0
17	High	1
43	High	2
68	Low	3
32	Low	4
20	High	5

Car Type	Risk	RID
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5

vertical splits of data

group on categorical attributes

Initial Attribute Lists for the root node:

sort on numerical attribute

Age	Risk	RID
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

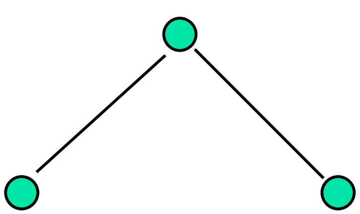
Car Type	Risk	RID
family	High	0
family	High	5
family	Low	3
sports	High	2
sports	High	1
truck	Low	4

Split Points: Continuous Attrib.

Attribute List

Age	Risk	RID
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

High	Low
4	2



Position of cursor in scan | State of Class Histograms: Left Child | Right Child | GINI Index:

0: Age < 17

High	Low
0	0

High	Low
4	2

GINI = undef

1: Age < 20

High	Low
1	0

High	Low
3	2

GINI = 0.4

3: Age < 32

High	Low
3	0

High	Low
1	2

GINI = 0.222

6

High	Low
4	2

High	Low
0	0

GINI = undef

Split Points: Categorical Attrib.

- Consider splits of the form: *value(A)* $\in$ *{x1, x2, ..., xn}*
  - Example: CarType $\in$ {family, sports}
- Evaluate this split-form for subsets of *domain(A)*
- To evaluate splits on attribute *A* for a given tree node:

```
initialize class/value matrix of node to zeroes;

for each record in the attribute list do
        increment appropriate count in matrix;

evaluate splitting index for various subsets using the constructed matrix;
```

Pros and Cons of decision trees

- Pros
  - + Reasonable training time
  - + Fast application
  - + Easy to interpret
  - + Easy to implement
  - + Intuitive

- Cons
  - – Not effective for very high dimensional data where information about the class is spread in small ways over many correlated features
    - –Example: words in text classification
  - –Not robust to dropping of important features even when correlated substitutes exist in data