Decision Tree Learning

Prof. Dr. M. Elif Karslıgil

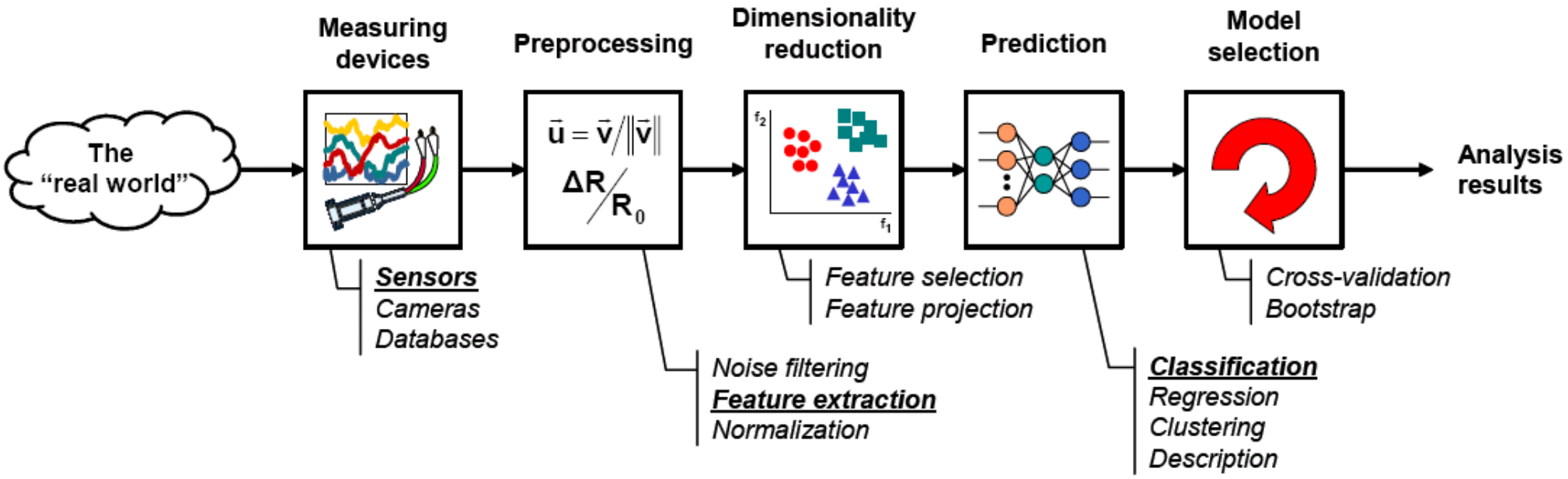
Yildiz Technical University

Computer Engineering Department

Intelligent Systems Laboratory

Adapted from slides by Maria-Florina (Nina) Balcan and Vinitra Swamy

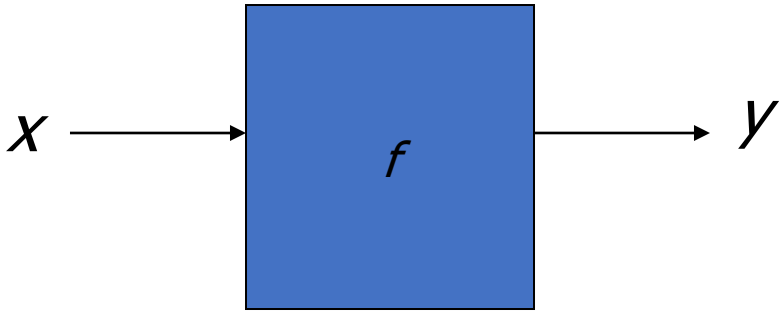
Stages of a machine learning system



Supervised Learning

- Learn to predict output when given an input vector

- Samples : $D = \{d_1, d_2, .. d_n\}$

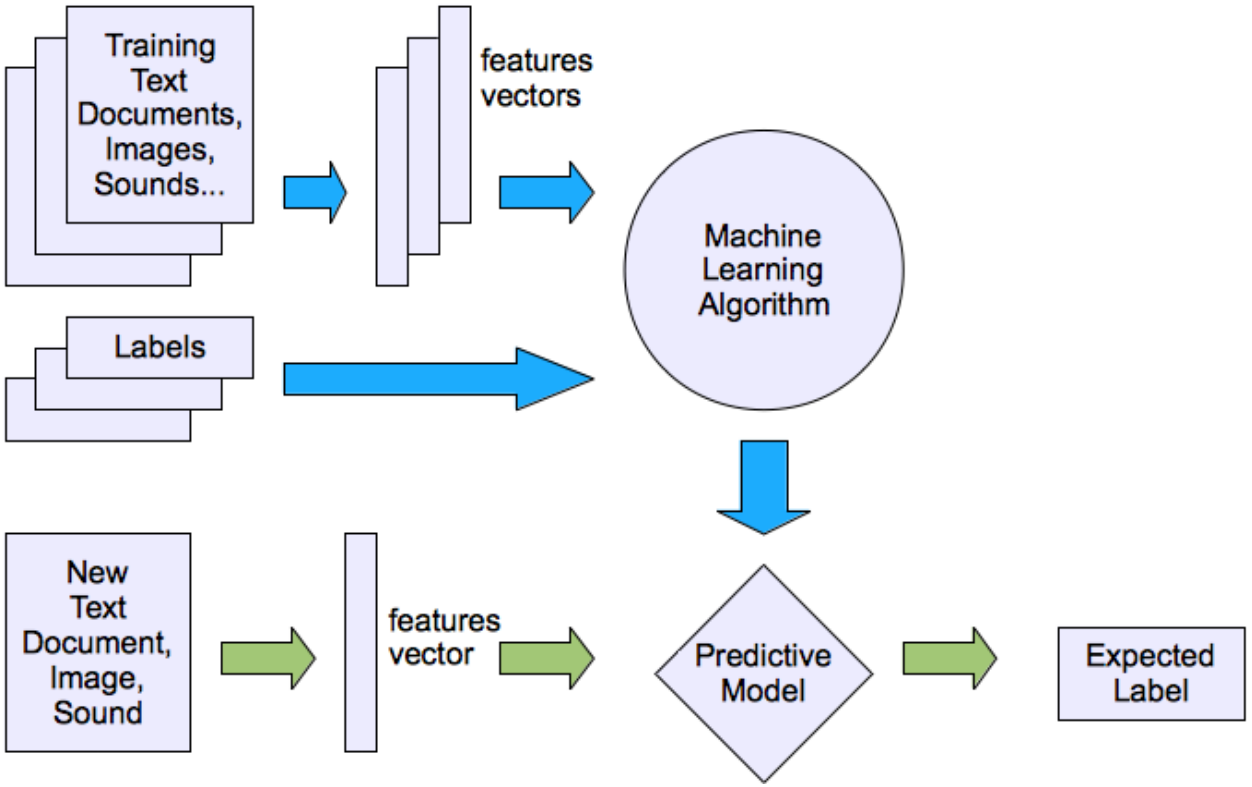


$d_i = <x_i, y_i>$

x_i : Input vector

y_i : Desired Output

Supervised Learning - Classification



Supervised Learning

- Regression:

  Input: continious or discrete → **Output:** continous

  Example: predicting the price of a house given house features

- Classification:

  Input: continious or discrete → **Output:** discrete

  Example: speech recognition, biometric identification

Classifiers

- Given inputs **x** and classes y we can do classification in several ways. How?

(features)

$\mathbf{x}$

e.g:
- height
- weight
- color

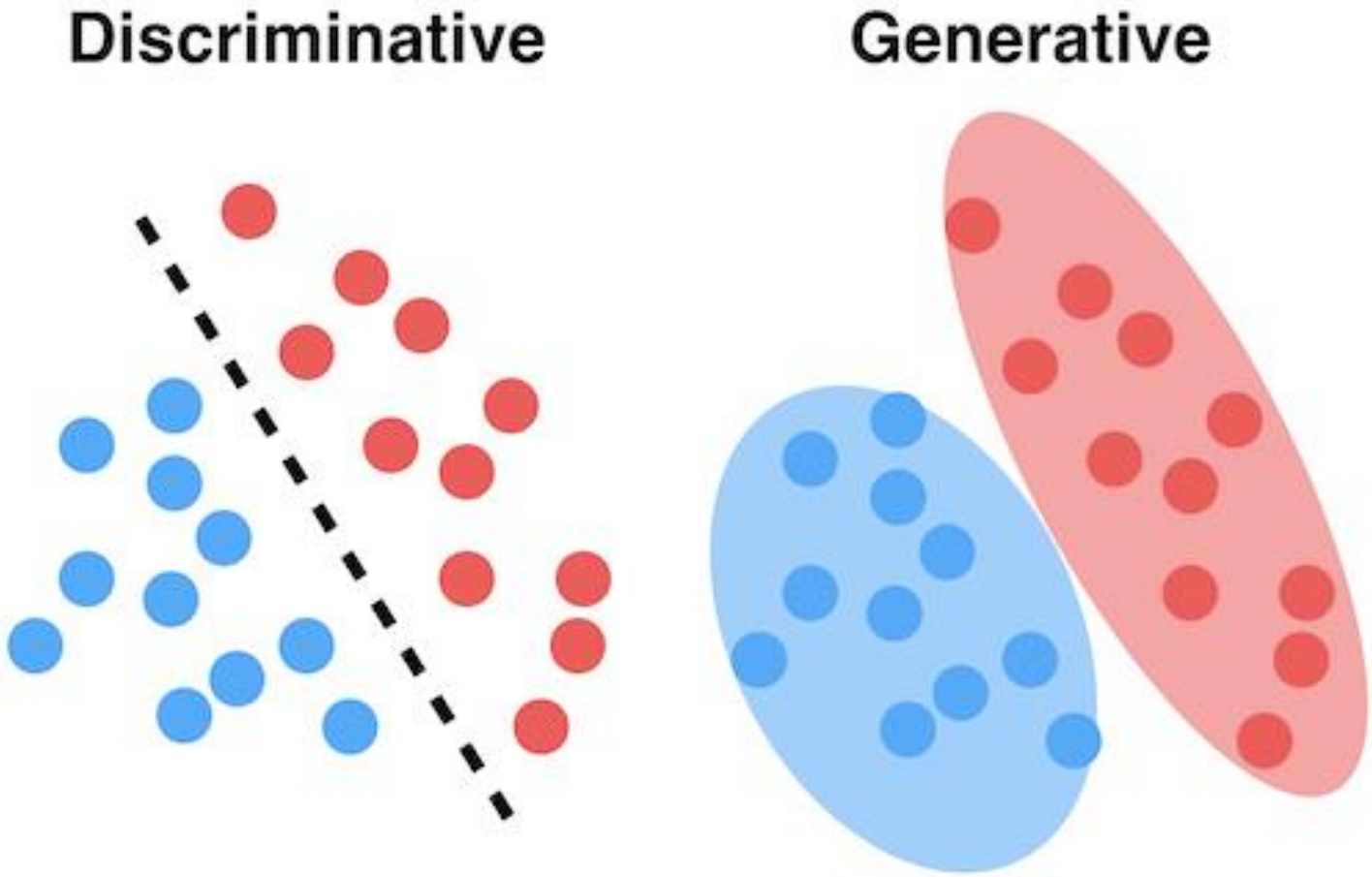
(class label)

y

elefant

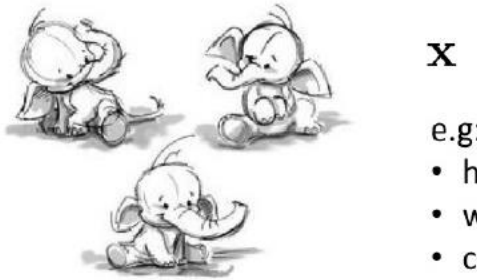
bear

Two Approaches to Supervised Learning



Discriminative Classifiers

- **Discriminative** classifiers try to either:
  - learn mappings directly from the space of inputs $\mathcal{X}$ to class labels $\{0, 1, 2, \ldots, K\}$



(features)

$\mathbf{x}$

e.g:
- height
- weight
- color

(class label)

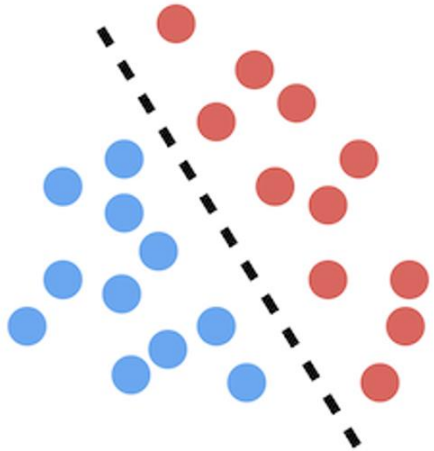
y

elefant

bear

For example:
- Linear regression (as a classifier)
- Neural Networks

Learn to **separate** classes directly



Generative Classifiers

- **Generative** classifiers try to model $p(\mathbf{x}|y)$
- Classification via Bayes rule (thus also called Bayes classifiers)

(prob. of features given label) $p(\mathbf{x}|y)$ ← (class label) y

e.g:
- height
- weight
- color

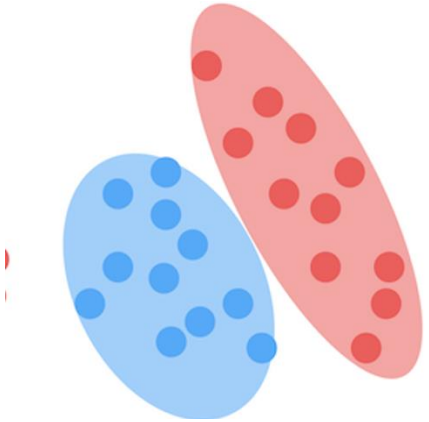
elefant

bear

Tries to model:
- How does data look like for a class?

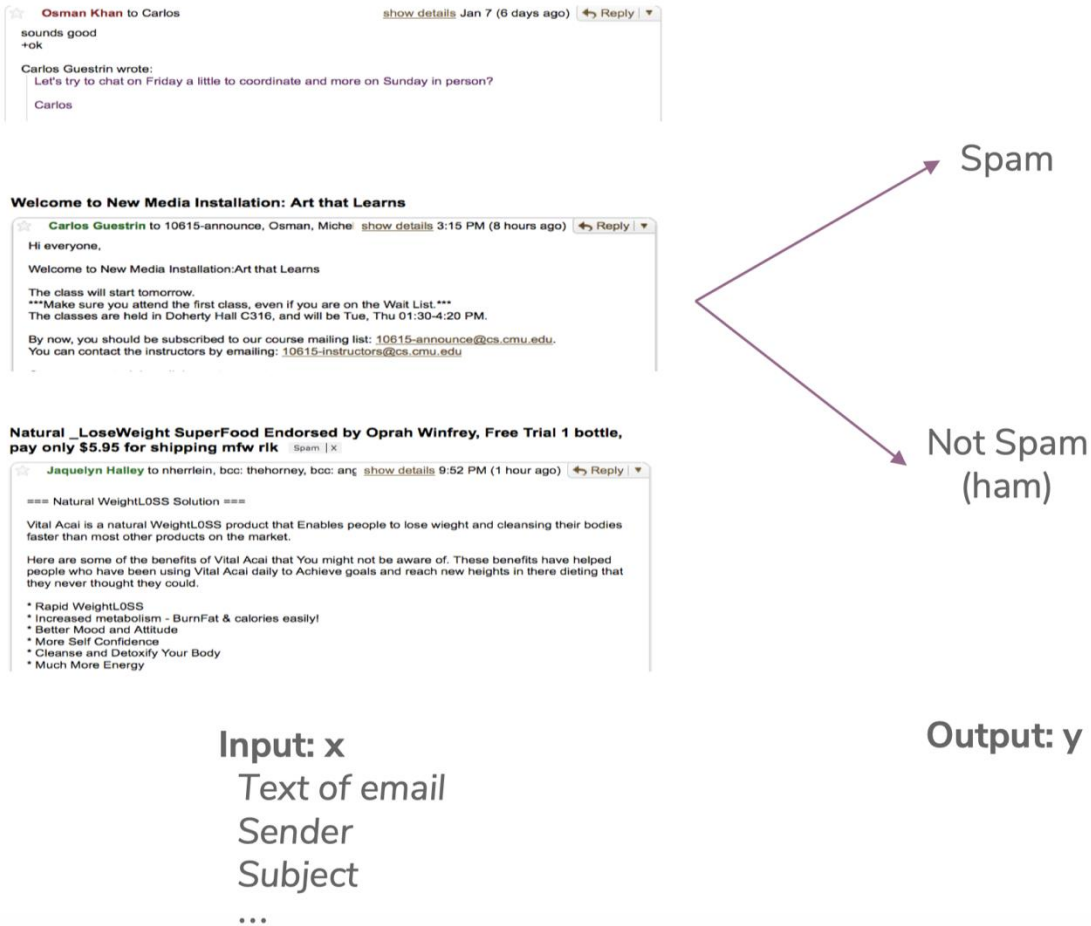
Classification (How?)

Learn the input distribution



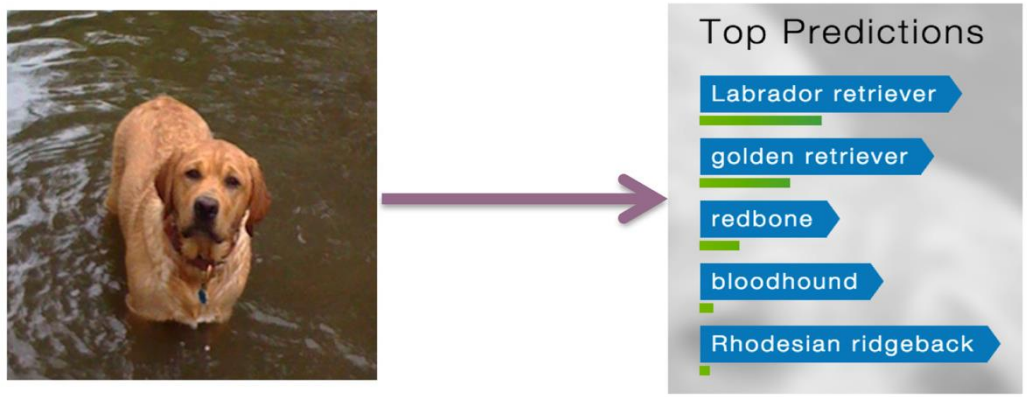
Binary Classification

Example: Spam Mail filter



Multiclass Classification

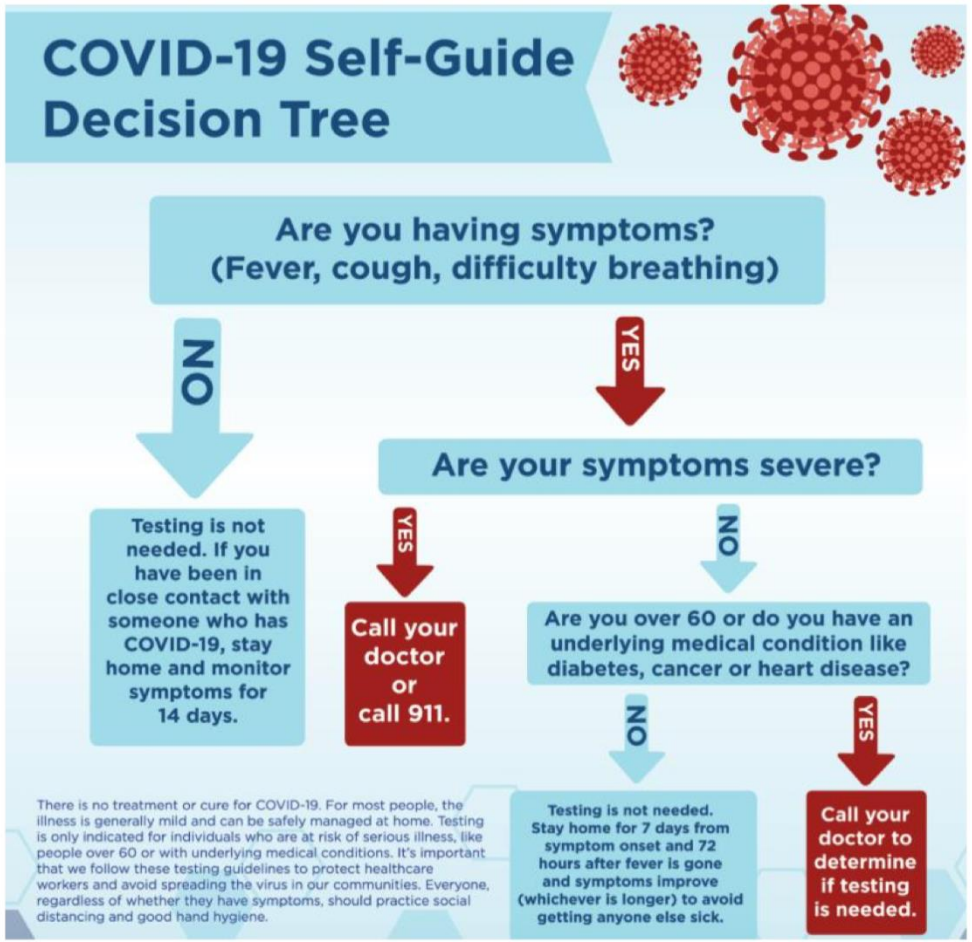
Example: Object Detection



Input: x
Pixels

Output: y
Class
(+ Probability)

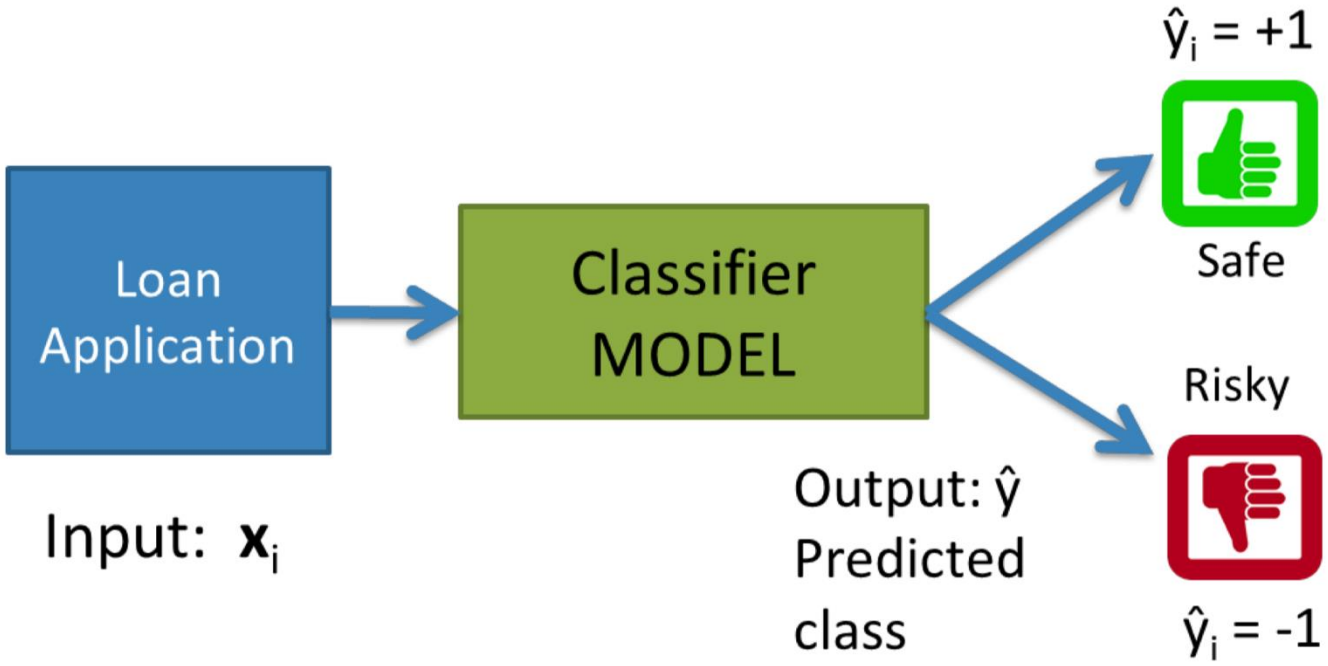
How do we make decisions ?



https://www.holzer.org/coronavirus-covid-19-updates/

Supervised Classification

Supervised Classification

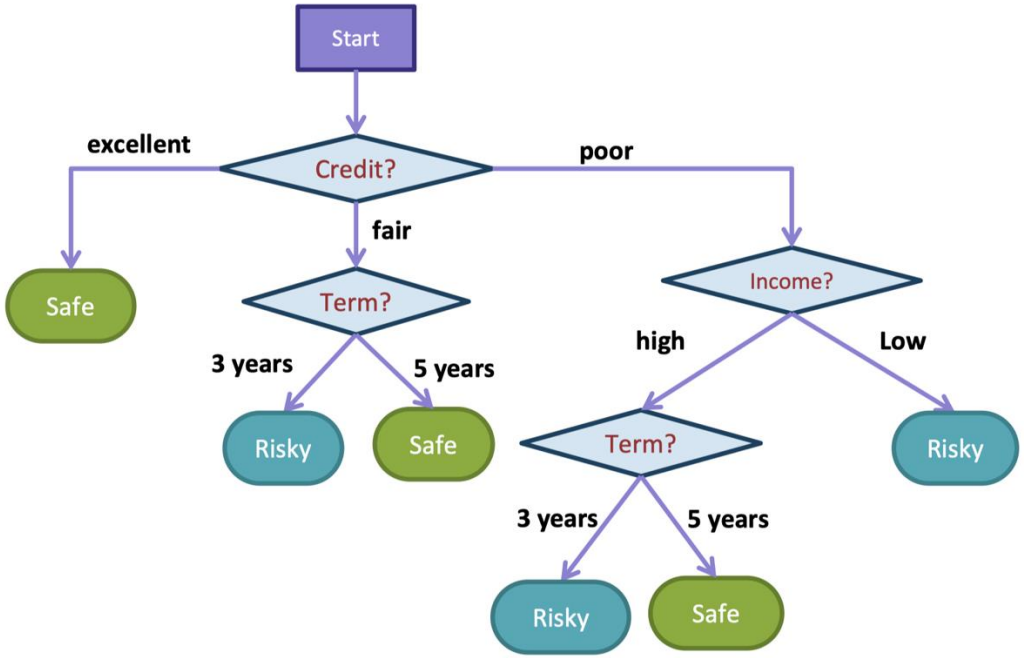


Supervised Classification

Data (N observations, 3 features)

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	safe
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Supervised Classification - Decision Trees

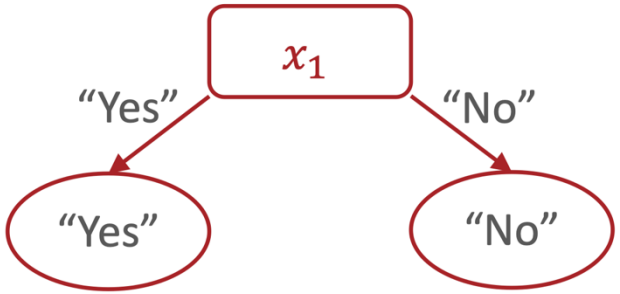


- **internal node:** testing a feature
- **branch:** splits into possible values of a feature
- **leaf:** final decision (the class value)

Decision Stump

- A decision stump is a Decision Tree, which uses only a single attribute for splitting.

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?	$\hat{y}$ Predictions
Yes	Low	Normal	No	Yes
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	No
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes



Decision Tree Learning

Example: decide whether a friend will play tennis or not in a given day

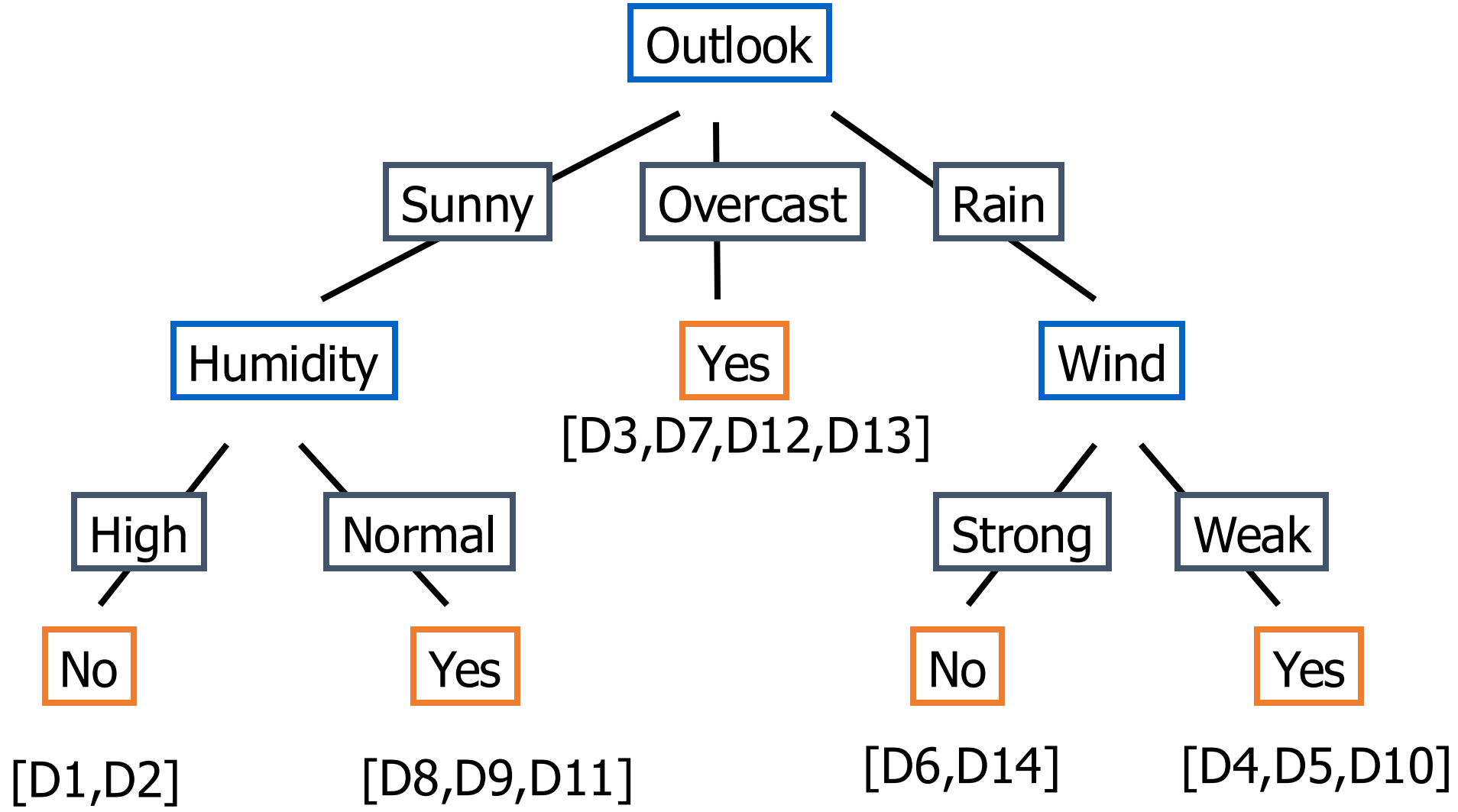
Simple Training Data Set

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

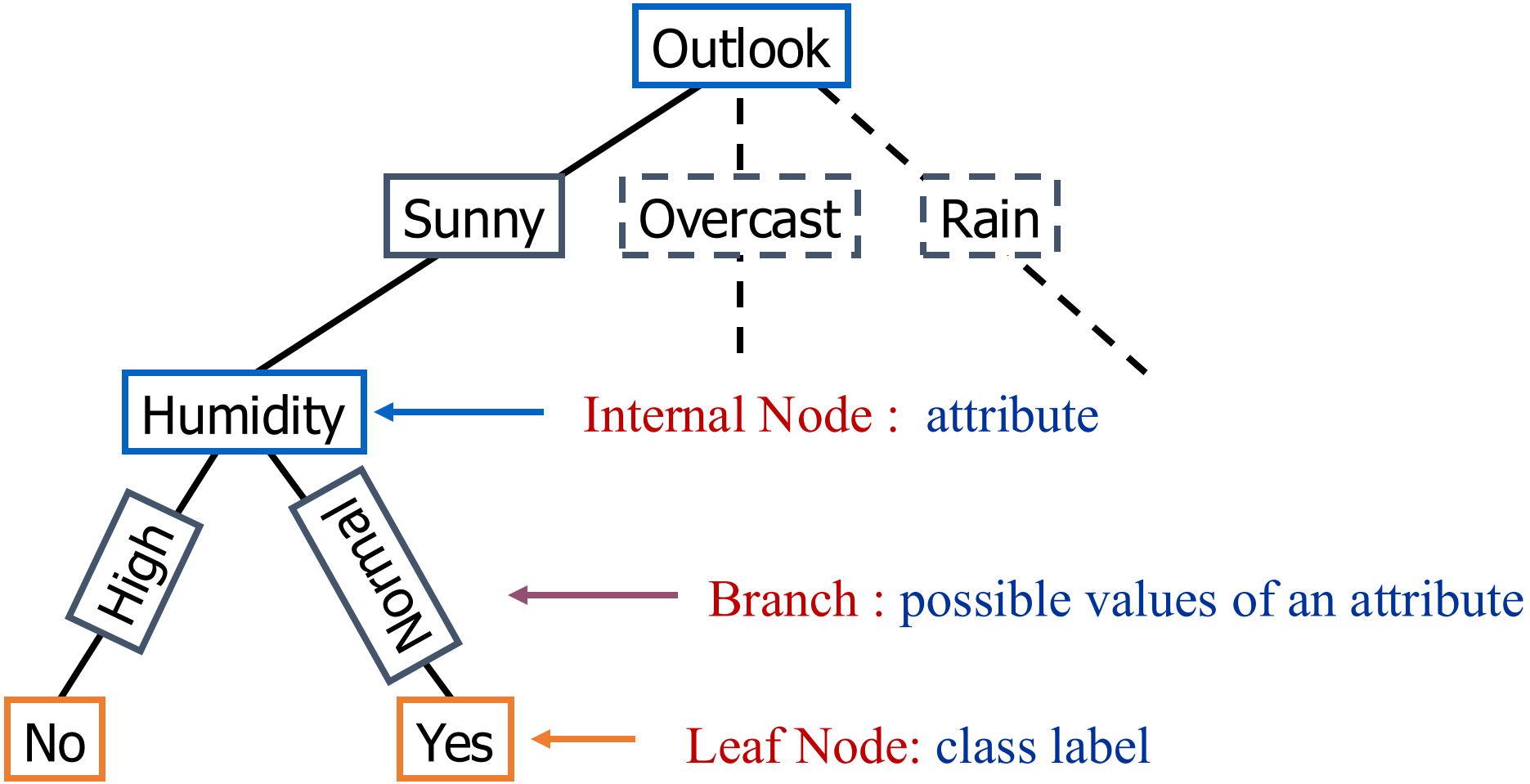
example (D4)

label (Play Tennis)

Decision Tree Learning



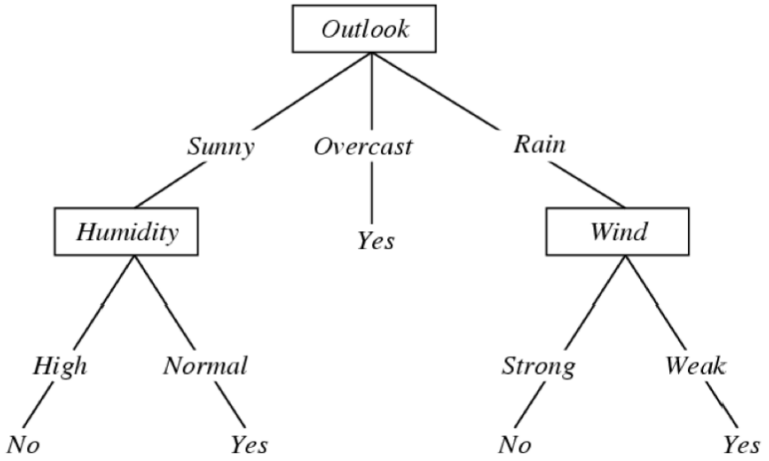
Decision Tree Learning



Decision Tree Learning

- Each internal node: test one (discrete-valued) attribute X_i
- Each branch from a node: corresponds to one possible values for X_i
- Each leaf node: predict Y (or P(Y=1|x $\in$ leaf))

Example: A Decision tree for
f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?



Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

E.g., x=(Outlook=sunny, Temperature-Hot, Humidity=Normal,Wind=High), f(x)=Yes.

Handling Real Valued Attributes

- For continuous attributes, use threshold splits
  - Split the tree into $x_k < t$ and $x_k \geq t$
  - Can split on the same attribute multiple times on the same path down the tree
- How to pick the threshold t?
  - Try every possible t

How many possible t are there?

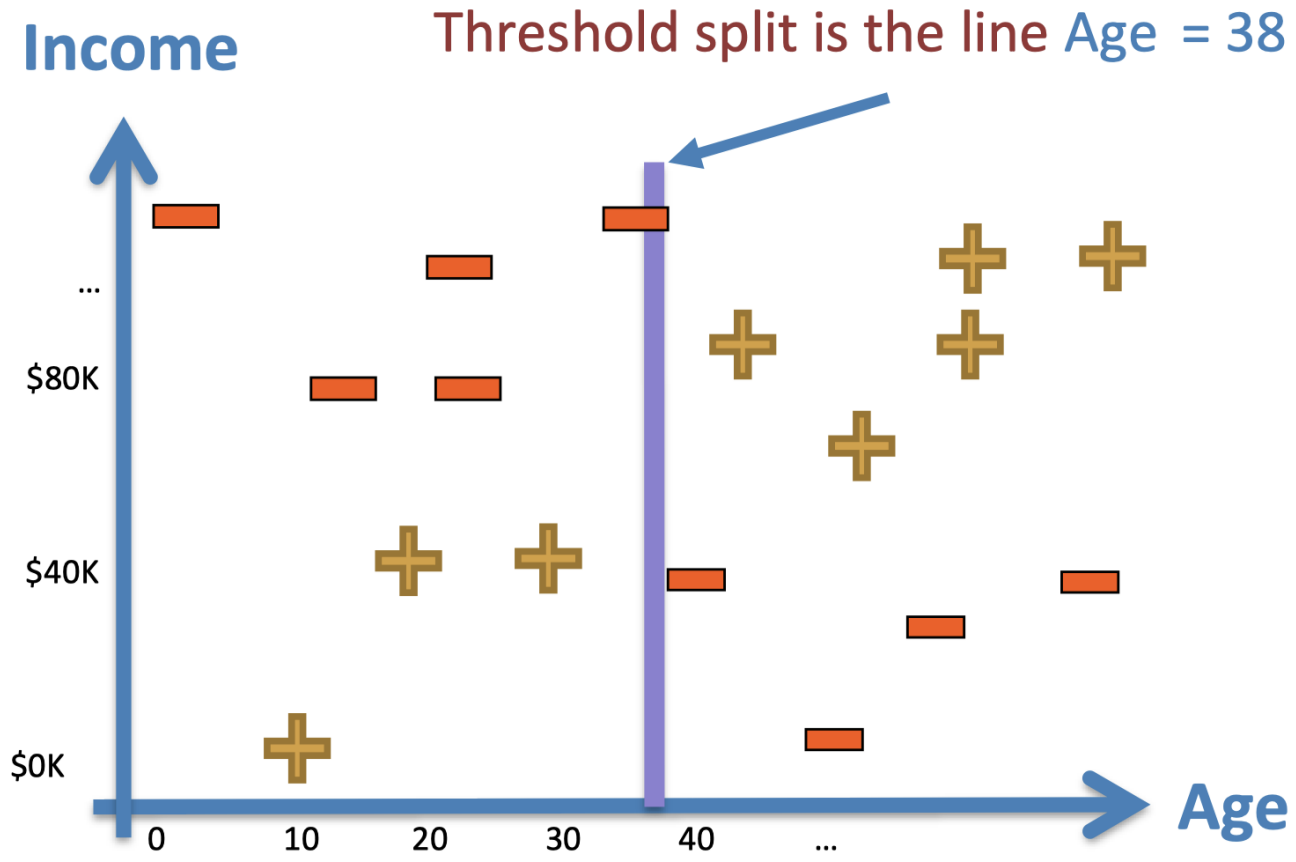
How do we select the threshold?

- **Step 1:** Sort the values of a feature $h_j(x)$:

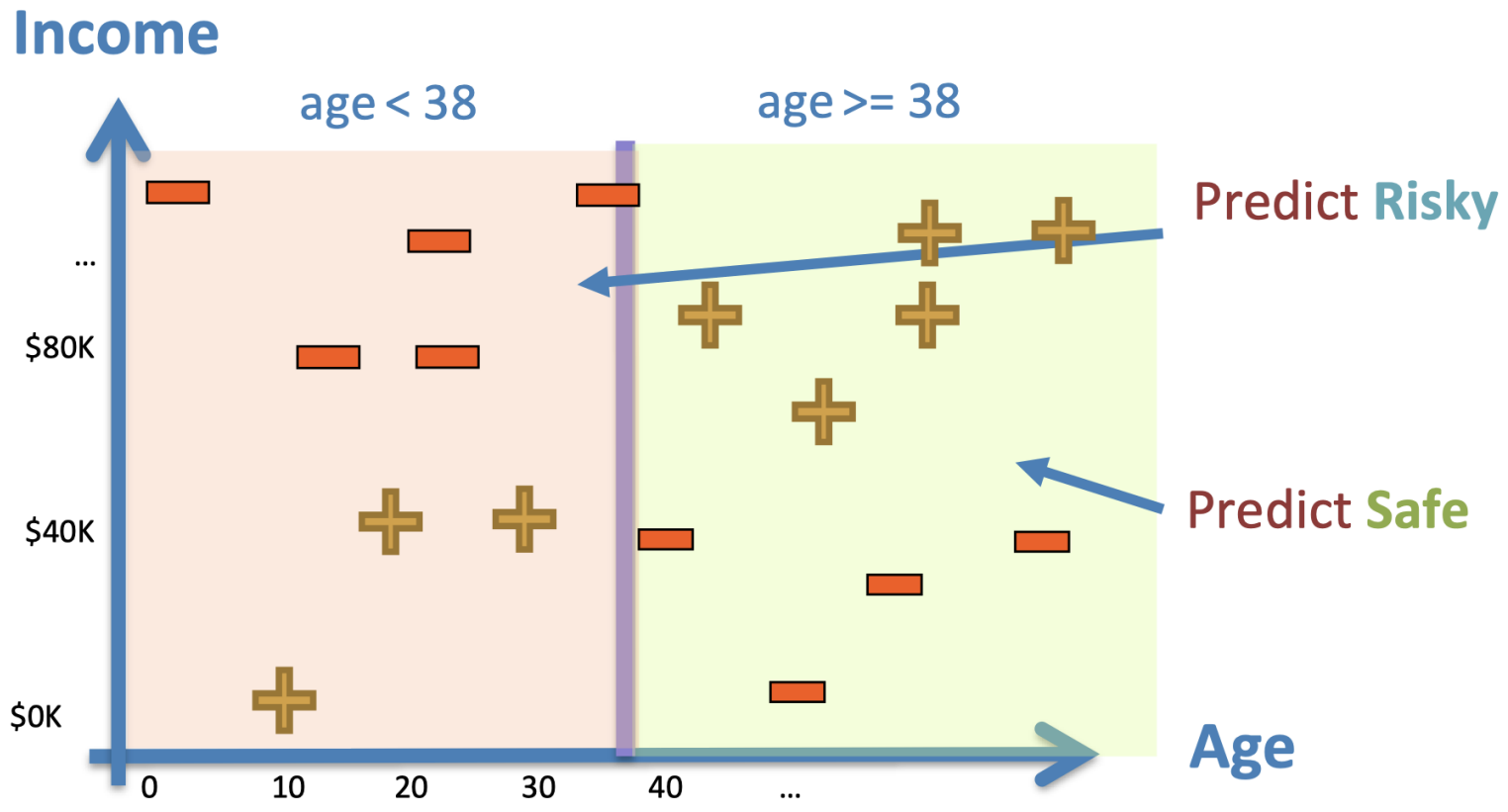
  Let $\{v_1, v_2, v_3, \ldots v_N\}$ denote sorted values

- **Step 2:**
  - For i = 1 … N-1
    - Consider split $t_i = (v_i + v_{i+1}) / 2$
    - Compute classification error for threshold split $h_j(x) >= t_i$
  - Chose the t^* with the lowest classification error

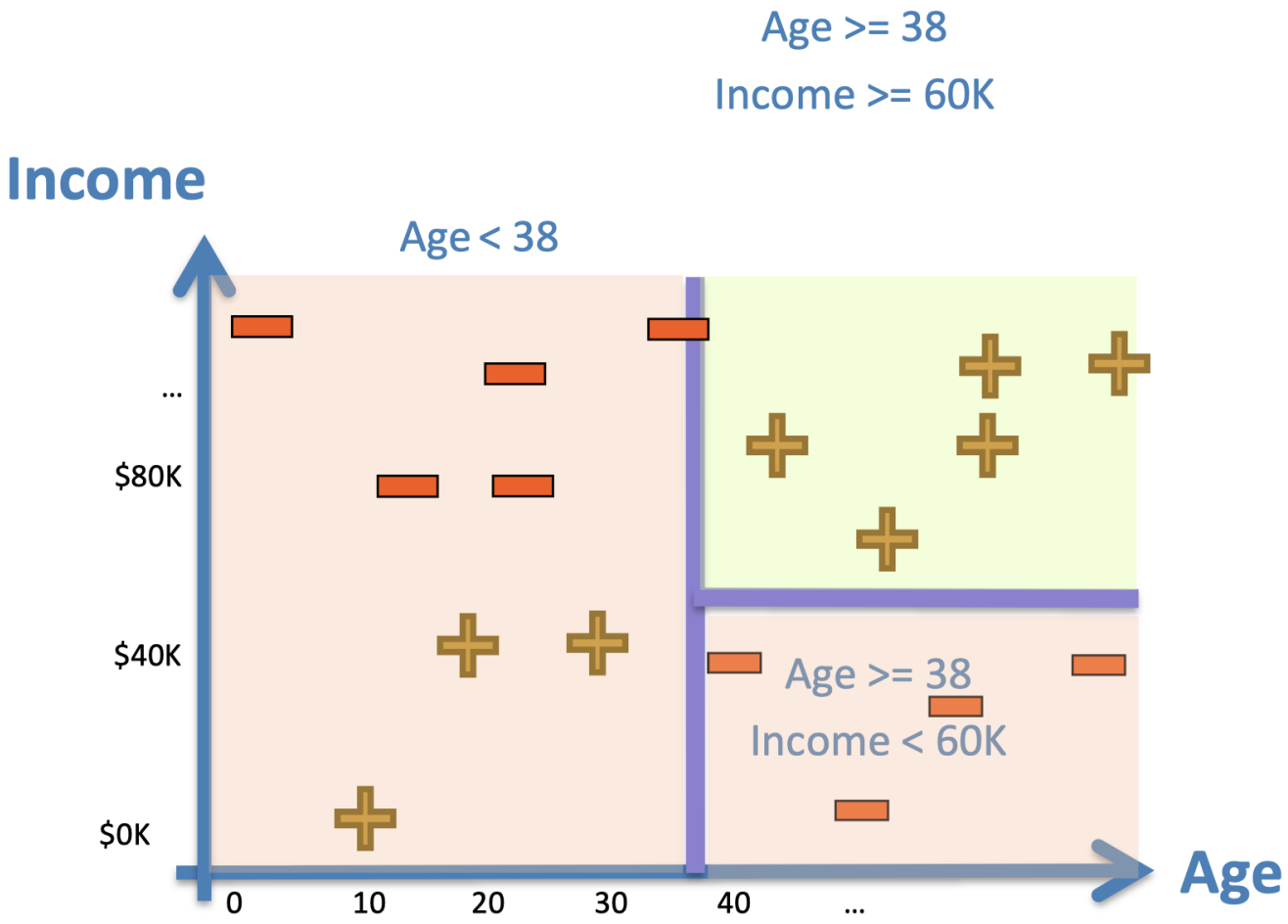
Visualizing the threshold split



Visualizing the threshold split
Split on Age >= 38

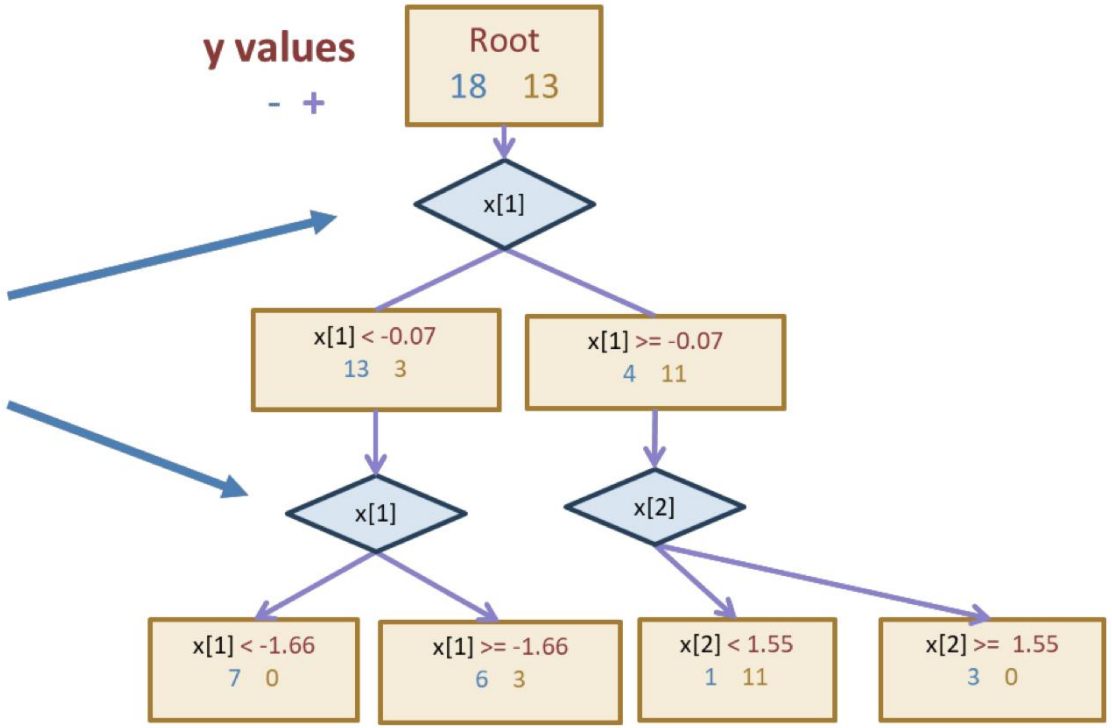


Visualizing the threshold split

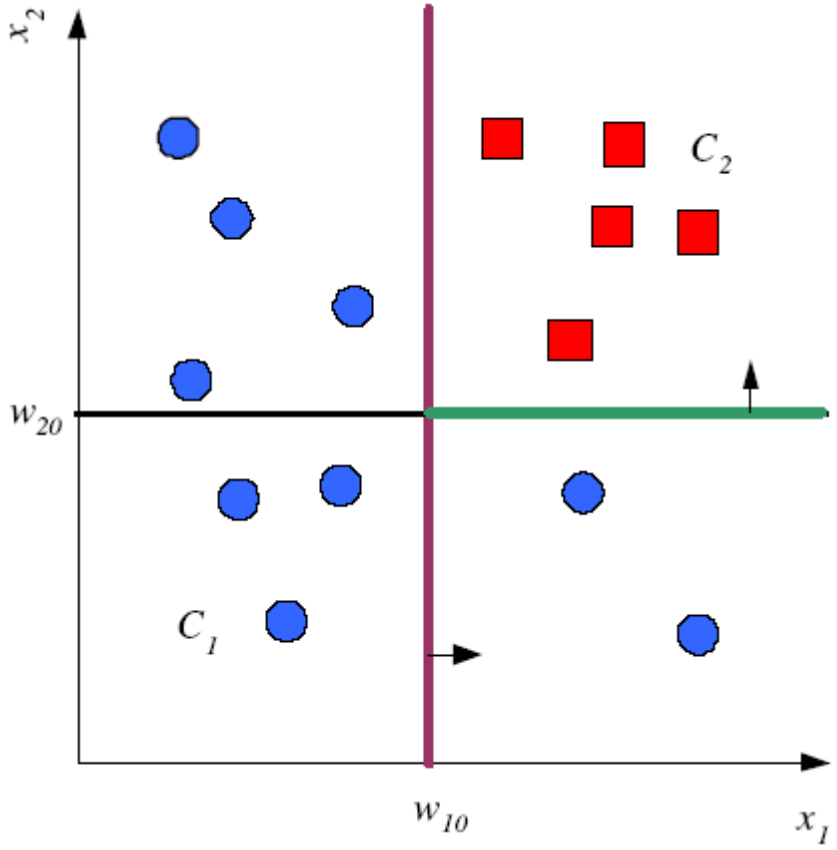


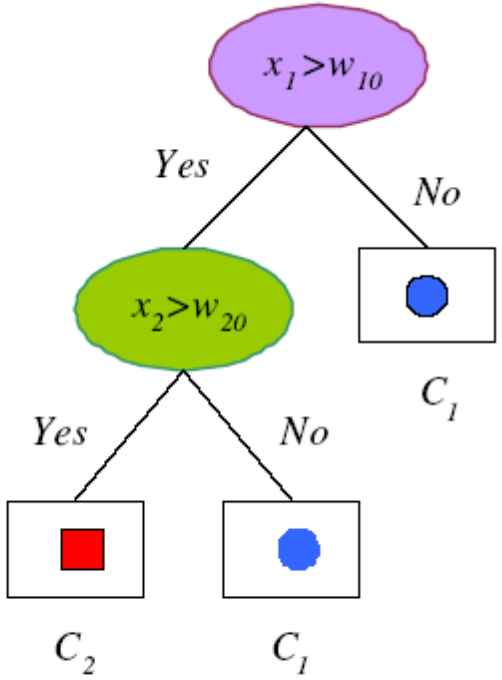
Threshold split

For threshold splits, same feature can be used multiple times

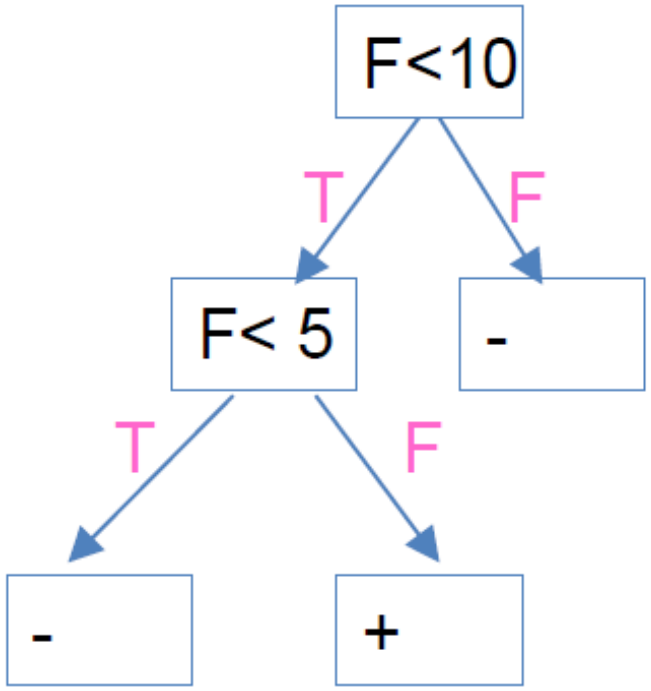


Decision Trees





Handling Real Valued Attributes



Handling Real Valued Attributes

- Temperature = 24.5^{0}C
- (Temperature **>** 20.0^{0}C) = {true, false}

Temperature	15^{0}C	18^{0}C	19^{0}C	22^{0}C	24^{0}C	27^{0}C
PlayTennis	No	No	Yes	Yes	Yes	No

Rule Representation

- *if* outlook = sunny AND humidity = normal

  OR

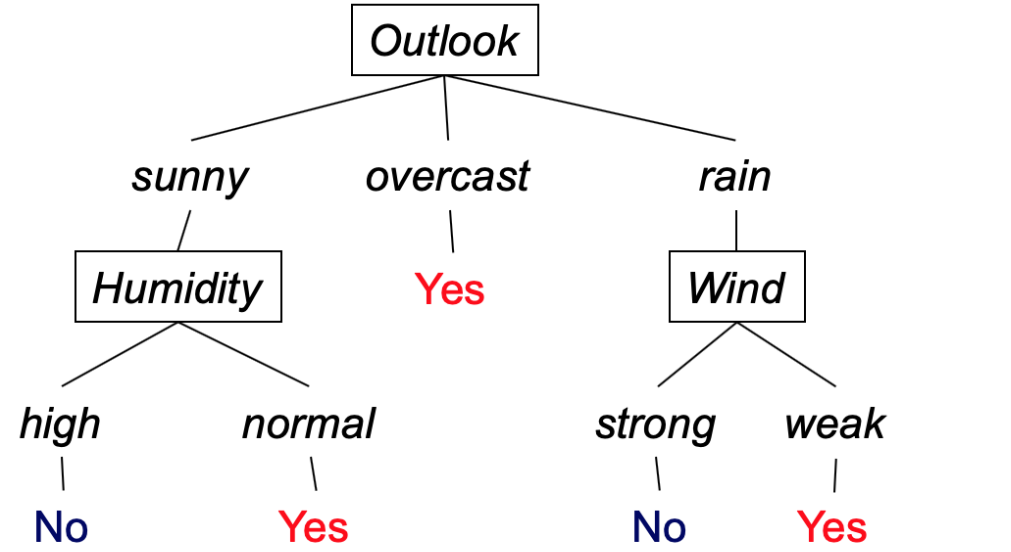
  if outlook = overcast

  OR

  if outlook = rain AND wind = weak

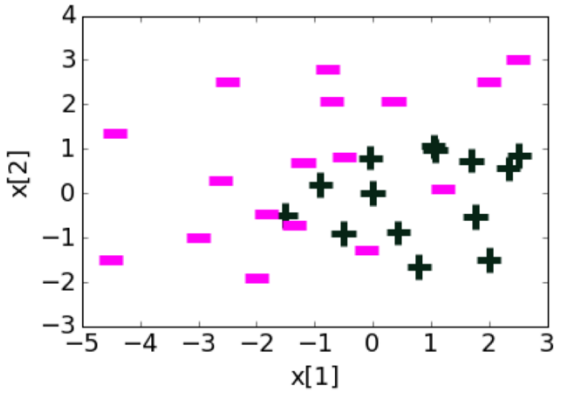
  then

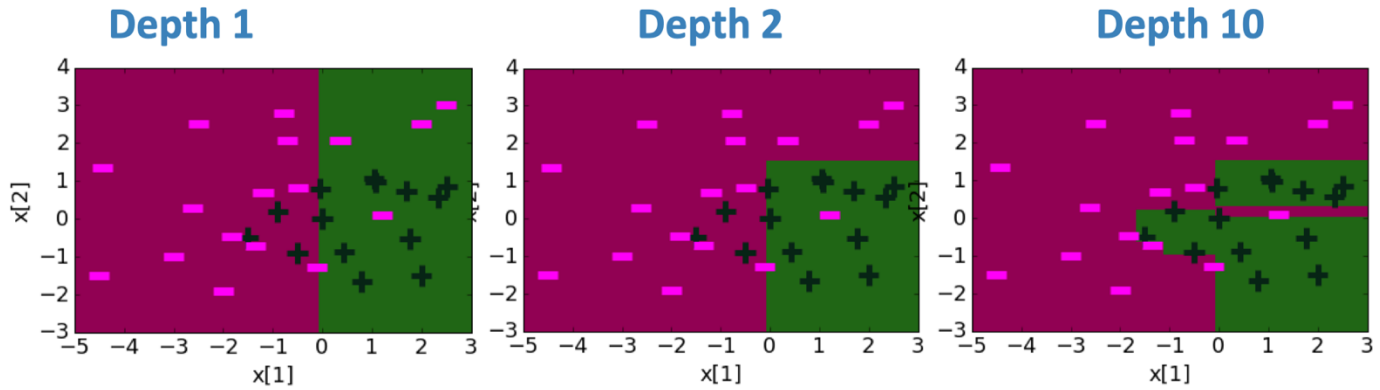
  playtennis



Decision Boundaries

- Decision boundaries can be complex!

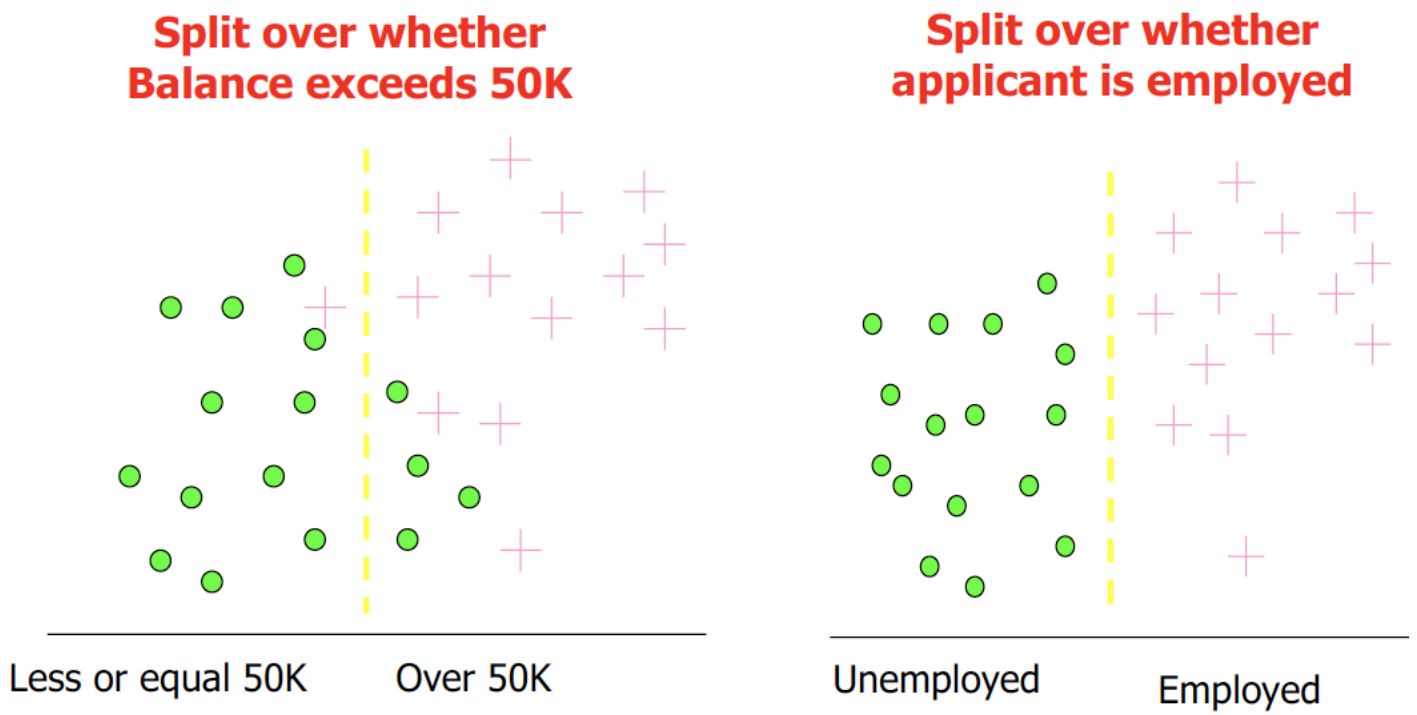




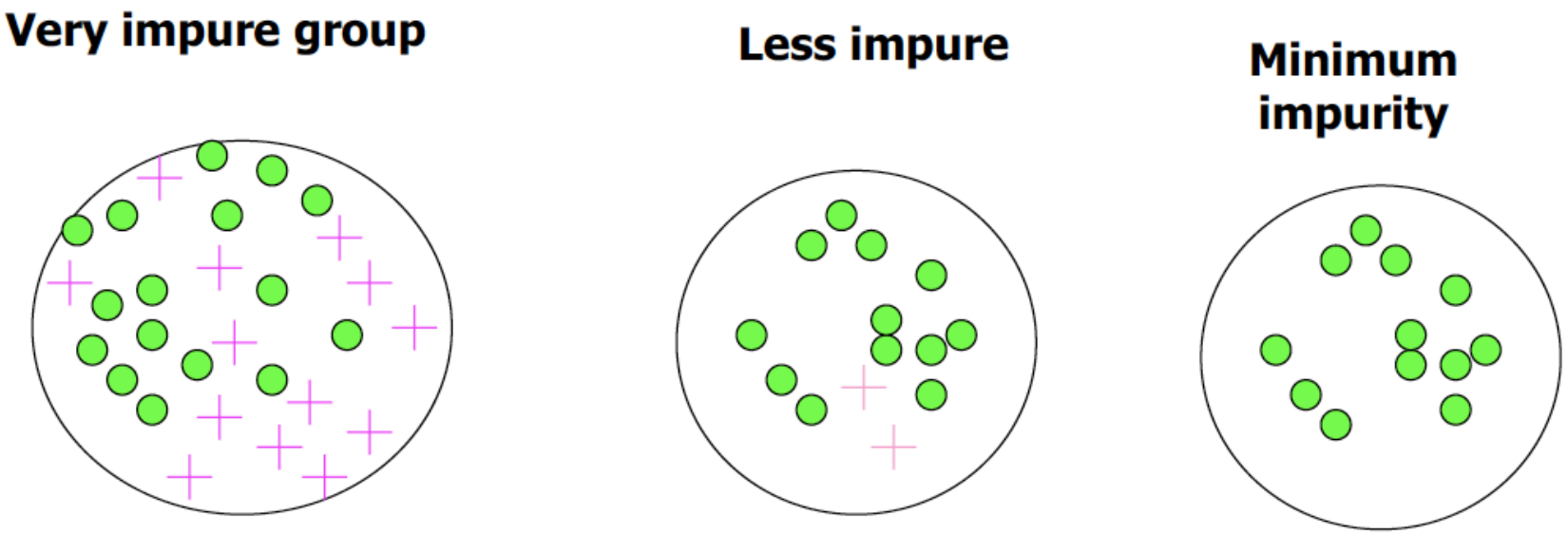
Which attribute should we use to split ?

- if we were able to find a small decision tree that explains data well, then good generalization guarantees.
  - **NP-hard** [Hyafil-Rivest'76]**:** unlikely to have a polynomial time algorithm

Which attribute should we use to split ?



Which attribute should we use to split ?



Decision Tree Learning

- Given a subset of data M (a node in a tree)
- For each remaining feature $h_i(x)$:
    1. Split data of M according to feature $h_i(x)$
    2. Compute classification error of split
- Chose feature $h^*(x)$ with lowest classification error

Which attribute should we use to split ?

ID3: The best attribute has highest information gain.

Pick best attribute to split at the root based on training data
Recurse on children that are impure

Information Gain: how well a given attribute separates the training examples according to the target classification

ID3 Algorithm (Ross Quinlan)

- ID3 (Iterative Dichotomiser 3)
  - **Iterative :** repeatedly
  - **Dichotomizes :** divides

  features into two or more groups at each step.

- ID3 : Top-Down Greedy Algorithm :
  - **top-down** : start building the tree from the top
  - **greedy :** select the best feature at the present moment to create a node.

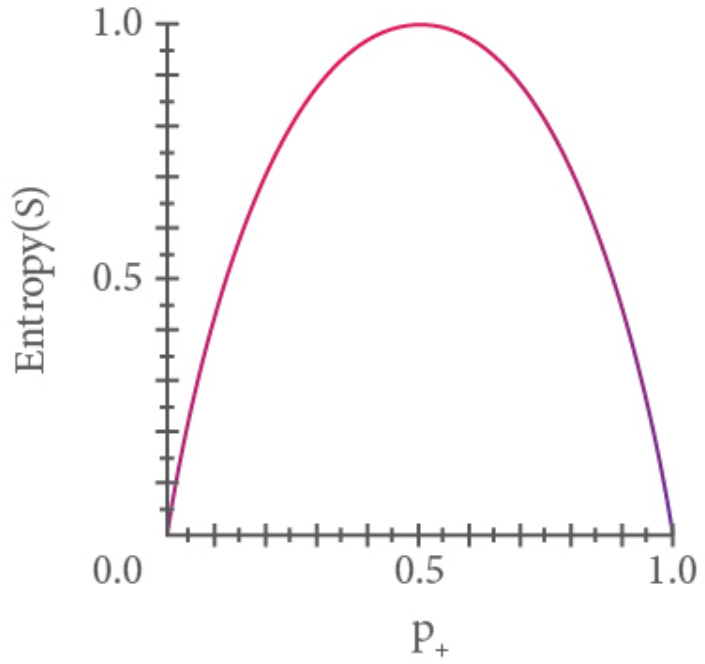
Top-down induction of decision trees

node = Root

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 descendent of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes.

Entropy



- **Entropy measures the impurity of S**

For binary classification:

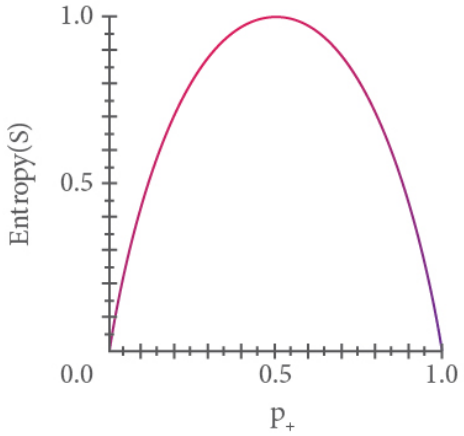
- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy is expected number of bits needed to encode class + or class -

$$\text{Entropy} = \sum_i -p_i \log_2 p_i$$

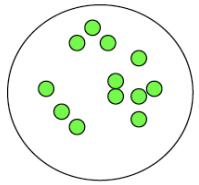
- If all samples are in one class then Entropy $= 0$

Entropy

Entropy = $\sum_i - p_i \log_2 p_i$



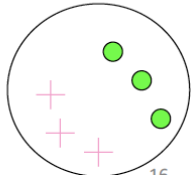
Minimum impurity



- If all samples are in one class then Entropy = 0

entropy = - 1 $\log_2 1 = 0$

Maximum impurity



- If 50/50 positive and negative then Entropy = 1

entropy = -0.5 $\log_2 0.5 - 0.5 \log_2 0.5 = 1$

Entropy

- Interpretation from information theory:
  - Entropy is expected number of bits needed to encode class (+ or -) of randomly drawn members of S
- If S is all positive, receiver knows label will be positive, don't need any bits.
- If S is 50/50 then need 1 bit.
- If S is 80/20, then in a long sequence of messages, can code with less than 1 bit on average (assigning shorter codes to positive examples and longer codes to negative examples).

Information Gain

Information Gain is **measure of expected reduction in entropy** resulting from splitting along an attribute A

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

entropy of original collection

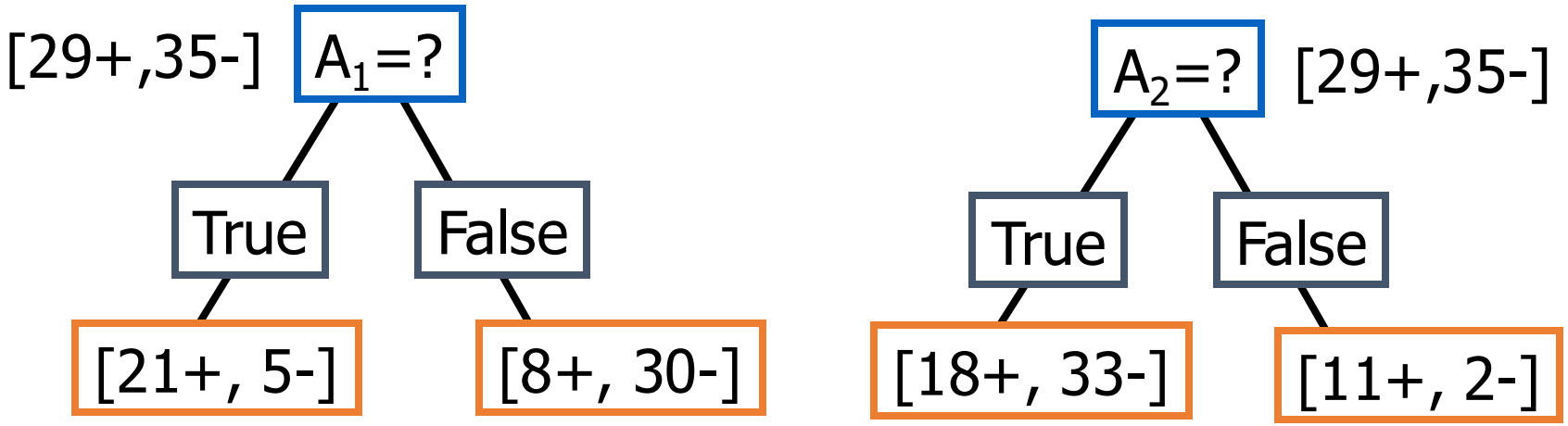
Expected entropy after S is partitioned using attribute A

sum of entropies of subsets S_v weighted by the fraction of examples that belong to S_v.

Information Gain

$$Gain(S,A)=Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$$

$$Entropy([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$$



Example : Good day for Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Information Gain of Wind

Values(wind)=weak, strong

S = [9+, 5-]

S_{weak} = [6+, 2-]

S_s = [3+, 3-]

Gain(S, wind)

= Entropy(S) - $\sum$ ($|S_v|$ / |S|) Entropy(S_v)

v ∈ {weak, s}

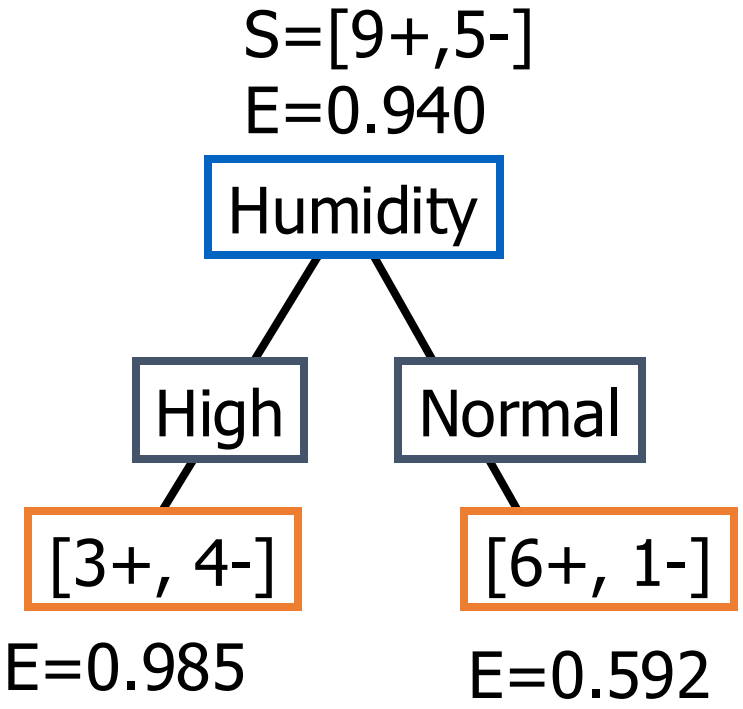
= Entropy(S) - 8/14 Entropy(S_{weak})
- 6/14 Entropy(S_s)

= 0.940 - (8/14) 0.811 - (6/14) 1.00

= .048

Day	Wind	Tennis?
d1	weak	n
d2	s	n
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	s	n
d7	s	yes
d8	weak	n
d9	weak	yes
d10	weak	yes
d11	s	yes
d12	s	yes
d13	weak	yes
d14	s	n

Information Gain of Humidity



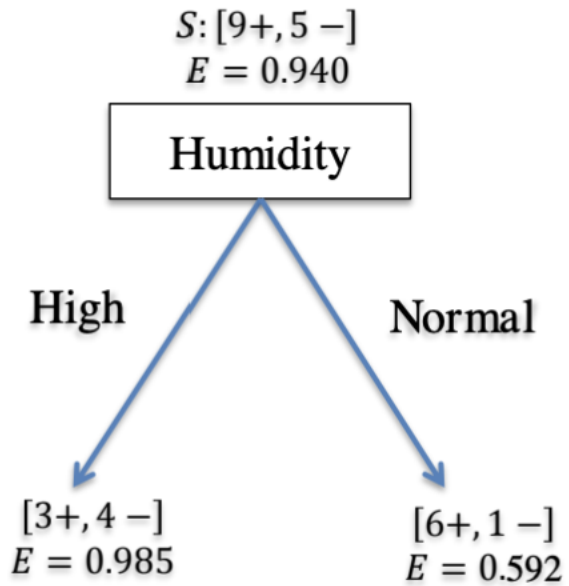
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

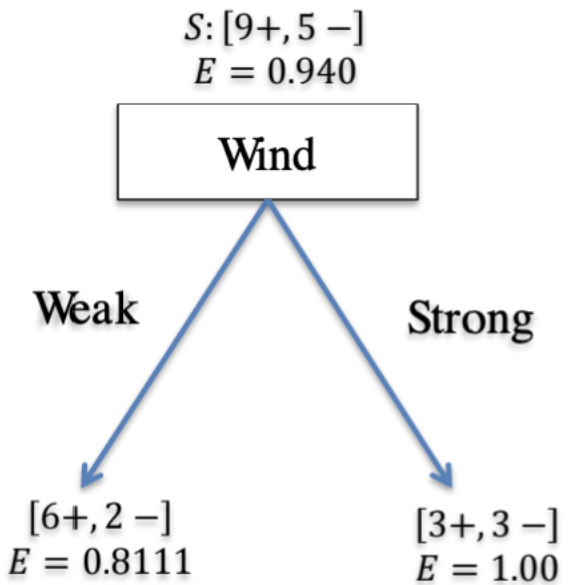
Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Selecting the Next Attribute

Which attribute is the best classifier?

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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





$$Gain(S, Humidity) = .940 - \left(\frac{7}{14}\right).985 - \left(\frac{7}{14}\right).592 = .151$$

$$Gain(S, Wind) = .940 - \left(\frac{8}{14}\right).811 - \left(\frac{6}{14}\right)1.0 = .048$$

Selecting the best Attribute

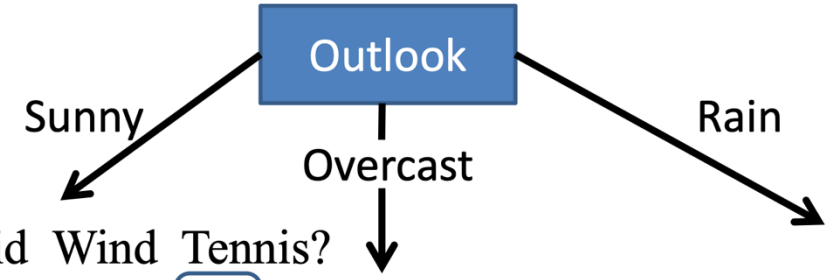
Information Gain:

- Gain(S,Outlook) =**0.247**
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

Selecting the next attribute

Recurse

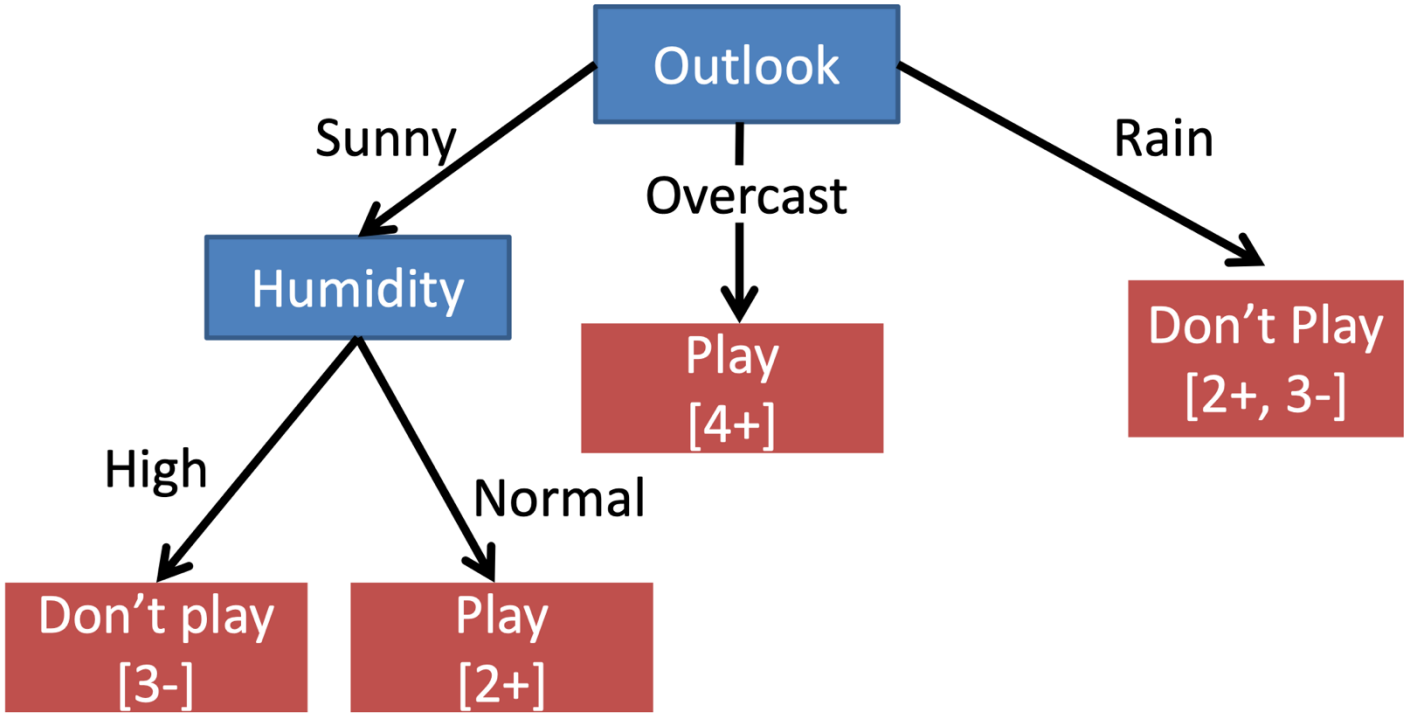
Good day for tennis?



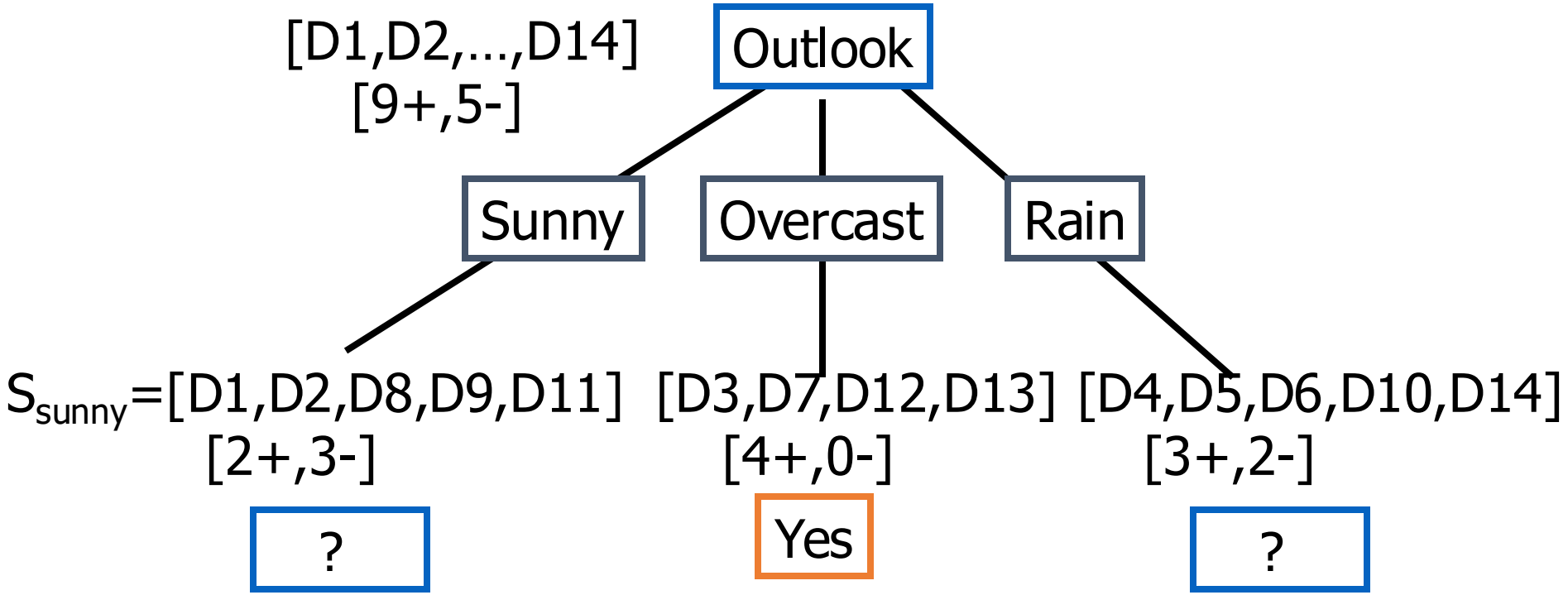
Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	n	weak	yes
d11	m	n	s	yes

Selecting the next attribute

Good day for tennis?



Selecting the next attribute

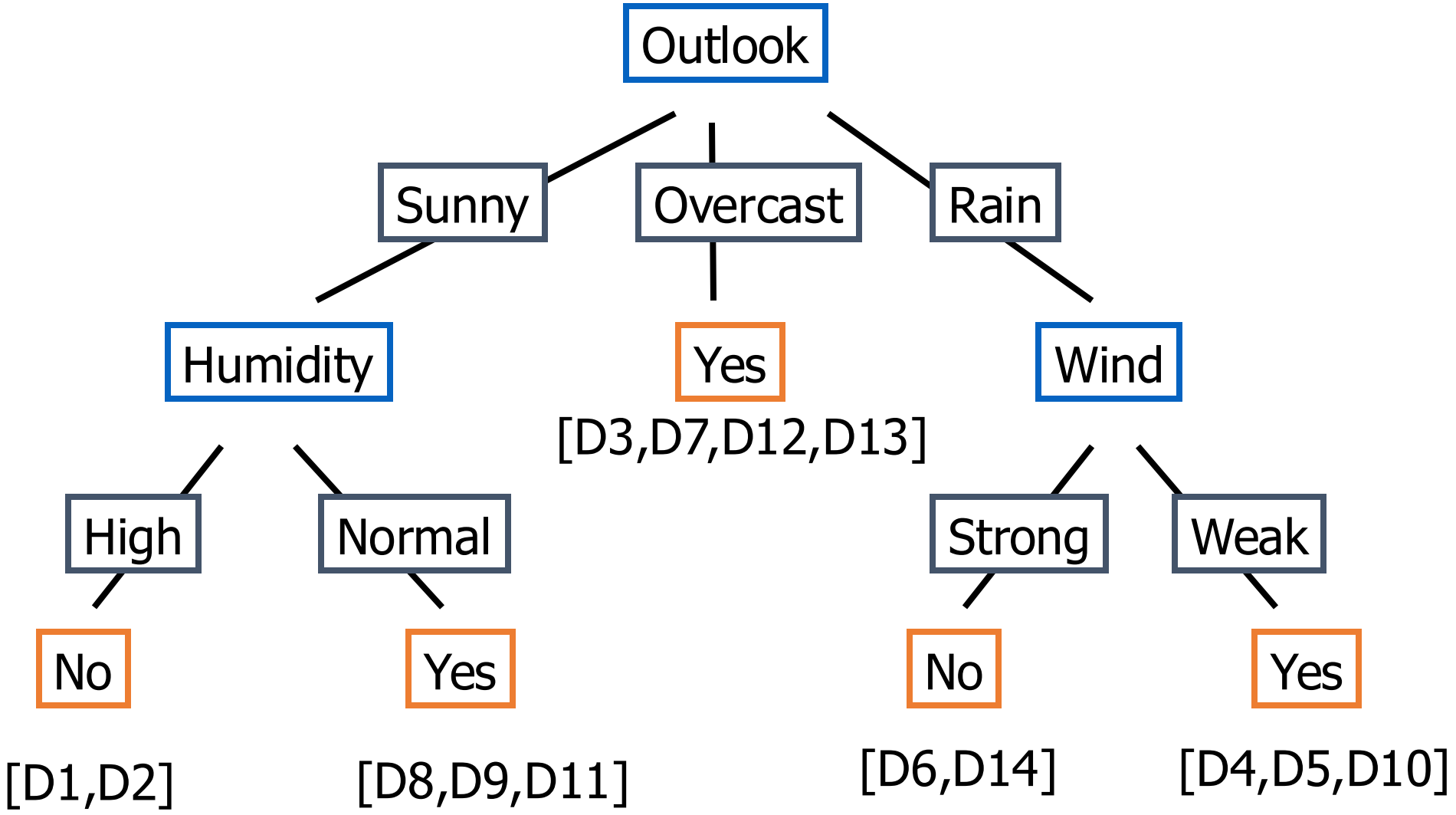


Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = **0.970**

Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny} , Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

Decision Tree



Overfitting

- Deep decision trees are prone to overfitting
  - Decision boundaries are interpretable but not stable
  - Small change in the dataset leads to big difference in the outcome
- Overcoming Overfitting:
  - Early stopping
    - Fixed length depth
    - Stop if error does not considerably decrease
  - Pruning
    - Grow full length trees
    - Prune nodes to balance a complexity penalty

Overfitting

- Many kinds of “noise” can occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
  - The instance was labeled incorrectly (+ instead of -)

- Also, some attributes are irrelevant to the decision-making process
  - e.g., color of a die is irrelevant to its outcome

Overfitting in Decision Trees

- Irrelevant attributes can result in *overfitting* the training example data
  - If hypothesis space has many dimensions (large number of attributes), we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features

- If we have too little training data, even a reasonable hypothesis space will 'overfit'

Overfitting with noisy data

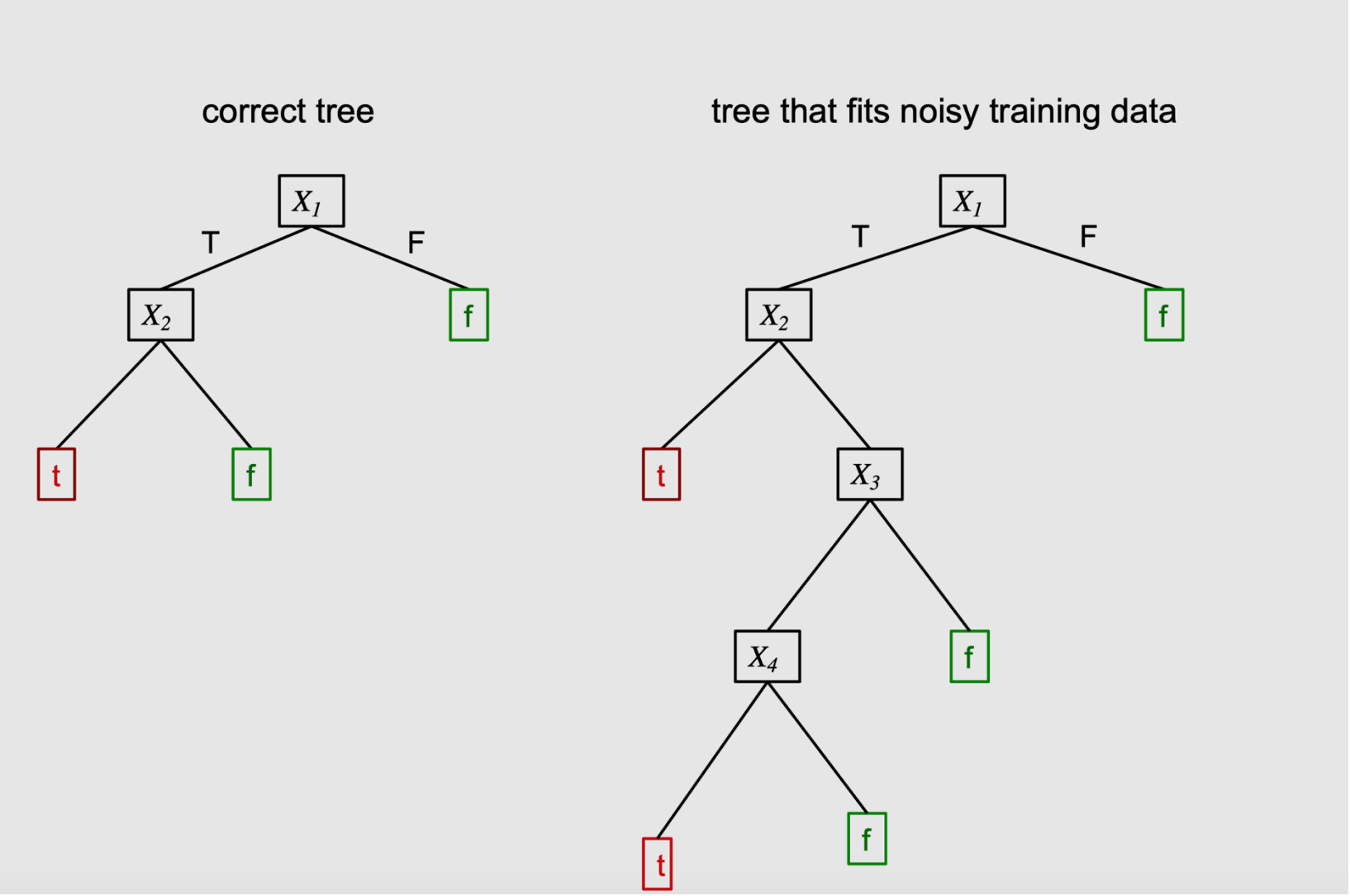
suppose

- the target concept is $Y = X_1 \wedge X_2$
- there is noise in some feature values
- we're given the following training set

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	f	f	t	…	t
t	f	t	t	f	…	t
t	f	f	t	f	…	f
t	f	t	f	f	…	f
f	t	t	f	t	…	f

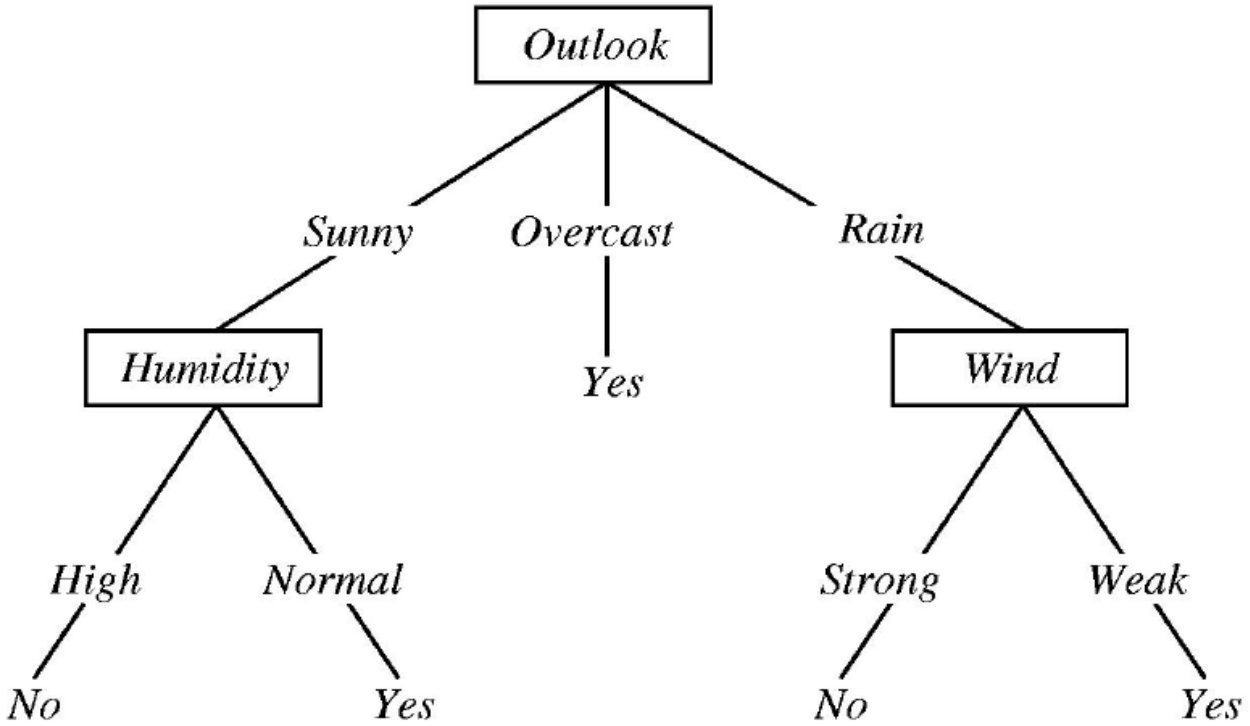
noisy value

Overfitting with noisy data



Overfitting in Decision Trees

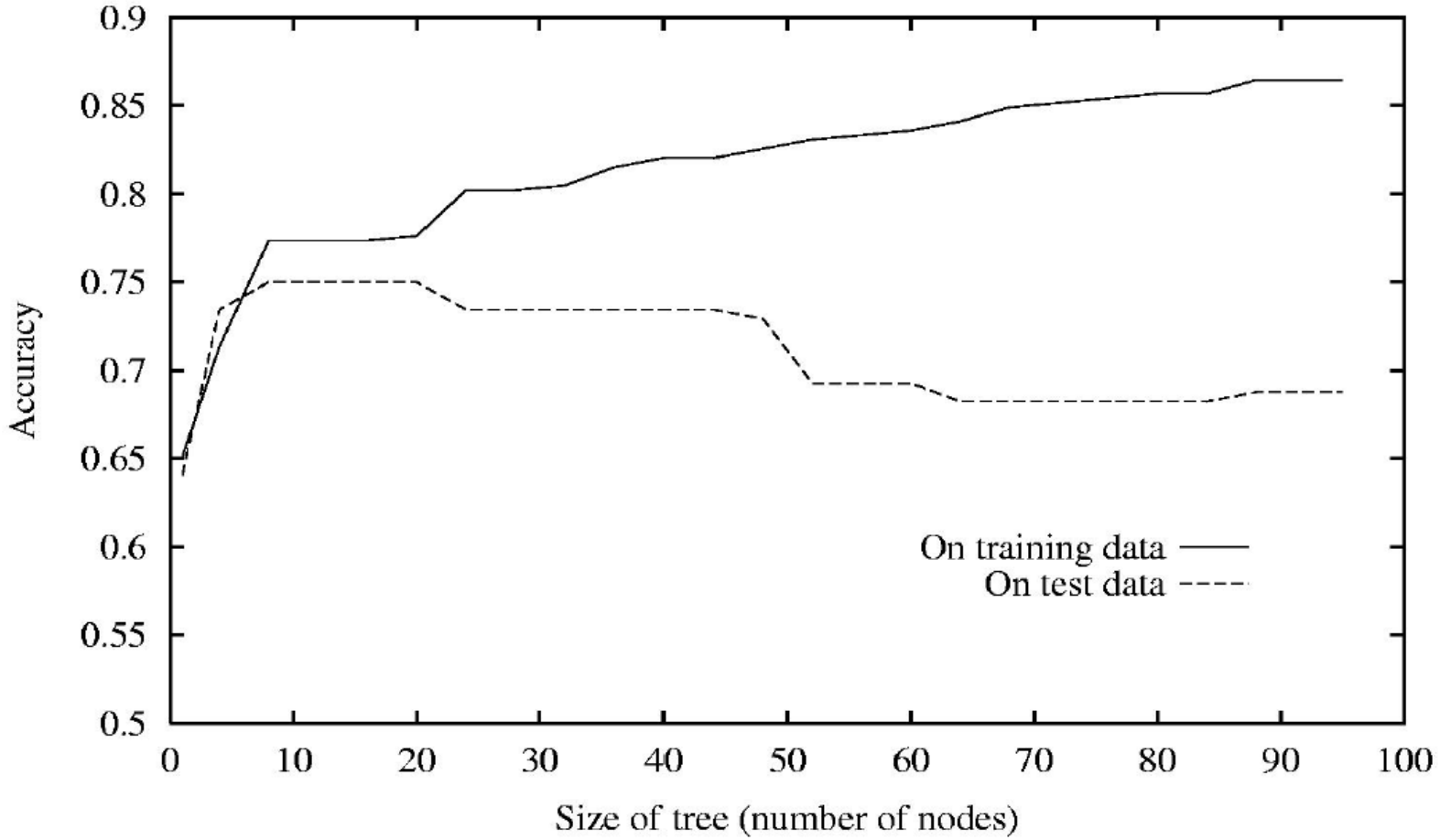
Consider adding a **noisy** training example to the following tree:



What would be the effect of adding:

<outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No> ?

Overfitting in Decision Trees



Overfitting in Decision Trees

- DT is *overfit* when exists another DT' and
  - DT has ***smaller*** error on training examples, but
  - DT has ***bigger*** error on test examples
- Causes of overfitting
  - Noisy data, or
  - Training set is too small
- Solutions
  - Reduced error pruning
  - Early stopping
  - Rule post pruning

Reduced-Error Pruning

Split training data further into *training* and *validation* sets

Grow tree based on *training 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 node that most improves *validation set* accuracy

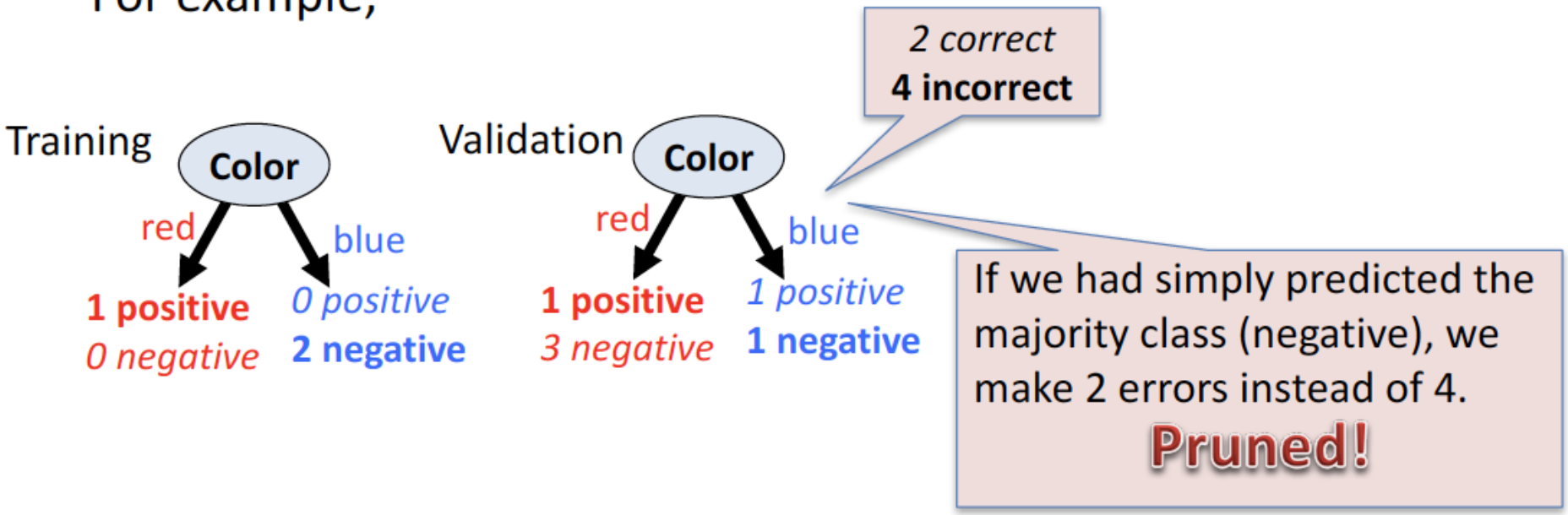
Reduced-Error Pruning

- Split data into train and validation set

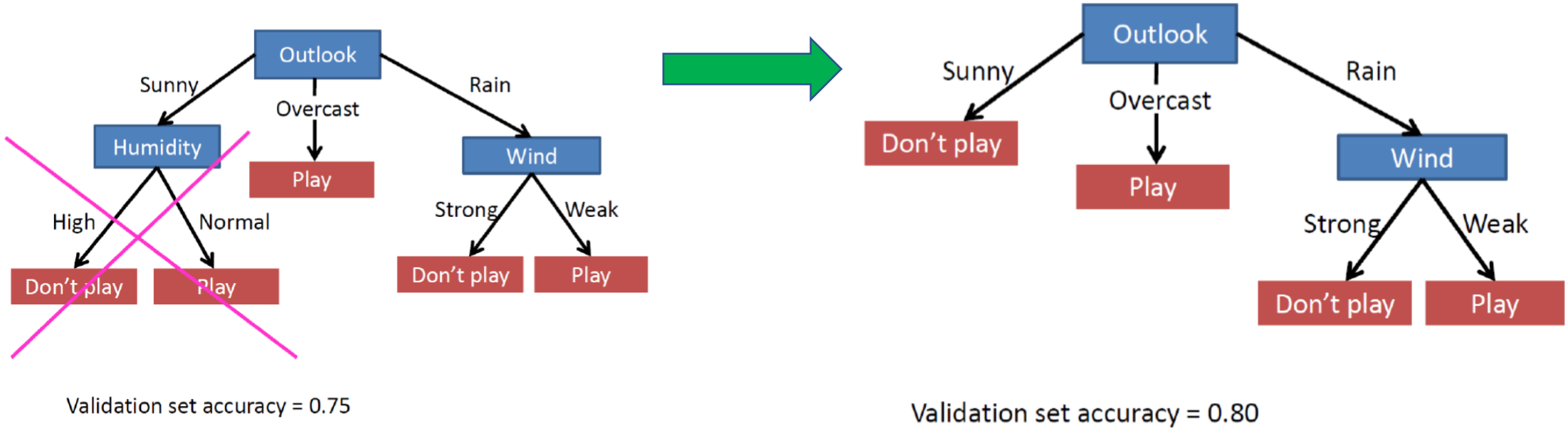
- Repeat until pruning is harmful
  - Remove each subtree and replace it with majority class and evaluate on validation set
  - Remove subtree that leads to largest gain in accuracy

Reduced-Error Pruning

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.
- For example,



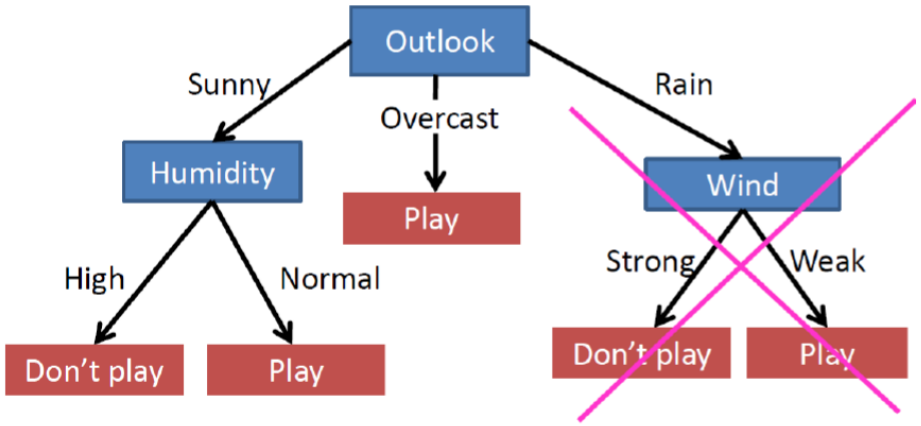
Reduced-Error Pruning - Example

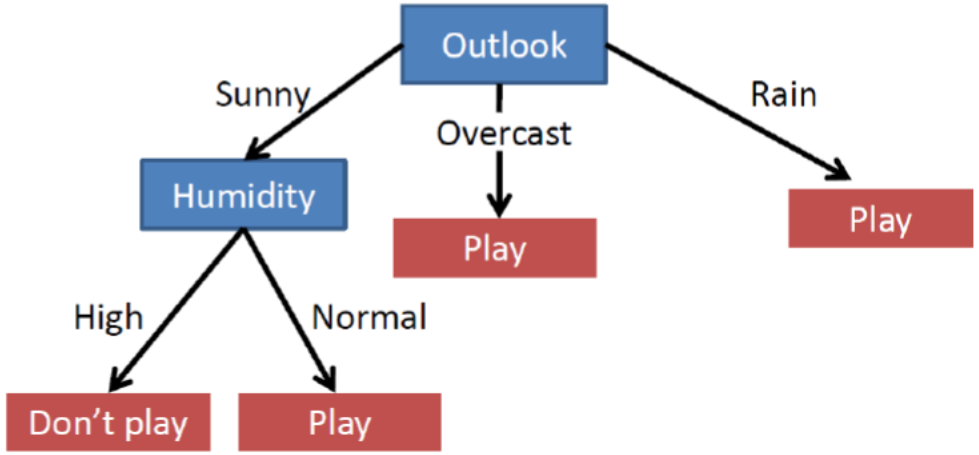


Validation set accuracy = 0.75

Validation set accuracy = 0.80

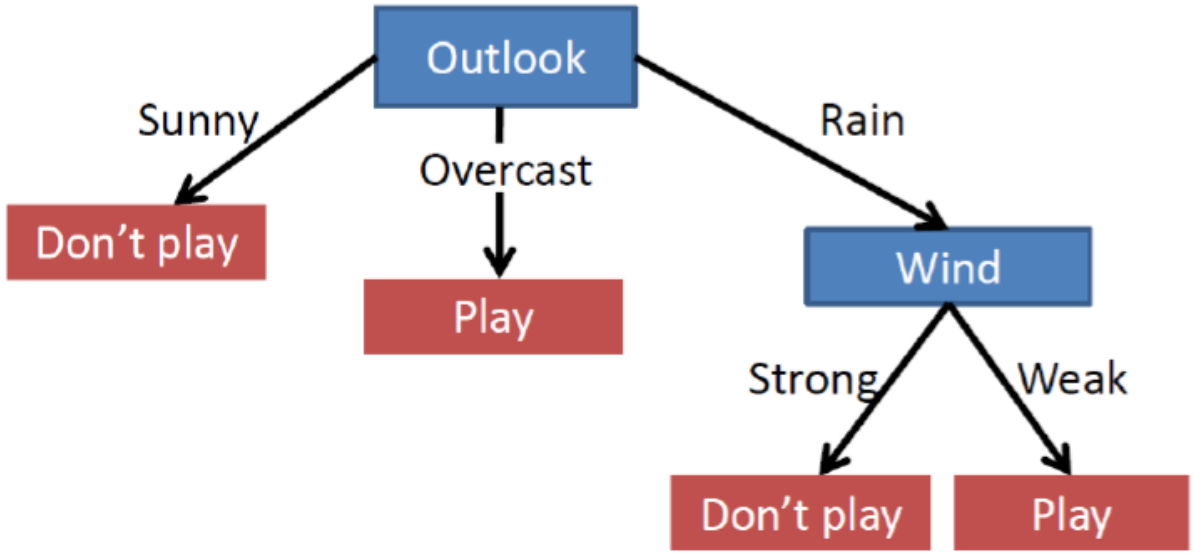
Reduced-Error Pruning - Example





Validation set accuracy = 0.70

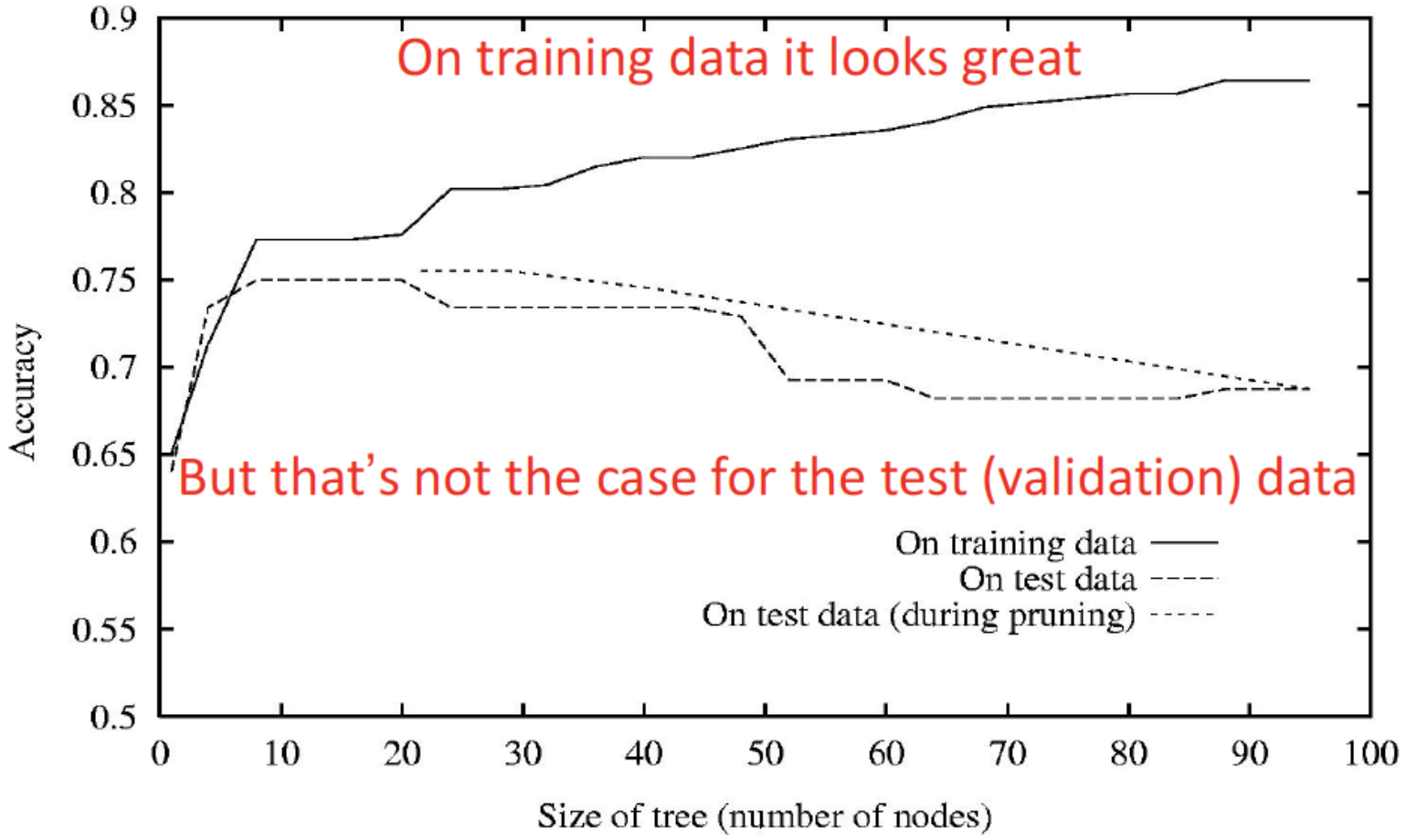
Reduced-Error Pruning - Example



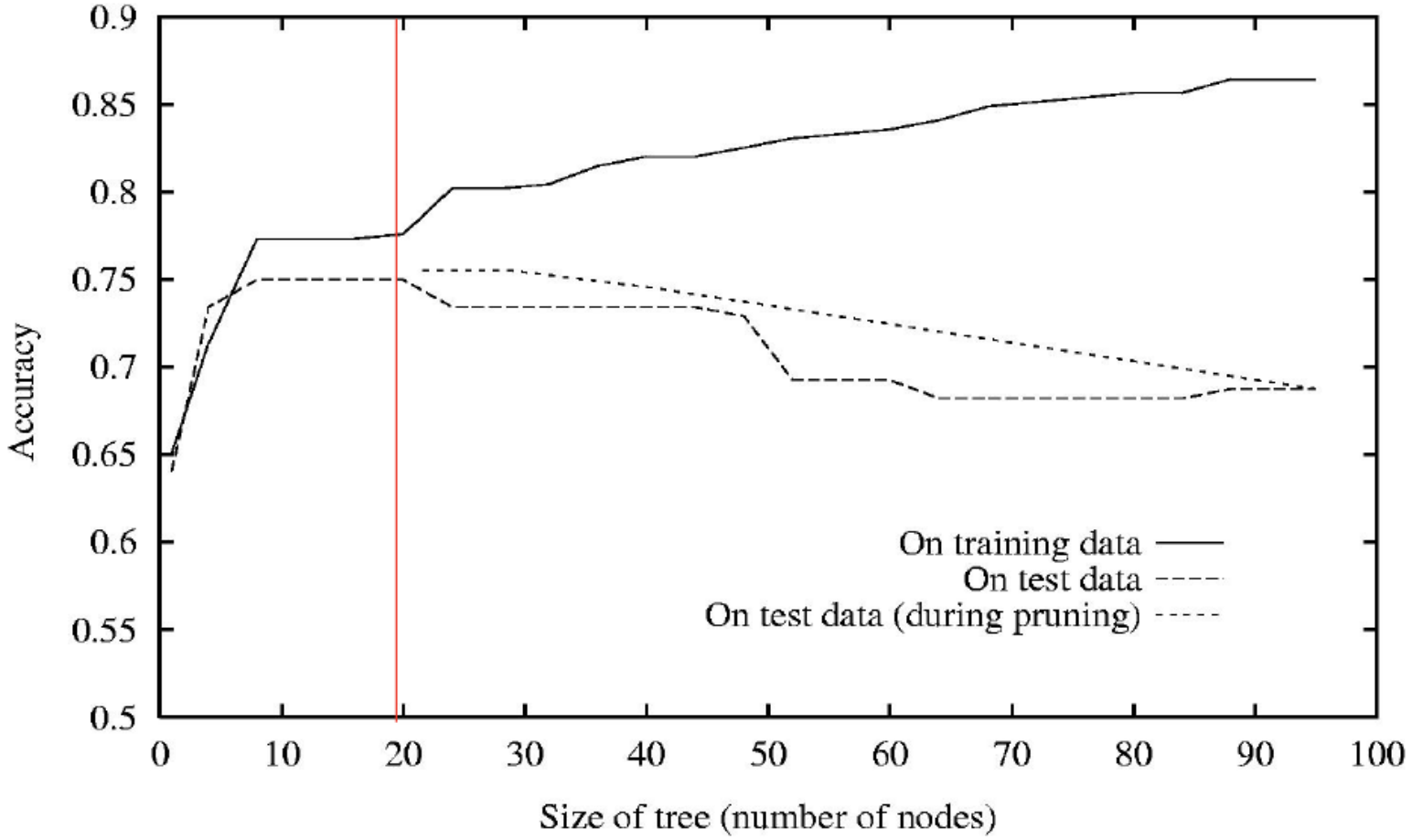
Validation set accuracy = 0.80

Use this as final tree

Effect of Reduced-Error Pruning

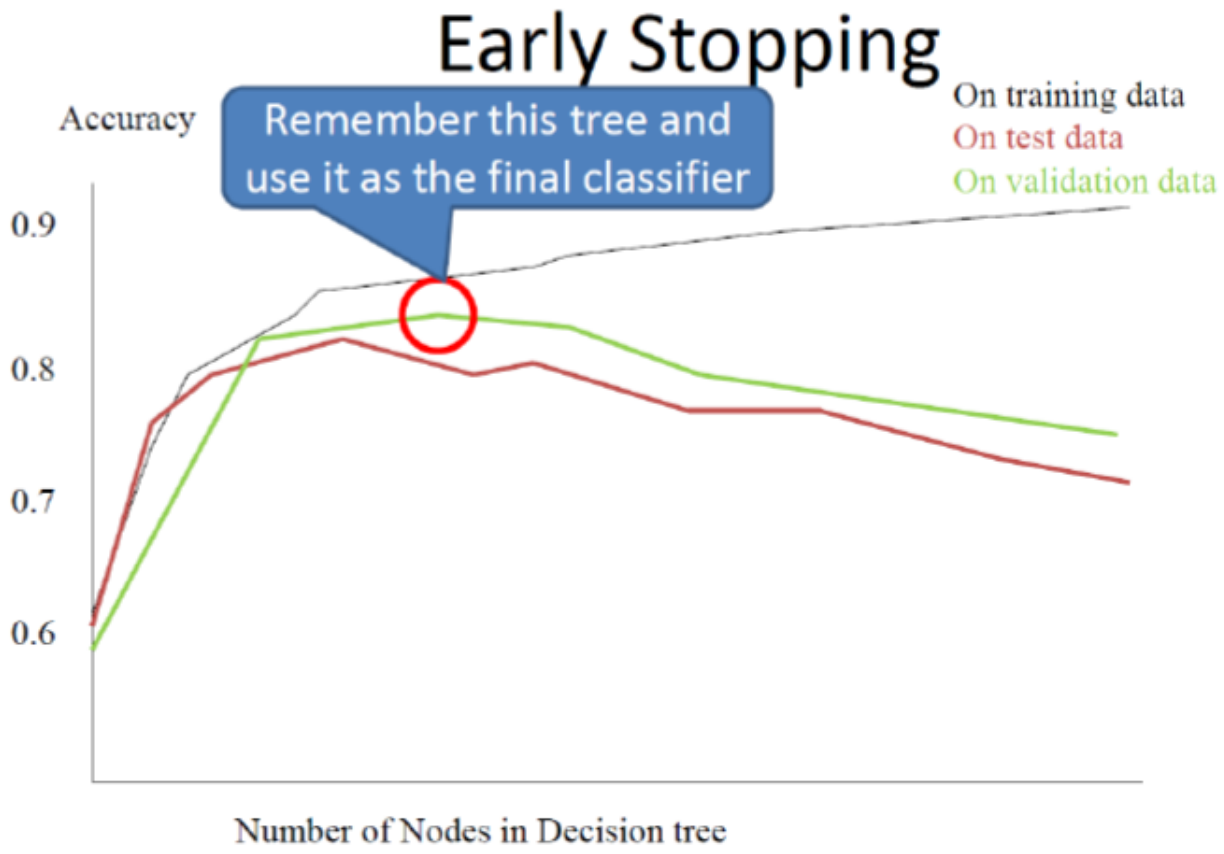


Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

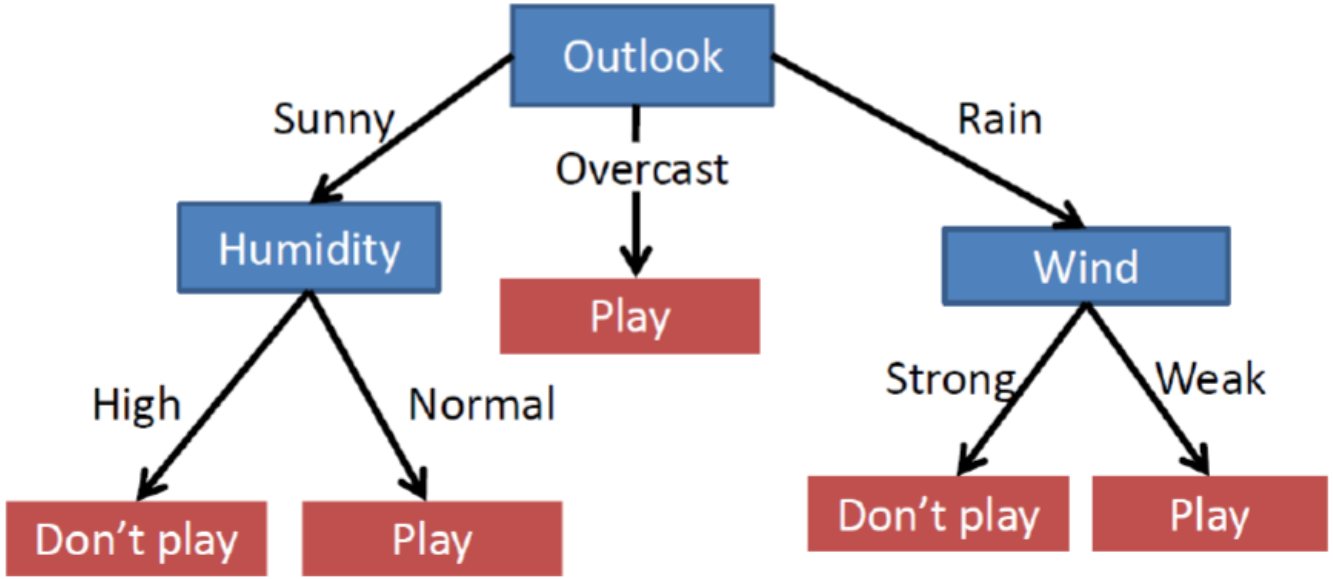
Early Stopping



Rule Post-Pruning

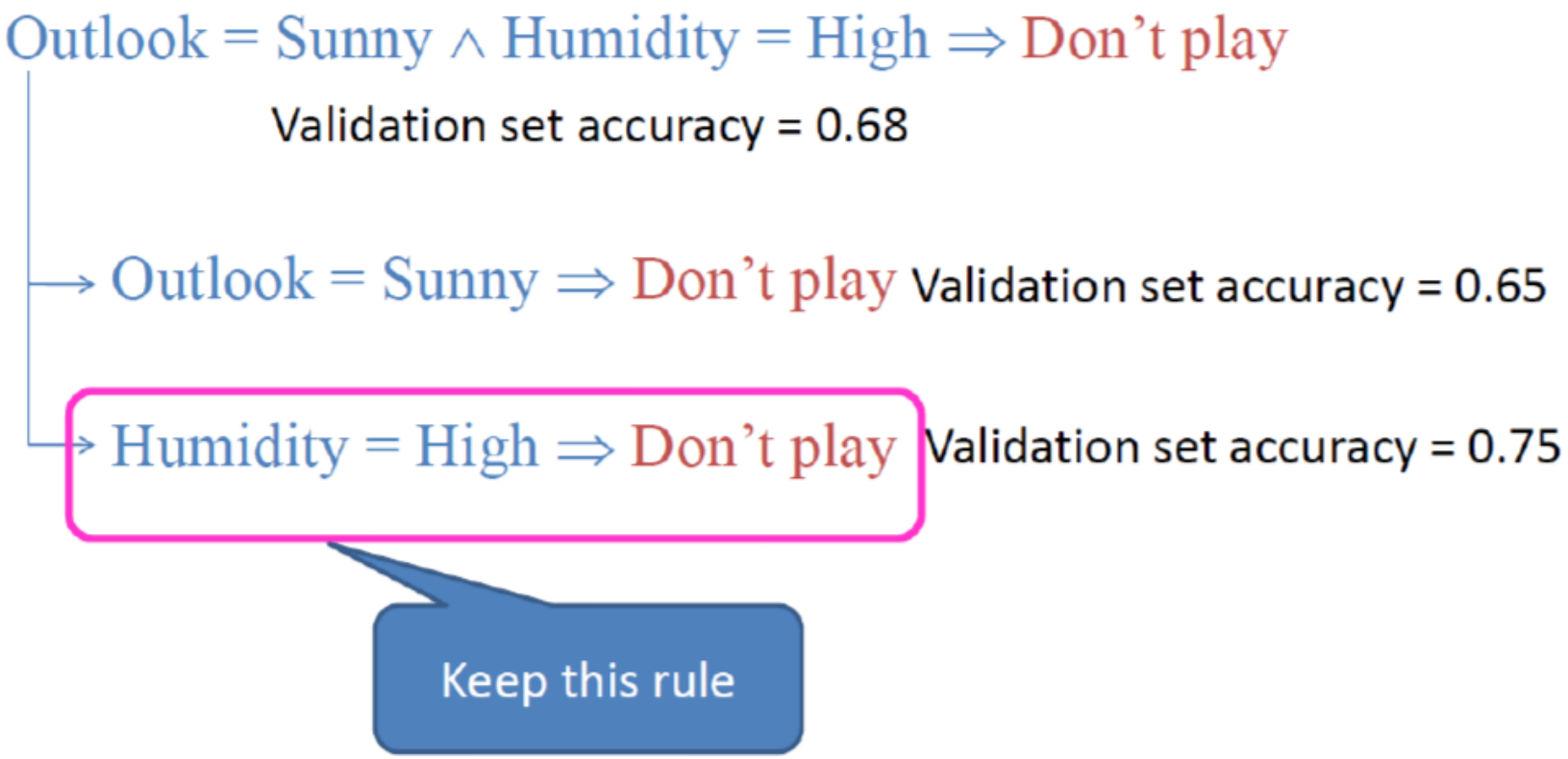
- Split data into train and validation set
- Prune each rule independently
  - Remove each pre-condition and evaluate accuracy
  - Pick pre-condition that leads to largest improvement in accuracy
- Note: ways to do this using training data and statistical tests

Conversion to Rule



$$\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{High} \Rightarrow \text{Don't play}$$
$$\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal} \Rightarrow \text{Play}$$
$$\text{Outlook} = \text{Overcast} \Rightarrow \text{Play}$$

Rule Post-Pruning - Example



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)

Summary

- Widely used in practice
- Strengths include
  - Fast and simple to implement
  - Can convert to rules
  - Handles noisy data
- Weaknesses include
  - Univariate splits/partitioning using only one attribute at a time --- limits types of possible trees
  - Large decision trees may be hard to understand
  - Requires fixed-length feature vectors
  - Non-incremental (i.e., batch method)