INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Decision Tree Learning

Dr. Animesh Chaturvedi

Assistant Professor: **IIIT Dharwad**

Young Researcher: **Heidelberg Laureate Forum** and **Pingala Interaction in Computing**

Young Scientist: **Lindau Nobel Laureate Meetings**

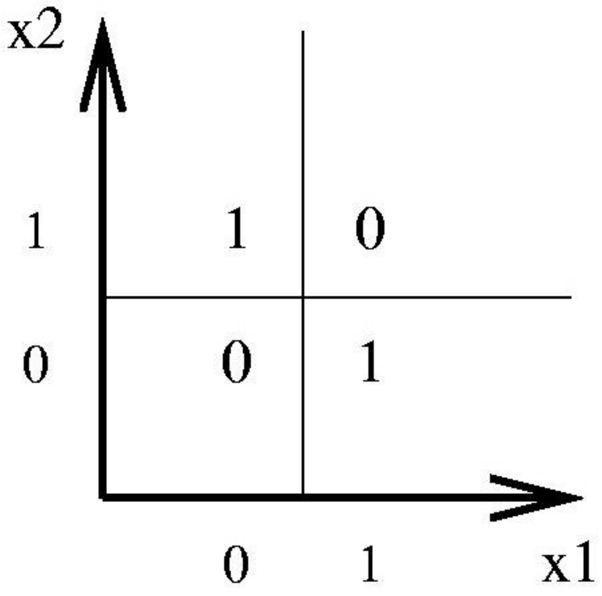
Postdoc: **King's College London** & **The Alan Turing Institute**

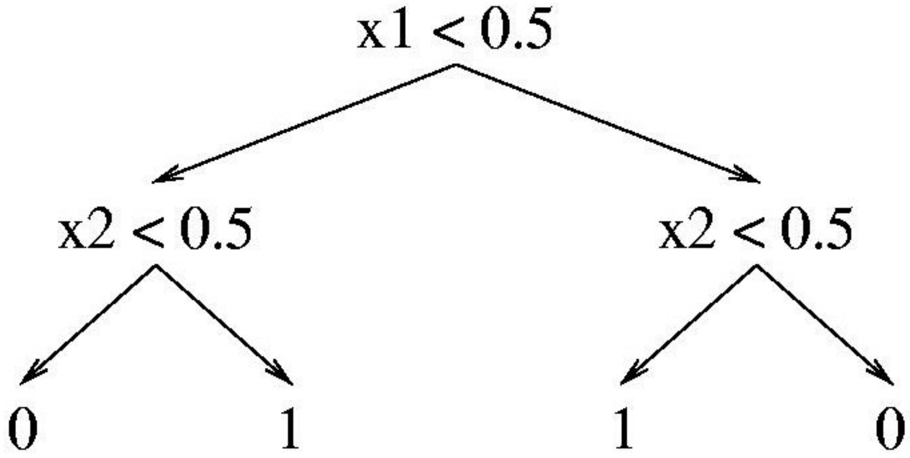
PhD: **IIT Indore** MTech: **IIITDM Jabalpur**

Decision Trees

- Convenient Representation
  - Developed with learning in mind
  - Deterministic
  - Comprehensible output
- Expressive
  - Equivalent to propositional Disjunctive Normal Form (DNF)
  - Handles discrete and continuous parameters
- Simple learning algorithm
  - Handles noise well
  - Classify
  - Constructive (build DT by adding nodes)

Boolean function can represents Decision tree

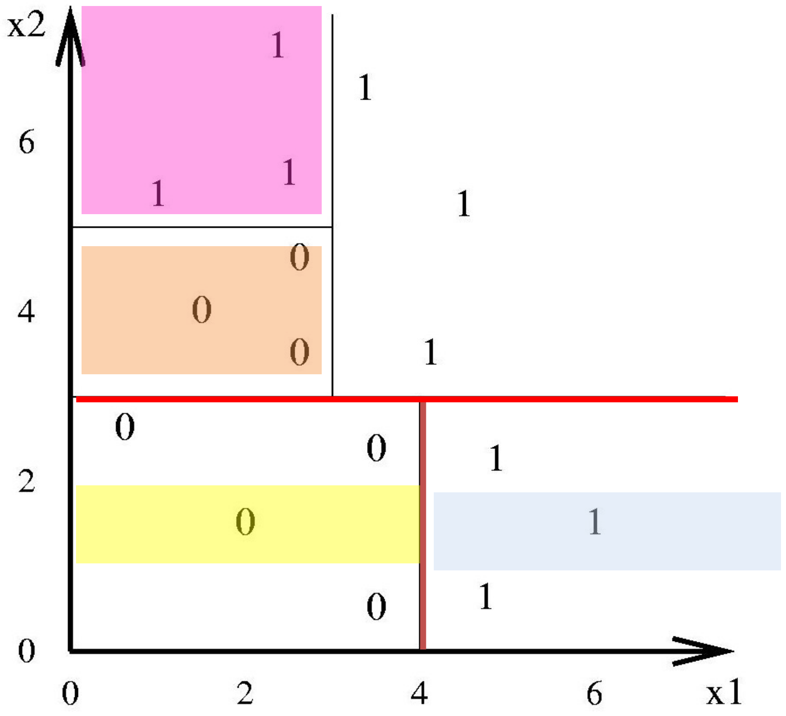


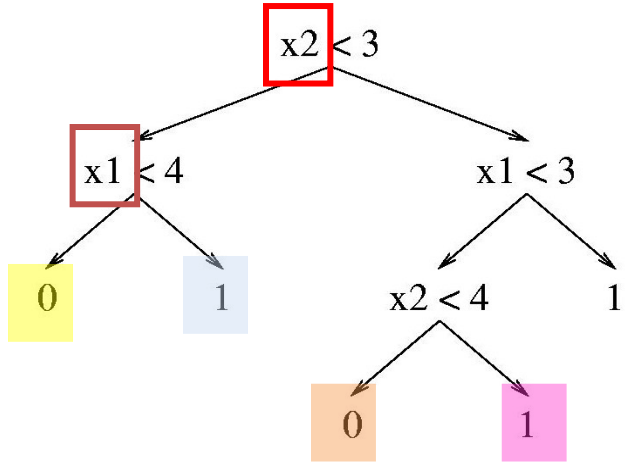


The tree will in the worst case require exponentially many nodes, however.

Decision tree to make Decision Boundaries

- Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





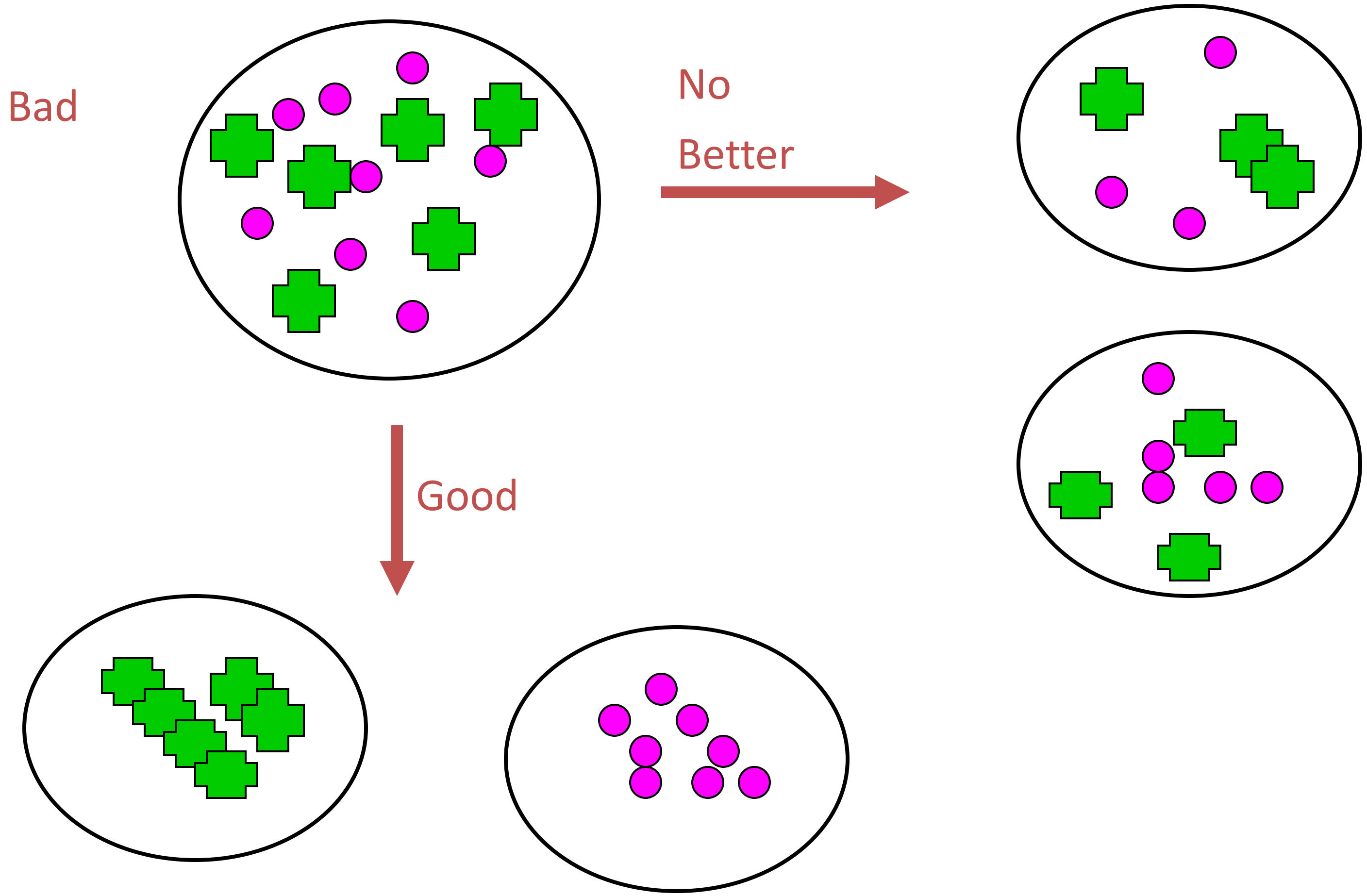
Depth of Decision tree and Boolean functions

Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)$)
- **etc.**

Disorder is bad Homogeneity is good



Which attribute should we use to split?

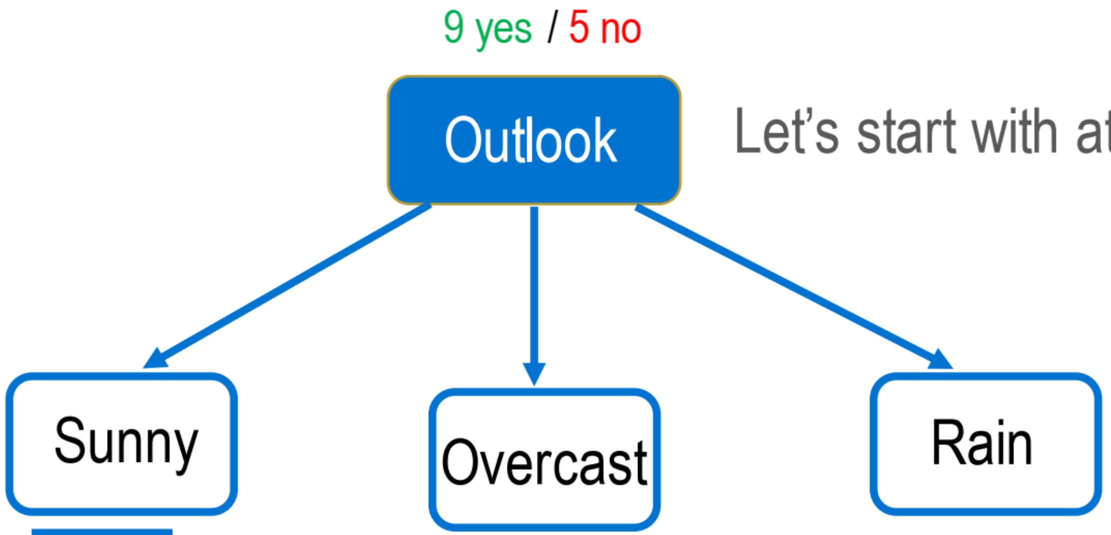
Example: “Good day for tennis”

- Attributes of instances
  - Outlook = {rainy (r), overcast (o), sunny (s)}
  - Temperature = {cool (c), medium (m), hot (h)}
  - Humidity = {normal (n), high (h)}
  - Wind = {weak (w), strong (s)}
- Class value
  - Play Tennis? = {don’t play (n), play (y)}
- Feature = attribute with one value
  - E.g., outlook = ***sunny***
- Sample instance
  - outlook=***sunny***, temp=***hot***, humidity=***high***, wind=***weak***

Experience: "Good day for tennis"

Day	Outlook	Temp	Humid	Wind	PlayTennis?
d1	s	h	h	w	n
d2	s	h	h	s	n
d3	o	h	h	w	y
d4	r	m	h	w	y
d5	r	c	n	w	y
d6	r	c	n	s	n
d7	o	c	n	s	y
d8	s	m	h	w	n
d9	s	c	n	w	y
d10	r	m	n	w	y
d11	s	m	n	s	y
d12	o	m	h	s	y
d13	o	h	n	w	y
d14	r	m	h	s	n

Split training data



Let's start with attribute Outlook

Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

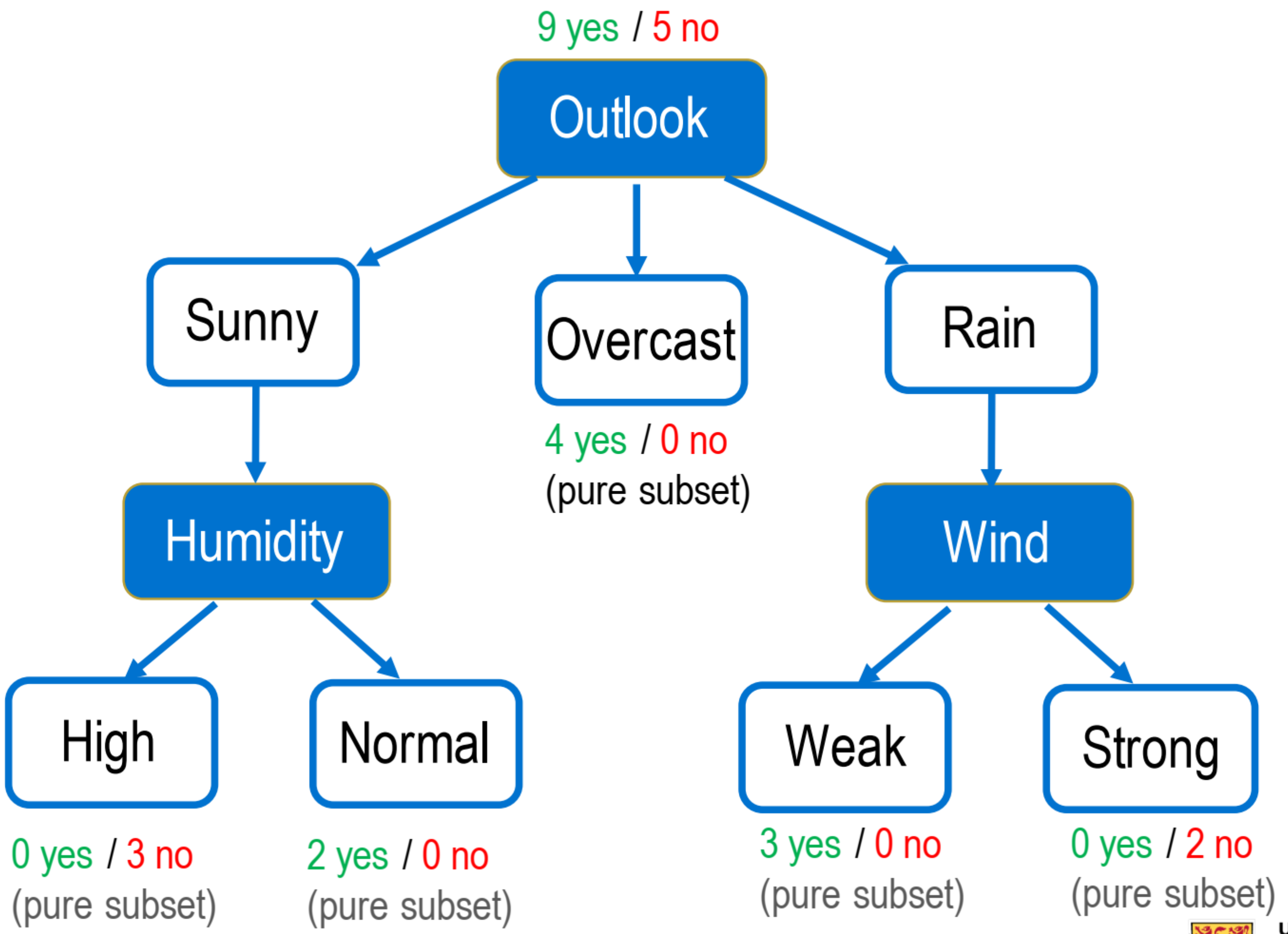
2 yes / 3 no
(split further based on Humid)

4 yes / 0 no
(pure subset)

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

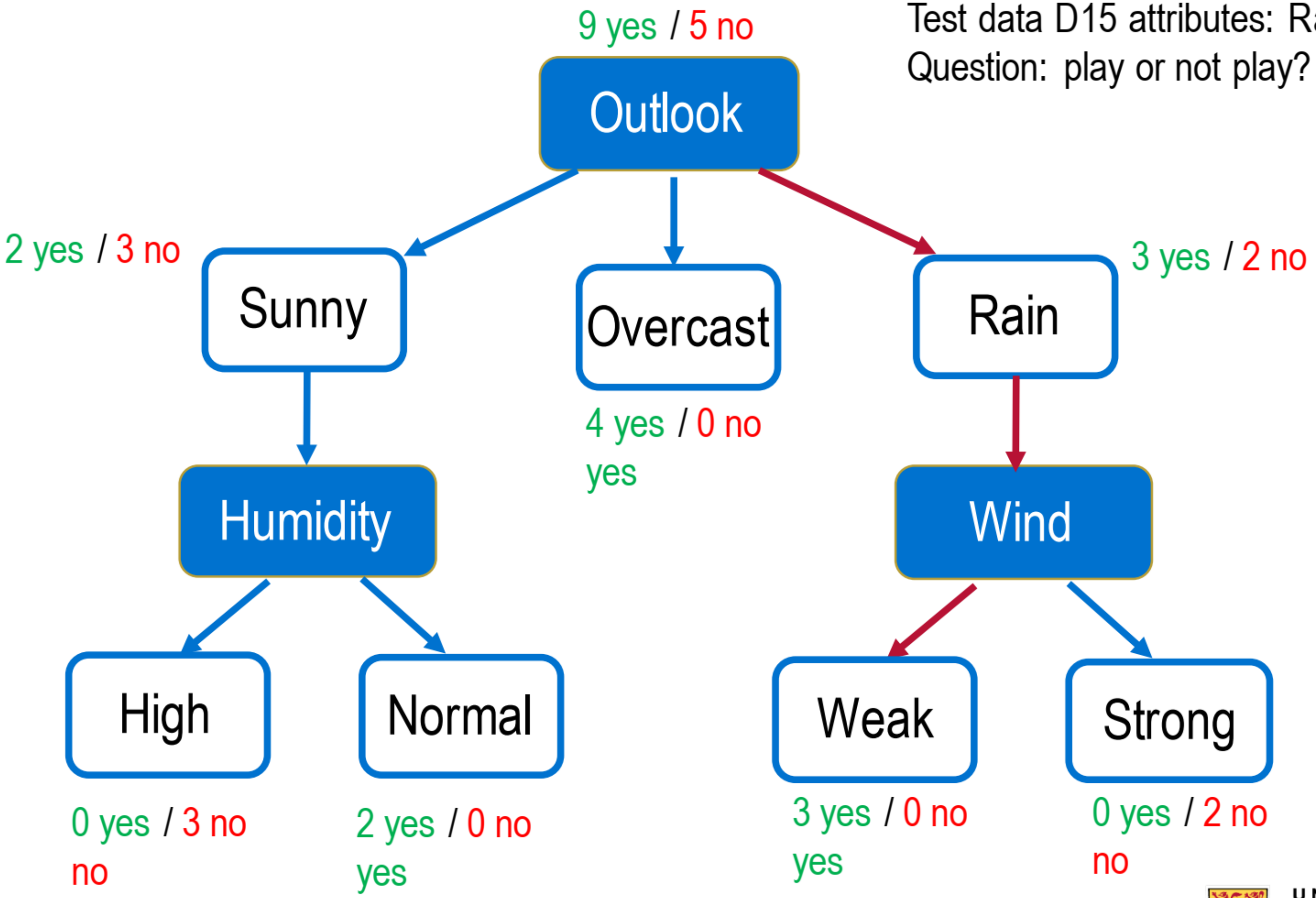
3 yes / 2 no
(split further based on Wind)

Split training data



Split training data

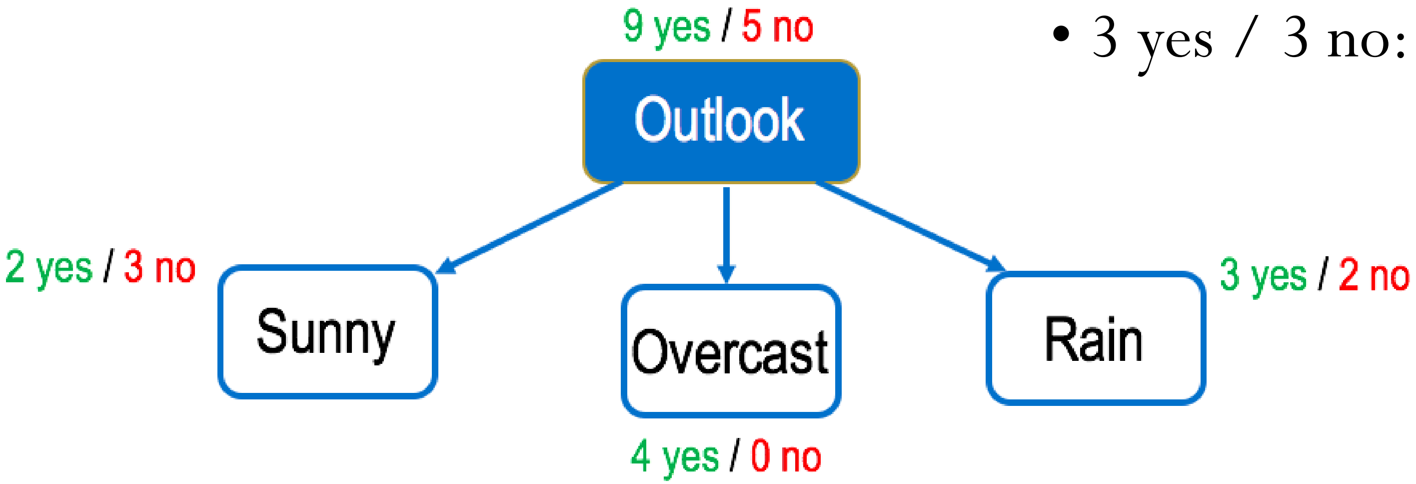
Test data D15 attributes: Rain High Weak
Question: play or not play? yes

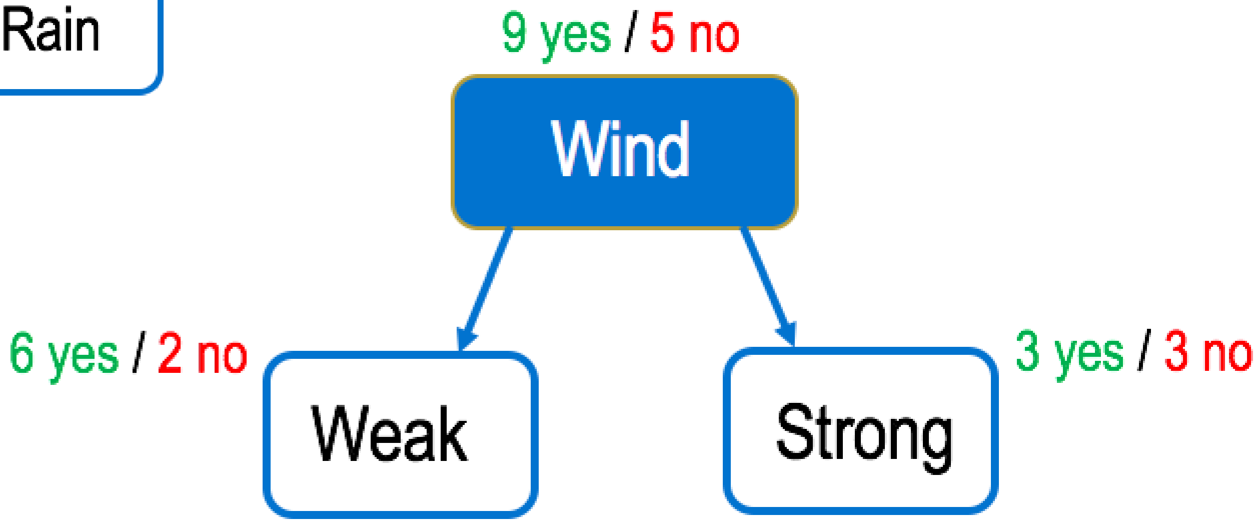


How to select an attribute?

We hope that uncertainty can be reduced after the split

- 4 yes / 0 no: 100 % certain about yes
- 3 yes / 3 no: 50% certain about yes





Decision Tree General Algorithm

BuildTree(TraingData)

    Split(TrainingData)

Split(D)

    If (all points in D are of the same class)

        Then Return

    For each attribute A

        Evaluate splits on attribute A

    Use best split to partition D into D1, D2

    Split (D1)

    Split (D2)

How to learn decision trees

- Constructing optimal binary decision trees is an NP Complete problem
  - Optimal tree is one which minimizes the expected number of tests required to identify the unknown object
  - NP-complete: belongs to both NP and NP-hard; easy to verify a solution to NP-complete, but hard to find a solution
- Often resort to heuristic algorithms
  - Build an empty decision tree à split à recurse (choosing a good attribute for splitting is important)
  - Some examples: ID3, C4.5, CART

ID3 (Iterative Dichotomiser 3) algorithm

- ID3 (node, {training data}) # Generate a DT

  1. Pick an attribute (A) with the maximum information gain for the considered training data

  2. For each value of A, create new child node

  3. Split training data to child nodes

  4. Check subset for each child node

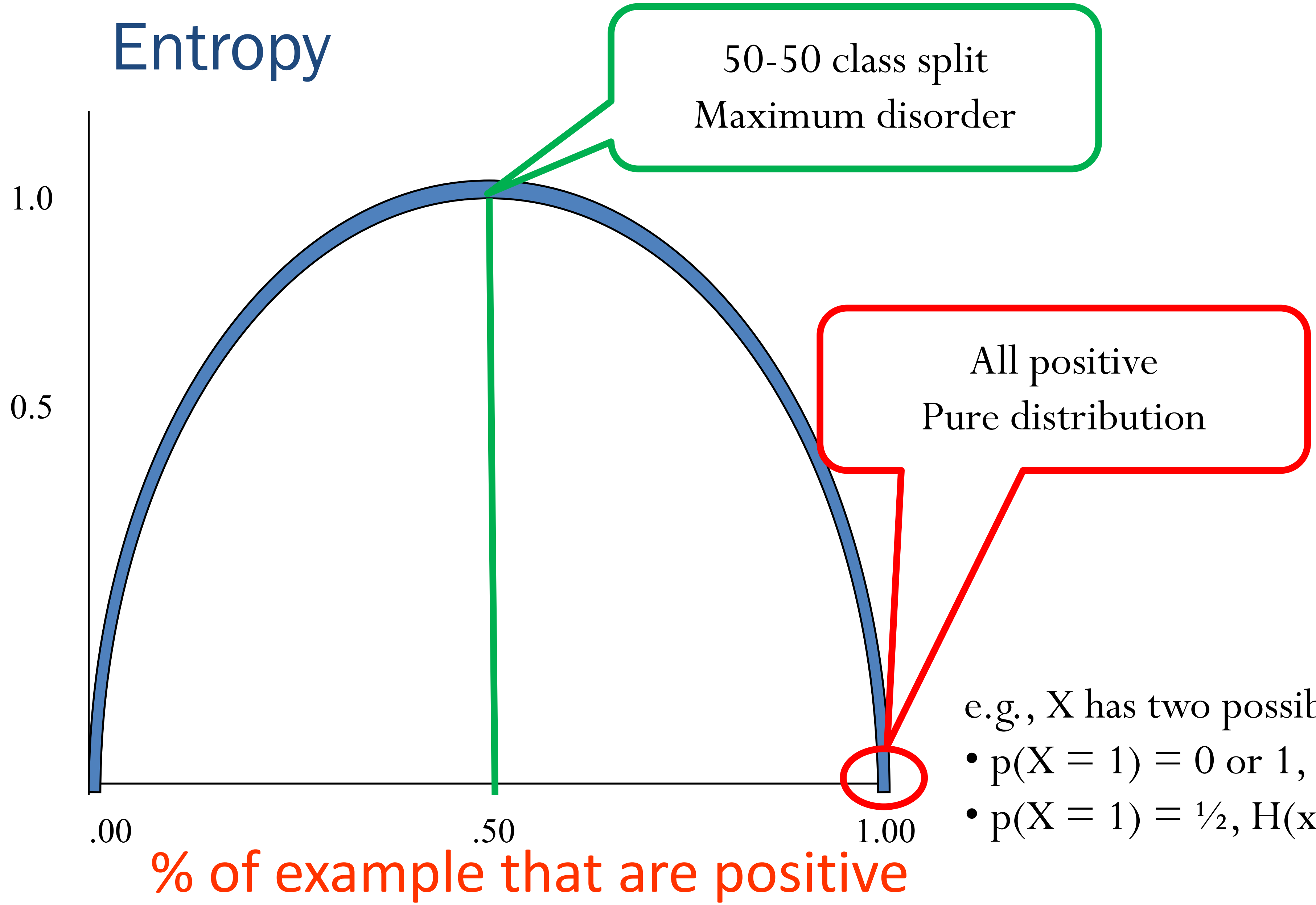
     - If subset is pure: stop
     - Else: ID3 (child node, {subset data})

Entropy (disorder) is bad
Homogeneity is good

- Let S be a set of examples
- Entropy(S) = $-P \log_2(P) - N \log_2(N)$
  - **P** is proportion of pos example
  - **N** is proportion of neg examples
  - $0 \log 0 == 0$
- Example: S has 9 pos and 5 neg

  $$\text{Entropy}([9+, 5-]) = -(9/14) \log_2(9/14) - (5/14)\log_2(5/14)$$
  $$= 0.940$$

Entropy



e.g., X has two possible values 0 and 1

- $p(X = 1) = 0$ or 1, $H(x) = 0$ bits (minimum)
- $p(X = 1) = \frac{1}{2}$, $H(x) = 1$ bits (maximum)

Information gain

Expected drop in entropy after split

$$\text{Gain}(S, A) = H(S) - \sum_{V \in \text{Values}(A)} \frac{|S_V|}{|S|} H(S_V)$$

uncertainty before split

uncertainty after split

- A: attribute
- S: set of training examples
- V: possible values of attribute A
- S_V: set of training examples with the value of attribute A = V
- Subsets with more examples have a larger effect

Maximizing Gain(S, A) is equivalent to minimizing uncertainty after split

Gain of Splitting on Wind

Values(wind)=weak, strong

S = [9+, 5-]

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

S_s = [3+, 3-]

Gain(S, wind)

$$= \text{Entropy}(S) - \sum_{v \in \{weak,\ s\}} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

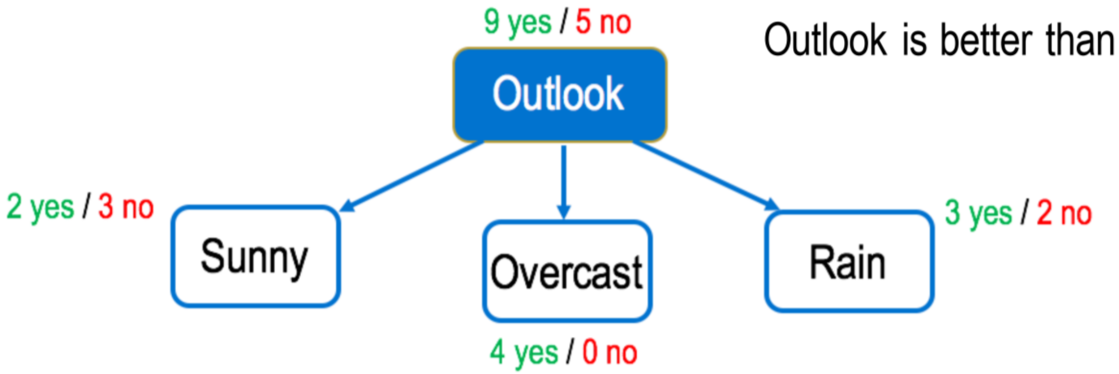
$$= \text{Entropy}(S) - 8/14\ \text{Entropy}(S_{weak}) - 6/14\ \text{Entropy}(S_s)$$

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

$$= .048$$

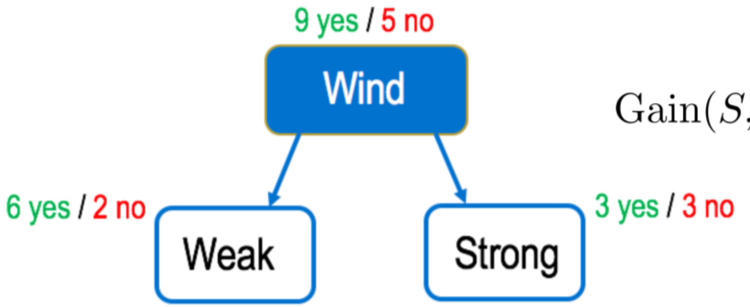
Day	Wind	Tennis?
d1	weak	n
d2	s	n
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	s	n
d7	s	yes
d8	weak	n
d9	weak	yes
d10	weak	yes
d11	s	yes
d12	s	yes
d13	weak	yes
d14	s	n

Information gain



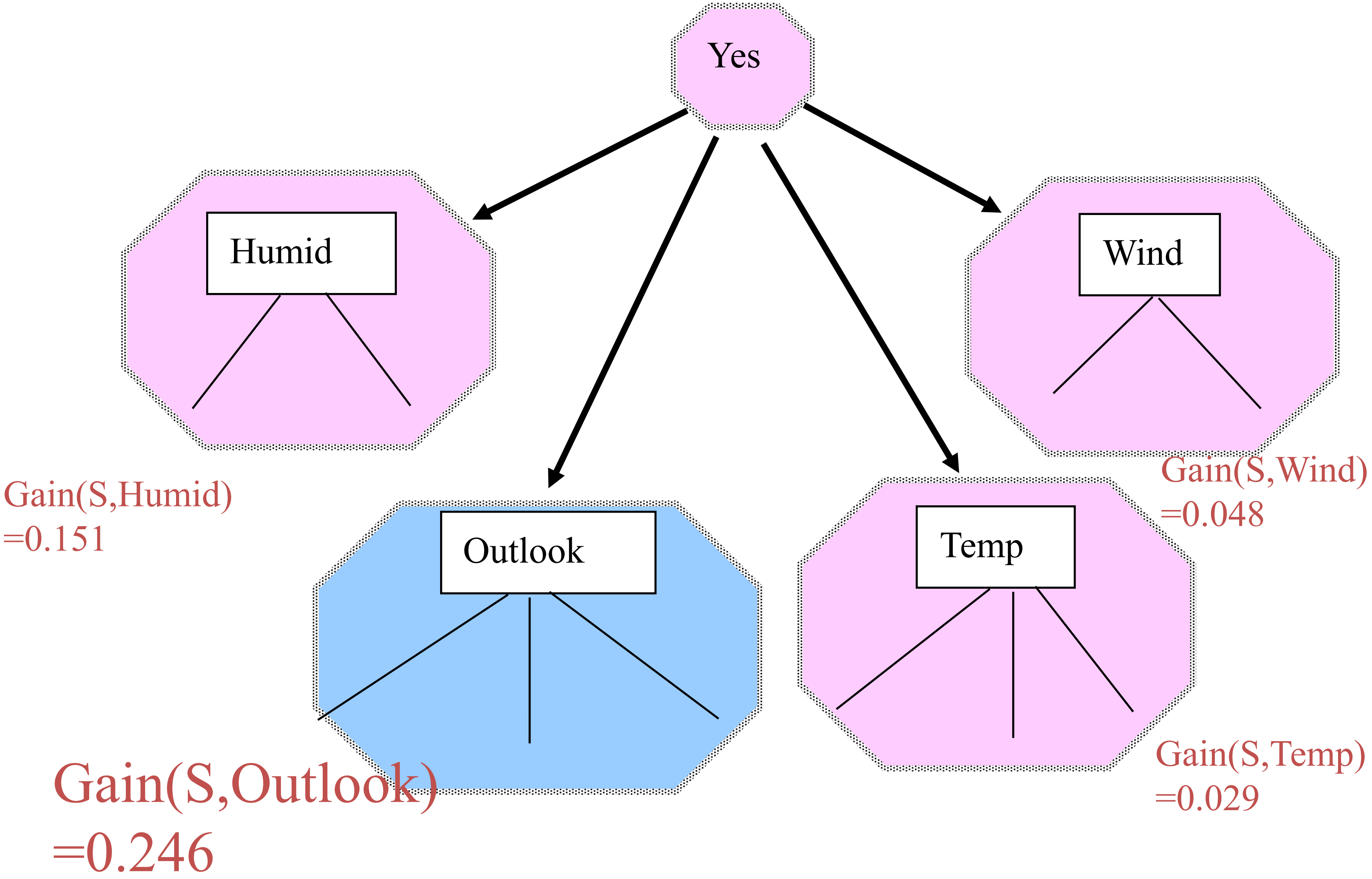
Outlook is better than Wind based on information gain

$$\mathrm{Gain}(S, \mathrm{Outlook}) = H(S) - \frac{5}{14}H(S_{\mathrm{sunny}}) - \frac{4}{14}H(S_{\mathrm{overcast}}) - \frac{5}{14}H(S_{\mathrm{rain}})$$
$$= 0.94 - \frac{5}{14} * 0.97 - \frac{4}{14} * 0 - \frac{5}{14} * 0.97 = 0.25$$



$$\mathrm{Gain}(S, \mathrm{Wind}) = H(S) - \frac{8}{14}H(S_{\mathrm{weak}}) - \frac{6}{14}H(S_{\mathrm{strong}})$$
$$= 0.94 - \frac{8}{14} * 0.81 - \frac{6}{14} * 1 = 0.049$$

Evaluating Attributes

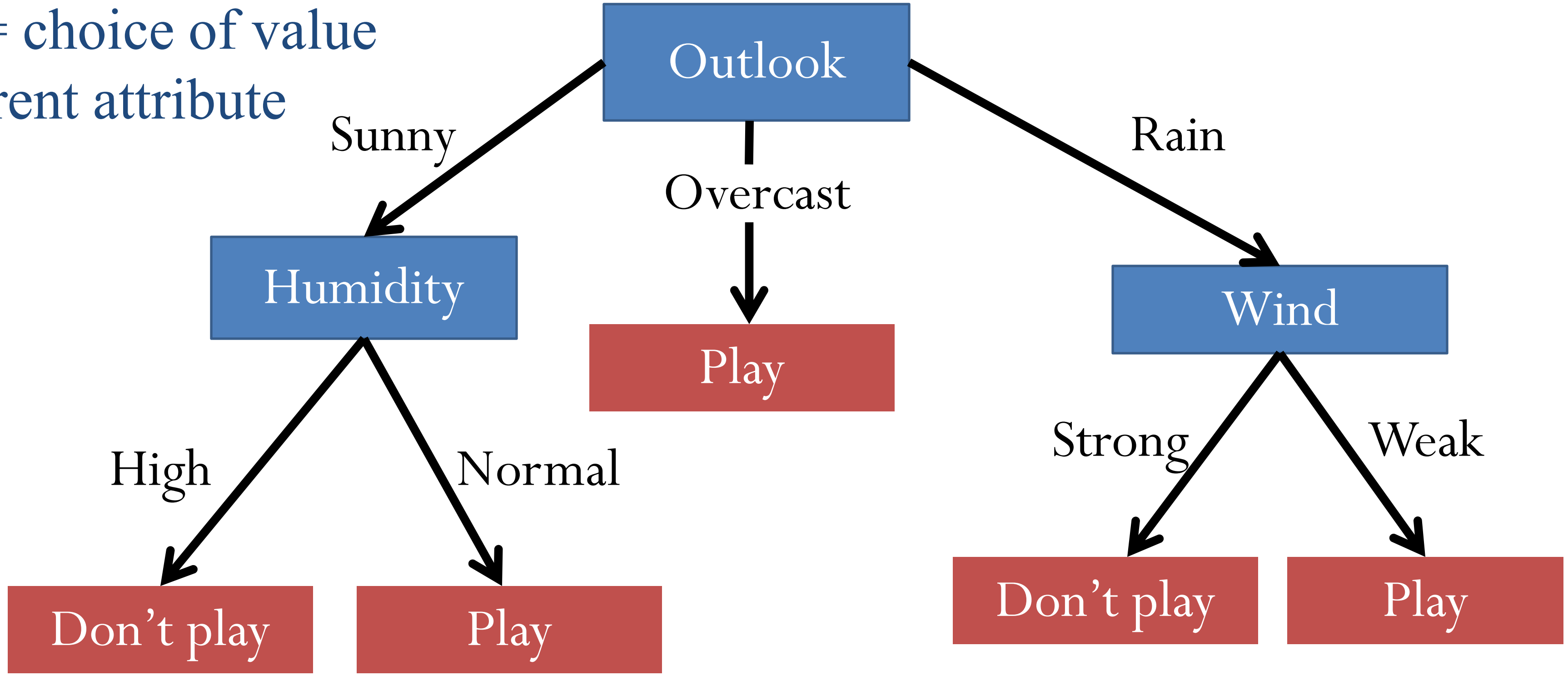


Decision Tree Representation

Good day for tennis?

Leaves = classification

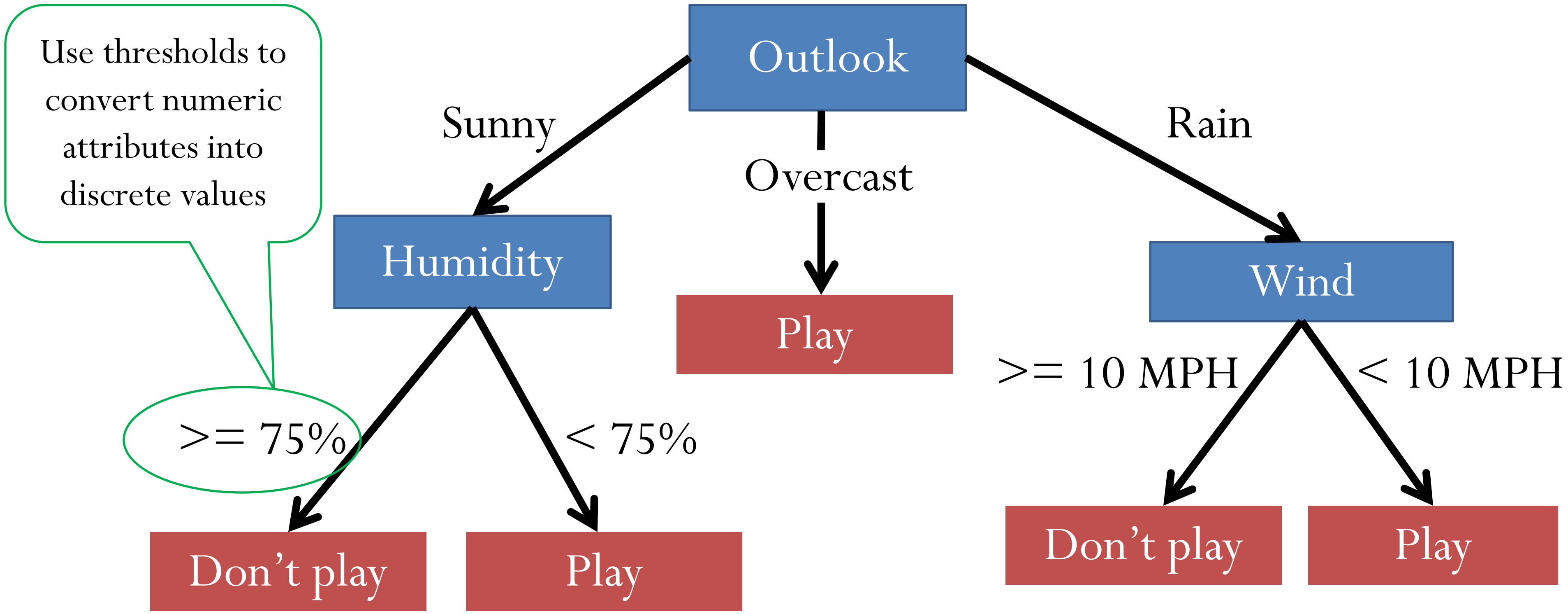
Arcs = choice of value for parent attribute



Decision tree is equivalent to logic in disjunctive normal form

$\text{Play} \Leftrightarrow (\text{Sunny} \wedge \text{Normal}) \vee \text{Overcast} \vee (\text{Rain} \wedge \text{Weak})$

Numeric Attributes

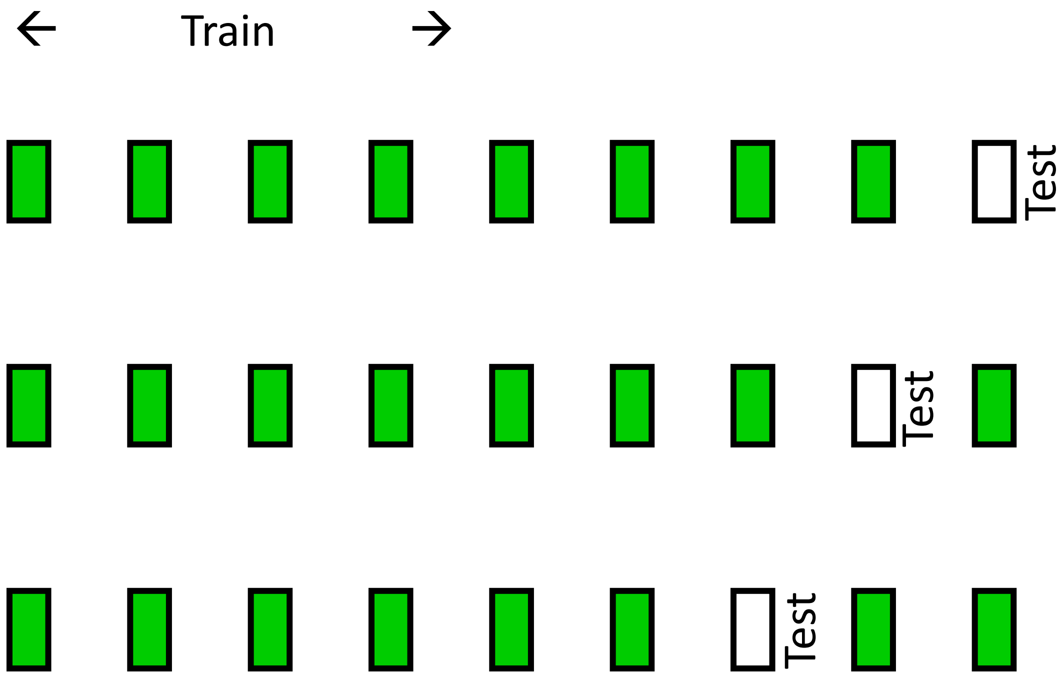


Issues

- Missing data
- Real-valued attributes
- Many-valued features
- Evaluation
- Overfitting

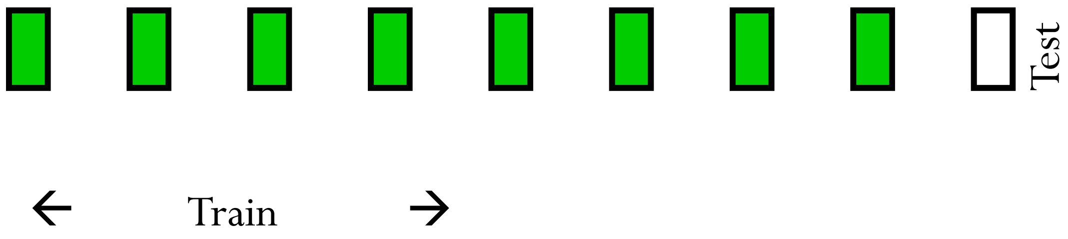
Evaluation: Cross Validation

- Partition examples into *k* disjoint sets
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data



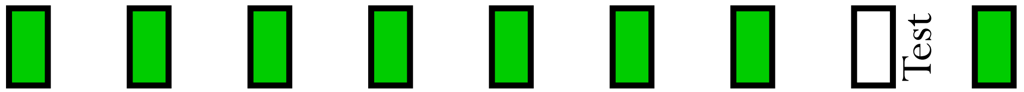
Cross validation

- Partition examples into *k* disjoint sets
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data



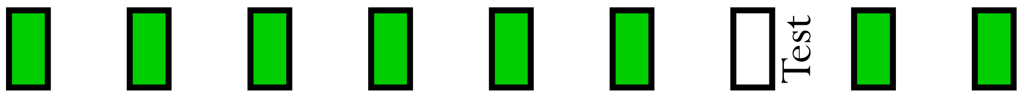
Cross Validation

- Partition examples into *k* disjoint sets
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data



Cross Validation

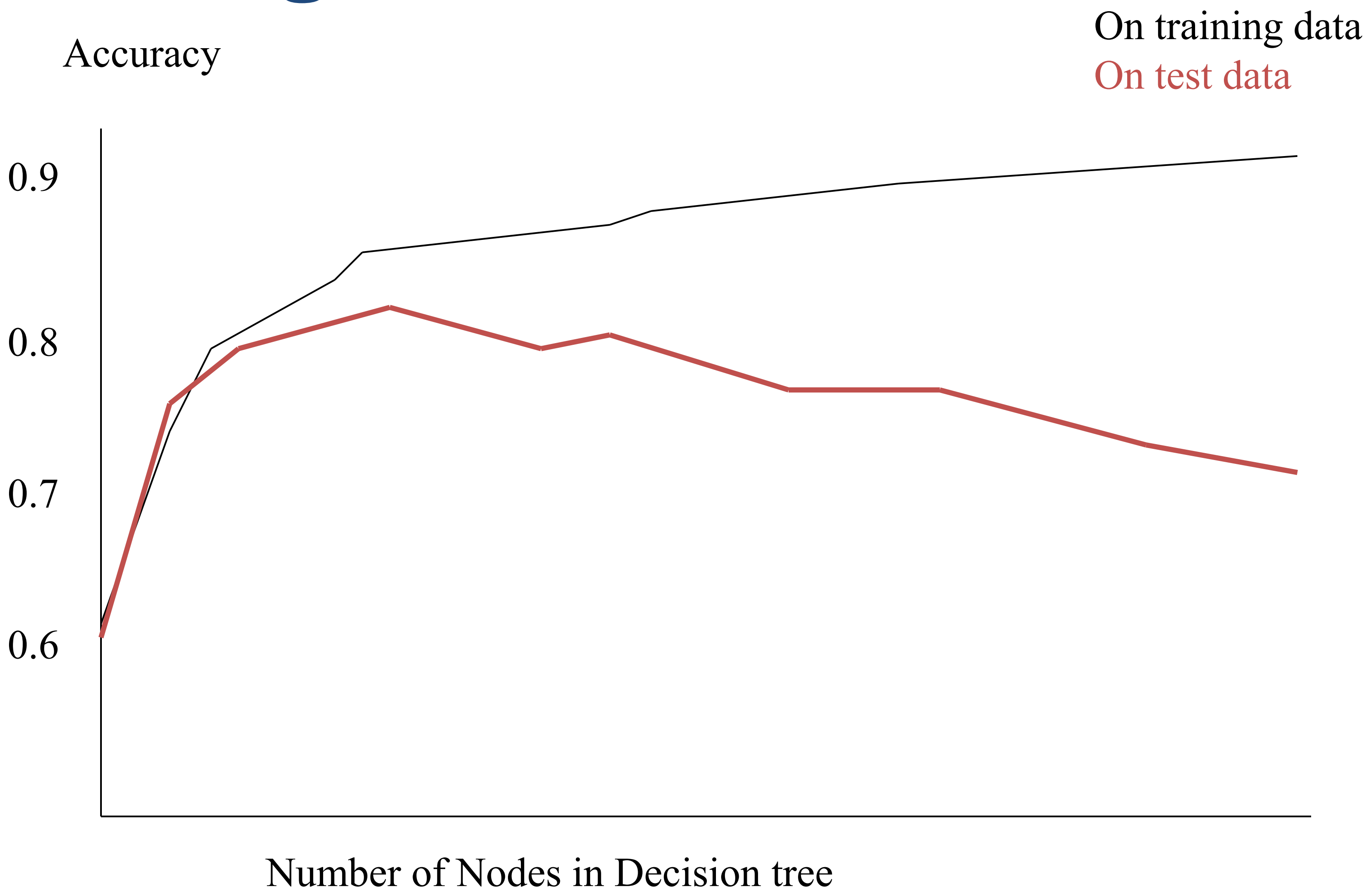
- Partition examples into *k* disjoint sets
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data



Cross-Validation (2)

- Training and validation sets
  - training set is used to build the tree
  - a separate validation set is used to evaluate the accuracy over subsequent data, and to evaluate the impact of pruning
  - justification: validation set is unlikely to exhibit the same noise and spurious correlation
  - rule of thumb: 2/3 to the training set, 1/3 to the validation set
- Leave-one-out
  - Use if $<$ 100 examples (rough estimate)
  - Hold out one example, train on remaining examples
- M of N fold
  - Repeat M times
  - Divide data into N folds, do N fold cross-validation

Overfitting



Overfitting Definition

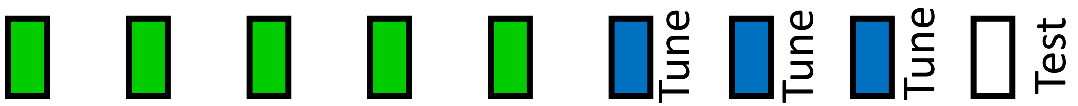
- DT is *overfit* when exists another DT' and
  - DT has ***smaller*** error on training examples, but
  - DT has ***bigger*** error on test examples
- Causes of overfitting
  - Noisy data, or
  - Training set is too small
- Solutions
  - Reduced error pruning
  - Early stopping
  - Rule post pruning

Avoid Overfitting

- How to avoid overfitting?
  - Stop growing the tree
    - before it perfectly classifies the training data
    - when data split is not statistically significant
  - Allow overfitting, but post-prune the tree
  - Grow full tree, then post-prune
  - Acquire more training data
  - Remove irrelevant attributes (manual process – not always possible)
- How to select “best” tree:
  - Measure performance over training data
  - Measure performance over separate validation data set
  - Add complexity penalty to performance measure (heuristic: simpler is better)

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 (resulting in errors) this will leads to largest gain in accuracy

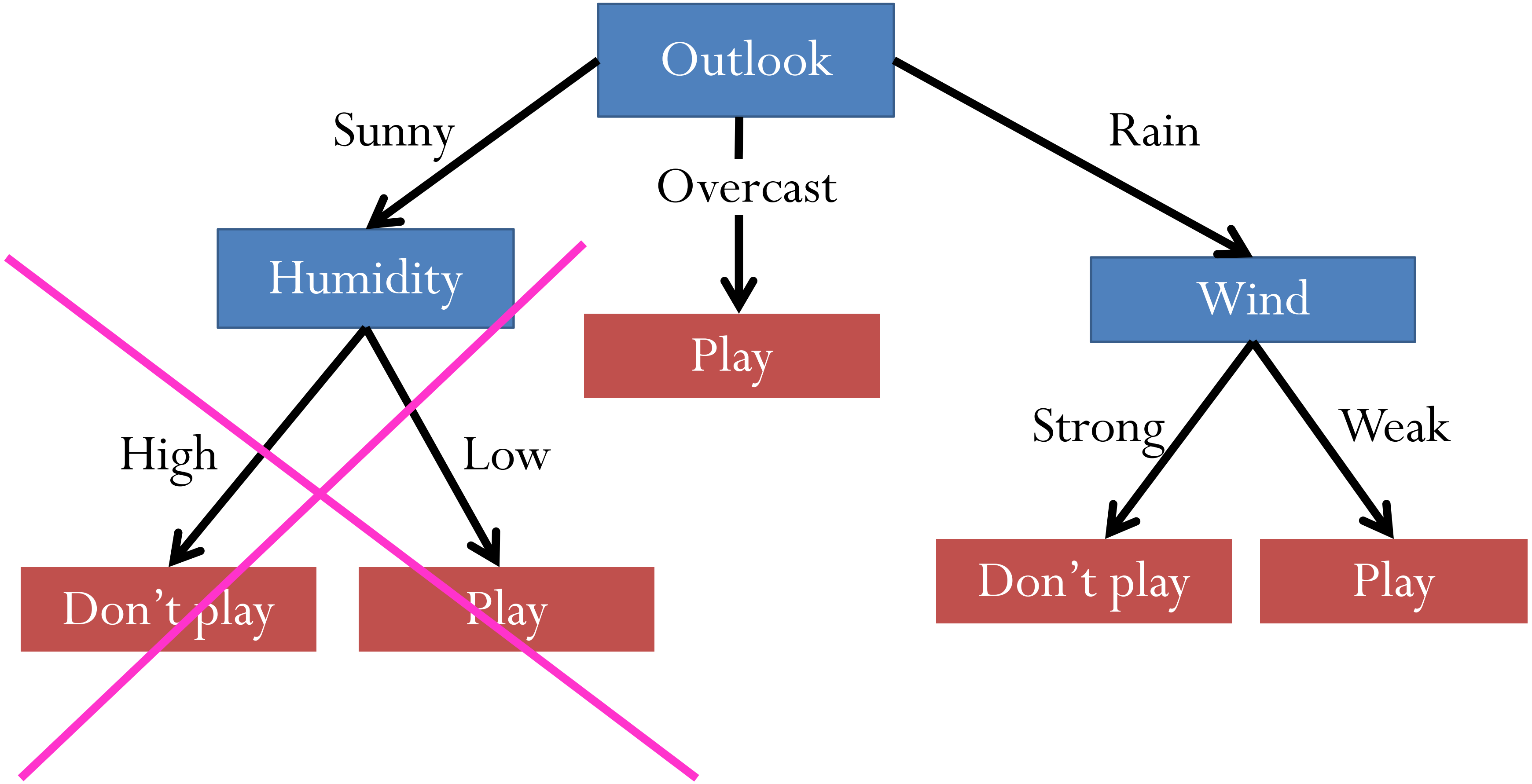
Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

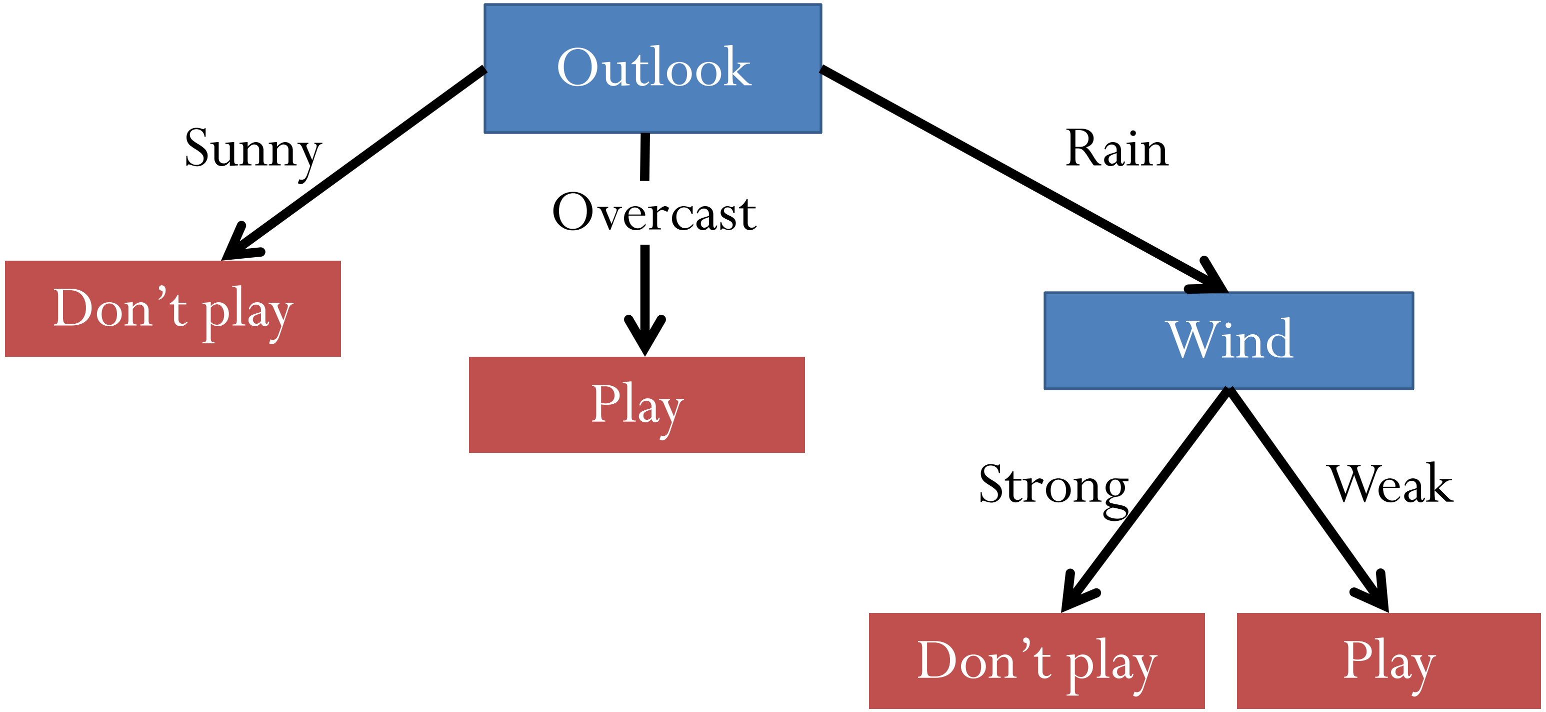
1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

Reduced Error Pruning Example



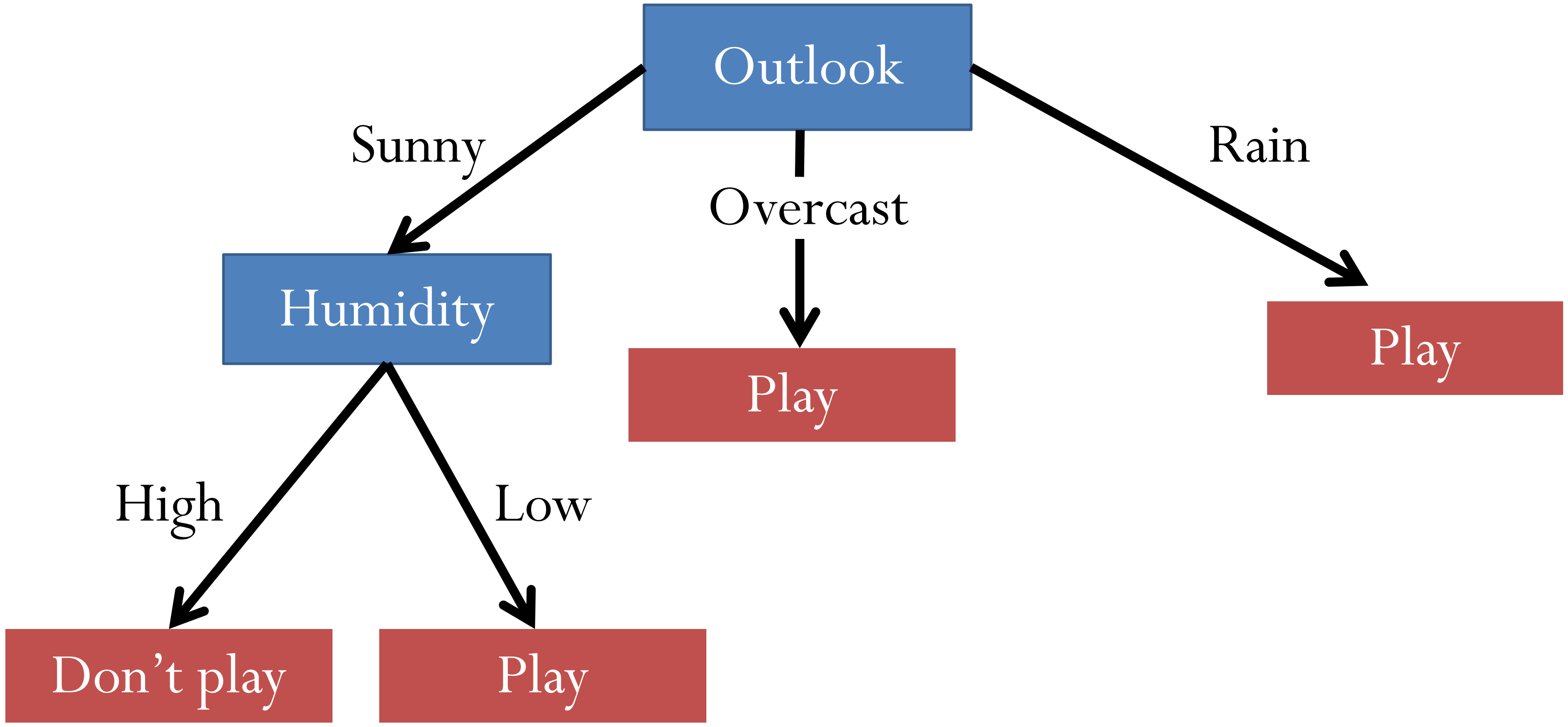
Validation set accuracy = 0.75

Reduced Error Pruning Example



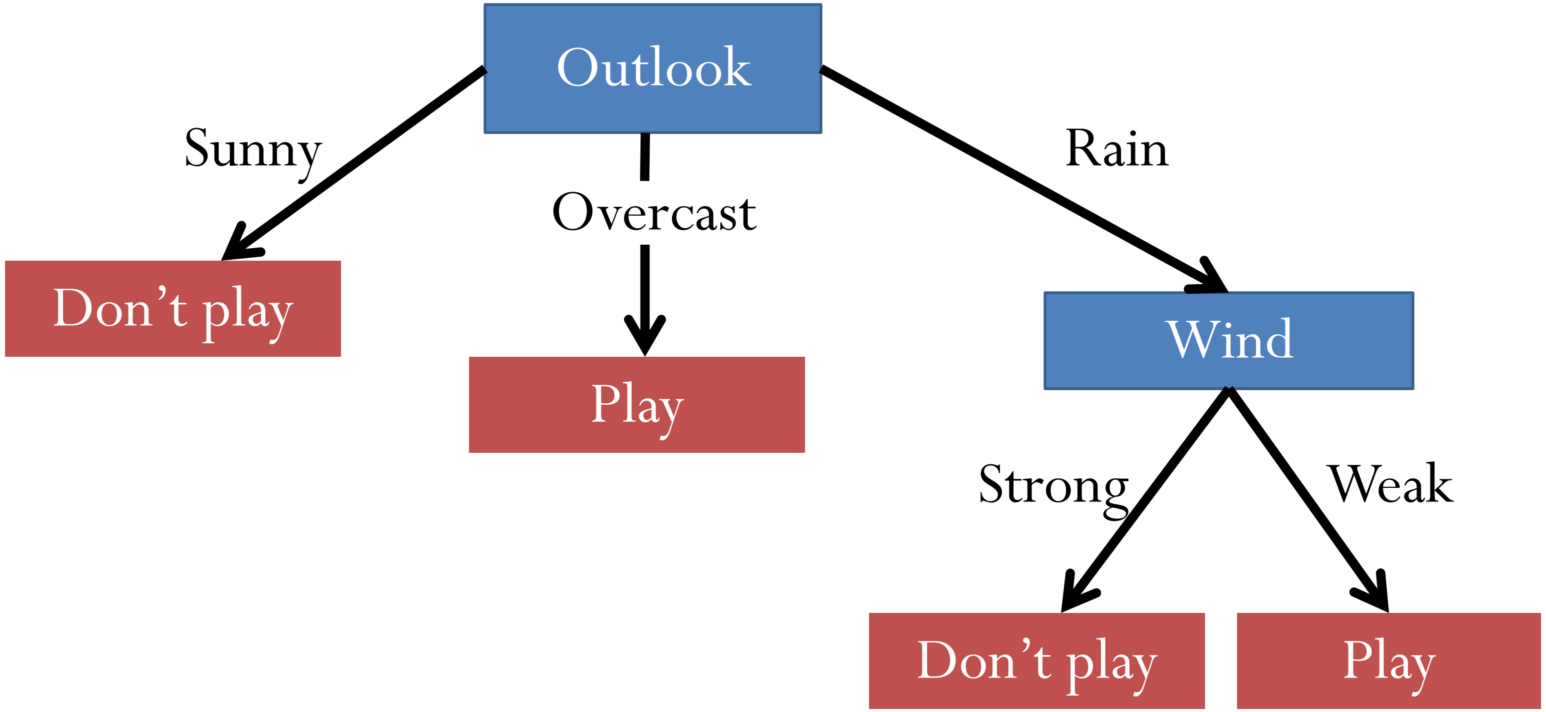
Validation set accuracy = 0.80

Reduced Error Pruning Example



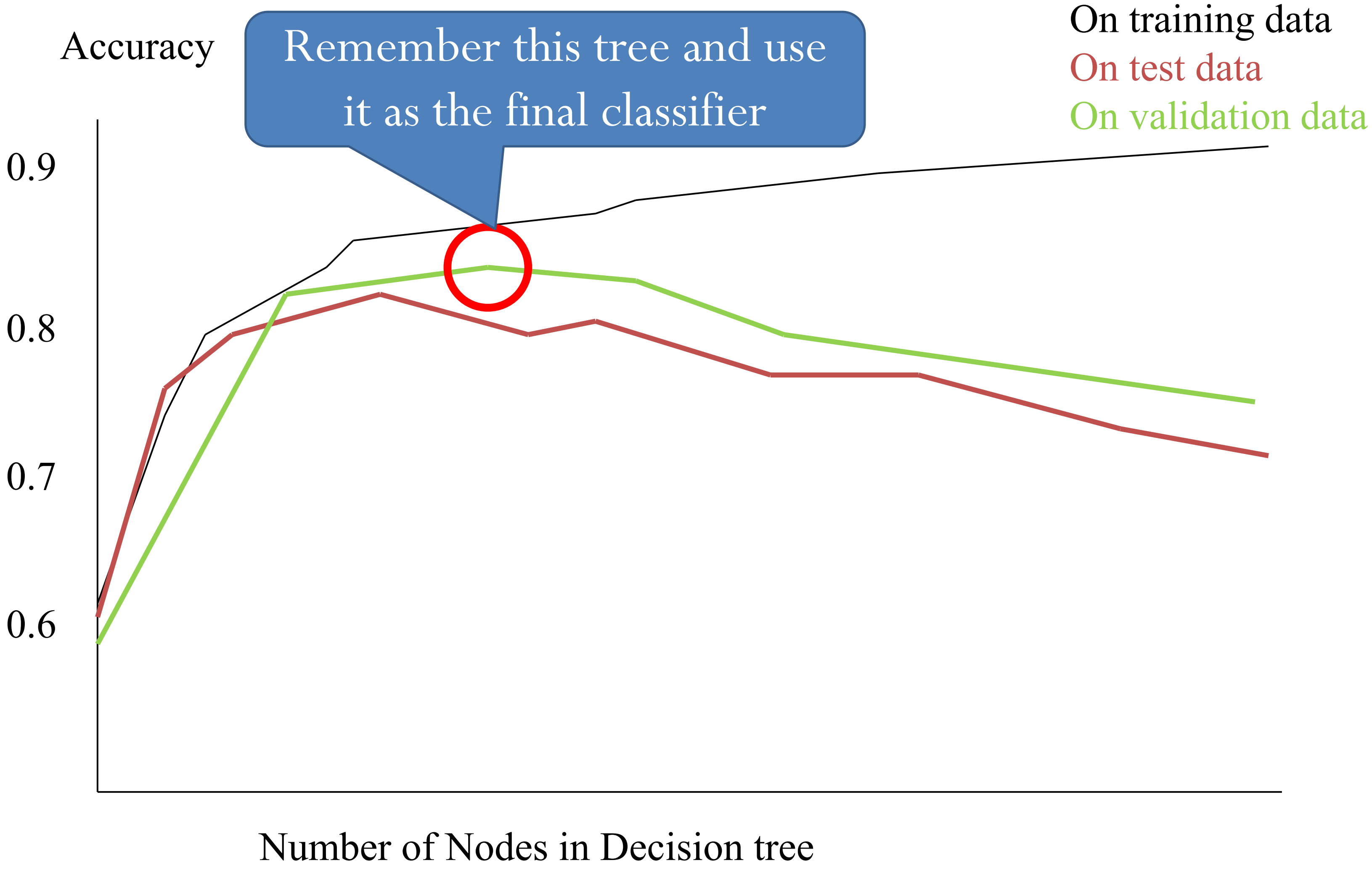
Validation set accuracy = 0.70

Reduced Error Pruning Example



Use this as final tree

Early Stopping

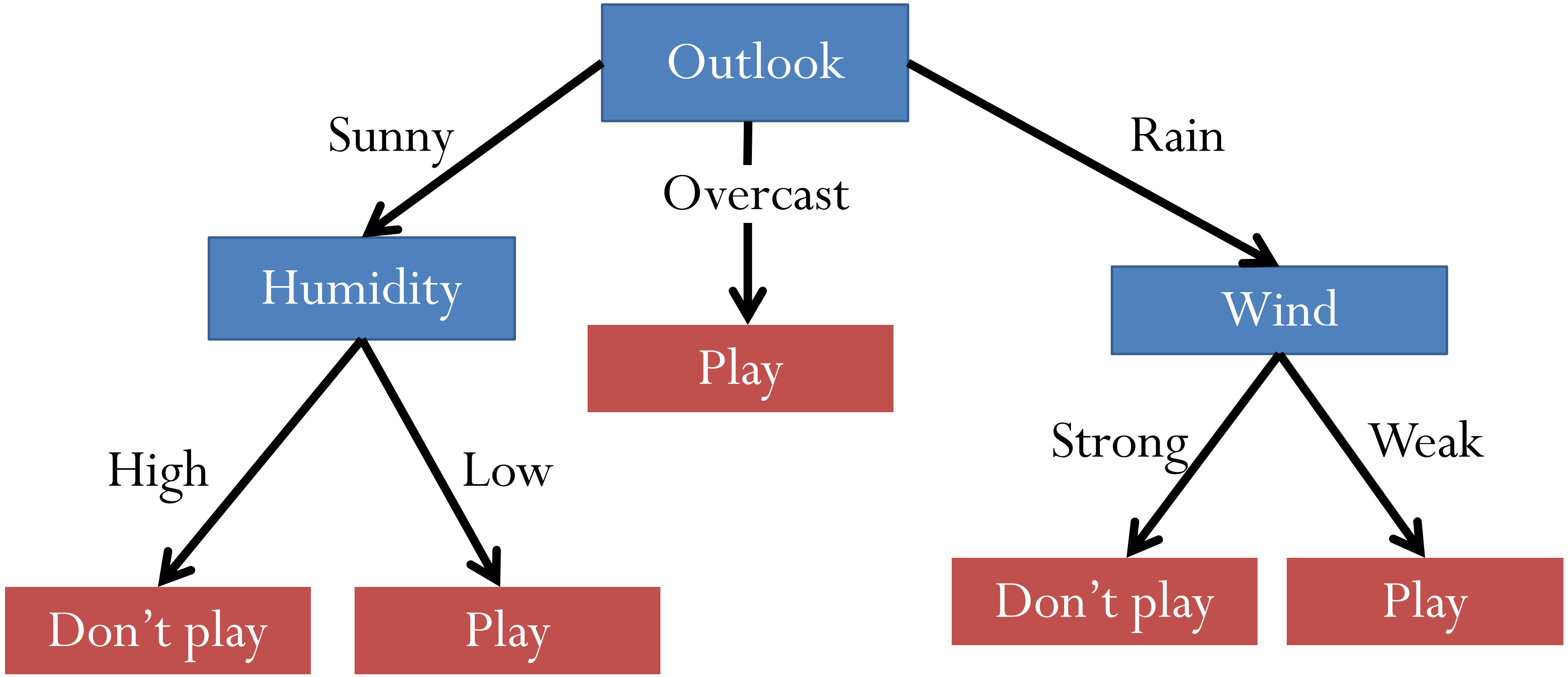


Post Rule Pruning

- 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

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

Conversion to Rule



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

$\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Low} \Rightarrow$ Play

$\text{Outlook} = \text{Overcast} \Rightarrow$ Play

…

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

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

$\ldots$

Example

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

Validation set accuracy = 0.68

$\text{Outlook} = \text{Sunny} \Rightarrow \text{Don't play}$ Validation set accuracy = 0.65

$\boxed{\text{Humidity} = \text{High} \Rightarrow \text{Don't play}}$ Validation set accuracy = 0.75

Keep this rule

Overfitting 2

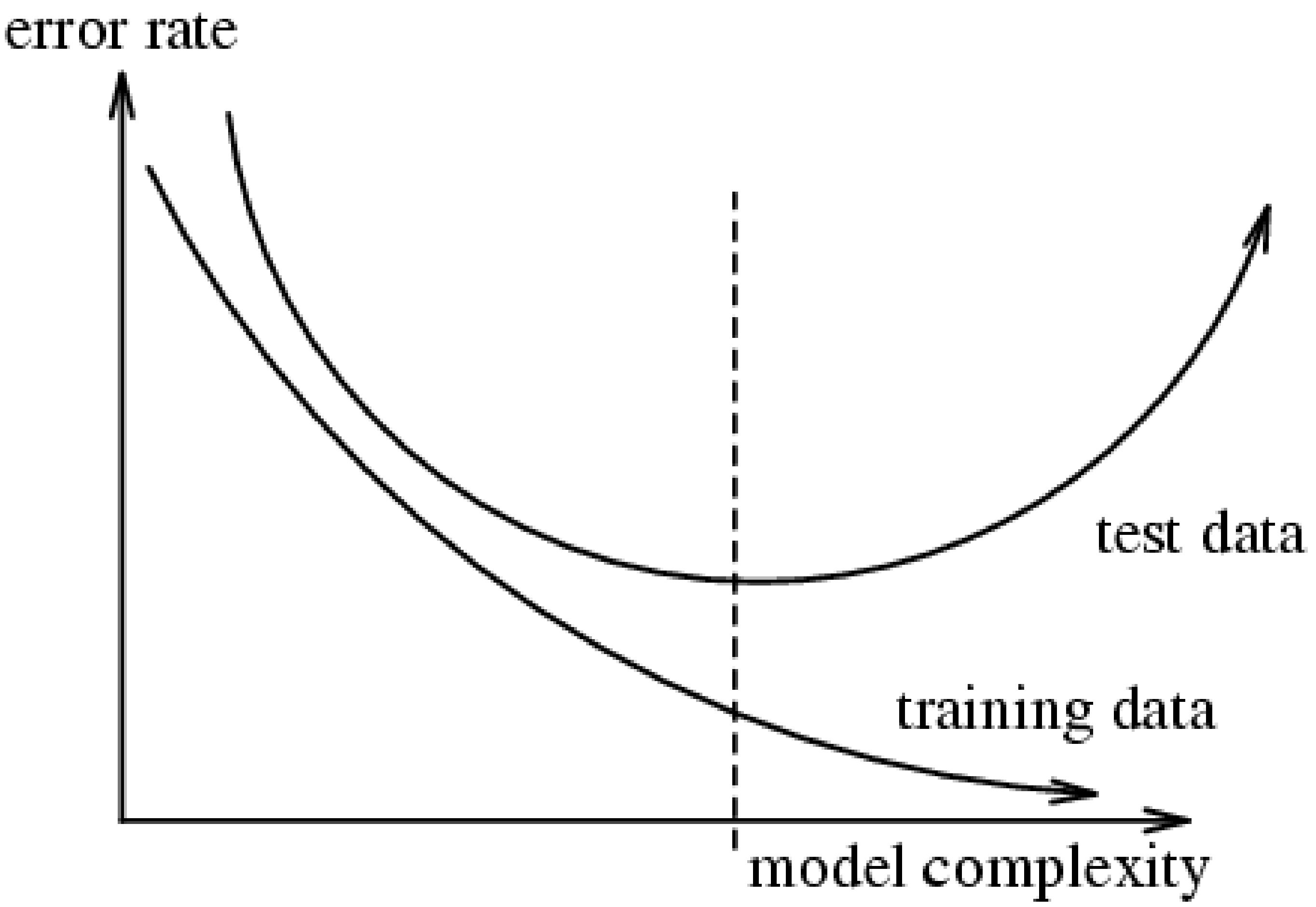
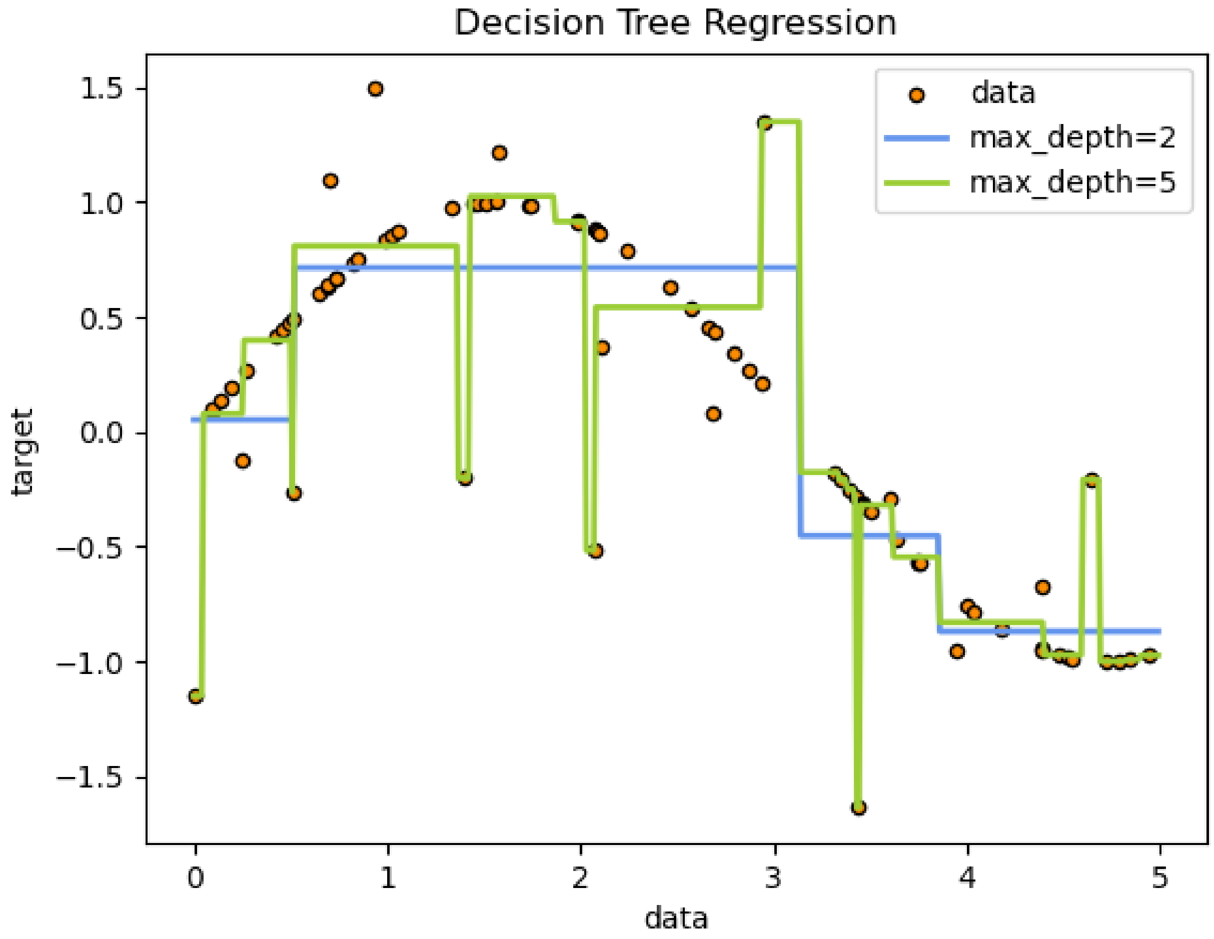


Figure from w.w.cohen

Scikit Learn on Decision Trees

- Regression



https://scikit-learn.org/stable/modules/tree.html

Scikit Learn on Decision Trees

Given training vectors $x_i \in R^n$, i=1,…, l and a label vector $y \in R^l$, a decision tree recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together.

Let the data at node m be represented by Q_m with n_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ subsets

$$Q_m^{left}(\theta) = \{(x, y) | x_j \leq t_m\}$$
$$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$$

The quality of a candidate split of node m is then computed using an impurity function or loss function $H()$, the choice of which depends on the task being solved (classification or regression)

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta))$$

Select the parameters that minimises the impurity

$$\theta^* = \operatorname{argmin}_\theta G(Q_m, \theta)$$

Recurse for subsets $Q_m^{left}(\theta^*)$ and $Q_m^{right}(\theta^*)$ until the maximum allowable depth is reached, $n_m < \min_{samples}$ or $n_m = 1$.

https://scikit-learn.org/stable/modules/tree.html

Scikit Learn on Decision Trees

- Gini and Log Loss or Entropy:

If a target is a classification outcome taking on values 0,1,…,K-1, for node m, let

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

be the proportion of class k observations in node m. If m is a terminal node, `predict_proba` for this region is set to p_{mk}. Common measures of impurity are the following.

Gini:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Log Loss or Entropy:

$$H(Q_m) = -\sum_k p_{mk} \log(p_{mk})$$

https://scikit-learn.org/stable/modules/tree.html

References

References

- Dietterich, T. G., (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation, 10* (7) 1895-1924
- Densar, J., (2006). Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets. The Journal of Machine Learning Research, pages 1-30.
- Machine Learning, Jesse Davis, jdavis@cs.washington.edu https://courses.cs.washington.edu/courses/cse573/08au/slides/
- Daniel S. Weld https://www.cs.washington.edu/people/faculty/weld
- CS489/698: Intro to ML, Lecture 19: Decision Tree, Instructor: Sun Sun (17/18/18)
- "Introductory Applied Machine Learning" by Victor Lavrenko and Nigel Goddard University of Edinburgh.
- https://scikit-learn.org/stable/modules/tree.html

תודה רבה
Hebrew

Ευχαριστώ
Greek

Спасибо
Russian

Danke

German

धन्यवादः
Sanskrit

Merci

French

ধন্যবাদ
Bangla

நன்றி
Tamil

شكراً
Arabic

ಧನ್ಯವಾದಗಳು
Kannada

Thank You
English

നന്ദി
Malayalam

多謝
Traditional Chinese

Grazie

Italian

ధన్యవాదాలు
Telugu

આભાર
Gujarati

ਧੰਨਵਾਦ
Punjabi

धन्यवाद
Hindi & Marathi

多谢
Simplified Chinese

Gracias

Spanish

https://sites.google.com/site/animeshchaturvedi07

Obrigado
Portuguese

ありがとうございました
Japanese

ขอบคุณ
Thai

감사합니다
Korean