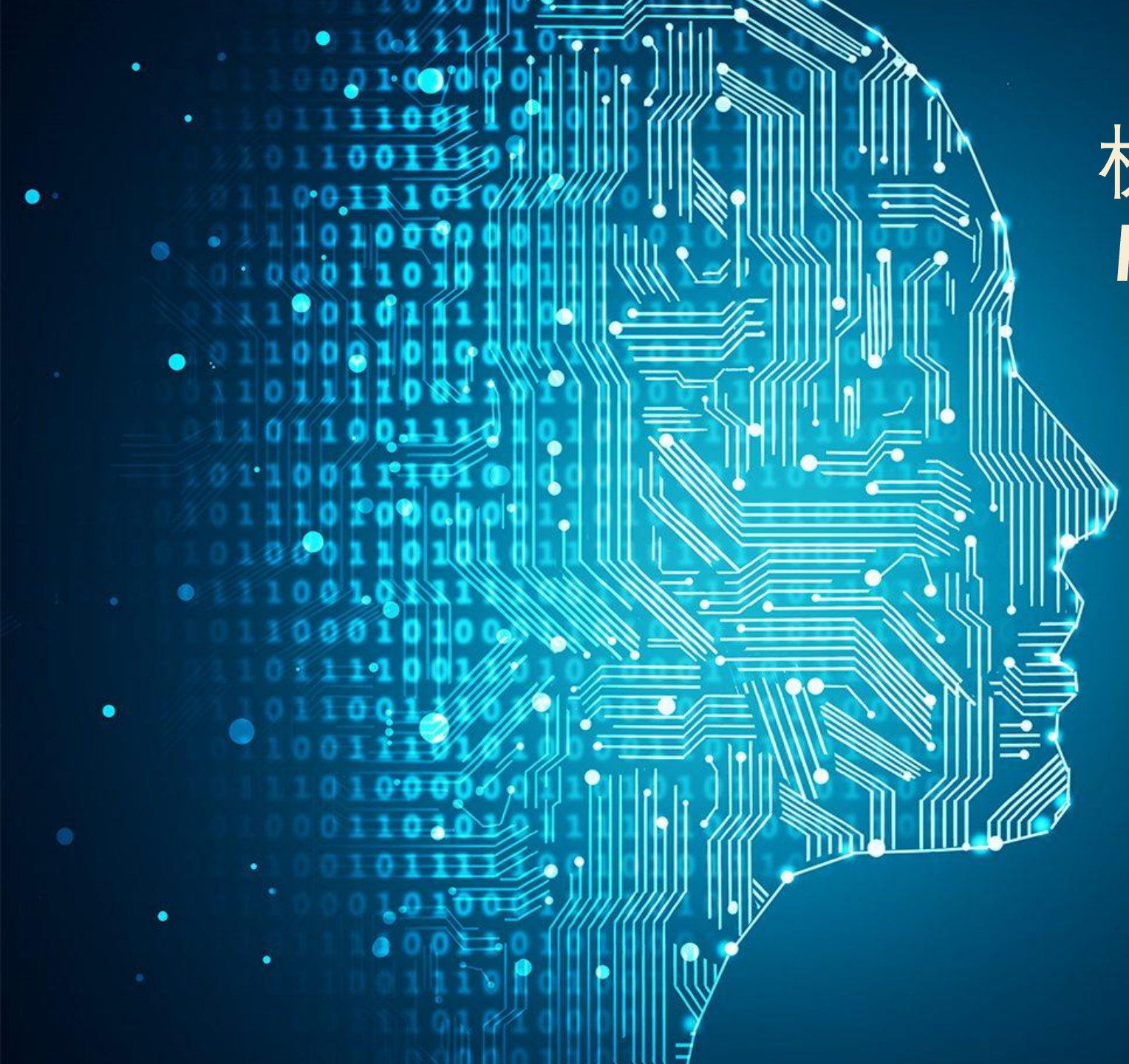


机器学习与人工智能
Machine Learning and Artificial Intelligence

Lecture 3 Decision Trees

Yingjie Zhang (张颖婕)
Peking University
yingjiezhang@gsm.pku.edu.cn

2021 Fall

Decision Trees

Classification: Terminology

- Inputs = Predictors = Independent Variables = Attributes
- Output = Responses = Dependent Variables = Class labels = Target variables
- Models = Classifiers
- With a classification algorithm, we aim to build a model to predict what output will be obtained from given inputs.

	A_1 Name	A_2 Gender	A_3 BMI	A_4 Systolic BP	A_5 Diastolic BP	A_6 Diabetes?	A_7 Heart attack risk?
x_1	Bob	Male	37	205	150	Yes	High
x_2	Kathy	Female	23	125	80	No	Low
x_3	John	Male	24	150	80	No	???

Sample Data

Outlook	Temp	Humidity	Windy	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Will I Play Tennis Today?

Input

Outlook: {Sunny, Overcast, Rain}

Temperature: {Hot, Mild, Cool}

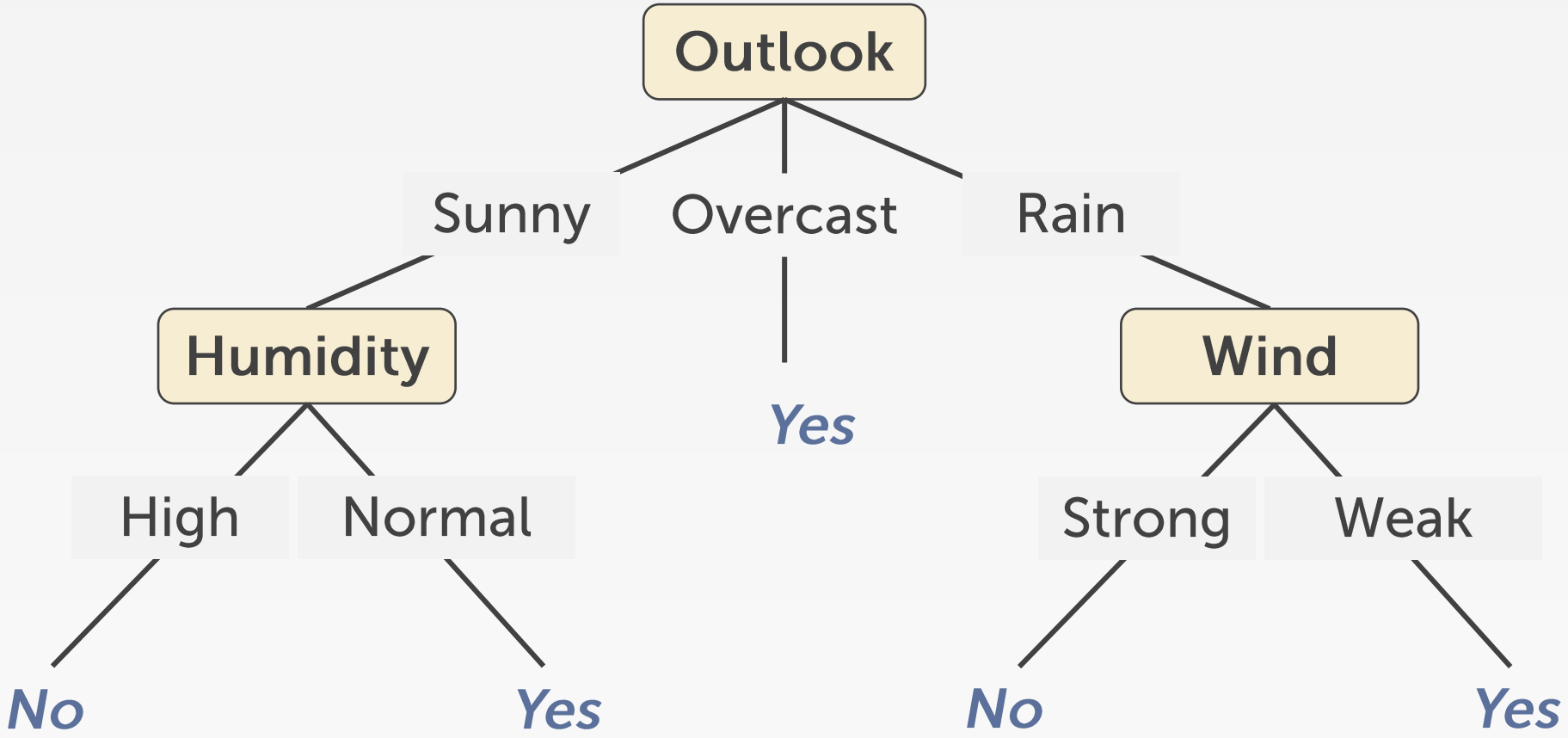
Humidity: {High, Normal, Low}

Wind: {Strong, Weak}

Labels

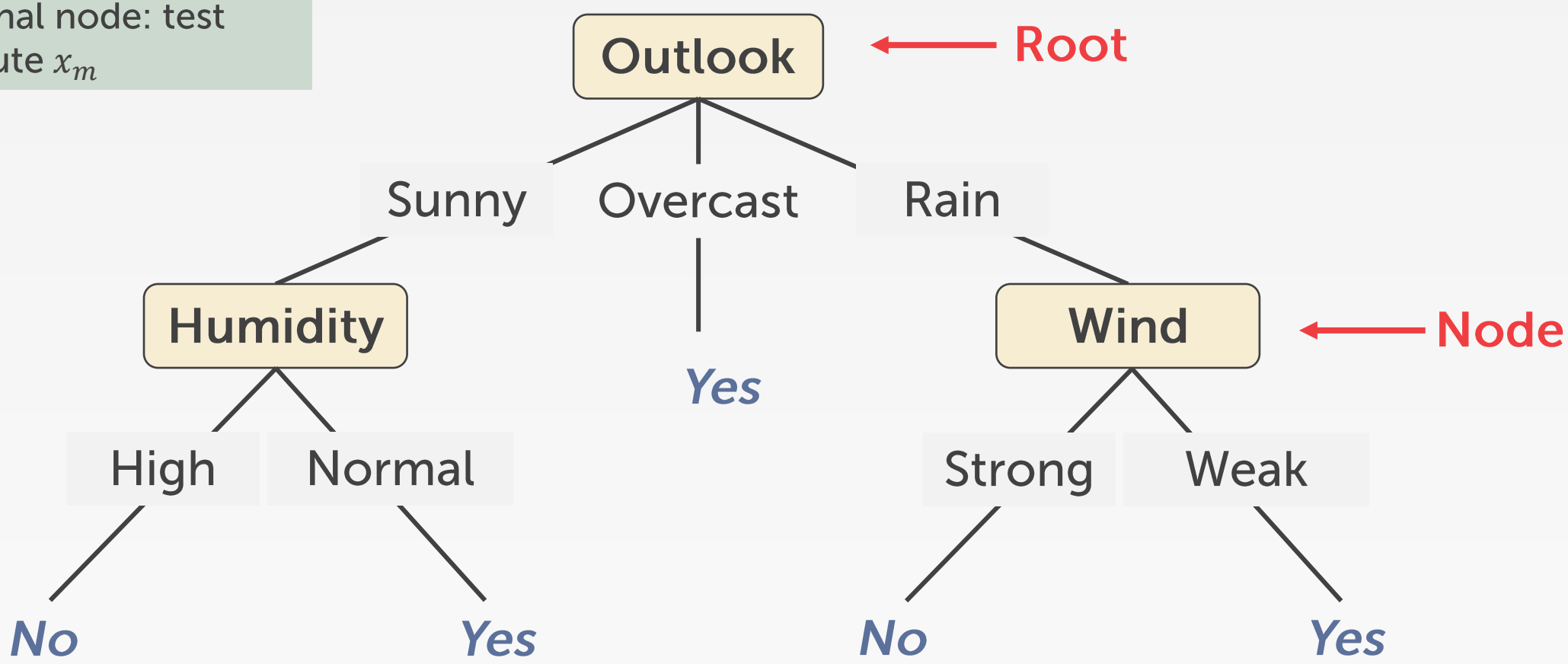
Binary classification task: Y = {Yes, No}

A Decision Tree



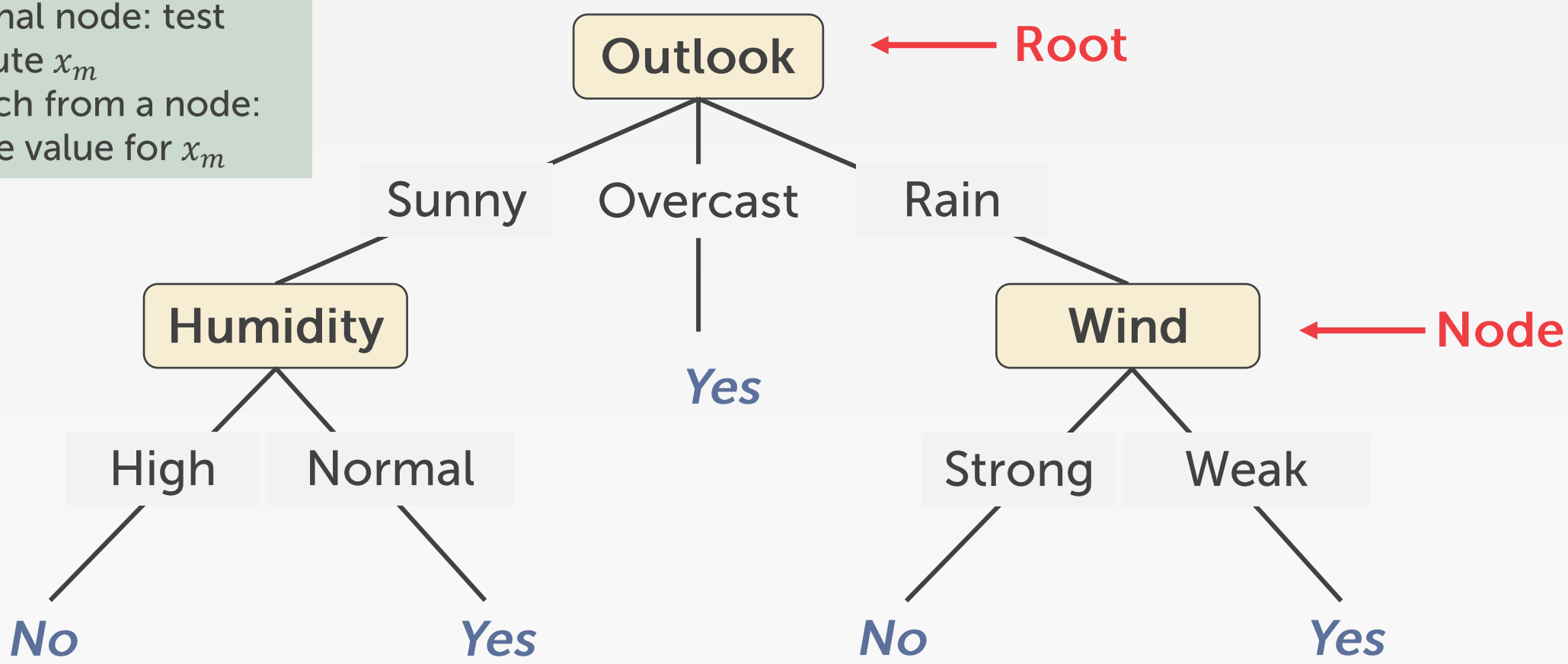
A Decision Tree

- Each internal node: test one attribute x_m



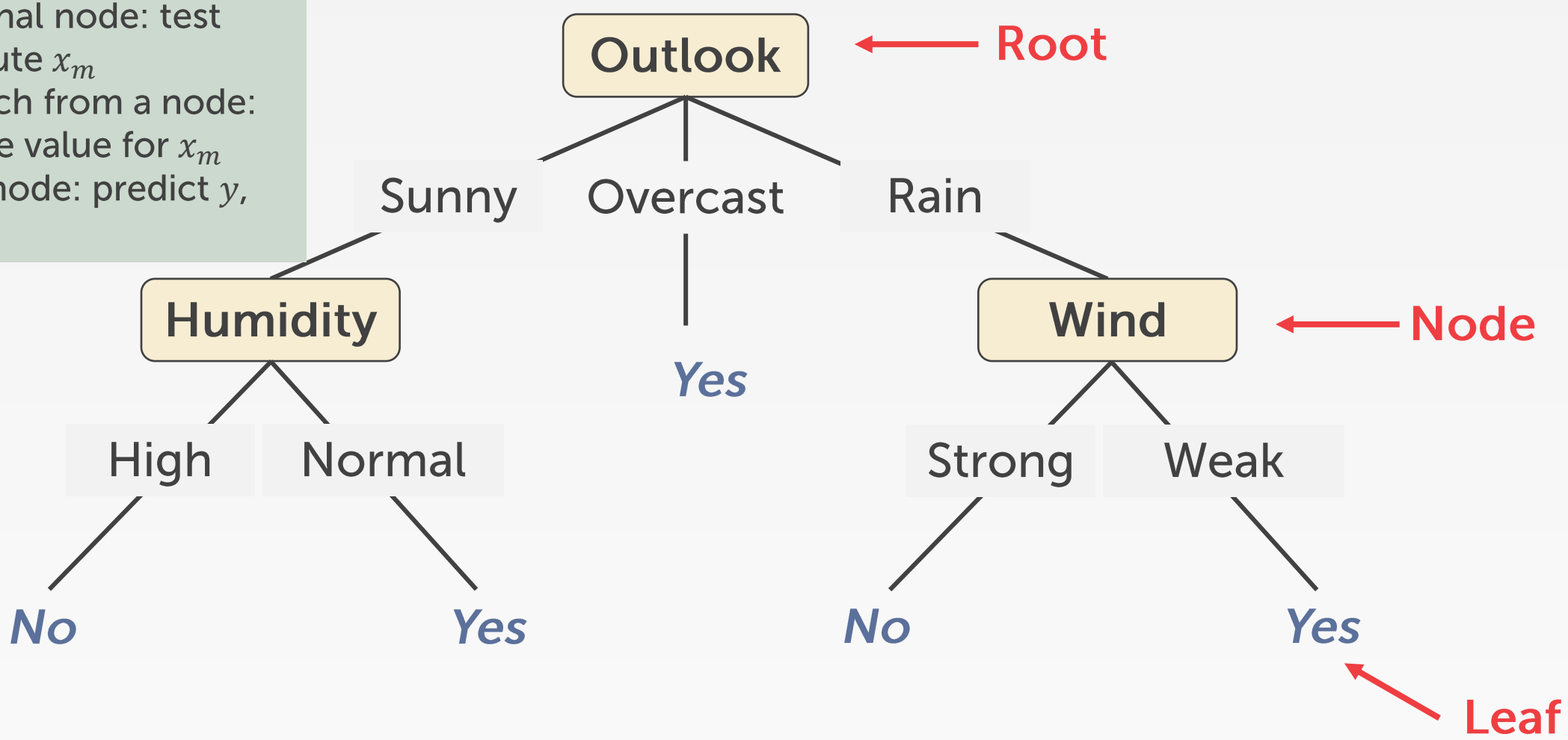
A Decision Tree

- Each internal node: test one attribute x_m
- Each branch from a node: selects one value for x_m



A Decision Tree

- Each internal node: test one attribute x_m
- Each branch from a node: selects one value for x_m
- Each leaf node: predict y, or $p(y|\vec{x})$



Build a Decision Tree?

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the “best” decision attribute for the next node.

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the “best” decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m, create a branch labeled with v

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the "best" decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m, create a branch labeled with v
4. Partition node's data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in node's\ data | x_m = v\}, \forall v$

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the "best" decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m, create a branch labeled with v
4. Partition node's data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in node's\ data | x_m = v\}, \forall v$
5. Recurse on each branch (for each v)

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the "best" decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m, create a branch labeled with v
4. Partition node's data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in node's\ data | x_m = v\}, \forall v$
5. Recurse on each branch (for each v)
6. If training examples are perfectly classified, stop and return node with label = majority vote (node's data).

Basic Algorithm (ID3, C4.5)

root = new Node (data = D)

Return train_tree(root)

def train_tree(node):

1. X_m ←the "best" decision attribute for the next node.
2. Assign X_m as decision attribute for node.
3. For each value v of X_m, create a branch labeled with v
4. Partition node's data into descendants $D_{x_m=v} = \{(\vec{x}, y) \in node's\ data | x_m = v\}, \forall v$
5. Recurse on each branch (for each v)
6. If training examples are perfectly classified, stop and return node with label = majority vote (node's data).

How do we choose which attribute is best?

Choose the Best Attribute

Key problem: choosing which attribute to split a given set of examples

Some possibilities are:

- Error rate (or accuracy if we want to pick the tree that maximizes the criterion)
- Information gain
- Gini gain
- Random
- ...

Toy Examples (DT with error rates)

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	0
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Toy Examples (DT with error rates)

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	0
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Y	A	B
-	1	0
-	1	0
+	1	0
+	1	0
+	1	1
+	1	1
+	1	1
+	1	1

Decision Tree Algorithm

- ID3 (Iterative Dichotomiser 3)
  - Information gain
- C4.5 (successor of ID3)
  - Gain ratio
- CART (Classification and Regression Tree)
  - Gini impurity …

Gini

- Gini impurity:

$$G(D) = 1 - \sum_{k=1}^{K} [p(y = k)]^2$$

- Given an attribute:

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

Gini

- Gini impurity:

$$G(D) = 1 - \sum_{k=1}^{K} [p(y = k)]^2$$

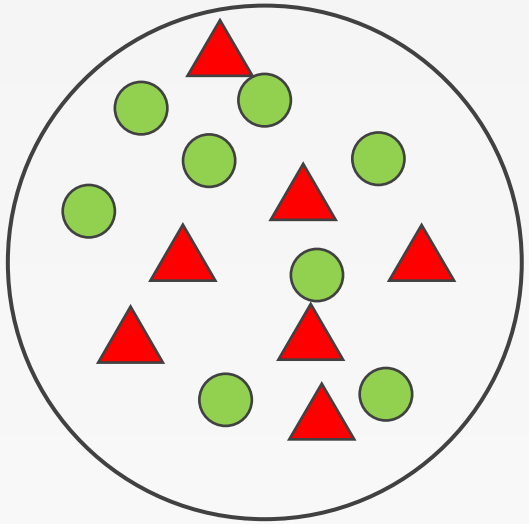
- Given an attribute:

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

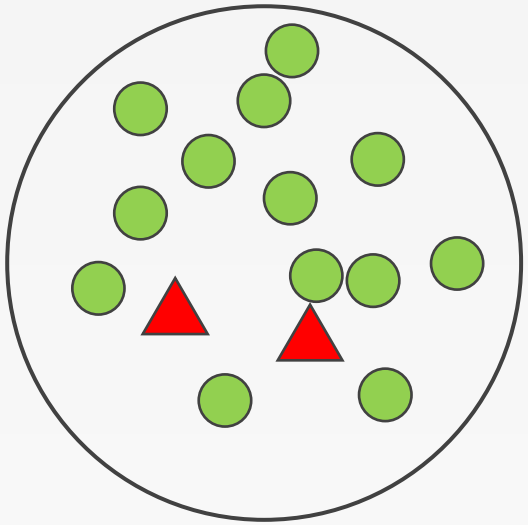
Y	A	B
-	1	0
-	1	0
+	1	0
+	1	0
+	1	1
+	1	1
+	1	1
+	1	1

Impurity

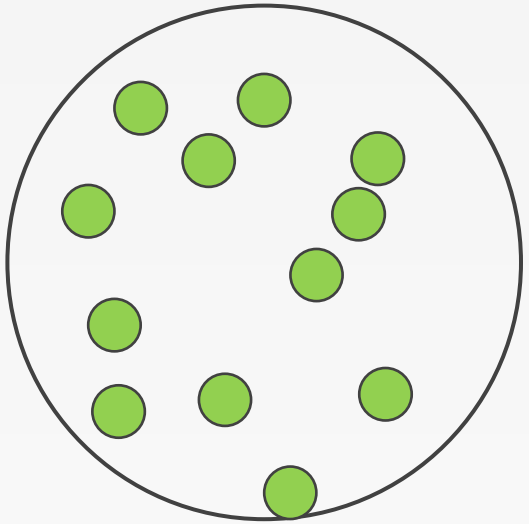
Very Impure Group



Less Impure



Minimum Impurity



Entropy

- Entropy (impurity, disorder) of a set of examples, D, relative to a binary classification is:

$$Entropy(D) = -p_+ \, log(p_+) - p\_log(p\_)$$

- p_+ is the proportion of positive examples in D and $p\_$ is the proportion of negative examples in D
  - If all samples belong to the same category: Entropy = 0
  - If all samples are equally mixed (0.5, 0.5): Entropy = 1
  - Entropy = Level of uncertainty

Entropy

- If there are more than two classes (K classes)...

$$Entropy(D) = \sum_{k=1}^{K} -p_k log(p_k)$$

p_k is the proportion of D belonging to class k

Information Gain

- The information gain of an attribute X_m is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

- D_v is the subset of D of which attribute X_m has value v
- Partitions of low entropy (imbalanced splits) lead to high gain

Sample Data

Outlook	Temp	Humidity	Windy	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

An Illustrative Example

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

? examples, ? with no and ? with yes

Before splitting

Entropy = ?

An Illustrative Example

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

14 examples, 5 with no and 9 with yes

Before splitting

$$\text{Entropy} = -\frac{9}{14} \cdot \log_2 \frac{9}{14} - \frac{5}{14} \cdot \log_2 \frac{5}{14} = \mathbf{0.940}$$

Information Gain: Outlook

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Outlook = Sunny: P(Y=1) = 2/5, $Entropy_{sunny} = 0.971$

Outlook = Overcast: P(Y=1) = 4/4, $Entropy_{Overcast} = 0$

Outlook = Rain: P(Y=1) = 3/5, $Entropy_{rain} = 0.971$

Expected entropy:

$$\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 = 0.694$$

Information gain:

$$0.940 - 0.694 = \mathbf{0.246}$$

Information Gain

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

Information gain:

Outlook: $\mathbf{0.246}$

Temperature:

Humidity:

Wind:

Information Gain

O	T	H	W	Play
S	H	H	W	No
S	H	H	S	No
O	H	H	W	Yes
R	M	H	W	Yes
R	C	N	W	Yes
R	C	N	S	No
O	C	N	S	Yes
S	M	H	W	No
S	C	N	W	Yes
R	M	N	W	Yes
S	M	N	S	Yes
O	M	H	S	Yes
O	H	N	W	Yes
R	M	H	S	No

$$Gain(D, X_m) = Entropy(D) - \sum_{v \in Values(X_m)} \frac{|D_v|}{|D|} Entropy(D_v)$$

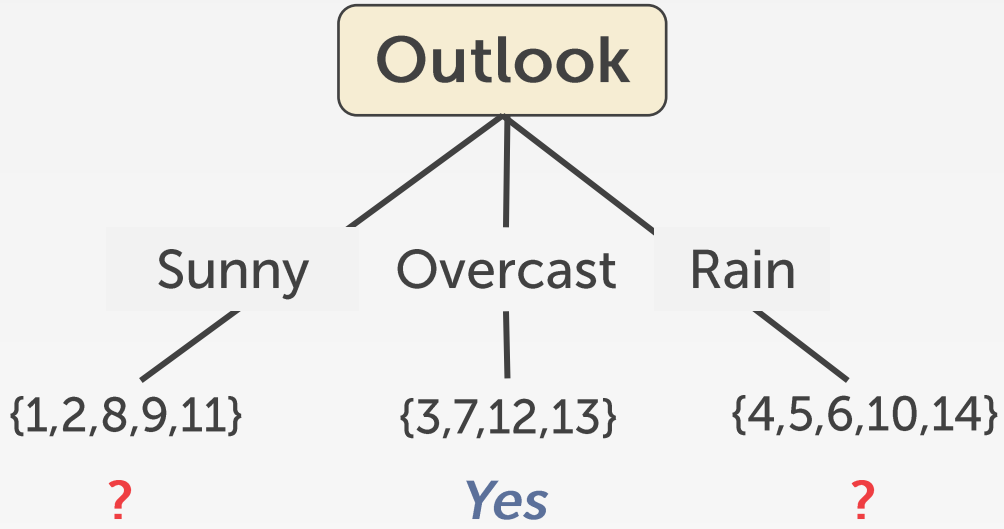
Information gain:

Outlook:	$\mathbf{0.246}$
Temperature:	0.029
Humidity:	0.151
Wind:	0.048

→ Split on Outlook

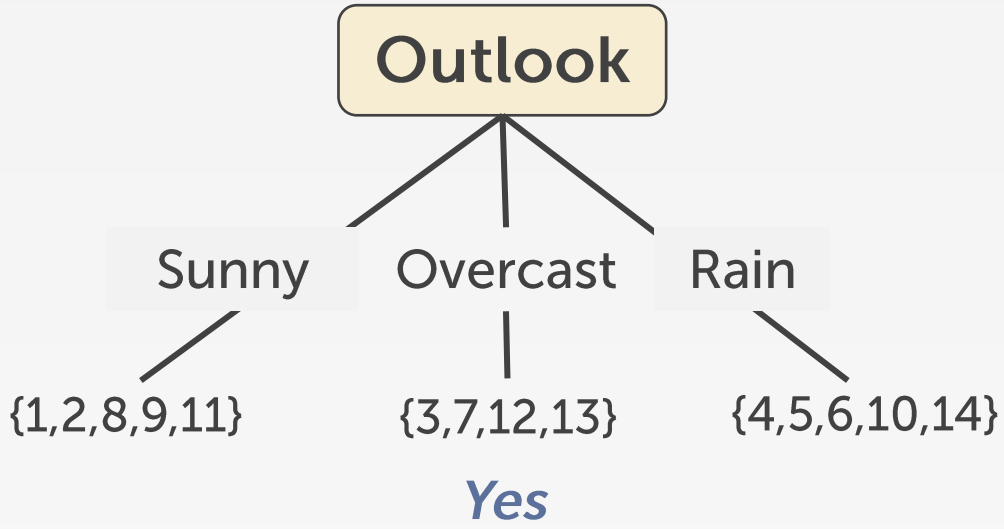
An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
3	O	H	H	W	Yes
4	R	M	H	W	Yes
5	R	C	N	W	Yes
6	R	C	N	S	No
7	O	C	N	S	Yes
8	S	M	H	W	No
9	S	C	N	W	Yes
10	R	M	N	W	Yes
11	S	M	N	S	Yes
12	O	M	H	S	Yes
13	O	H	N	W	Yes
14	R	M	H	S	No



An Illustrative Example

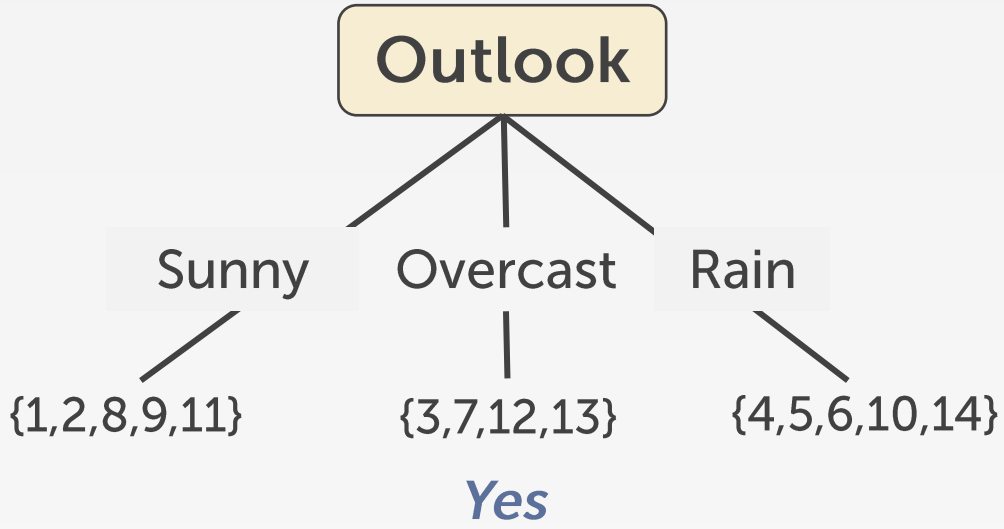
	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
8	S	M	H	W	No
9	S	C	N	W	Yes
11	S	M	N	S	Yes



$$Gain(S_{sunny}, Temperature) =$$
$$Gain(S_{sunny}, Humidity) =$$
$$Gain(S_{sunny}, Wind) =$$

An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
8	S	M	H	W	No
9	S	C	N	W	Yes
11	S	M	N	S	Yes



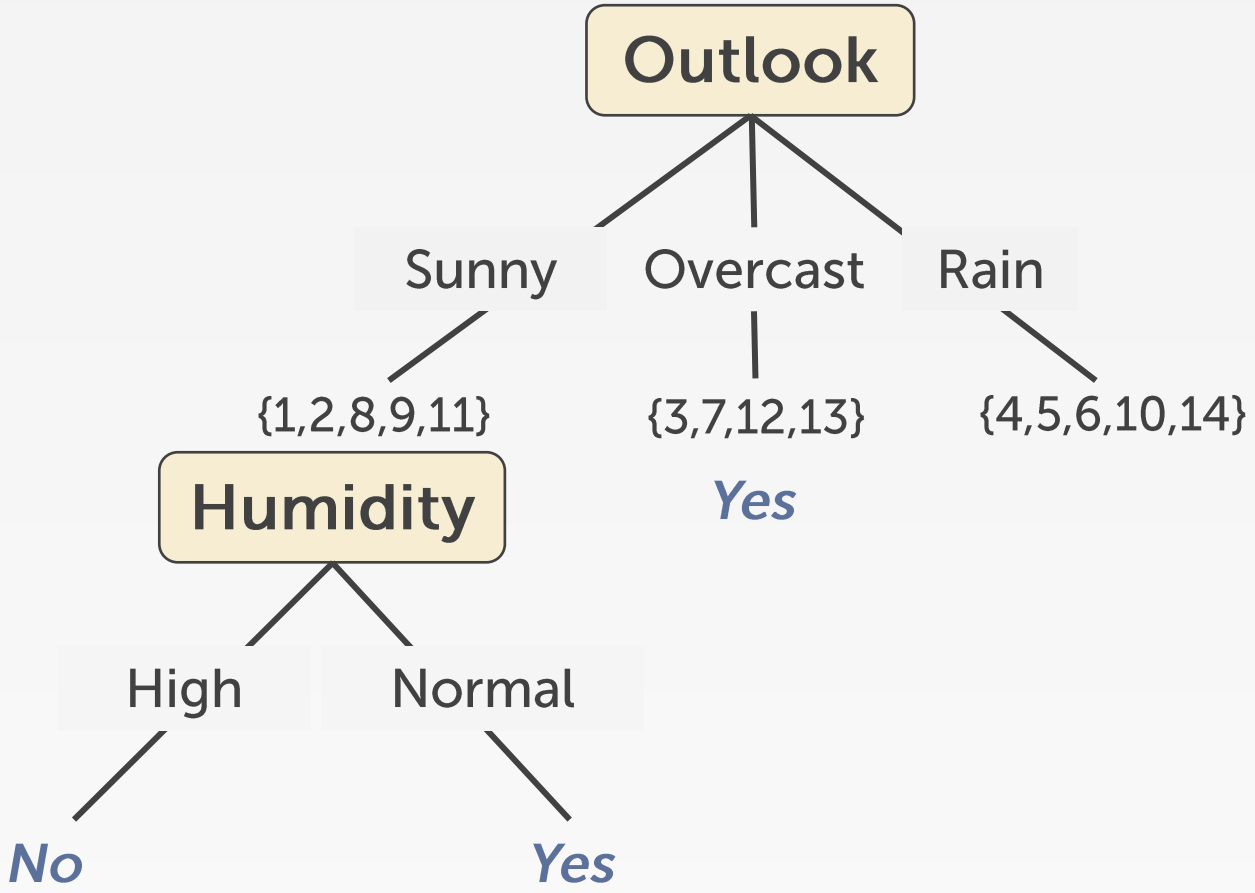
$$Gain(S_{sunny}, Temperature) = 0.97 - \frac{2}{5} \times 0 - \frac{1}{5} \times 0 - \frac{2}{5} \times 1 = 0.57$$

$$Gain(S_{sunny}, Humidity) = 0.97 - \frac{3}{5} \times 0 - \frac{2}{5} \times 0 = \mathbf{0.97}$$

$$Gain(S_{sunny}, Wind) = 0.97 - \frac{3}{5} \times 0.92 - \frac{2}{5} \times 1 = 0.02$$

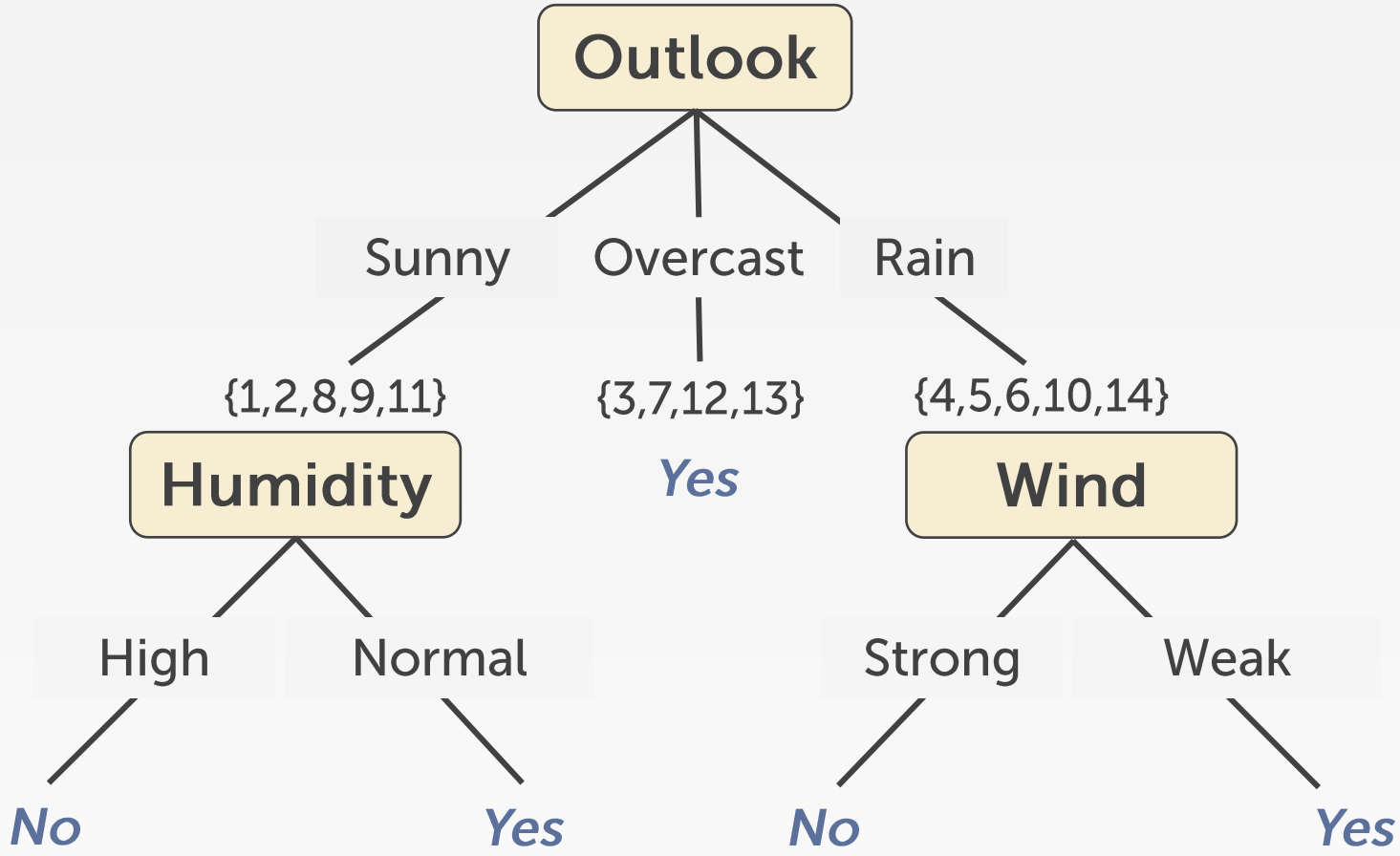
An Illustrative Example

	O	T	H	W	Play
1	S	H	H	W	No
2	S	H	H	S	No
8	S	M	H	W	No
9	S	C	N	W	Yes
11	S	M	N	S	Yes



An Illustrative Example

	O	T	H	W	Play
4	R	M	H	W	Yes
5	R	C	N	W	Yes
6	R	C	N	S	No
10	R	M	N	W	Yes
14	R	M	H	S	No



Gain Ratio (C4.5)

$$GainRatio(D, X_m) = \frac{Gain(D, X_m)}{IV(X_m)}$$

Intrinsic value (or split info):

$$IV(X_m) = -\sum_{v \in Value(X_m)} \frac{|D_v|}{|D|} \log_2 \frac{|D_v|}{|D|}$$

Comparison

Features	*ID3*	*C4.5*	*CART*
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression

Comparison

Features	*ID3*	*C4.5*	*CART*
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression

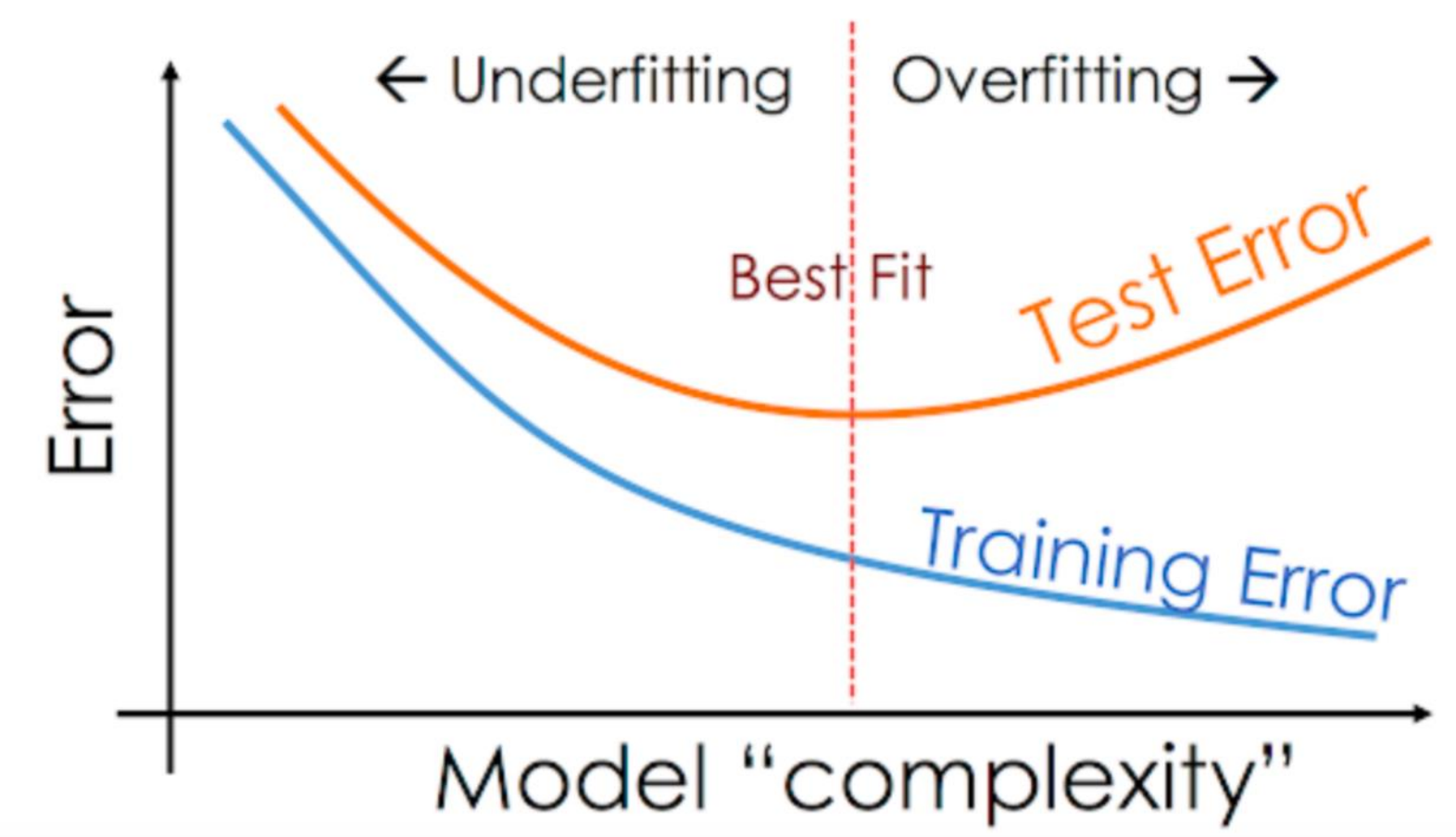
The smallest decision tree that correctly classifies all of the training examples is best

Occam's Razor: Prefer the simplest hypothesis that explains the data

Why Overfitting?

- Too much noise in the training data
  - 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 -)
- Too much variance in the training data
  - Training data is not a representative sample of the instance space
  - We split on features that are actually irrelevant
  - Too little training data

Overfitting

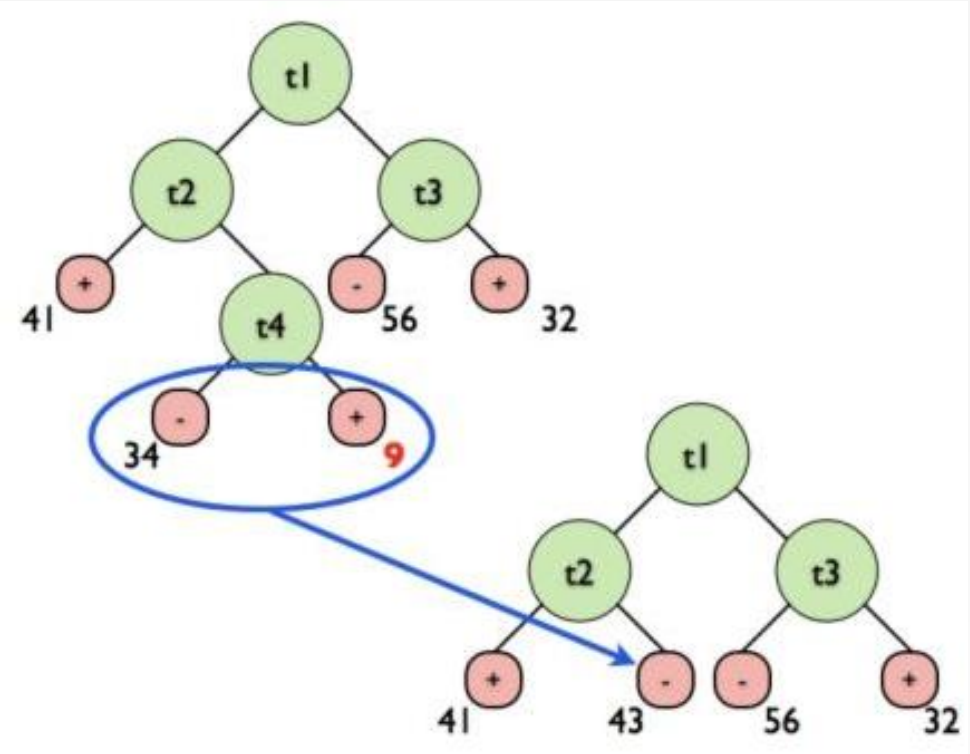


Avoid Overfitting

- How can we avoid overfitting?
  - Acquire more training data
  - Remove irrelevant attributes (manual process – not always possible)
  - Do not grow tree beyond some maximum depth
  - Do not split if splitting criterion (e.g. information gain) is below some threshold
  - Stop growing when data split is not statistically significant
  - Grow the full tree, then post-prune

Pruning a Tree

- Prune = remove leaves and assign majority labels of the parent to all items



Pruning

- Two basic approaches
  - **Pre-pruning** stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices
    - Set a depth cut-off (maximum tree depth) a priori
    - Set a minimum number of data points for each node
    - Stop growing if a split is not statistically significant
  - **Post-pruning** grow the full tree and then remove nodes that seem not to have sufficient evidence

Post-pruning

- Split data into training set and validation set
- Train a tree to classify training set as well as possible

- Do until further pruning reduces validation set accuracy:
  - 1. For each internal tree node, consider making it a leaf node (pruning the tree below it)
  - 2. Greedily choose the above pruning step that best improves error over validation set

- Produces smallest version of the most accurate pruned tree

Continuous Attributes

- Discretize real-values attributes into ranges: e.g., big, medium, small
- Alternatively, based on the thresholds: A<c vs. A>=c
- How to find the split with the highest gain? (Bi-partition in C4.5)

Temperature	40	48	60	72	80	90
Play Tennis	No	No	Yes	Yes	Yes	No

Missing Values

- Data D and attribute a
- $\widetilde{D}$ is the data that do not have missing values on a
- Value of a: $\{a^1, a^2, \dots, a^V\}$
- $\widetilde{D}^v$,$\widetilde{D}_k$ where $k = 1,2, \dots, |Y|$
- $\rho = \frac{\sum_{x \in \widetilde{D}} w_x}{\sum_{x \in D} w_x}$; $\widetilde{p_k} = \frac{\sum_{x \in \widetilde{D}_k} w_x}{\sum_{x \in \widetilde{D}} w_x}$; $\widetilde{r_v} = \frac{\sum_{x \in \widetilde{D}^v} w_x}{\sum_{x \in \widetilde{D}} w_x}$
- $Gain(D, a) = \rho \times Gain(\widetilde{D}, a) = \rho \times \left(Ent(\widetilde{D}) - \sum_{v=1}^{V} \widetilde{r_v} Ent(\widetilde{D}^v)\right)$

  $Ent(\widetilde{D}) = -\sum_{k=1}^{|Y|} \widetilde{p_k} \log_2 \widetilde{p_k}$
- *Split info* is calculated the same as before but with the missing values considered a separate state that an attribute can take.

Missing Value

Day	Outlook	Temp	Humidity	Windy	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	?	Hot	High	Strong	No
D3	?	?	High	?	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	?	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	?	Mild	High	?	No
D9	?	Cool	Normal	Weak	Yes
D10	?	?	Normal	?	Yes
D11	?	Mild	Normal	?	Yes
D12	Overcast	Mild	?	Strong	Yes
D13	Overcast	Hot	?	Weak	Yes
D14	Rain	Mild	High	Strong	No

Comparison

Features	*ID3*	*C4.5*	*CART*
Formula	Information Gain	Gain Ratio	Gini
Pruning	No	Yes	Yes
Type of data	Categorical	Categorical / Continuous	Categorical / Continuous
Missing Values	Can't	Can	Can
Prediction	Classification	Classification	Classification / Regression

CART (Regression)

- $R_1 = \{x|x \leq s\}, R_2 = \{x|x > s\}$
- $c_1 = \frac{1}{N_1}\sum_{x_i \in R_1} y_i \, , c_2 = \frac{1}{N_2}\sum_{x_i \in R_2} y_i$
- $\min_{j,s}\left[\min_{c_1}\sum_{x_i \in R_1}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i \in R_2}(y_i - c_2)^2\right]$
- j^{th} variable; partition s

Model Selection

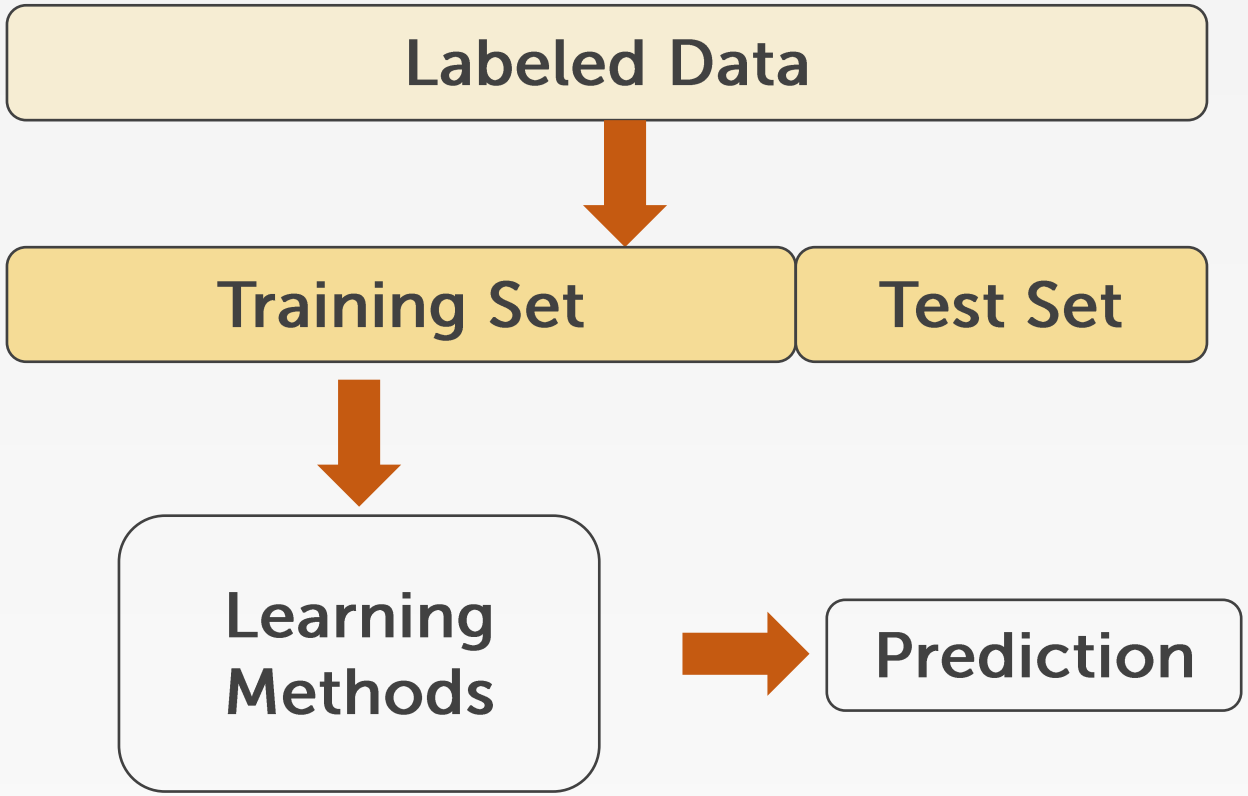
Model Selection

- Machine Learning
  - *Def: (loosely)* a **model** defines the hypothesis space over which learning performs its search
  - *Def*: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
  - Def: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
  - Def: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does not select

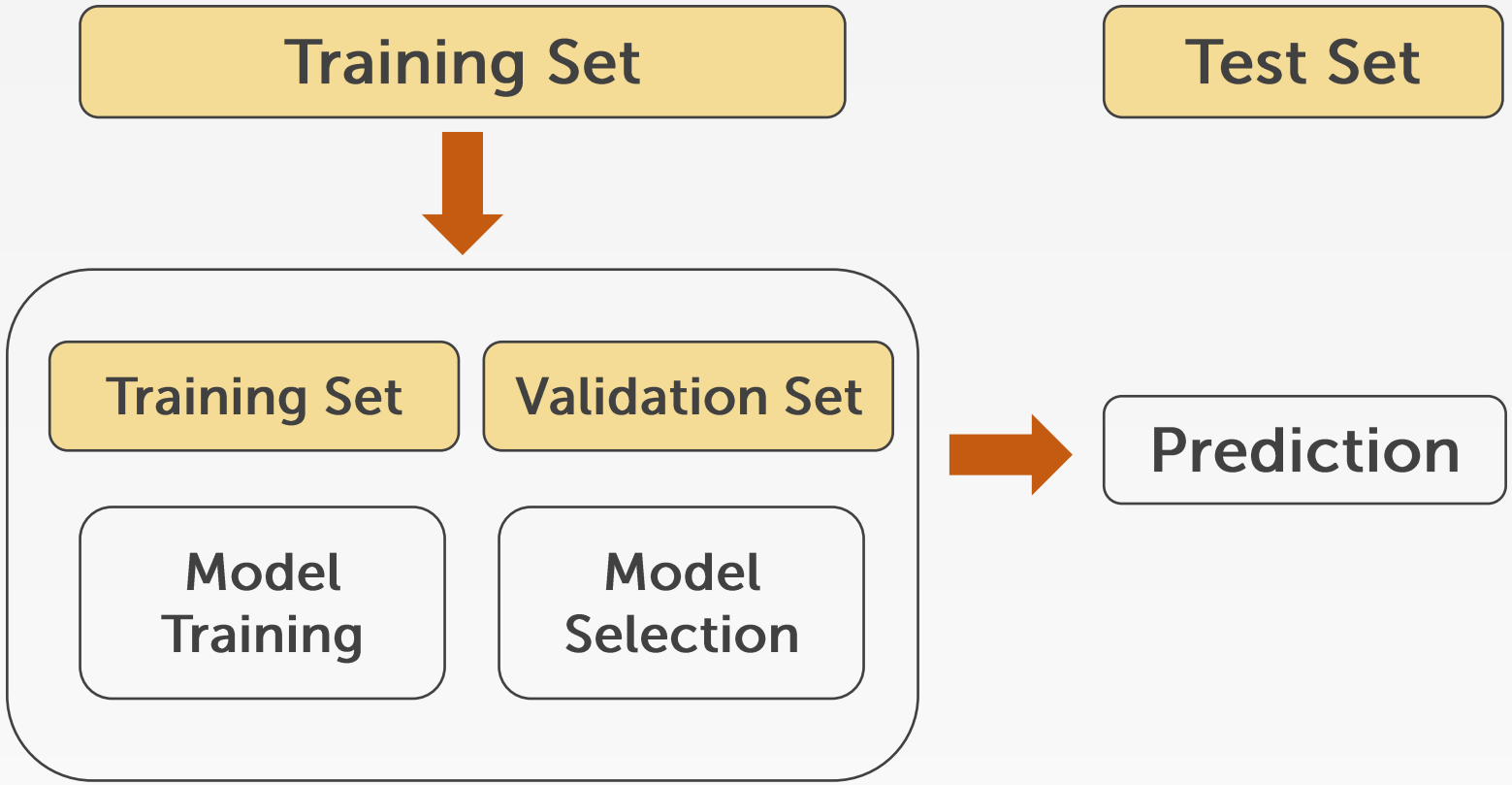
- Example
  - model = set of all possible trees, possibly restricted by some hyperparameters (e.g. max depth)
  - parameters = structure of a specific decision tree
  - learning algorithm = ID3, CART, etc.
  - hyperparameters = max depth, threshold for splitting criterion, etc.

Cross Validation

Train a Supervised Model



Select a Model

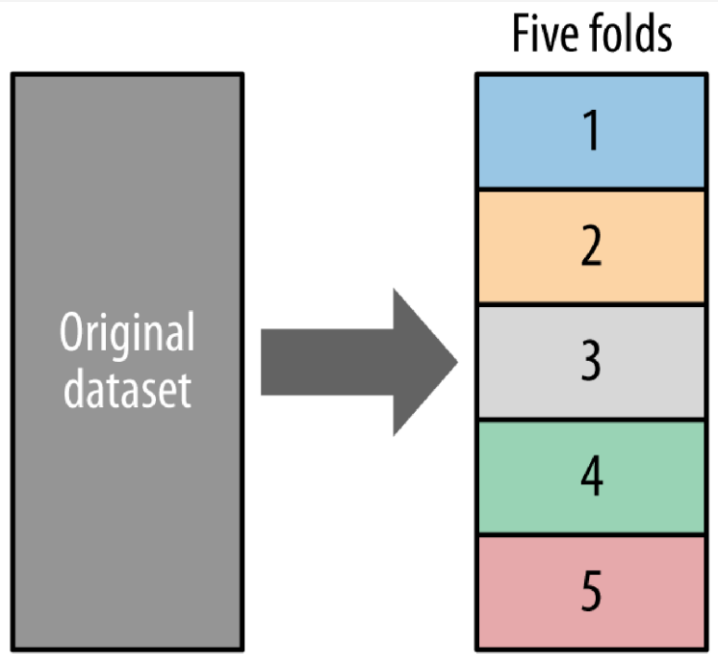


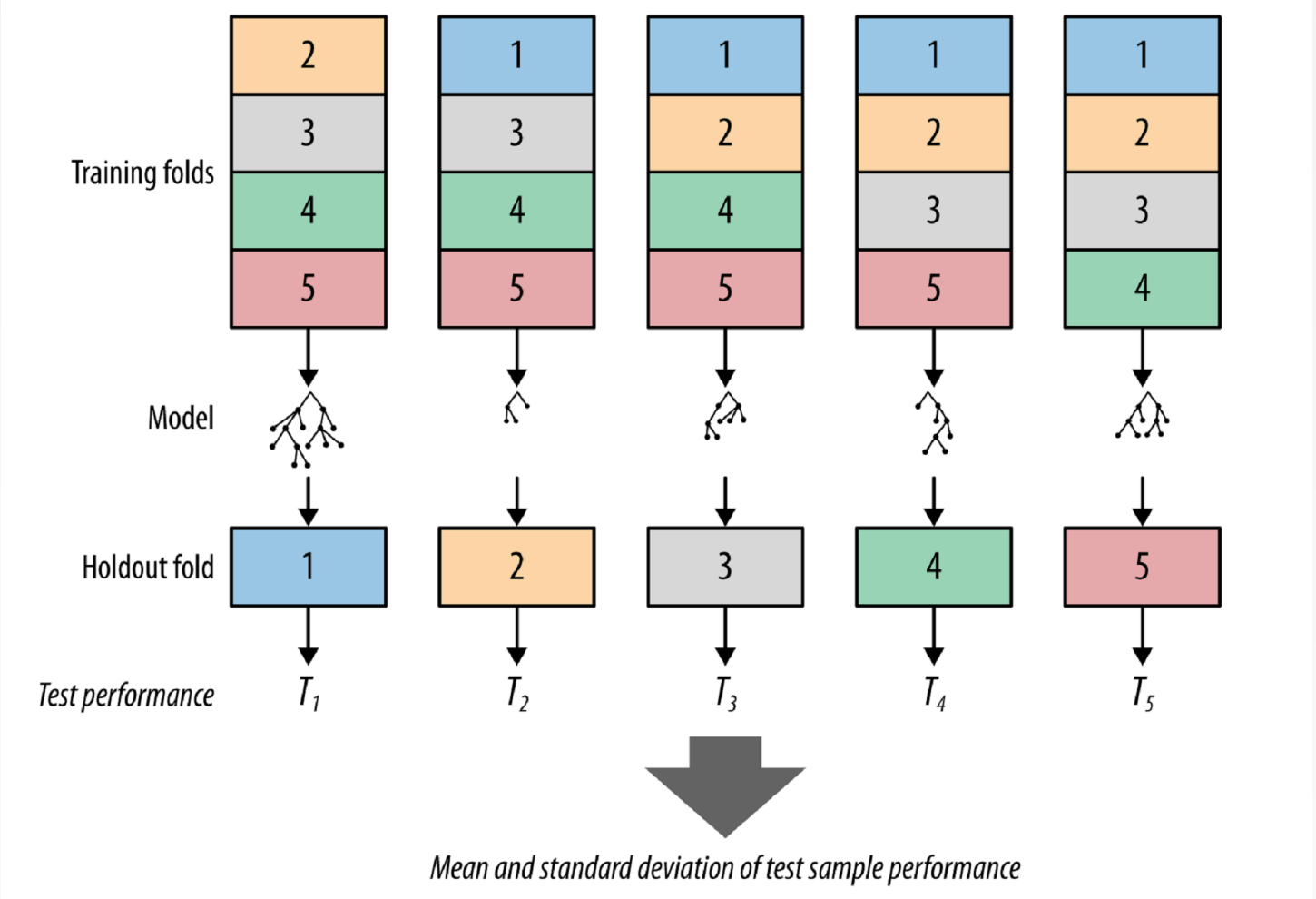
Limitation of a Single Partition

- We may not have enough data to have both sufficiently large training and test set:
  - A large test set: more reliable estimate of prediction accuracy
  - A large training set: more representative for learning process

- Solution: Cross Validations
  - **Sometime also called "N-fold Cross-Validation"**
    - **N: number of folds used**

N-fold Cross Validation





Grid Search

- Approaches to hyperparameter tuning:
  - **Grid search**
  - Random search
  - Bayesian optimization
  - ...

Grid Search

	C=0.001	C=0.01	C=10
$\alpha = 0.001$	M (C=0.001, $\alpha = 0.001$)	M (C=0.01, $\alpha = 0.001$)	M (C=10, $\alpha = 0.001$)
$\alpha = 0.1$	M (C=0.001, $\alpha = 0.1$)	M (C=0.01, $\alpha = 0.1$)	M (C=10, $\alpha = 0.1$)
…			
$\alpha = 100$	M (C=0.001, $\alpha = 100$)	M (C=0.01, $\alpha = 100$)	M (C=10, $\alpha = 100$)

Grid Search with CV

- Manually set a grid of discrete hyperparameter values.
- Set a metric for scoring model performance.
- Search exhaustively through the grid.
- For each set of hyperparameters, evaluate each model's CV score.
- The optimal hyperparameters are those of the model achieving the best CV score.

Tuning is expensive

- Hyperparameter tuning:
  - Computationally expensive,
  - Sometime leads to very slight improvement

- Weight the impact of tuning on the whole project

Model Evaluation (Binary Classification)

Accuracy

Number of correct predictions/Data size

Is Accuracy Enough?

Is Accuracy Enough?

- Potential Issues:
  - There is a large class skew
    - E.g., Is 98% accuracy good if 97% of the instances are negative?

Is Accuracy Enough?

- Potential Issues:
  - There is a large class skew
    - E.g., Is 98% accuracy good if 97% of the instances are negative?
  - There are differential misclassification costs
    - E.g., Medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

Is Accuracy Enough?

- Potential Issues:
  - There is a large class skew
    - E.g., Is 98% accuracy good if 97% of the instances are negative?
  - There are differential misclassification costs
    - E.g., Medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
- Require better and more evaluation measurements!

Confusion Matrix

A Two-class Problem

		Predicted class: Positive	Predicted class: Negative
Actual class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Additional Metrics

Precision % of those cases the model classified as positive that are correct

$$\frac{TP}{TP + FP}$$

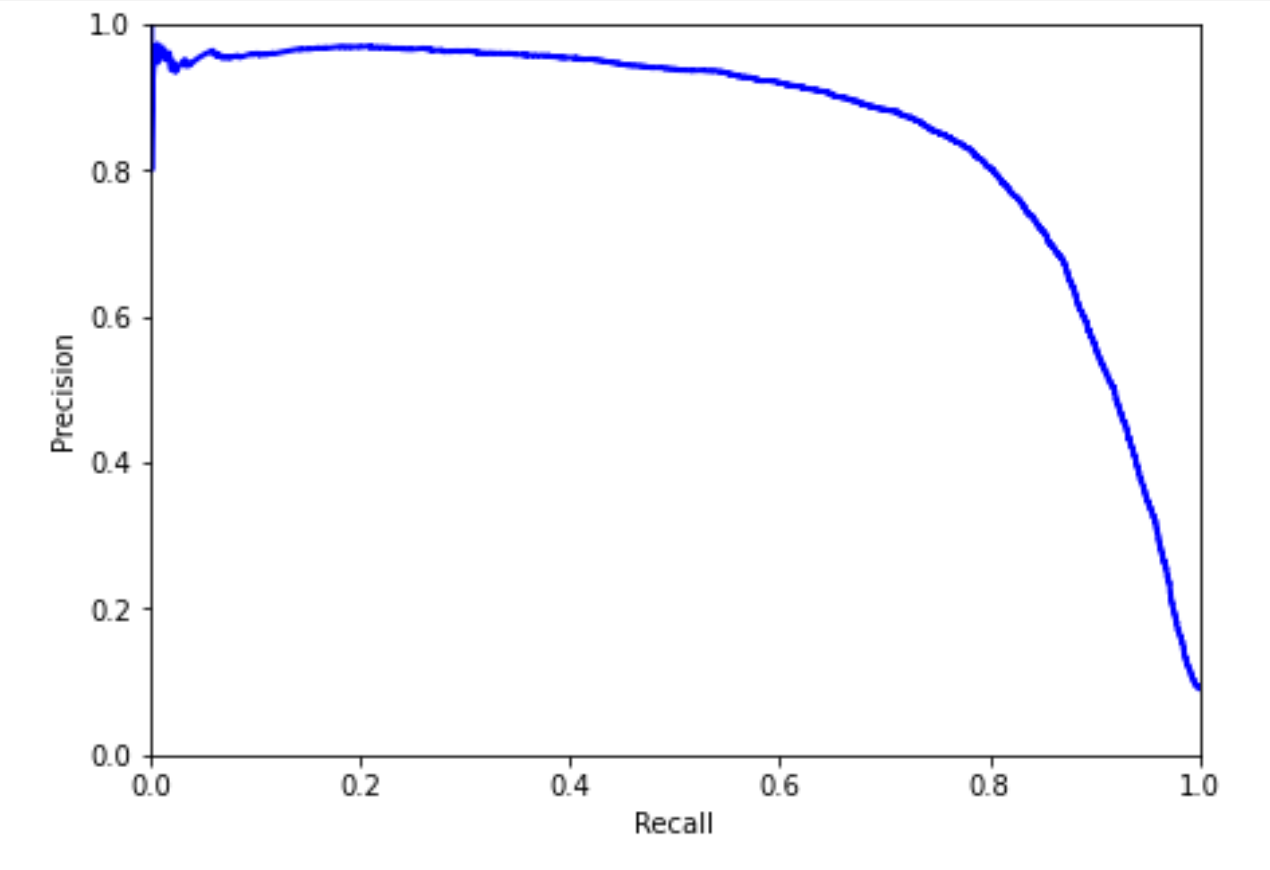
Measures model's accuracy for those it classified as the class of interest

Recall (sensitivity): % of actual positive class (class of interest) correctly classified

$$\frac{TP}{TP + FN}$$

Measures how well the model detect the class of interest

Precision-Recall Curve



Additional Metrics

Precision % of those cases the model classified as positive that are correct

$$\frac{TP}{TP + FP}$$

Measures model's accuracy for those it classified as the class of interest

Recall (sensitivity): % of actual positive class (class of interest) correctly classified

$$\frac{TP}{TP + FN}$$

Measures how well the model detect the class of interest

Specificity % of negative class correctly classified

$$\frac{TN}{TN + FP}$$

Measures how well the model rules out the other class correctly

F_1 score

$$2 \times (Precision \times recall)/(Precision + recall)$$

Practice

- Scenario 1: Mining systems to identify sites
  - Classifier 1: low recall, high precision
  - Classifier 2: high recall, low precision

Practice

- Scenario 1: Mining systems to identify sites
  - Classifier 1: low recall, high precision
  - Classifier 2: high recall, low precision

- Scenario 2: Identify terrorists
  - Classifier 1: low recall, high precision
  - Classifier 2: high recall, low precision

Illustrative Examples

Model I

	Predicted Class	
Actual Class	1 (Yes)	0 (No)
(Yes) 1	500	0
(No) 0	200	300

Model II

	Predicted Class	
Actual Class	1 (Yes)	0 (No)
(Yes) 1	300	200
(No) 0	0	500

An Illustrative Example I

	Predicted Class	
Actual Class	1 (Yes)	0 (No)
(Yes) 1	500	0
(No) 0	200	300

Sensitivity (recall) – how well it detects class 1

Specificity– how well it rules out class 0

Precision– accuracy when identifying class 1

F_1 – balance between precision and recall

An Illustrative Example I

$$Accuracy = \frac{500 + 300}{1000} = 80\%$$

	Predicted Class	
Actual Class	1 (Yes)	0 (No)
(Yes) 1	500	0
(No) 0	200	300

Sensitivity (recall) – how well it detects class 1

Specificity– how well it rules out class 0

Precision– accuracy when identifying class 1

F_1 – balance between precision and recall

An Illustrative Example I

$$Accuracy = \frac{500 + 300}{1000} = 80\%$$

	Predicted Class	
Actual Class	1 (Yes)	0 (No)
(Yes) 1	500	0
(No) 0	200	300

Sensitivity (recall) – how well it detects class 1 — 500/(500+0) = 1.0

Specificity– how well it rules out class 0 — 300/(200+300) = 0.6

Precision– accuracy when identifying class 1 — 500/(500+200) = 0.7143

F_1 – balance between precision and recall

$$2 \times \frac{0.7143 * 1}{0.7143 + 1} = 0.8333$$

An Illustrative Example II

$$Accuracy = \frac{500 + 300}{1000} = 80\%$$

	Predicted Class	
Actual Class	1 (Yes)	0 (No)
(Yes) 1	300	200
(No) 0	0	500

Sensitivity (recall) – how well it detects class 1 — 300/(300+200) = 0.6

Specificity– how well it rules out class 0 — 500/500 = 1

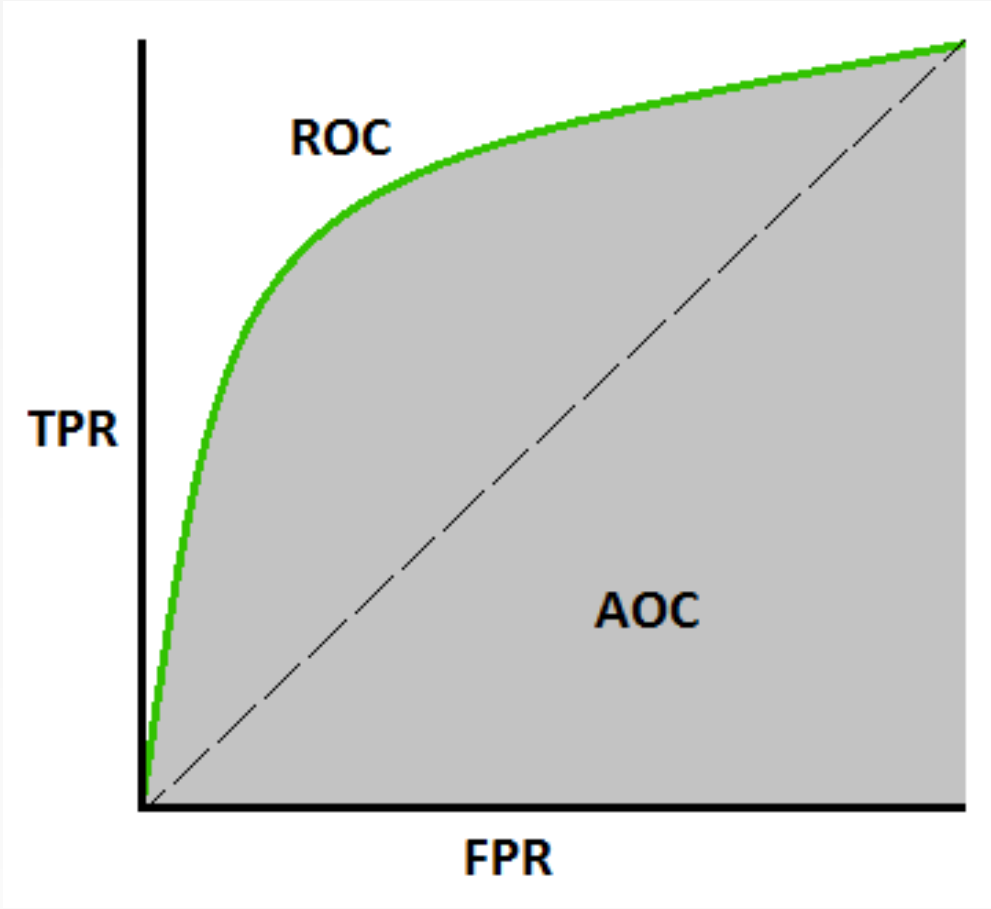
Precision– accuracy when identifying class 1 — 300/300 = 1

F_1 – balance between precision and recall — $2 \times \frac{0.6 * 1}{0.6 + 1} = 0.75$

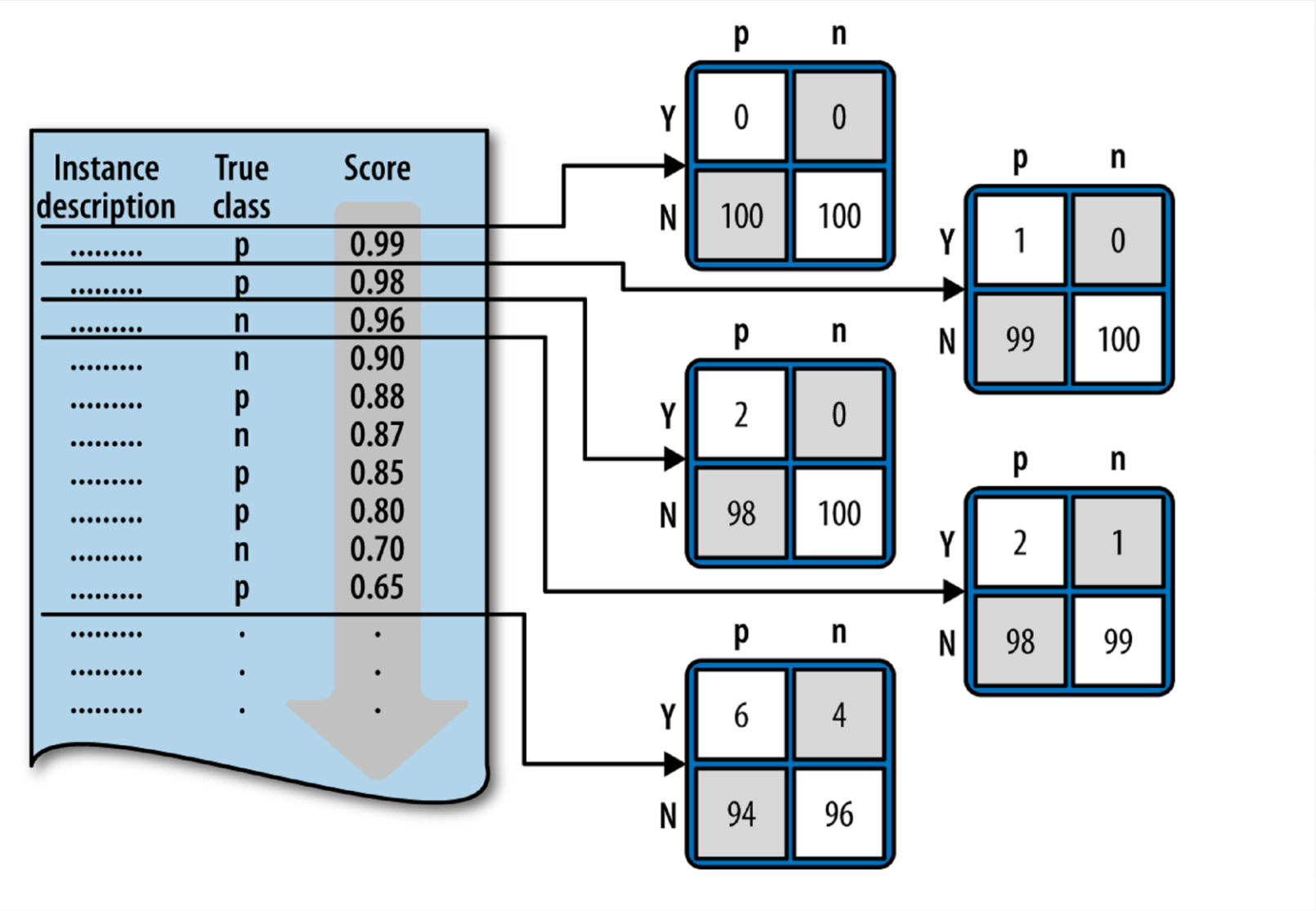
ROC

ROC Curves

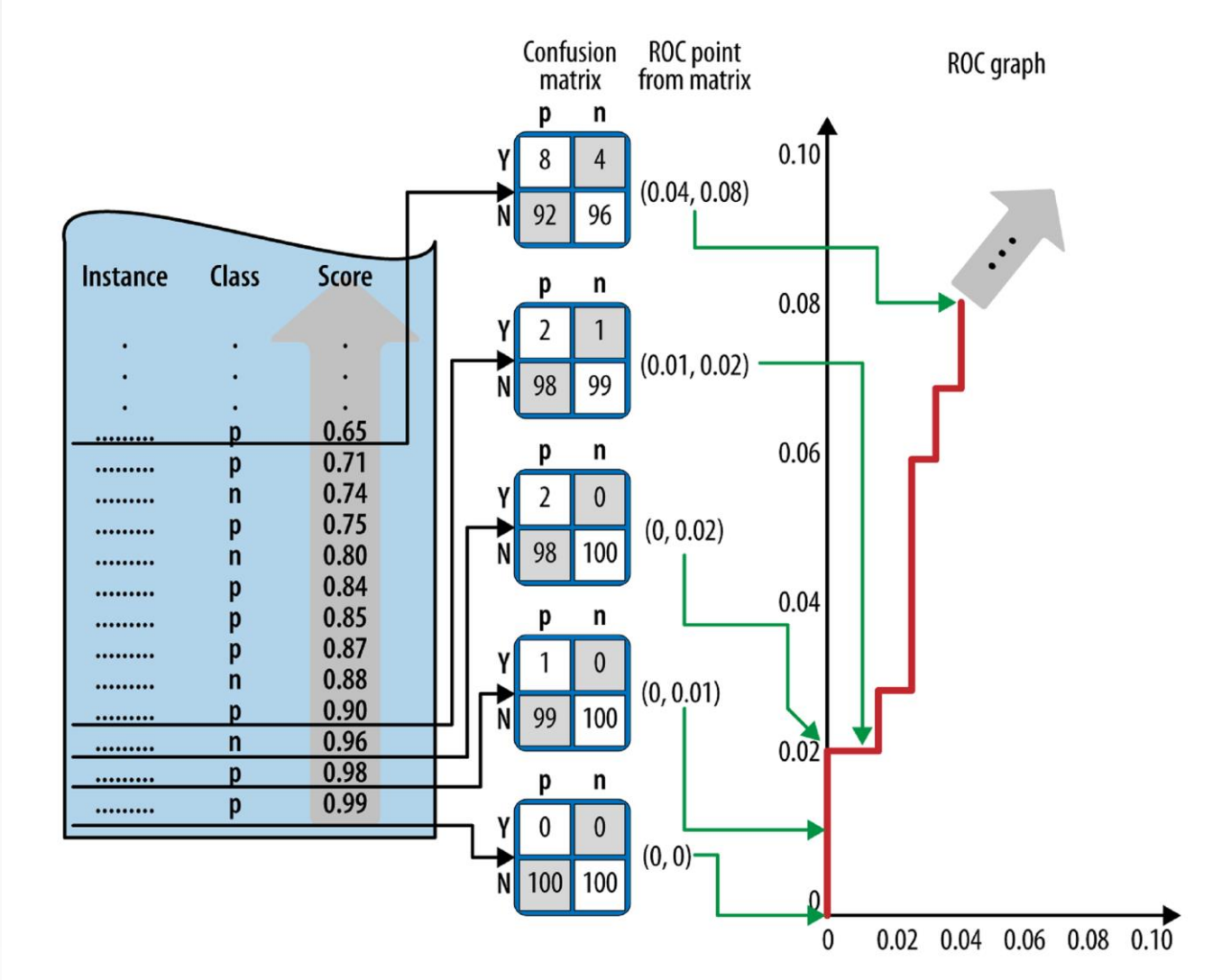
- A Receiver Operating Characteristic (ROC) curve plots the TP-rate (sensitivity) vs. the FP-rate (1-specificity) as a threshold on the confidence of an instance being positive is varied



Generate a ROC curve



Generate a ROC curve



Algorithm Detail

- sort test-set predictions according to probability that each instance is positive

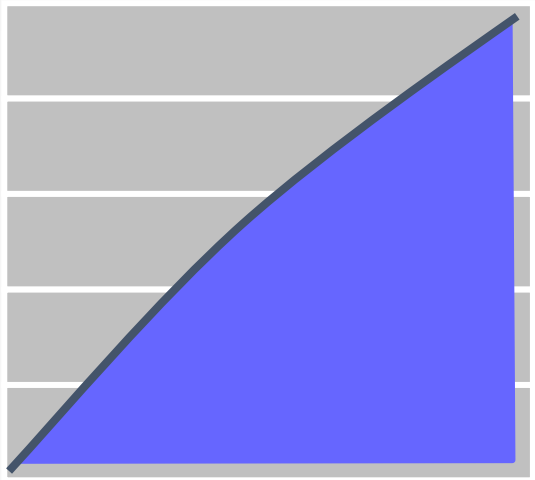
- step through sorted list from high to low probability
  - locate a threshold between instances with opposite classes (keeping instances with the same confidence value on the same side of threshold)
  - compute TPR, FPR for instances above threshold
  - output (FPR, TPR) coordinate

Area Under the ROC Curve (AUC)

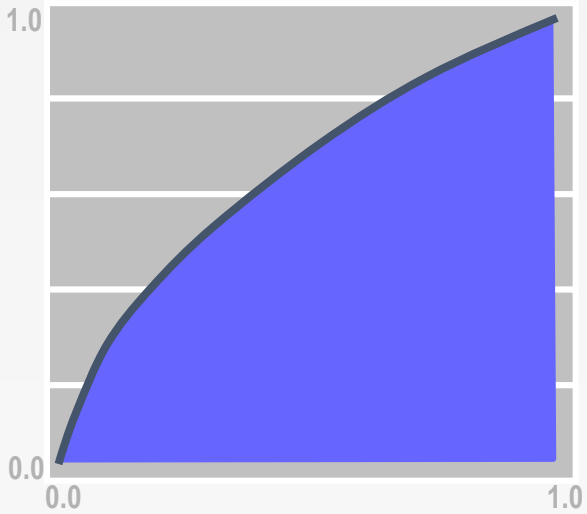
- The area under a classifier's curve expressed as a fraction of the unit square

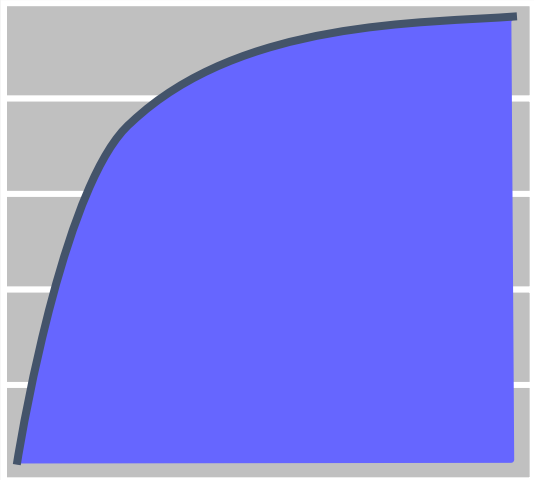
- The AUC is useful when a single number is needed to summarize performance
  - A ROC curve provides more information than its area

AUC



weak model
ROC Index < 0.6





strong model
ROC Index > 0.7