



# Data Mining

## Classification – Basic Concepts

---

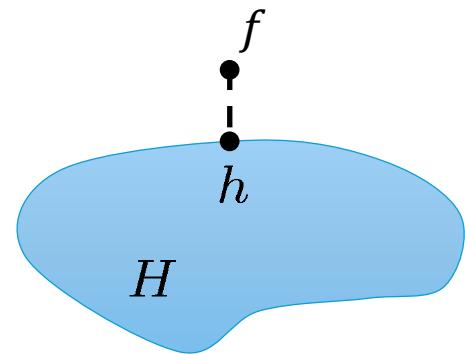


# Topics

- **Introduction**
- Decision Trees
  - Overview
  - Tree Induction
- Overfitting and other Practical Issues
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Feature Selection

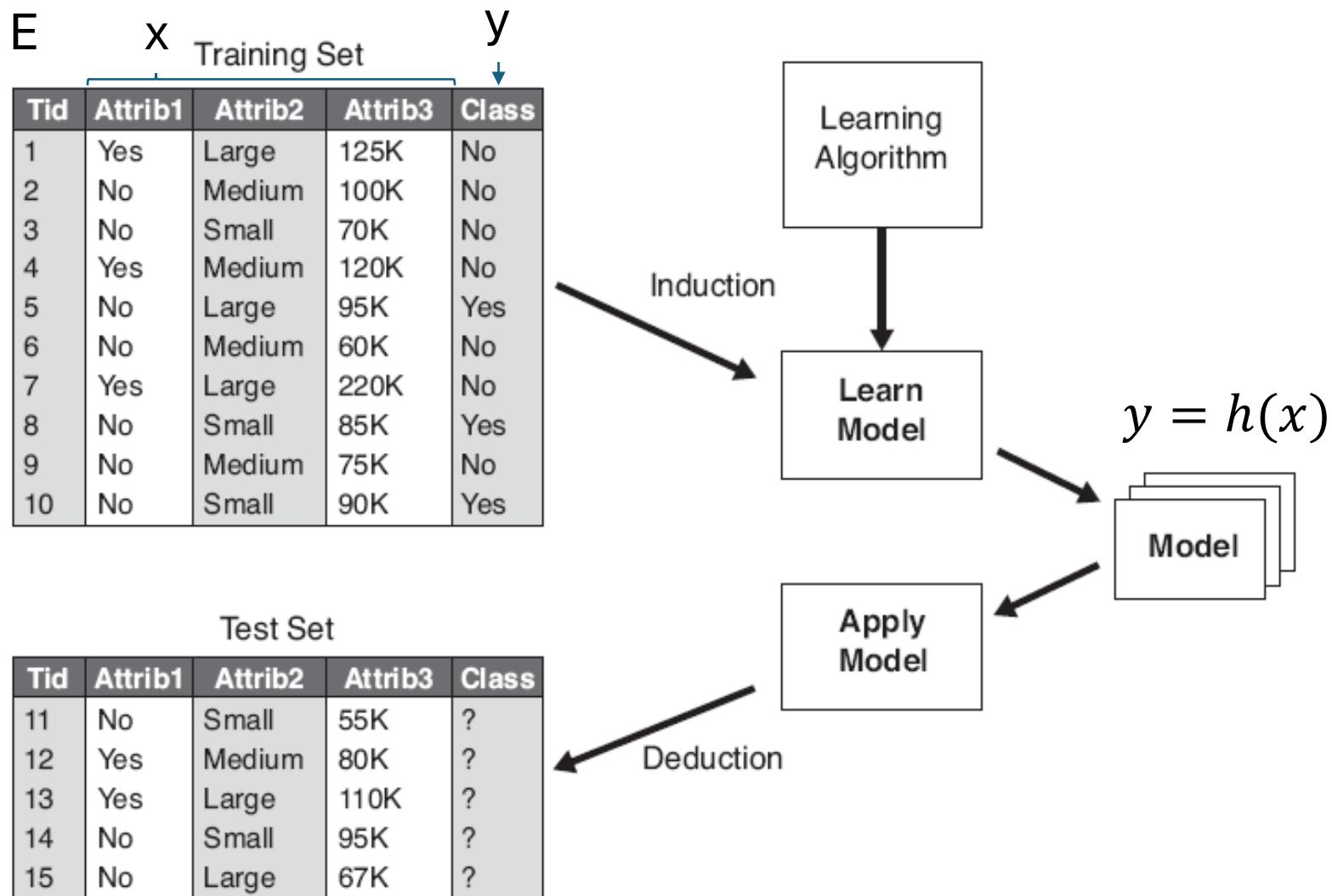
# Supervised Learning – Learning from Examples

- Examples
  - Input-output pairs:  $E = (x_1, y_1), \dots, (x_i, y_i), \dots, (x_N, y_N)$ .
  - We assume that the examples are produced iid (with noise and errors) from a target function  $y = f(x)$ .
- Learning problem
  - Given a hypothesis space  $H$
  - Find a hypothesis  $h \in H$  such that  $\hat{y}_i = h(x_i) \approx y_i$
  - That is, we want to approximate  $f$  by  $h$  using  $E$ .
- Includes
  - **Regression** (outputs = real numbers). Goal: Predict the number accurately.  
E.g.,  $x$  is a house and  $f(x)$  is its selling price.
  - **Classification** (outputs = class labels). Goal: Assign new records to a class.  
E.g.,  $x$  is an email and  $f(x)$  is spam / ham



You already know linear regression. We focus on Classification.

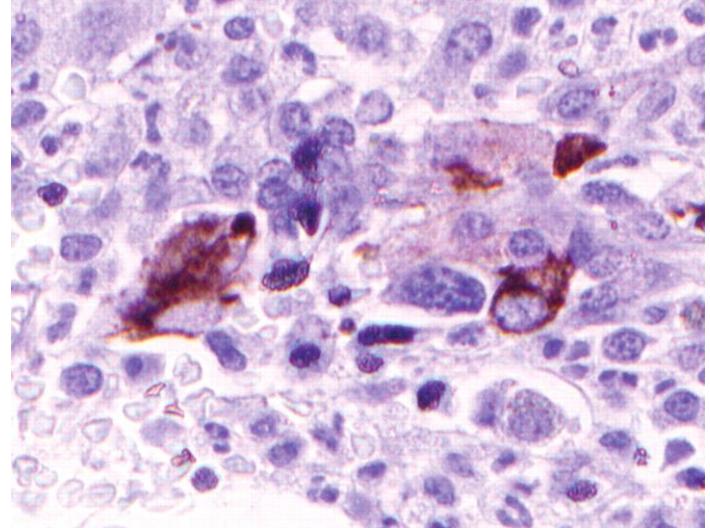
# Illustrating Classification Task

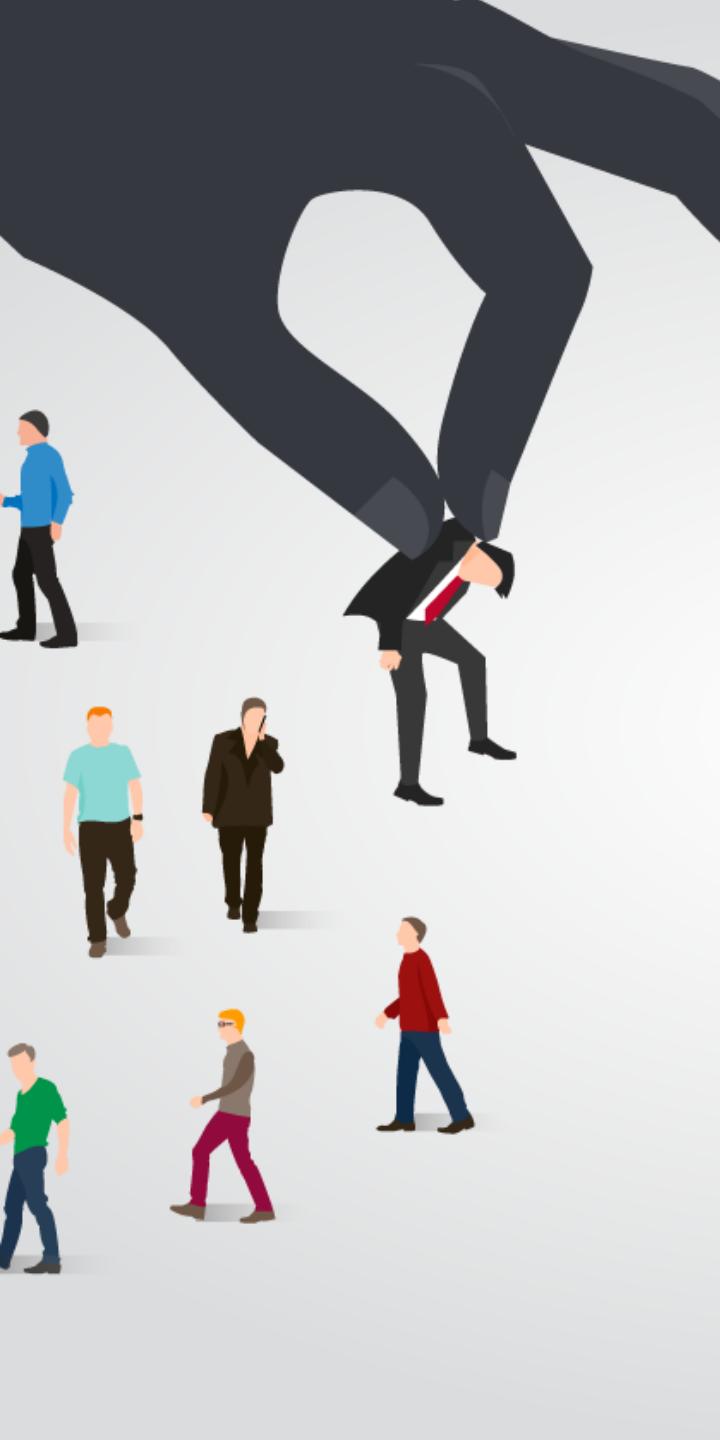


# Examples of Classification Task

---

- Predicting tumor cells as benign or malignant.
- Classifying credit card transactions as legitimate or fraudulent.
- Categorizing news stories as finance, weather, entertainment, sports, etc.



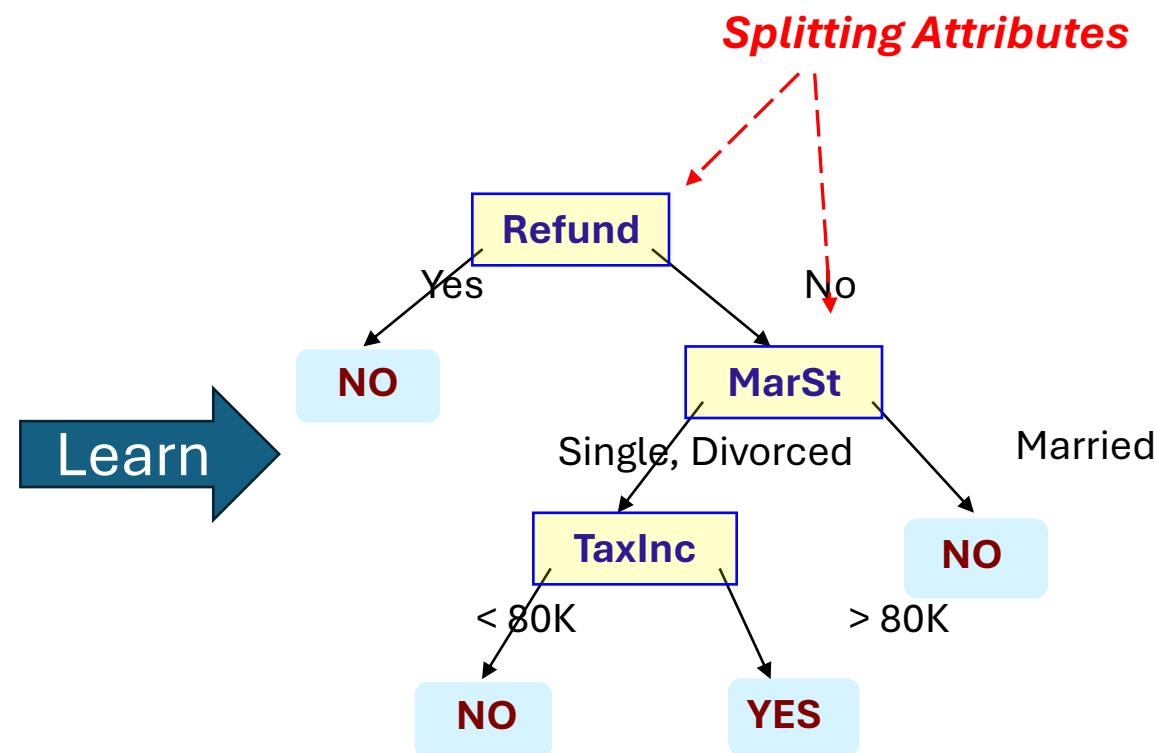


# Topics

- Introduction
- **Decision Trees**
  - Overview
  - Tree Induction
- Overfitting and other Practical Issues
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Feature Selection

# Example of a Decision Tree

					categorical	categorical	continuous	class
Tid	Refund	Marital Status	Taxable Income	Cheat				
1	Yes	Single	125K	No				
2	No	Married	100K	No				
3	No	Single	70K	No				
4	Yes	Married	120K	No				
5	No	Divorced	95K	Yes				
6	No	Married	60K	No				
7	Yes	Divorced	220K	No				
8	No	Single	85K	Yes				
9	No	Married	75K	No				
10	No	Single	90K	Yes				

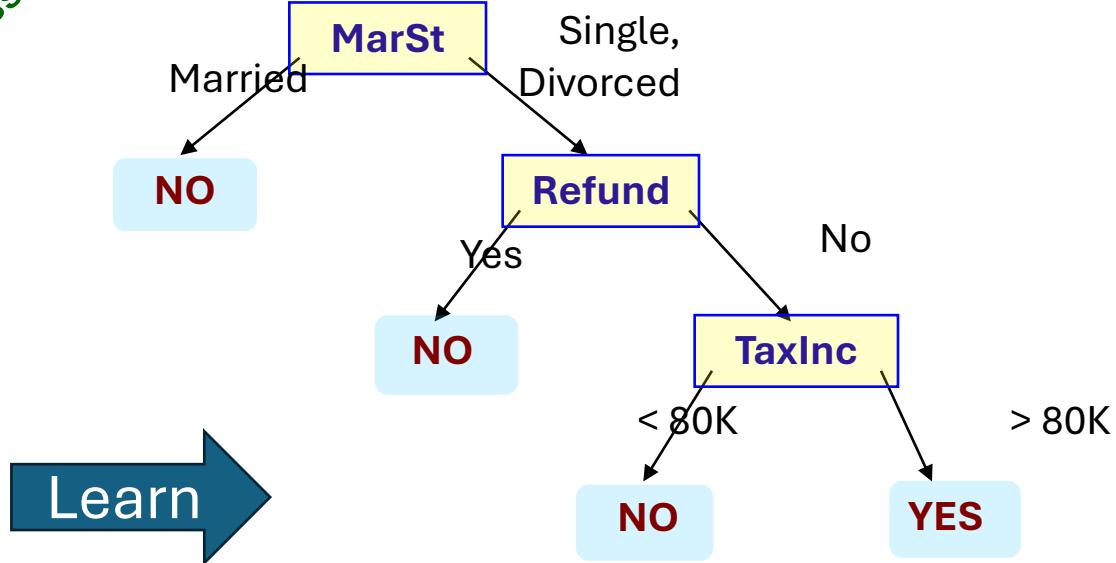


Training Data

Model: Decision Tree

# Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	class	
				categorical	categorical
1	Yes	Single	125K	No	
2	No	Married	100K	No	
3	No	Single	70K	No	
4	Yes	Married	120K	No	
5	No	Divorced	95K	Yes	
6	No	Married	60K	No	
7	Yes	Divorced	220K	No	
8	No	Single	85K	Yes	
9	No	Married	75K	No	
10	No	Single	90K	Yes	



Learn

There could be more than one tree that fits the same data!

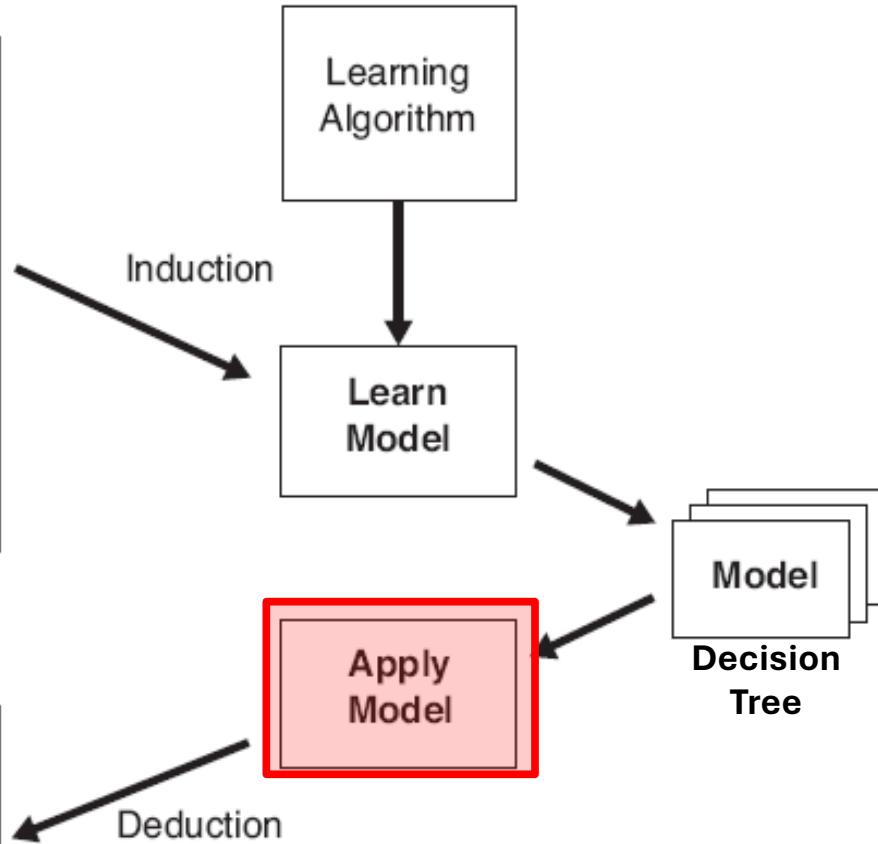
# Decision Tree: Deduction

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

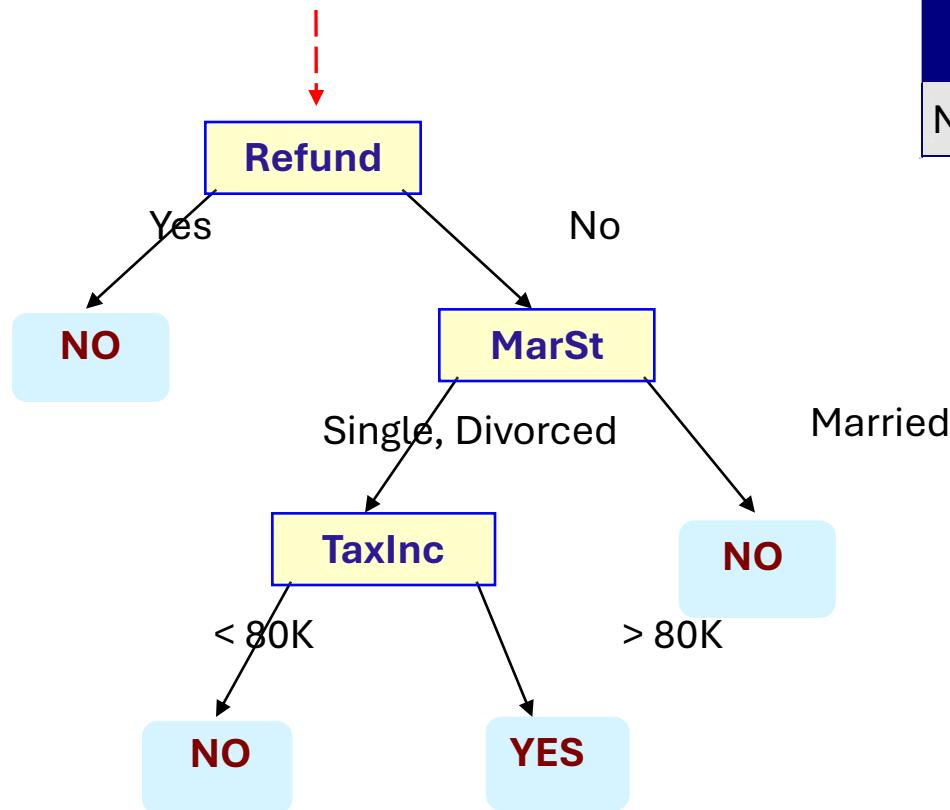
Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?



# Apply Model to Test Data

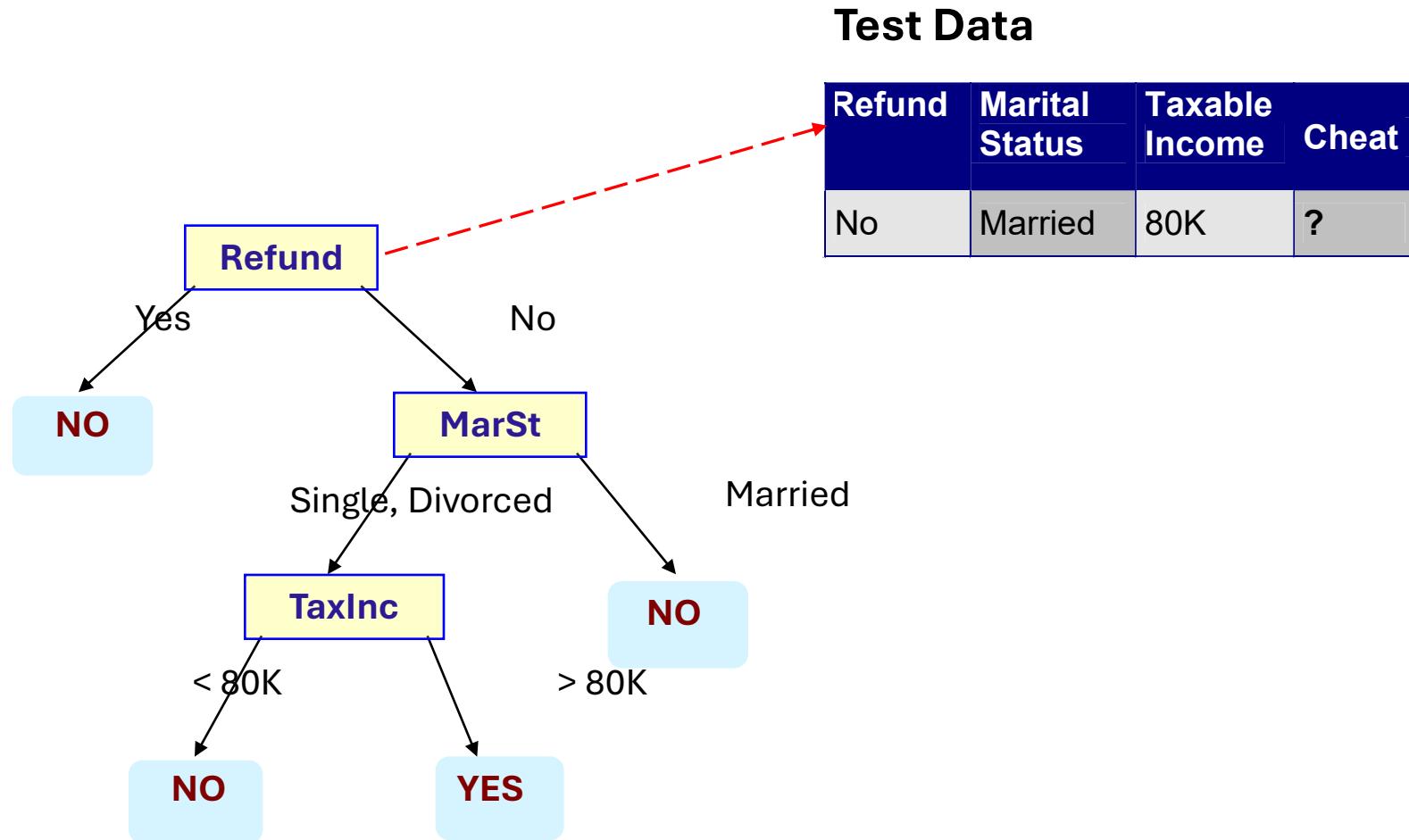
Start from the root of tree.



## Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

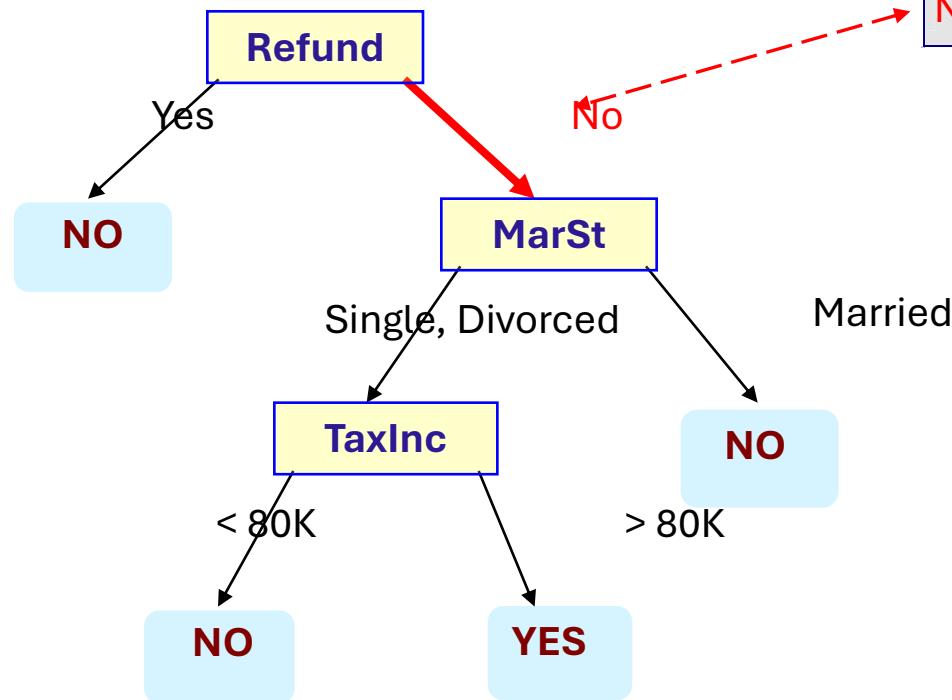
# Apply Model to Test Data



# Apply Model to Test Data

Test Data

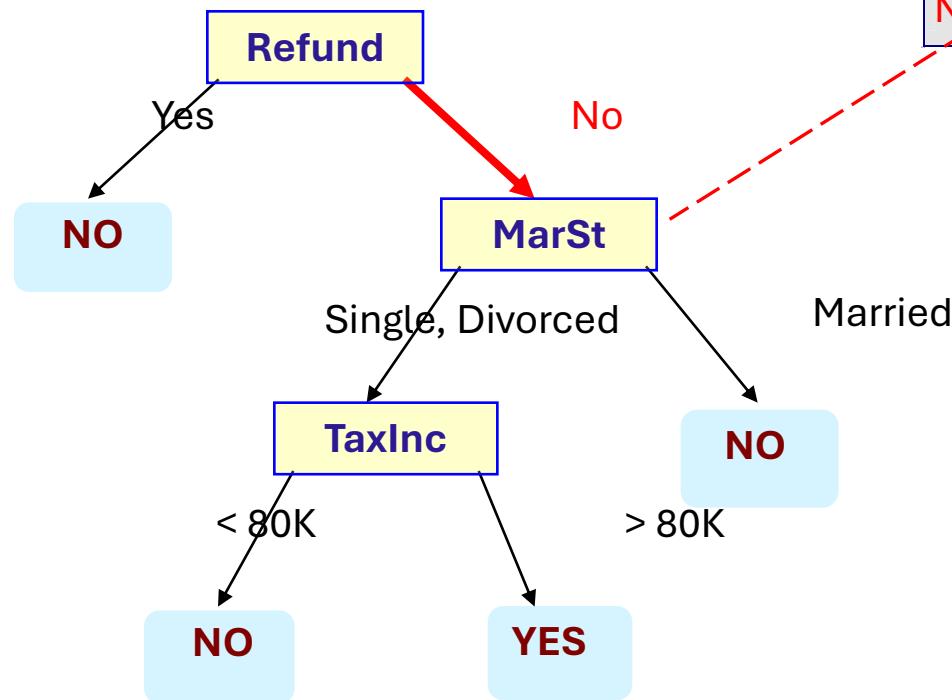
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

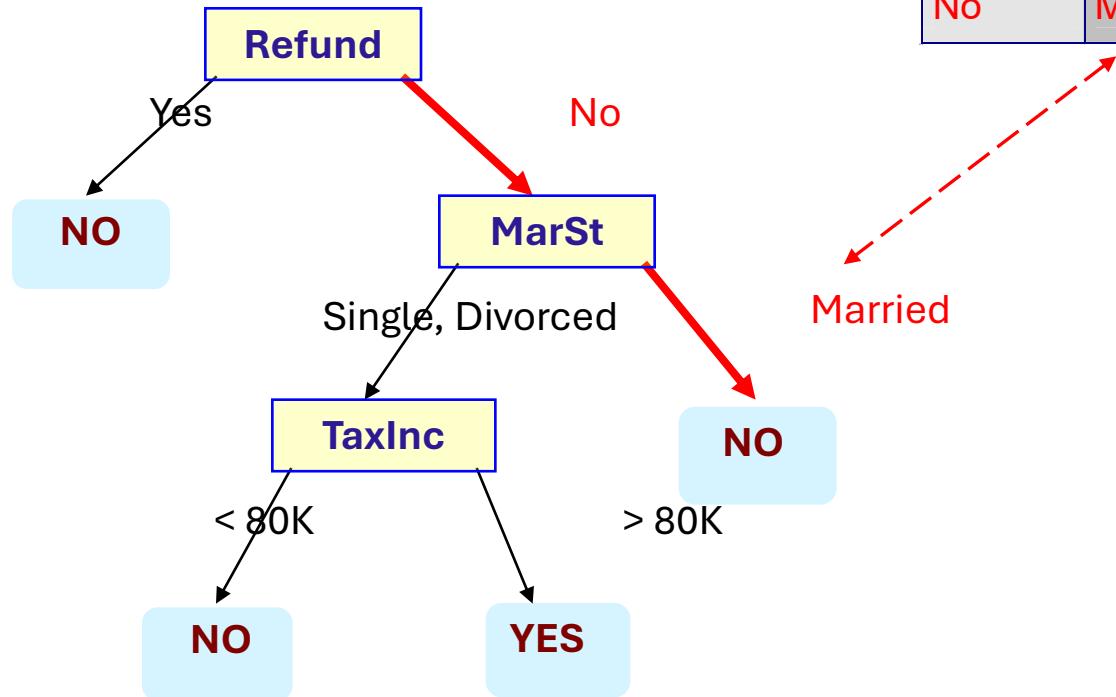
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



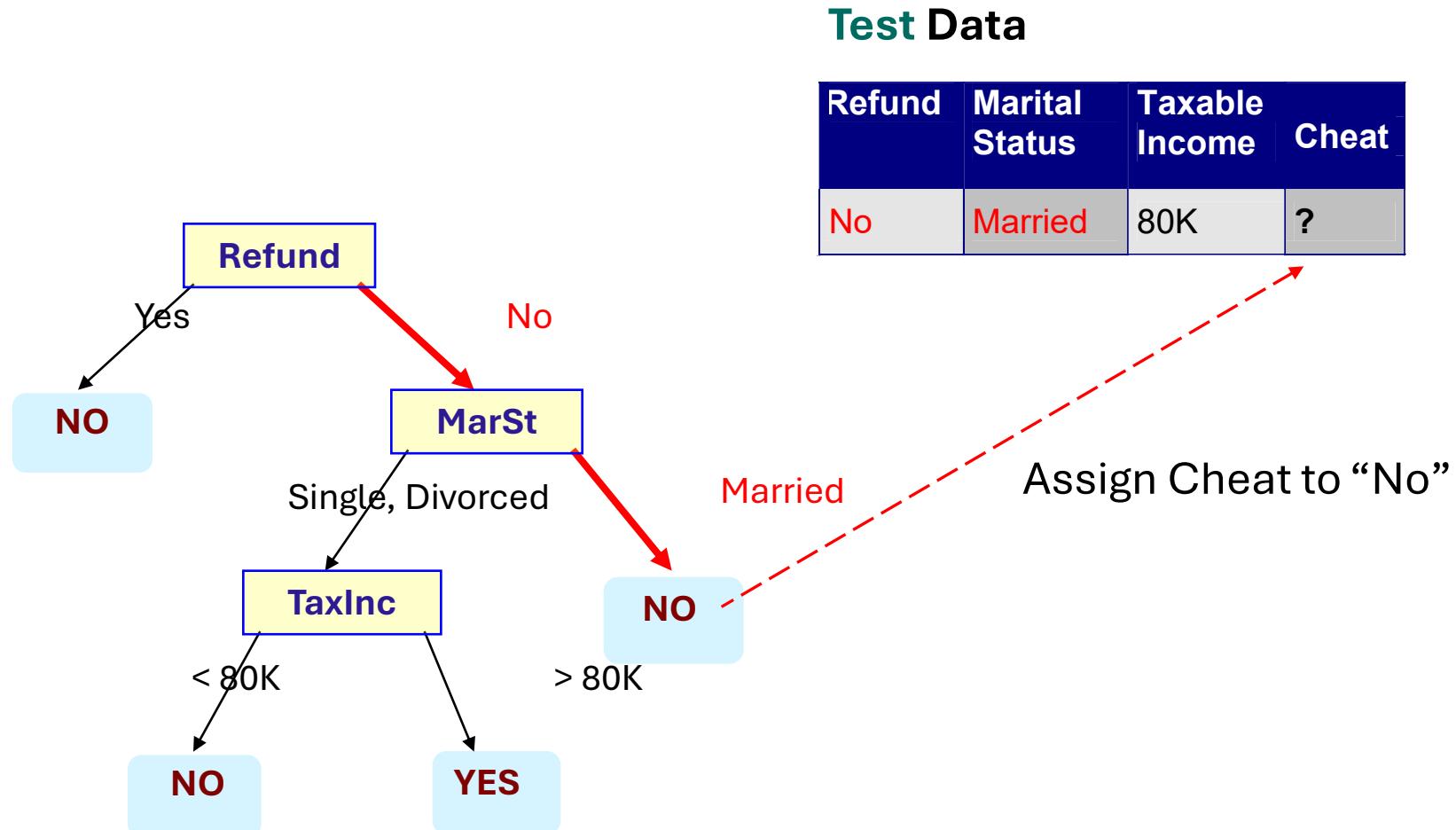
# Apply Model to Test Data

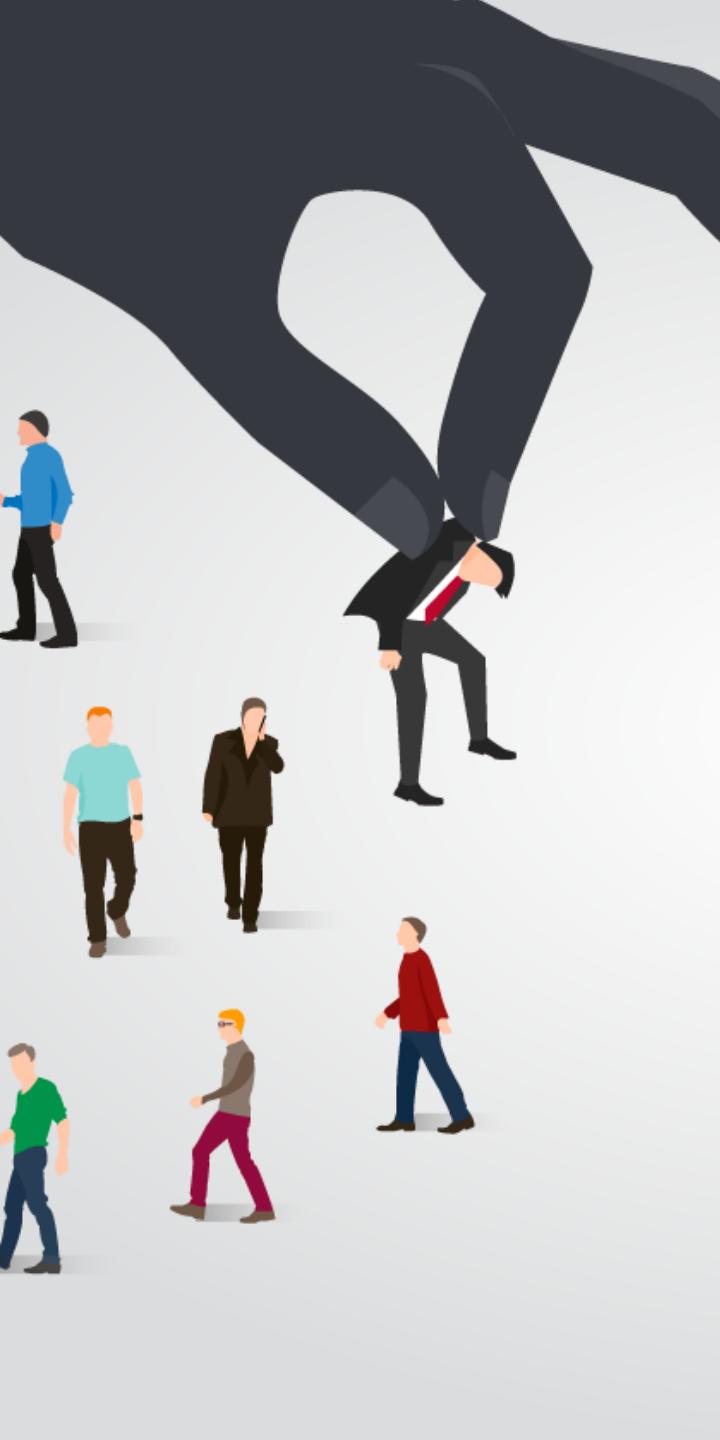
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data





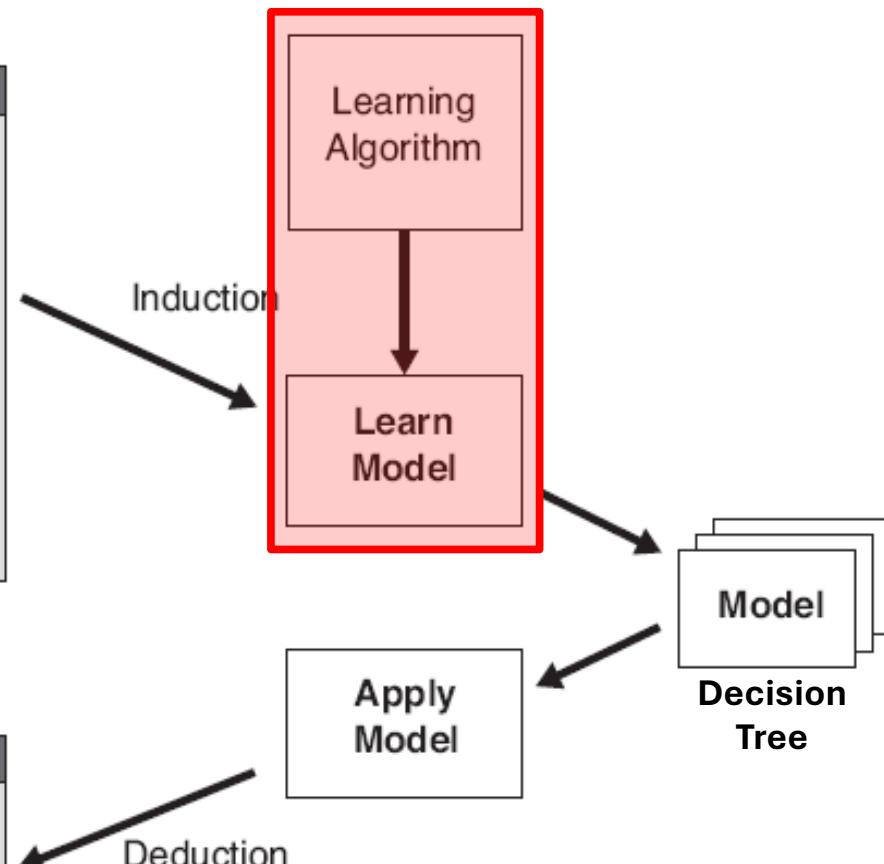
# Topics

- Introduction
- Decision Trees
  - Overview
  - **Tree Induction**
- Overfitting and other Practical Issues
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Feature Selection

# Decision Tree: Induction

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?



# Decision Tree Induction

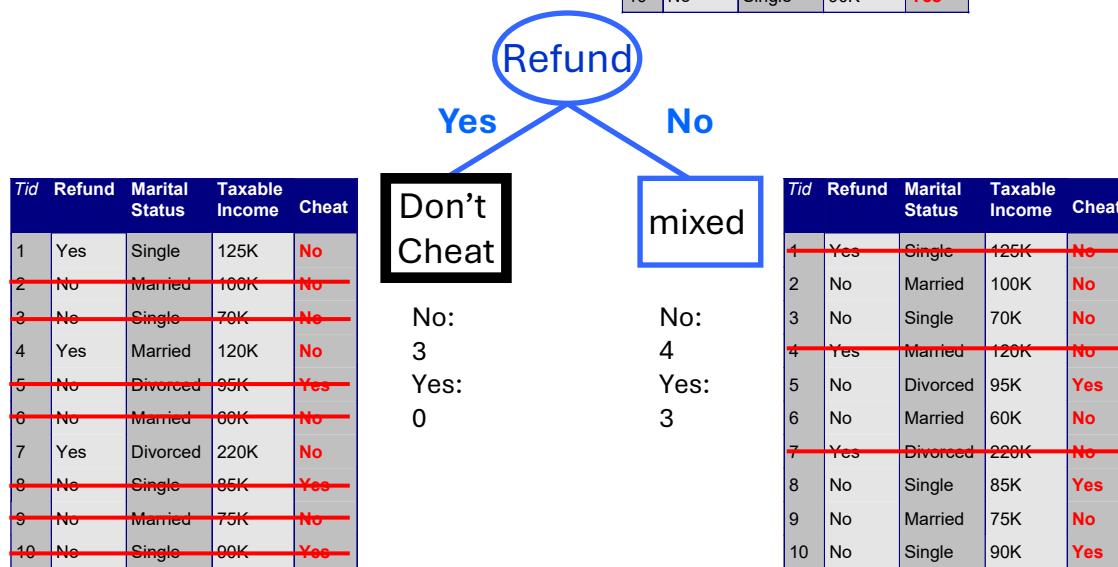
Many Algorithms:

- Hunt's Algorithm (one of the earliest)
- CART (Classification And Regression Tree)
- ID3, C4.5, C5.0 (by Ross Quinlan, introduced information gain)
- CHAID (CHi-squared Automatic Interaction Detection)
- MARS (Improvement for numerical features)
- SLIQ, SPRINT
- Conditional Inference Trees (recursive partitioning using statistical tests)

All algorithms use a simple, greedy top-down splitting strategy!

# The Effect of a Split

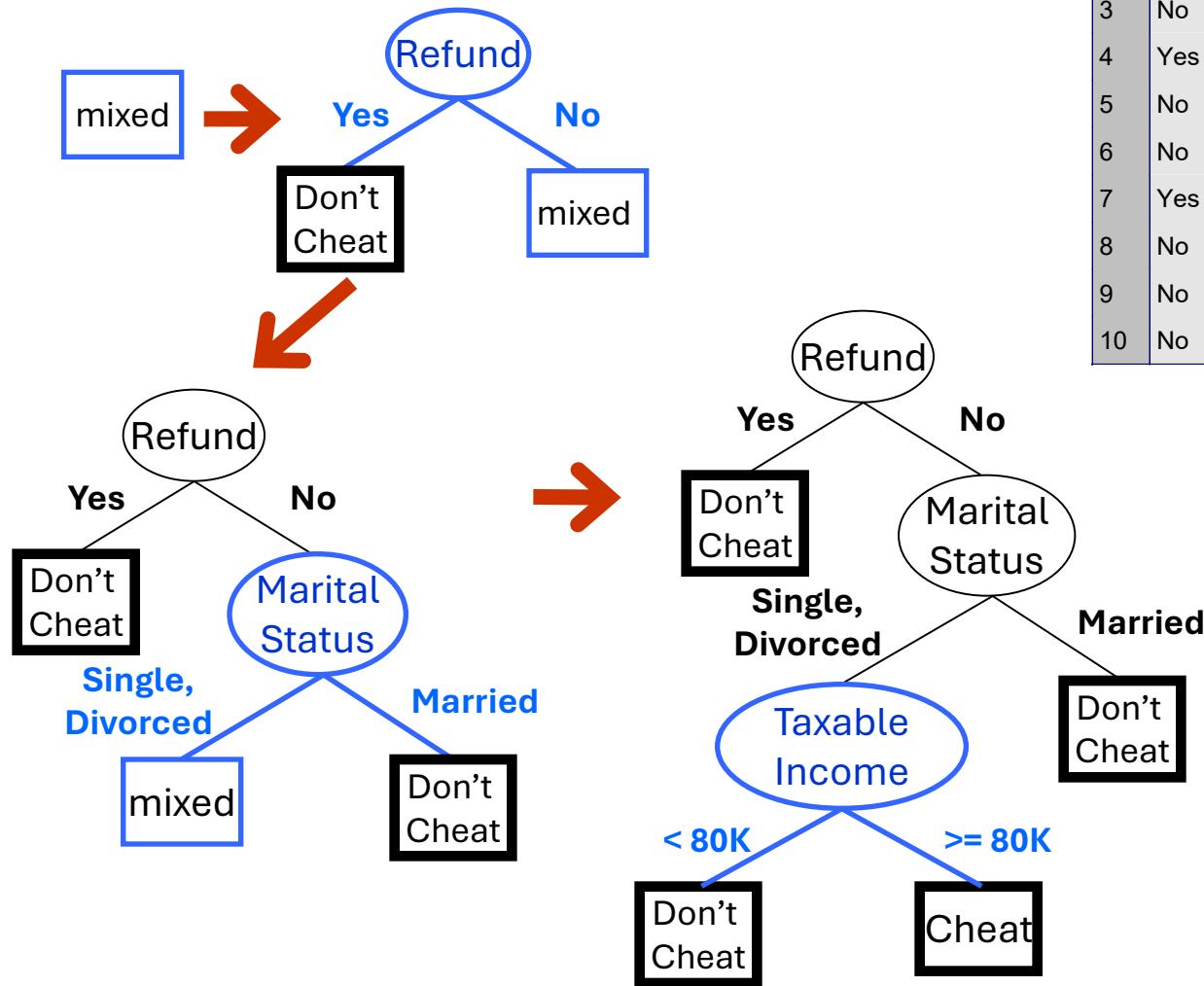
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Every split partitions the data set into two subsets.

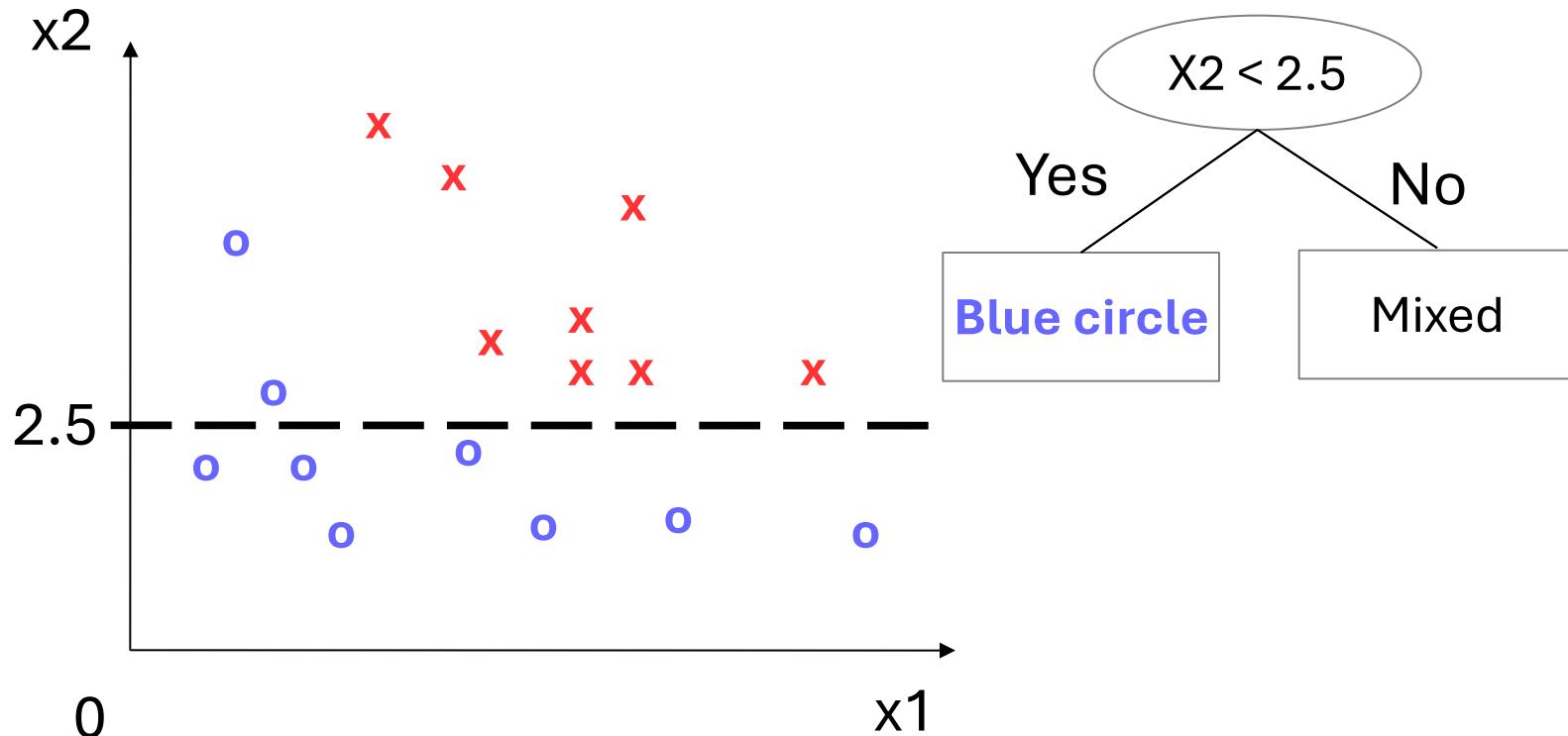
# Hunt's Algorithm

"Use attributes to split the data recursively, till each split contains only a single class."



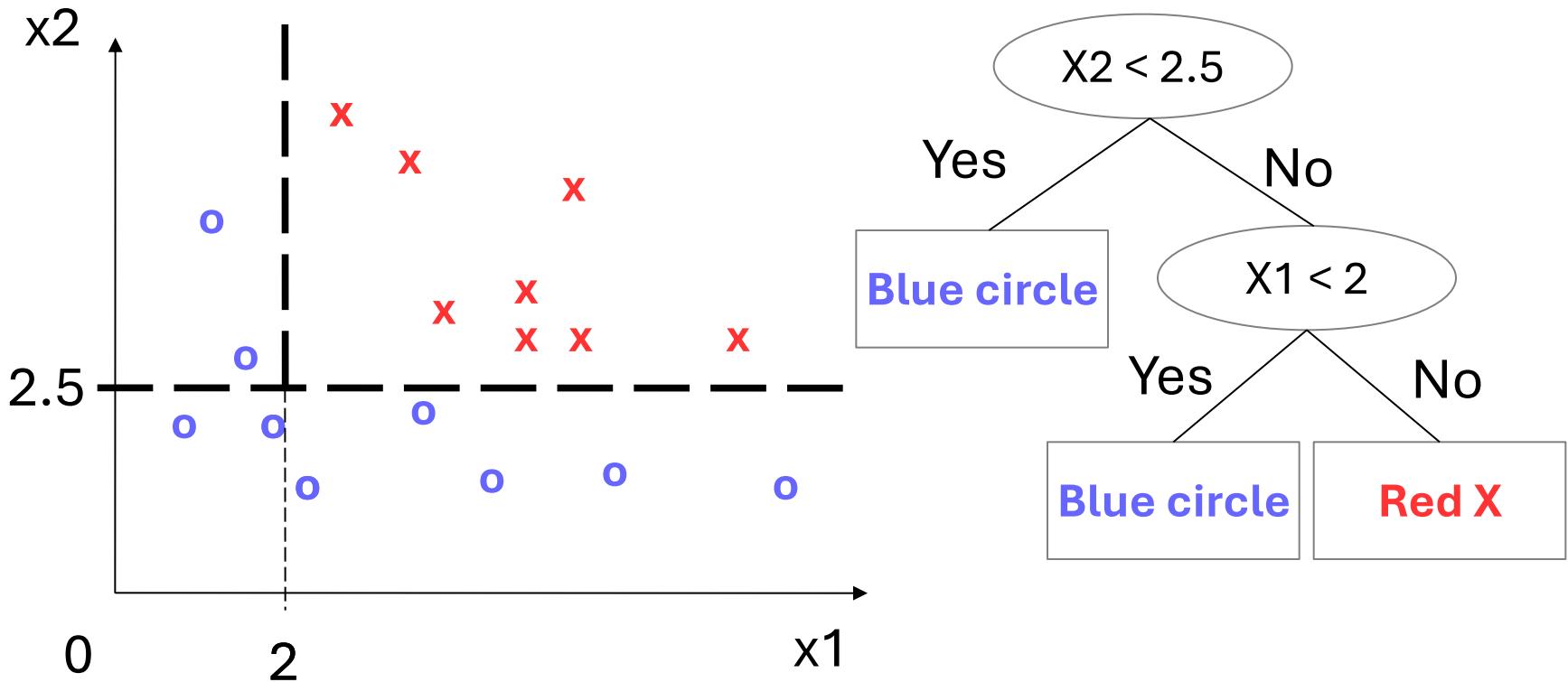
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Example: Creating a Decision Tree



Decision trees can only cut parallel to an axis!

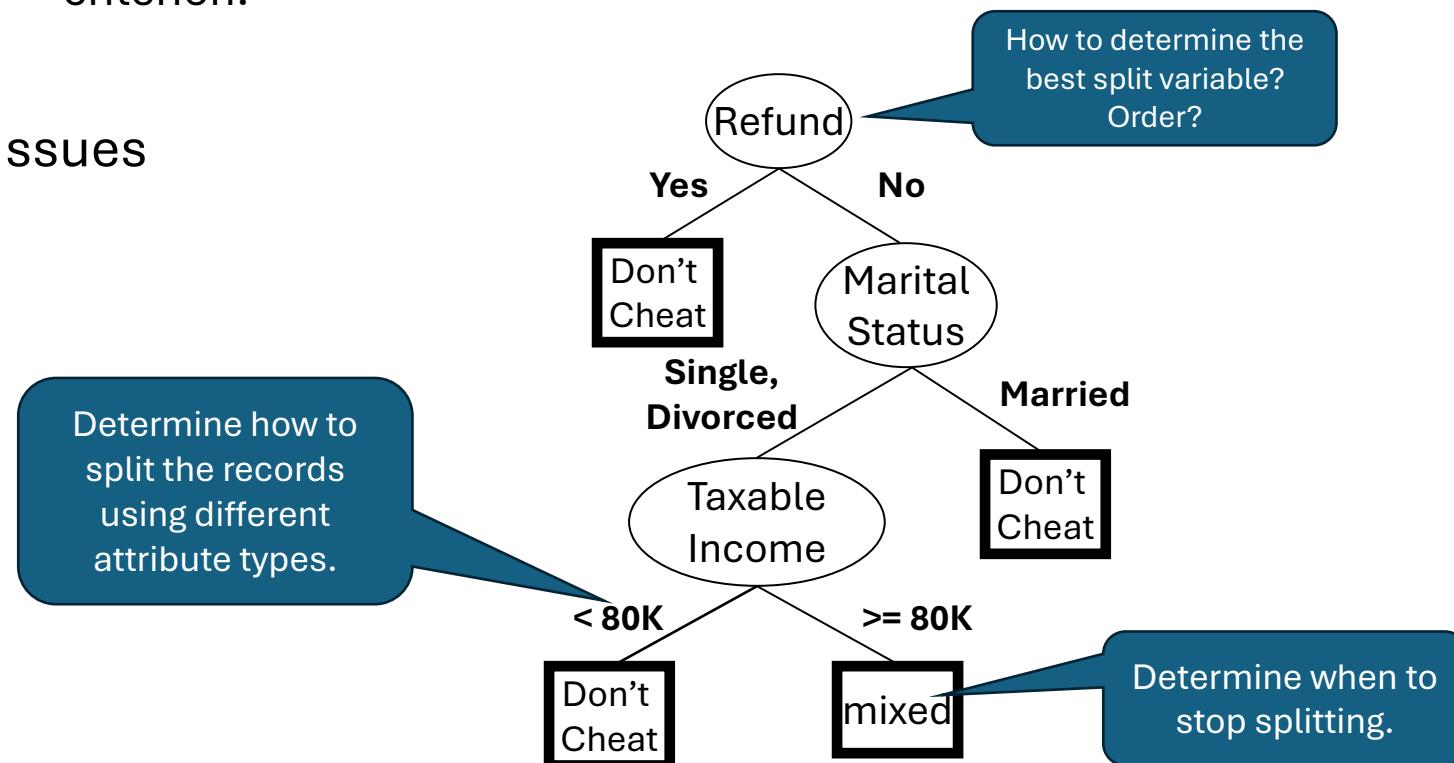
# Example: Creating a Decision Tree



# Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.

- Issues



# Tree Induction

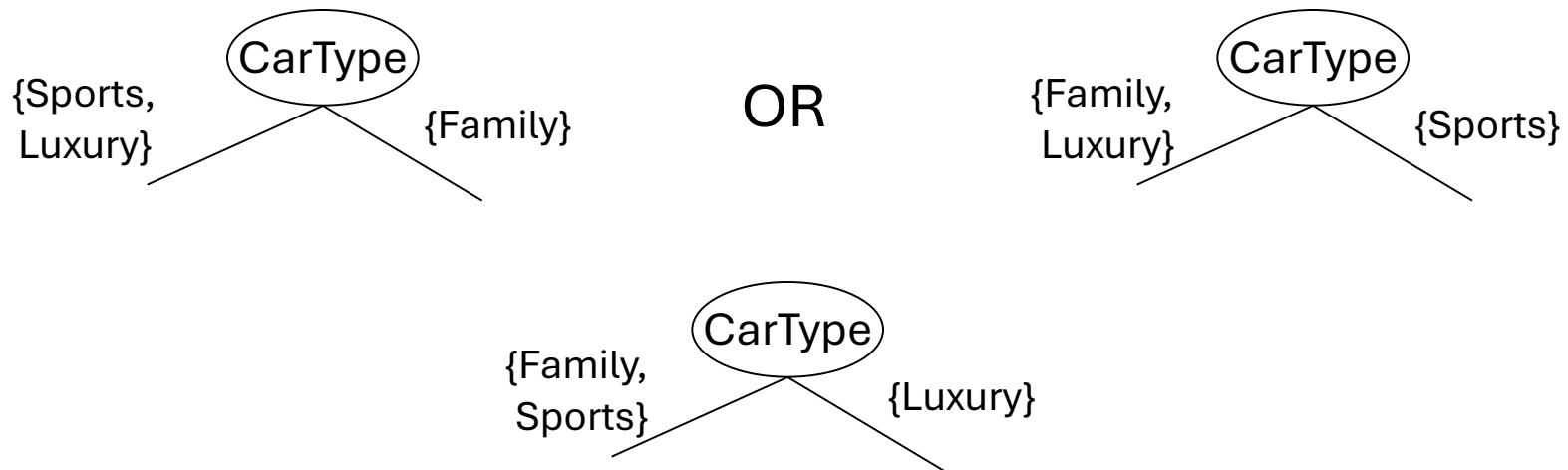
- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.
- Issues
  - **Determine how to split the records using different attribute types.**
  - How to determine the best split variable?
  - Determine when to stop splitting.

# How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous (interval/ratio)

# Splitting Based on Nominal Attributes

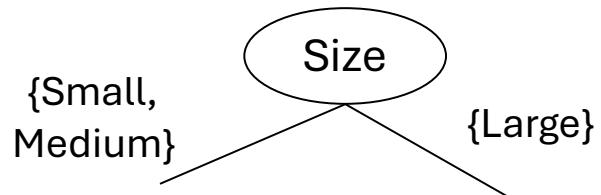
- Divide the unordered values into two subsets.
- We need to find optimal partitioning.



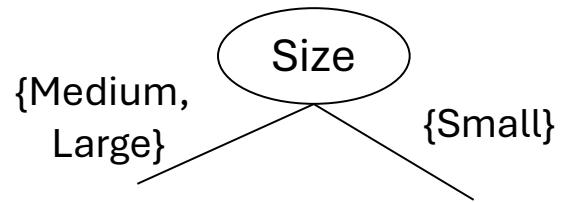
Best decision depends on what we want to predict!

# Splitting Based on Ordinal Attributes

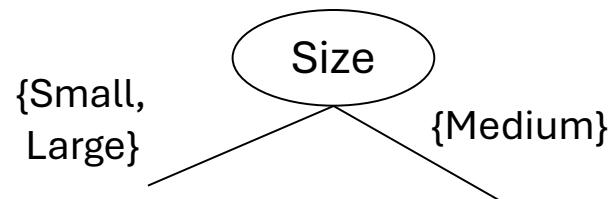
- Divide the ordered values into two subsets.



OR

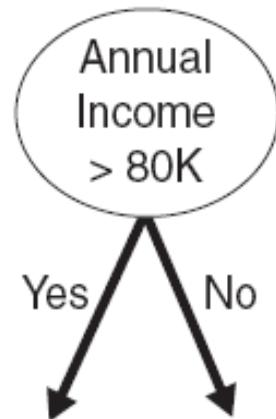


- What about this split?

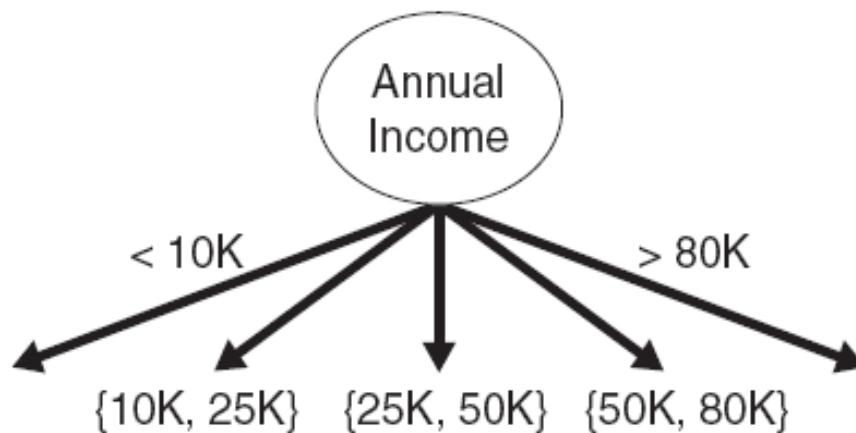


# Splitting Based on Continuous Attributes

Binary split



Multi-way split



Discretization to form an ordinal categorical attribute:

- **Static** – discretize the data set once at the beginning (equal interval, equal frequency, etc.).
- **Dynamic** – discretize during the tree construction.
  - Example: For a binary decision ( $A < \nu$ ) or ( $A \geq \nu$ ) consider all possible splits and finds the best cut. This can be done efficiently.

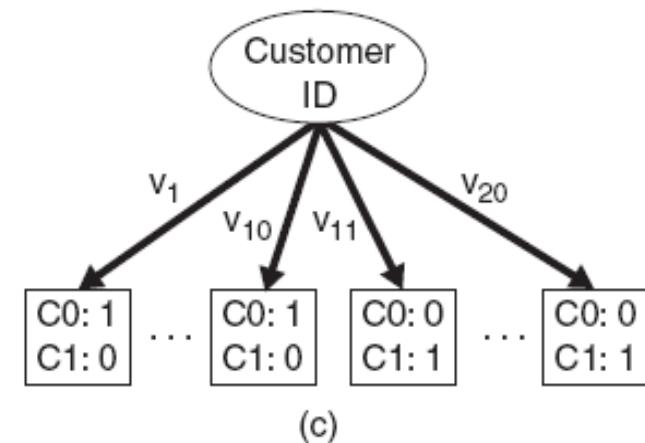
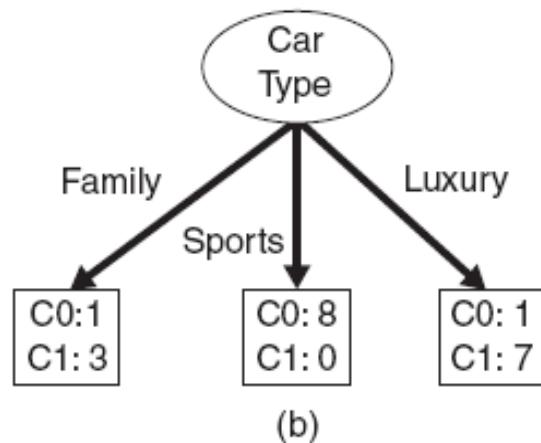
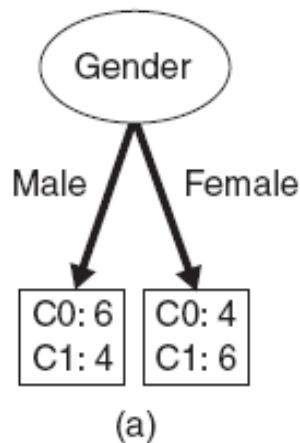
# Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.
- Issues
  - Determine how to split the records using different attribute types.
  - **How to determine the best split variable?**
  - Determine when to stop splitting

# How to determine the Best Split

**Before Splitting: 10 records of class 0,  
10 records of class 1**

C0: 10  
C1: 10



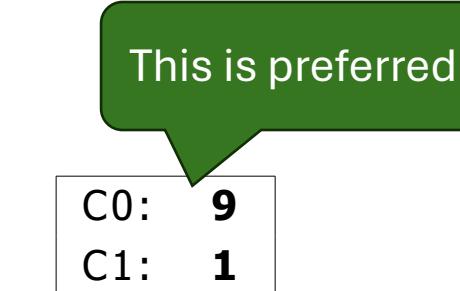
**Which splitting variable is the best?**

# Determine the Quality of a Node: Node Impurity

- Nodes represent a subset of data that satisfy the splitting condition.
- We want to create nodes with homogeneous class distributions.
- Need a measure of node impurity:

C0:	<b>5</b>
C1:	<b>5</b>

**Non-homogeneous,  
High degree of impurity**

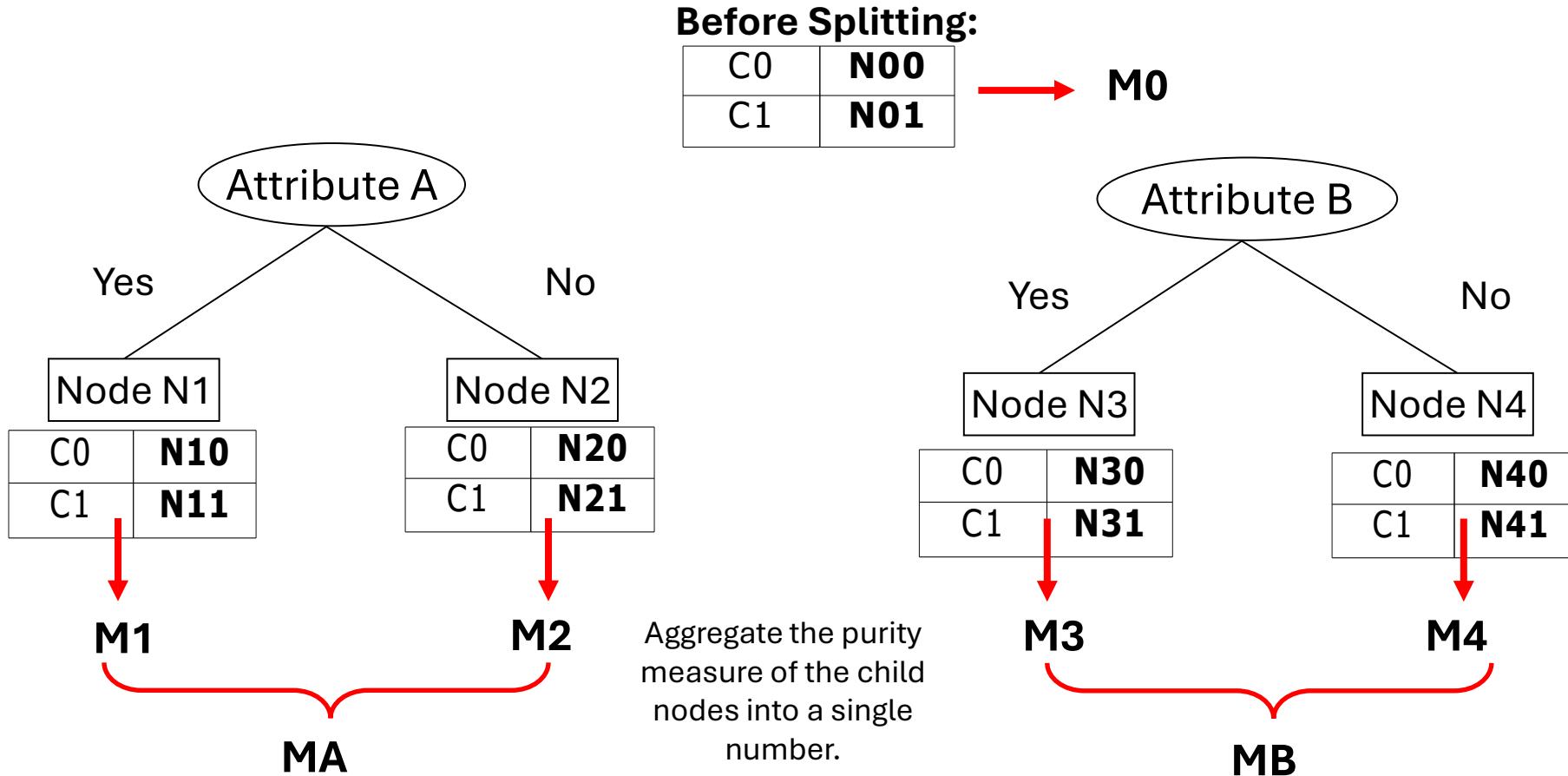


**Homogeneous,  
Low degree of impurity**

- General rule for measures of impurity:
  - Smaller is better.
  - 0 represents the complete purity.

# Find the Best Split: General Framework

Assume we have a measure **M** that tells us how "pure" a node is.



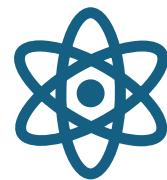
We look at the improvement called the gain:

**Gain =  $M_0 - MA$  vs.  $M_0 - MB$**  → Choose best split

# Measures of Node Impurity



**Gini Index**



Entropy



Classification  
error

# Measure of Impurity: Gini Index of a Node

- Gini Index for a given node t :

$$GINI(t) = \sum_j p(j | t)(1 - p(j | t)) = 1 - \sum_j p(j | t)^2$$

$p(j | t)$  is estimated as the relative frequency of class j at node t

- Origin: The Gini index is a measure of statistical dispersion intended to represent the income inequality within nations. Here it is used as a statistical measure that quantifies how mixed or impure the class distribution in a node is.
- Maximum Impurity:  $1 - 1/n_c$  (number of classes) when records are equally distributed among all classes. For a binary decision it is 0.5.
- Minimum Impurity: 0 when all records belong to one class.
- Examples

C1	<b>0</b>
C2	<b>6</b>
<b>Gini=0.000</b>	

C1	<b>1</b>
C2	<b>5</b>
<b>Gini=0.278</b>	

C1	<b>2</b>
C2	<b>4</b>
<b>Gini=0.444</b>	

C1	<b>3</b>
C2	<b>3</b>
<b>Gini=0.500</b>	

# Examples: Gini Index of a Node

$$GINI(t) = 1 - \sum_j p(j | t)^2$$

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = \mathbf{0}$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = \mathbf{0.278}$$

C1	<b>2</b>
C2	<b>4</b>

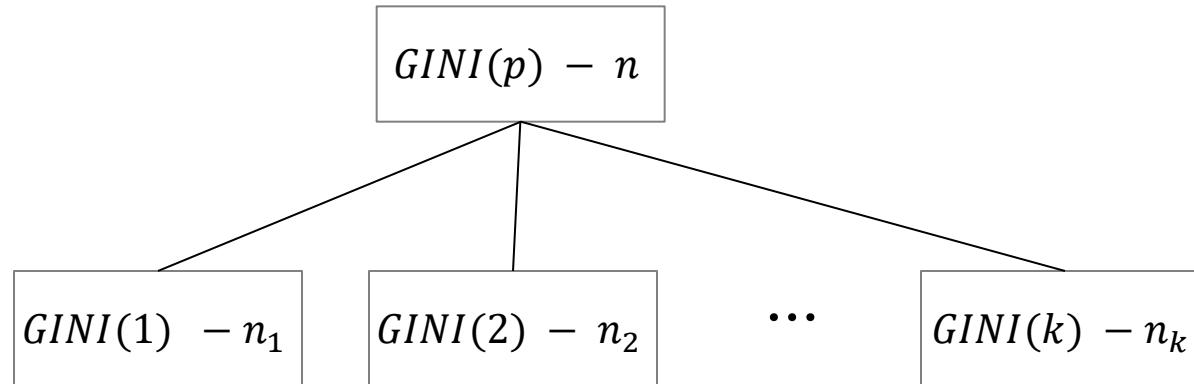
$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = \mathbf{0.444}$$

Maximal impurity here is  $\frac{1}{2} = .5$

# Splitting Based on the Gini Index

When a node  $p$  is split into  $k$  partitions (children), the quality of the split is computed as a weighted:



$$GINI_{split} = \sum_i^k \frac{n_i}{n} GINI(i)$$

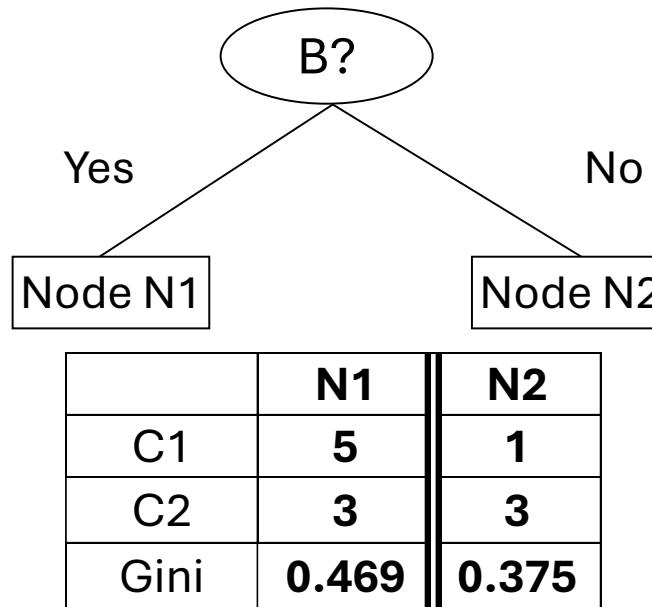
where  $n_i$  is the number of records at child  $i$ , and  $n$  is the number of records at node  $p$ .

Used in the algorithms CART, SLIQ, SPRINT.

# Example: Splitting based on the Gini Index

- Effect of weighing partitions: Larger **and** purer partitions are preferred.

	Parent
C1	6
C2	6
<b>Gini = 0.5</b>	



$$\text{Gini}(N1) = 1 - (5/8)^2 - (3/8)^2 = 0.469$$

$$\text{Gini}(N2) = 1 - (1/4)^2 - (3/4)^2 = 0.375$$

$$\begin{aligned}\text{Gini of the split} &= 8/12 * 0.469 + \\ &4/12 * 0.375 \\ &= 0.438\end{aligned}$$

$$\begin{aligned}\text{Gain} &= 0.5 - 0.438 \\ &= 0.062\end{aligned}$$

**GINI improves!**

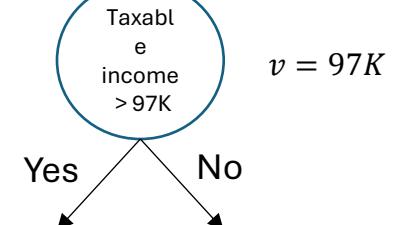
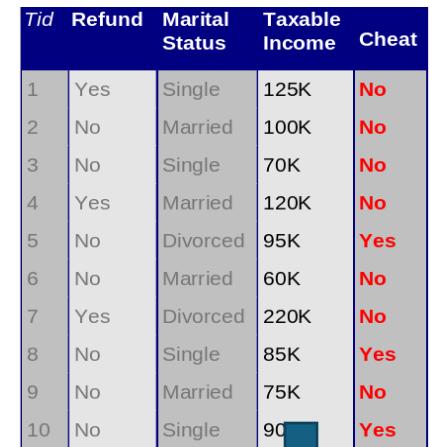
# Continuous Attributes: Computing Gini Index

- How does the algorithm choose the splitting value  $v$ ? (= dynamic discretization)
  - Number of possible splitting values = Number of distinct values
- Efficient Method: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing Gini index
  - Choose the split position that has the smallest Gini index

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Sorted Values →

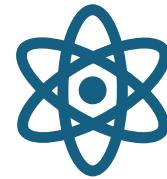
Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No	
Taxable Income											
	60	70	75	85	90	95	100	120	125	220	
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420



# Measures of Node Impurity



Gini Index



**Entropy**



Classification  
error

# Measure of Impurity: Entropy

- Entropy at a given node t:

$$\text{Entropy}(t) = - \sum_j p(j | t) \log(p(j | t))$$

$p(j | t)$  is the relative frequency of class j at node t;  
 $0 \log(0) \stackrel{\text{def}}{=} 0$  is used!

- Origin: In information theory, entropy quantifies the amount of uncertainty involved in the value of a random. Here the random variable is the class label of a randomly chosen observation in a node.
- Maximum Impurity:  $\log(n_c)$  when records are equally distributed among all classes.
- Minimum Impurity: 0 when all records belong to one class. We can perfectly predict the class label of each observation in the node.

# Examples: Entropy

$$\text{Entropy}(t) = - \sum_j p(j | t) \log(p(j | t))$$

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Entropy} = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Entropy} = -(1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65$$

C1	<b>3</b>
C2	<b>3</b>

$$P(C1) = 3/6 \quad P(C2) = 3/6$$

$$\text{Entropy} = -(3/6) \log_2 (3/6) - (3/6) \log_2 (3/6) = 1$$

# Splitting based on Information Gain

$$GAIN_{split} = Entropy(p) - \left( \sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent Node,  $p$  is split into  $k$  partitions;  
 $n_i$  is number of records in partition  $i$

- Measures reduction in Entropy achieved because of the split.  
Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3, C4.5 and C5.0
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

# Splitting based on the Gain Ratio

$$GainRatio_{split} = \frac{GAIN_{split}}{SplitInfo}$$

$$SplitInfo = - \sum_{i=1}^k \frac{n_i}{n} \log \left( \frac{n_i}{n} \right)$$

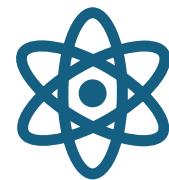
Parent Node,  $p$  is split into  $k$  partitions;  
 $n_i$  is number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitInfo). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain.

# Measures of Node Impurity



Gini Index



Entropy



**Classification  
error**

# Splitting Criteria based on Classification Error

- Classification error at a node  $t$ :

$$Error(t) = 1 - \max_i p(i | t)$$

$p(j | t)$  is the relative frequency of class  $j$  at node  $t$

- Measures the classification error made in a node by a simple classifier that always predict the majority class (given by the  $\max(\cdot)$  in the equation).
- Maximum Impurity:  $1 - \frac{1}{n_c}$  when records are equally distributed among all classes (maximal error).
- Minimum Impurity: 0 when all records belong to one class = maximal purity (no error)
- Splitting decision: Use weighted averages or gain as for the other indices to make the splitting decision.

# Examples: Classification Error

$$Error(t) = 1 - \max_i p(i | t)$$

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Error} = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Error} = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

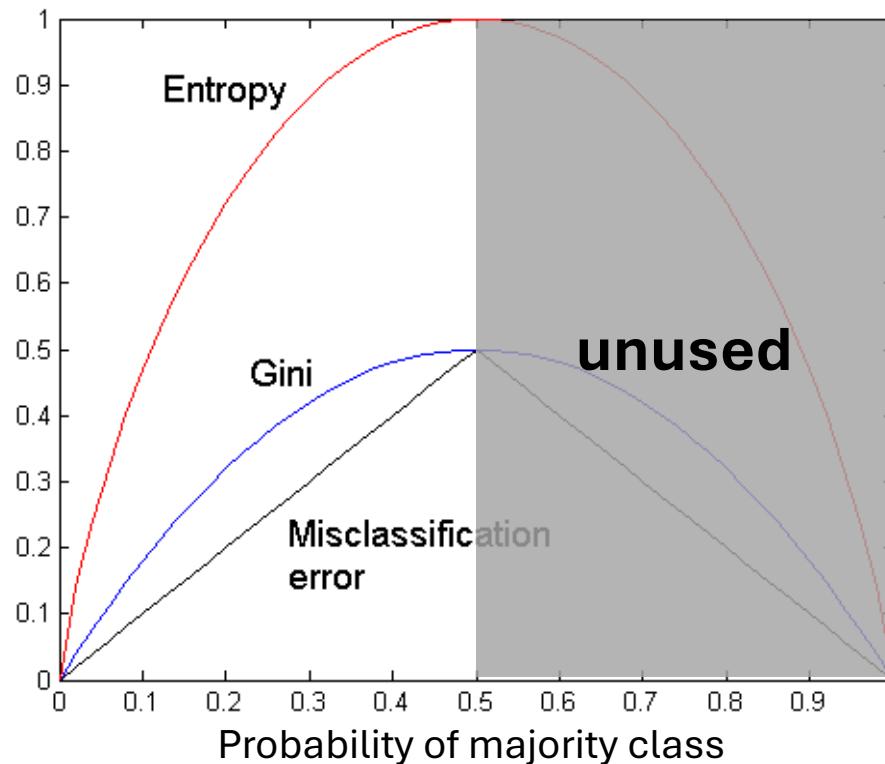
C1	<b>3</b>
C2	<b>3</b>

$$P(C1) = 3/6 \quad P(C2) = 3/6$$

$$\text{Error} = 1 - \max(3/6, 3/6) = 1 - 3/6 = .5$$

# Comparison among Splitting Criteria

For a 2-class problem: Probability of the majority class  $p$  is always  $> .5$



**Note:** The order is the same no matter what splitting criterion is used, however, the gain (differences) are not since they depend on the slope.

# Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.
- Issues
  - Determine how to split the record using different attribute types.
  - How to determine the best split?
  - **Determine when to stop splitting**

# Stopping Criteria for Tree Induction

- Stop expanding a node when **all the records belong to the same class** (used Hunt's algorithm).
- Stop expanding a node when all the records in the node have the **same attribute values**. Splitting becomes impossible.
- **Early termination criterion.** Stop when more splits will lead to overfitting the training data. We will discuss this later with tree pruning.

Standard  
method

# Advantages of Decision Trees



INEXPENSIVE TO  
CONSTRUCT



EXTREMELY FAST AT  
CLASSIFYING  
UNKNOWN RECORDS



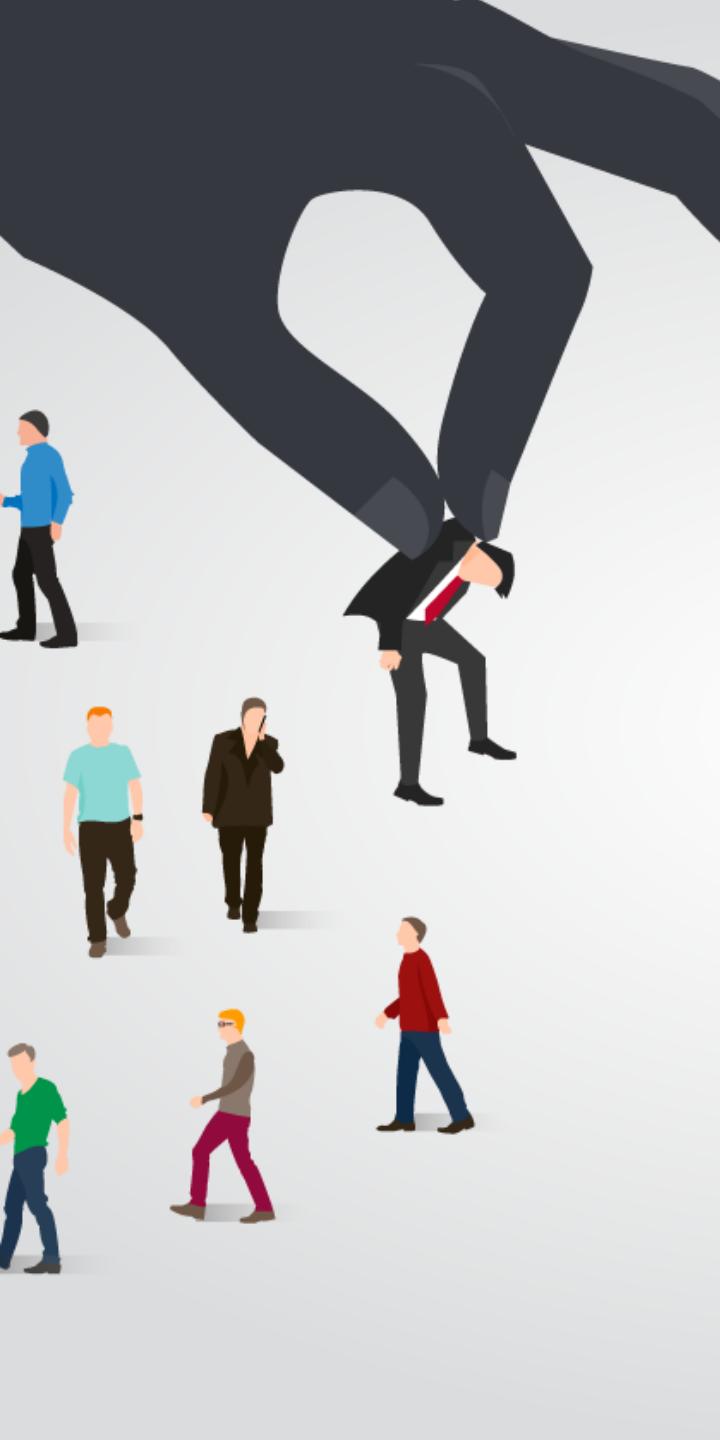
EASY TO INTERPRET  
FOR SMALL-SIZED  
TREES



ACCURACY IS  
COMPARABLE TO  
OTHER  
CLASSIFICATION  
TECHNIQUES FOR  
MANY SIMPLE DATA  
SETS

# Example: C4.5

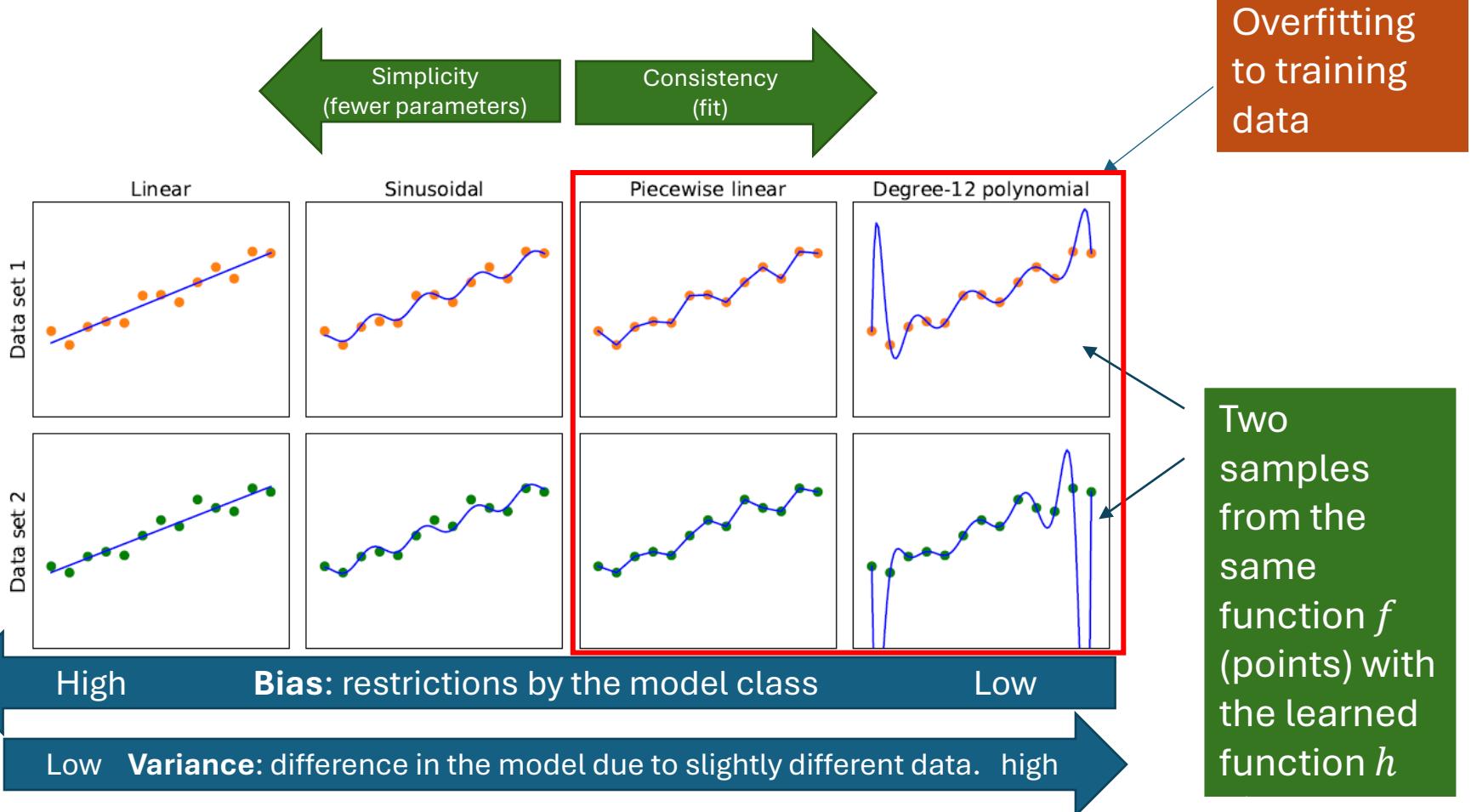
- Simple depth-first construction.
- Uses Information Gain (improvement of the entropy measure).
- Handling both continuous and discrete attributes (continuous attributes are split at threshold).
- Needs entire data to fit in memory (unsuitable for large datasets).
- Final trees are pruned to remove branches that hurt performance.
  
- Code available at
  - <http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz>
  - Open-Source implementation as J48 in Weka/rWeka



# Topics

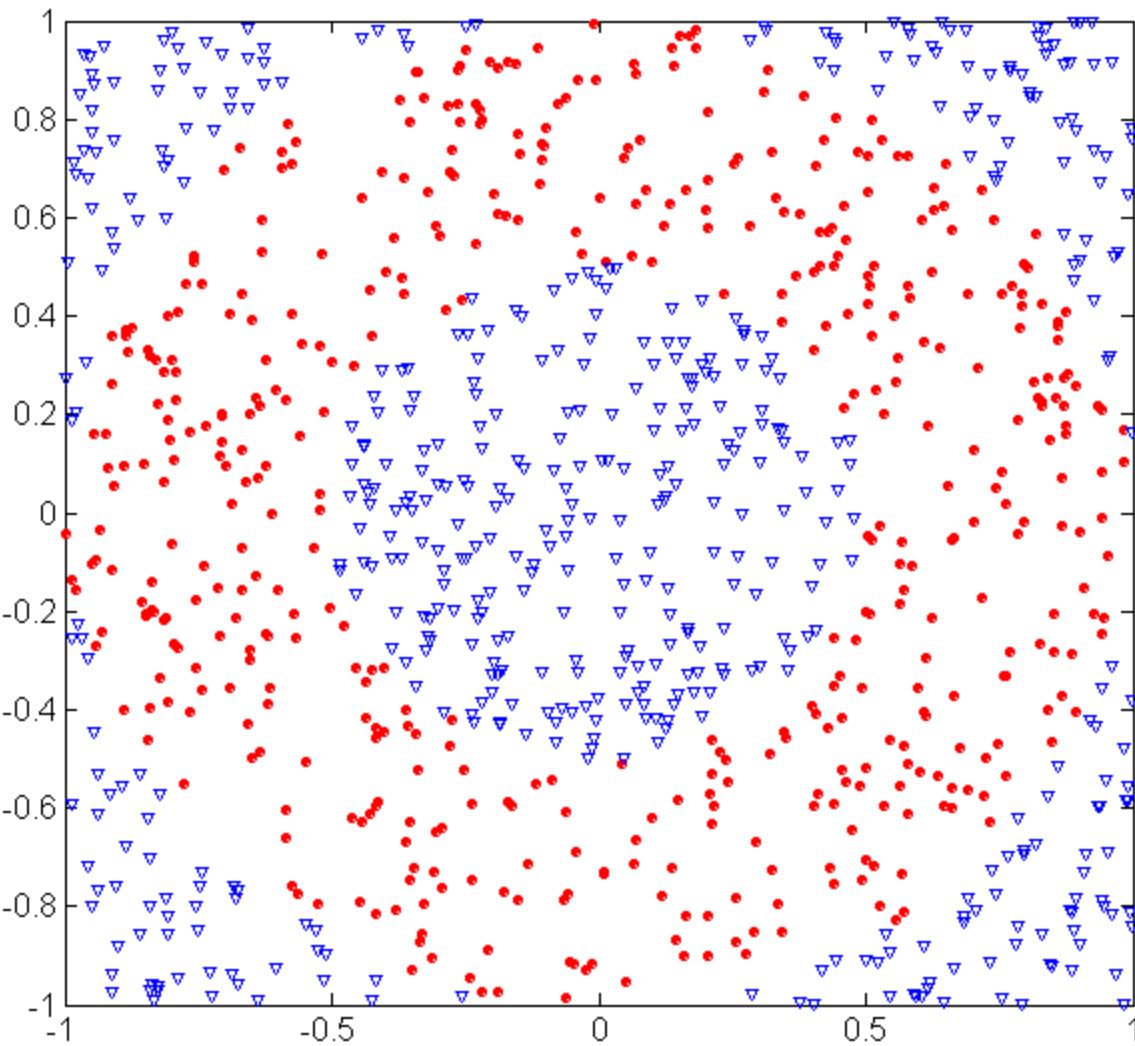
- Introduction
- Decision Trees
  - Overview
  - Tree Induction
- **Overfitting and other Practical Issues**
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Feature Selection

# Model Selection: Bias vs. Variance



Note: This trade-off applies to any model.

# Example: Underfitting and Overfitting



How is the data generated?

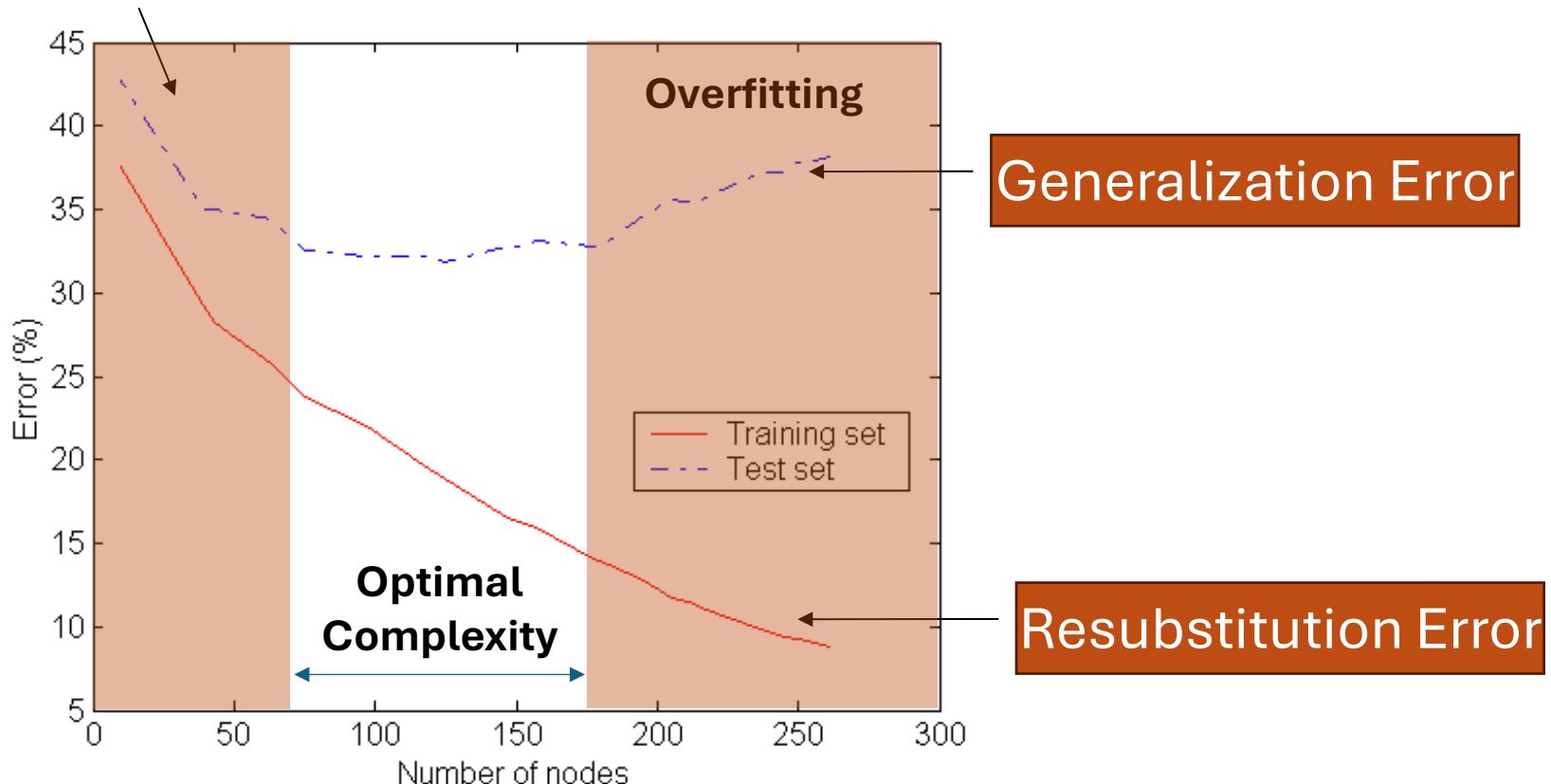
500 circular and 500 triangular data points.

Circular points:  
 $0.5 \geq \sqrt{x_1^2 + x_2^2} \leq 1$

Triangular points:  
 $\sqrt{x_1^2 + x_2^2} < 0.5$  or  
 $\sqrt{x_1^2 + x_2^2} > 1$

# Example: Underfitting and Overfitting

**Underfitting**

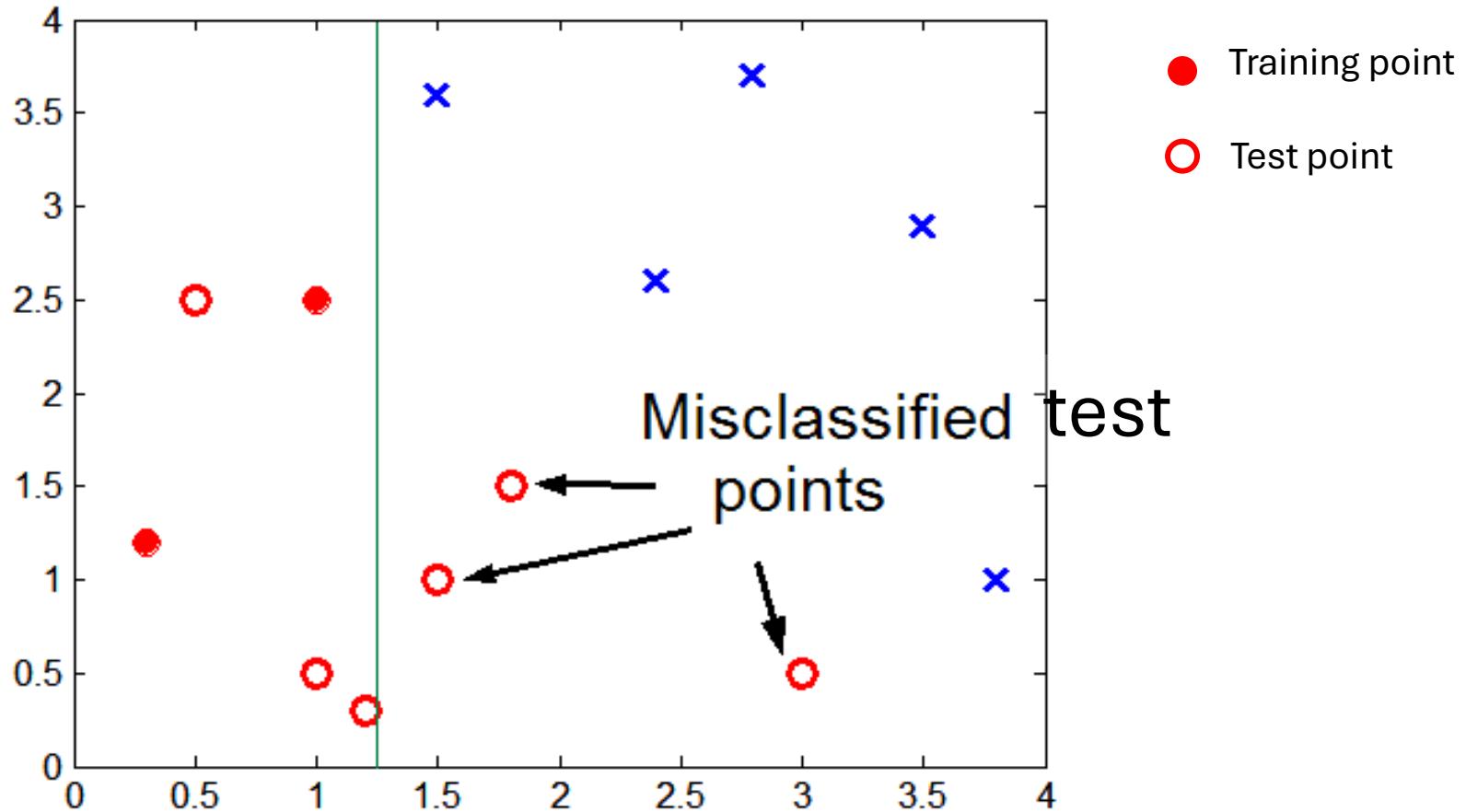


**Underfitting:** The model is too simple, both training and test errors are large.

**Overfitting:** The model is too complicated and starts memorizing the training data.

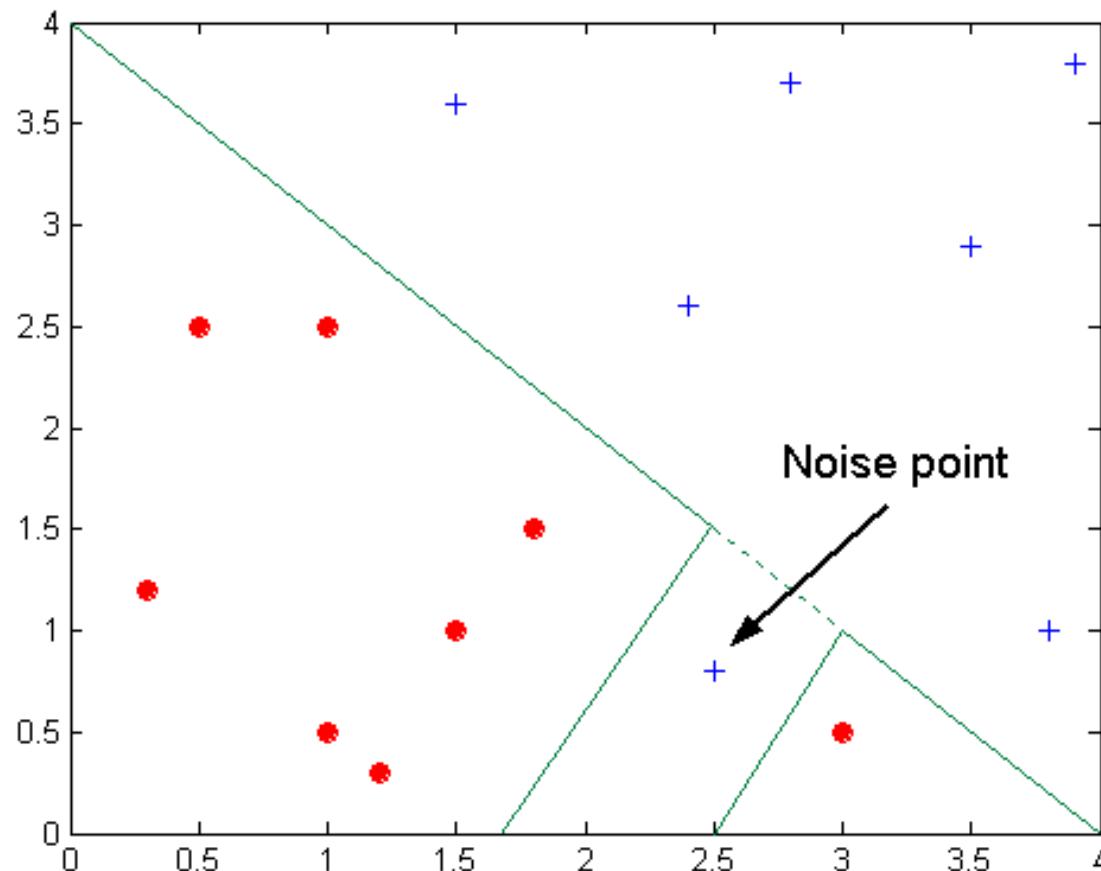
Generalization error goes up again.

# Example: Underfitting due to Insufficient Examples



Lack of training data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

# Example: Overfitting due to Noise



**Decision boundary is distorted to accommodate a noise point**

# Training Error vs. Generalization Error

- Training error is reduced by **overfitting** and results in decision trees that **are more complex than necessary**.
- Training error does not provide a good estimate of how well the tree will perform on new example (e.g., test data).
- We need to estimate the **Generalization Error** expected for new data.

# Estimating the Generalization Error

- **Resubstitution error  $e$ :** error on training set
- **Generalization error  $e'$ :** error on testing set

Methods for estimating generalization errors:

1. **Optimistic approach:** assume  $e' = e$
2. **Pessimistic approach:**
  - Estimate as  $e' = e + N \times 0.5$  ( $N$ : number of leaf nodes)
  - For a tree with 30 leaf nodes and 10 errors on training out of 1000 training instances:  
Training error  $e = 10/1000 = 1\%$   
Estimated generalization error  $e' = (10 + 30 \times 0.5)/1000 = 2.5\%$
3. **Validation approach:**
  - uses a validation (test) data set (or cross-validation) to estimate the generalization error.

Penalty for  
model complexity!  
0.5 per leave node is often  
used for binary splits.



**"Simpler is better"**

## Occam's Razor

### The Principle of Parsimony

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.
- Reason: Complex models have a greater chance of overfitting. I.e., it fitted accidentally errors in the training data.

**Therefore, one should consider also model complexity when evaluating a model.**

# How to Address Overfitting in Decision Trees

- **Full tree (will overfit)**
  - Stop if all instances belong to the **same class**.
  - Stop if all the **attribute values are the same**.
- **Reduce overfitting with pre-pruning / early stopping**
  - Stop if **number of instances** is less than some user-specified threshold (estimates become bad for small sets of instances).
  - Stop if class distribution of instances are **independent** of the available features (e.g., using a  $\chi^2$  test).
  - Stop if expanding the current node **does not improve impurity** measures more than a user-specified threshold (e.g., Gini or information gain).

# How to Address Overfitting in Decision Trees

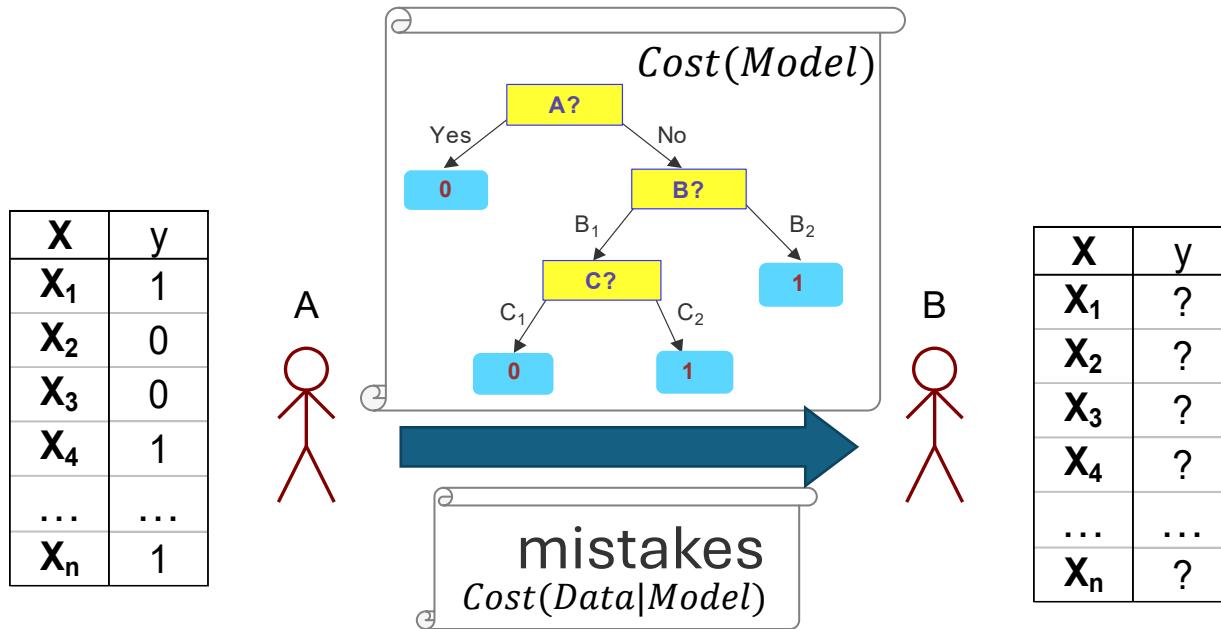
## Reduce overfitting with post-pruning

1. Grow complete decision tree.
2. Try to prune sub-trees of the decision tree in a bottom-up fashion.

Options:

- **Generalization error:** If generalization error improves after pruning a sub-tree, replace the sub-tree by a leaf node with the majority class of the training instances as the predicted label.
- **Penalty for complexity:** You can use Maximum Description Length (MDL).

# Refresher: Minimum Description Length (MDL)



- $Cost(Model)$  encodes each node (splitting condition and children).
- $Cost(Data|Model)$  encodes information to correct misclassification errors.
- $Cost(Model, Data) = Cost(Data|Model) + Cost(Model) \rightarrow \min$ 
  - Cost is the number of bits needed for encoding.

Penalty for model complexity!  
This is equivalent to the pessimistic generalization error.

# Example: Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

**Before split:**

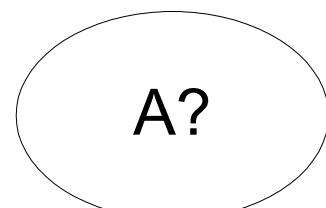
Training Error = 10/30

Pessimistic error =  $(10 + 1 \times 0.5)/30 = 10.5/30$

**After split:**

Training Error = 9/30

Pessimistic error =  $(9 + 4 \times 0.5)/30 = 11/30$



Training error decreases but pessimistic error estimate increases! **PRUNE!**

Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Error = 9/30

## Other issues:

# Data Fragmentation and Search Strategy

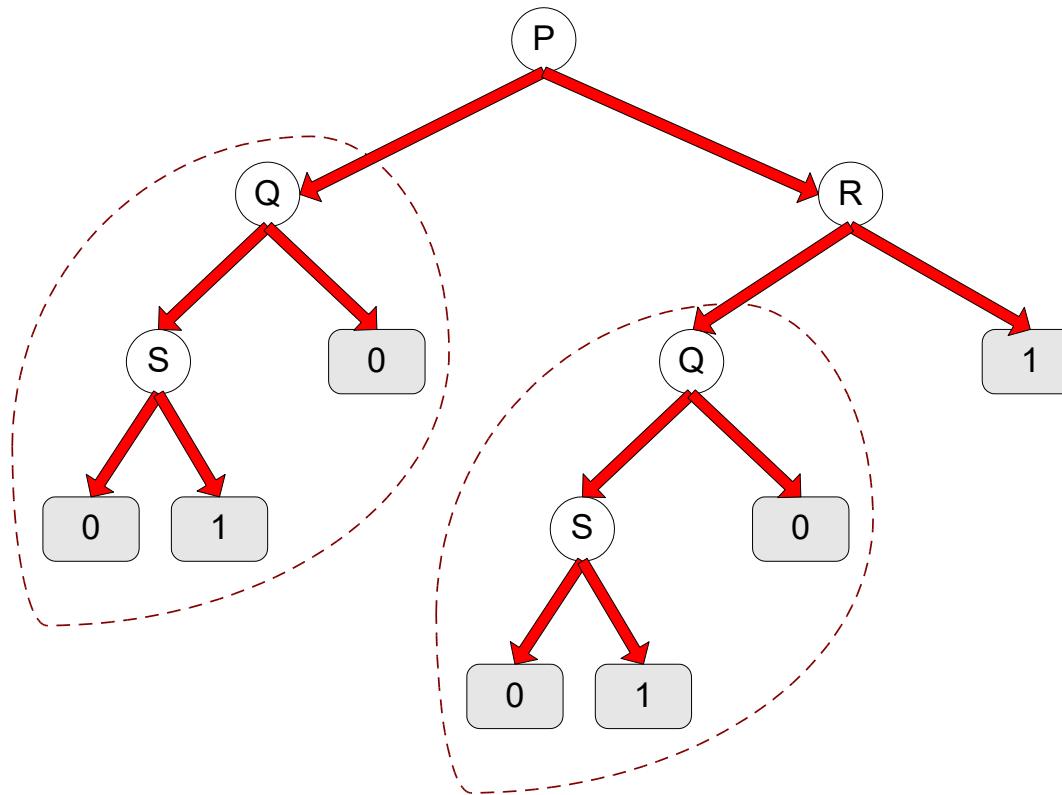
## Data Fragmentation

- Number of instances gets smaller as you traverse down the tree and can become too small to make a statistically significant decision (splitting or determining the class in a leaf node)  
→ Many algorithms **stop when a node has not enough instances.**

## Search Strategy

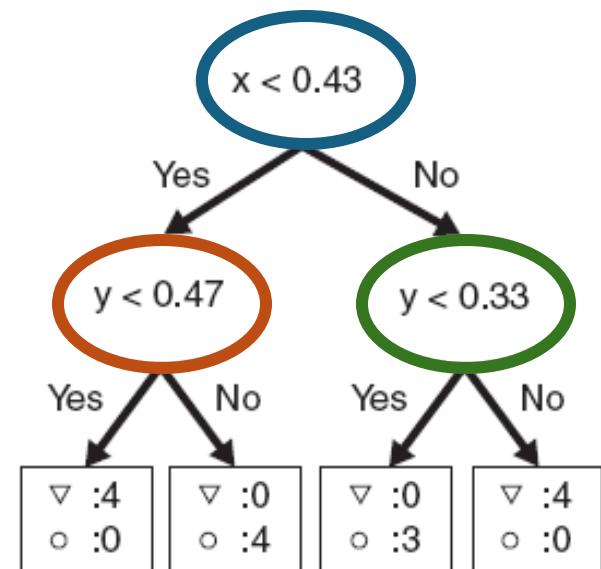
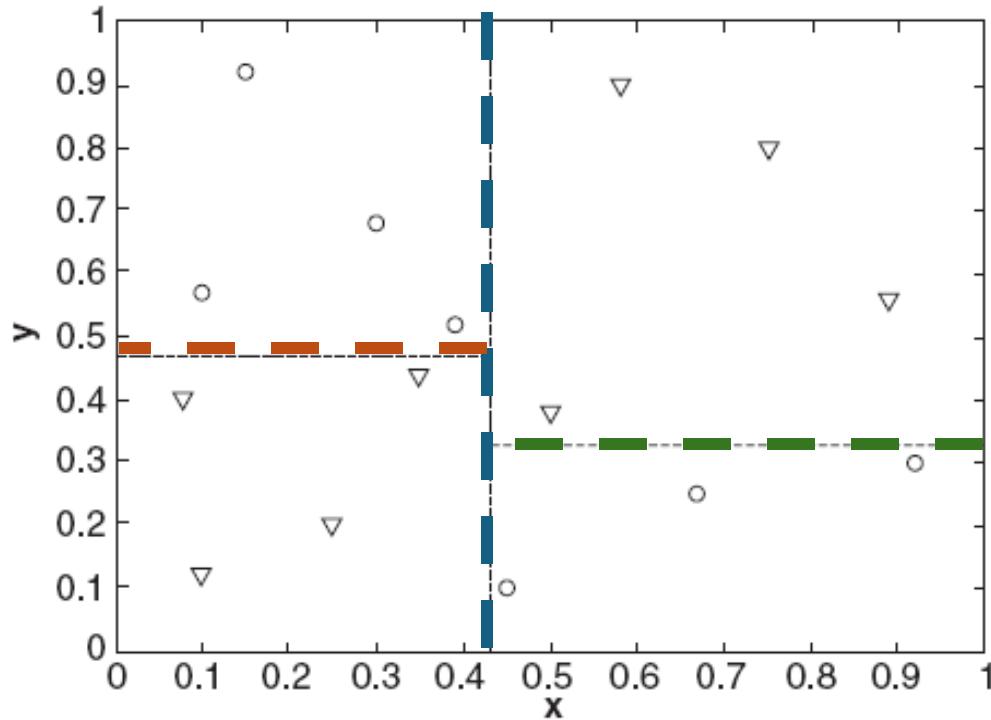
- Finding an optimal decision tree is NP-hard  
→ Most algorithm use a **greedy, top-down, recursive partitioning strategy** to induce a reasonable solution.

# Other issues: Tree Replication



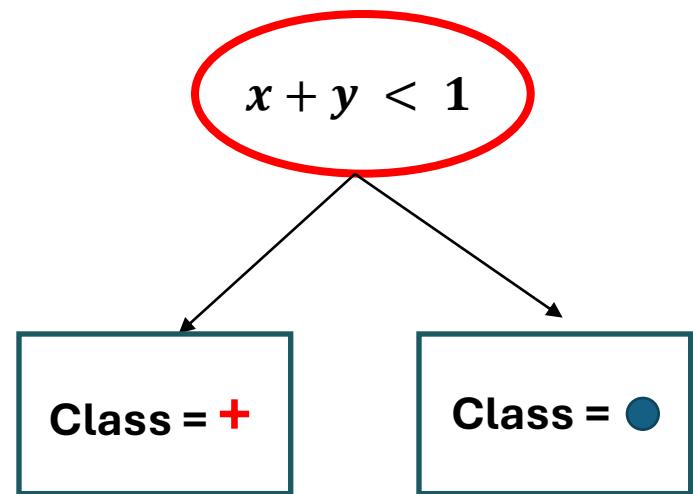
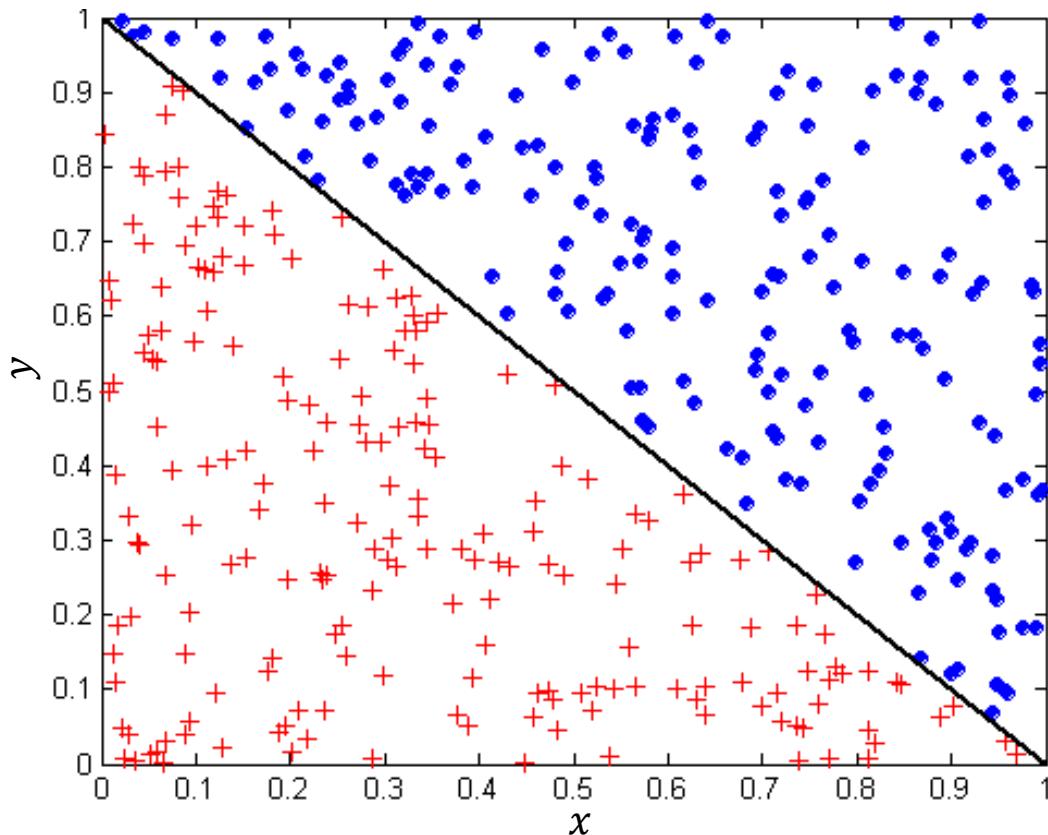
- Same subtree appears in multiple branches.
- Makes the model more complicated and harder to interpret.

# Decision Boundary of a Classifier



- The border line between two neighboring regions of different classes is known as the decision boundary.
- The decision boundary of decision trees is parallel to the axes because each test condition represents a threshold on a single attribute.
- Not expressive enough for modeling continuous variables directly. Discretization is performed for the splits.

# Oblique Decision Trees



- The test condition may involve multiple attributes.
- More expressive representation.
- Finding the optimal test condition is computationally expensive!

**Not used in practice for decision trees** but Linear Discriminant Analysis (LDA) can learn a single oblique decision boundary.

## EVALUATION

Relevance	✗ ○ ○ ○
Efficiency	○ ○ ✗ ○ ○
Effectiveness	○ ✗ ○ ○ ○
Sustainability	○ ○ ○ ○ ✗
Impact	○ ○ ○ ✗ ○

# Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
- Overfitting and other Practical Issues
- **Model Selection and Evaluation**
  - **Metrics for Performance Evaluation**
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- Feature Selection

# Metrics for Performance Evaluation: Confusion Matrix

- Focuses on the predictive capability of a model (not speed, scalability, etc.)
- For simplicity, we will present a binary classification problem here, but most measures generalize to multi-class problems.

## Confusion Matrix

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

# Metrics for Performance Evaluation: Statistical Test

From Statistics: Null Hypotheses  $H_0$  is that the actual class is Yes.

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes		<b>Type I error (FN)</b>
	Class>No	<b>Type II error (FP)</b>	

$\leftarrow H_0$

Type I error:  $P(\text{NO} \mid H_0 \text{ is true})$  → Significance level  $\alpha$

Type II error:  $P(\text{Yes} \mid H_0 \text{ is false})$  → Power  $1 - \beta$

# Metrics for Performance Evaluation: Accuracy

Most widely-used metric:

- How many do we predict correct (in percent)?

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{N}$$

# Limitation of Accuracy



Consider a 2-class problem with a total population of

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

A model that predicts everything to be class 0, has an accuracy of  
 $9990/10000 = 99.9\%$

Accuracy is misleading because the model does not detect any class 1 example!

→ This is a very common problem called the  
**class imbalance problem**

# Cost Matrix

Different types of error can have different cost!

		PREDICTED CLASS		
		C(i j)	<b>Class=Yes</b>	<b>Class&gt;No</b>
<b>ACTUAL CLASS</b>	<b>Class=Yes</b>	C(Yes Yes)	C(No Yes)	
	<b>Class&gt;No</b>	C(Yes No)	C(No No)	

$C(i | j)$ : Cost of misclassifying class  $j$  example as class  $i$

# Computing the Cost of Classification

Cost Matrix		PREDICTED CLASS	
ACTUAL CLASS	C(i j)	+	-
	+	-1	100
	-	1	0

Missing a '+' case is really expensive!

Model M <sub>1</sub>	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%

$$\text{Cost} = -1*150 + 100*40 + 1*60 + 0*250 = \mathbf{3910}$$

Model M <sub>2</sub>	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%

$$\text{Cost} = \mathbf{4255}$$

# Cost-Biased Measures (from Information Retrieval)

$$Precision (p) = \frac{a}{a + c}$$

$$Recall (r) = \frac{a}{a + b}$$

$$F - measure (F) = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

		PREDICTED CLASS	
ACTUAL CLASS		Class Yes	Class No
	Class Yes	a (TP)	b (FN)
	Class No	c (FP)	d (TN)

- Precision only considers cost for examples predicted as Yes.
- Recall only considers cost for examples that are truly Yes.
- F-measure combines precision and recall and ignores d.

# Kappa Statistic

**Idea:** Compare the accuracy of the classifier with a **random classifier**. The classifier should be better than random!

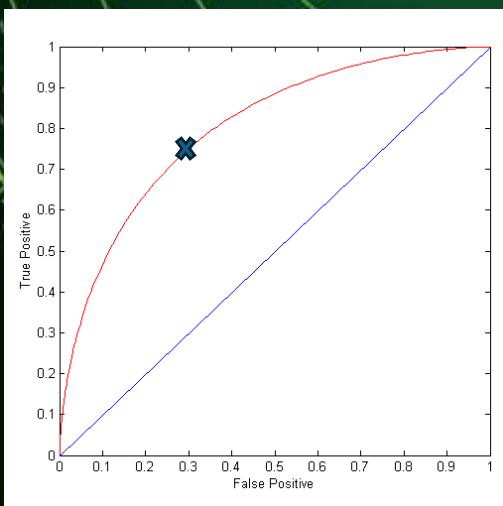
	PREDICTED CLASS	
ACTUAL CLASS	Class Yes	Class No
	Class Yes	a (TP)
	Class No	c (FP)
		b (FN)
		d (TN)

$$\kappa = \frac{\text{total accuracy} - \text{random accuracy}}{1 - \text{random accuracy}}$$

$$\begin{aligned}\text{total accuracy} &= \frac{TP + TN}{N} \\ \text{random accuracy} &= \frac{TP + FP \times TN + FN + FN + TN \times FP + TP}{N^2}\end{aligned}$$

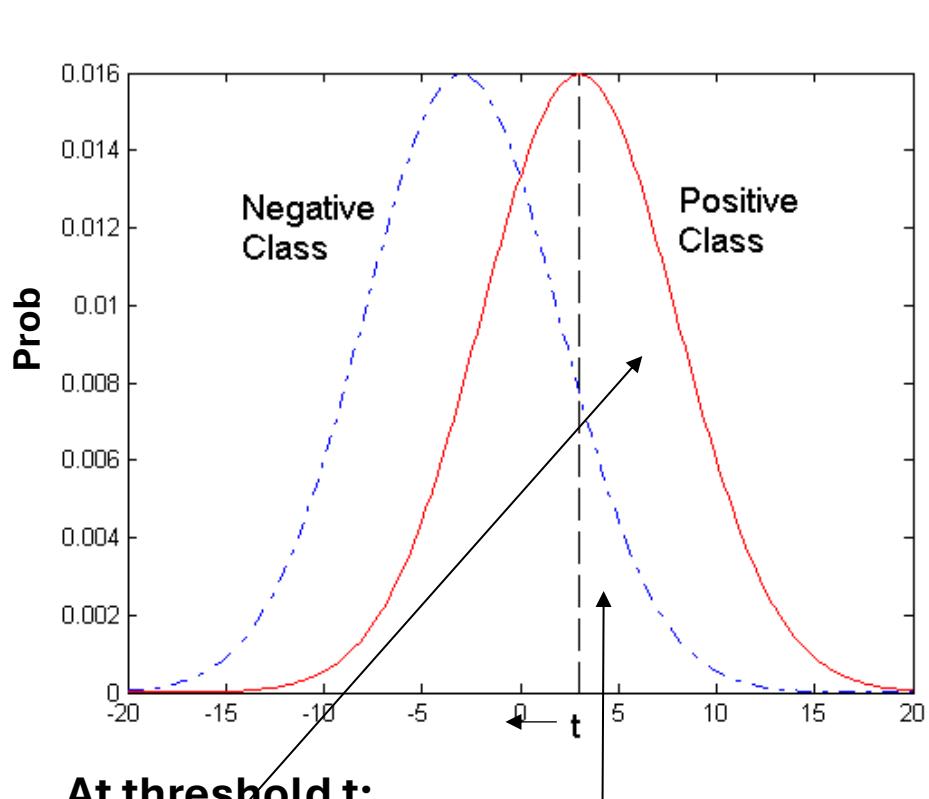
# Receiver Operating Characteristic (ROC)

- Developed in 1950s for signal detection theory to analyze noisy signals to characterize the trade-off between positive hits and false alarms.
- Works only for binary classification (two-class problems).
- ROC curve plots TPR (true positive rate) on the y-axis against FPR (false positive rate) on the x-axis.
- Performance of each classifier represented as a point. Changing the threshold of the algorithm, sample distribution or cost matrix changes the location of the point and forms a curve.



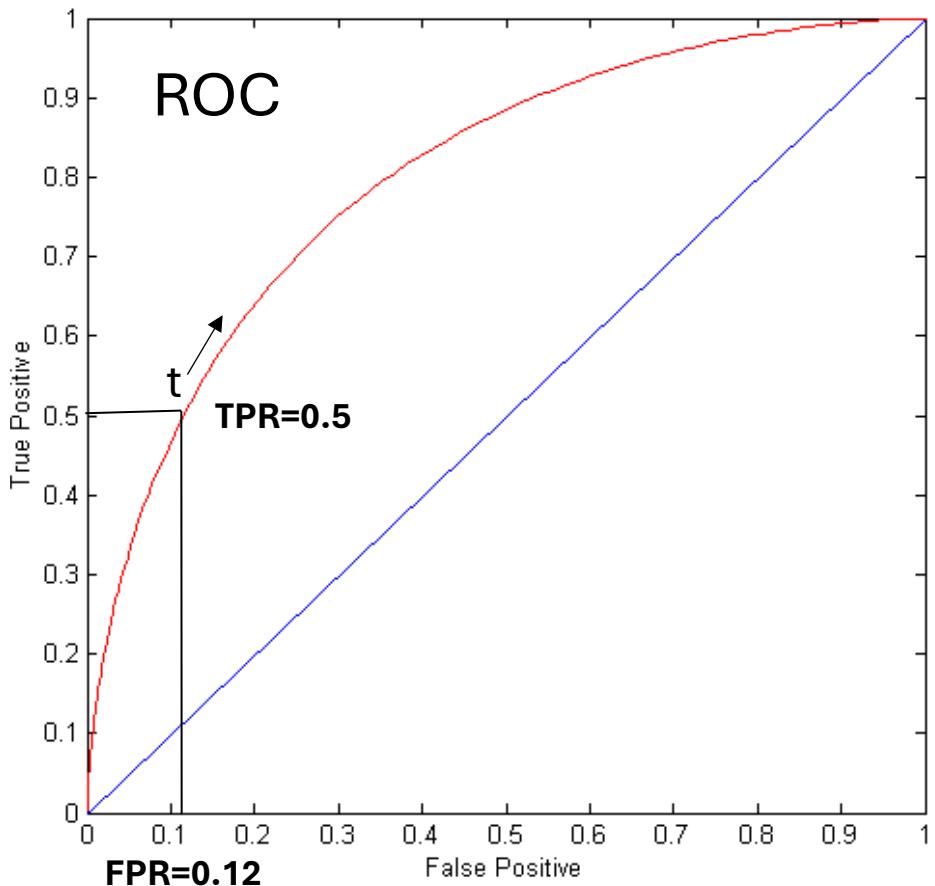
# ROC Curve

- Example with 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at  $x > t$  is classified as positive



At threshold  $t$ :

**TPR=0.5, FNR=0.5, FPR=0.12, FNR=0.88**



- Move  $t$  to get the other points on the ROC curve.

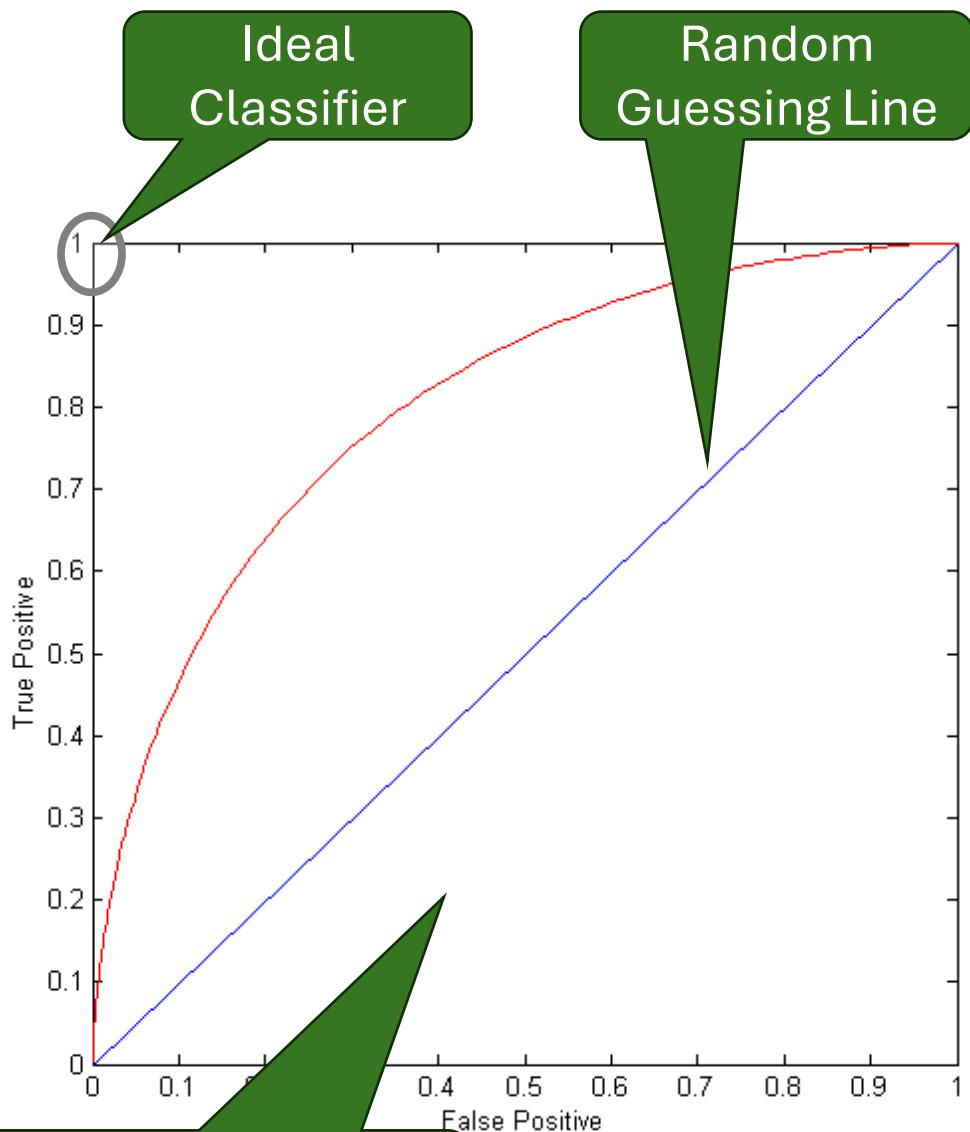
# ROC Curve

(TPR,FPR):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

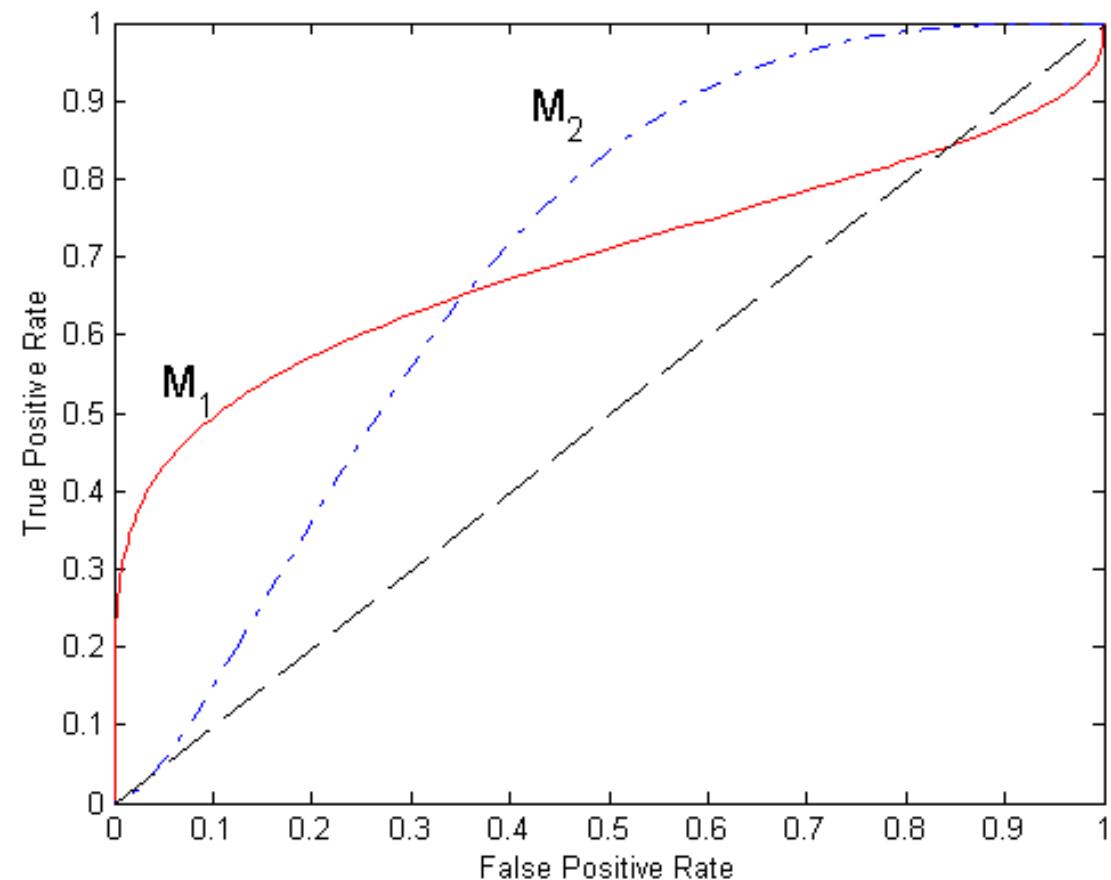
Diagonal line:

- Random guessing
- Below diagonal line:  
prediction is opposite of the true class



Below the diagonal:  
predict the opposite class

# Using ROC for Model Comparison



No model consistently outperform the other

- $M_1$  is better for small FPR
- $M_2$  is better for large FPR

## Area Under the ROC curve (AUC)

- Ideal:
  - $AUC = 1$
- Random guess:
  - $AUC = 0.5$

## EVALUATION

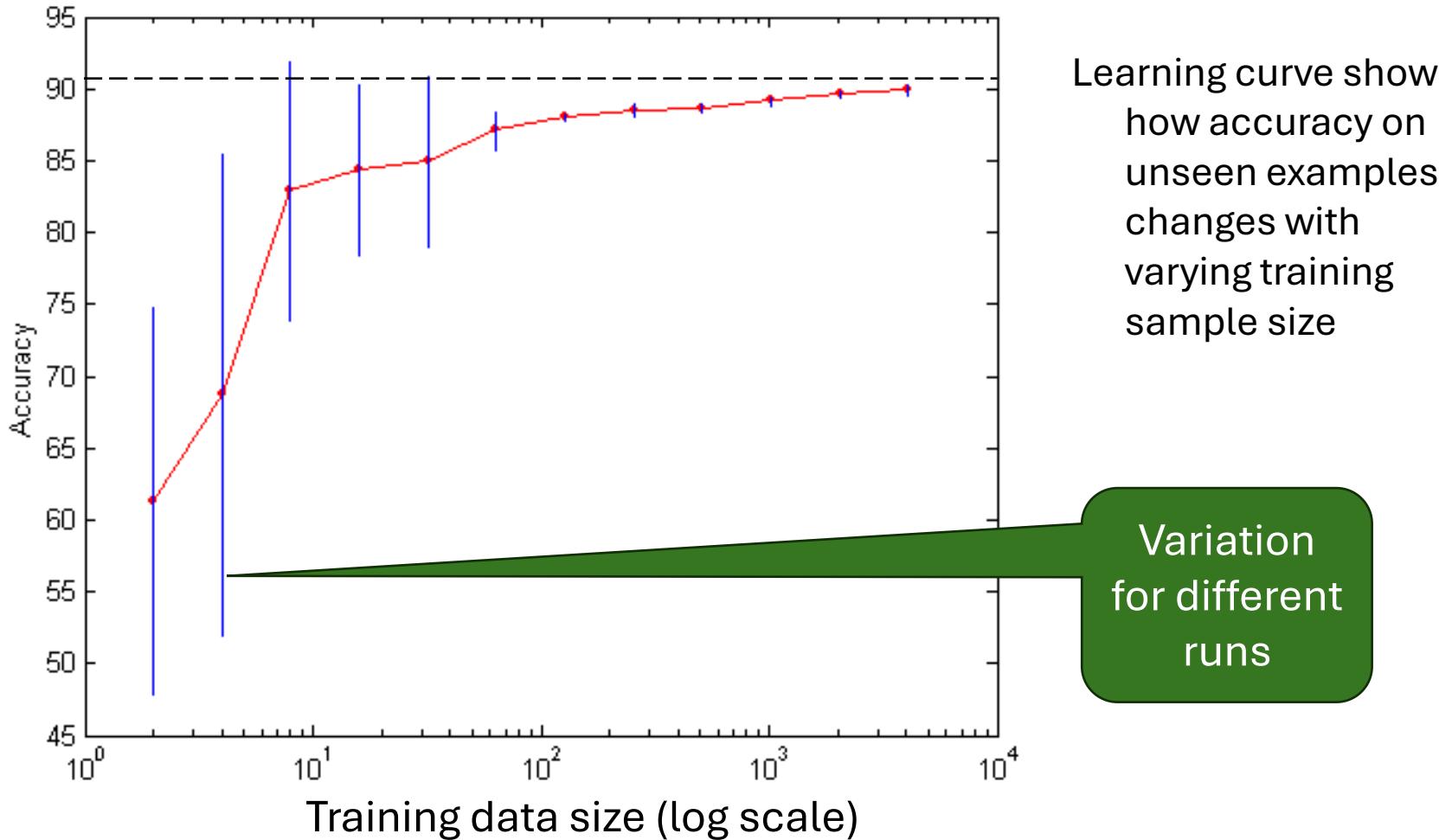
Relevance	✗ ○ ○ ○
Efficiency	○ ○ ✗ ○ ○
Effectiveness	○ ✗ ○ ○ ○
Sustainability	○ ○ ○ ○ ✗
Impact	○ ○ ○ ✗ ○

# Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
- Overfitting and other Practical Issues
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - **Methods to Obtain Reliable Estimates**
  - Model Comparison (Relative Performance)
- Feature Selection

# Learning Curve

Accuracy and variance between runs depend on the size of the training data.



# Estimating the Generalization Error Using Test Data

- To estimate generalization error we need to separate the data into a set to train and a set to test.
- **Holdout testing/Random splits:** Split the data randomly into, e.g., 80% training and 20% testing.

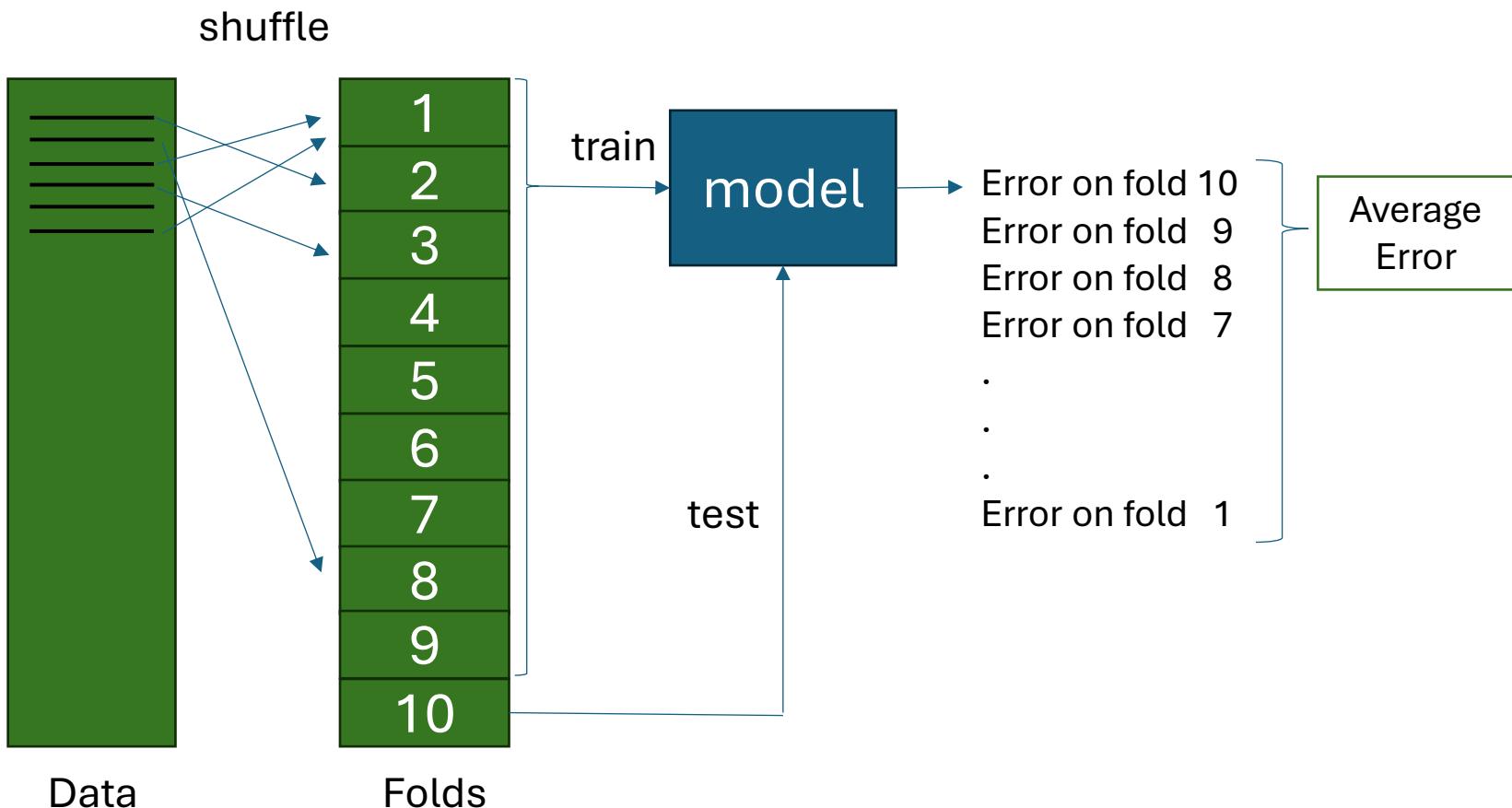
**Very important:** the algorithm can never look at the test set during learning!



# $k$ -fold Cross Validation

**$k$ -fold cross validation:** Use data better to estimate the generalization error:

- Split the data randomly into  $k$  folds.
- For  $k$  rounds hold 1 fold back for testing and use the remaining  $k - 1$  folds for training.
- Use the average of the error/accuracy as a better estimate.
- Some algorithms/tools do that internally.



# Training and Testing with Hyperparameters

**Hyperparameters:** Many algorithms allow choices for learning. E.g.,

- maximal decision tree depth
- selected features

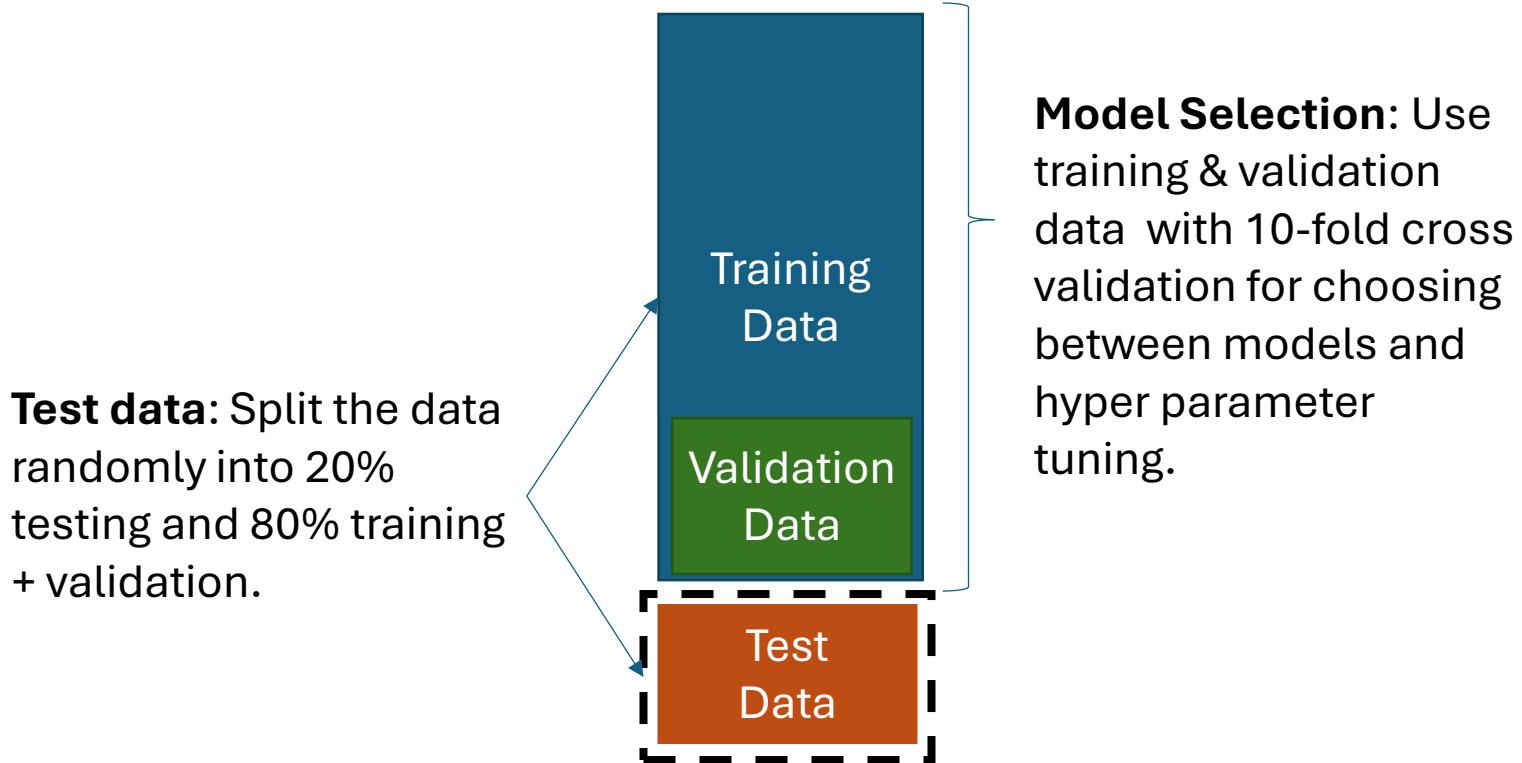
We do not want to overfit the hyperparameters!!!

Use a generalization error estimate twice:

1. **Train:** Learn models on the **training data** (without the validation data) using different hyperparameters.
  - A grid of possible hyperparameter combinations
  - greedy search
2. **Model Selection:** Evaluate the models using the **validation data** and choose the hyperparameters with the best accuracy. Rebuild the model using all the training data.
3. **Test** the final model using the **test data**.



# Typical Data Use with Model Selection



# Confidence Interval for Accuracy

- The observed accuracy is an **estimate** of the true accuracy of the model. How good is the estimate?
- Each prediction can be regarded as a **Bernoulli trial**: A Bernoulli trial (a biased coin toss) has 2 possible outcomes:  
heads (correct) or tails (wrong)

We use  $p$  for the true chance that a prediction is correct (= true accuracy).

- 
- Predictions for a test set of size  $N$  are a collection of  $N$  Bernoulli trials. The number of correct predictions  $x$  has a **Binomial distribution**:  
$$X \sim \text{Binomial}(N, p)$$
  - Example: Toss a fair coin 50 times, how many heads would turn up?  
Expected number of heads  $E[X] = Np = 50 \times 0.5 = 25$
  - **Application for Accuracy:** If we observe  $x$  correct predictions then the observed accuracy is

$$\hat{p} = x/N$$

Can we give bounds for the true accuracy of model  $p$ ?

# Confidence Interval for Accuracy

For large test sets ( $N > 30$ ) we can approximate the Binomial distribution

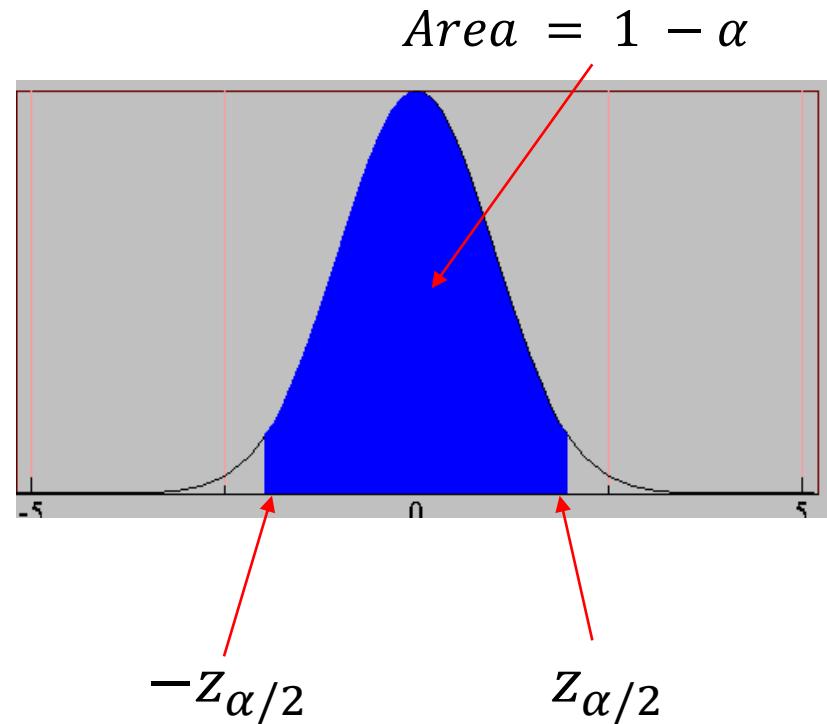
$$X \sim \text{Binomial}(N, p)$$

by a Normal distribution:

$$X \sim \text{Normal}(Np, Np(1 - p))$$

Confidence Interval for  $p = \frac{\hat{X}}{N}$   
(Wald Method):

$$\hat{p} \pm z_{\alpha/2} \sqrt{\frac{\hat{p}(1 - \hat{p})}{N}}$$



# Confidence Interval for Accuracy

Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:

1.  $N = 100, acc = 0.8$
2. Let  $1 - \alpha = 0.95$  (95% confidence)
3. Find the critical value for the normal distribution.  
 $z_{\alpha/2} = 1.96$
4. Calculate the interval around the accuracy.

$$\hat{p} \pm z_{\alpha/2} \sqrt{\frac{\hat{p}(1 - \hat{p})}{N}} = \begin{cases} 0.722 \\ 0.878 \end{cases}$$

$1 - \alpha/2$	$z_{\alpha/2}$
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

Table or  
R `qnorm(1 - \alpha/2)`

Data mining tools typically calculate this for us.

## EVALUATION

Relevance	✗ ○ ○ ○
Efficiency	○ ○ ✗ ○ ○
Effectiveness	○ ✗ ○ ○ ○
Sustainability	○ ○ ○ ○ ✗
Impact	○ ○ ○ ✗ ○

# Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
- Overfitting and other Practical Issues
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - **Model Comparison (Relative Performance)**
- Feature Selection

# Comparing Performance between 2 Models

Given two models, say  $M_1$  and  $M_2$ , which is better? This is a statistical **model selection** problem.

For large test sets ( $N > 30$ ) we can approximate the observed accuracies (sampled from a Binomial distribution) using the true but unknown model accuracies  $p_1$  and  $p_2$ :

$$\begin{aligned} acc_1 &\sim Normal(Np_1, Np_1(1 - p_1)) \\ acc_2 &\sim Normal(Np_2, Np_2(1 - p_2)) \end{aligned}$$

Perform a paired t-test with:

H0: There is no difference between the observed accuracies of the models.

H1: There is a difference.

## Notes

- **Hyperparameter** tuning is also a model selection problem.
- Comparing more than two models: You need to **correct for multiple comparisons!** For example, using Bonferroni correction or False Discovery Rate (FDR).



# Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
- Overfitting and other Practical Issues
- Model Selection and Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- **Feature Selection**

# Feature Selection

What features should be used in the model?

## Univariate feature importance score

- Measures how related each feature is to the class variable.
- E.g., chi-squared statistic, information gain.

## Feature subset selection

- Tries to find the best set of features.
- Often uses a black box approach where different subsets are evaluated using a greedy search strategy.
- E.g.: Stepwise backward selection tries to remove one feature at a time.



## Conclusion

---

- Classification is **supervised learning** with the goal to find a model that predicts well (i.e., has a low generalization error).
- **Generalization error** can be estimated using test sets/cross-validation and should be used for model selection.
- Model evaluation and comparison needs to take **model complexity** into account.