

# **Classification**

## **Lecture 1: Basics, Methods**

**Jing Gao**  
**SUNY Buffalo**

# Outline

- **Basics**
  - Problem, goal, evaluation
- **Methods**
  - Nearest Neighbor
  - Decision Tree
  - Naïve Bayes
  - Rule-based Classification
  - Logistic Regression
  - Support Vector Machines
  - Ensemble methods
  - .....
- **Advanced topics**
  - Semi-supervised Learning
  - Multi-view Learning
  - Transfer Learning
  - .....

# Readings

- Tan, Steinbach, Kumar, Chapters 4 and 5.
- Han, Kamber, Pei. Data Mining: Concepts and Techniques. Chapters 8 and 9.
- Additional readings posted on website

# Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

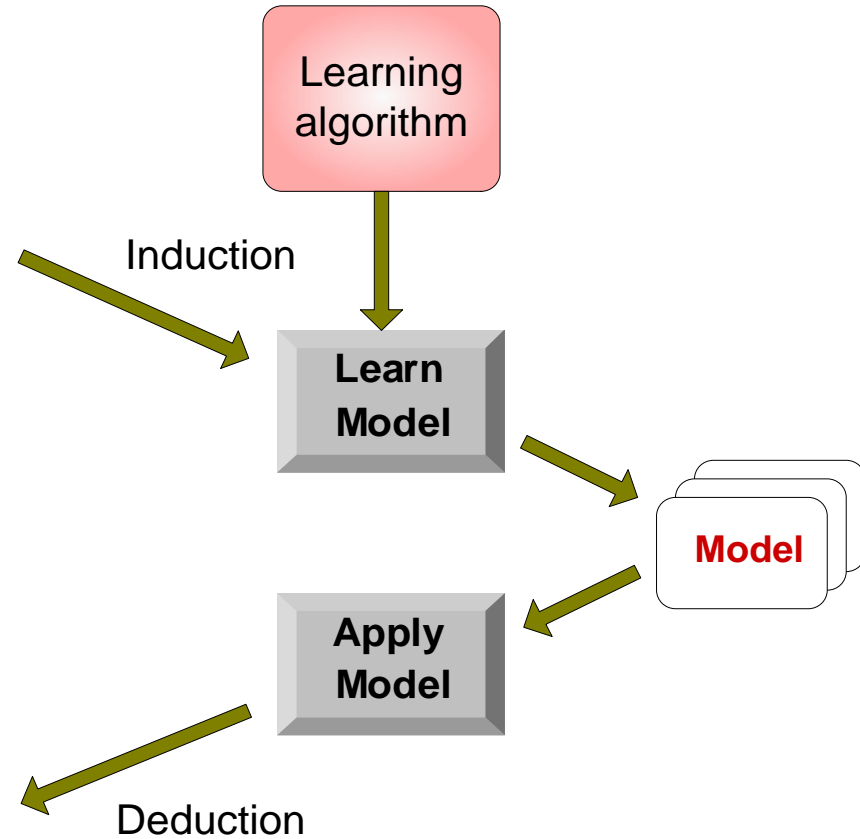
# Illustrating Classification Task

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

Training 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	?

Test Set



# Examples of Classification Task

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

# Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

- a: TP (true positive)
- b: FN (false negative)
- c: FP (false positive)
- d: TN (true negative)

# Metrics for Performance Evaluation

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

- Most widely-used metric:

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



# Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is  $9990/10000 = 99.9\%$ 
  - Accuracy is misleading because model does not detect any class 1 example

# Cost-Sensitive Measures

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

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

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

# Methods of Estimation

- **Holdout**
  - Reserve 2/3 for training and 1/3 for testing
- **Random subsampling**
  - Repeated holdout
- **Cross validation**
  - Partition data into  $k$  disjoint subsets
  - $k$ -fold: train on  $k-1$  partitions, test on the remaining one
  - Leave-one-out:  $k=n$
- **Stratified sampling**
  - oversampling vs undersampling
- **Bootstrap**
  - Sampling with replacement

# Classification Techniques

- Nearest Neighbor
- Decision Tree
- Naïve Bayes
- Rule-based Classification
- Logistic Regression
- Support Vector Machines
- Ensemble methods
- .....

# Nearest Neighbor Classifiers

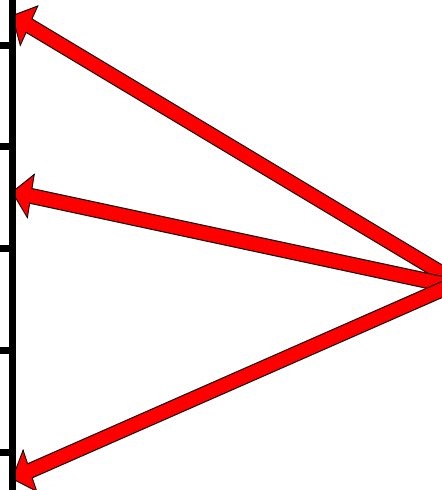
Set of Stored Cases

Atr1	.....	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

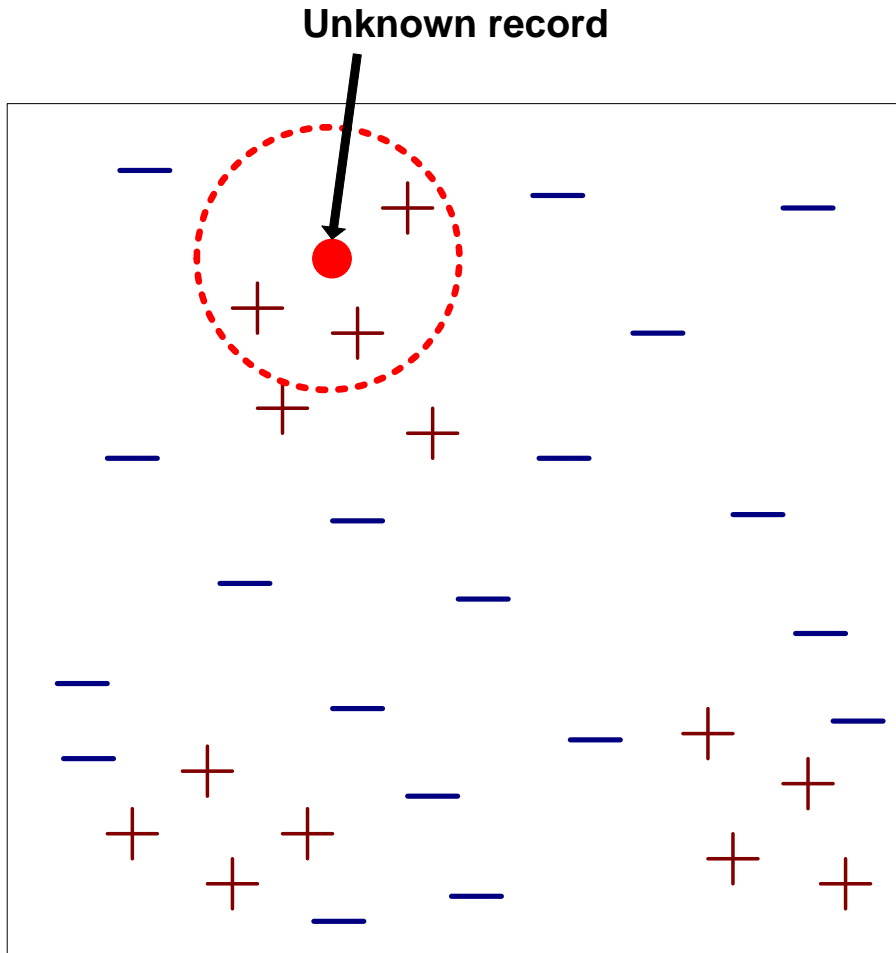
- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

Atr1	.....	AtrN

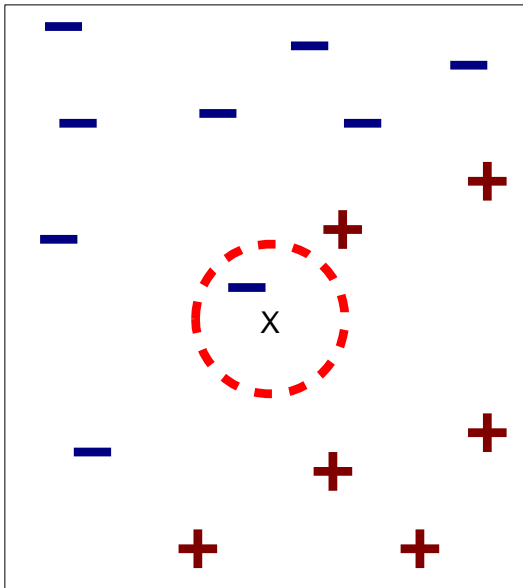


# Nearest-Neighbor Classifiers

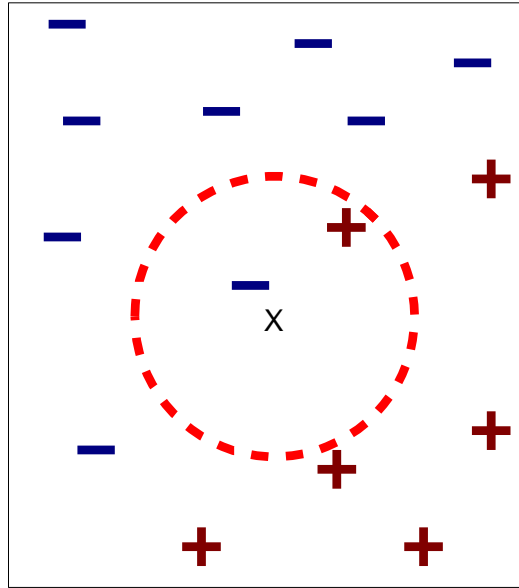


- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of  $k$ , the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify  $k$  nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

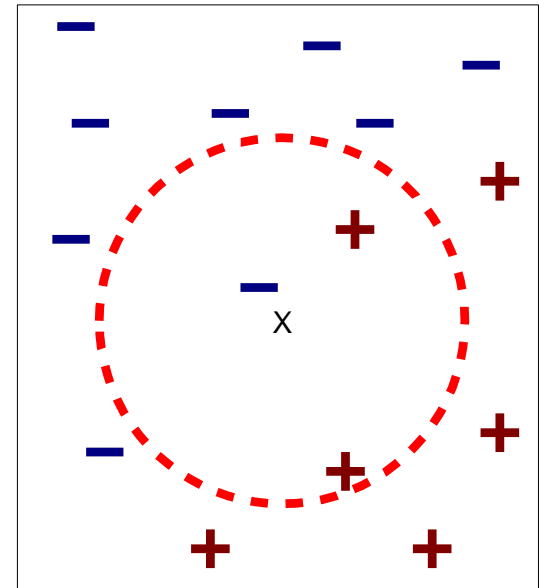
# Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

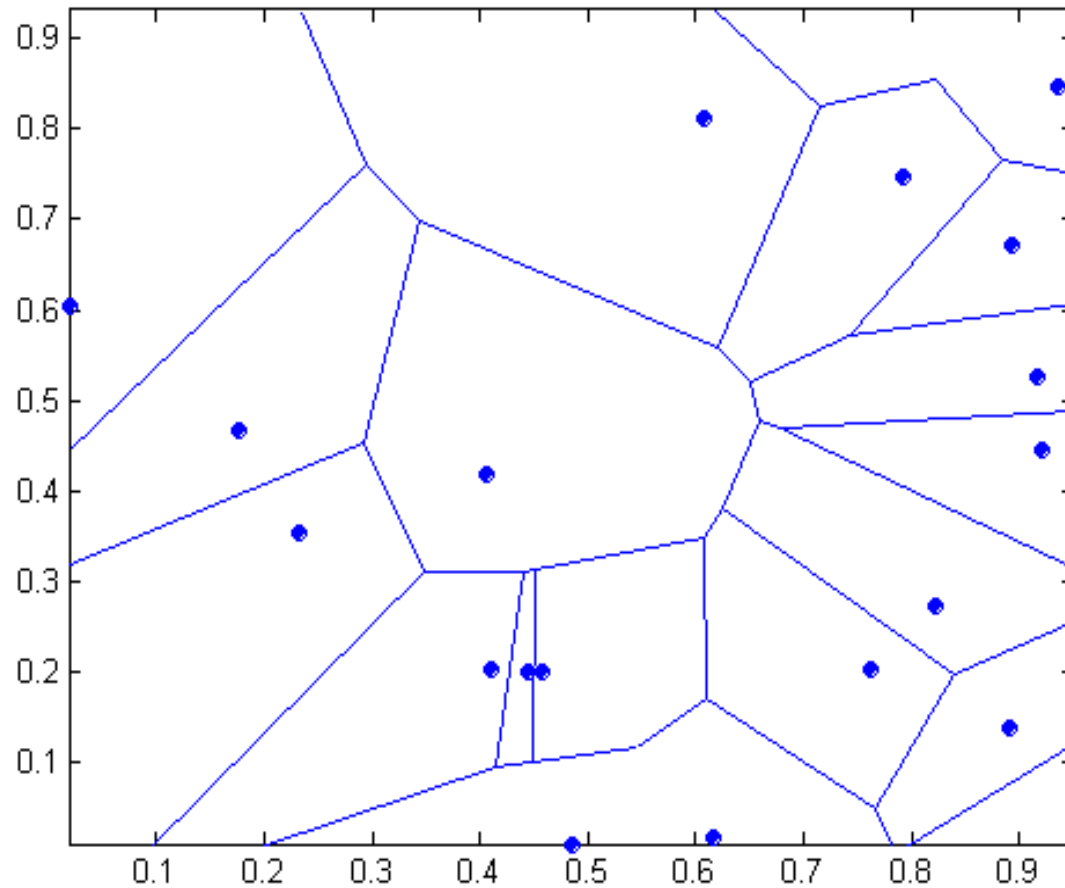


(c) 3-nearest neighbor

K-nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distance to  $x$

# 1 nearest-neighbor

Voronoi Diagram





# Nearest Neighbor Classification

- Compute distance between two points:

- Euclidean distance

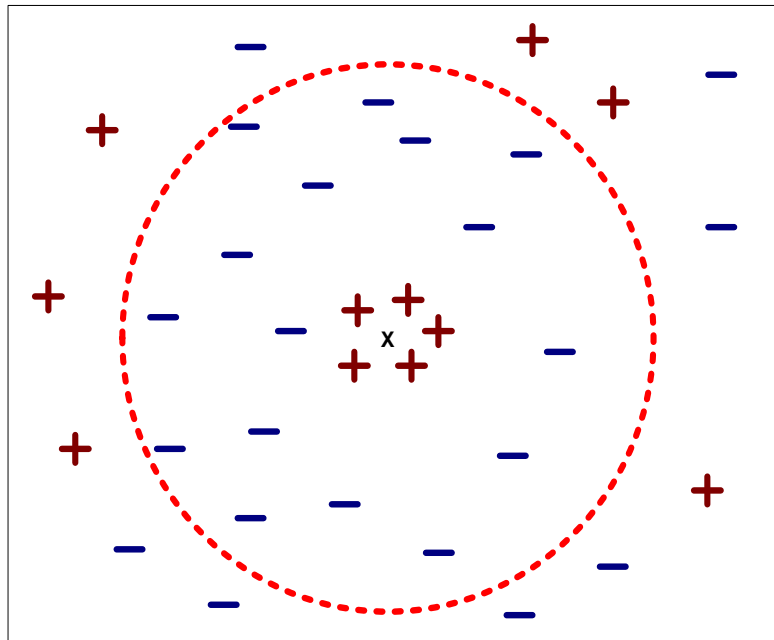
$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor,  $w = 1/d^2$

# Nearest Neighbor Classification

- **Choosing the value of k:**

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes



# Nearest Neighbor Classification

- **Scaling issues**

- Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Example:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from \$10K to \$1M

# Nearest neighbor Classification

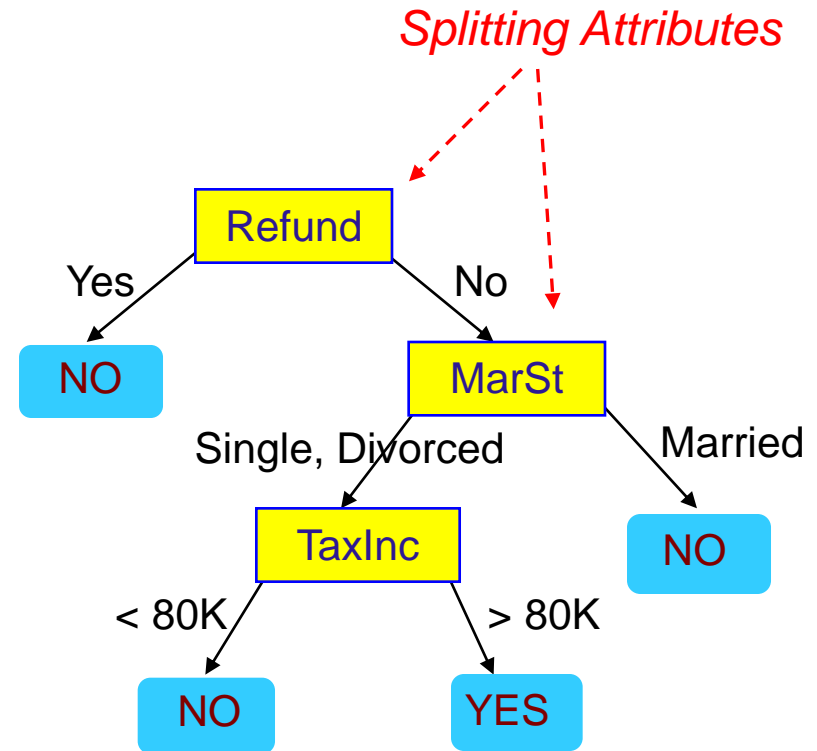
- **k-NN classifiers are lazy learners**
  - It does not build models explicitly
  - Different from eager learners such as decision tree induction
  - Classifying unknown records are relatively expensive

# 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

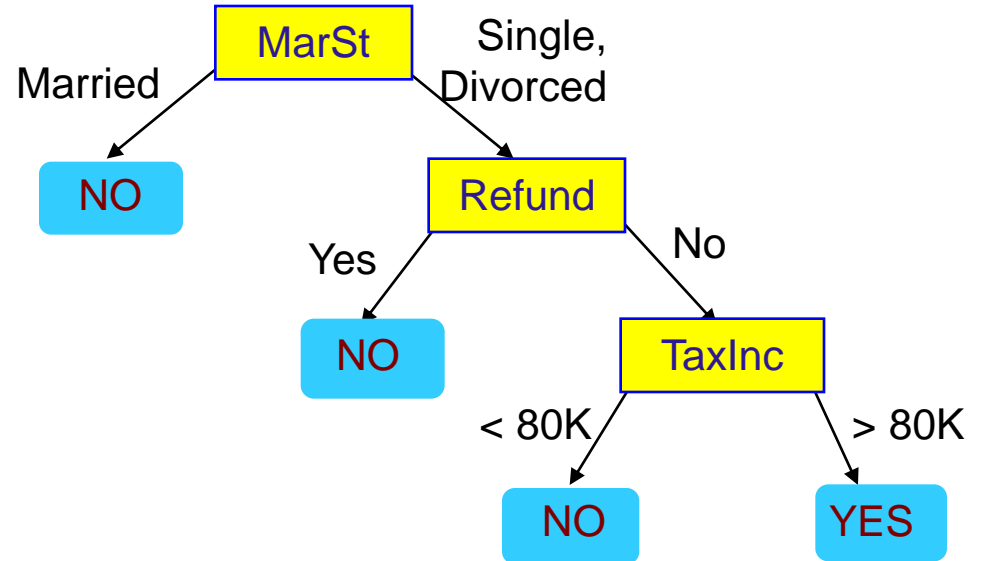


Model: Decision Tree

# Another Example of Decision Tree

categorical  
categorical  
continuous  
class

<i>Tid</i>	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



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

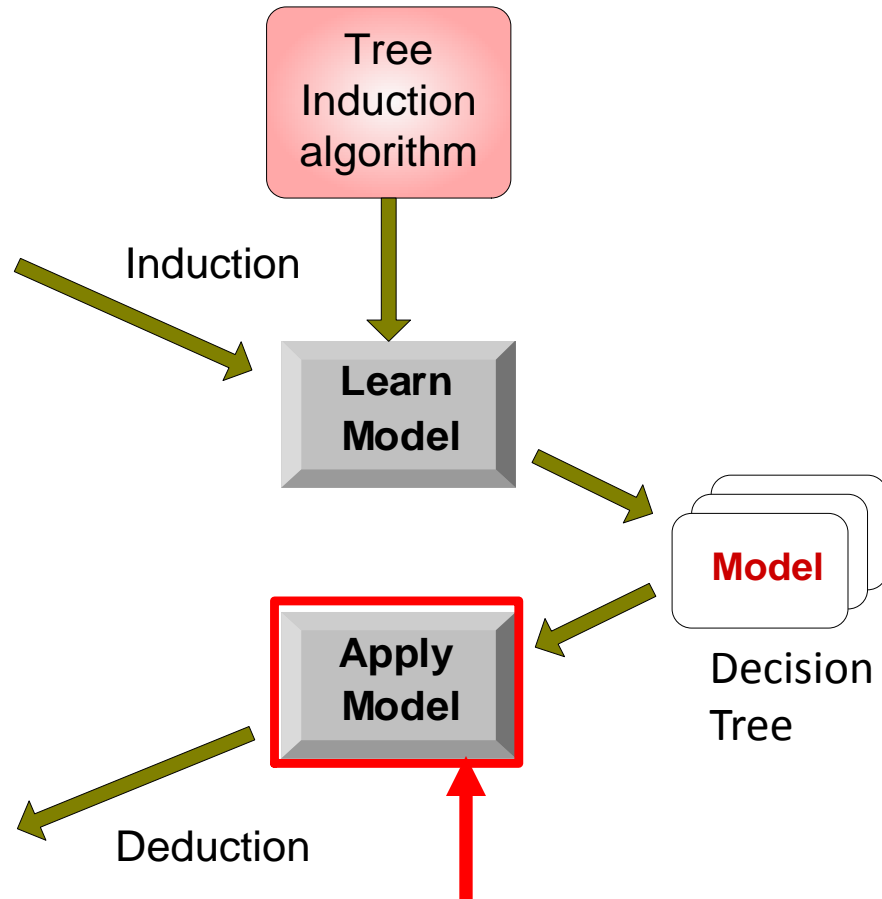
# Decision Tree Classification Task

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

Training Set

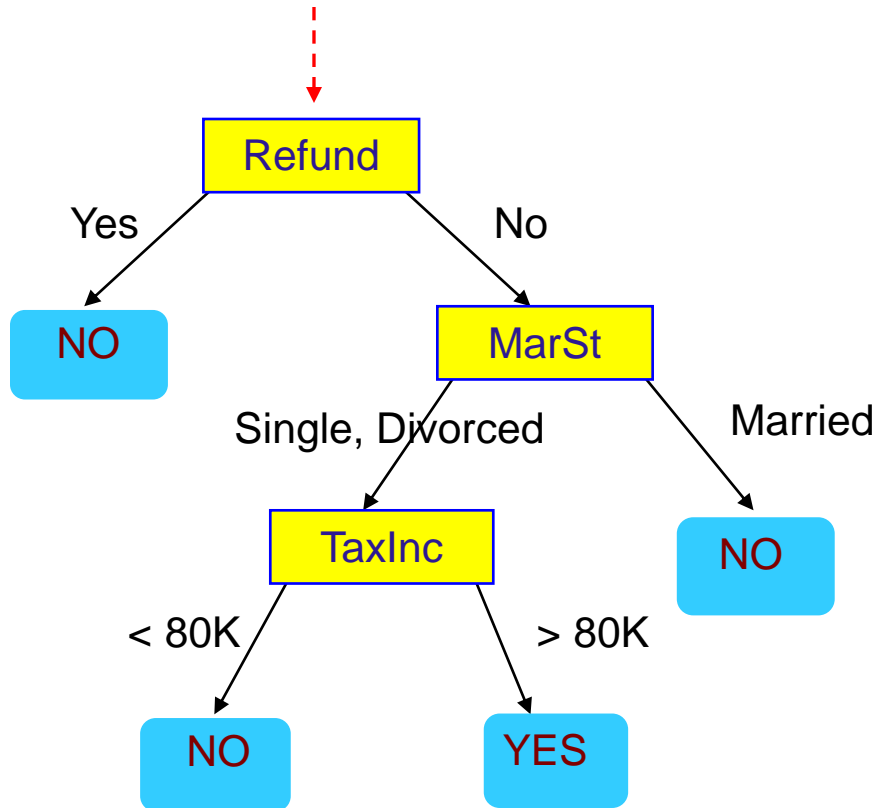
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



# Apply Model to Test Data

Start from the root of tree.



Test Data

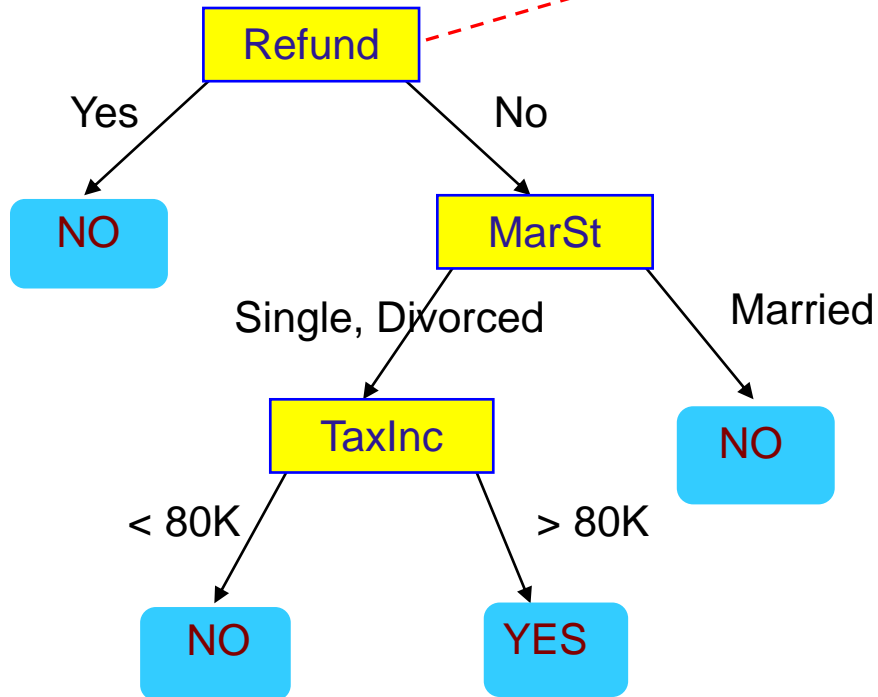
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

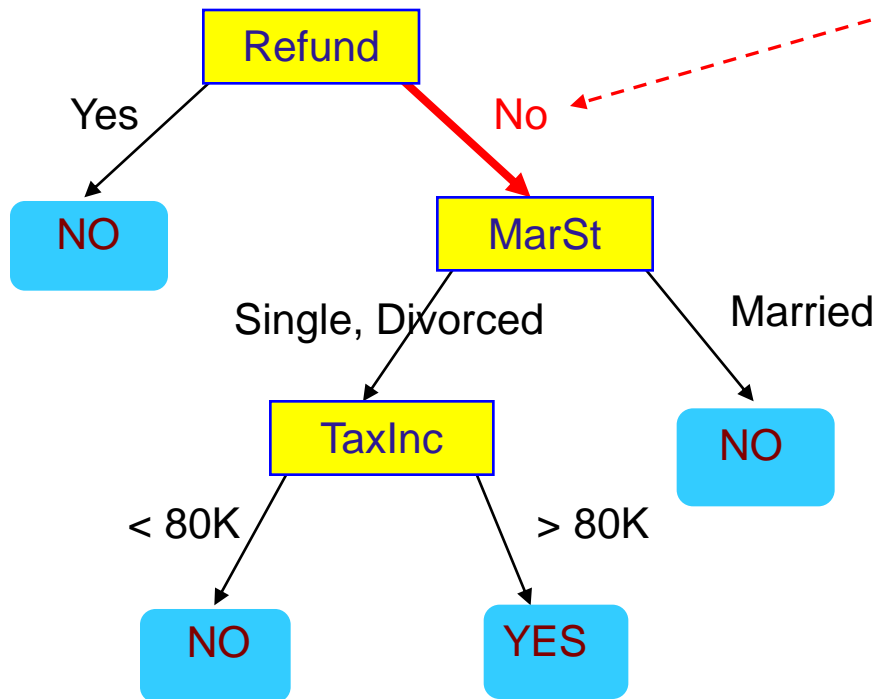
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

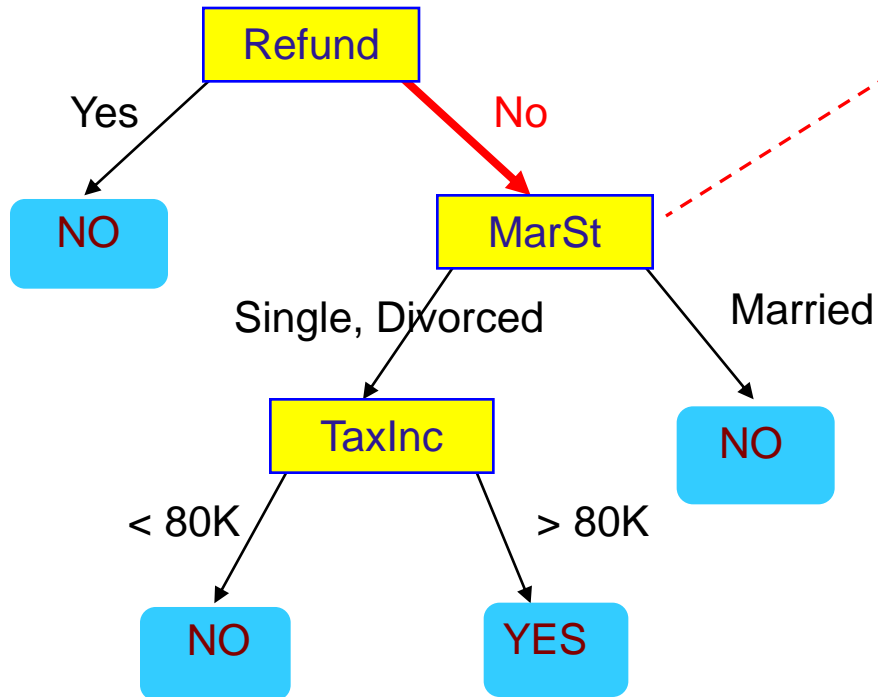
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

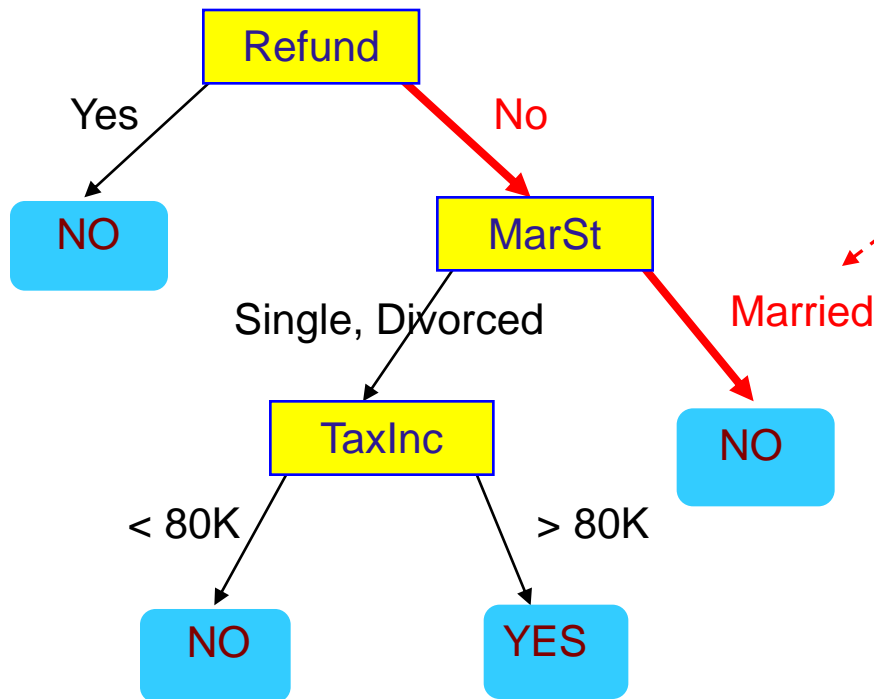
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

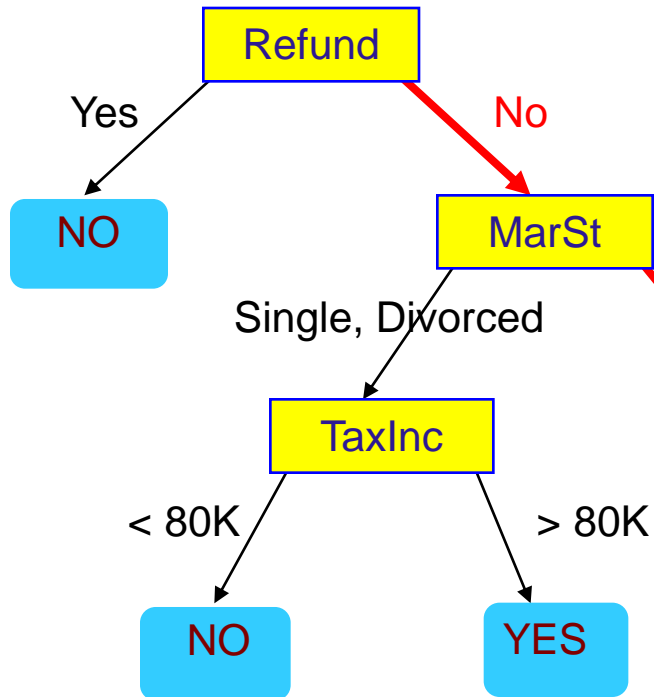
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



# Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to "No"

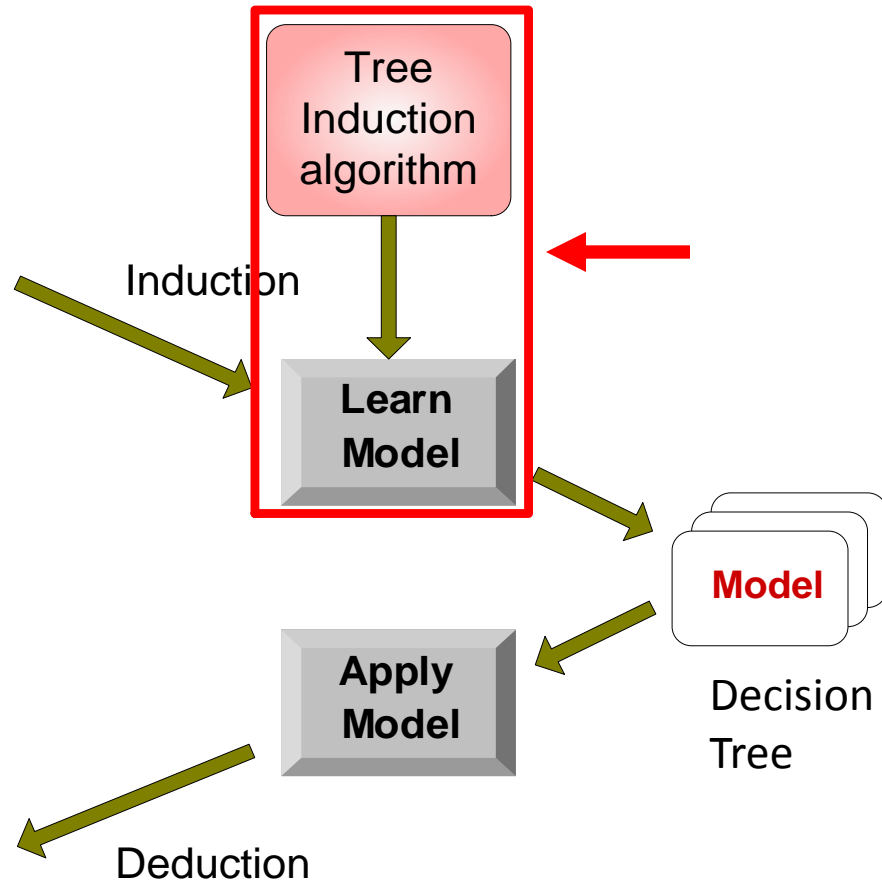
# Decision Tree Classification Task

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

Training 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	?

Test Set



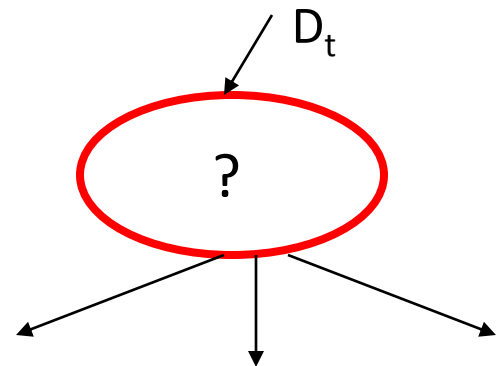
# Decision Tree Induction

- **Many Algorithms:**
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT
  - .....

# General Structure of Hunt's Algorithm

- Let  $D_t$  be the set of training records that reach a node  $t$
- **General Procedure:**
  - If  $D_t$  contains records that belong the same class  $y_t$ , then  $t$  is a leaf node labeled as  $y_t$
  - If  $D_t$  contains records that belong to more than one class, use an attribute to split the data into smaller subsets. Recursively apply the procedure to each subset

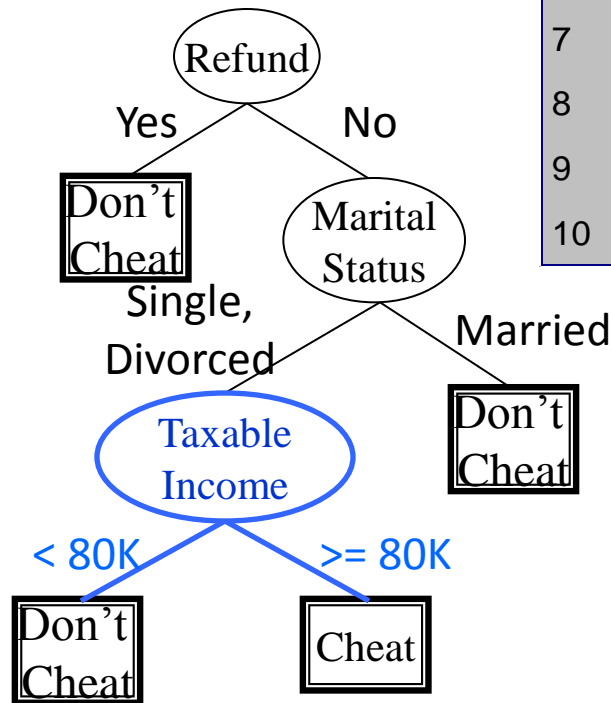
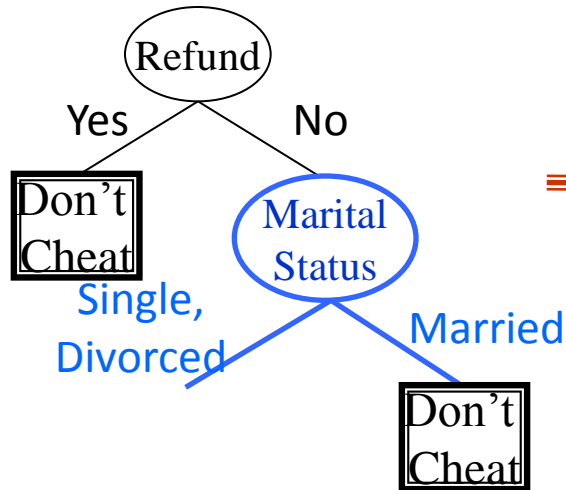
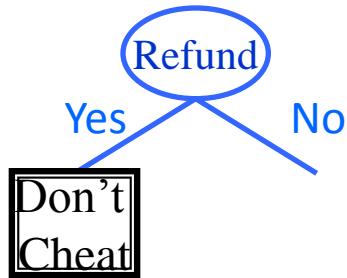
<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes





# Hunt's Algorithm

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



# Tree Induction

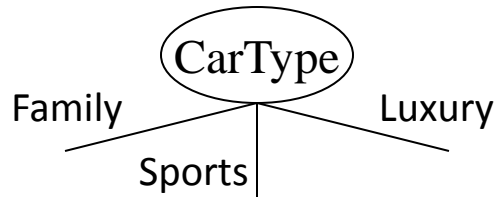
- **Greedy strategy**
  - Split the records based on an attribute test that optimizes certain criterion
- **Issues**
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

# How to Specify Test Condition?

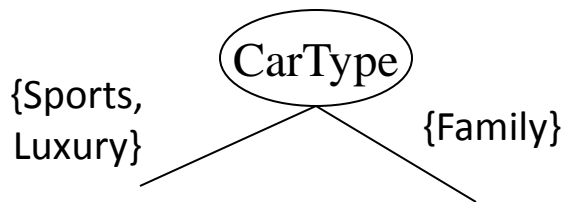
- **Depends on attribute types**
  - Nominal
  - Ordinal
  - Continuous
- **Depends on number of ways to split**
  - 2-way split
  - Multi-way split

# Splitting Based on Nominal Attributes

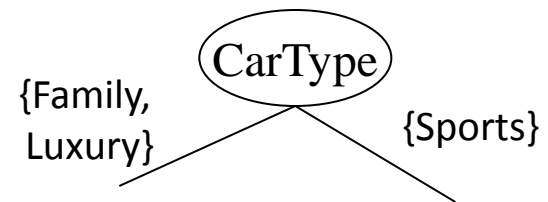
- **Multi-way split:** Use as many partitions as distinct values



- **Binary split:** Divides values into two subsets  
Need to find optimal partitioning

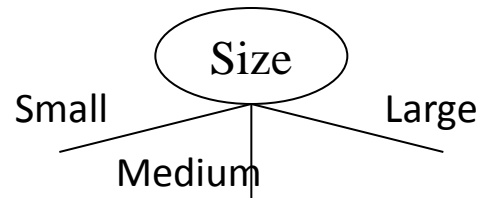


OR

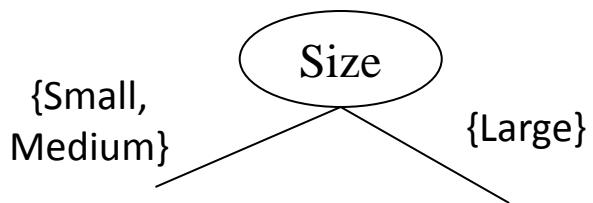


# Splitting Based on Ordinal Attributes

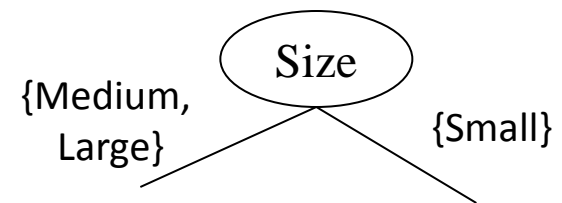
- **Multi-way split:** Use as many partitions as distinct values.



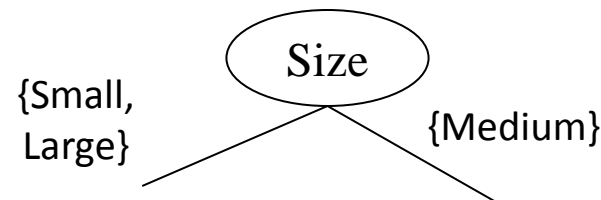
- **Binary split:** Divides values into two subsets  
Need to find optimal partitioning



OR



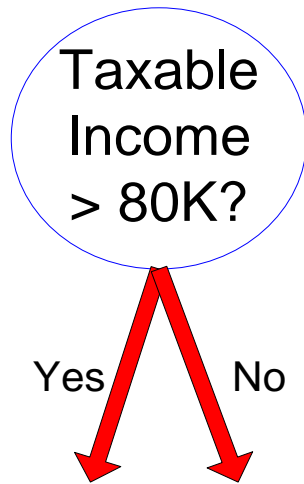
- What about this split?



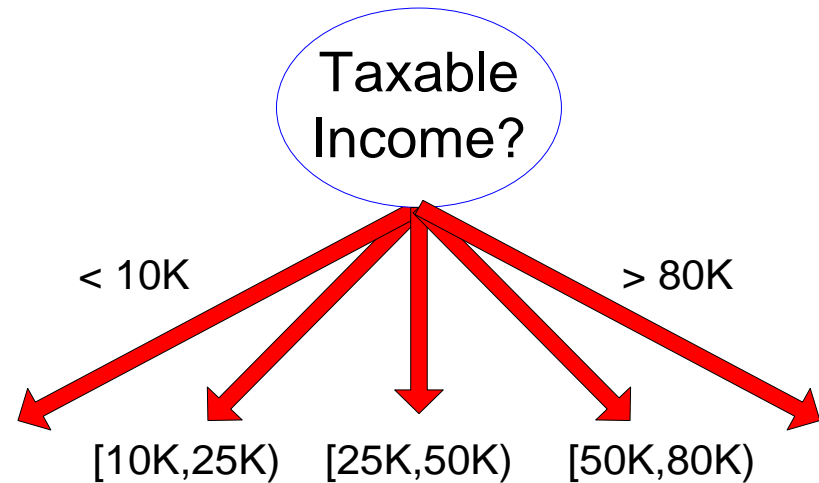
# Splitting Based on Continuous Attributes

- Different ways of handling
  - **Discretization** to form an ordinal categorical attribute
  - **Binary Decision:**  $(A < v)$  or  $(A \geq v)$ 
    - consider all possible splits and finds the best cut
    - can be more computation intensive

# Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

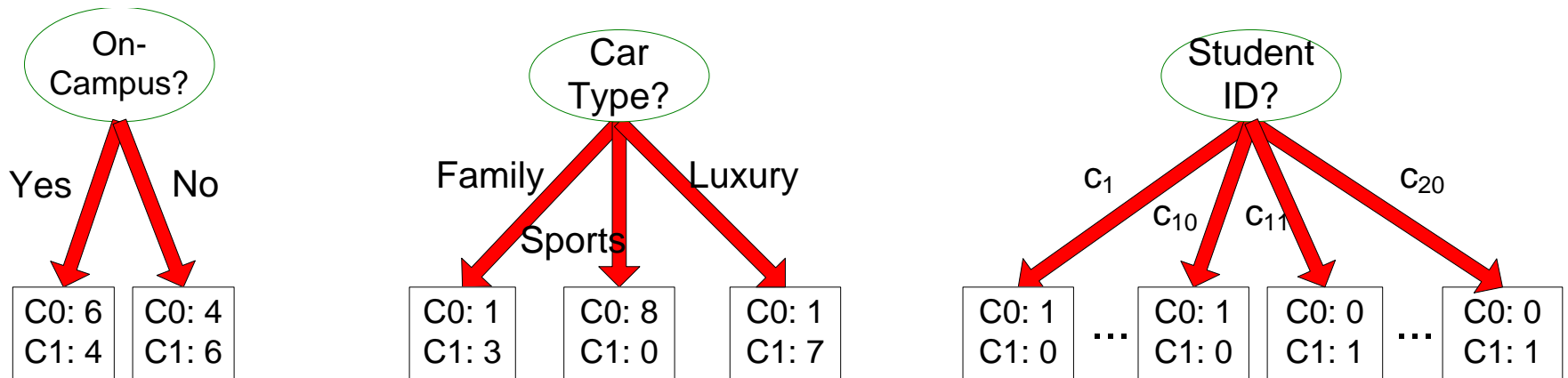
# Tree Induction

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



# How to determine the Best Split

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



Which test condition is the best?

# How to determine the Best Split

- **Greedy approach:**

- Nodes with **homogeneous** class distribution are preferred

- **Need a measure of node impurity:**

C0: 5
C1: 5

Non-homogeneous,  
High degree of impurity

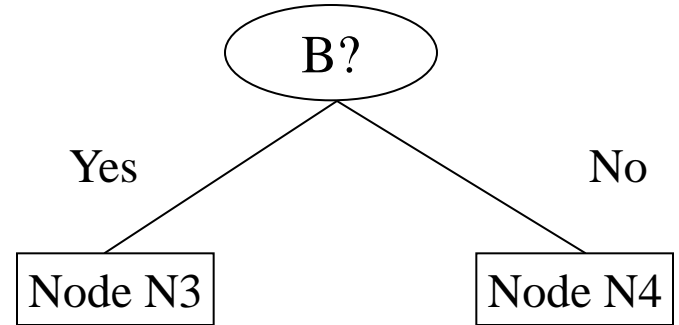
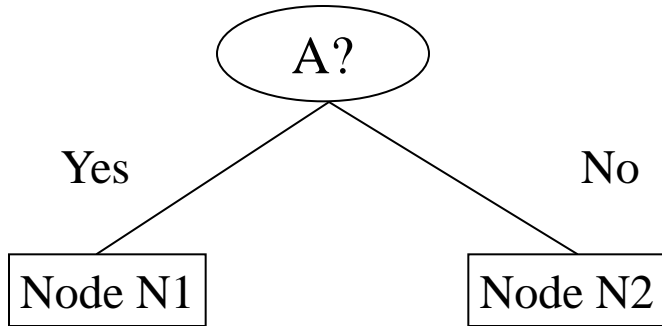
C0: 9
C1: 1

Homogeneous,  
Low degree of impurity

# How to Find the Best Split

Before Splitting:

C0	<b>N00</b>	→ M0
C1	<b>N01</b>	



C0	<b>N10</b>
C1	<b>N11</b>

C0	<b>N20</b>
C1	<b>N21</b>

C0	<b>N30</b>
C1	<b>N31</b>

C0	<b>N40</b>
C1	<b>N41</b>



M1



M2



M12



M3



M4



M34

$$\text{Gain} = M0 - M12 \text{ vs } M0 - M34$$

# Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

# Measure of Impurity: GINI

- Gini Index for a given node  $t$  :

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

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

- Maximum ( $1 - 1/n_c$ ) when records are equally distributed among all classes, implying least interesting information
- Minimum (0) when all records belong to one class, implying most useful information

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 for computing GINI

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

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

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

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

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

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

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

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

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

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

# Splitting Based on GINI

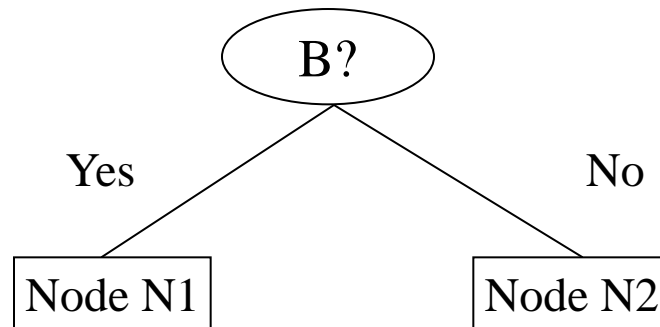
- Used in CART, SLIQ, SPRINT.
- When a node  $p$  is split into  $k$  partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where,  $n_i$  = number of records at child  $i$ ,  
 $n$  = number of records at node  $p$ .

# Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for



	<b>Parent</b>
C1	<b>6</b>
C2	<b>6</b>
<b>Gini = 0.500</b>	

$$\begin{aligned} \text{Gini}(N1) &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.408 \end{aligned}$$

$$\begin{aligned} \text{Gini}(N2) &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.32 \end{aligned}$$

	<b>N1</b>	<b>N2</b>
C1	<b>5</b>	<b>1</b>
C2	<b>2</b>	<b>4</b>
<b>Gini=0.333</b>		

$$\begin{aligned} \text{Gini(Children)} &= 7/12 * 0.408 + \\ &\quad 5/12 * 0.32 \\ &= 0.371 \end{aligned}$$



# Entropy

- Entropy at a given node t:

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

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

## – Measures purity of a node

- Maximum ( $\log n_c$ ) when records are equally distributed among all classes implying least information
- Minimum (0.0) when all records belong to one class, implying most information

# Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

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

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

$$Entropy = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

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

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

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

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

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

$$Entropy = - (2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

# Splitting Based on Information Gain

- **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 and C4.5

# Splitting Criteria based on Classification Error

- Classification error at a node  $t$  :

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

- Measures misclassification error made by a node.
  - Maximum  $(1 - 1/n_c)$  when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

# Examples for Computing Error

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

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

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

$$Error = 1 - \max(0, 1) = 1 - 1 = 0$$

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

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

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

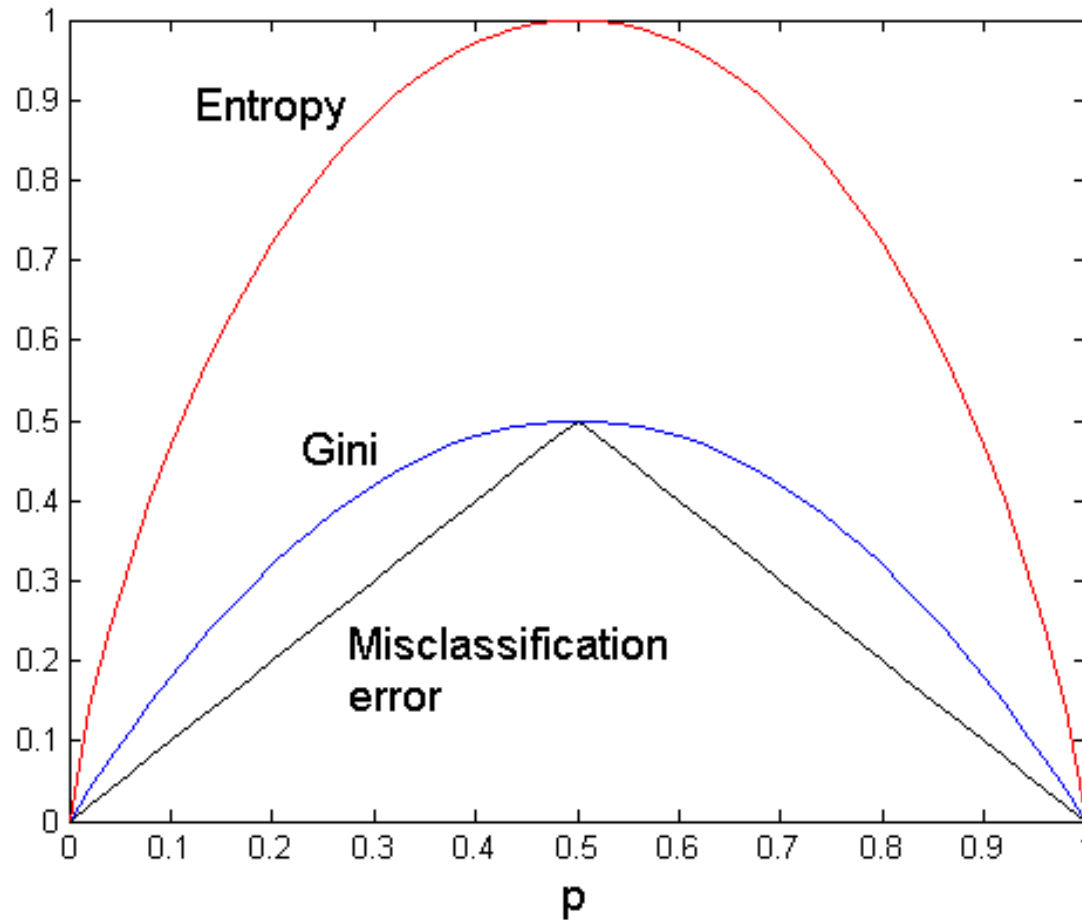
C1	<b>2</b>
C2	<b>4</b>

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

$$Error = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

# Comparison among Splitting Criteria

For a 2-class problem:



# Tree Induction

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

# Stopping Criteria for Tree Induction

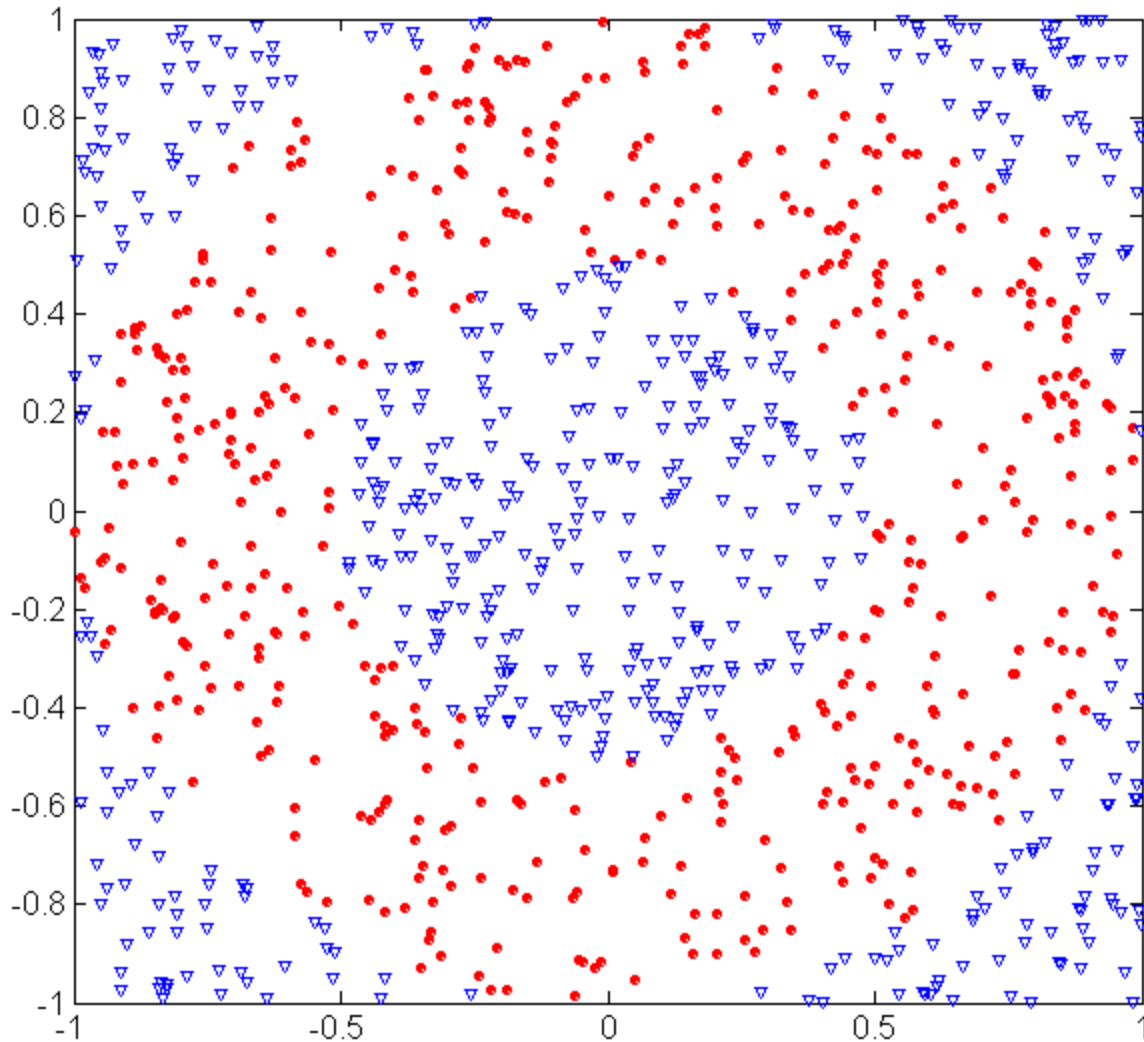
- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)



# Decision Tree Based Classification

- **Advantages:**
  - 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

# Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

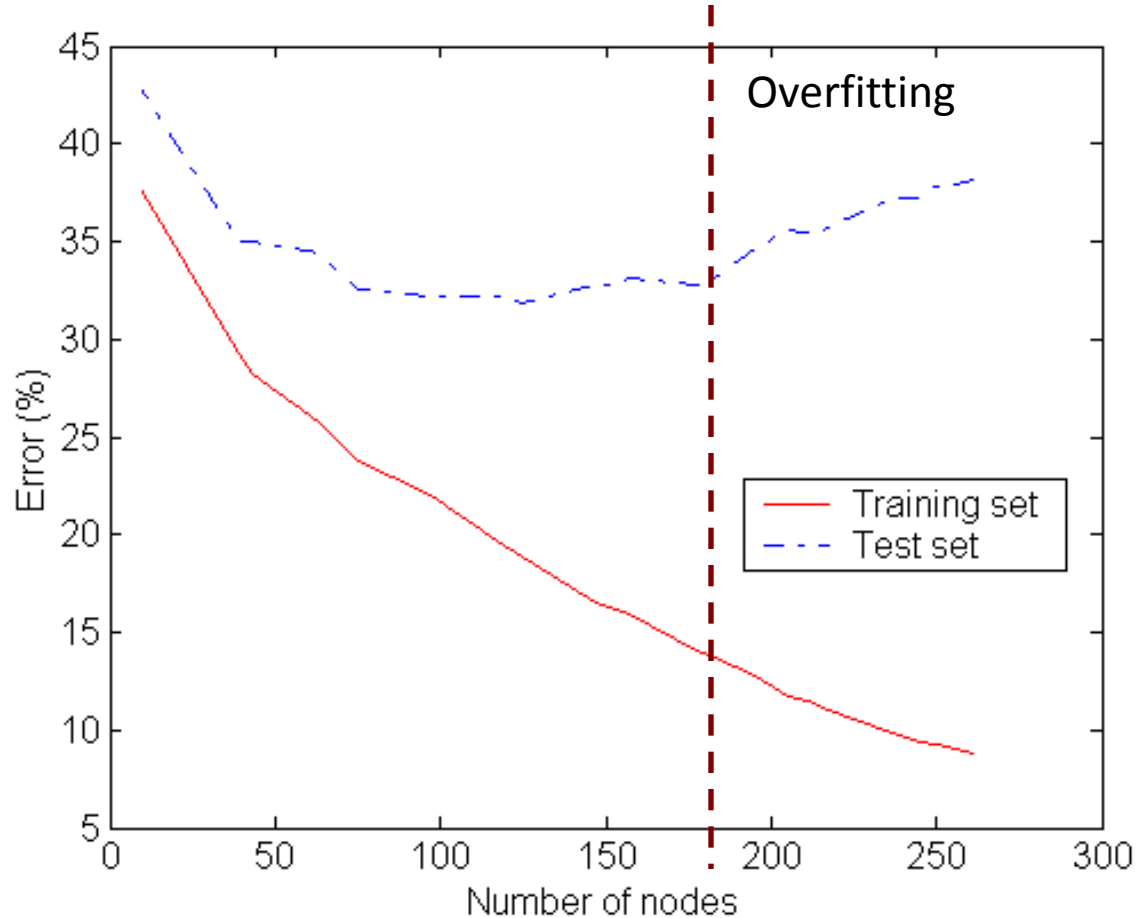
$$0.5 \leq \text{sqrt}(x_1^2 + x_2^2) \leq 1$$

Triangular points:

$$\text{sqrt}(x_1^2 + x_2^2) > 0.5 \text{ or}$$

$$\text{sqrt}(x_1^2 + x_2^2) < 1$$

# Underfitting and Overfitting



# Occam's Razor

- Given two models of similar errors, one should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

# How to Address Overfitting

- **Pre-Pruning (Early Stopping Rule)**

- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

# How to Address Overfitting

- **Post-pruning**
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree

# Handling Missing Attribute Values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

# Computing Impurity Measure

Tid	Refund	Marital Status	Taxable Income	Class
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	?	Single	90K	Yes

Missing value

Before Splitting:

Entropy(Parent)

$$= -0.3 \log(0.3) - (0.7) \log(0.7) = 0.8813$$

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

Entropy(Refund=Yes) = 0

Entropy(Refund=No)

$$= -(2/6) \log(2/6) - (4/6) \log(4/6) = 0.9183$$

Entropy(Children)

$$= 0.3 (0) + 0.6 (0.9183) = 0.551$$

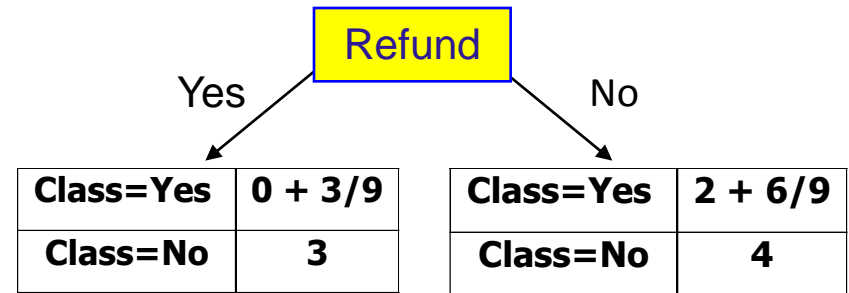
$$\text{Gain} = 0.9 \times (0.8813 - 0.551) = 0.3303$$



# Distribute Instances

<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
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

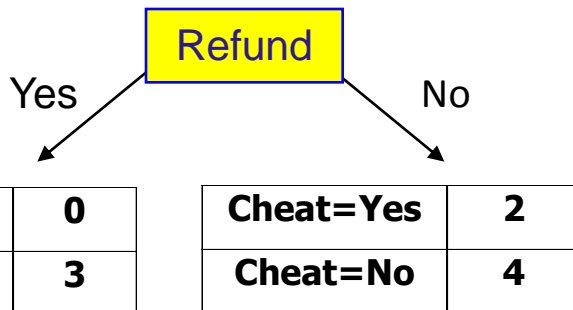
<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

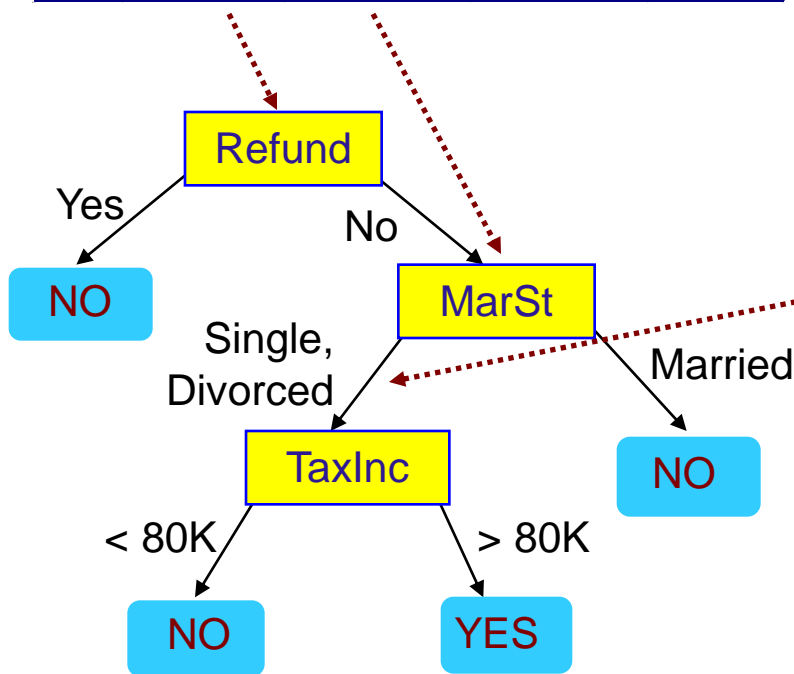


# Classify Instances

New record:

<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?

	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	6/9	1	1	2.67
Total	3.67	2	1	6.67



Probability that Marital Status = Married is  $3.67/6.67$

Probability that Marital Status = {Single, Divorced} is  $3/6.67$

# Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

# Data Fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

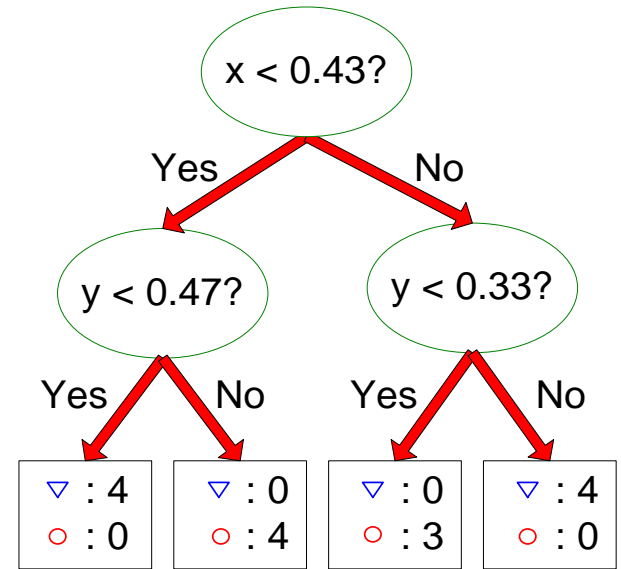
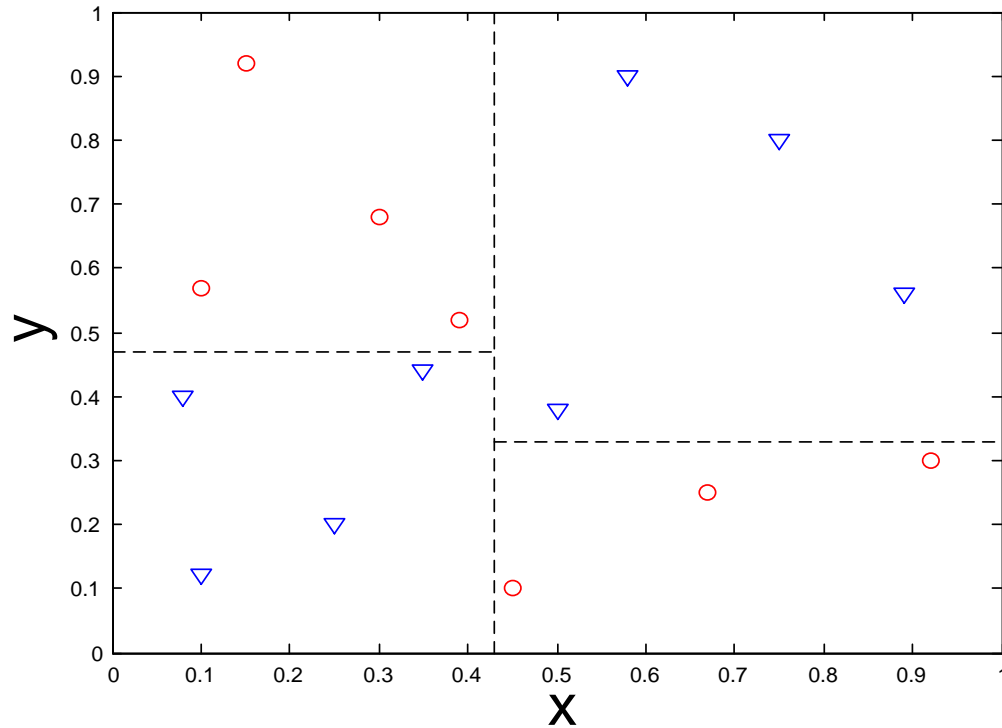
# Search Strategy

- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

# Expressiveness

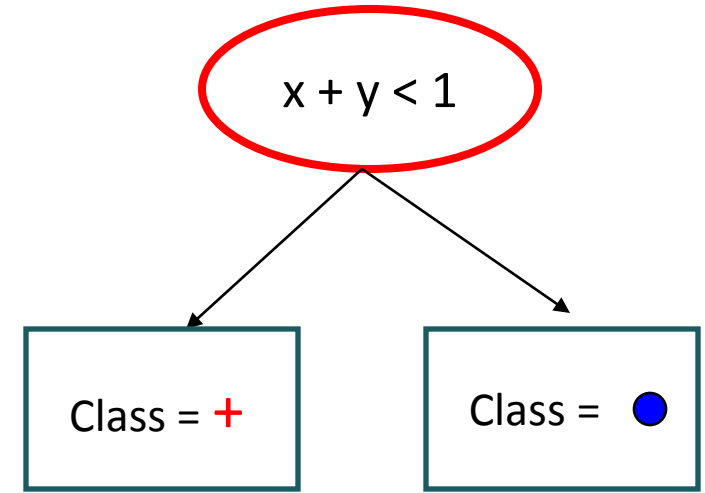
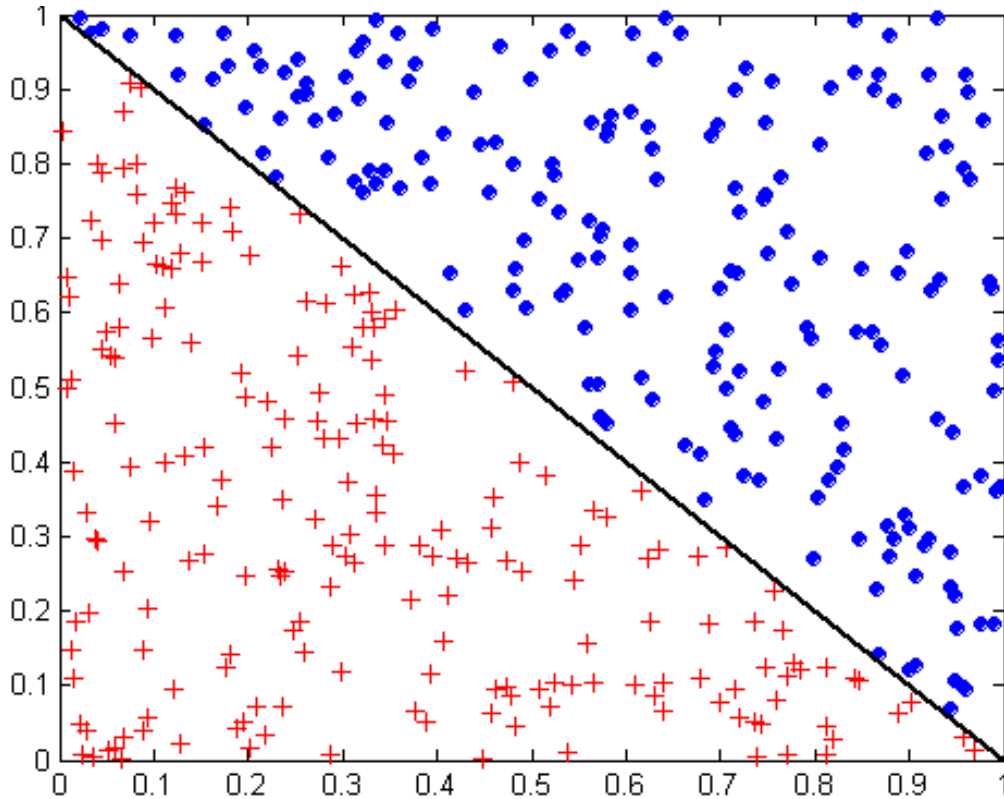
- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

# Decision Boundary



- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

# Oblique Decision Trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive



# Take-away Message

- What's classification?
- How to evaluate classification model?
- How to use decision tree to make predictions?
- How to construct a decision tree from training data?
- How to compute gini index, entropy, misclassification error?
- How to avoid overfitting by pre-pruning or post-pruning decision tree?