

# CDA 305: Decision Tree

---

# 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

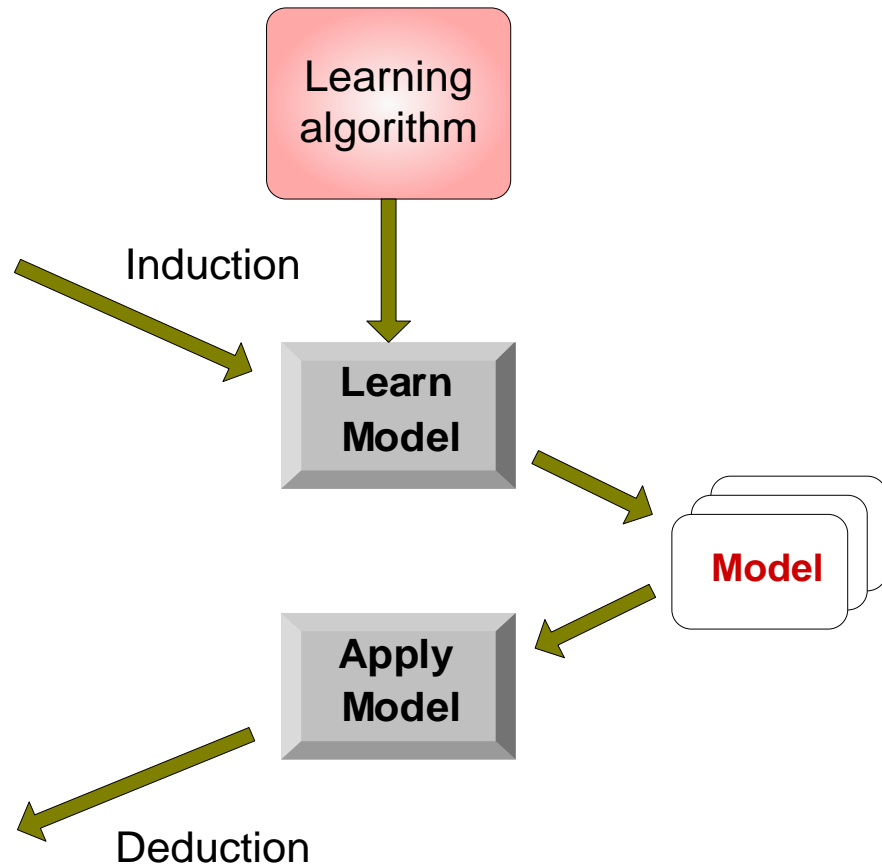
# Illustrating Classification Task

| 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

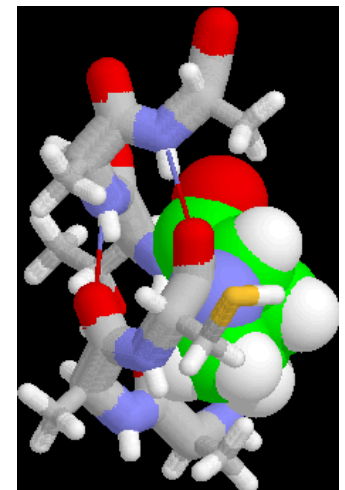
| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|-----|---------|---------|---------|-------|
| 11  | No      | Small   | 55K     | ?     |
| 12  | Yes     | Medium  | 80K     | ?     |
| 13  | Yes     | Large   | 110K    | ?     |
| 14  | No      | Small   | 95K     | ?     |
| 15  | No      | Large 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

Test Set



# Examples of Classification Task

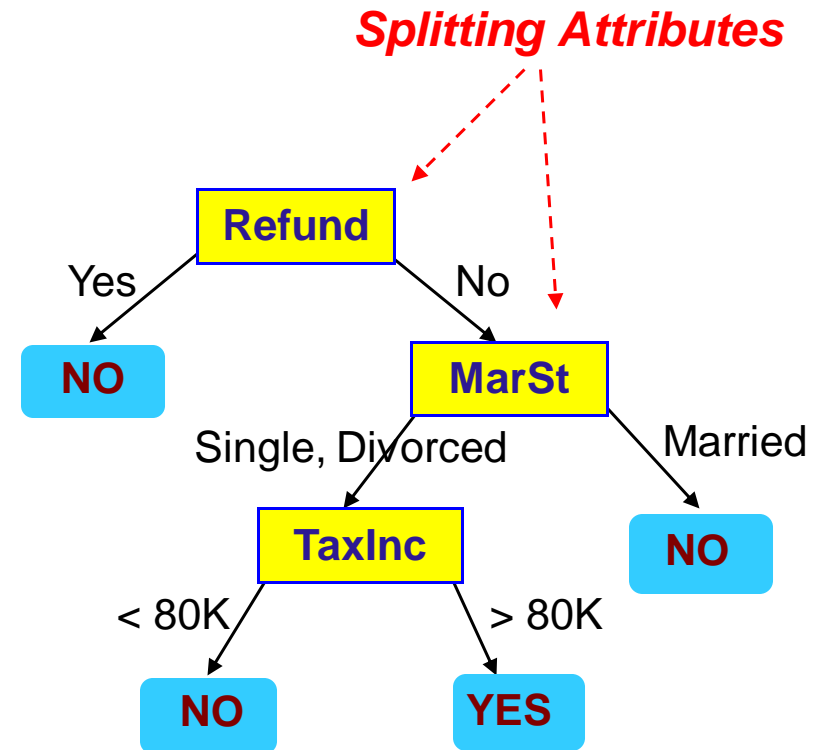
- Predicting tumor cells as **benign** or **malignant**
- Classifying credit card transactions as **legitimate** or **fraudulent**
- Classifying secondary structures of protein as **alpha-helix**, **beta-sheet**, or **random coil**
- Categorizing news stories as **finance**, **weather**, **entertainment**, **sports**, etc



# Example of a Decision Tree

| <i>Tid</i> | <i>Refund</i> | <i>Marital Status</i> | <i>Taxable Income</i> | <i>Cheat</i> |
|------------|---------------|-----------------------|-----------------------|--------------|
| 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

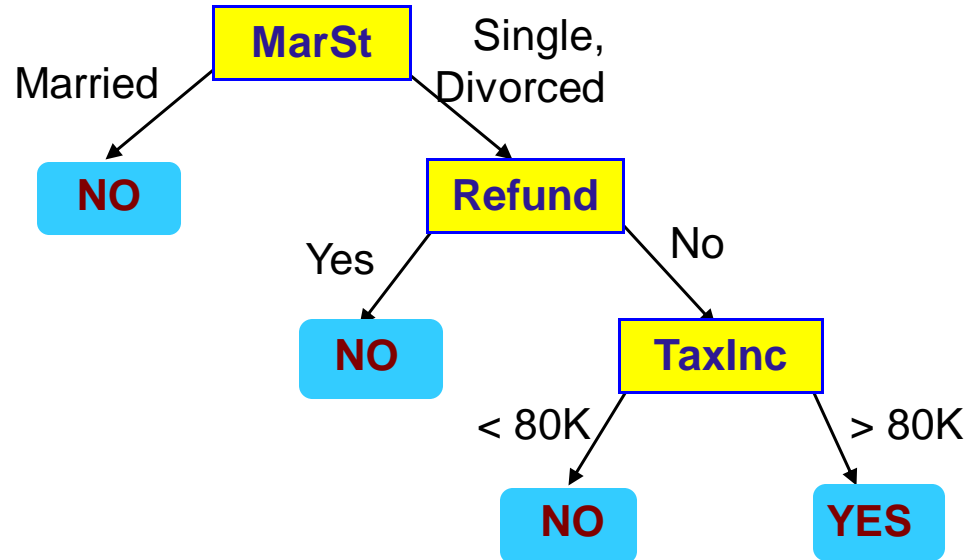
# Another Example of Decision Tree

| <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   |

categorical

categorical

continuous  
class



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

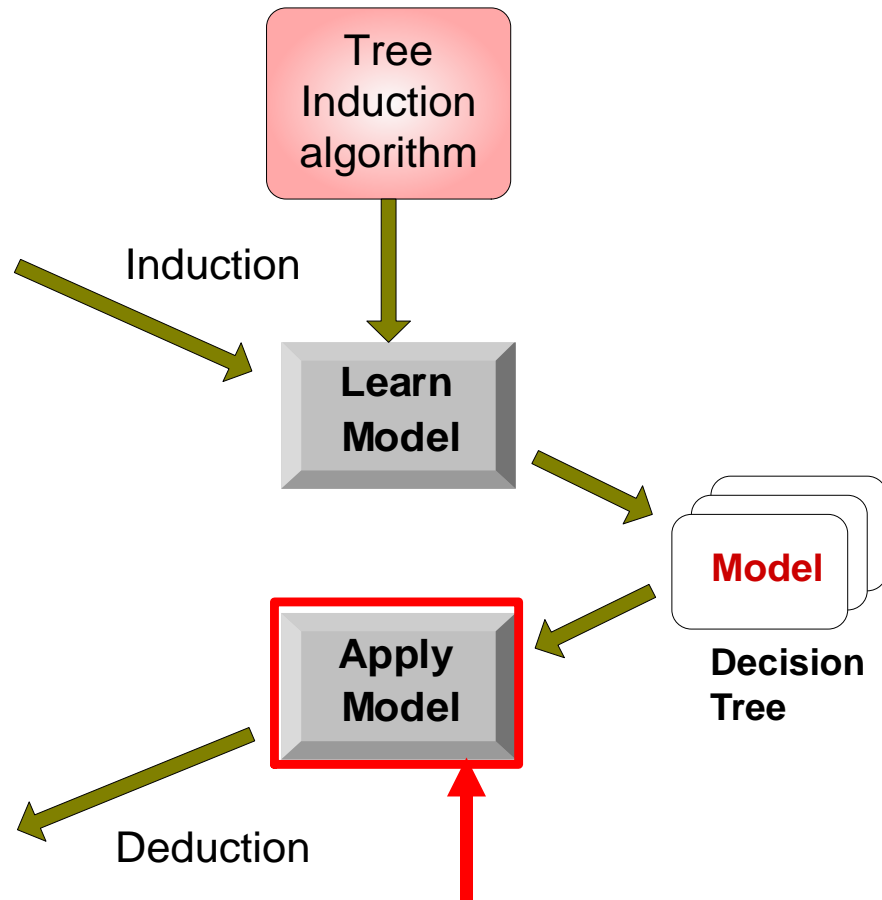
# Decision Tree Classification Task

| 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

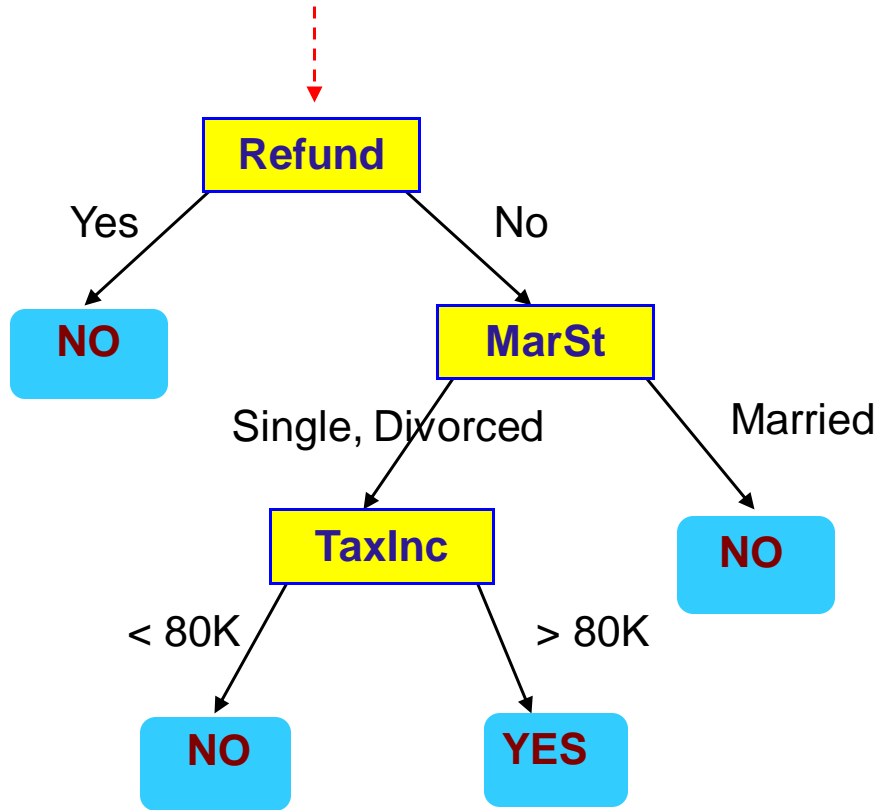
| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|-----|---------|---------|---------|-------|
| 11  | No      | Small   | 55K     | ?     |
| 12  | Yes     | Medium  | 80K     | ?     |
| 13  | Yes     | Large   | 110K    | ?     |
| 14  | No      | Small   | 95K     | ?     |
| 15  | No      | Large 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

Test Set



# Apply Model to Test Data

Start from the root of tree



## Test Data

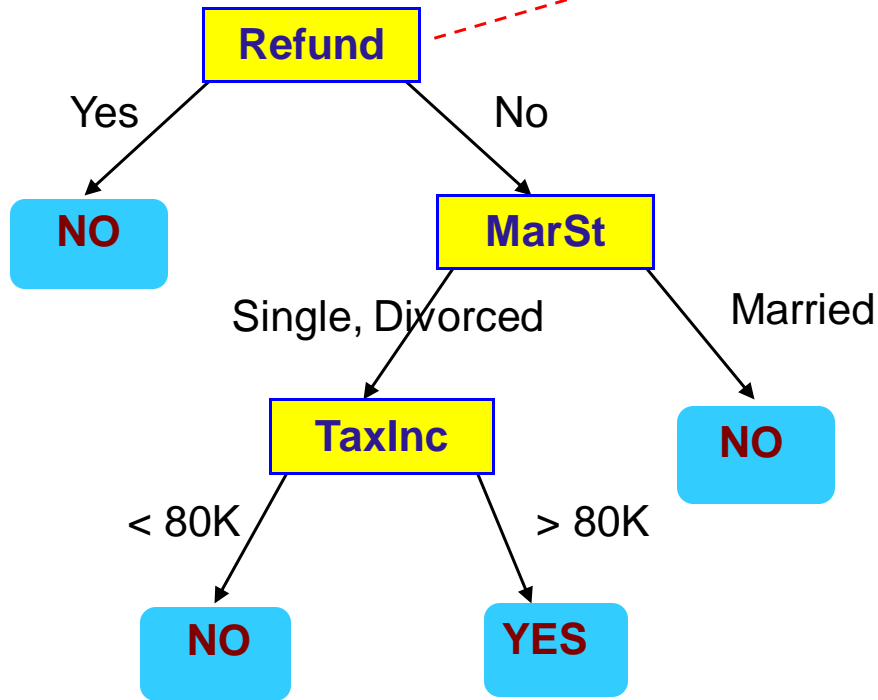
| Refund | Marital Status | Taxable Income | Cheat |
|--------|----------------|----------------|-------|
| No     | Married        | 80K            | ?     |



# Apply Model to Test Data

## Test Data

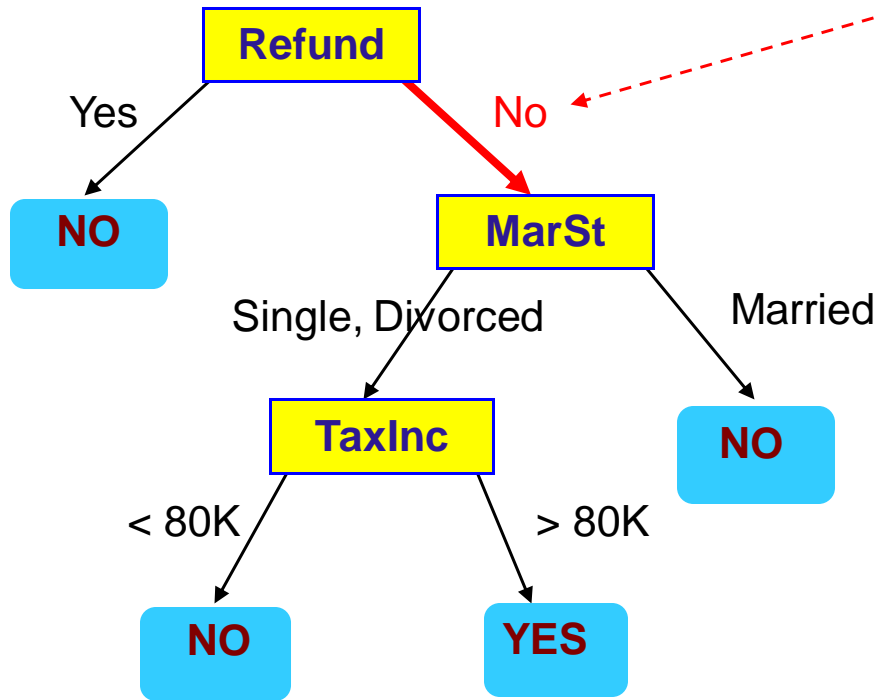
| Refund | Marital Status | Taxable Income | Cheat |
|--------|----------------|----------------|-------|
| No     | Married        | 80K            | ?     |



# Apply Model to Test Data

## Test Data

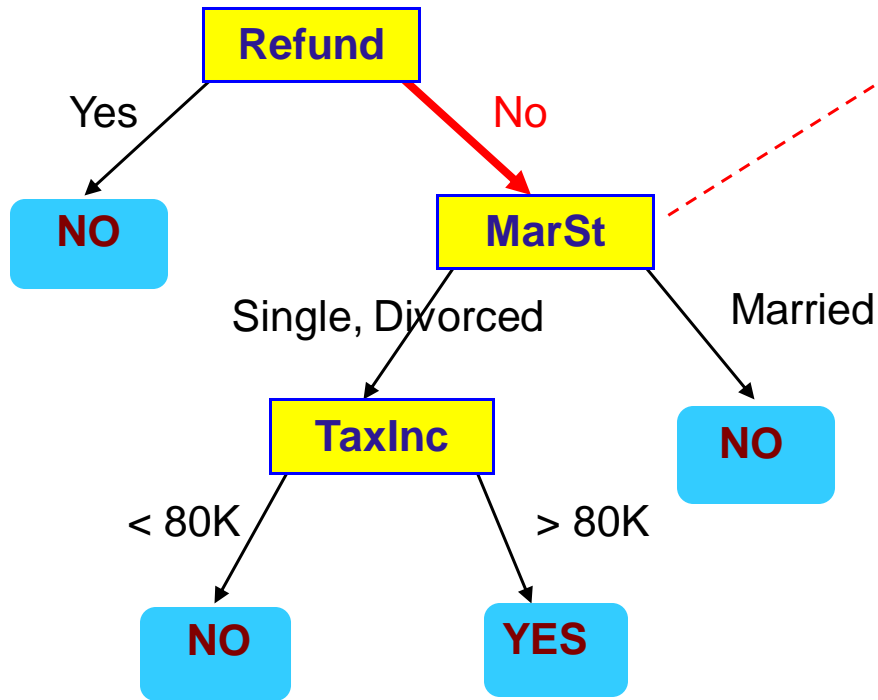
| Refund | Marital Status | Taxable Income | Cheat |
|--------|----------------|----------------|-------|
| No     | Married        | 80K            | ?     |



# Apply Model to Test Data

## Test Data

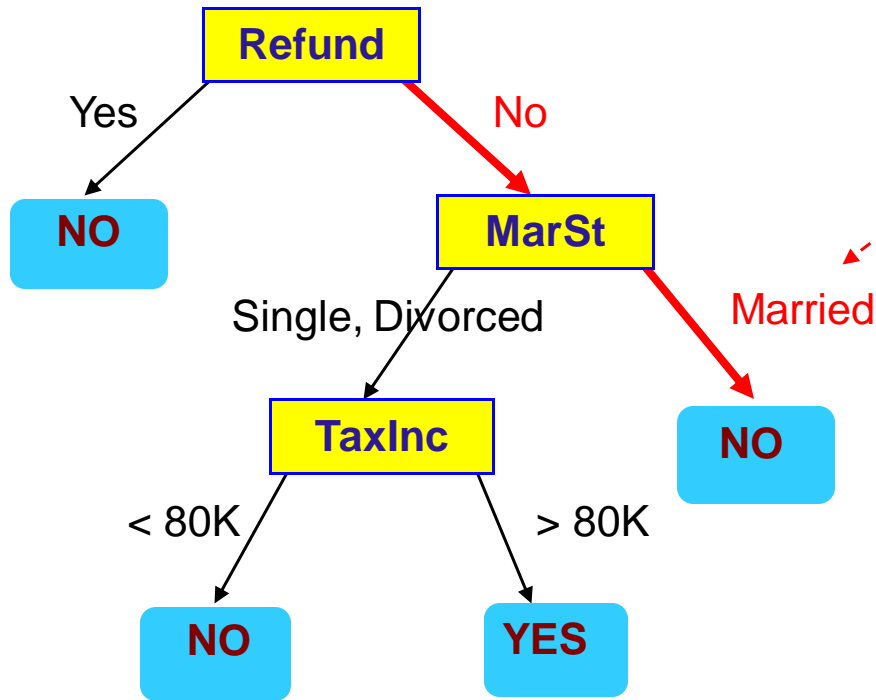
| Refund | Marital Status | Taxable Income | Cheat |
|--------|----------------|----------------|-------|
| No     | Married        | 80K            | ?     |



# Apply Model to Test Data

## Test Data

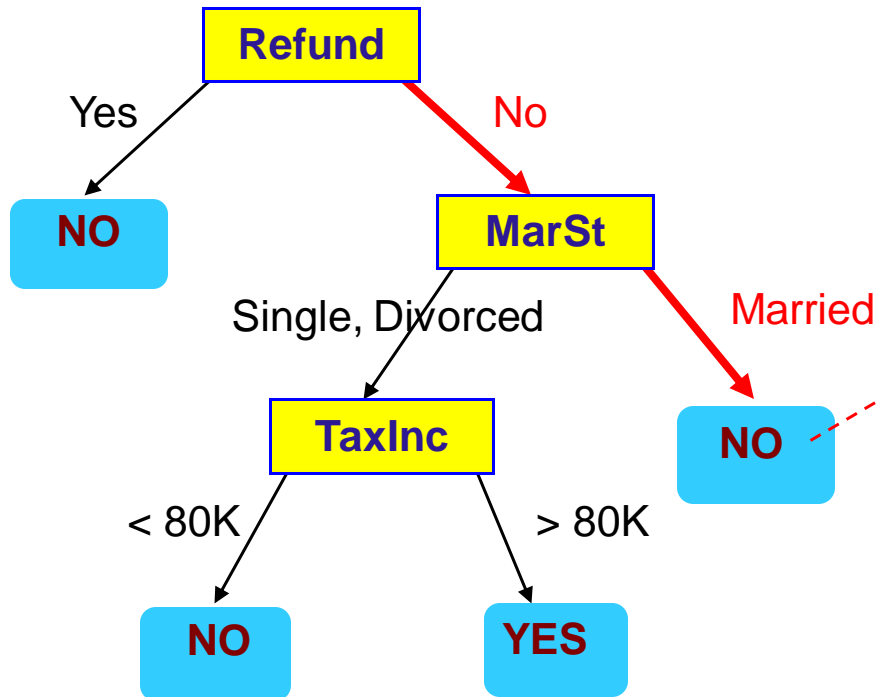
| Refund | Marital Status | Taxable Income | Cheat |
|--------|----------------|----------------|-------|
| No     | Married        | 80K            | ?     |



# Apply Model to Test Data

## Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|--------|----------------|----------------|-------|
| No     | Married        | 80K            | ?     |



Assign Cheat to "No"

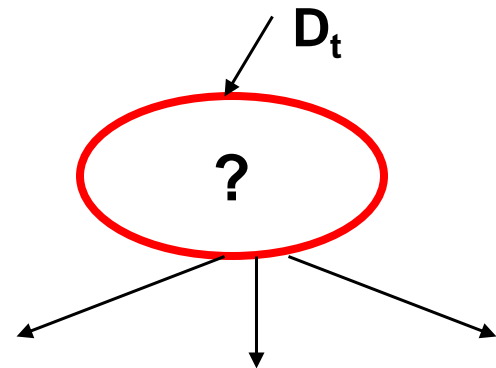
# Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (*one of the earliest*)
  - CART (*Classification and Regression Tree*)
  - ID3, C4.5
  - SLIQ (*Fast scalable algorithm for large application*)
    - ◆ Can handle both numeric and categorical attributes
  - SPRINT (*scalable parallel classifier for datamining*)

# General Structure of Hunt's Algorithm

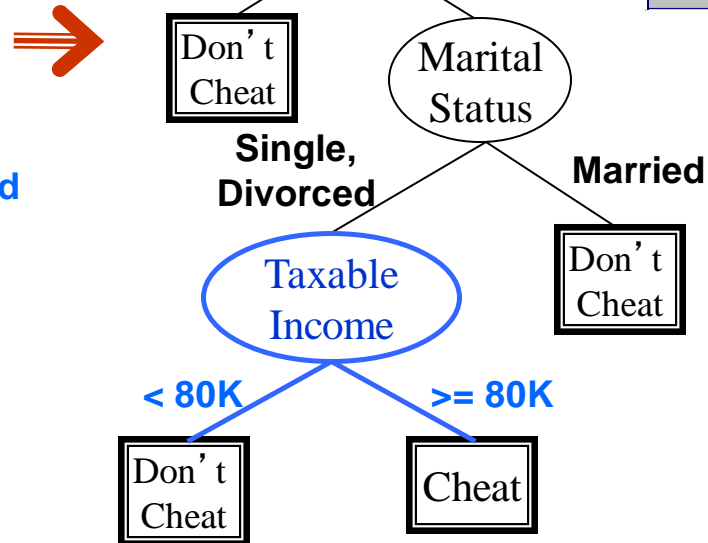
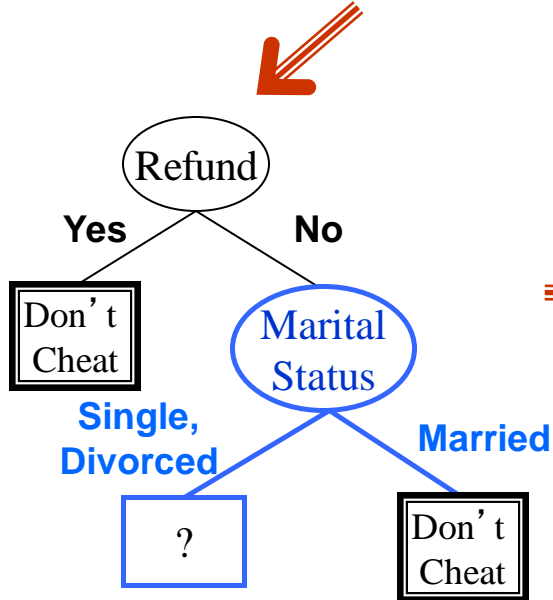
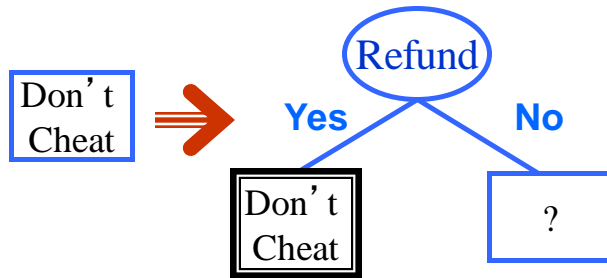
- Let  $D_t$  be the set of training records that reach a node  $t$
- General Procedure:
  - If  $D_t$  contains records that belong the same class  $y_t$ , then  $t$  is a leaf node labeled as  $y_t$
  - If  $D_t$  is an empty set, then  $t$  is a leaf node labeled by the default class,  $y_d$
  - If  $D_t$  contains records that belong to more than one class, use an attribute test 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   |





# Limitations of Hunt's Algorithm

- Too stringent for use in most practical situations
  - Works only if every combination of attribute values is present in the training data, and
  - Each combination has unique class label
- Additional conditions are needed
  - Empty child node
    - ◆ None of the training records has the combination of attribute values associated with such nodes (i.e. no records associated with this node)
    - ◆ Node is a leaf node with same class label as the majority class labels of its parent node (*solution*)
  - All the records have identical attribute values (not possible to split further)
    - ◆ Class label will be same as that of the majority class of the records associated with the node

# 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

# Tree Induction

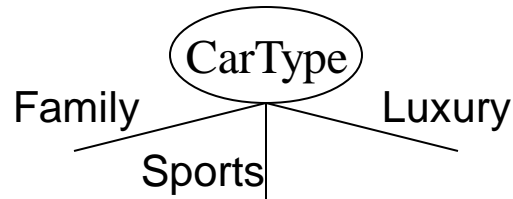
- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - ◆ How to specify the attribute test condition?
    - ◆ How to determine the best split?
  - Determine when to stop splitting

# How to Specify Test Condition?

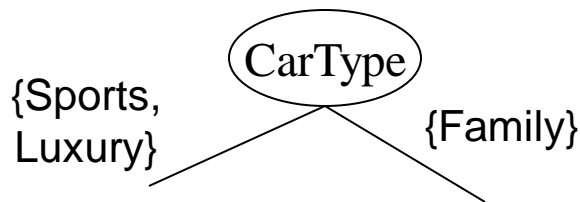
- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

# Splitting Based on Nominal Attributes

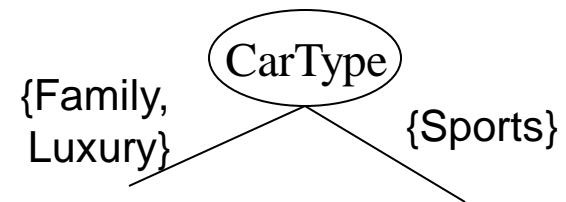
- **Multi-way split:** Use as many partitions as distinct values



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

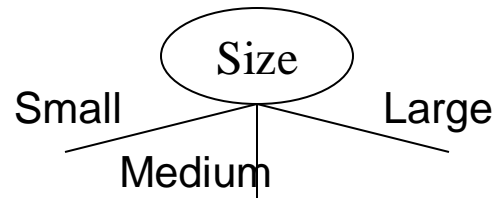


OR

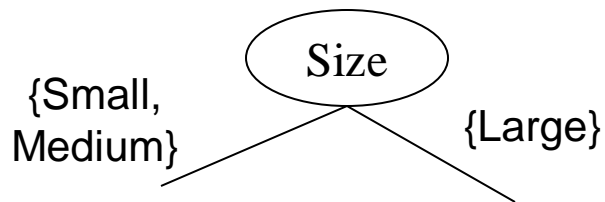


# Splitting Based on Ordinal Attributes

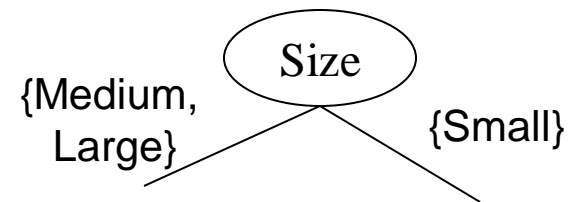
- **Multi-way split:** Use as many partitions as distinct values.



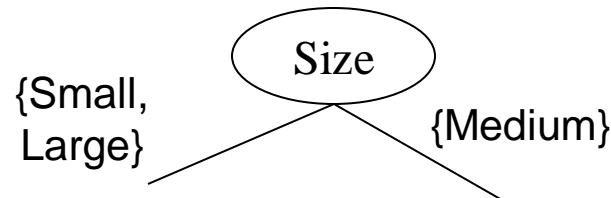
- **Binary split:** Divides values into two subsets.  
Need to find optimal partitioning.



OR



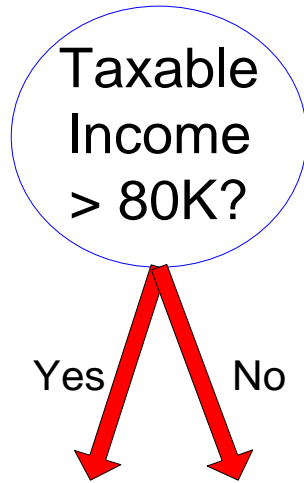
- What about this split?



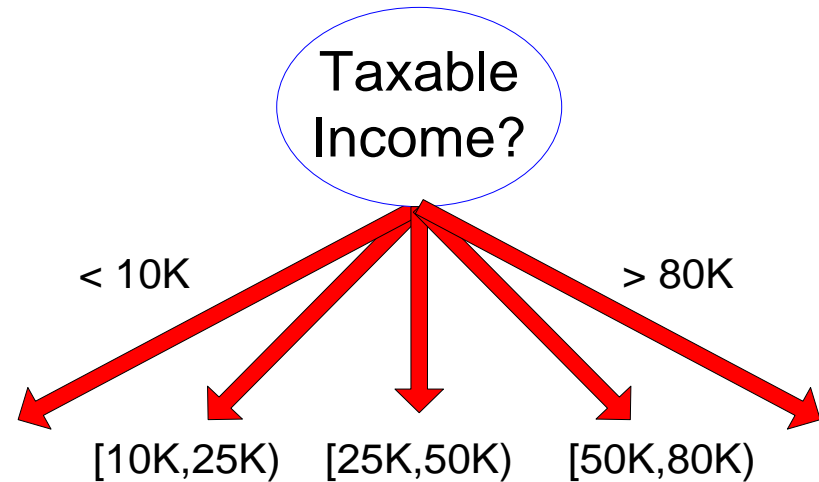
# Splitting Based on Continuous Attributes

- Different ways of handling
  - **Discretization** to form an ordinal categorical attribute
    - ◆ Static – discretize once at the beginning
    - ◆ Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
  - **Binary Decision**:  $(A < v)$  or  $(A \geq v)$ 
    - ◆ consider all possible splits and finds the best cut
    - ◆ can be more compute intensive

# Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

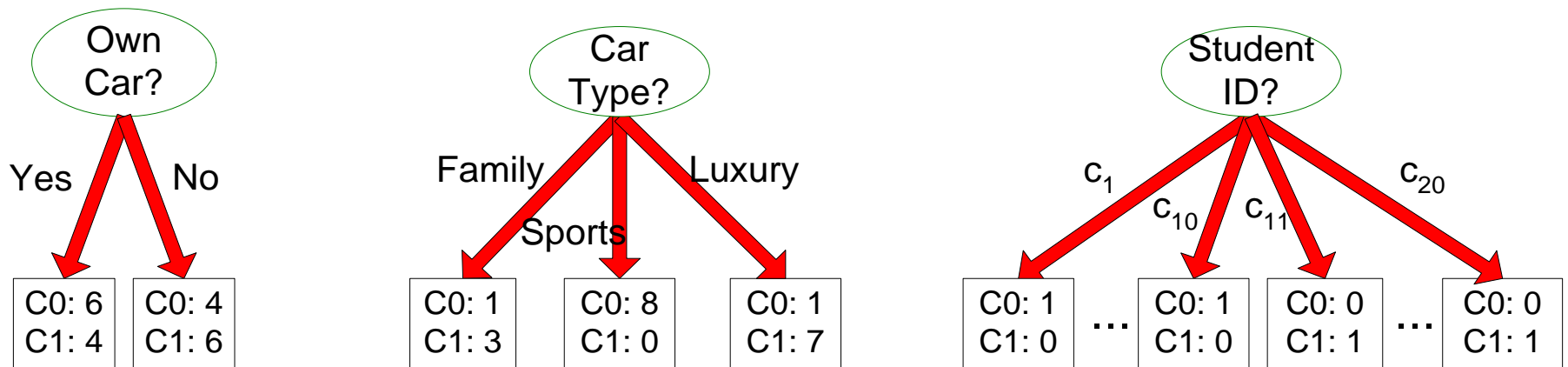


# Tree Induction

- 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**

# Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

# How to Find the Best Split?

Before Splitting:

|    |            |
|----|------------|
| C0 | <b>N00</b> |
| C1 | <b>N01</b> |

→ **M0**

A?

Yes

No

Node N1

Node N2

C0    **N10**

C1    **N11**

C0    **N20**

C1    **N21**



**M1**



**M2**

**M12**

B?

Yes

No

Node N3

Node N4

C0    **N30**

C1    **N31**

C0    **N40**

C1    **N41**



**M3**



**M4**

**M34**

**Gain = M0 – M12 vs M0 – M34**

# 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 (0.0) when all records belong to one class, implying most interesting 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,

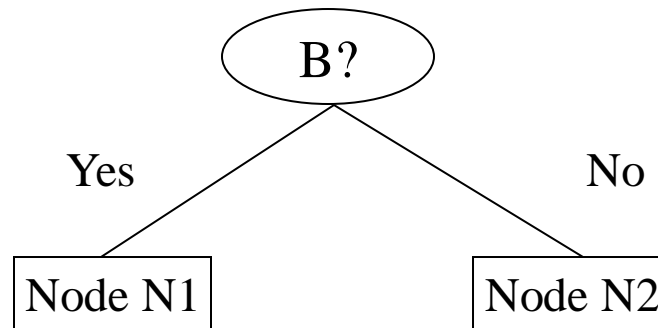
$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

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.***



**Gini(N1)**

$$= 1 - (5/7)^2 - (2/7)^2 \\ = 0.428$$

**Gini(N2)**

$$= 1 - (1/5)^2 - (4/5)^2 \\ = 0.528$$

|                   | N1 | N2 |
|-------------------|----|----|
| C1                | 5  | 1  |
| C2                | 2  | 4  |
| <b>Gini=0.469</b> |    |    |

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

**Gini(Children)**

$$= 7/12 * 0.428 + \\ 5/12 * 0.528 \\ = 0.469$$

# Categorical Attributes: **Computing Gini Index**

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

|      | CarType |        |        |
|------|---------|--------|--------|
|      | Family  | Sports | Luxury |
| C1   | 1       | 2      | 1      |
| C2   | 4       | 1      | 1      |
| Gini | 0.393   |        |        |

Two-way split  
(find best partition of values)

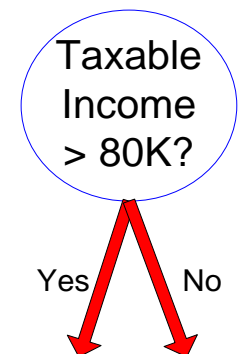
|      | CarType          |          |
|------|------------------|----------|
|      | {Sports, Luxury} | {Family} |
| C1   | 3                | 1        |
| C2   | 2                | 4        |
| Gini | 0.400            |          |

|      | CarType  |                  |
|------|----------|------------------|
|      | {Sports} | {Family, Luxury} |
| C1   | 2        | 2                |
| C2   | 1        | 5                |
| Gini | 0.419    |                  |

# Continuous Attributes: Computing Gini Index

- Use binary decisions based on one value
- Several choices for splitting the value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions,  $A < v$  and  $A \geq v$
- Simple method to choose the best  $v$ 
  - For each  $v$ , scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! (Repetition of works)

| <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   |



# Continuous Attributes: Computing Gini Index...

- For efficient computation: 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 least GINI index

|                 |   | Cheat | No             |   | No    |   | No    |   | Yes   |   | Yes   |   | Yes   |   | No           |   | No    |   | No    |   | No    |   |       |   |
|-----------------|---|-------|----------------|---|-------|---|-------|---|-------|---|-------|---|-------|---|--------------|---|-------|---|-------|---|-------|---|-------|---|
|                 |   |       | Taxable Income |   |       |   |       |   |       |   |       |   |       |   |              |   |       |   |       |   |       |   |       |   |
| Sorted Values   | → |       | 60             |   | 70    |   | 75    |   | 85    |   | 90    |   | 95    |   | 100          |   | 120   |   | 125   |   | 220   |   |       |   |
|                 |   |       | 55             |   | 65    |   | 72    |   | 80    |   | 87    |   | 92    |   | 97           |   | 110   |   | 122   |   | 172   |   | 230   |   |
| Split Positions | → |       | <=             | > | <=    | > | <=    | > | <=    | > | <=    | > | <=    | > | <=           | > | <=    | > | <=    | > | <=    | > | <=    | > |
|                 |   |       |                |   |       |   |       |   |       |   |       |   |       |   |              |   |       |   |       |   |       |   |       |   |
|                 |   | 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 |   | <u>0.300</u> |   | 0.343 |   | 0.375 |   | 0.400 |   | 0.420 |   |

# Alternative Splitting Criteria based on INFO

- 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 homogeneity 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
- ***Entropy based computations are similar to the GINI index computations***

# 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 INFO...

- 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**
- **Disadvantage**: Tends to prefer splits that result in *large number of partitions*, each being *small but pure*

# Splitting Based on INFO...

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO} \quad SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

$n_i$  is the 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



# 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> |

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

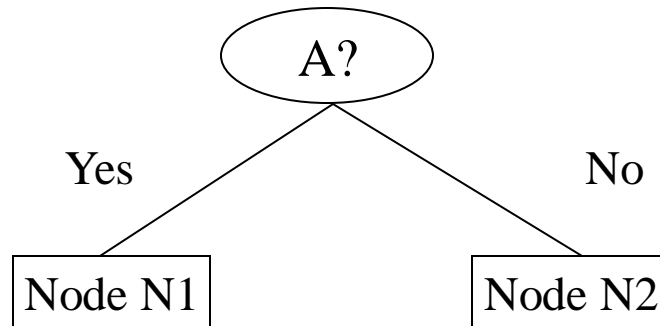
$$Error = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

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

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

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

# Misclassification Error vs Gini



|                    | Parent |
|--------------------|--------|
| C1                 | 7      |
| C2                 | 3      |
| <b>Gini = 0.42</b> |        |

$$\begin{aligned}\text{Gini}(N1) \\ &= 1 - (3/3)^2 - (0/3)^2 \\ &= 0\end{aligned}$$

|                   | N1 | N2 |
|-------------------|----|----|
| C1                | 3  | 4  |
| C2                | 0  | 3  |
| <b>Gini=0.361</b> |    |    |

$$\begin{aligned}\text{Gini}(N2) \\ &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.489\end{aligned}$$

$$\begin{aligned}\text{Gini(Children)} \\ &= 3/10 * 0 \\ &+ 7/10 * 0.489 \\ &= 0.342\end{aligned}$$

**Gini improves !!**

# 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*)

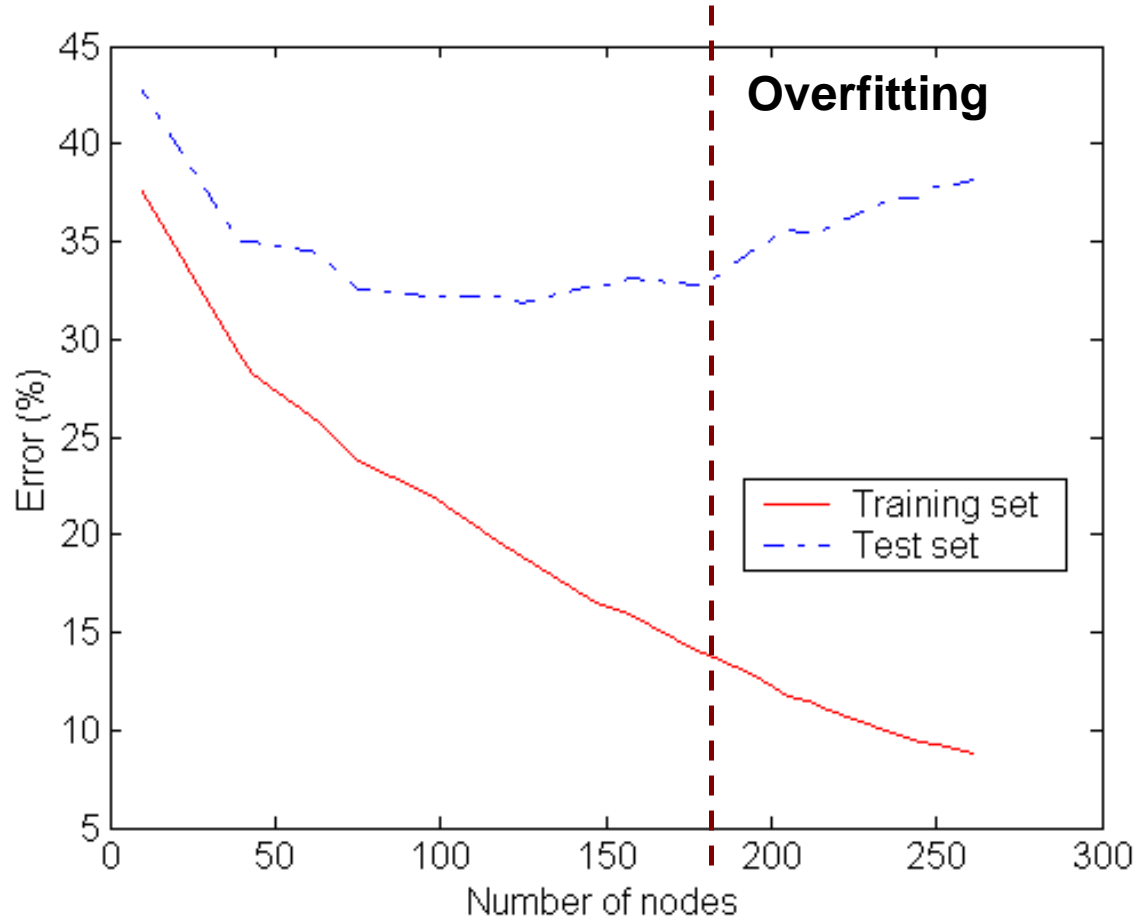
# Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

# Errors

- **Training errors** (resubstitution error): # misclassifications in training records
- **Generalization error**: expected error of the model on the previously unseen records
- **Good model**- must have *low training error as well as low generalization error*
- Model that fits training data well *can have a poorer generalization error than a model with a higher training error*

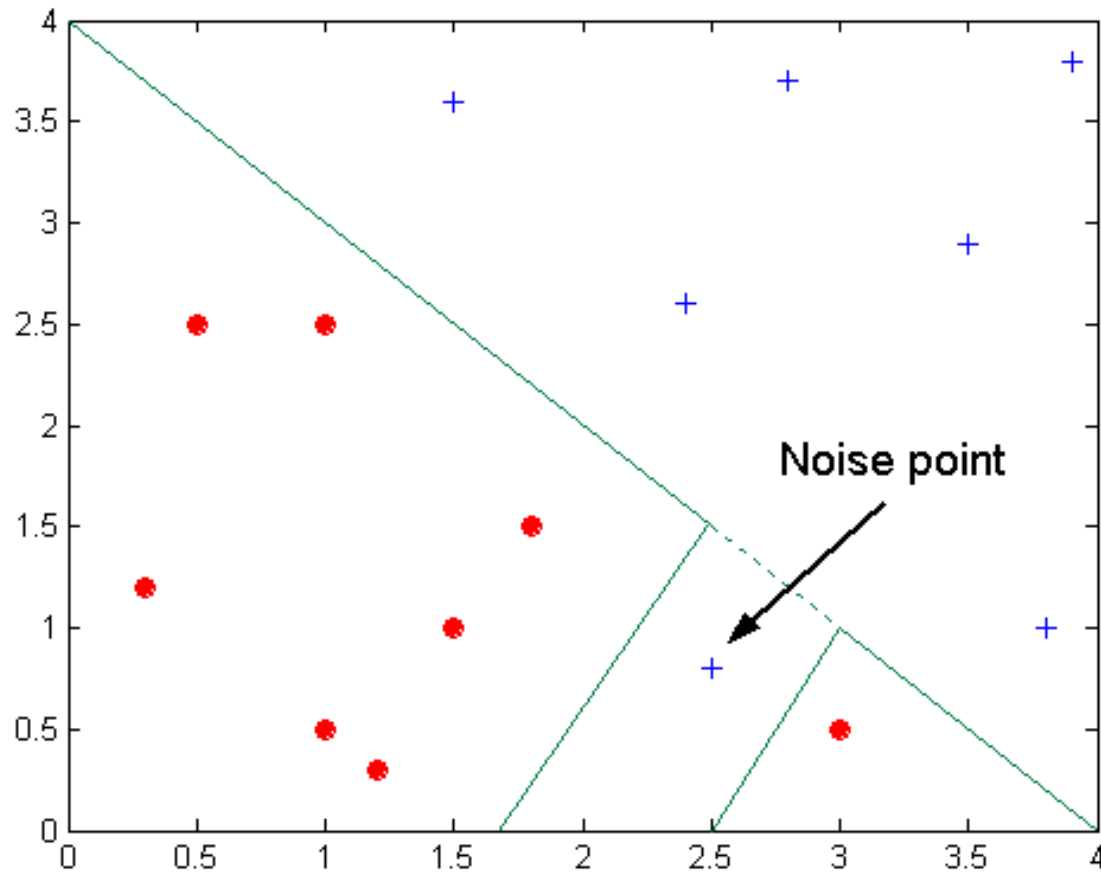
# Underfitting and Overfitting



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

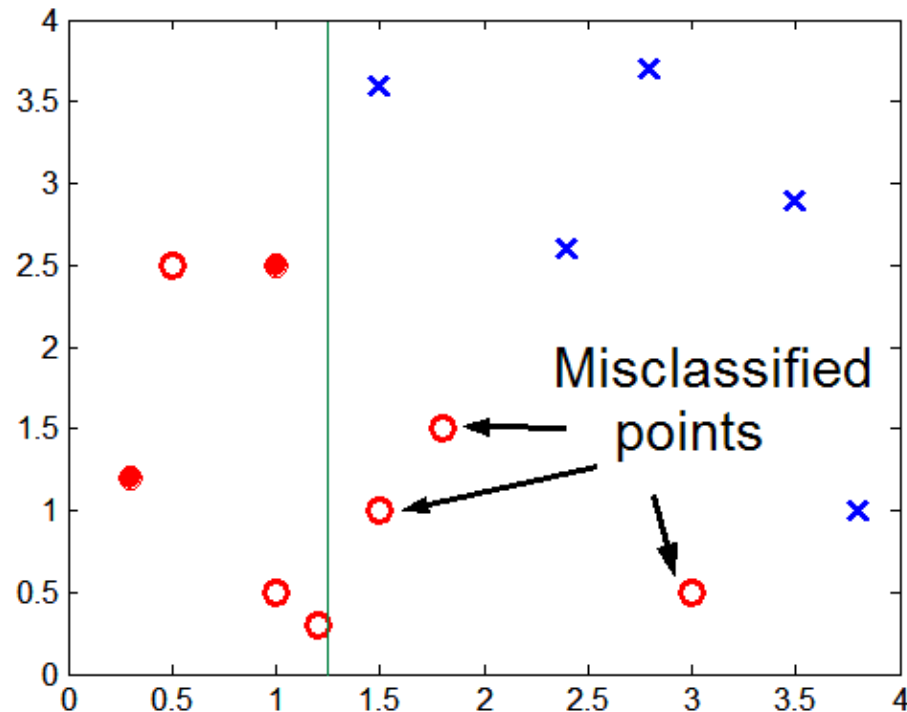


# Overfitting due to Noise



**Decision boundary is distorted by noise point**

# Overfitting due to Insufficient Examples



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

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

# Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Needs new ways for estimating errors

# Estimating Generalization Errors

- **Re-substitution errors:** error on training ( $\sum e(t)$ )
- **Generalization errors:** error on testing ( $\sum e'(t)$ )
- Methods for estimating generalization errors:
  - **Optimistic approach:**  $e'(t) = e(t)$
  - **Pessimistic approach:**
    - ◆ For each leaf node:  $e'(t) = (e(t) + 0.5)$
    - ◆ Total errors:  $e'(T) = e(T) + N \times 0.5$  (N: number of leaf nodes)
    - ◆ For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):  
Training error =  $10/1000 = 1\%$   
Generalization error =  $(10 + 30 \times 0.5)/1000 = 2.5\%$
  - **Reduced error pruning (REP):**
    - ◆ Uses validation data set to estimate generalization error

# Occam's Razor

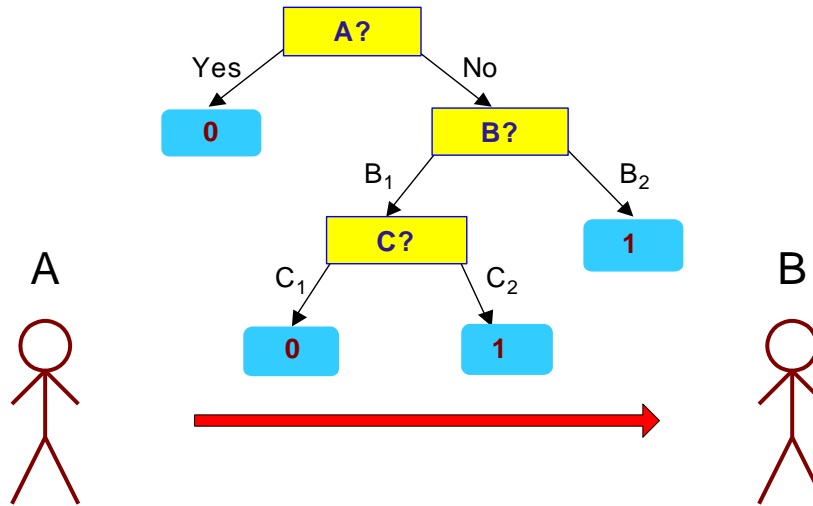
- Given two models of *similar generalization errors*, one should *prefer the simpler model over the more complex model*
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

# Two methods for model complexity

- Pessimistic error estimates (*already discussed*)
  - Training error + penalty term
- Minimum description length
  - An information theoretic approach

# Minimum Description Length (MDL)

| X              | y   |
|----------------|-----|
| X <sub>1</sub> | 1   |
| X <sub>2</sub> | 0   |
| X <sub>3</sub> | 0   |
| X <sub>4</sub> | 1   |
| ...            | ... |
| X <sub>n</sub> | 1   |



| X              | y   |
|----------------|-----|
| X <sub>1</sub> | ?   |
| X <sub>2</sub> | ?   |
| X <sub>3</sub> | ?   |
| X <sub>4</sub> | ?   |
| ...            | ... |
| X <sub>n</sub> | ?   |

- B obtains classification of each record by requesting A to transmit
  - Cost is  $O(n)$ :  $n$ -bits required to transmit message, i.e. number of records
- Alternatively, A builds a classification model that summarizes the relationships between  $x$  and  $y$ 
  - For 100% accurate model, cost of transmission is equivalent to the cost of encoding the model

# Minimum Description Length (MDL)

- $\text{Cost}(\text{Model}, \text{Data}) = \text{Cost}(\text{Data}|\text{Model}) + \text{Cost}(\text{Model})$ 
  - Cost is the number of bits needed for encoding
  - Search for the least costly model
- $\text{Cost}(\text{Data}|\text{Model})$  encodes the misclassification errors
- $\text{Cost}(\text{Model})$  uses node encoding (*number of children*) plus splitting condition encoding



# 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 (e.g., using  $\chi^2$  test)
  - ◆ 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* or by the frequently occurring branch
- Class label of leaf node is determined from *majority class* of instances in the sub-tree
- Can use MDL for post-pruning

# Example of Post-Pruning

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

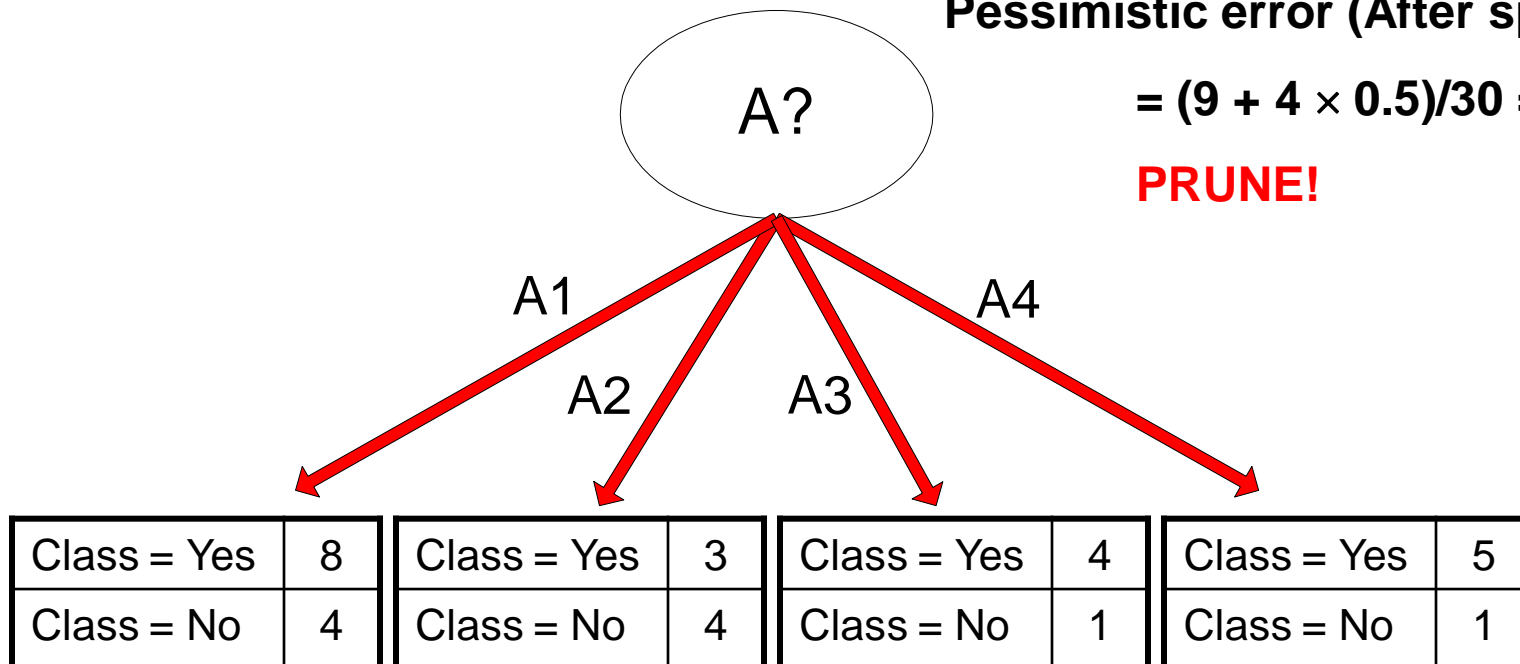
Training Error (Before splitting) = 10/30

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

Training Error (After splitting) = 9/30

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

**PRUNE!**



# Examples of Post-pruning

- Optimistic error?

Don't prune for both cases

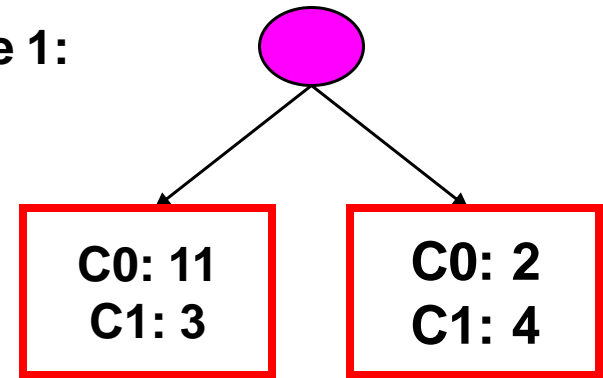
- Pessimistic error?

Don't prune case 1, prune case 2

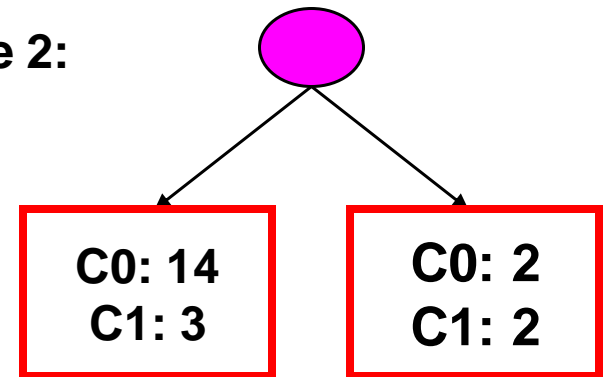
- Reduced error pruning?

Depends on validation set

Case 1:



Case 2:



# 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<br>= Yes | Class<br>= No |
|------------|----------------|---------------|
| Refund=Yes | 0              | 3             |
| Refund=No  | 2              | 4             |
| Refund=?   | 1              | 0             |

**Split on Refund:**

Entropy(Refund=Yes) = 0

Entropy(Refund=No)

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

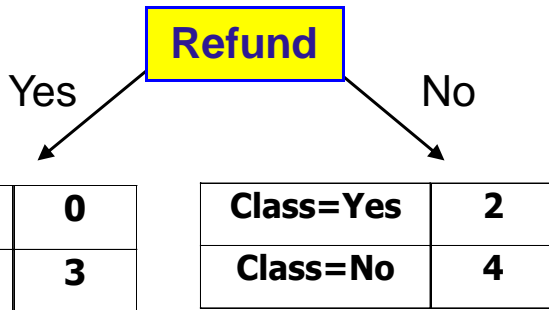
Entropy(Children)

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

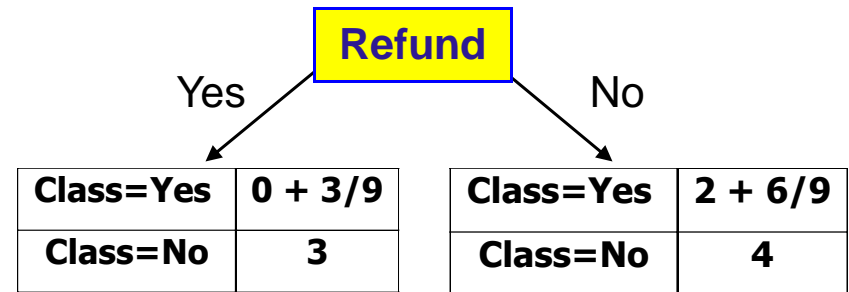
$$\text{Gain} = 0.9 \times (0.8813 - 0.551) = 0.3303$$

# 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         | ?      | Married        | 90K            | Yes   |



Probability that Refund=Yes is 3/9

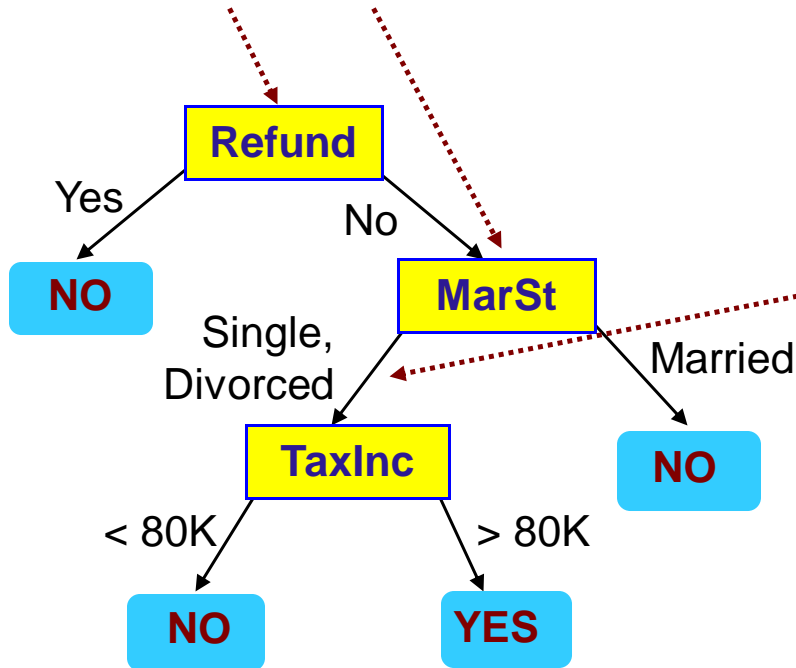
Probability that Refund=No is 6/9

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

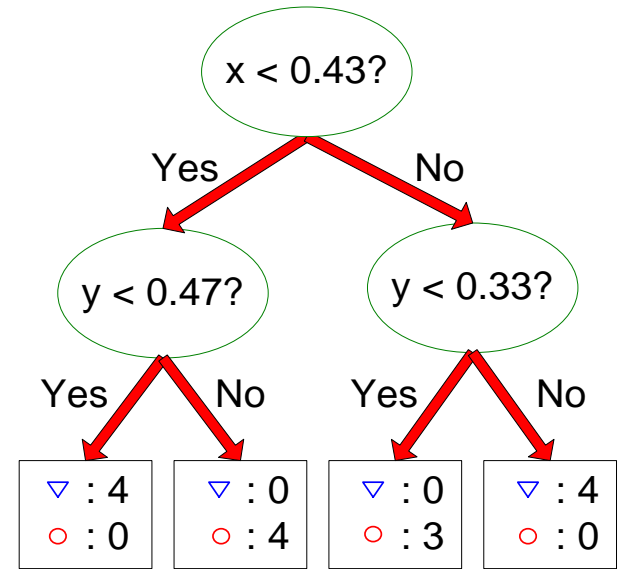
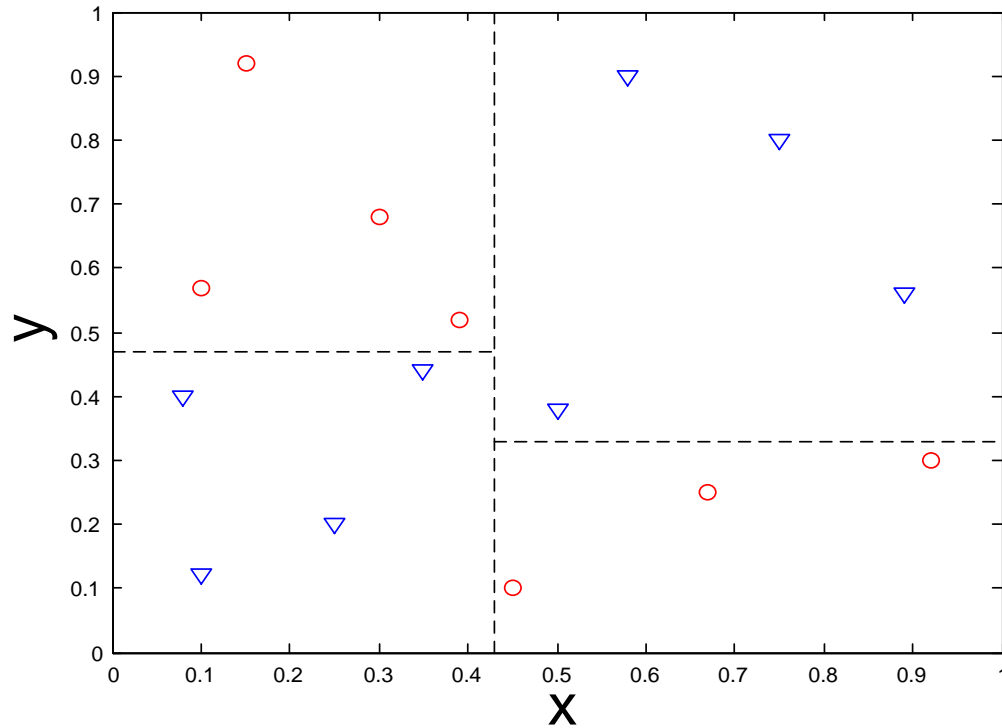
# Search Strategy

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

# Expressiveness

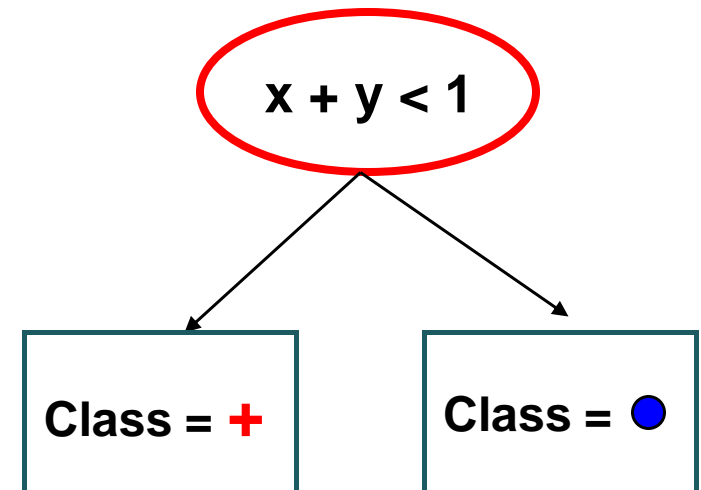
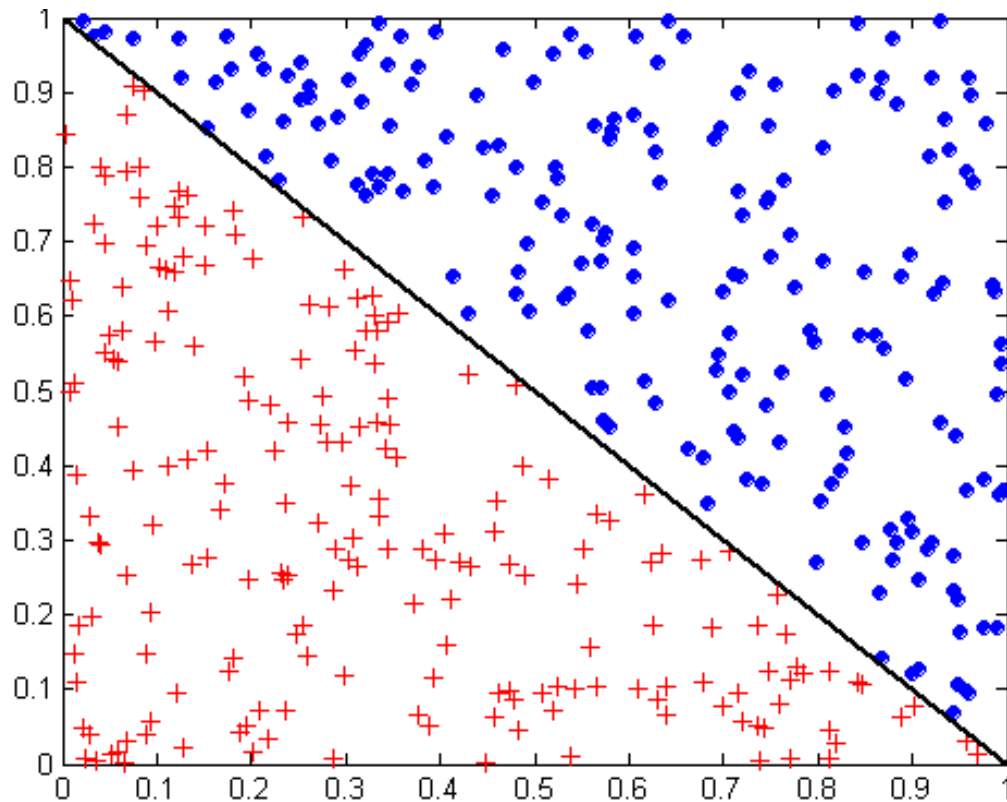
- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - ◆ Example: parity function:
      - Class = 1 if there is an even number of boolean attributes with truth value = True
      - Class = 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



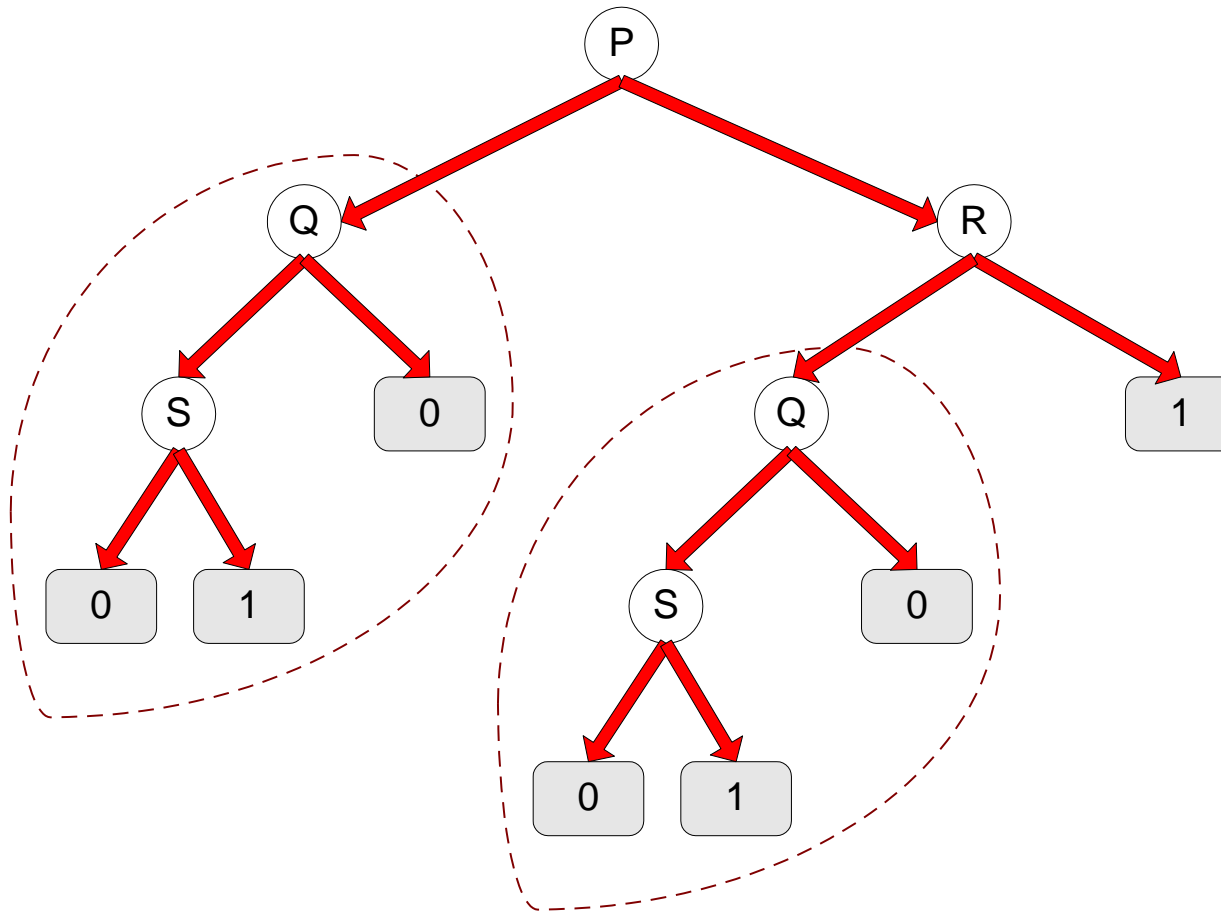
- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

# Oblique Decision Trees



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

# Tree Replication



- Same subtree appears in multiple branches

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?



# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?

# 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:**

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

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

# Metrics for Performance Evaluation...

| ACTUAL CLASS | PREDICTED CLASS |           |
|--------------|-----------------|-----------|
|              |                 |           |
|              | Class=Yes       | Class=No  |
| Class=Yes    | a<br>(TP)       | b<br>(FN) |
|              | c<br>(FP)       | d<br>(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 Matrix

|  | PREDICTED CLASS |                            |                           |
|--|-----------------|----------------------------|---------------------------|
|  | $C(i j)$        | Class=Yes                  | Class=No                  |
|  | Class=Yes       | $C(\text{Yes} \text{Yes})$ | $C(\text{No} \text{Yes})$ |
|  | Class=No        | $C(\text{Yes} \text{No})$  | $C(\text{No} \text{No})$  |

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

# Computing Cost of Classification

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

| Model $M_1$  | PREDICTED CLASS |     |     |
|--------------|-----------------|-----|-----|
| ACTUAL CLASS |                 | +   | -   |
|              | +               | 150 | 40  |
|              | -               | 60  | 250 |

Accuracy = 80%

Cost = 3910

| Model $M_2$  | PREDICTED CLASS |     |     |
|--------------|-----------------|-----|-----|
| ACTUAL CLASS |                 | +   | -   |
|              | +               | 250 | 45  |
|              | -               | 5   | 200 |

Accuracy = 90%

Cost = 4255

# Cost vs Accuracy

| Count        | PREDICTED CLASS |           |          |
|--------------|-----------------|-----------|----------|
| ACTUAL CLASS |                 | Class=Yes | Class=No |
|              | Class=Yes       | a         | b        |
|              | Class=No        | c         | d        |

Accuracy is proportional to cost if

1.  $C(\text{Yes}|\text{No})=C(\text{No}|\text{Yes}) = q$
2.  $C(\text{Yes}|\text{Yes})=C(\text{No}|\text{No}) = p$

$$N = a + b + c + d$$

$$\text{Accuracy} = (a + d)/N$$

| Cost         | PREDICTED CLASS |           |          |
|--------------|-----------------|-----------|----------|
| ACTUAL CLASS |                 | Class=Yes | Class=No |
|              | Class=Yes       | p         | q        |
|              | Class=No        | q         | p        |

$$\text{Cost} = p(a + d) + q(b + c)$$

$$= p(a + d) + q(N - a - d)$$

$$= qN - (q - p)(a + d)$$

$$= N[q - (q - p) \times \text{Accuracy}]$$

# Cost-Sensitive Measures

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

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

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

- Precision is biased towards  $C(\text{Yes}|\text{Yes})$  &  $C(\text{Yes}|\text{No})$
- Recall is biased towards  $C(\text{Yes}|\text{Yes})$  &  $C(\text{No}|\text{Yes})$
- F-measure is biased towards all except  $C(\text{No}|\text{No})$

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$



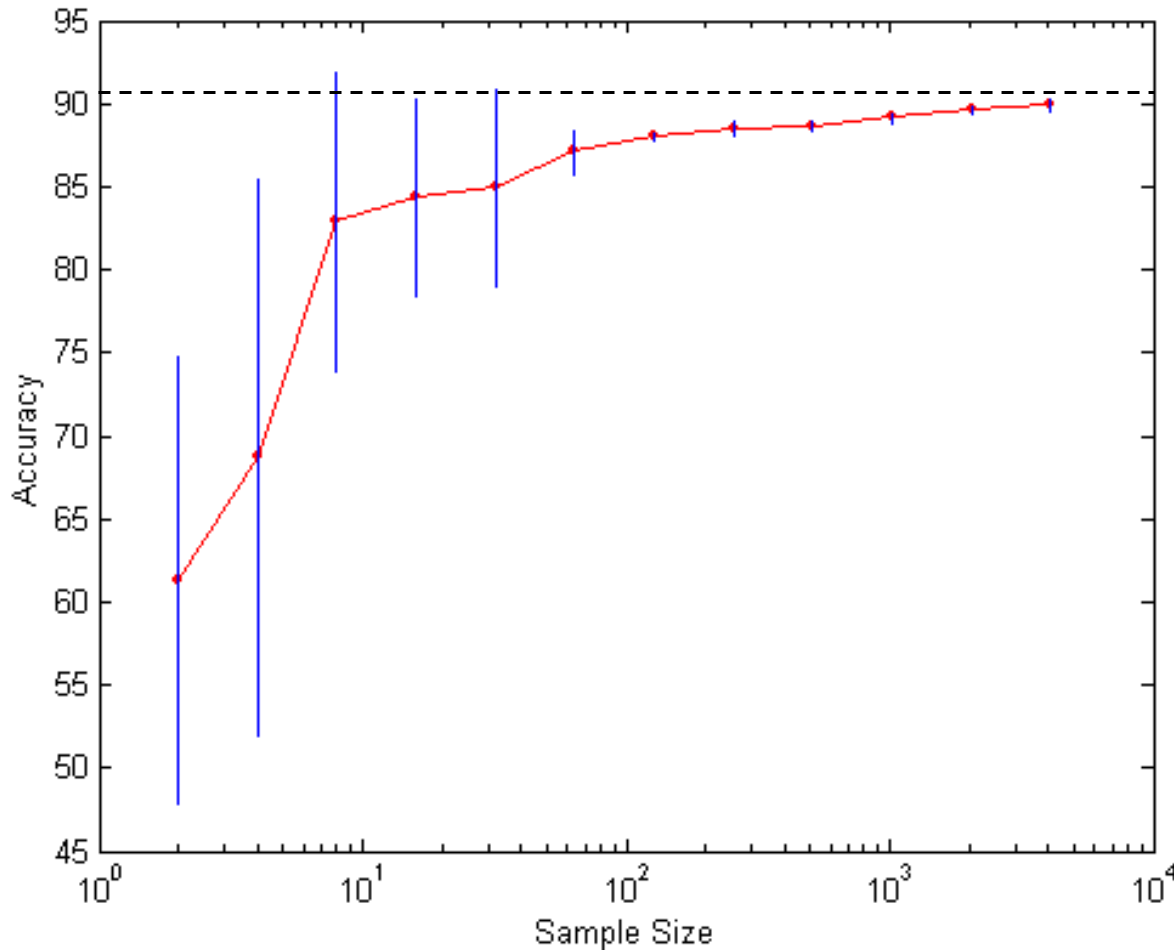
# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?

# Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?
- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

# Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:

- **Arithmetic sampling (Langley, et al)**
- **Geometric sampling (Provost et al)**

Effect of small sample size:

- **Bias in the estimate**
- **Variance of**

The **bias** of a classifier is the expected error of the classifier due to the fact that the classifier is not perfect

The **variance** of a classifier is the expected error due to the particular training set used

# Methods of Estimation

- Holdout

- Reserve  $2/3$  for training and  $1/3$  for testing
- Less examples are available for training
- Model depends heavily on the composition of training and test sets
  - ◆ Too small training set -larger variance
  - ◆ Too large training set- estimated accuracy not very reliable

- Random subsampling

- Repeated holdout
- Some instances may be used more than others for training

- Cross validation

- Partition data into  $k$  disjoint subsets
- $k$ -fold: train on  $k-1$  partitions, test on the remaining one
- Leave-one-out:  $k=n$  (one record in each subset)

- Stratified sampling

- Oversampling vs. Undersampling

- Bootstrap

- Sampling with replacement