# DATA MINING LECTURE 9

**Classification**

Basic Concepts

Decision Trees

# What is classification?

Tax-return data

| 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 |

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

# What is classification?

- **Classification** is the task of *learning a target function* **f** that maps attribute set **x** to one of the predefined class labels **y**

*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 |

One of the attributes is the class attribute
In this case: Cheat
Two class labels: Yes, No

# What is classification?



Input
Attribute set
(**x**)

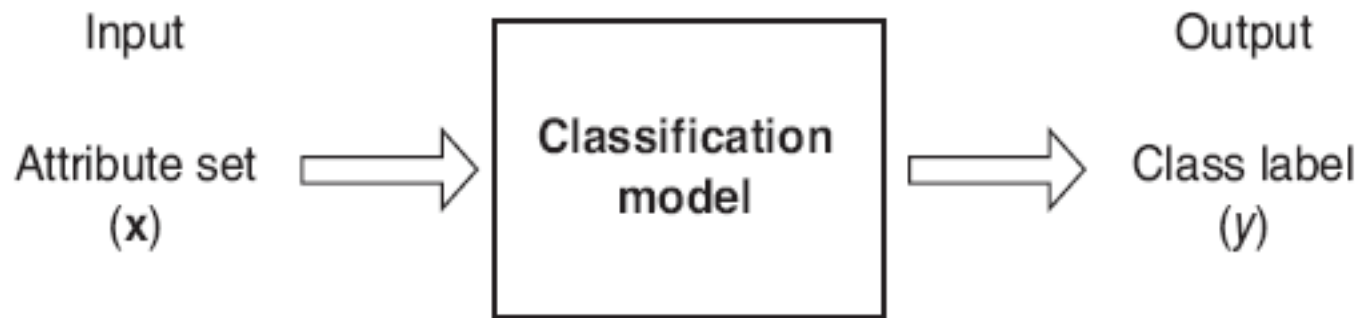**Classification model**

Output
Class label
(*y*)

**Figure 4.2.** Classification as the task of mapping an input attribute set $x$ into its class label $y$.

# Why classification?

- The target function **f** is known as a ***classification model***

- **Descriptive modeling:** ***Explanatory tool*** to distinguish between objects of different classes (e.g., description of who can pay back his loan)

- **Predictive modeling:** Predict a class of a previously ***unseen*** record

# 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

- Identifying spam email, spam web pages, adult content

- Categorizing web users, and web queries

# General approach to classification

- ***Training set*** consists of records with ***known class labels***

- Training set is used to ***build a classification model***

- The classification model is applied to new records with **unknown class labels**
  - A ***test set*** of ***previously unseen*** data records is used to evaluate the quality of the model.
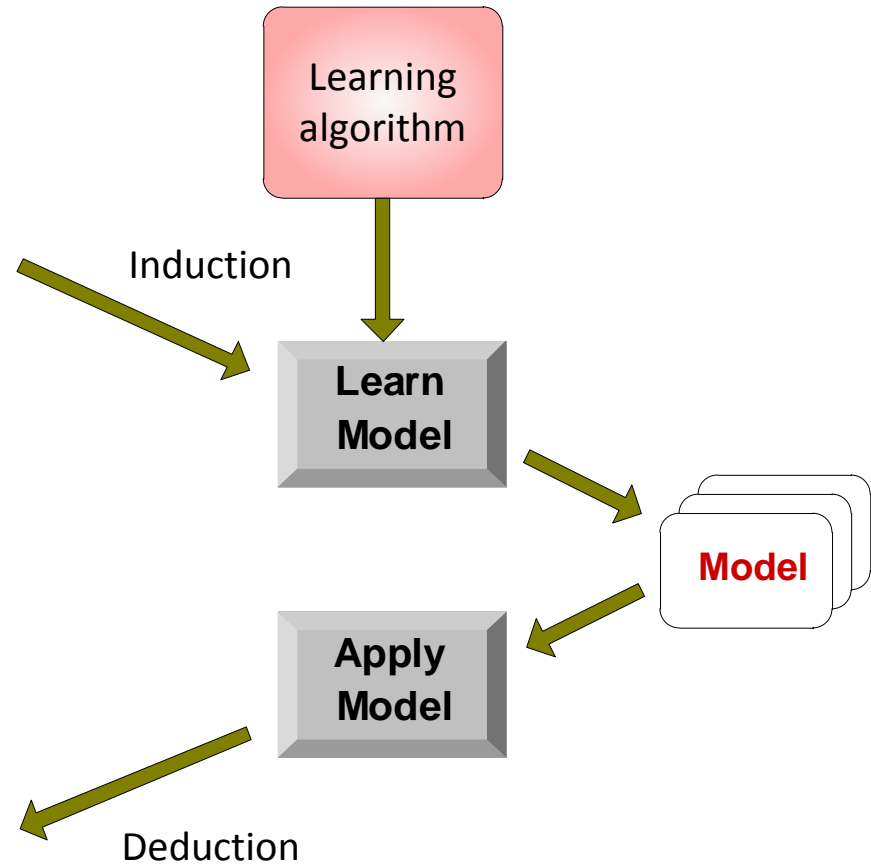
# Illustrating Classification Task

| 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

Learning algorithm

Induction

Learn Model

Model

Apply Model

| 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

Deduction

# Evaluation of classification models

- Counts of test records that are correctly (or incorrectly) predicted by the classification model
- **Confusion matrix**

|  | | Predicted Class | |
|---|---|---|---|
| | | **Class = 1** | **Class = 0** |
| **Actual Class** | **Class = 1** | $f_{11}$ | $f_{10}$ |
| | **Class = 0** | $f_{01}$ | $f_{00}$ |

$$\text{Accuracy} = \frac{\#\,\text{correct predictions}}{\text{total}\,\#\,\text{of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\#\,\text{wrong predictions}}{\text{total}\,\#\,\text{of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

# Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

# Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
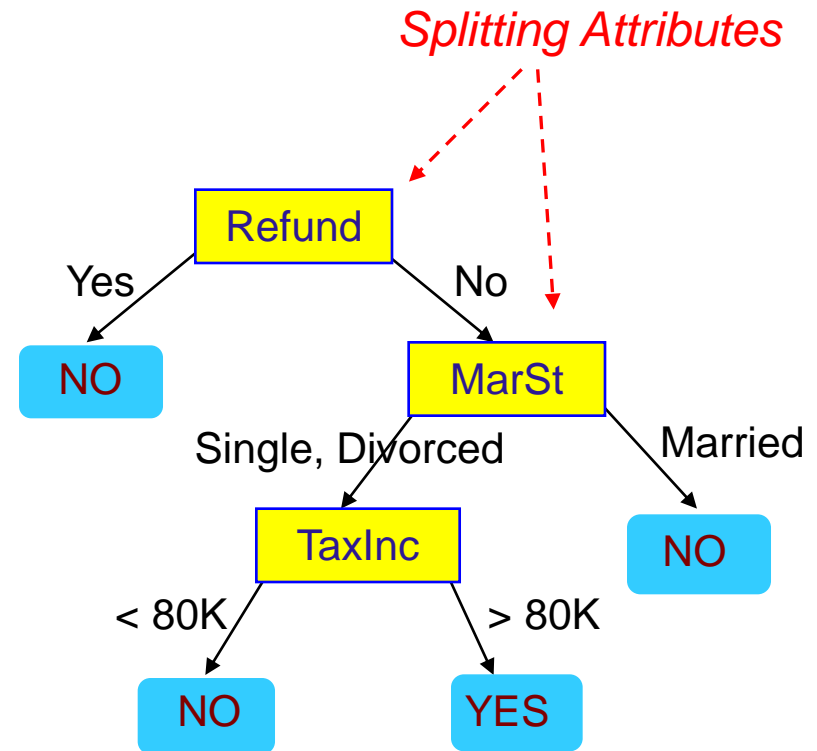- Support Vector Machines

# Decision Trees

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution

# 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

*Splitting Attributes*

Refund

Yes / No

NO    MarSt

Single, Divorced / Married

TaxInc    NO

< 80K / > 80K

NO    YES

Model:  Decision Tree

# Another Example of 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   |

MarSt

Married

Single, Divorced

NO

Refund

Yes

No

NO

TaxInc

< 80K

> 80K

NO

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

Tree Induction algorithm

Induction

**Learn Model**

**Model**

Decision Tree

| 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**

Deduction

# Apply Model to Test Data

Test Data

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

Start from the root of tree.

# Apply Model to Test Data

Test Data

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

Refund

Yes | No

NO

MarSt

Single, Divorced | Married

TaxInc

NO

< 80K | > 80K

NO | YES

# Apply Model to Test Data

Test Data

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

Refund

Yes → NO

No → MarSt

Single, Divorced → TaxInc

Married → NO

< 80K → NO

> 80K → YES

# Apply Model to Test Data
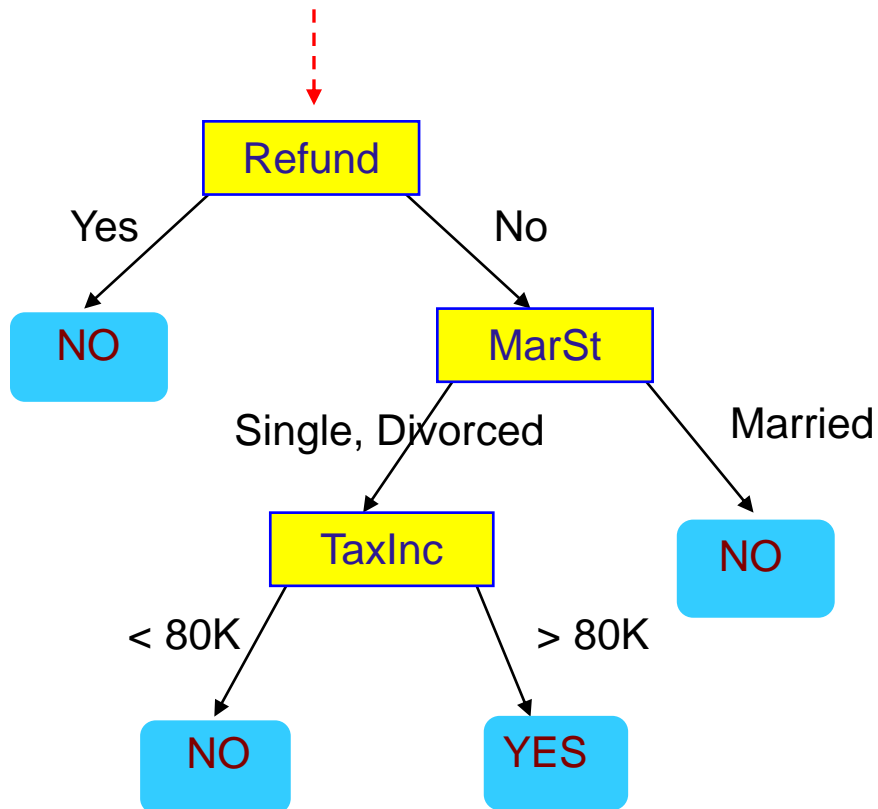
Test Data

| 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 | ? |

Refund

Yes / No

NO

MarSt

Single, Divorced / Married

TaxInc

NO

< 80K / > 80K

NO

YES

# Apply Model to Test Data

Test Data

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

Refund

Yes → NO

No → MarSt

Single, Divorced → TaxInc

< 80K → NO

> 80K → YES
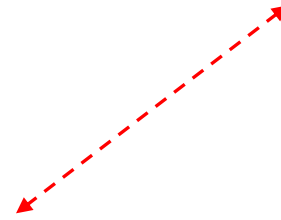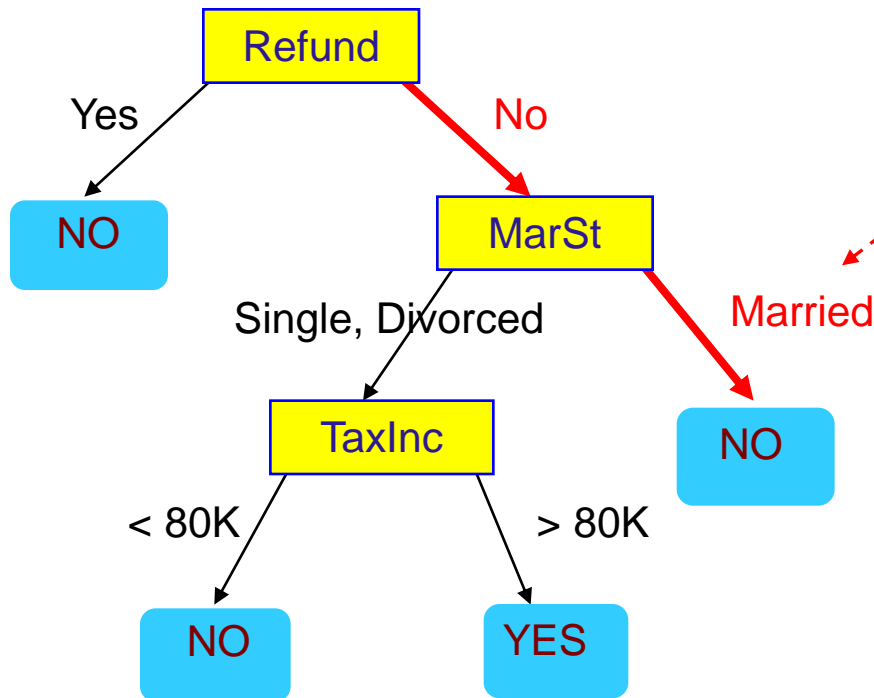
Married → NO
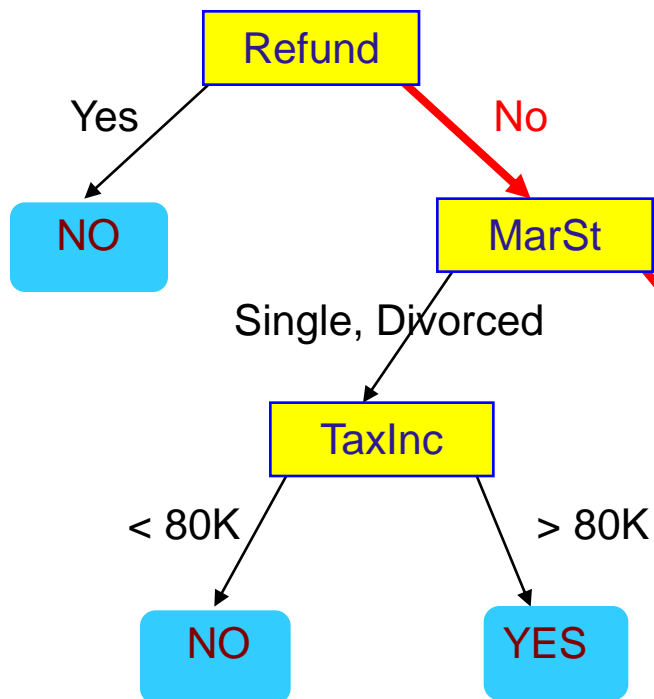
Assign Cheat to "No"

# 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

Induction

Tree Induction algorithm

Learn Model

Model

Decision Tree

Apply Model

| 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

Deduction

# 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 $D_t$ 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.

| 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 |

$D_t$

?

# 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 |

Don't Cheat ⇒

Refund
Yes    No

Don't Cheat

Don't Cheat    Don't Cheat

Refund
Yes    No

Don't Cheat

Marital Status

Single, Divorced    Married

Cheat    Don't Cheat

⇒

Refund
Yes    No

Don't Cheat

Marital Status

Single, Divorced    Married

Don't Cheat

Taxable Income

< 80K    >= 80K

Don't Cheat    Cheat

# Tree Induction

- Finding the best decision tree is NP-hard


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

# Constructing decision-trees (pseudocode)

**GenDecTree**(Sample **S**, Features **F**)

1. **If stopping_condition(S,F)** = true **then**
   a. **leaf = createNode()**
   b. **leaf.label= Classify(S)**
   c. **return leaf**
2. **root = createNode()**
3. **root.test_condition = findBestSplit(S,F)**
4. **V = {v| v a** possible outcome of **root.test_condition}**
5. **for** *each* value **v∈V**:
   a. **$S_v$: = {s | root.test_condition(s)** = **v** and **s ∈ S}**;
   b. **child = TreeGrowth($S_v$ ,F)** ;
   c. Add **child** as a descent of **root** and label the edge **(root→child) as v**
6. **return root**

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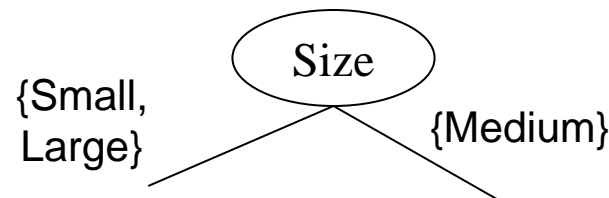
- **Multi-way split:** Use as many partitions as distinct values.
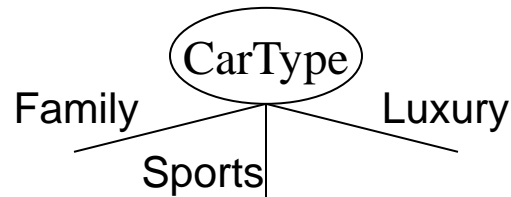


- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.
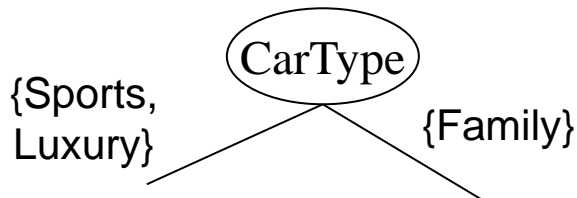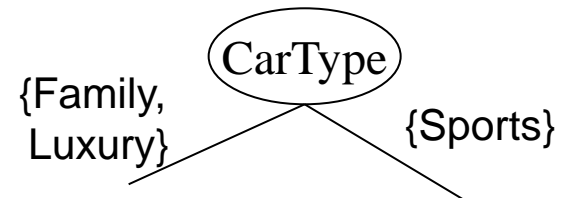
# Splitting Based on Ordinal Attributes

- **Multi-way split:** Use as many partitions as distinct values.
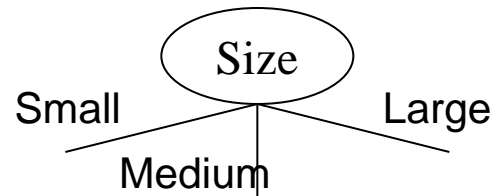
```
          ( Size )
   Small  /   |   \  Large
         /  Medium  \
```

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

```
          ( Size )                OR          ( Size )
{Small,  /       \                  {Medium, /       \
Medium} /         \ {Large}         Large}  /         \ {Small}
```
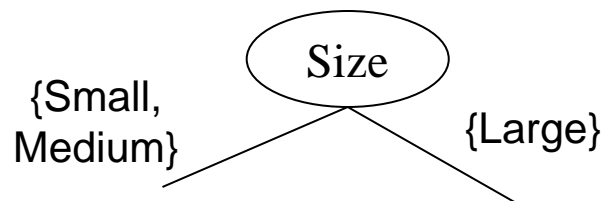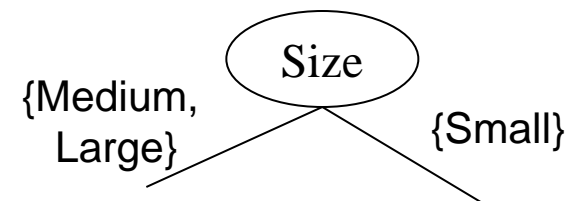
- What about this split?

```
                    ( Size )
        {Small,    /        \
        Large}    /          \ {Medium}
```

# 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
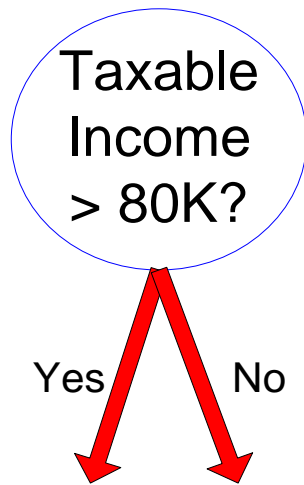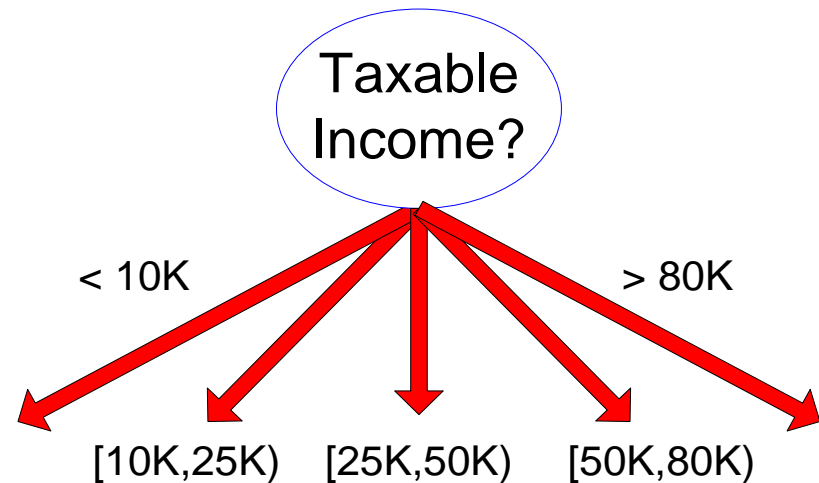


(i) Binary split                    (ii) Multi-way split

# 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
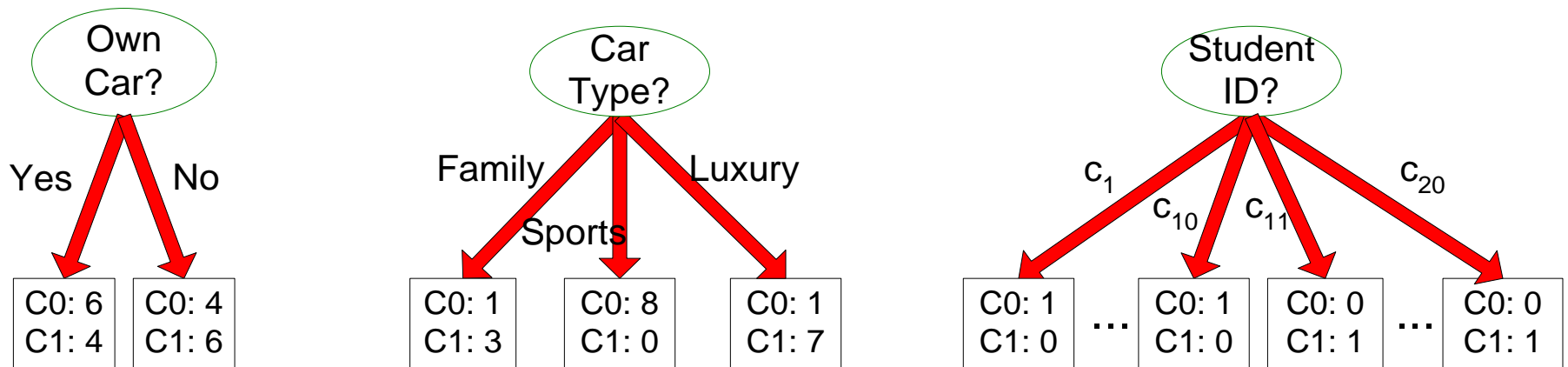


Which test condition is the best?

# 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

# Measuring Node Impurity

- **p(i|t)**: fraction of records associated with node **t** belonging to class **i**

$$\text{Entropy}(t) = -\sum_{i=1}^{c} p(i \mid t) \log p(i \mid t)$$

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} \left[ p(i \mid t) \right]^2$$

$$\text{Classification error}(t) = 1 - \max_i \left[ p(i \mid t) \right]$$

# Gain

- ***Gain of an attribute split:*** compare the impurity of the parent node with the impurity of the child nodes

$$\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

- Maximizing the gain $\Leftrightarrow$ Minimizing the weighted average impurity measure of children nodes

- If **I() = Entropy(),** then **Δ$_{info}$** is called **information gain**

# Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

(Reminder: $p(j \mid t)$ 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 | 0 |
|---|---|
| C2 | 6 |
| Gini=0.000 | |

| C1 | 1 |
|---|---|
| C2 | 5 |
| Gini=0.278 | |

| C1 | 2 |
|---|---|
| C2 | 4 |
| Gini=0.444 | |

| C1 | 3 |
|---|---|
| C2 | 3 |
| Gini=0.500 | |

# Examples for computing GINI

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

| C1 | 0 |
|----|---|
| C2 | 6 |

P(C1) = 0/6 = 0    P(C2) = 6/6 = 1

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

| C1 | 1 |
|----|---|
| C2 | 5 |

P(C1) = 1/6       P(C2) = 5/6

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

| C1 | 2 |
|----|---|
| C2 | 4 |

P(C1) = 2/6       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.
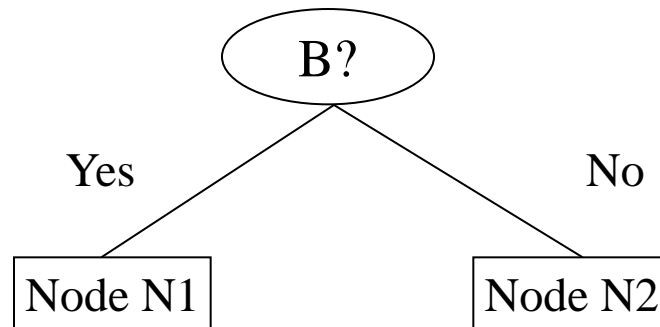
|  | Parent |
|---|---|
| C1 | 6 |
| C2 | 6 |
| **Gini = 0.500** | |

B?

Yes — Node N1

No — Node N2

Gini(N1)
$= 1 - (5/6)^2 - (2/6)^2$
$= 0.194$

Gini(N2)
$= 1 - (1/6)^2 - (4/6)^2$
$= 0.528$

|  | N1 | N2 |
|---|---|---|
| C1 | 5 | 1 |
| C2 | 2 | 4 |
| **Gini=0.333** | | |

Gini(Children)
$= 7/12 * 0.194 +$
$5/12 * 0.528$
$= 0.333$

# 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 | Two-way split
(find best partition of values)

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

| 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 the splitting 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 best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

| 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 |

Taxable Income > 80K?

Yes    No

# 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 | |
| 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 |
| 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 | |

# 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

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

| C1 | 0 |
|----|---|
| C2 | 6 |

P(C1) = 0/6 = 0     P(C2) = 6/6 = 1

Entropy = − 0 log 0 − 1 log 1 = − 0 − 0 = 0

| C1 | 1 |
|----|---|
| C2 | 5 |

P(C1) = 1/6      P(C2) = 5/6

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

| C1 | 2 |
|----|---|
| C2 | 4 |

P(C1) = 2/6      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
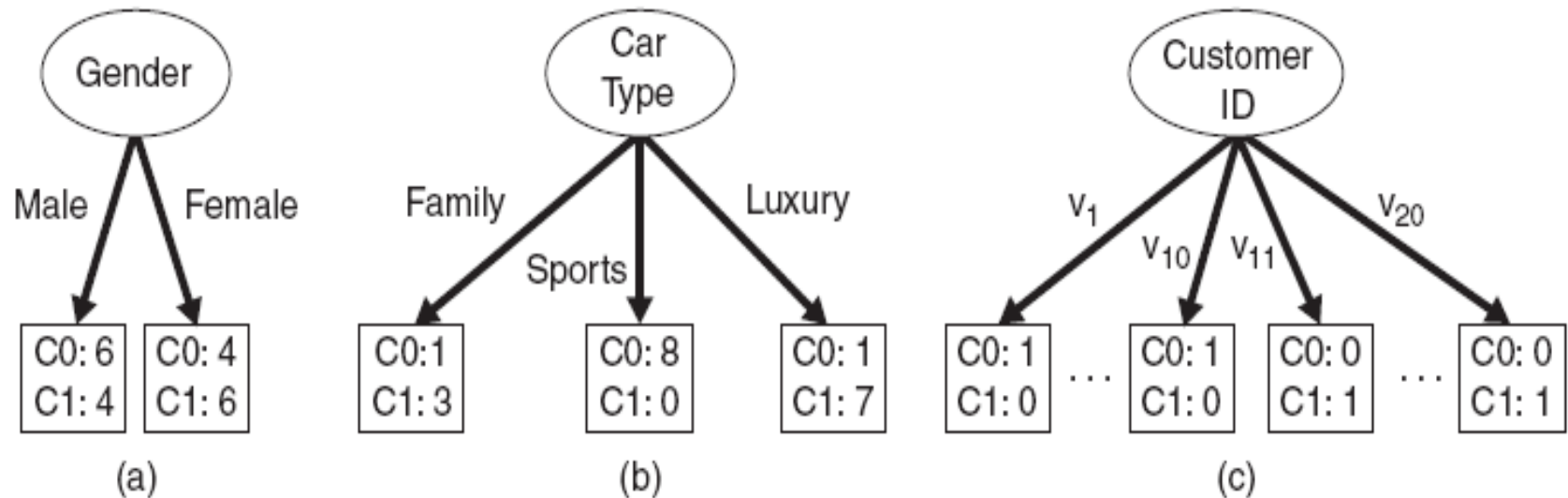


Figure 4.12. Multiway versus binary splits.

# Splitting based on INFO

- Impurity measures favor attributes with large number of values

- A test condition with large number of outcomes may not be desirable
  - # of records in each partition is too small to make predictions

# Splitting Based on INFO...

- Gain Ratio:

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

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 :

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

- 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

$$Error(t) = 1 - \max_{i} P(i \mid t)$$

| | |
|---|---|
| C1 | **0** |
| C2 | **6** |

P(C1) = 0/6 = 0    P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

| | |
|---|---|
| C1 | **1** |
| C2 | **5** |

P(C1) = 1/6        P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

| | |
|---|---|
| C1 | **2** |
| C2 | **4** |

P(C1) = 2/6        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:
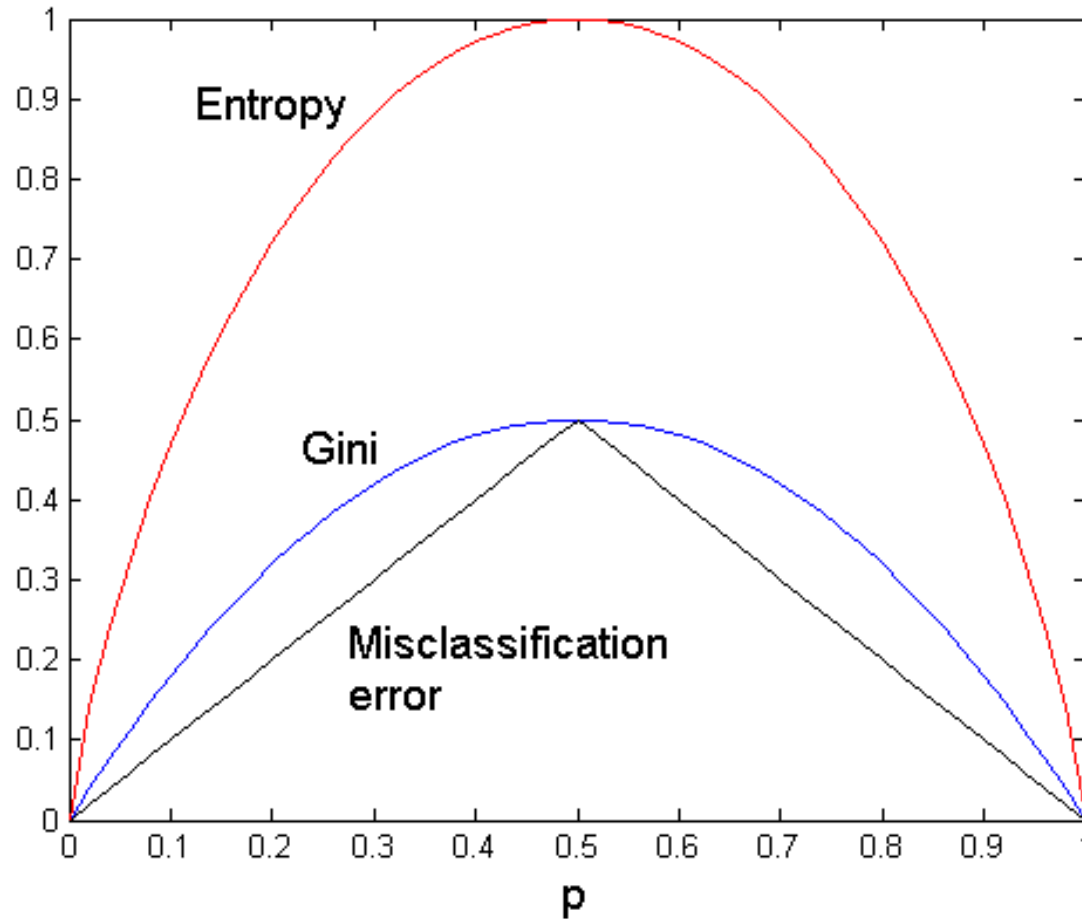
# Misclassification Error vs Gini
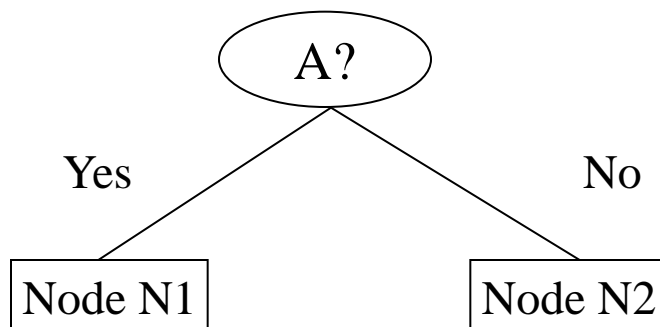


|     | Parent |
| --- | --- |
| C1  | 7 |
| C2  | 3 |
| **Gini = 0.42** | |
| **Error = 0.3** | |

Gini(N1)
$= 1 - (3/3)^2 - (0/3)^2 = 0$
Error(N1)
$= 1 - \max(0,1) = 0$

Gini(N2)
$= 1 - (4/7)^2 - (3/7)^2 = 0.489$
Error(N2)
$= 1 - \max(3/7, 4/7) = 3/7$

|     | N1 | N2 |
| --- | --- | --- |
| C1  | 3 | 4 |
| C2  | 0 | 3 |
| **Gini=0.342** | | |
| **Error = 0.3** | | |

Gini(Children)
$= 3/10 * 0 + 7/10 * 0.489$
$= 0.342$
Error(Children)
$= 3/10*0 + 7/10*3/7 = 0.3$

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)

# 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

# Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

# Practical Issues of Classification

- Underfitting and Overfitting

- Missing Values

- Costs of Classification

# Underfitting and Overfitting (Example)
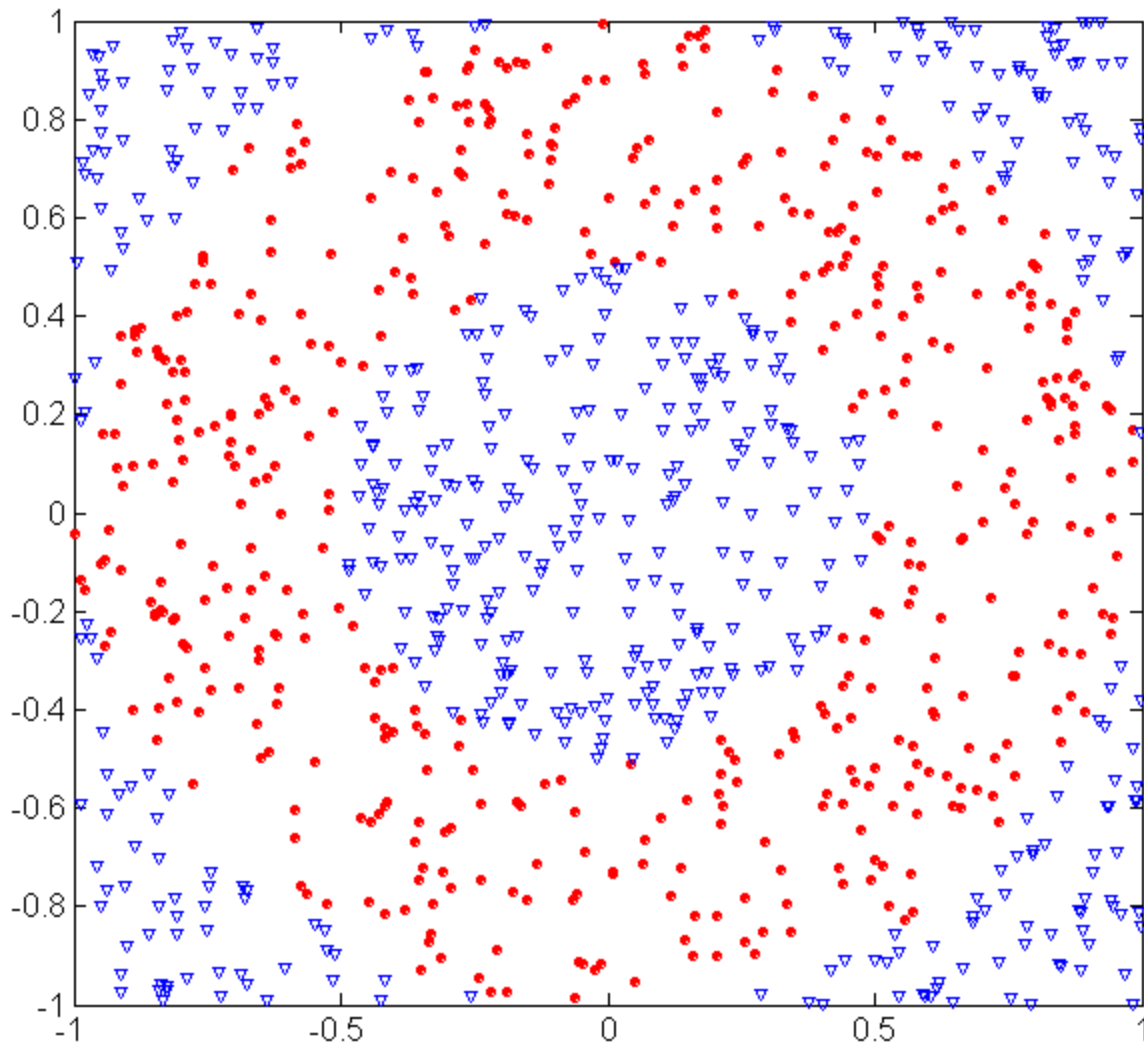


500 circular and 500 triangular data points.

Circular points:

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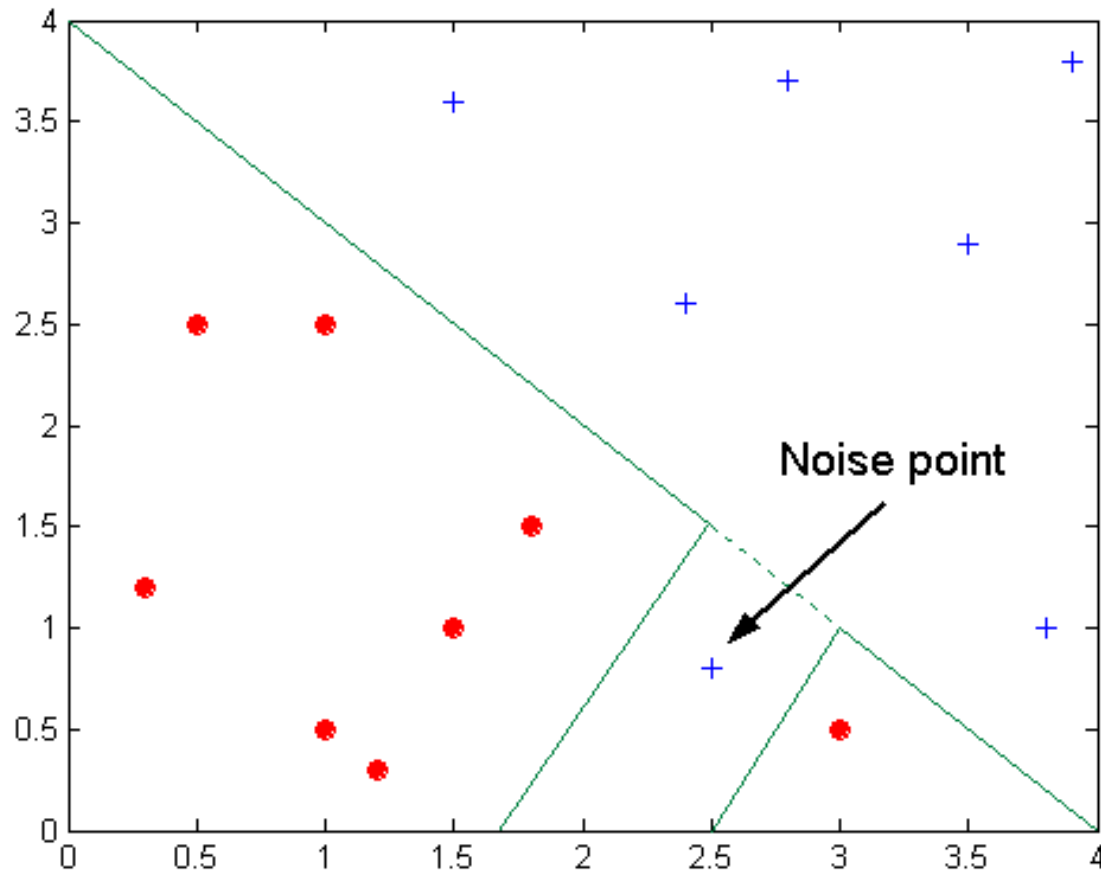
Triangular points:

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

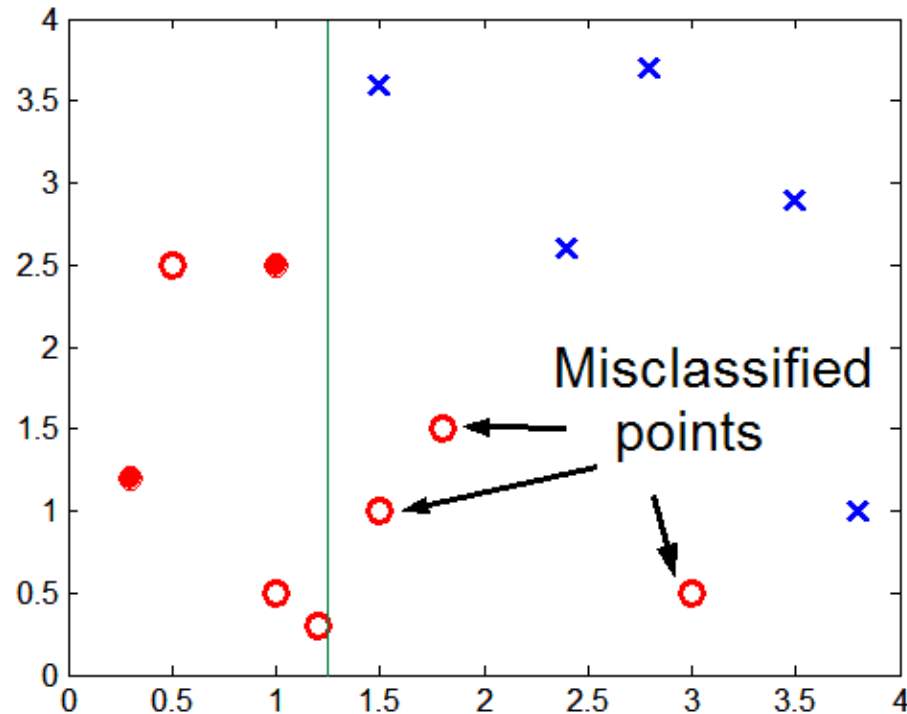$\text{sqrt}(x_1{}^2 + x_2{}^2) < 1$

# Underfitting and Overfitting
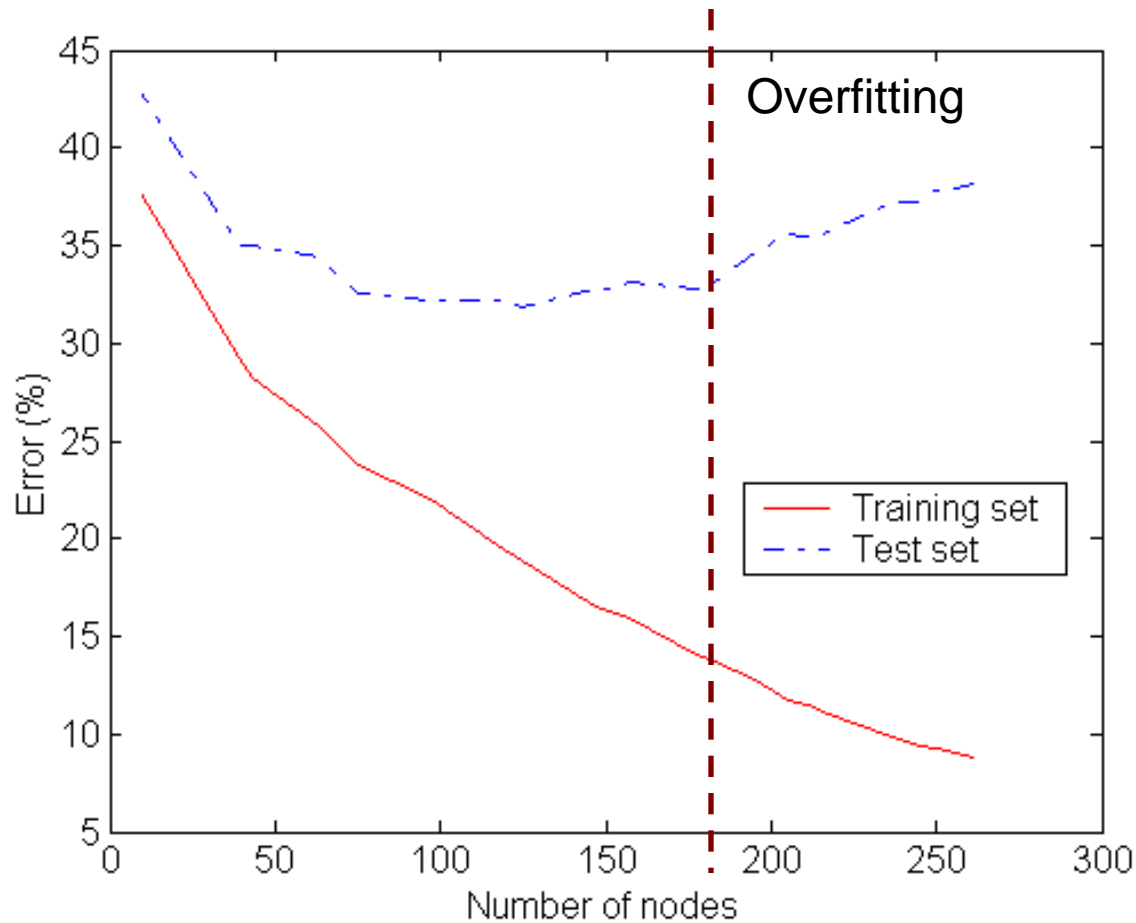


Underfitting: when model is too simple, both training and test errors are large

# 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
  - The model does not generalize well

- Need new ways for estimating errors

# Estimating Generalization Errors

- Re-substitution errors: error on training ($\Sigma$ e(t) )
- Generalization errors: error on testing ($\Sigma$ 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 dataset 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

# Minimum Description Length (MDL)
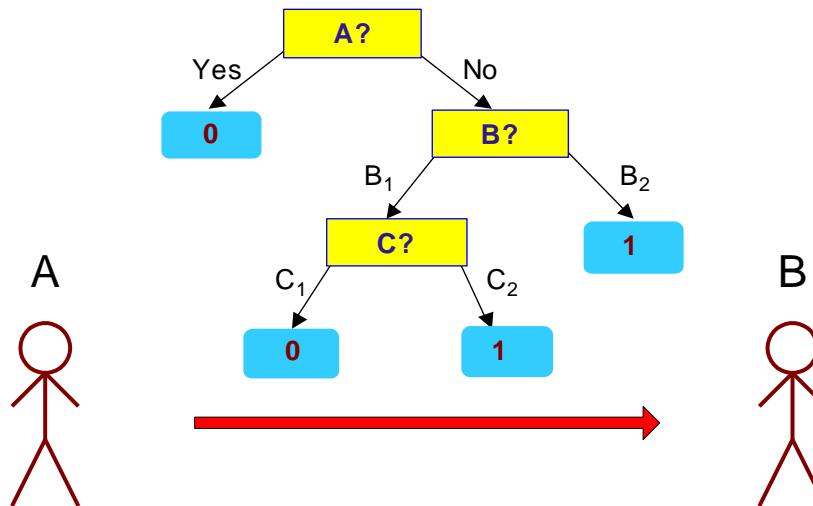


- Cost(Model,Data) = Cost(Data|Model) + Cost(Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

# 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.
  - 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
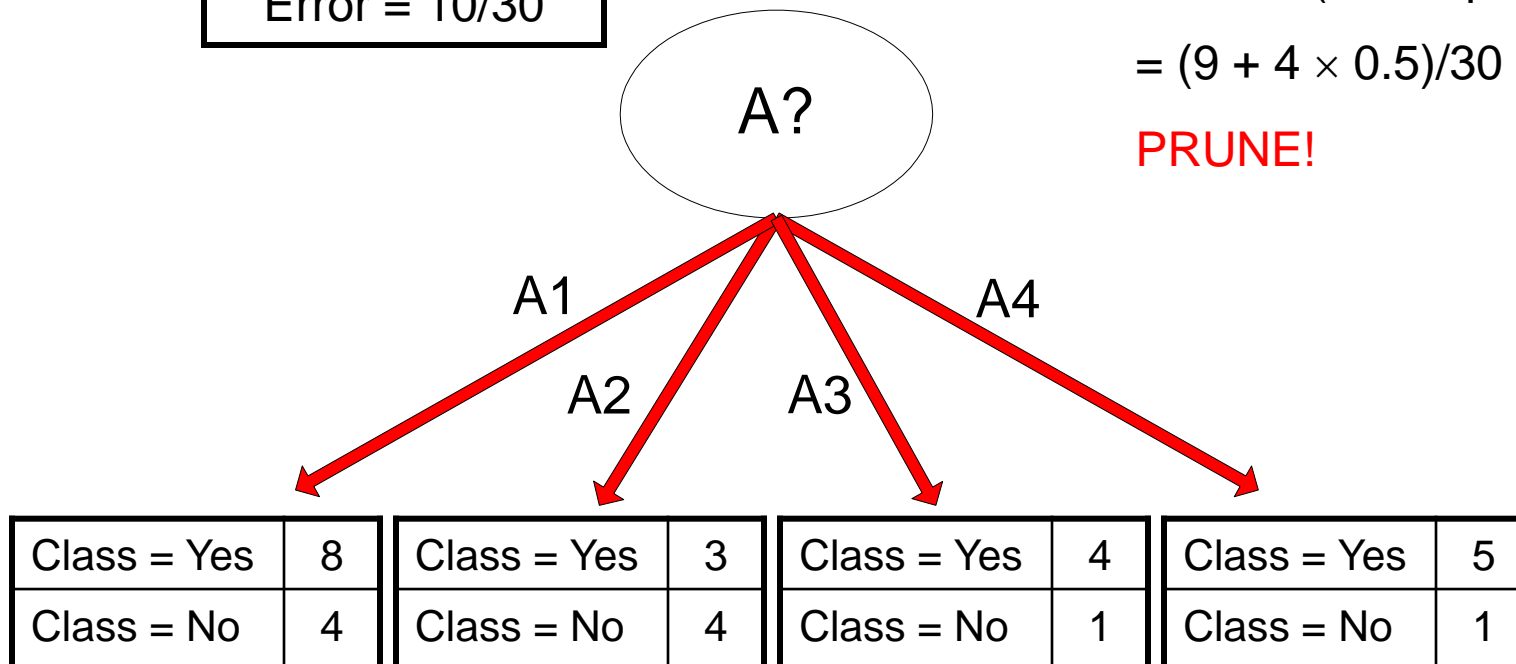
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

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

PRUNE!

A?

A1    A2    A3    A4

| Class = Yes | 8 |
|---|---|
| Class = No | 4 |

| Class = Yes | 3 |
|---|---|
| Class = No | 4 |

| Class = Yes | 4 |
|---|---|
| Class = No | 1 |

| Class = Yes | 5 |
|---|---|
| Class = No | 1 |

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

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

# Distribute Instances

| 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 |

| Tid | Refund | Marital Status | Taxable Income | Class |
|-----|--------|----------------|----------------|-------|
| 10 | ? | Single | 90K | Yes |

Refund

Yes          No

| Class=Yes | 0 |
|-----------|---|
| Class=No | 3 |

| Cheat=Yes | 2 |
|-----------|---|
| Cheat=No | 4 |

Refund

Yes          No

| Class=Yes | 0 + 3/9 |
|-----------|---------|
| Class=No | 3 |

| Class=Yes | 2 + 6/9 |
|-----------|---------|
| Class=No | 4 |
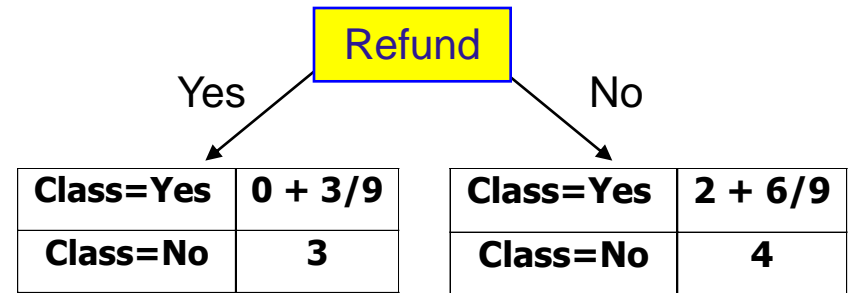
Probability that Refund=Yes is 3/9

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
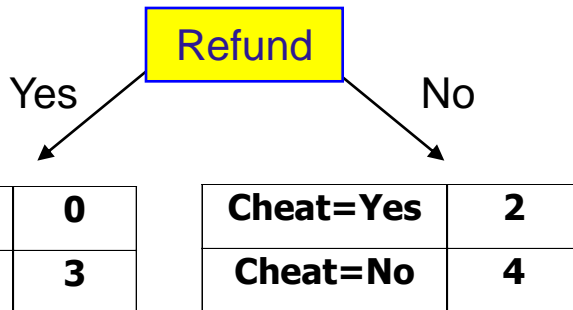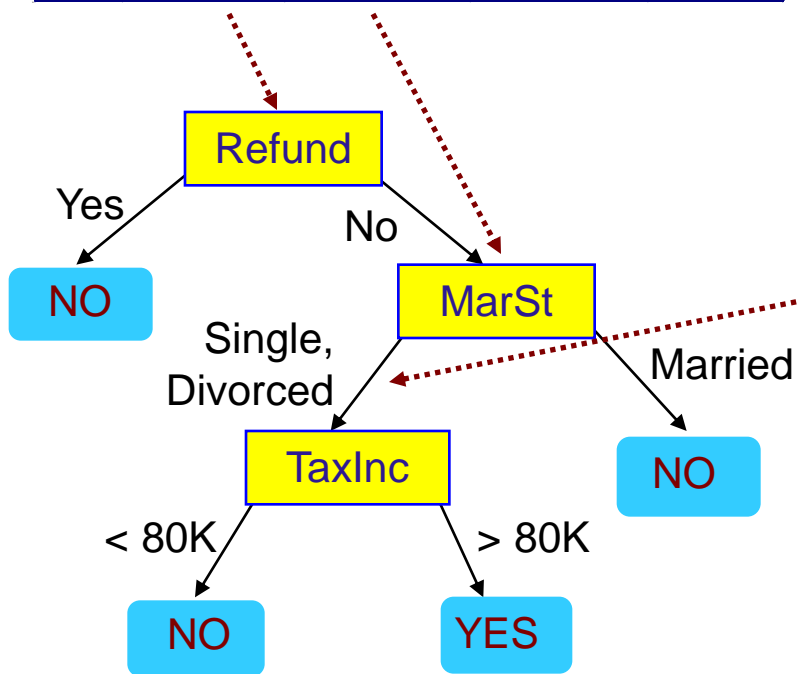
# Classify Instances

New record:

| Tid | 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  |

Refund

Yes → NO

No → MarSt

MarSt — Single, Divorced → TaxInc

MarSt — Married → NO

TaxInc — < 80K → NO

TaxInc — > 80K → YES

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
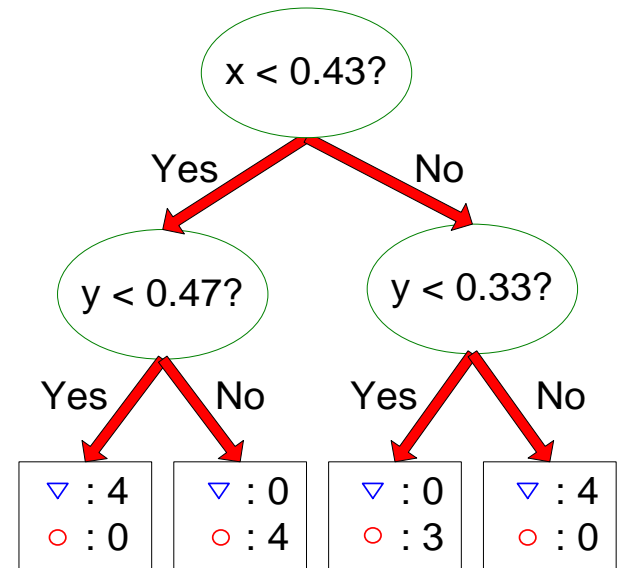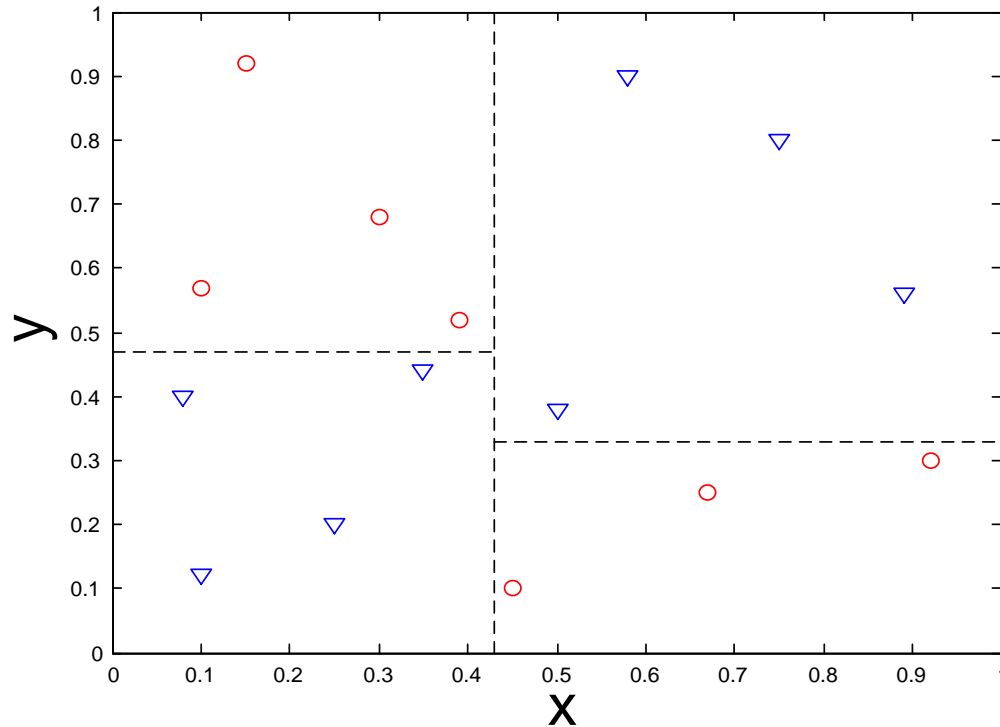


• Border line between two neighboring regions of different classes is known as decision boundary

• Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

# Oblique Decision Trees



$$x + y < 1$$

Class = **+**     Class = ●
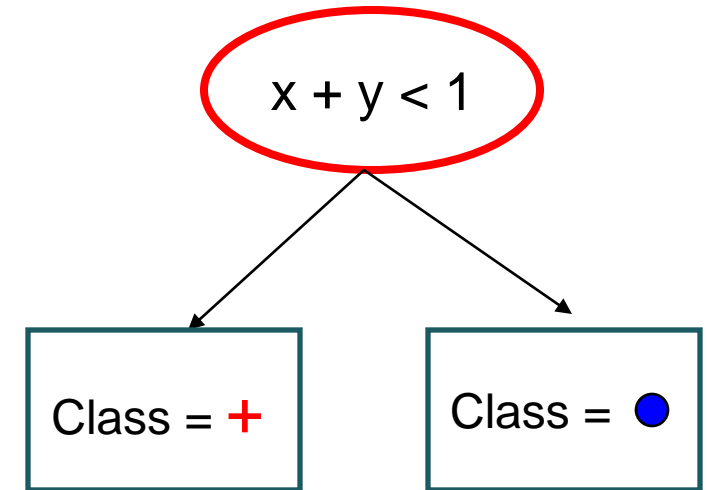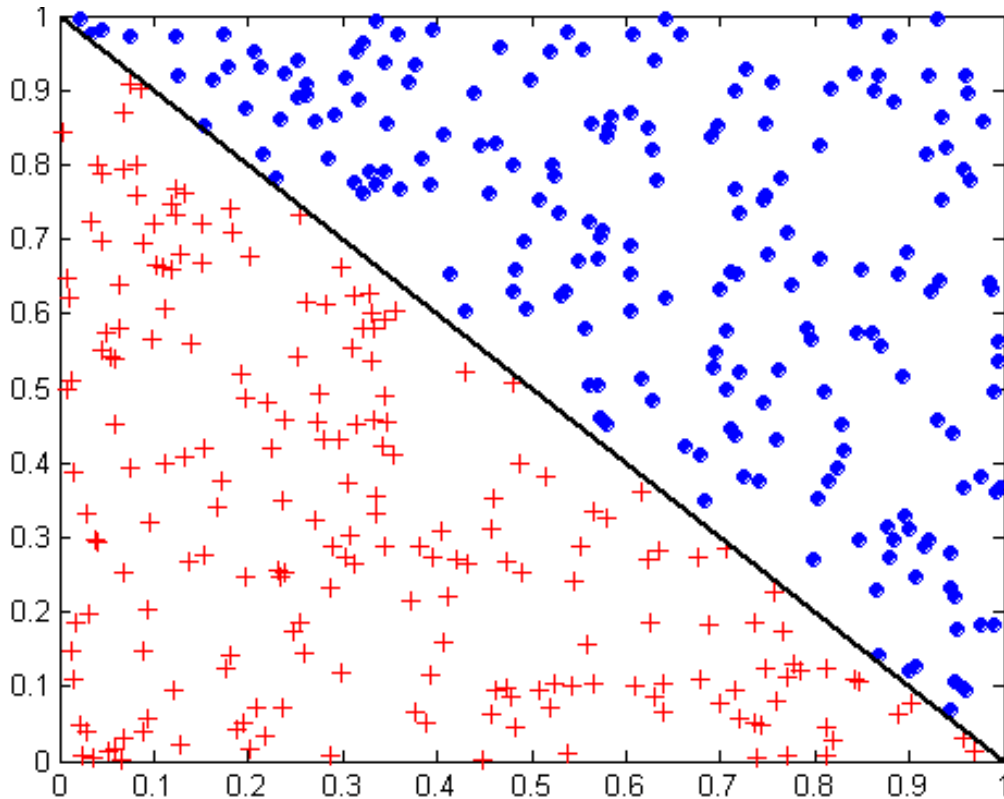
- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

# Tree Replication
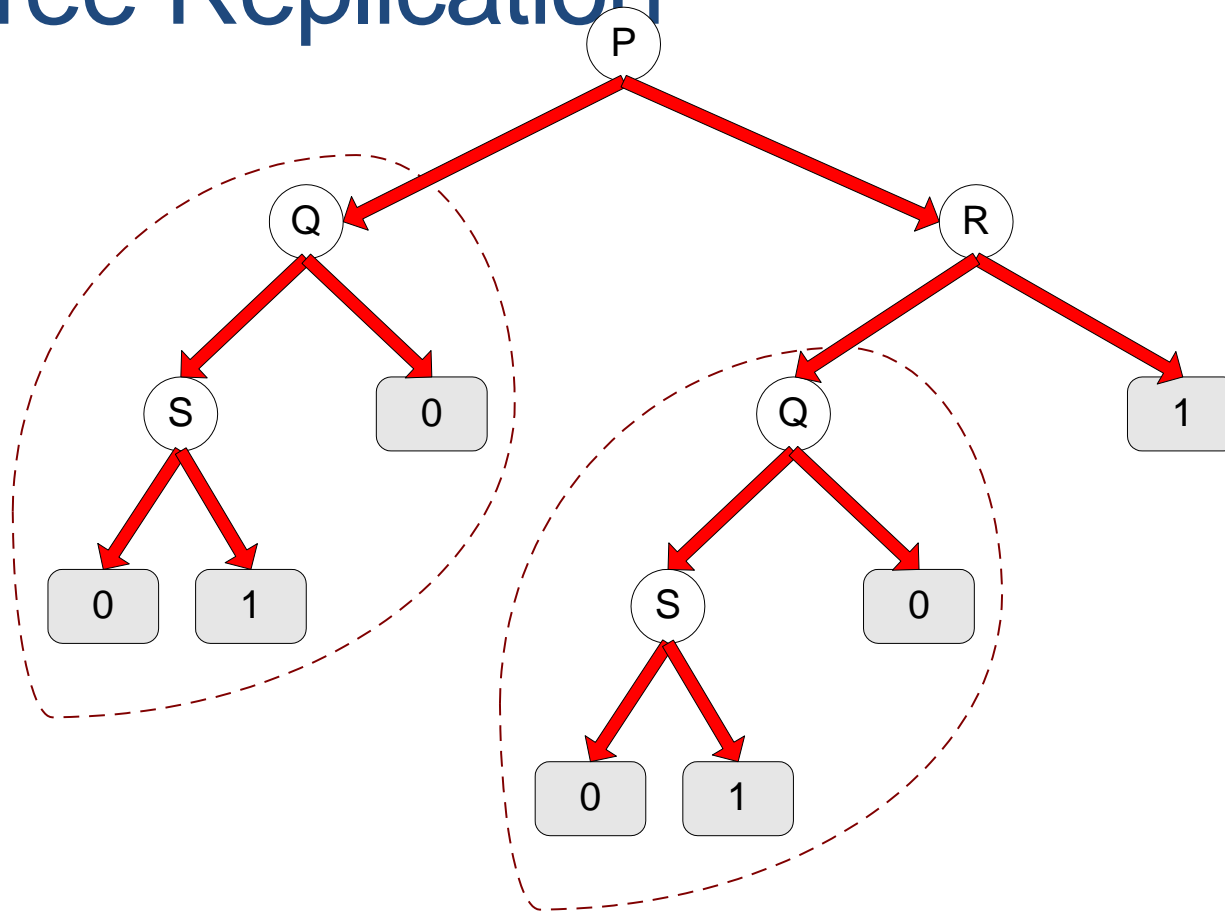


- Same subtree appears in multiple branches