# Data Mining Classification: Basic Concepts

Chiara Renso
KDD-LAB
ISTI- CNR, Pisa, Italy

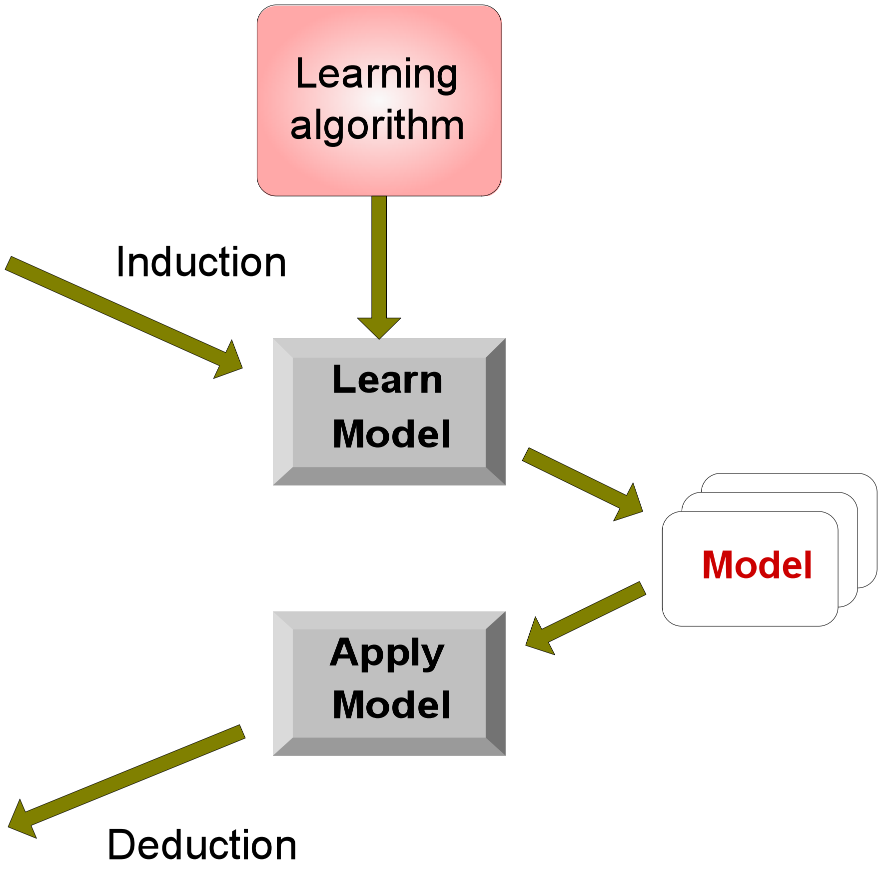
# Classification: Definition

- 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



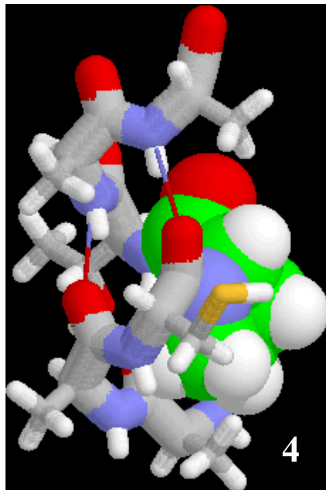

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

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

# Classification Techniques

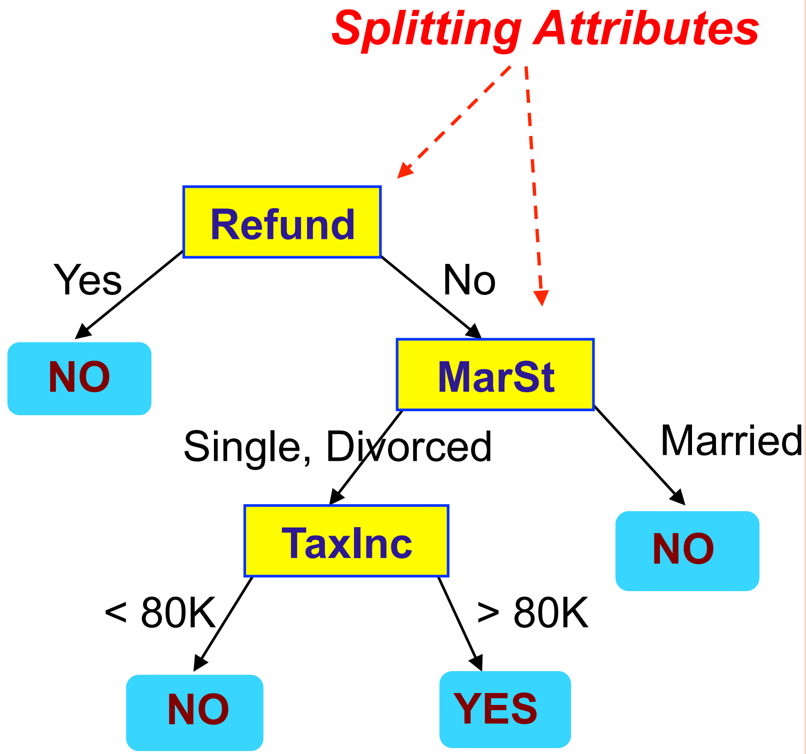
- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

# 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

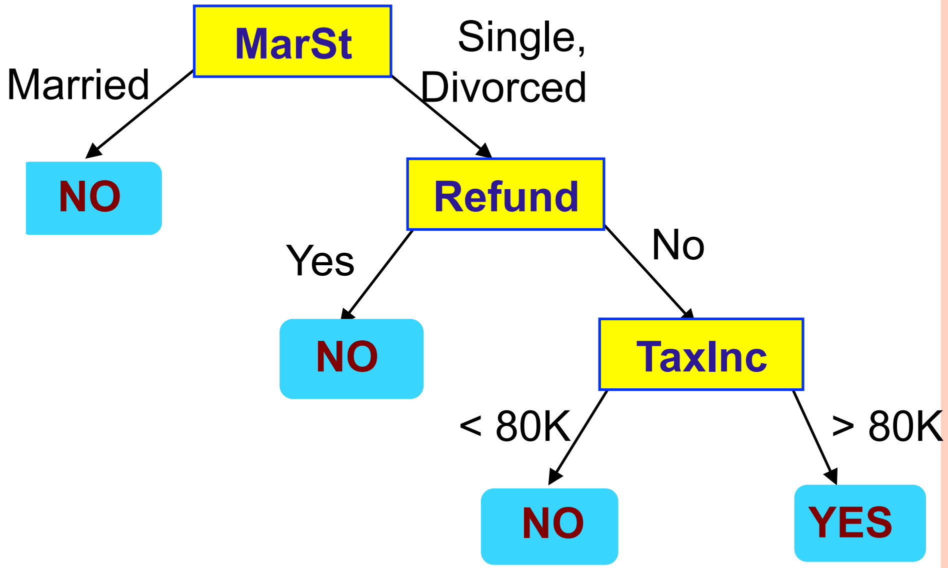



Model: Decision Tree

# Another Example of Decision Tree

categorical categorical continuous class

| *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 |

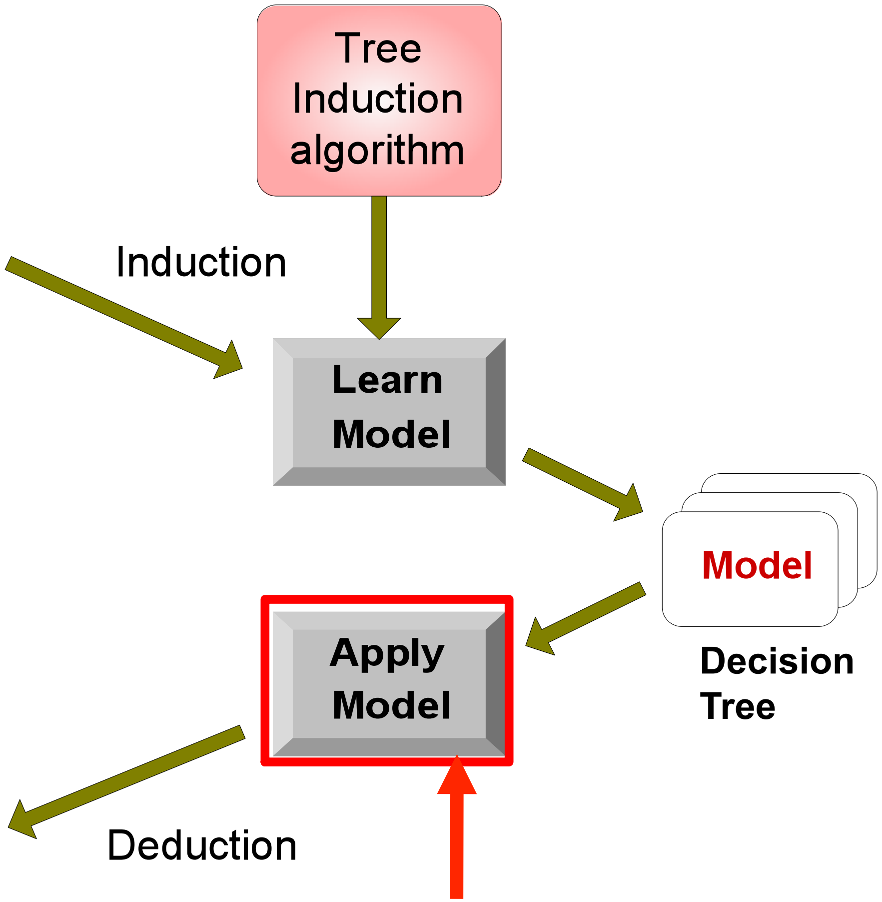



**There could be more than one tree that fits the same data!**

# 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

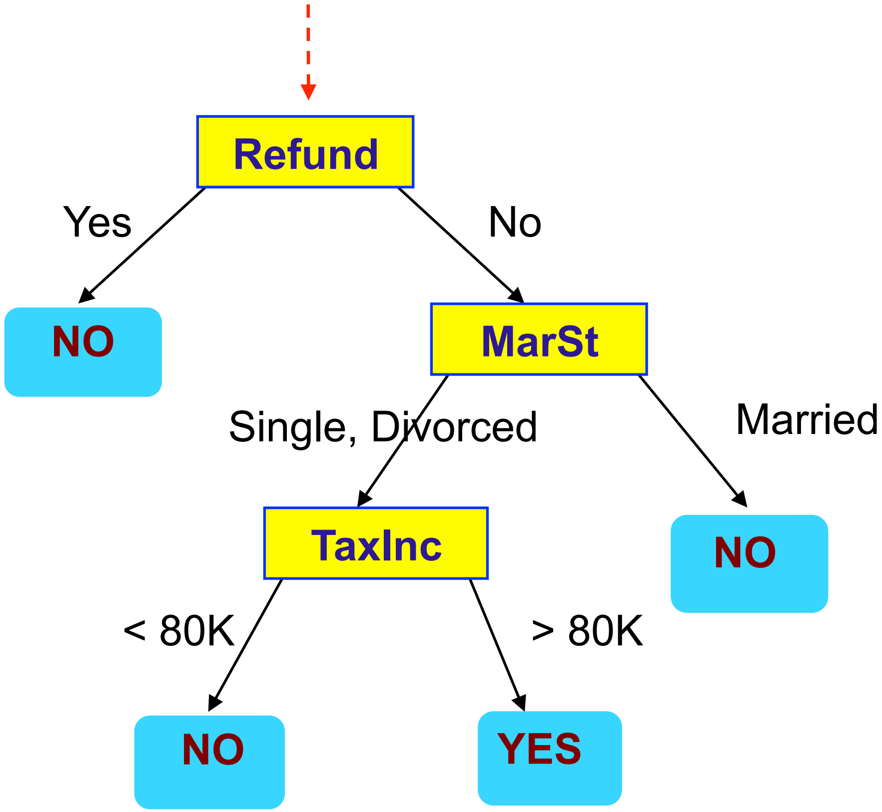



| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set

# Apply Model to Test Data

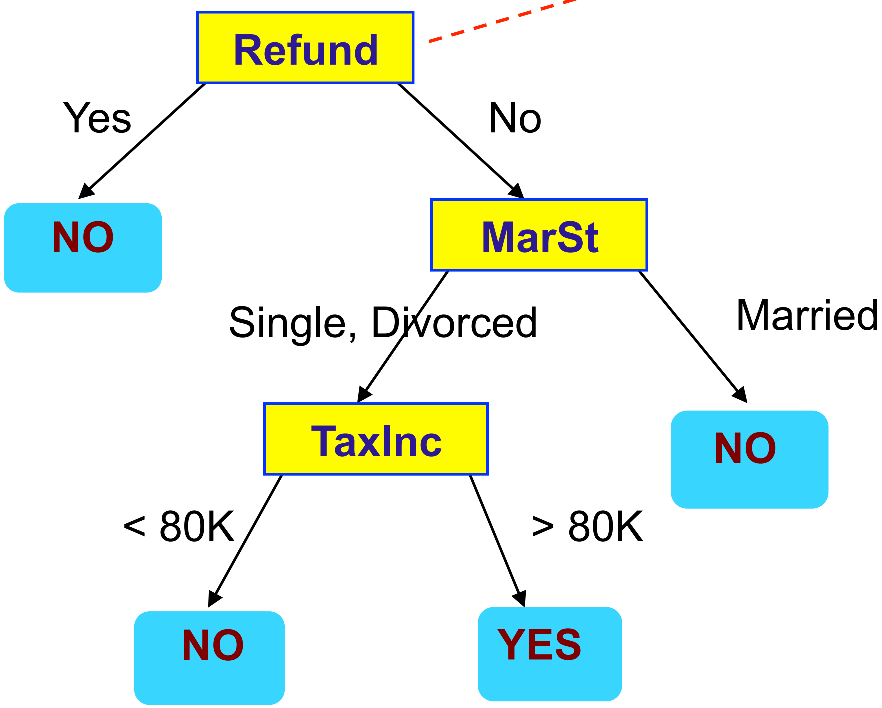
Start from the root of tree.




**Test Data**

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

# Apply Model to Test Data



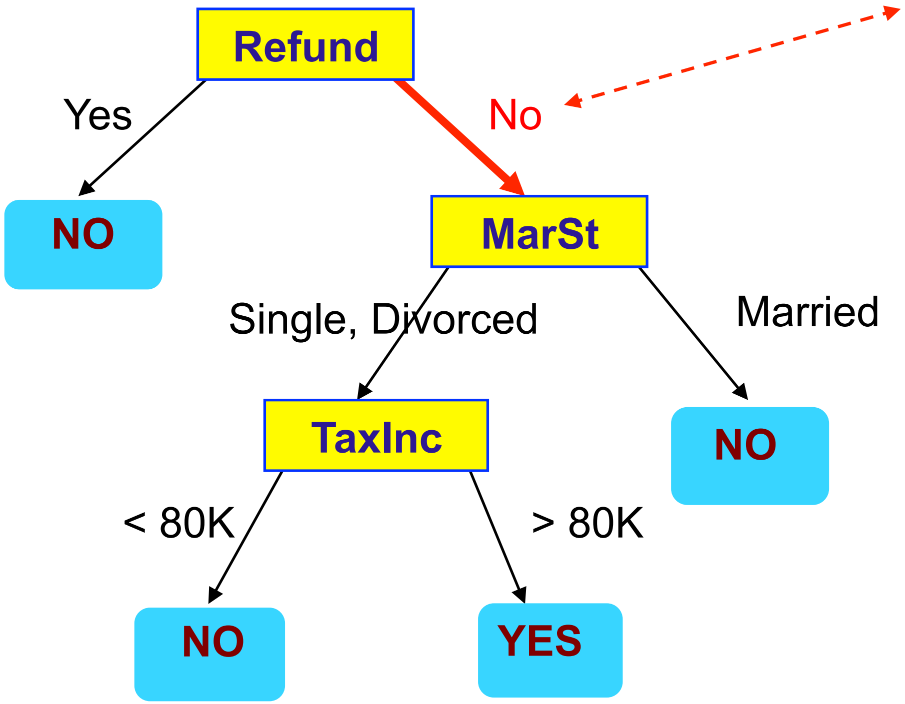

**Test Data**

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

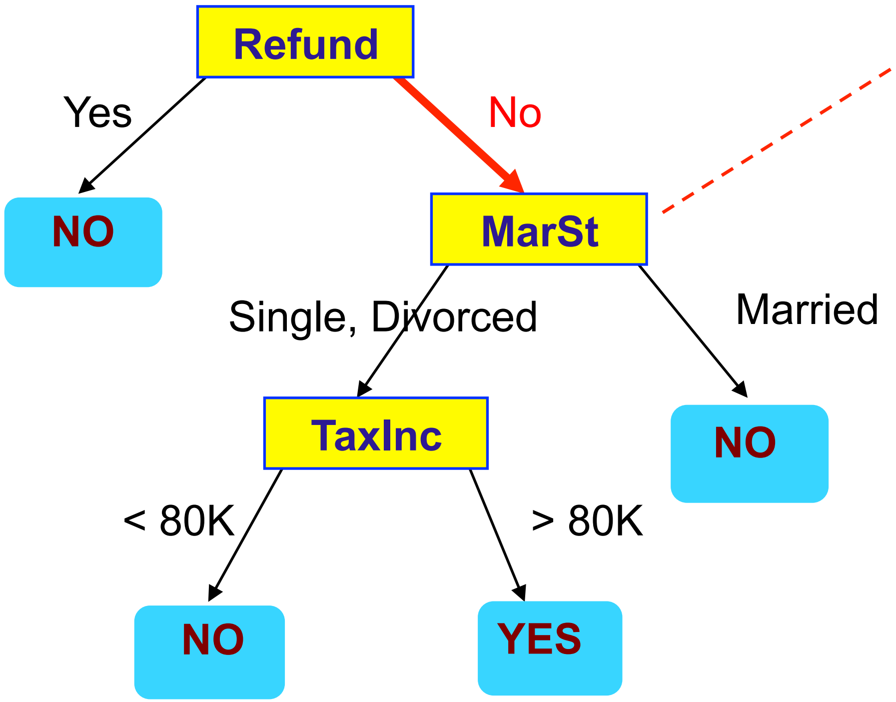
# Apply Model to Test Data

Test Data

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



# Apply Model to Test Data



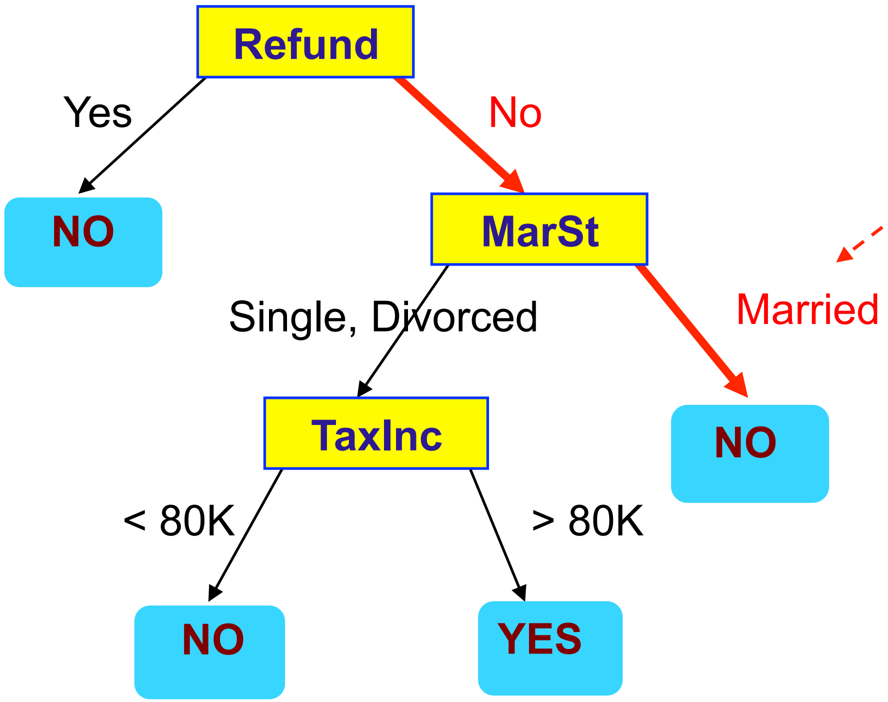

**Test Data**

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

# APPLY MODEL TO TEST DATA

**Test Data**

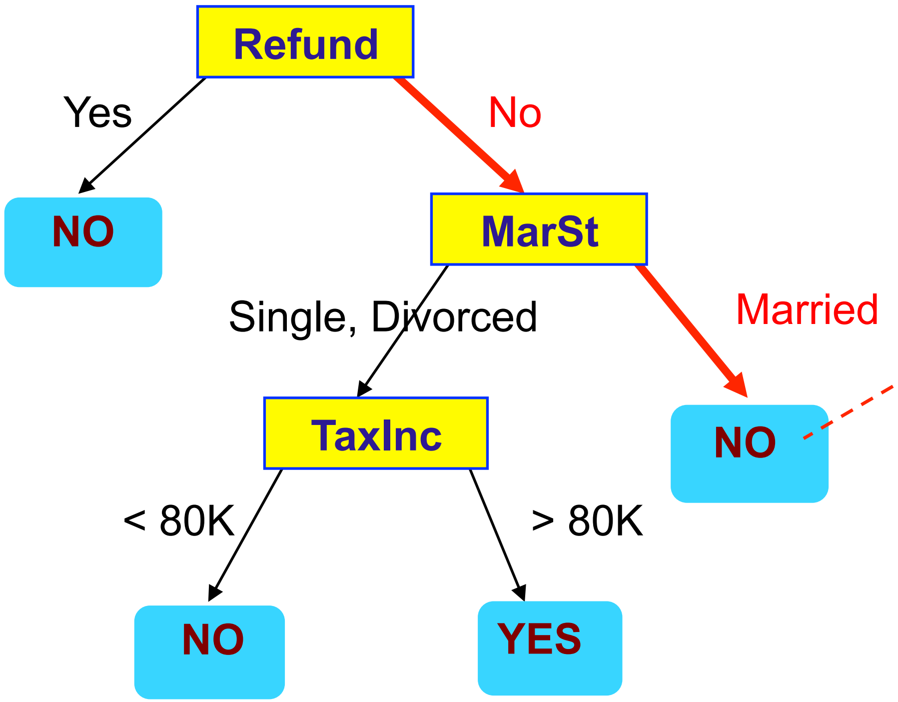
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply Model to Test Data

**Test Data**

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

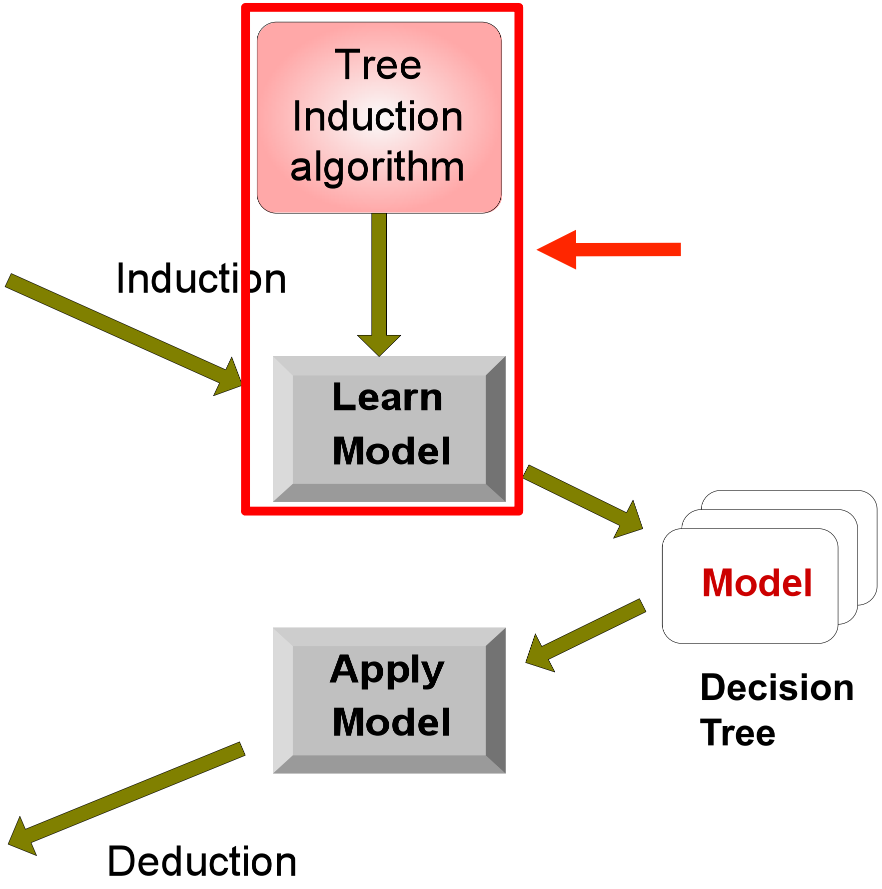



Assign Cheat to "No"

# 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

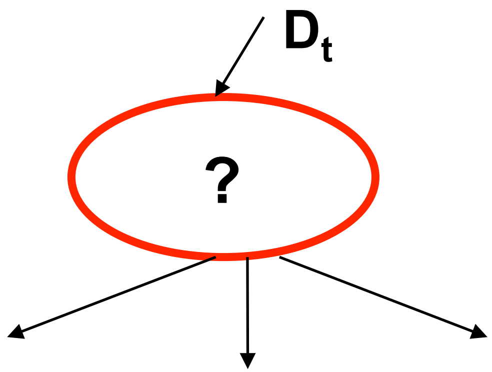
# Decision Tree Induction

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

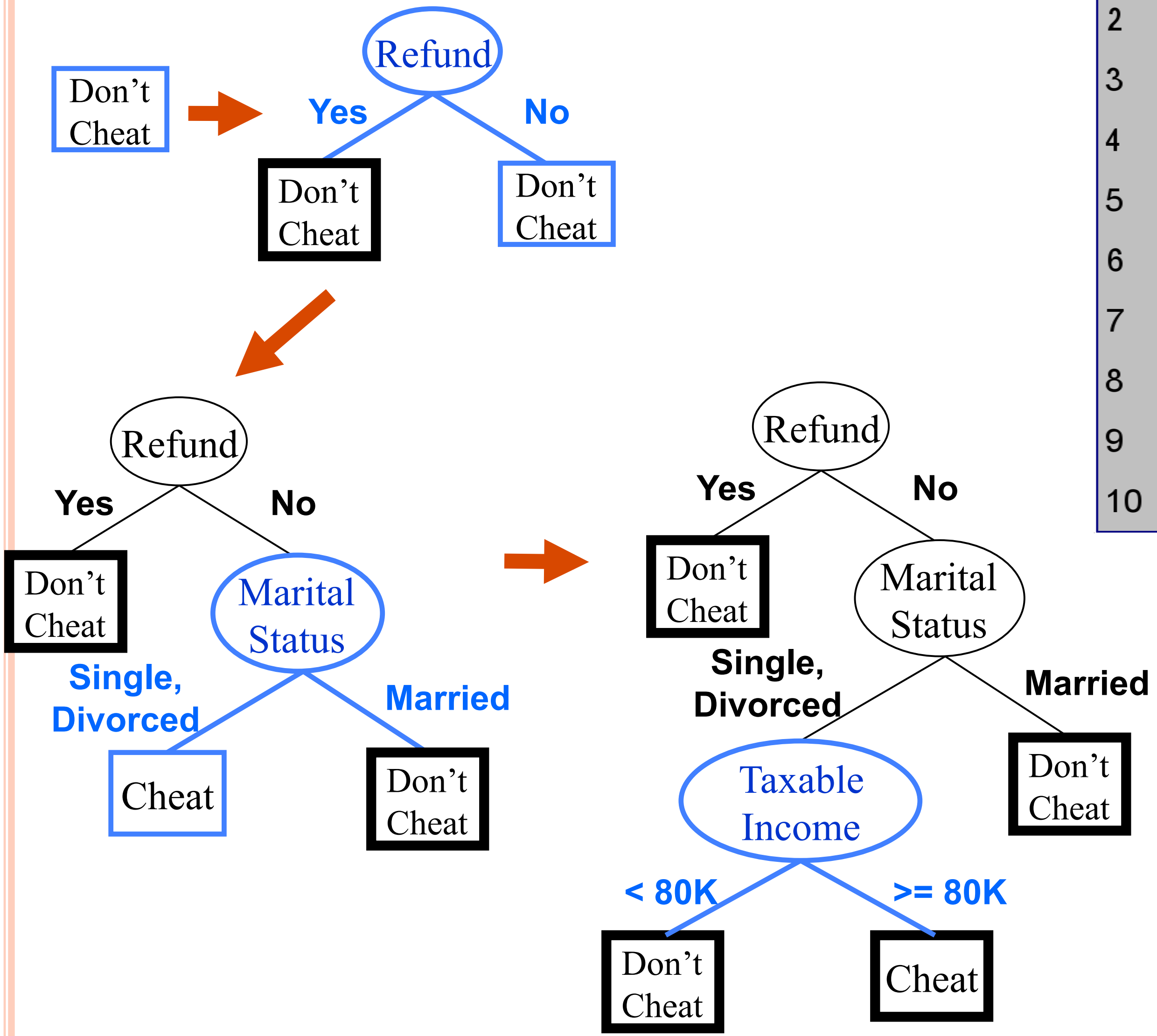
# General Structure of Hunt's Algorithm

- Let Dt be the set of training records that are associated with node t and y = $\{y_1, y_2, \ldots, y_c\}$ be the class labels.
- 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 |



# 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

# 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

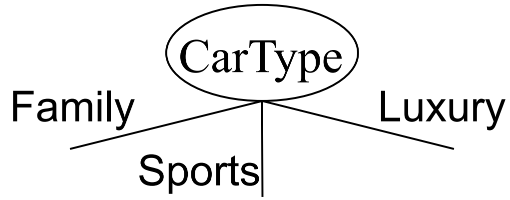
# How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

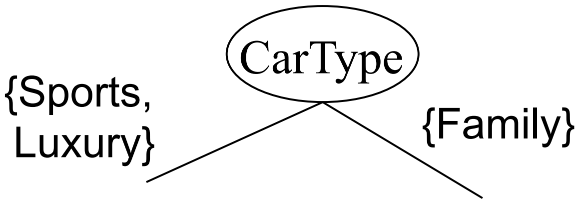
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

# SPLITTING BASED ON NOMINAL ATTRIBUTES

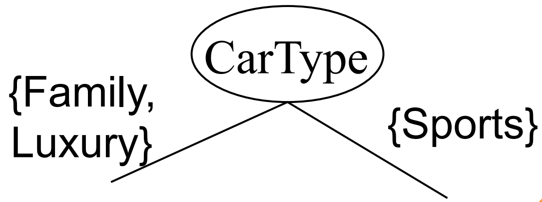
- Multi-way split: Use as many partitions as distinct values.




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

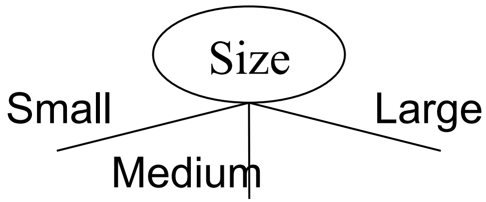



OR

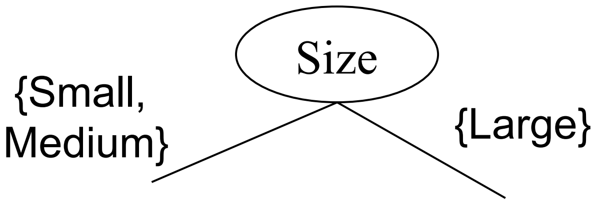


# SPLITTING BASED ON ORDINAL ATTRIBUTES

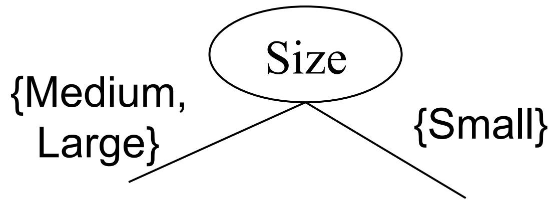
- Multi-way split: Use as many partitions as distinct values.



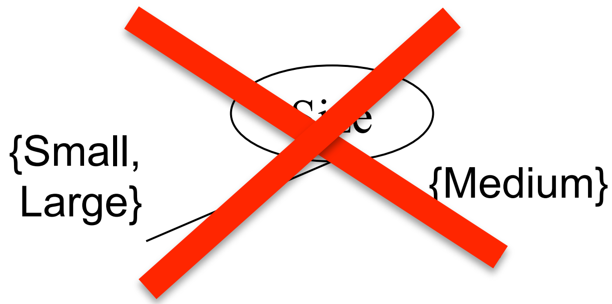

- Binary split: Divides values into two subsets. Need to find optimal partitioning.




OR



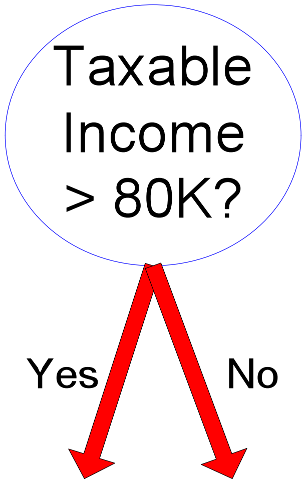

- What about this split?



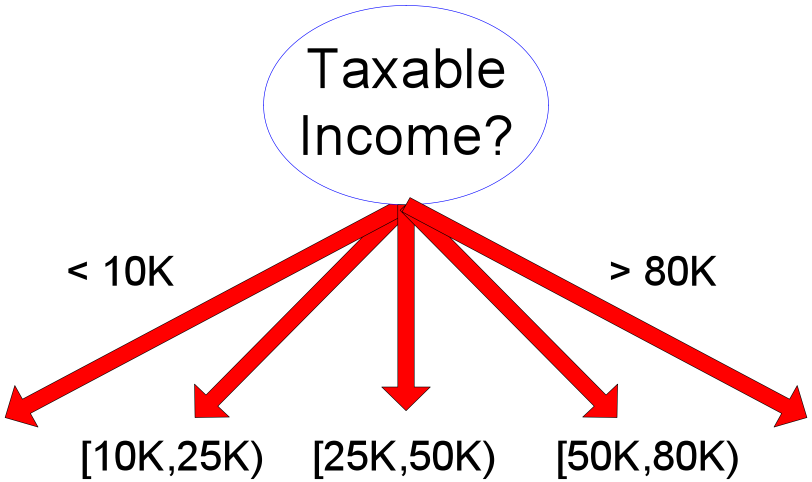
# Splitting Based on Continuous Attributes

- 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

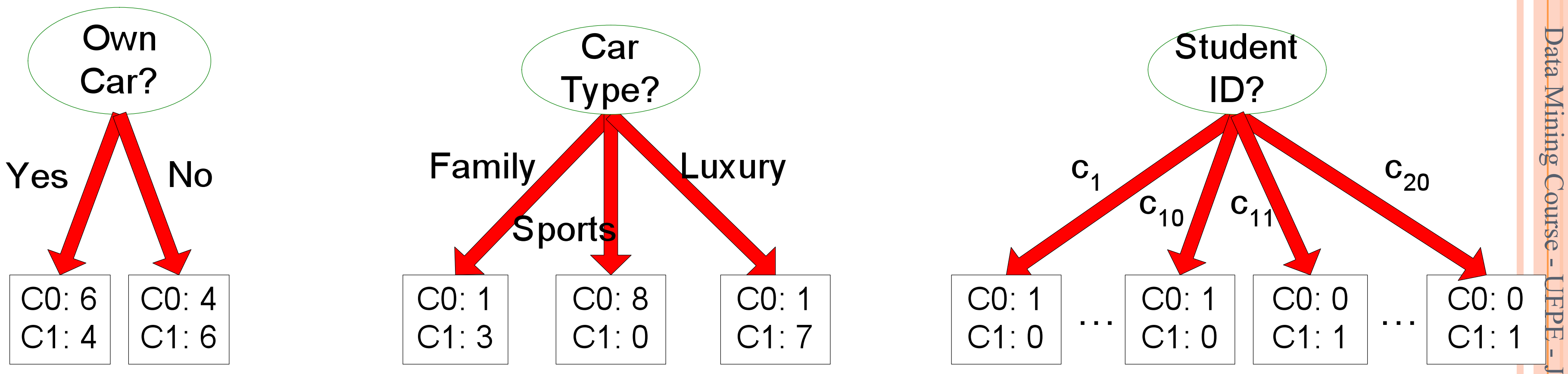
# Tree Induction

- 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<br>C1: 5 | C0: 9<br>C1: 1 |
| **Non-homogeneous,**<br>**High degree of impurity** | **Homogeneous,**<br>**Low degree of impurity** |

# Measures of Node Impurity

- Gini Index

- Entropy

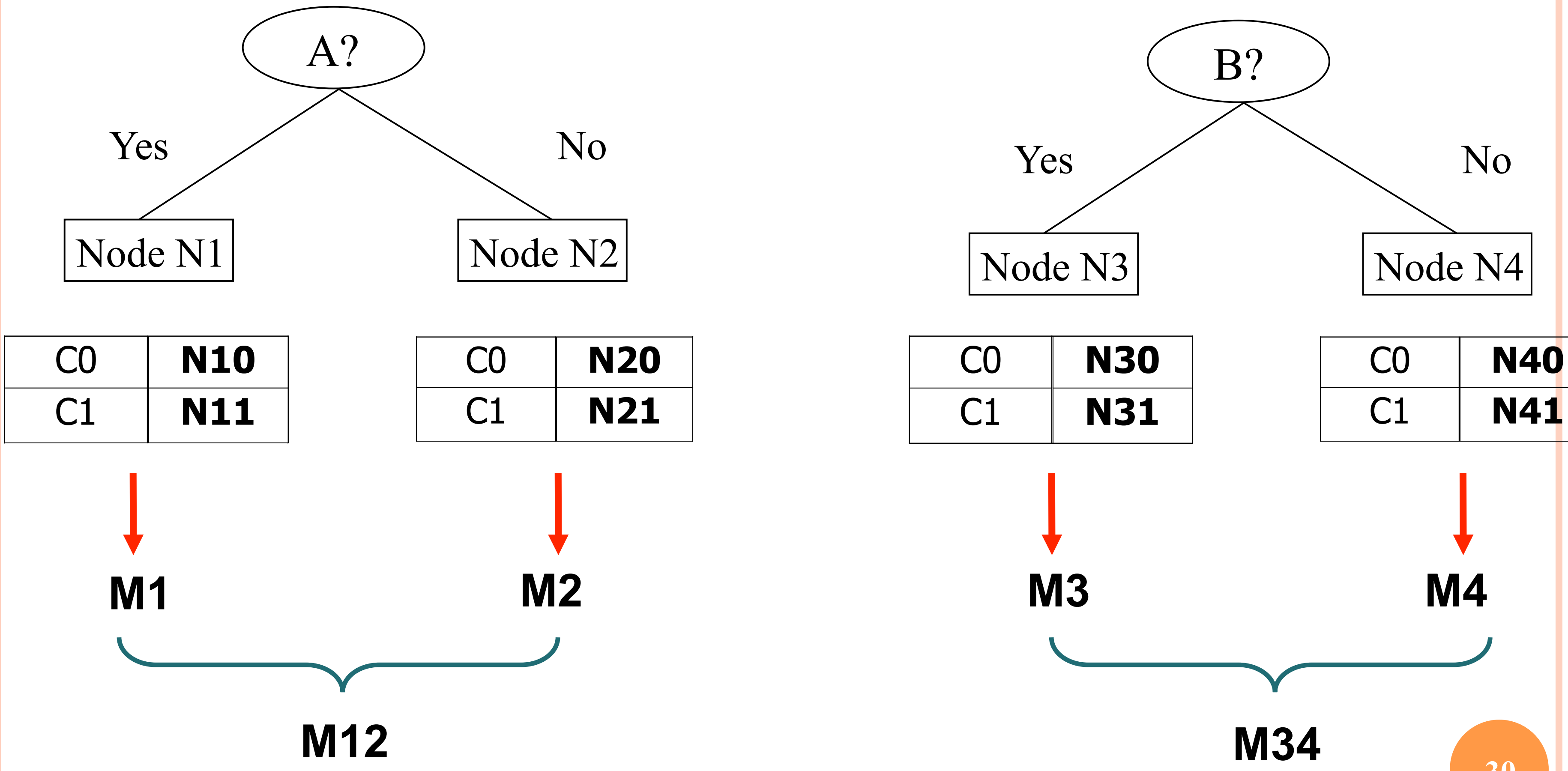
- Misclassification error

# How to Find the Best Split

**Before Splitting:**

| | |
|---|---|
| C0 | **N00** |
| C1 | **N01** |

→ **M0**




| | |
|---|---|
| C0 | **N10** |
| C1 | **N11** |

| | |
|---|---|
| C0 | **N20** |
| C1 | **N21** |

| | |
|---|---|
| C0 | **N30** |
| C1 | **N31** |

| | |
|---|---|
| C0 | **N40** |
| C1 | **N41** |

**Gain = M0 – M12 vs M0 – M34**

# MEASURE OF IMPURITY: GINI

- Gini Index for a given node t :

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

(NOTE: $p(j \mid t)$ 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

# 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

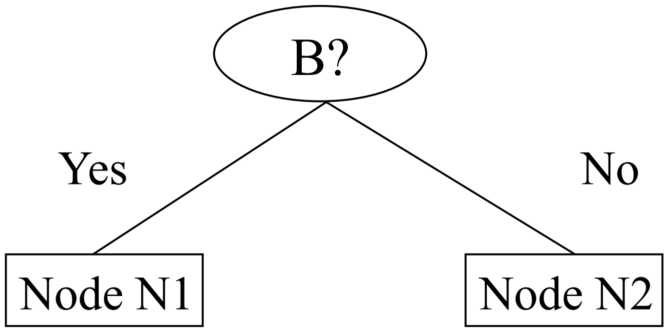
- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

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

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.




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

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

**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

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

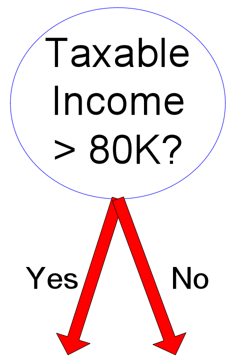
Two-way split
(find best partition of values)

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

# 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 ≥ 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 |



# 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 | 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:

$$Entropy(t) = -\sum_{j} p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ 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 Criteria based on Classification Error

- Classification error at a node t :

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

- 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** |

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

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

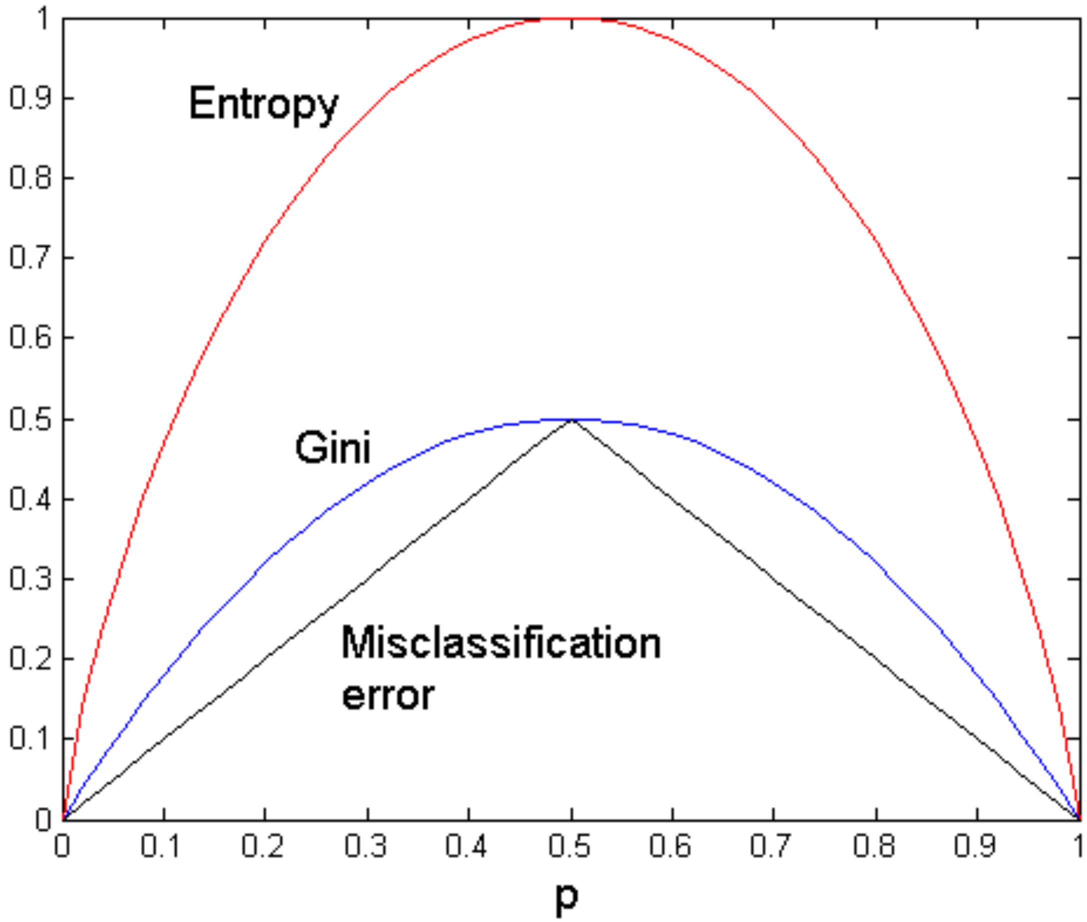
| | |
|---|---|
| C1 | **2** |
| C2 | **4** |

P(C1) = 2/6     P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1

# 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

# 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

# Estimating Generalization Errors

# UNDERFITTING AND OVERFITTING

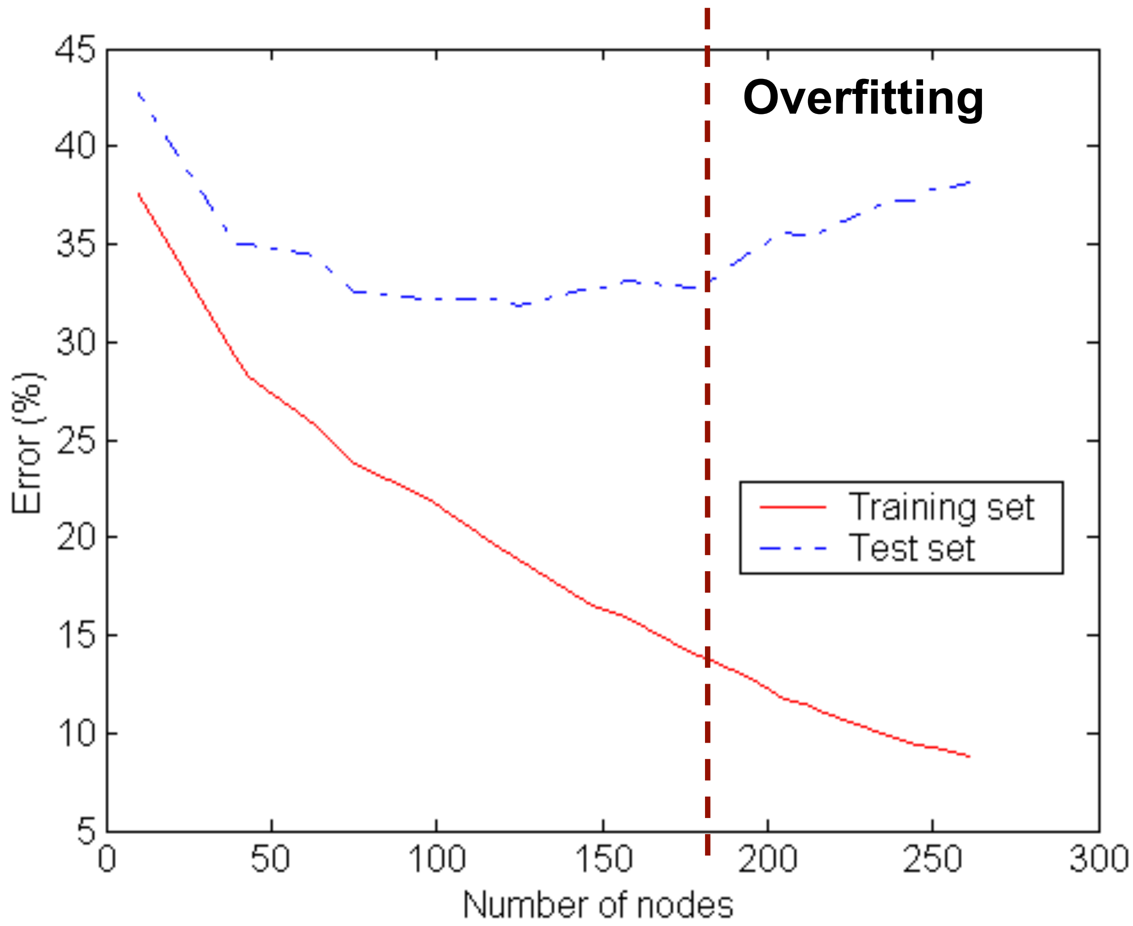
Re-substitution or training errors: error on training
Generalization or test errors: error on testing

Good model= low training and test errors

Model Underfitting: high training and high test errors → the tree is too simple and not fully developed.

Model Overfitting: low training error, high test error → tree too complex and fits only the training test.
Overfitting could depend on noise on datasets or due to lack of representative samples.

# Underfitting and Overfitting




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

# 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

# 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

# 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

# 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 Model Comparison
  - How to compare the relative performance among competing models?

# 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 Matrix

| | PREDICTED CLASS | | |
|---|---|---|---|
| **ACTUAL CLASS** | C(i\|j) | **Class=Yes** | **Class=No** |
| | **Class=Yes** | C(Yes\|Yes) | C(No\|Yes) |
| | **Class=No** | C(Yes\|No) | C(No\|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 |

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

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

# OTHER MEASURES

$$\text{Precision } (p) = \frac{TP}{TP + FP}$$

$$\text{Recall } (r) = \frac{TP}{TP + FN}$$

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

- Precision is a measure of exactness: Precision of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly)
- Recall is a measure of completeness: Recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C)
- F-measure is the harmonic mean of precision and recall