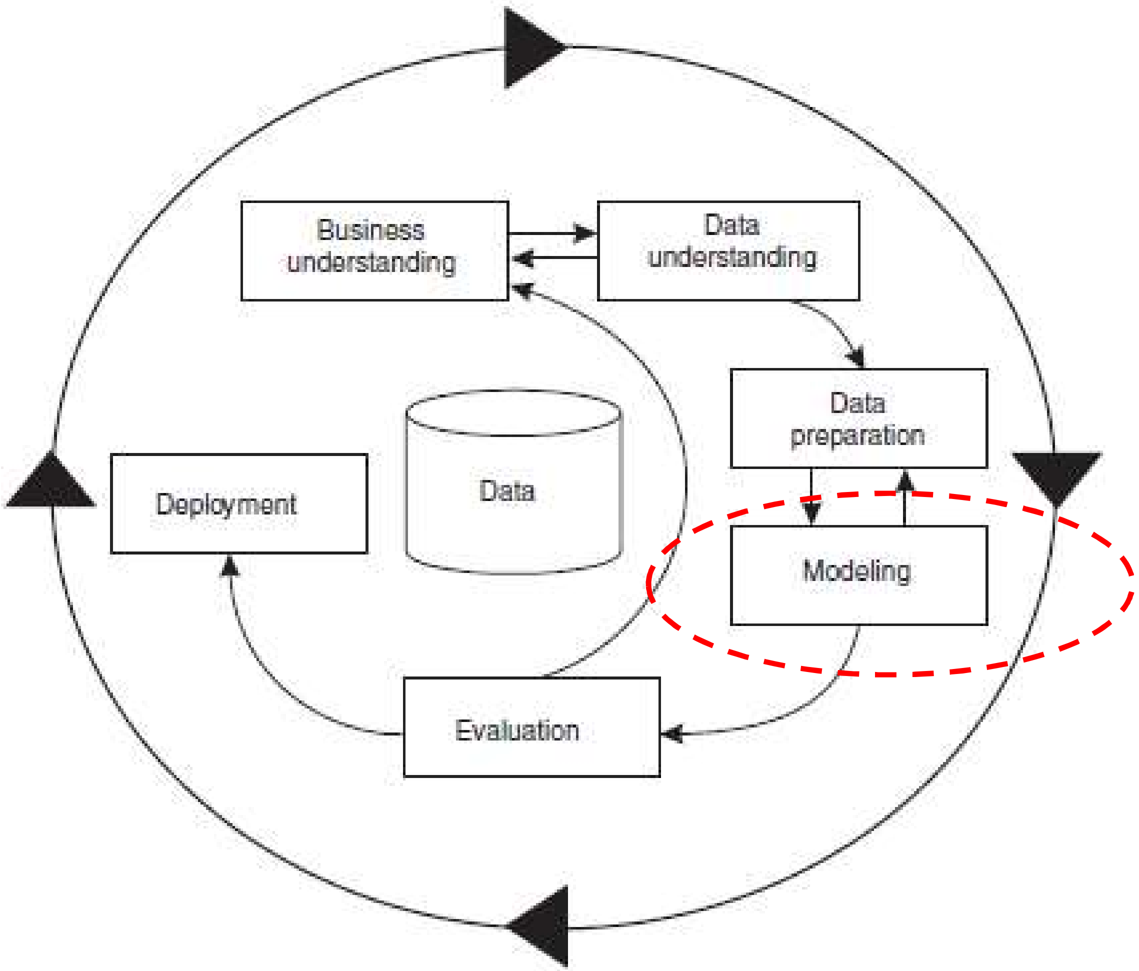
Classification

Prof. Dongping Song
University of Liverpool Management School
Email: Dongping.song@liv.ac.uk

Mainly based on Tan, P.N., Steinbach, M., and Kumar, V. (2014) Introduction to Data Mining, Pearson

The CRISP-DM Process Model



CRISP=Cross-Industry Standard Process

Mariscal et al (2010). A survey of data mining and knowledge discovery process models and methodologies, Knowledge Engineering Review, 25, 137-166.

Example: Questions to Think

- How to help bank to judge loan applications based on data?

		Customer Database			
		Income	Married	Cars	Approved?
Applicants	Beatrice	50K	y	1	Yes
	Dylan	80K	n	2	Yes
	Mathew	30K	n	1	No
	Larry	40K	n	0	No
	Basil	80K	n	1	Yes

- What to do to help Sammy to get loan approval?

Learning Outcomes

- Introduce classification techniques
- Illustrate decision tree classification technique
- Appreciate decision tree induction methods
- Explain Hunt's algorithm
- Determine how to split the records
  - Specify the attribute test condition
  - Impurity measures: Gini, Entropy, Misclassification Error
  - Gain, Information Gain, Gain Ratio
- Determine when to stop splitting
- Discuss practical issues of classification
- Discuss case study

Review of Classification

- 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.

Example: Income, Married, Cars -> Approved?

Purposes of Classification

- In general a classification model can be used for the following **purposes**:
  - It can serve as a **explanatory** tool for distinguishing objects of different classes. This is the **descriptive** element of the classification model
  - It can be used to **predict** the class labels of new records. This is the **predictive** element of the classification model

Each classification technique applies a learning algorithm

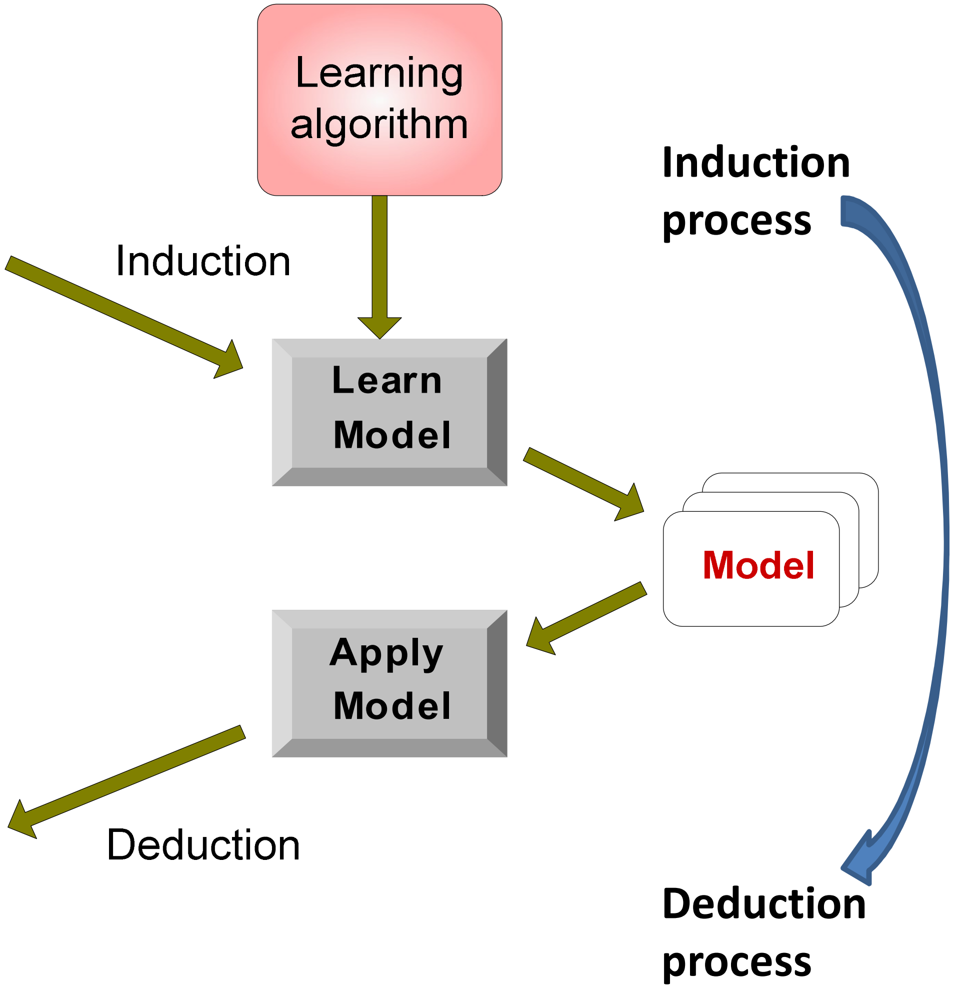
Illustrating Classification Task

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



Rigorous Definition of Classification

Classification – learning a target function *f* that maps each attribute set **x** to one of the predefined class labels *y*.

Input

Attributes set (**x**)

Classification model:
$y = f(\mathbf{x})$

Output

Class label (y)

- Each record is characterized by a tuple (**x**, y);
- Where **x** is the attribute set, or a vector.
- Where y is often termed class label, category or target attribute.
- The attribute set can contain continuous features; while the class label must be a discrete attribute.

Examples of Classification Task

- Predict tumor cells as benign or malignant;
- Classify credit card transactions as legitimate or fraudulent;
- Detect spam email messages;
- Classify secondary structures of protein as alpha-helix, beta-sheet, or random coil;
- Categorize news stories as finance, weather, entertainment, sports, etc.
- A bank loan officer wants to analyse the data in order to know which customer are risky/safe.
- A marketing manager needs to analyse to guess a customer with a given profile will buy a new computer.

Classification Techniques

A classification technique is a **systematic approach** to building classification models from an input data set.

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

The model generated by a learning algorithm should both **fit** the input data well and correctly **predict** the class labels of records it has never seen before.

Decision Tree Based Classification

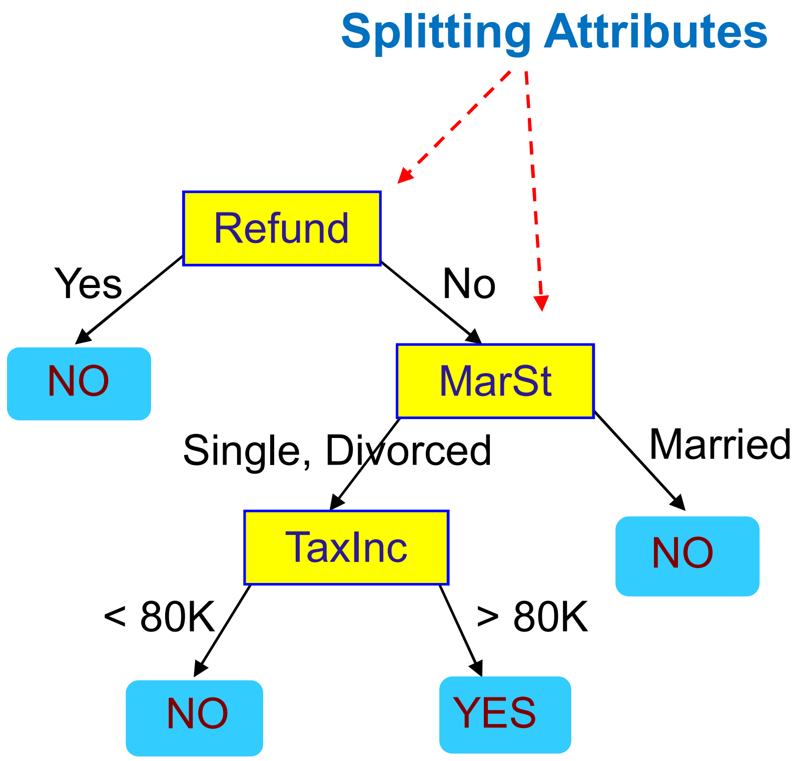
- A decision tree is a hierarchical structure of **nodes** and **directed edges**. There are three types of nodes in a decision tree:
  - A **root node**, which has no incoming edges and zero or more outgoing edges
  - **Internal nodes**, each of which have exactly one incoming edge and two or more outgoing edges
  - **Leaf nodes**, each of which have exactly one incoming edge and no outgoing edges. Each leaf node also has a class label attached to it

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

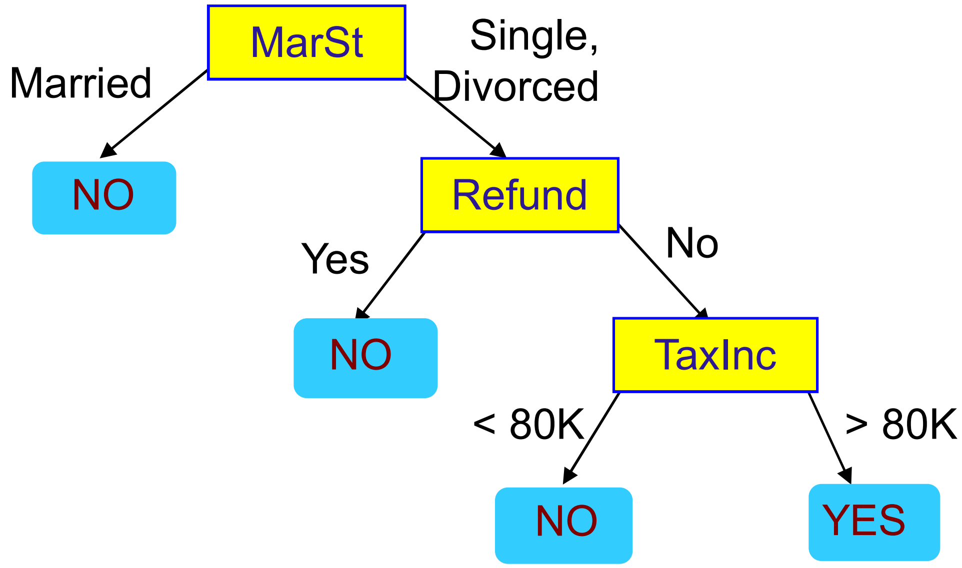


Model: Decision Tree

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

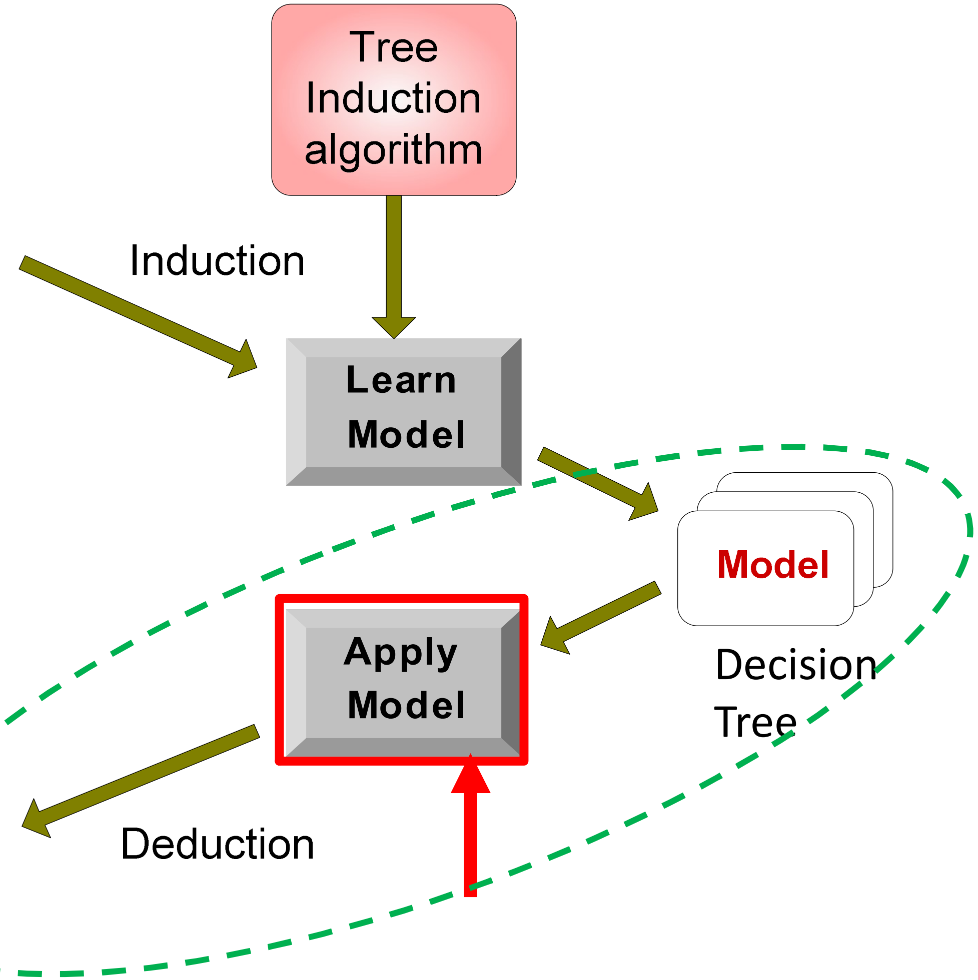


There could be many decision trees that fit the same data!
How many potential decision trees in this example?

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

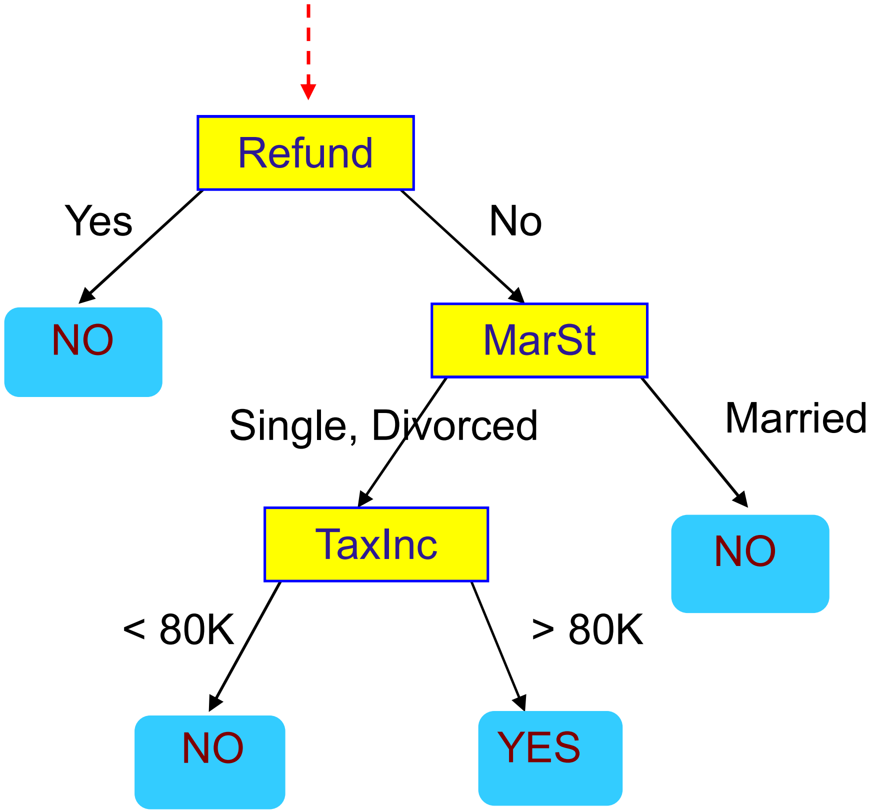


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Apply Model to Test Data

Start from the root of tree.



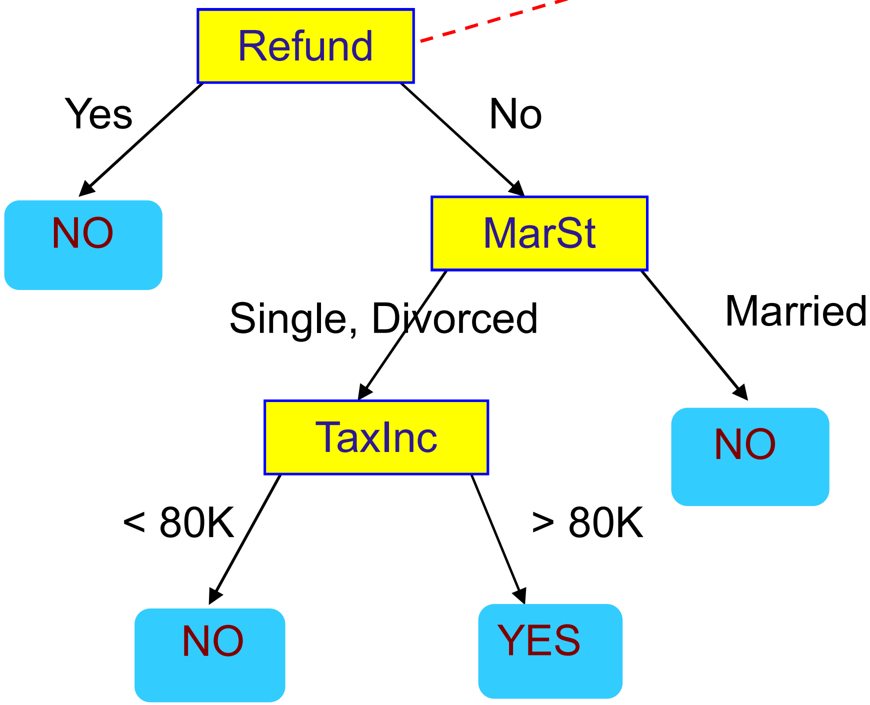
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

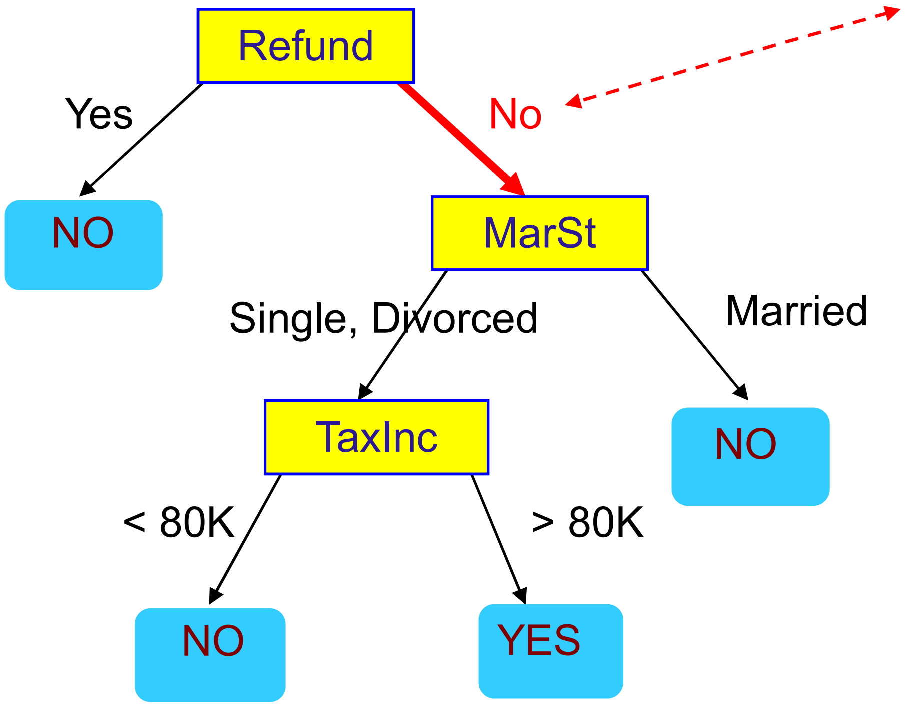
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

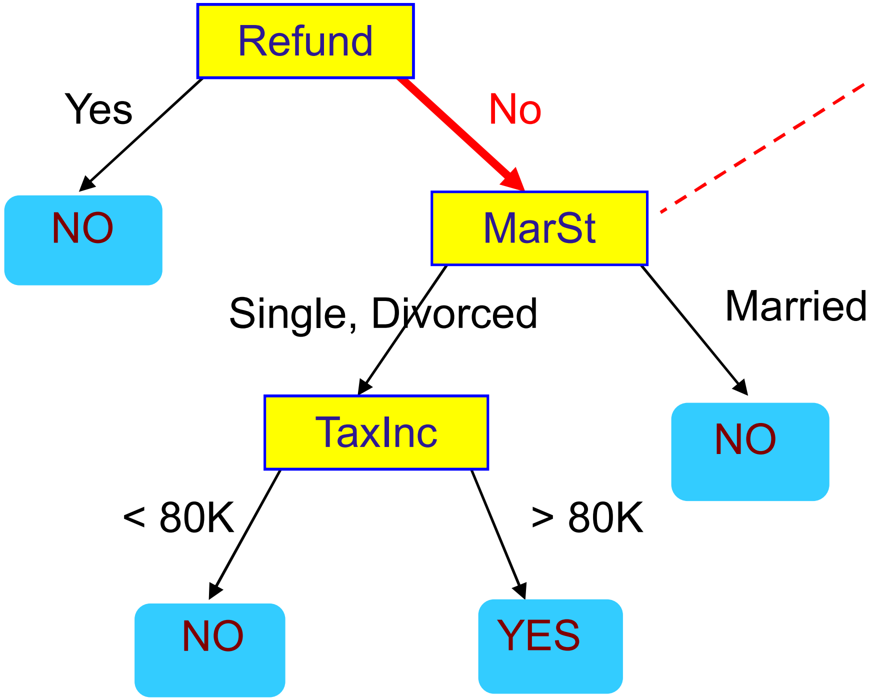
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

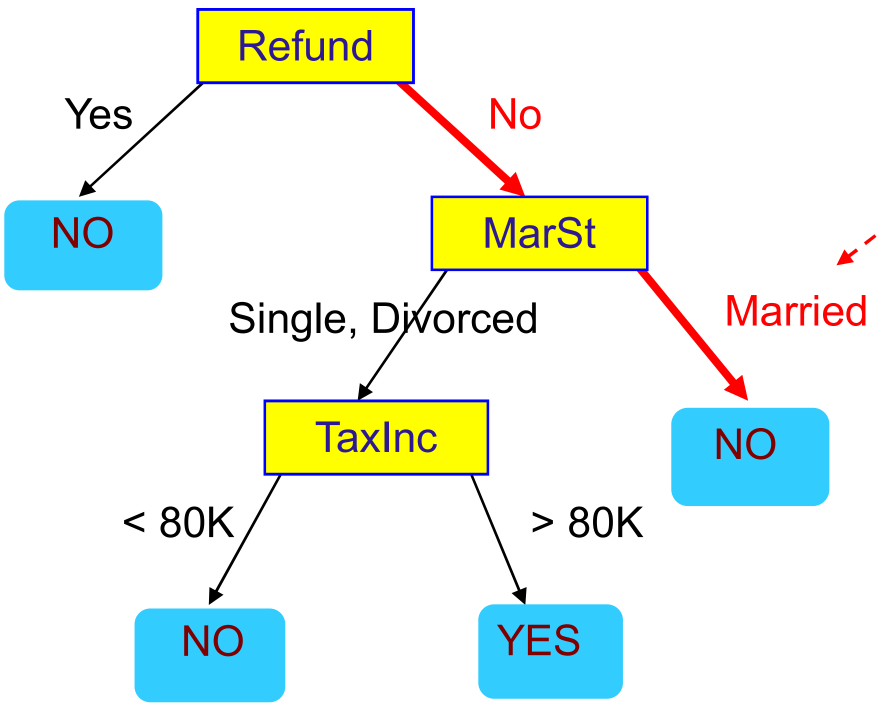
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

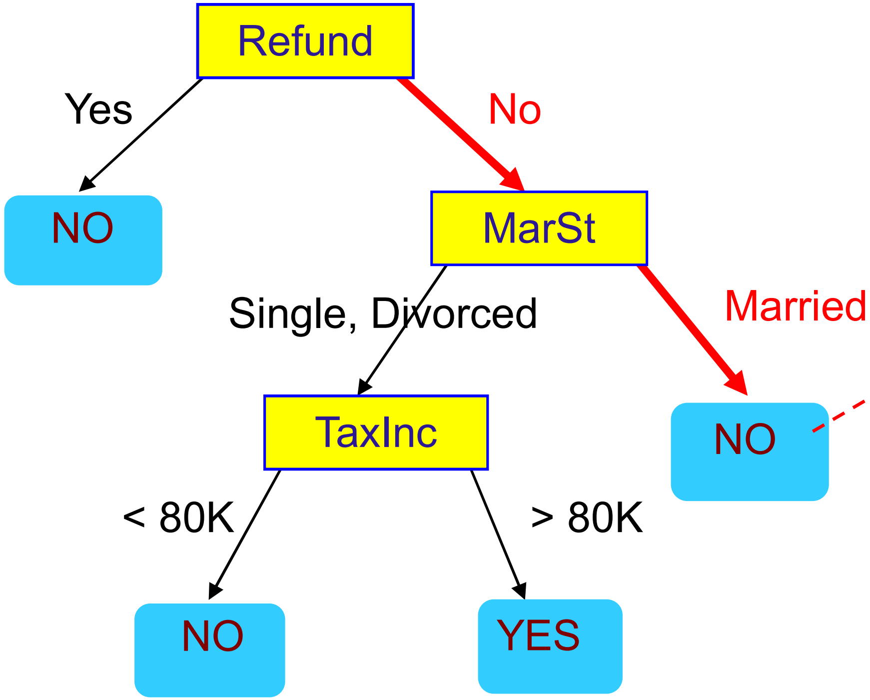
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

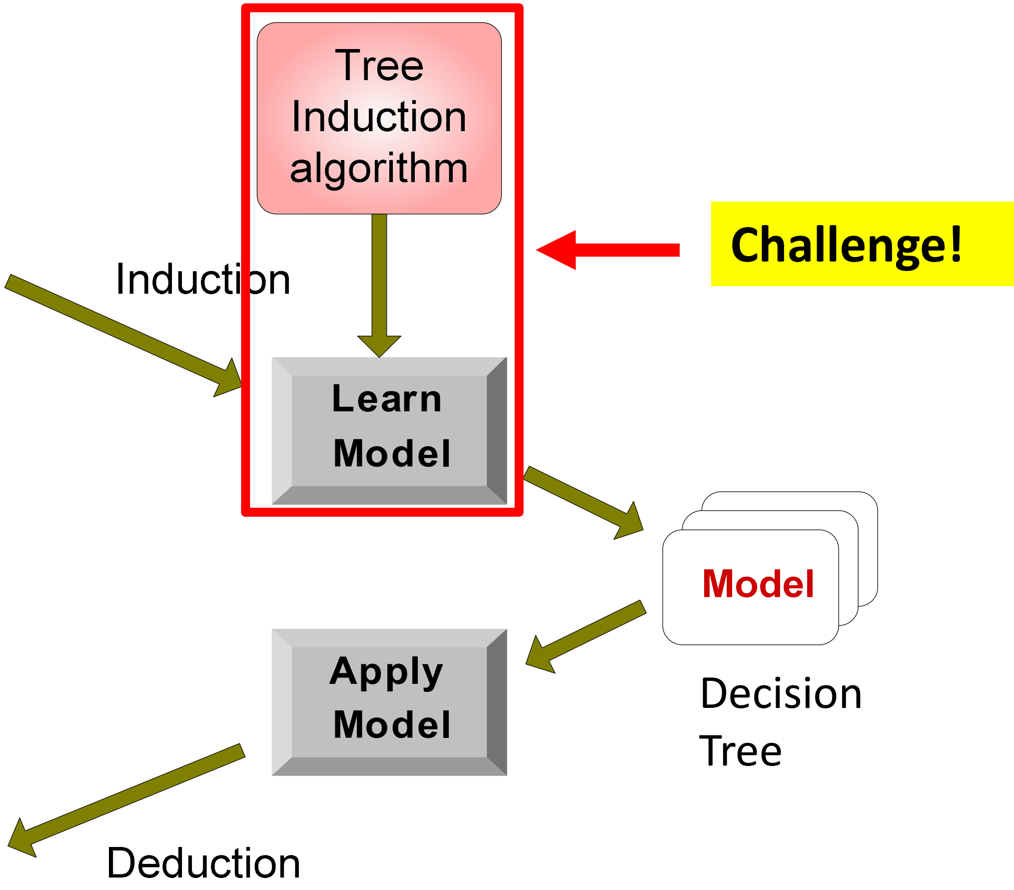


Assign Cheat to “No”

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 Based Classification

- One of the most widely used classification technique
- Highly **expressive** in terms of capturing relationships among discrete variables
- Relatively **inexpensive** to construct and extremely fast at classifying new records
- Easy to interpret
- Can effectively handle both missing values and noisy data
- Comparable or better accuracy than other techniques in many applications

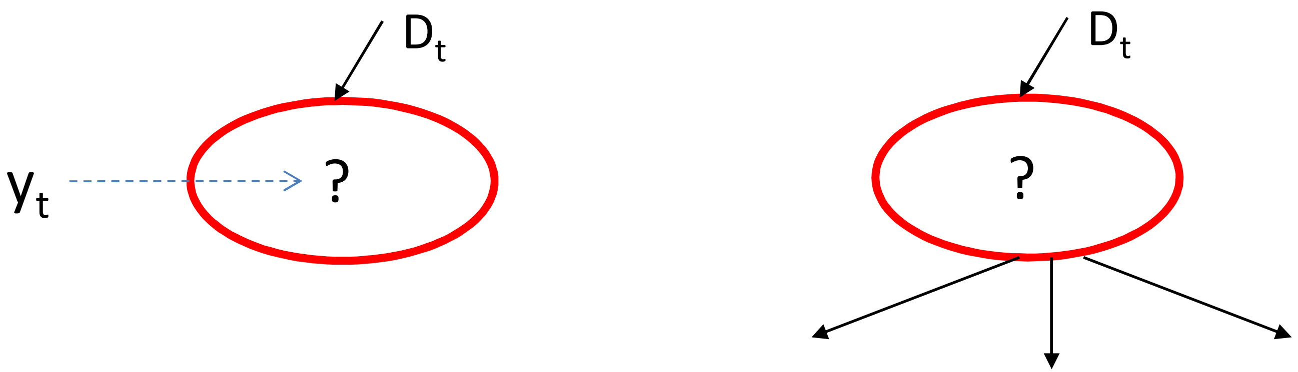
Decision Tree Induction Methods

- **Hunt's Algorithm** (the basis)
- CART (Classification and regression tree):
  - binary tree; a non-parametric decision tree learning technique; GINI impurity;
- ID3 (Iterative Dichotomiser 3)
  - info gain approach is generally used to determine suitable property for each node of a generated decision tree; entropy reduction;
- C4.5
  - Extension of ID3; use info gain as splitting criteria; can handle missing value;

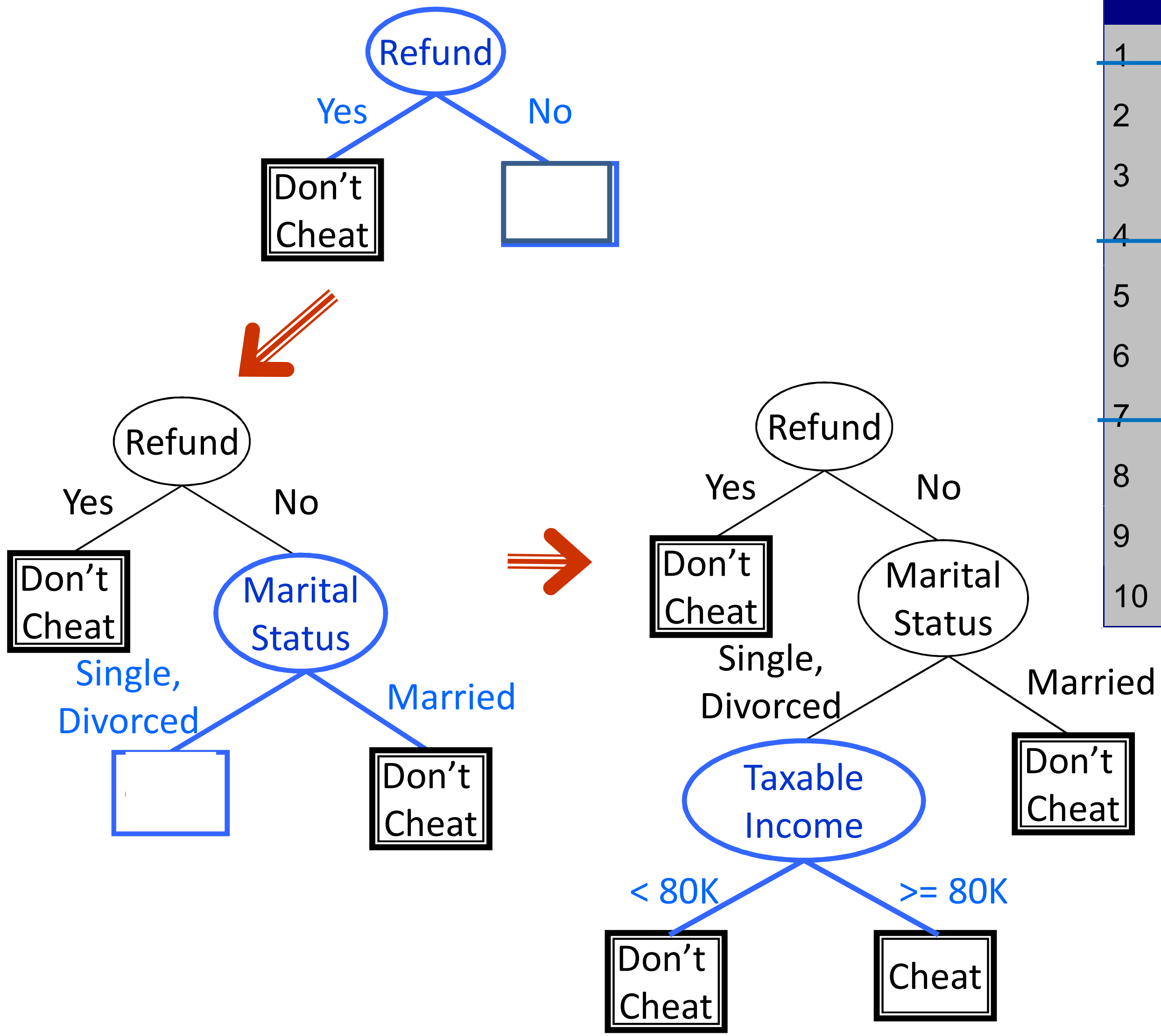
General Structure of Hunt's Algorithm

Recursive Procedure: Let D_t be the set of training records that reach a node t; and y = $\{y_1, y_2, \ldots, y_c\}$ be the class labels.

- **Step 1**: If D_t contains records that belong to the same class y_t, then t is a leaf node labeled as y_t.
- **Step 2**: 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.



Hunt's Algorithm



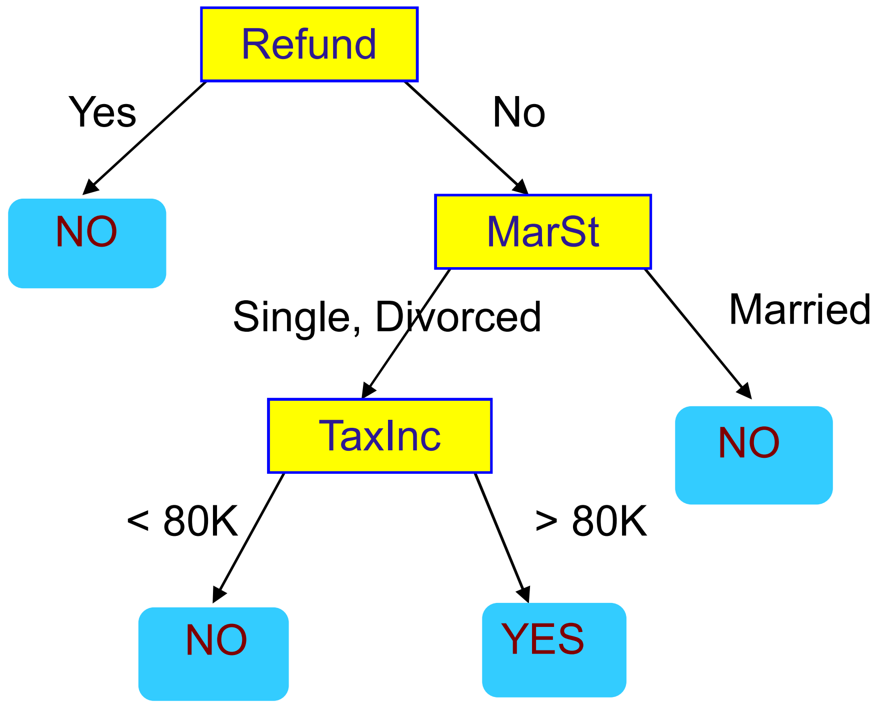
Tid	Refund	Marital Status	Taxable Income	Cheat
~~1~~	~~Yes~~	~~Single~~	~~125K~~	~~No~~
2	No	~~Married~~	~~100K~~	~~No~~
3	No	Single	70K	No
~~4~~	~~Yes~~	~~Married~~	~~120K~~	~~No~~
5	No	Divorced	95K	Yes
6	No	~~Married~~	~~60K~~	~~No~~
~~7~~	~~Yes~~	~~Divorced~~	~~220K~~	~~No~~
8	No	Single	85K	Yes
9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

Notes: Hunt's Algorithm

- Hunt's algorithm will work if **every combination** of attribute values is present in the training data and each combination has **a unique class label**.
- It is possible for some of the child nodes created in Step 2 to be **empty**; In this case the node is declared a leaf node with the same class label as the majority class of training records associated with its parent node.
- In Step 2, if all the records associated with D_t have **identical attribute values** (**except for the class label**), then it is not possible to split these records any further. In this case, the node is declared a leaf node with the same class label as the majority of training records associated with this node.

Comments on Decision Tree

- Viewing decision trees as **segmentation**
- Each segment would be one of the leaves of the tree
- Segmentation of customers, products, and sales regions
- Get a high-level view of a large amount of data



Example: Predict Ship Delay

ID	Speed	GT	Carrier	Delay	ID	Speed	GT	Carrier	Delay
1	high	high	G6	No	20	high	high	O3	No
2	high	high	G6	Yes	21	high	medium	G6	Yes
3	high	high	G6	Yes	22	high	medium	G6	Yes
4	high	high	G6	Yes	23	high	medium	O3	No
5	high	high	G6	Yes	24	high	medium	O3	Yes
6	high	high	G6	Yes	25	low	low	G6	Yes
7	high	high	O3	No	26	low	low	G6	Yes
8	high	high	O3	No	27	low	low	G6	Yes
9	high	high	O3	No	28	low	low	G6	Yes
10	high	high	O3	No	29	low	low	G6	Yes
11	high	high	O3	No	30	low	low	G6	Yes
12	high	high	O3	No	31	low	low	G6	Yes
13	high	high	O3	No	32	medium	high	G6	Yes
14	high	high	O3	No	33	medium	low	G6	Yes
15	high	high	O3	No	34	medium	medium	G6	Yes
16	high	high	O3	No	35	medium	medium	G6	Yes
17	high	high	O3	No	36	medium	medium	G6	Yes
18	high	high	O3	No	37	medium	medium	O3	Yes
19	high	high	O3	No	38	medium	medium	O3	Yes

Discuss: Predict Ship Delay

Exercise: “Good day for tennis”

Day	Outlook	Humid	Wind	PlayTennis?
d1	s	h	w	n
d2	s	h	s	n
d3	o	h	w	y
d4	r	h	w	y
d5	r	n	w	y
d6	r	n	s	y
d7	o	n	s	y
d8	s	h	w	n
d9	s	n	w	y
d10	r	n	w	y
d11	s	n	s	y
d12	o	h	s	y
d13	o	n	w	y
d14	r	h	s	n

- Outlook = sunny, overcast, rain
- Humidity = high, normal
- Wind = weak, strong

Build a decision tree starting with the Outlook attribute.

Discuss: “Good day for tennis”

Design Issues of 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

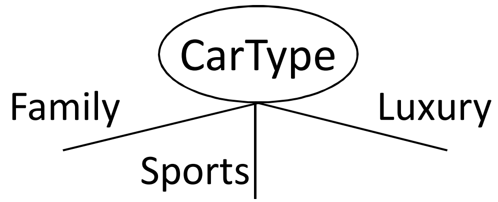
How to Specify Test Condition?

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

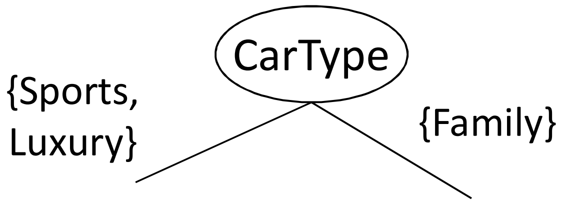
A binary attribute can generate two potential outcomes.
Other attributes can also be split in a multi-way.

Splitting Based on Nominal Attributes

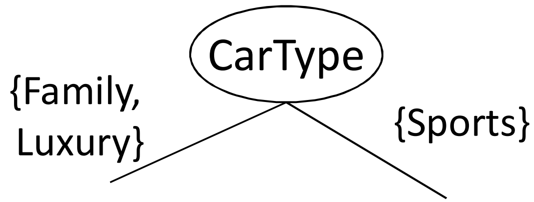
- **Multi-way split:** Use as many partitions as distinct values.



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



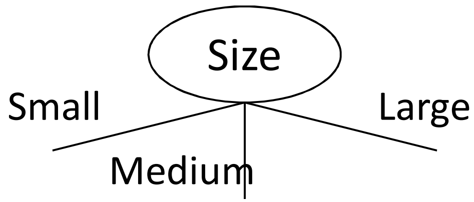
OR



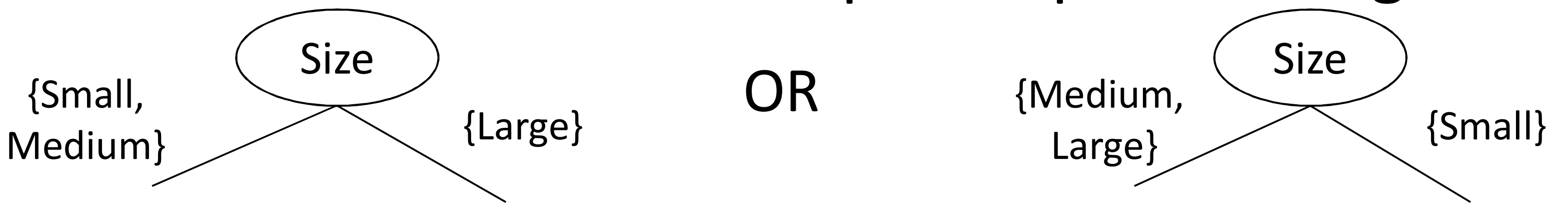
If the attribute has k distinct values, then there are $2^{k-1} - 1$ ways to split.

Splitting Based on Ordinal Attributes

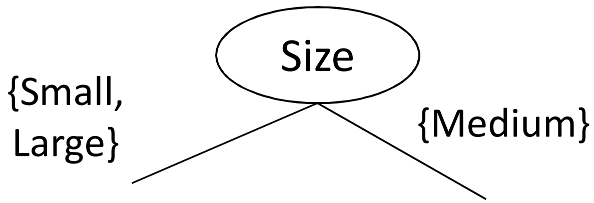
- **Multi-way split:** Use as many partitions as distinct values.



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



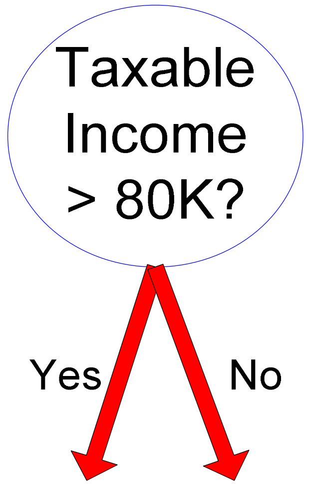
What about this split?



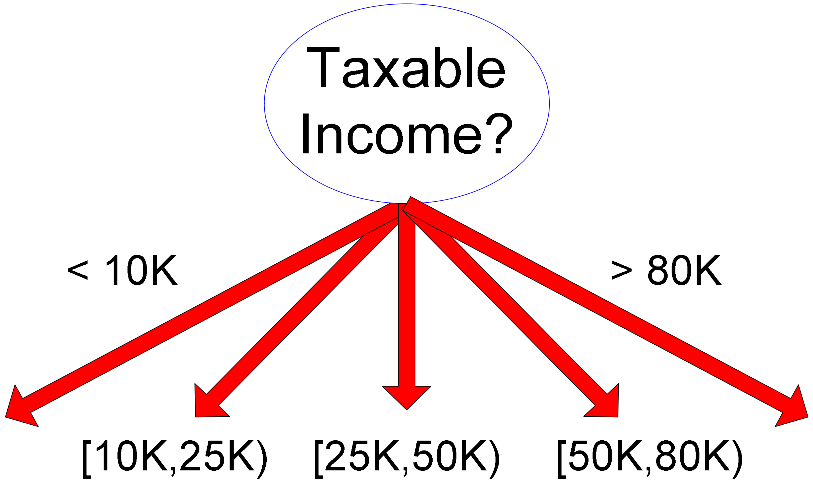
Splitting Based on Continuous Attributes

- 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 computation intensive

Splitting Based on Continuous Attributes



(i) Binary split



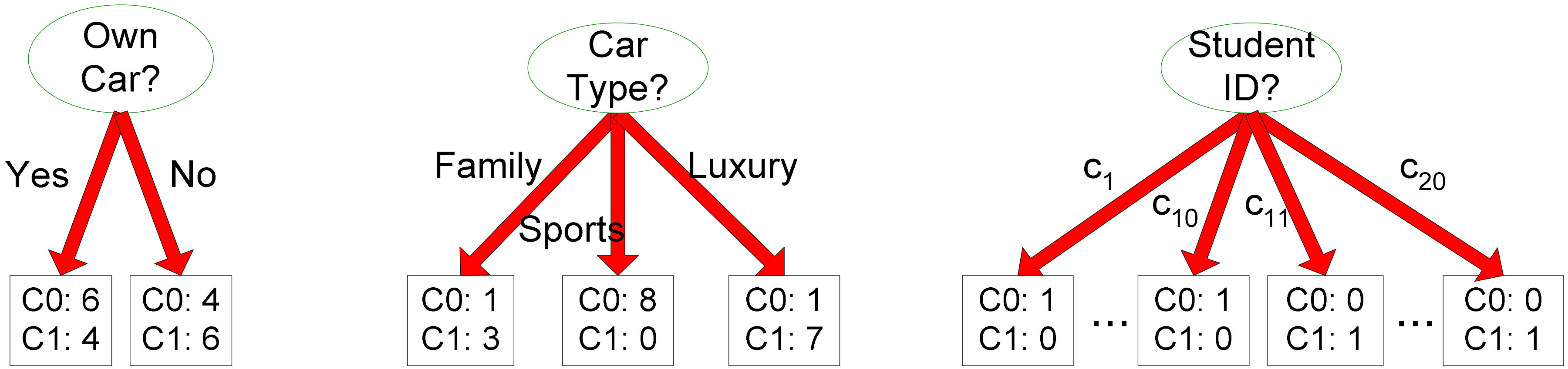
(ii) Multi-way split

Tree Induction

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

How to determine the Best Split

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



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with **homogeneous** class distribution are preferred
- Need a measure of node impurity:

C0: 5	C0: 9
C1: 5	C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

The smaller degree of impurity the better.

Splitting Criterion

- There are many test conditions one could apply to partition a collection of records into smaller subsets
- Various **measures** are available to determine which test condition provides the best split
  - Gini Index
  - Entropy / Information Gain
  - Classification Error

Measures of Node Impurity

- Gini Index

$$Gini\ Index = 1 - \sum_j p_j^2$$

- Entropy

$$Entropy = \sum_j -p_j log_2 p_j$$

- Classification error

$$Classification\ Error = 1 - max\{p_j\}$$

Note: p_j is the relative frequency of class j.

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) 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

C1	**3**
C2	**3**

P(C1) = ?   P(C2) = ?
Gini = ?

Procedure of Computing GINI

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

Step 1: count the number of objects that belong to **node t**;

Step 2: among those objects, count the number of objects for **each class j**;

Step 3: calculate the **relative frequency** of class j at node t, i.e. p(j | t);

Step 4: calculate **Gini(t)** according to the formula.

Splitting Based on GINI

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

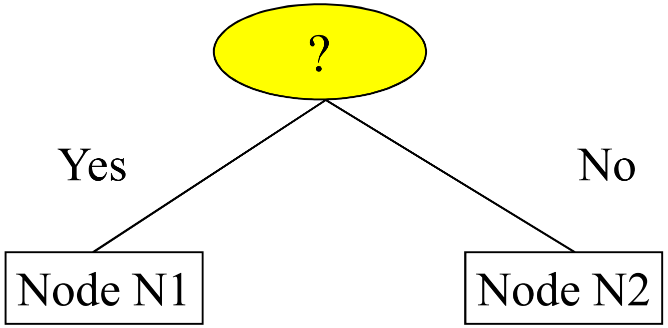
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node t.

Greedy strategy: the **lowest Gini split** should be selected.

Binary Attributes: Computing GINI Index



Node N1:

C1	5
C2	2

Node N2:

C1	1
C2	4

	Parent
C1	**6**
C2	**6**
Gini = 0.500	

Gini(N1)
$= 1 – (5/7)^2 – (2/7)^2$
$= 0.408$

Gini(N2)
$= 1 – (1/5)^2 – (4/5)^2$
$= 0.320$

Gini(split) = Gini(Children)
$= 7/12*0.408 + 5/12*0.320$
= **0.371**

Categorical Attributes: Computing Gini Index

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

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

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		

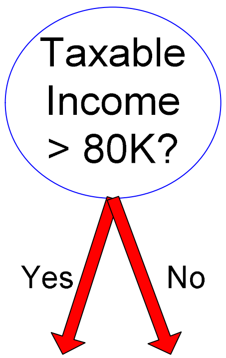
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

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

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



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

Sorted Values →

Split Positions →

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	

Gini(split at 97) =

Exercise: Gini Index

ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Exercise: Gini Index

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

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

Measure of Impurity: ENTROPY

- Entropy at a given node t:

$$Entropy(t) = -\sum_{j} p(j \mid t) \log_2 p(j \mid t)$$

(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

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

C1	**0**
C2	**6**

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

Entropy = – 0 $\log_2$ 0 – 1 $\log_2$ 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) $\log_2$ (5/6) = 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

C1	3
C2	3

P(C1) = ?   P(C2) = ?

Entropy = ?

Measure of Impurity: 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 Classification 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 = 0.167

C1	**2**
C2	**4**

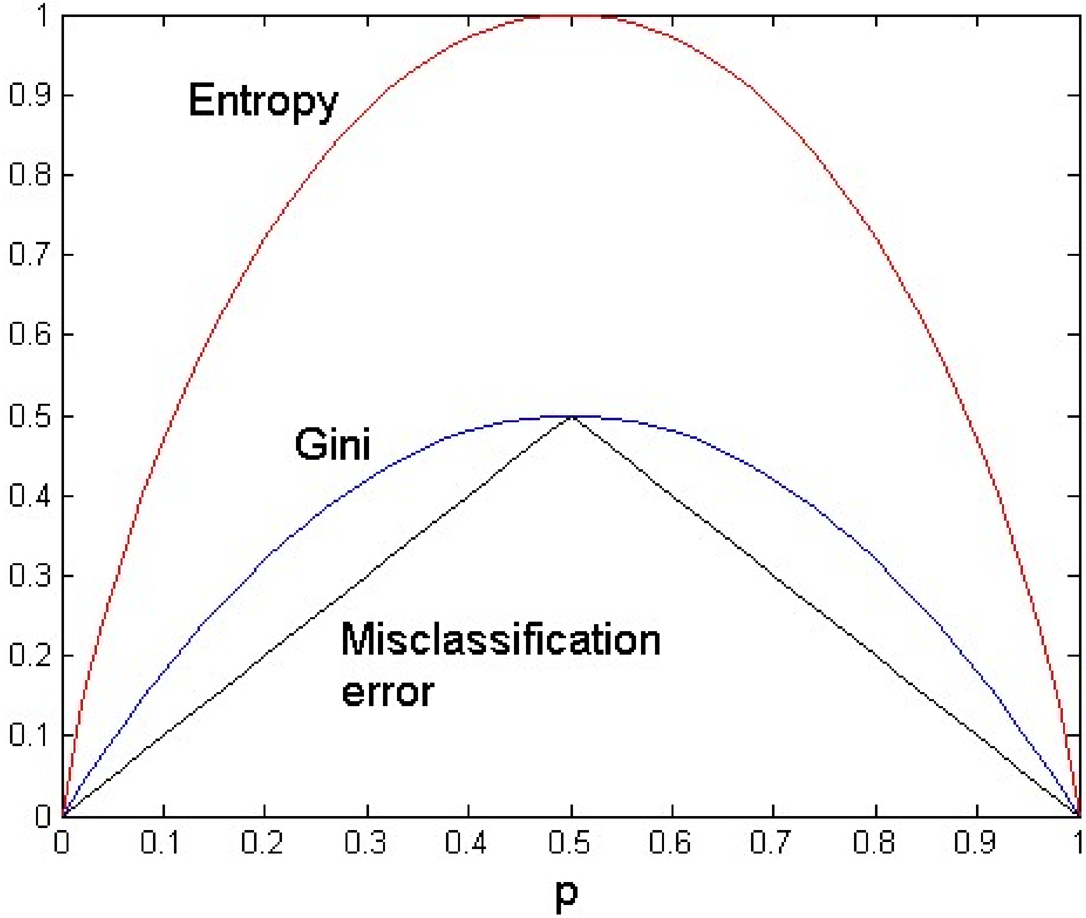
P(C1) = 2/6   P(C2) = 4/6
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 0.333

C1	3
C2	3

P(C1) = ?   P(C2) = ?
Error = ?

Comparison Among Splitting Criteria

For a **2-class** problem:



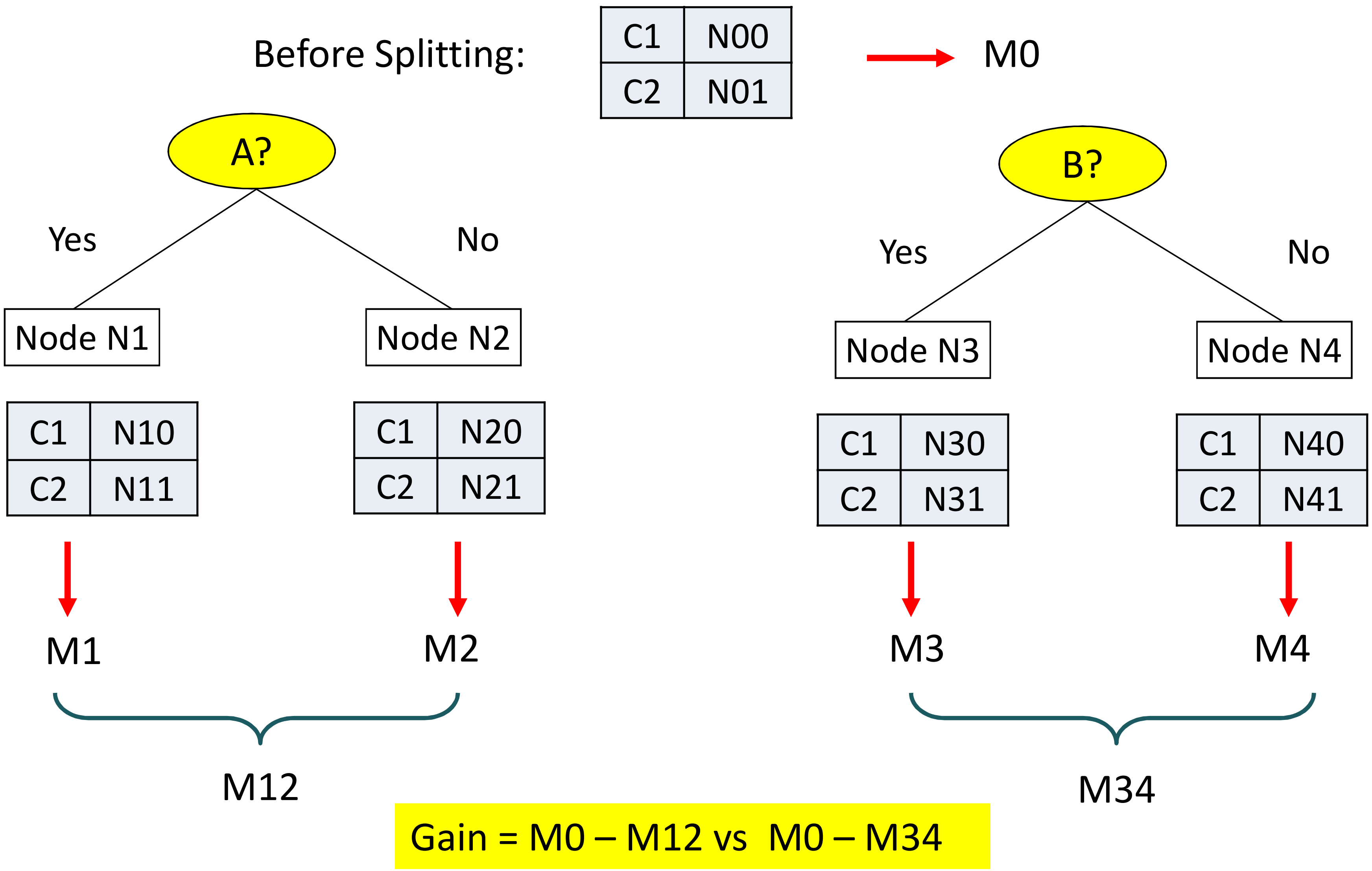
Gain and Information Gain

- To determine how well a test condition performs, we need to compare the **degree of impurity** of the parent node (before splitting) with the degree of impurity of the child nodes (after splitting).
- The **larger** their difference, the better the test condition.
- The difference is defined as gain (called **Info Gain** when entropy measure is used):

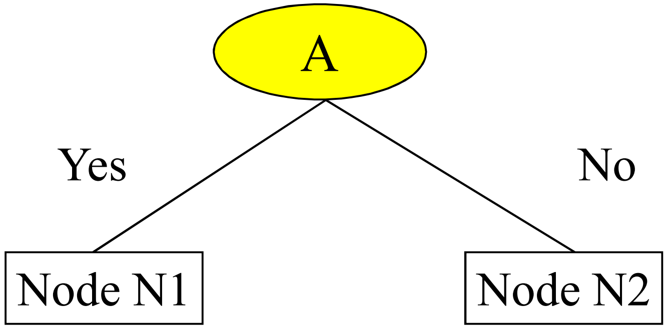
$$\boxed{GAIN_{split} = I(parent) - \left(\sum_{j=1}^{k} \frac{n_j}{n} I(j) \right)}$$

- *I*(.) is the impurity measure of a given node;
- Parent Node is split into k partitions;
- n_j is number of records in partition j.

How to Find the Best Split: Principle



Binary Attribute Example: Computing GINI for (N1, N2)



C1	5
C2	2

C1	1
C2	4

	Parent
C1	**6**
C2	**6**
Gini = 0.500	

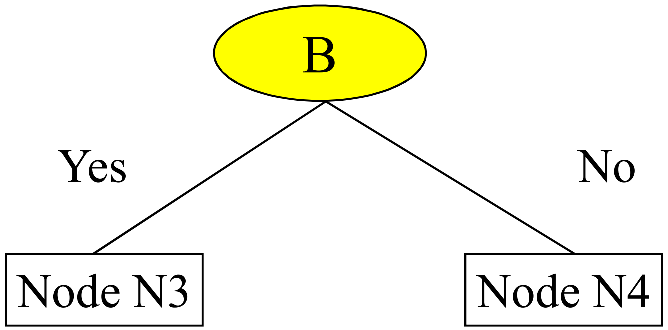
Gini(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.408$

Gini(N2)
$= 1 - (1/5)^2 - (4/5)^2$
$= 0.320$

Gini(Children)
$= 7/12 * 0.408 +$
$5/12 * 0.320$
$=$ **0.371**

Gain(split) = Gini(parent) – Gini(Children)
= 0.5 – 0.371 = **0.129**

Binary Attribute Example: Computing GINI for (N3, N4)



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

Node N3

C1	6
C2	2

Node N4

C1	0
C2	4

Gini(N3)
$= 1 – (6/8)^2 – (2/8)^2$
$= 0.375$

Gini(N4)
$= 1 – (0/4)^2 – (4/4)^2$
$= 0$

Gini(Children)
= 8/12 * 0.375 +
4/12 * 0.0
= **0.250**

Gain(split) = Gini(parent) – Gini(Children)
= ?

Procedure of Computing Gain

Use **Gini index** as the measure of impurity. Assume we are at parent node t:

Step 1: Calculate the Gini index for **parent node t**;
Step 2: Select **one split option**, which generates a few child nodes;
Step 3: Calculate the Gini index for **each child node**;
Step 4: Calculate the Gini index for **the split**;
Step 5: Calculate the **Gain** for the split;
Step 6: Select **another split option** and repeat Step 2~5;
Step 7: Among all the split options, find the split with the **greatest Gain(split)**.

Gain Ratio

- Impurity measures tend to favor attributes with a large number of distinct values, leading to **small** but **pure** partitions.
- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

  – Adjusts the Info Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
  – Used in C4.5

Exercise: Classification Error Split
(CE = 1 – max{p_i})

The following table summarizes a data set with three attributes A, B, C and two class labels +, −. Build a **two-level** decision tree.

			Number of Instances	
A	**B**	**C**	**+**	**–**
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

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 Induction Algorithms: C4.5 & C5.0

- C4.5 is often referred as a statistical classifier.
  - Simple depth-first construction.
  - Uses Information Gain
  - Sorts Continuous Attributes at each node.
  - Needs entire data to fit in memory.
  - Unsuitable for Large Datasets.
- C5.0 is an extension of C4.5
  - can apply in big data set.
  - easily handle the multi-value attribute and missing attribute from data set.

Comparison of Different Decision Tree Algorithms

	ID3	C4.5	CART
Type of data	Categorical	Continuous and Categorical	continuous and nominal data
Speed	Low	Faster than ID3	Average
Pruning	Pre-pruning	Pre-pruning	Post pruning
Procedure	Top-down	Top-down	Construct binary decision tree
Missing Values	Can't deal with	Can deal with	Can deal with
Formula	Use info entropy and info Gain	Use split info and gain ratio	Use Gini diversity index

Decision Tree Based Classification

- A non-parametric approach; 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

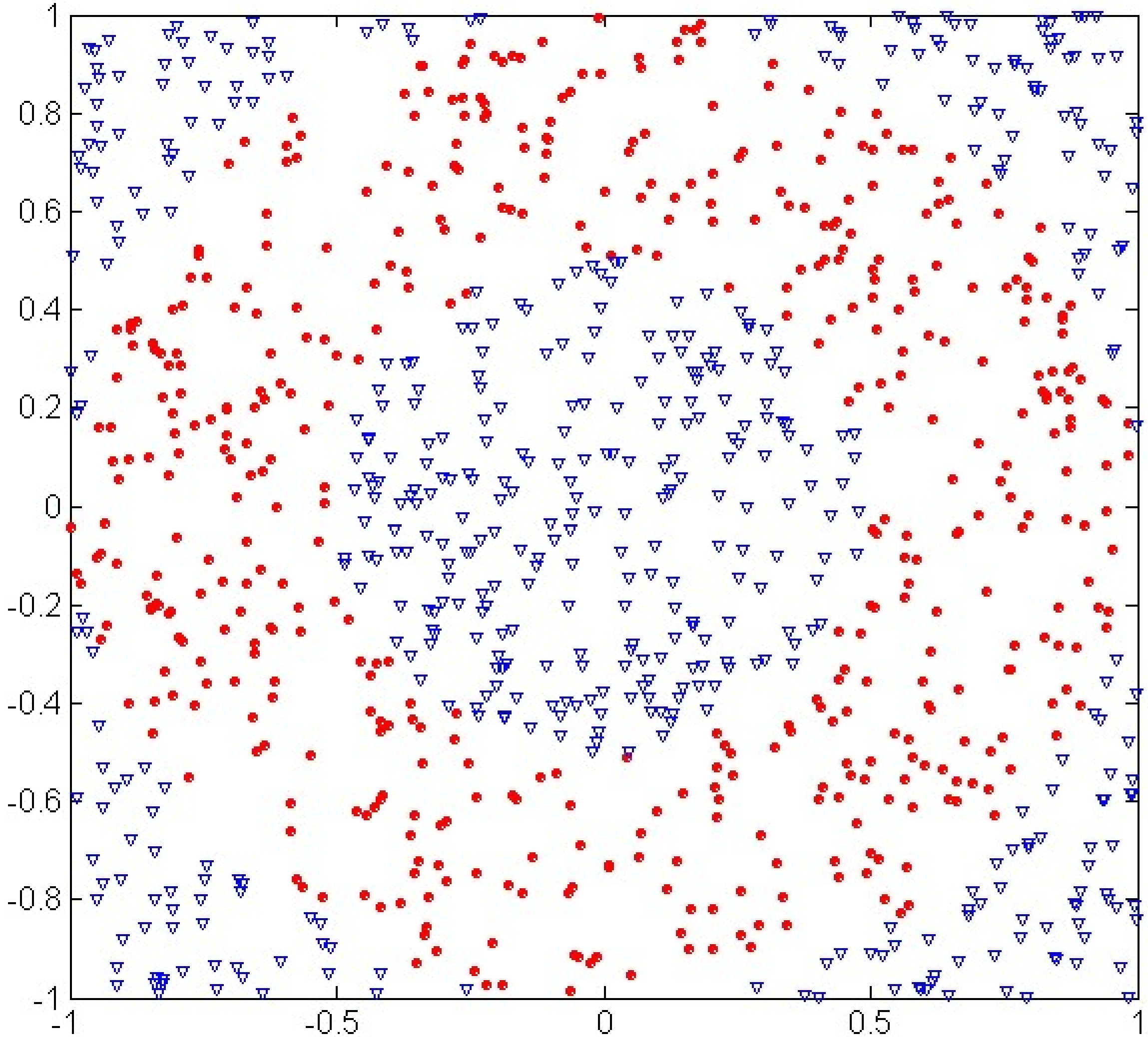
- Finding an optimal decision tree is an NP-complete problem
- Not effective at approximating linear or smooth functions or boundaries
- Trees always include high-order interactions
- Large variance: each split is conditional on all of its ancestor splits.

Practical Issues of Classification

- Underfitting
- Overfitting
- Noise
- Generalization Error
- Occam's Razor
- Pruning
- Missing values

If a decision tree is fully grown, it may lose some generalization capability. This is a phenomenon known as **overfitting**.

Underfitting and Overfitting by Examples



500 circular and 500 triangular data points.

Circular points:

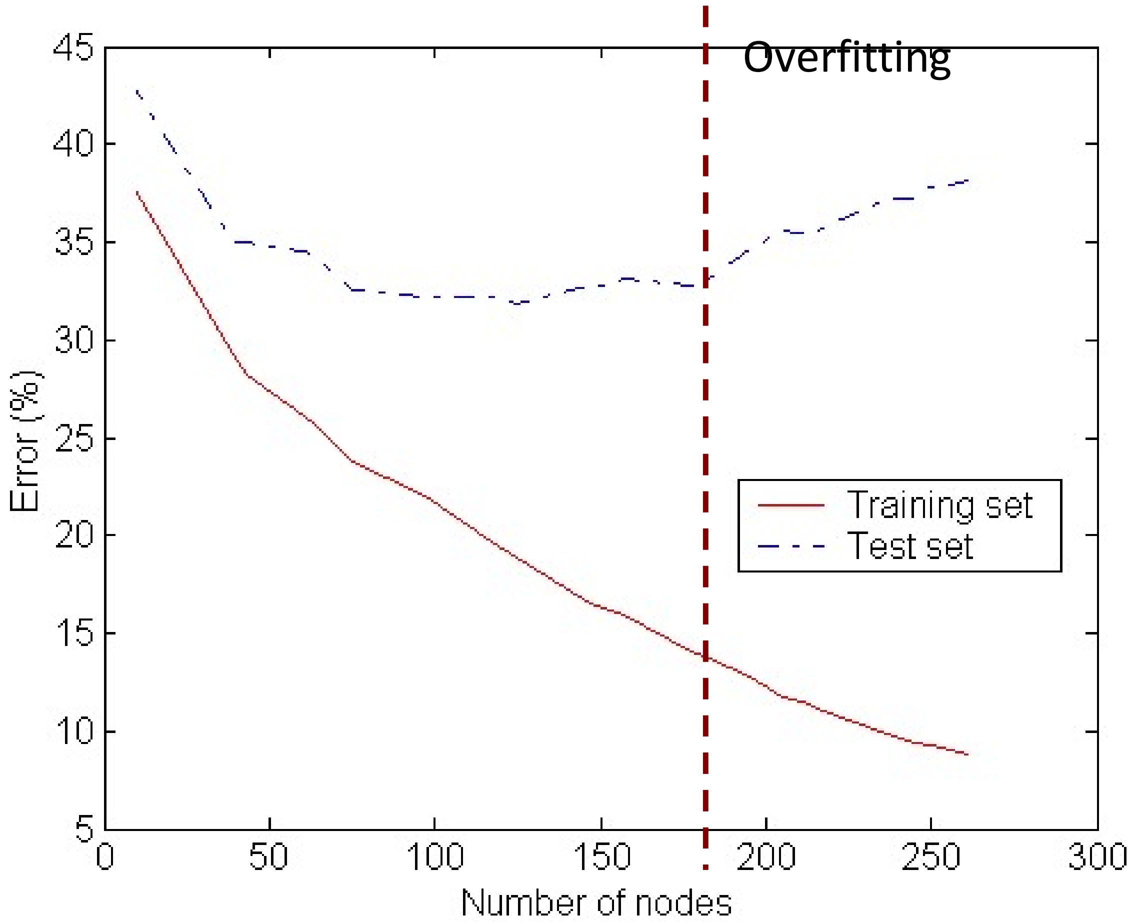
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

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

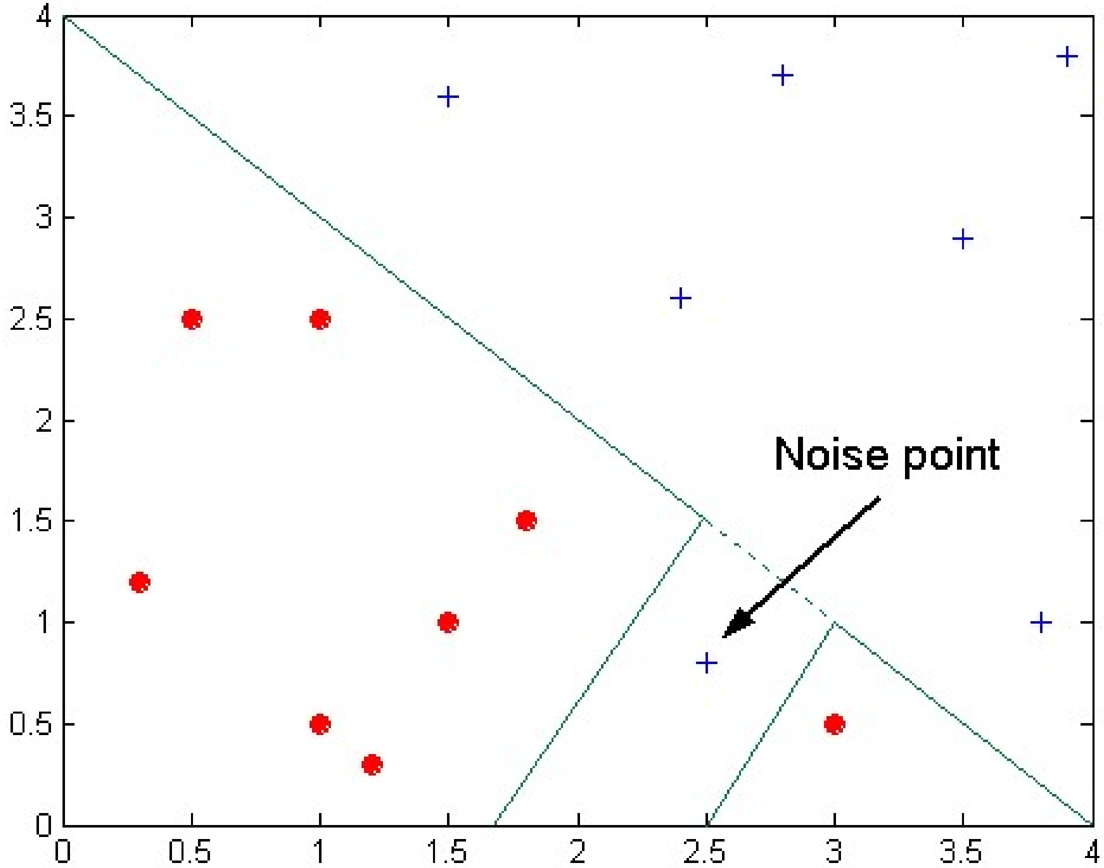
$\text{sqrt}(x_1^2+x_2^2) > 1$

Underfitting and Overfitting



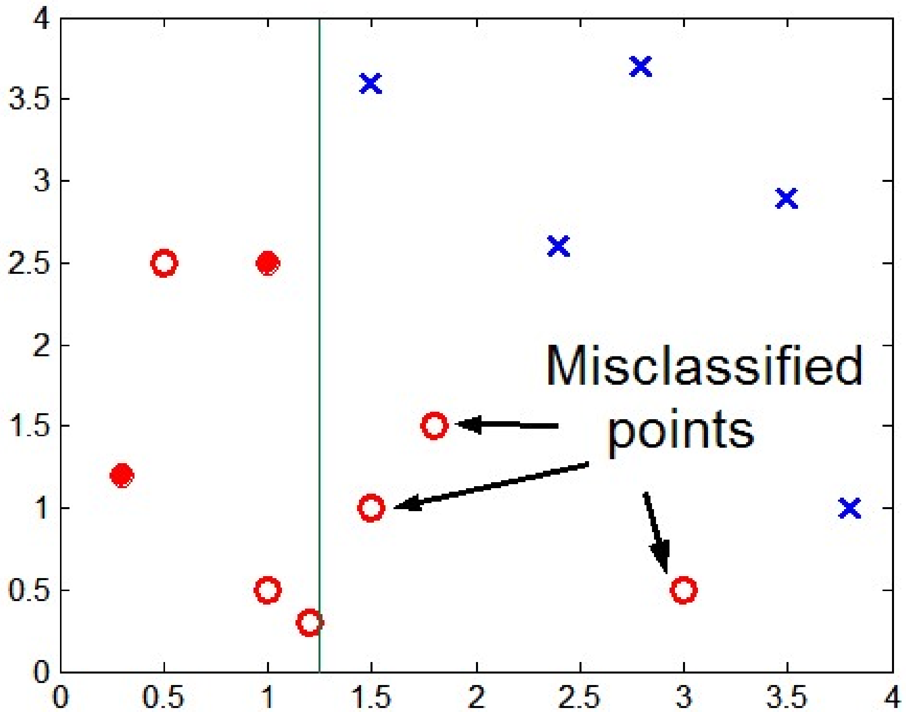
Underfitting: when model is too simple, both training and test errors are large

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

Overfitting Due to Noise: An Example

Training Set to classify mammals

Name	Body Temp.	Give Birth	4-legged	Hibernates	Class Label
Porcupine	Warm-bld	Yes	Yes	Yes	**Yes**
Cat	Warm-bld	Yes	Yes	No	**Yes**
Bat	Warm-bld	Yes	No	Yes	**No***
Whale	Warm-bld	Yes	No	No	**No***
Salamander	Cold-bld	No	Yes	Yes	**No**
Komodo dragon	Cold-bld	No	Yes	No	**No**
Python	Cold-bld	No	No	Yes	**No**
Salmon	Cold-bld	No	No	No	**No**
Eagle	Warm-bld	No	No	No	**No**
Guppy	Cold-bld	Yes	No	No	**No**

Note: Asterisks denote mislabelings (noise)

Overfitting Due to Noise: An Example

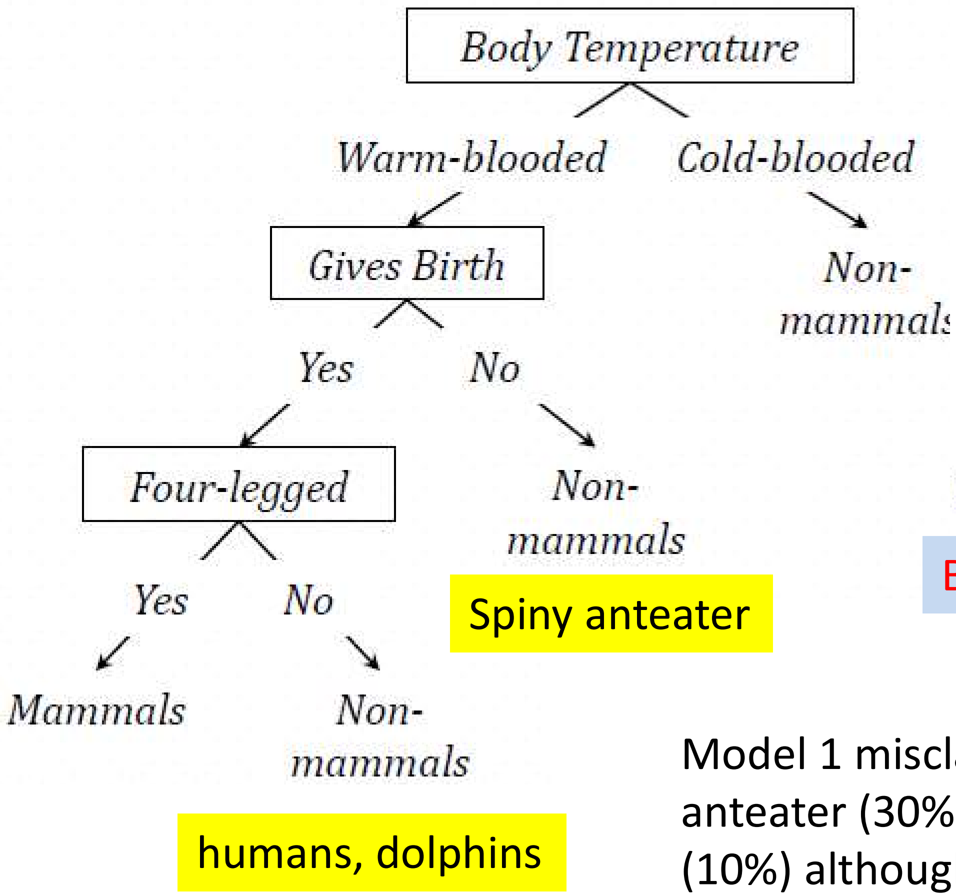
Test Set to classify mammals

Name	Body Temp.	Gives Birth	Four-legged	Hibernates	Class Label
Human	Warm-bld	Yes	No	No	
Pigeon	Warm-bld	No	No	No	
Elephant	Warm-bld	Yes	Yes	No	
Leopard shark	Cold-bld	Yes	No	No	
Turtle	Cold-bld	No	Yes	No	
Penguin	Cold-bld	No	No	No	
Eel	Cold-bld	No	No	No	
Dolphin	Warm-bld	Yes	No	No	
Spiny anteater	Warm-bld	No	Yes	Yes	
Gila monster	Cold-bld	No	Yes	Yes	

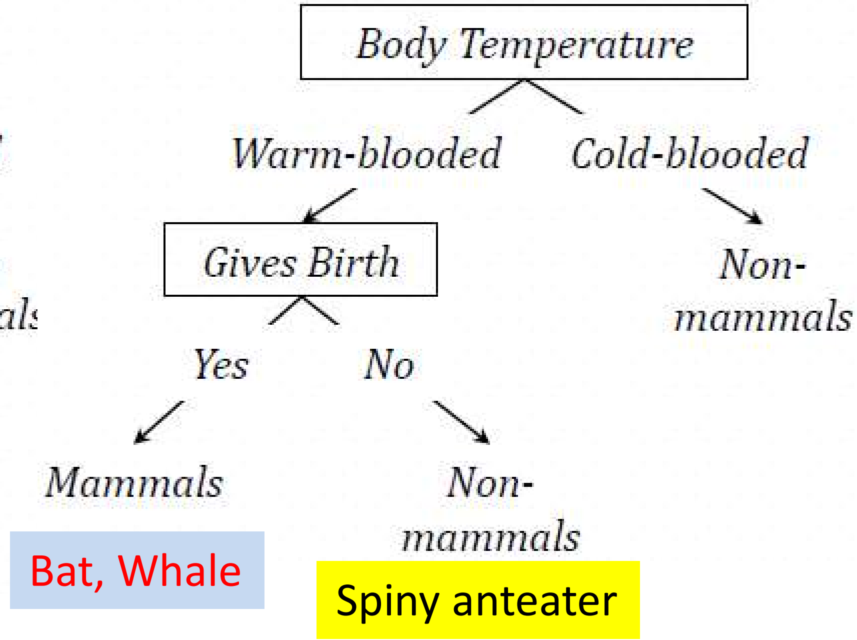
To determine

Overfitting Due to Noise

Model 1



Model 2

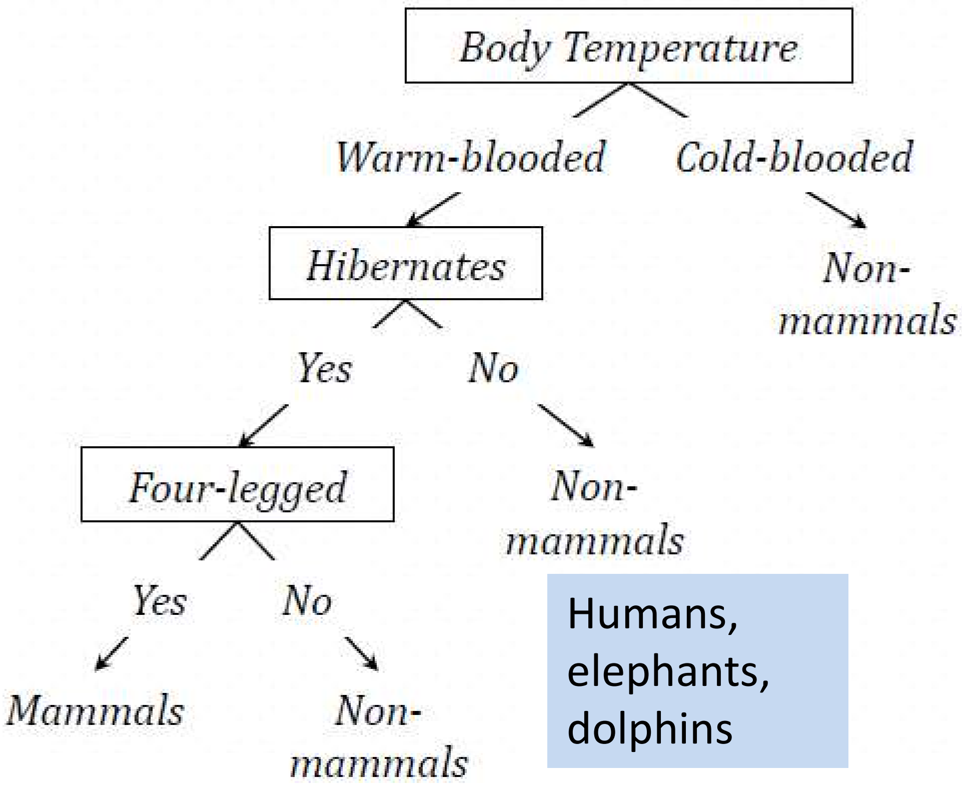


Model 1 misclassifies humans, dolphins, spiny anteater (30%). Model 2 has a lower test error rate (10%) although its training error rate is higher (20%).

Overfitting due to Lack of Samples: An Example **Training Set**

Name	Body Temp.	Gives Birth	4-legged	Hibernates	Class Label
Salamander	Cold-bld	No	Yes	Yes	**No**
Guppy	Cold-bld	Yes	No	No	**No**
Eagle	Warm-bld	No	No	No	**No**
Poorwill	Warm-bld	No	No	Yes	**No**
Platypus	Warm-bld	No	Yes	Yes	**Yes**

Overfitting due to Lack of Samples



Although the model's training error is zero, its error rate on the test set is 30%.

Humans, elephants, and dolphins are misclassified because the decision tree classifies all warm-blooded vertebrates that do not hibernate as non-mammals. The tree arrives at this classification decision because there is **only one training records**, which is an eagle, with such characteristics.

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- When there is noise, training error no longer provides a good estimate of how well the tree will perform on previously unseen records.

- Need new ways for estimating errors

On overfitting: You can't learn anything without inductive bias

Estimating Generalization Errors

- **Re-substitution errors:** error on training (Σ e(t))
- **Generalization errors:** error on testing (Σ 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

Penalty for each node

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) Principle

- A formalization of Occam's razor.

The idea: The best hypothesis (a model and its parameters) for a given set of data is the one that leads to the best compression of the data.

- **Minimum Description Length** (MDL): prefer the hypothesis h that minimizes the space required to describe a theory plus the space required to describe the theory's mistakes.

From Theory to Practice

Let's look at how to turn these ideas of model selection criteria into practice

Decision Tree Pruning Methodologies

- **Pre-pruning** (top-down)
  - Stopping criteria while growing the tree
- **Post-pruning** (bottom-up)
  - Grow the tree, then prune
  - More popular

Key point: do not overfit the train data

Avoiding Overfitting in Decision Trees

- Stop growing the tree when the data split is not statistically significant
- Grow the full tree, then prune
  - Do we really needs all the “small” leaves with perfect coverage?

- How to select (MDL principle)
  - Measure performance over training data (and include some estimates for generalization)
  - Measure performance over separate validation data
  - Use Minimum Description Length Principle (MDL) to minimize:

$$\boxed{\text{size(tree)} + \text{size(misclassification(tree))}}$$

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 χ^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

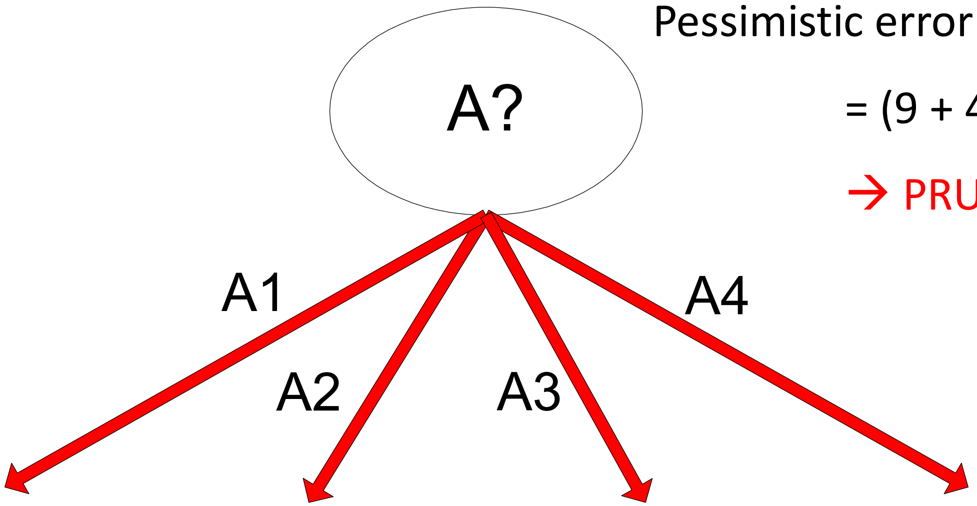
Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

→ PRUNE!

Assume one error is reduced by the split.



Class = Yes	8	Class = Yes	3	Class = Yes	4	Class = Yes	5
Class = No	4	Class = No	4	Class = No	1	Class = No	1

Examples of Post-pruning

Assume one error is reduced by each split.

– Optimistic error?

Don’t prune for both cases

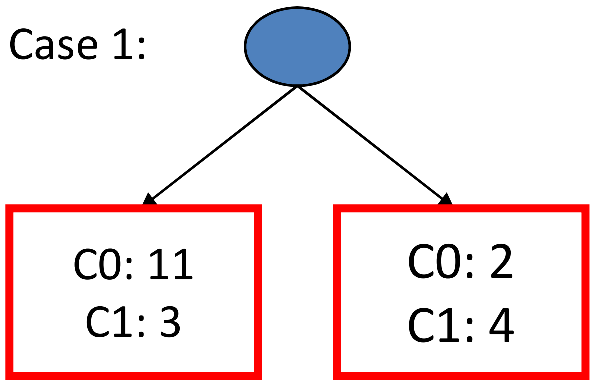
– Pessimistic error?

Don’t prune case 1, prune case 2 if penalty for each node is 0.5 in case 1 and 1.0 in case 2.

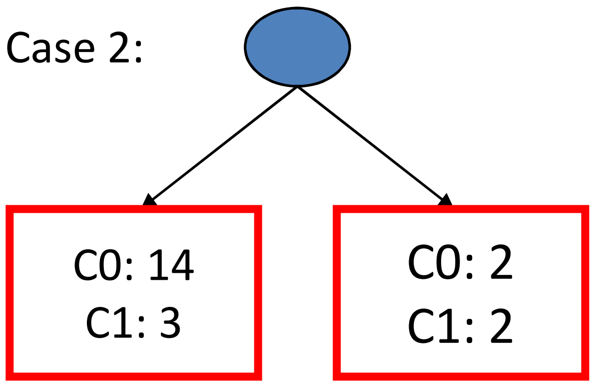
– Reduced error pruning?

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

Missing Values: 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

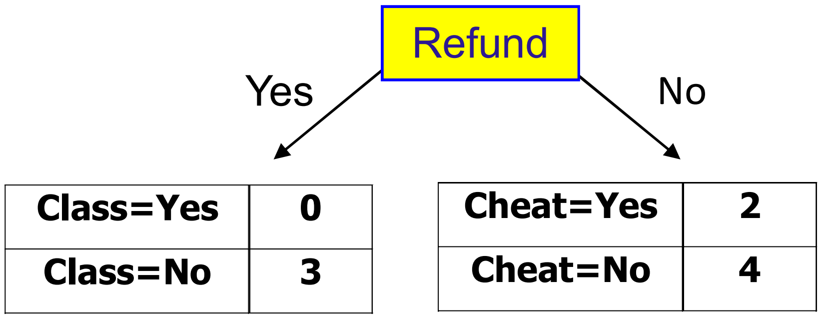
Entropy(Refund=No)
= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183

Entropy(Children)
= 3/9 (0) + 6/9 (0.9183) = 0.6122

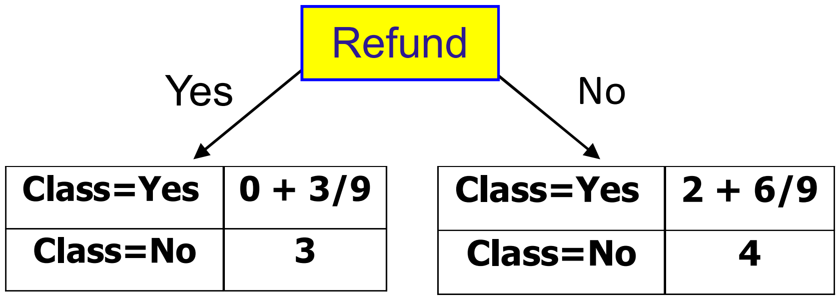
Gain = **0.9** × (0.8813 – 0.6122) = 0.2422

Missing Values: 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



Probability that Refund=Yes is **3/9**

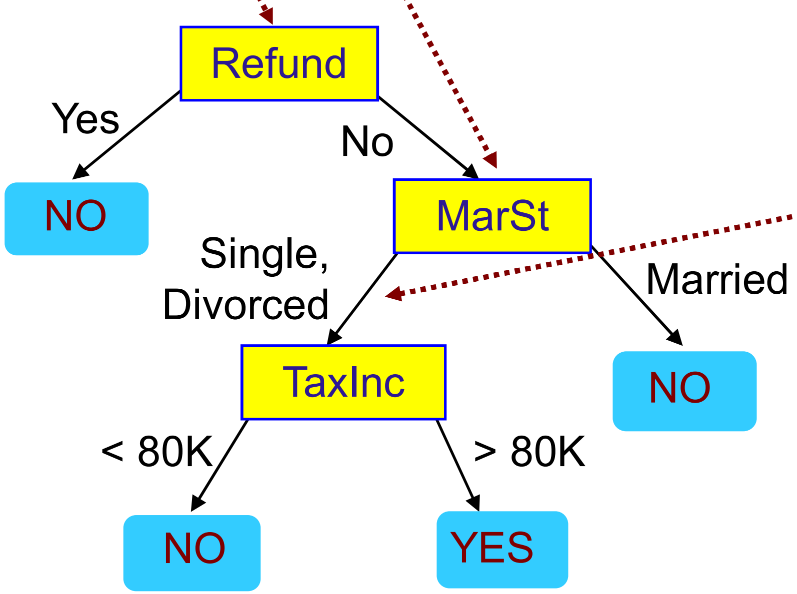
Probability that Refund=No is **6/9**

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

Missing Values: 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	0	1+**6/9**	1	2.67
Total	3	2.67	1	6.67

From previous slide

Probability of Marital Status being Married is 3/6.67

Probability of Marital Status being {Single,Divorced} is 3.67/6.67

Other Issues

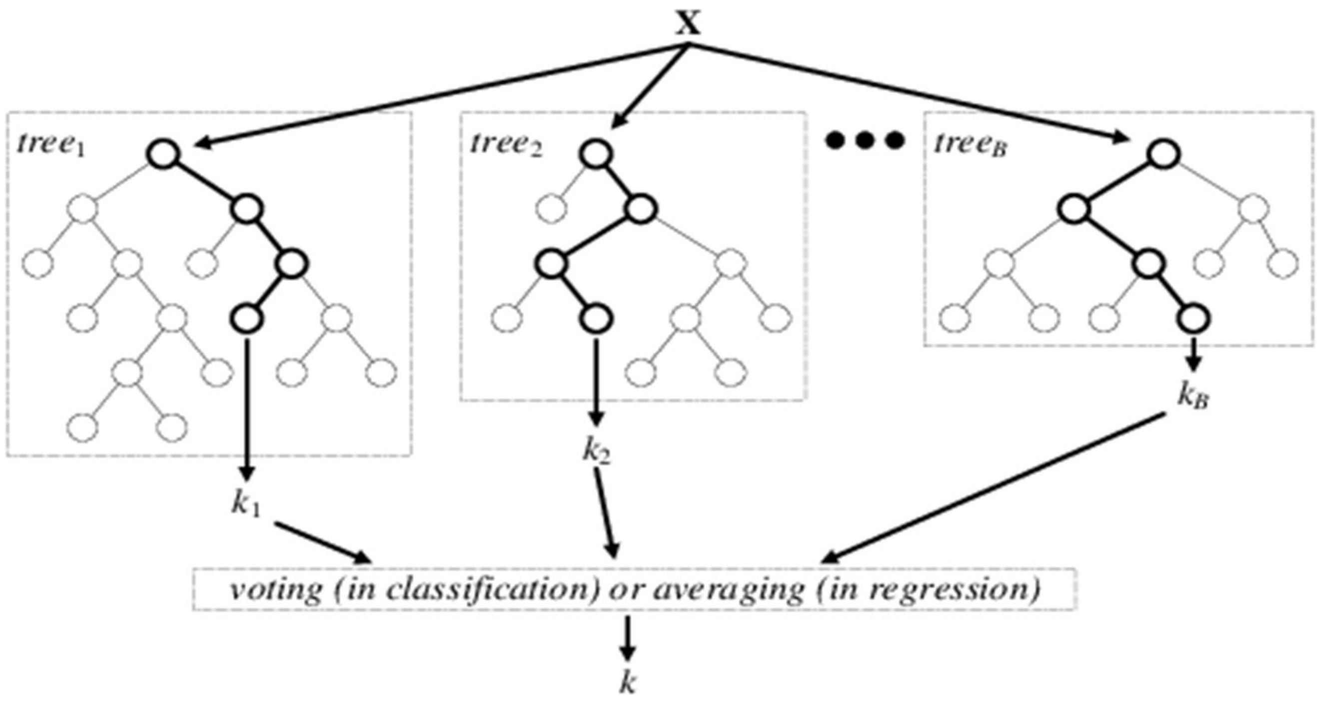
- Data Fragmentation
  - Number of records get smaller as you traverse down the tree
  - Number of records at the leaf nodes could be too small to make any statistically significant decision
- Difficult to interpret large-sized trees
  - Tree could be large because of using a single attribute in the test condition
- Tree Replication
  - Subtree may appear at different parts of a decision tree
  - Constructive induction: create new attributes by combining existing attributes
- **Search Strategy**
- Expressiveness

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

Case Study: Web Robot Detection

Random Forest in Machine Learning



[Random Forest - Fun and Easy Machine Learning]

Summary of Decision Tree Classification

1. What is the purpose of classification?
2. What is a decision tree?
3. How to build a decision tree?
4. How to build a good decision tree?
5. How to measure whether a decision tree is good?