北京航空航天大學
BEIHANG UNIVERSITY

信息系统分析与设计 Part II
Classification

数据智能研究中心
RESEARCH CENTER FOR DATA INTELLIGENCE

Classification: Definition

- Given a collection of records (training set)
  - Each record is by characterized by a tuple $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is the attribute set and y is the class label
    - $\boldsymbol{x}$: attribute, predictor, independent variable, input
    - y: class, response, dependent variable, output

- Task:
  - Learn a model that maps each attribute set $\boldsymbol{x}$ into one of the predefined class labels y

Examples of Classification Task

Task	Attribute set, ***x***	Class label, *y*
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

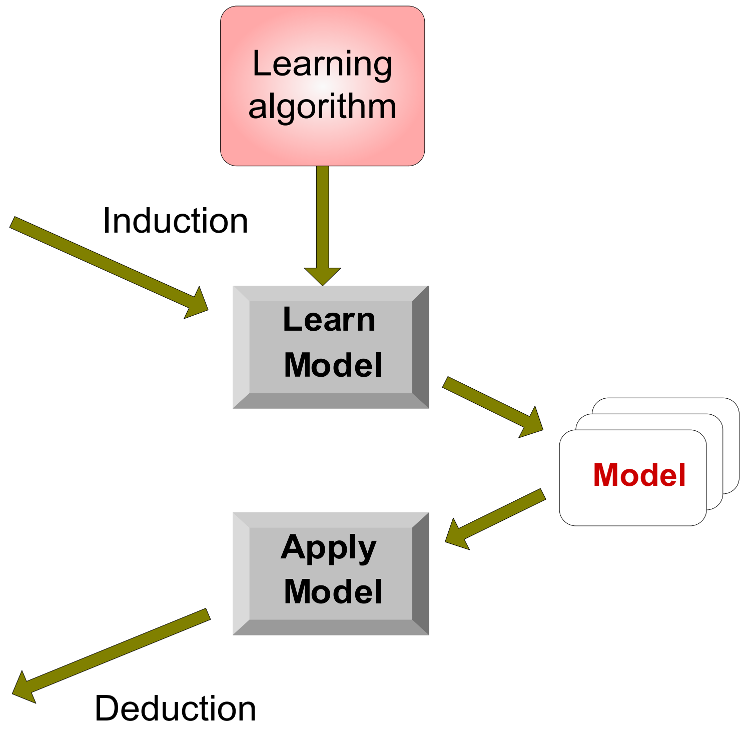
General Approach for Building Classification Model

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



Classification Techniques

- Base Classifiers
  - Rule-based Methods
  - Decision Tree based Methods
  - Support Vector Machines

  - Nearest-neighbor
  - Neural Networks
  - Naïve Bayes and Bayesian Belief Networks

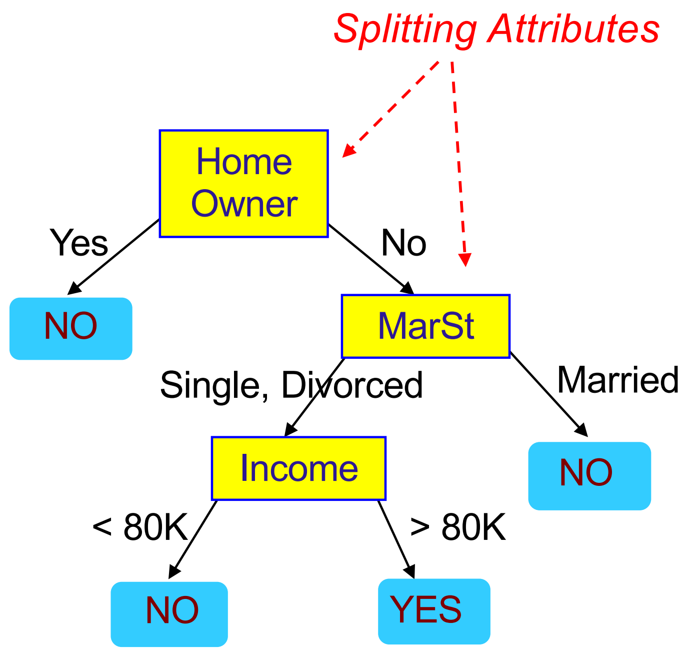
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

Example of a Decision Tree

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

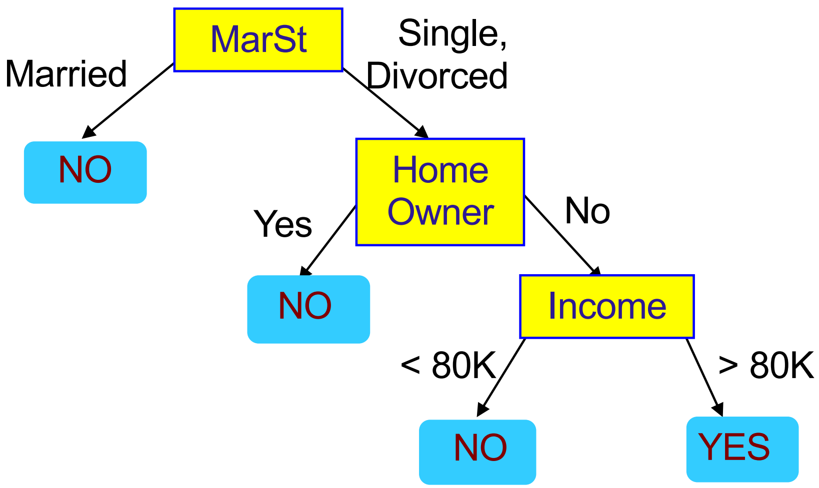


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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!

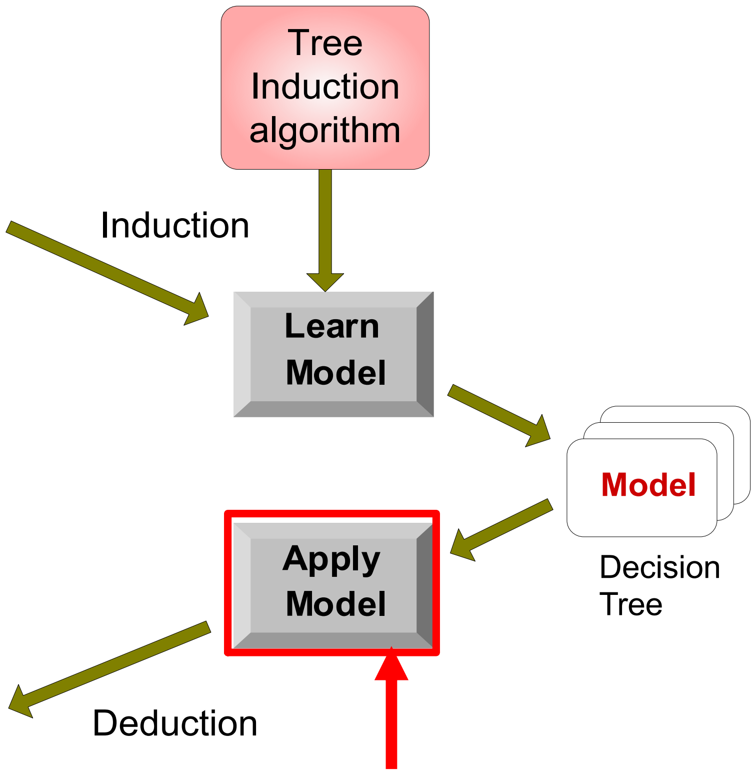
Decision Tree Classification Task

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

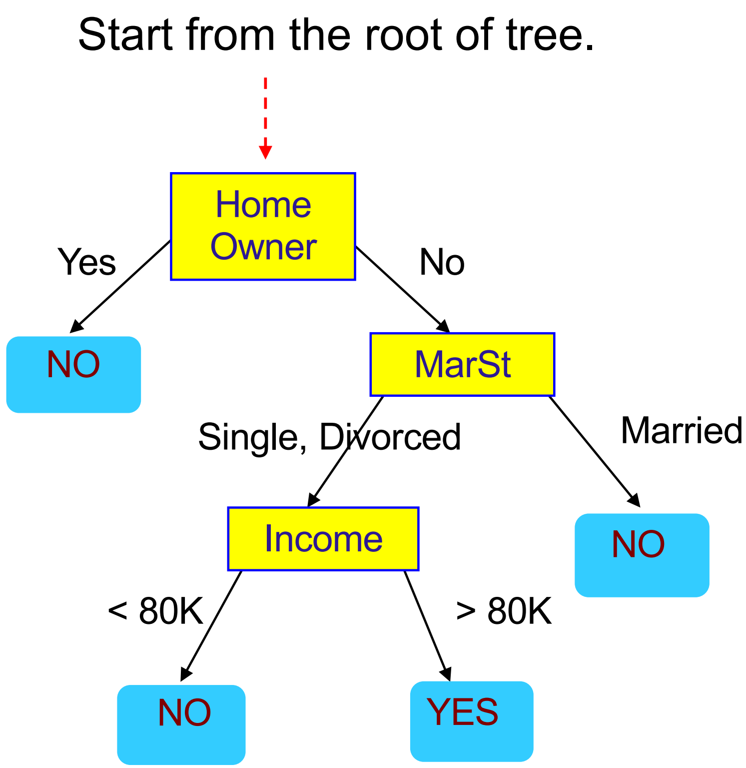
Training Set

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



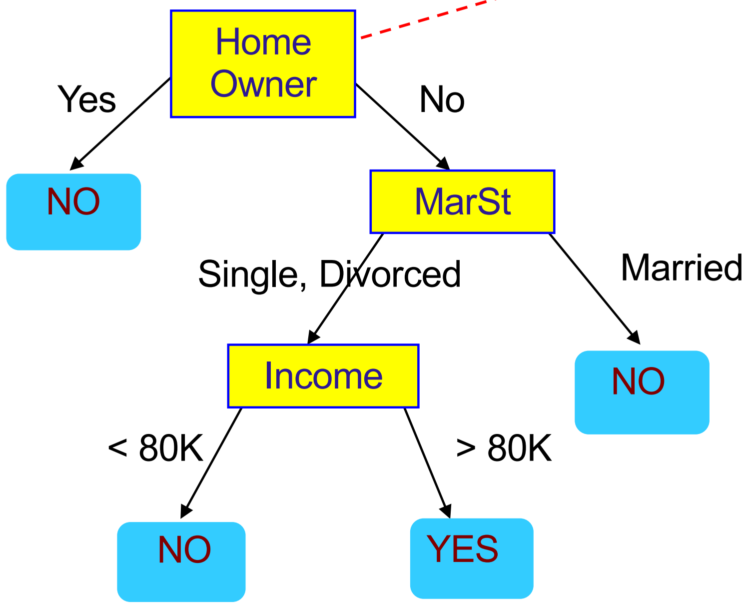
Apply Model to Test Data



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

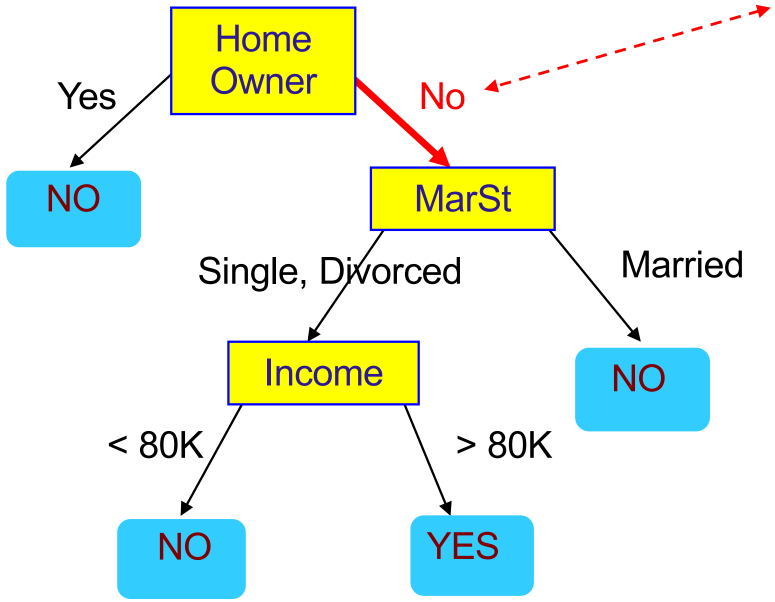
Apply Model to Test Data



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Apply Model to Test Data



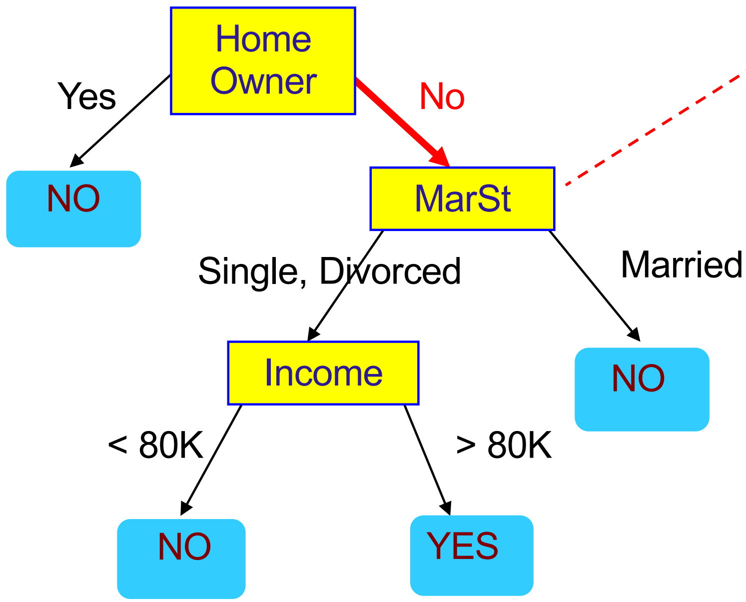
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

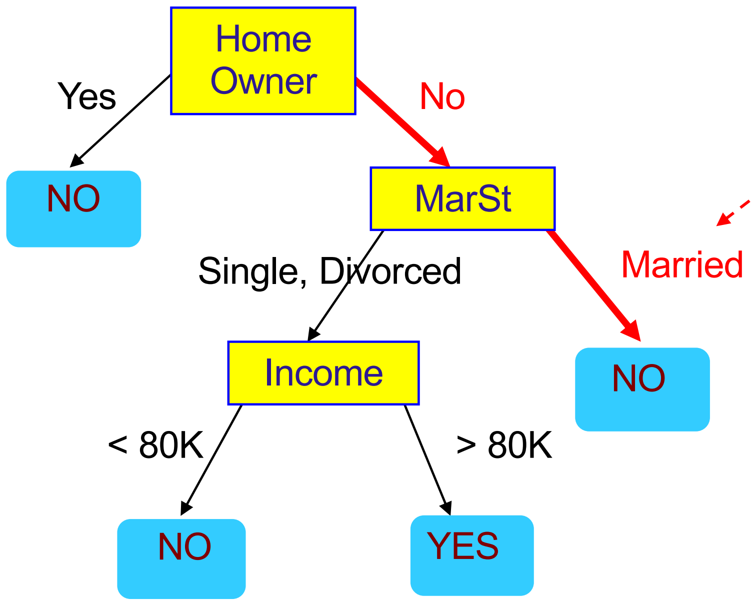
Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data



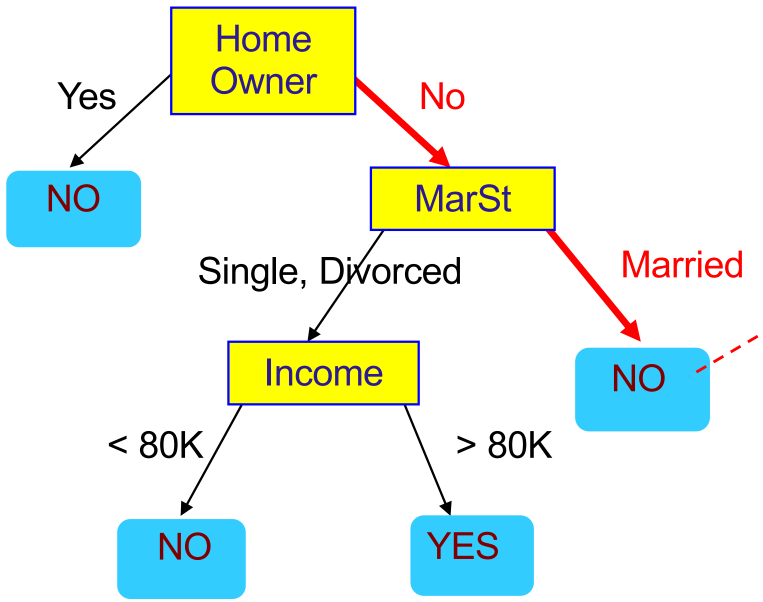
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

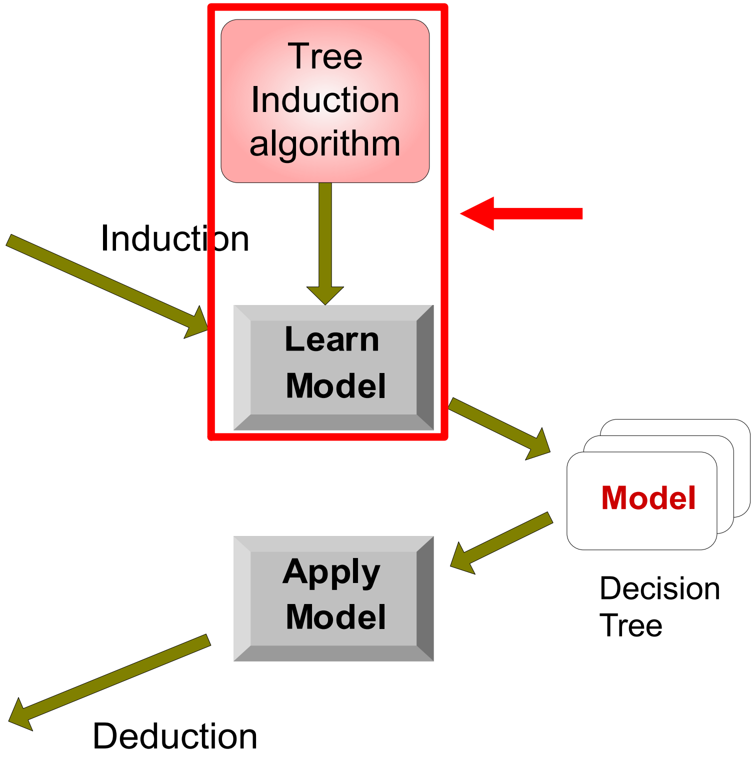


Assign Defaulted to "No"

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



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

Test Set

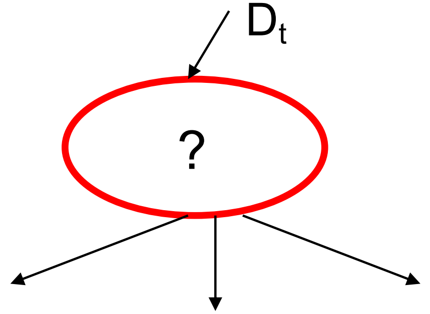
Decision Tree Induction

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

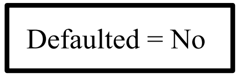
General Structure of Hunt's Algorithm

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

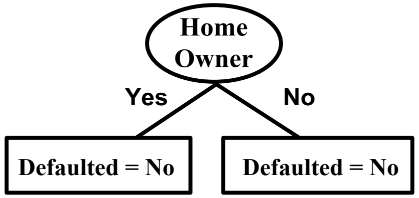
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt's Algorithm

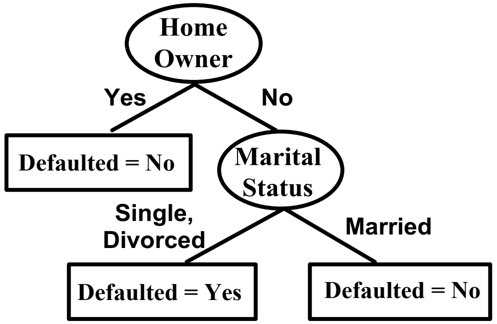


(a)

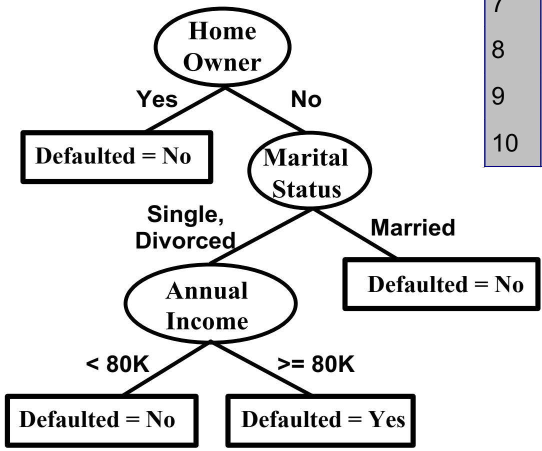


(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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



(c)



(d)

Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for specifying test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

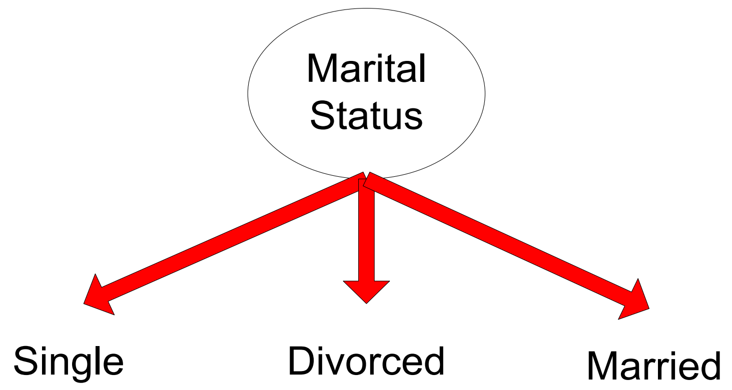
Methods for Expressing Test Conditions

- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

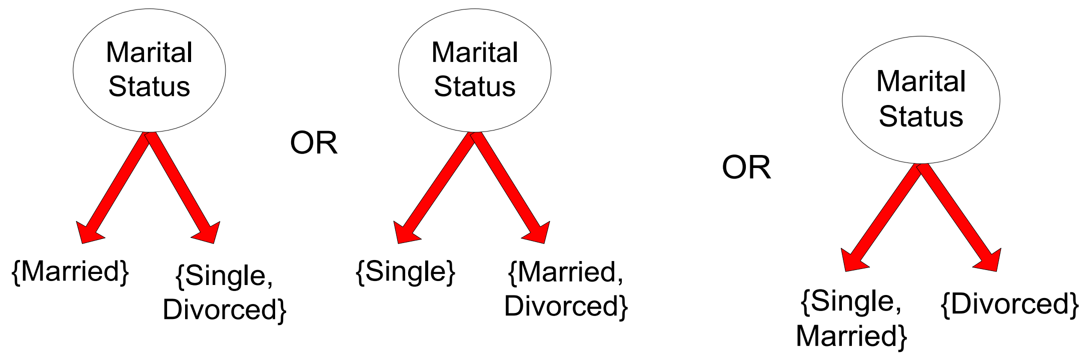
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Test Condition for Nominal Attributes

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

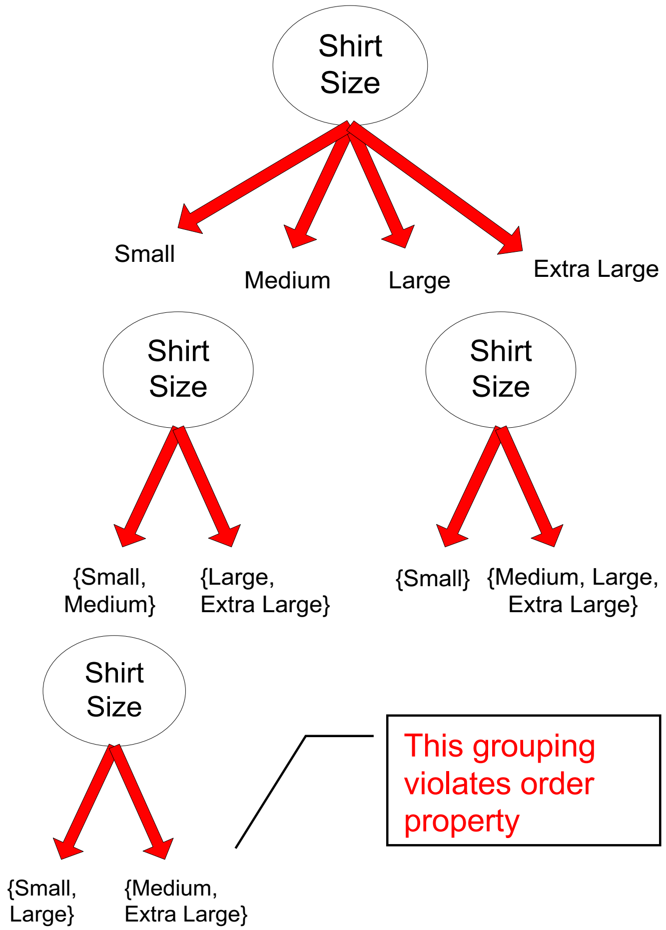


- Binary split:
  - Divides values into two subsets
  - Need to find optimal partitioning.

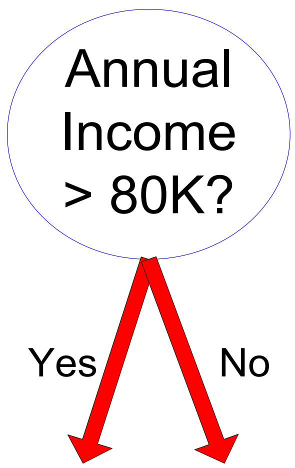


Test Condition for Ordinal Attributes

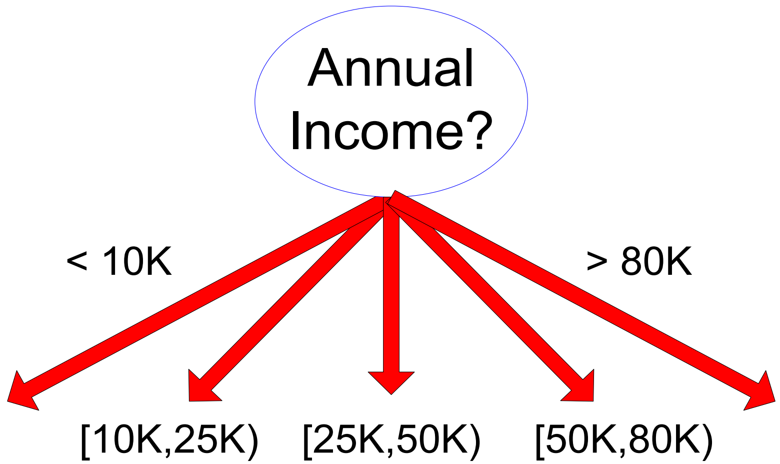
- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets
  - Need to find optimal partitioning
  - Preserve the order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split



(ii) Multi-way split

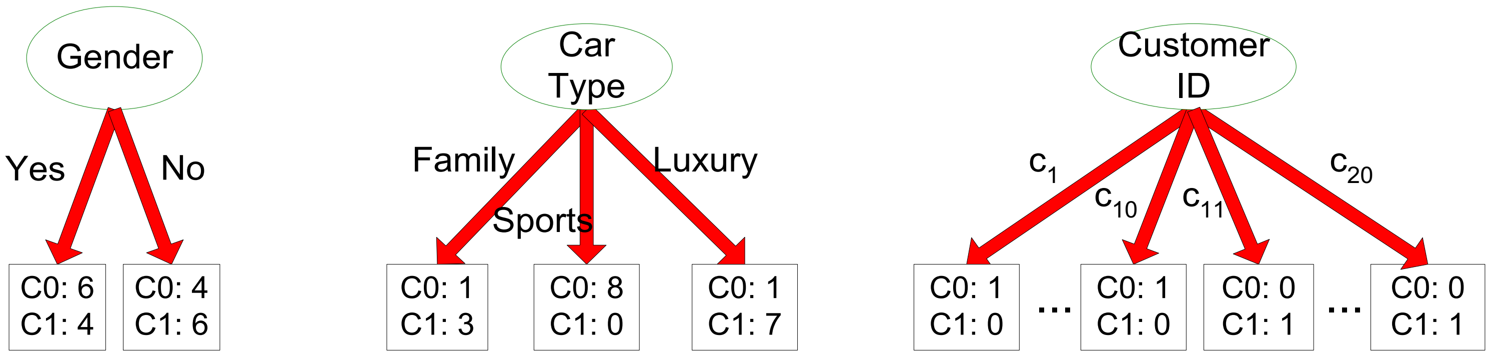
Splitting Based on Continuous Attributes

- 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

How to determine the Best Split

Customer 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

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 purer class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

High degree of impurity

C0: 9
C1: 1

Low degree of impurity

Measures of Node Impurity

- Gini Index

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

- Entropy

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

- Misclassification error

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

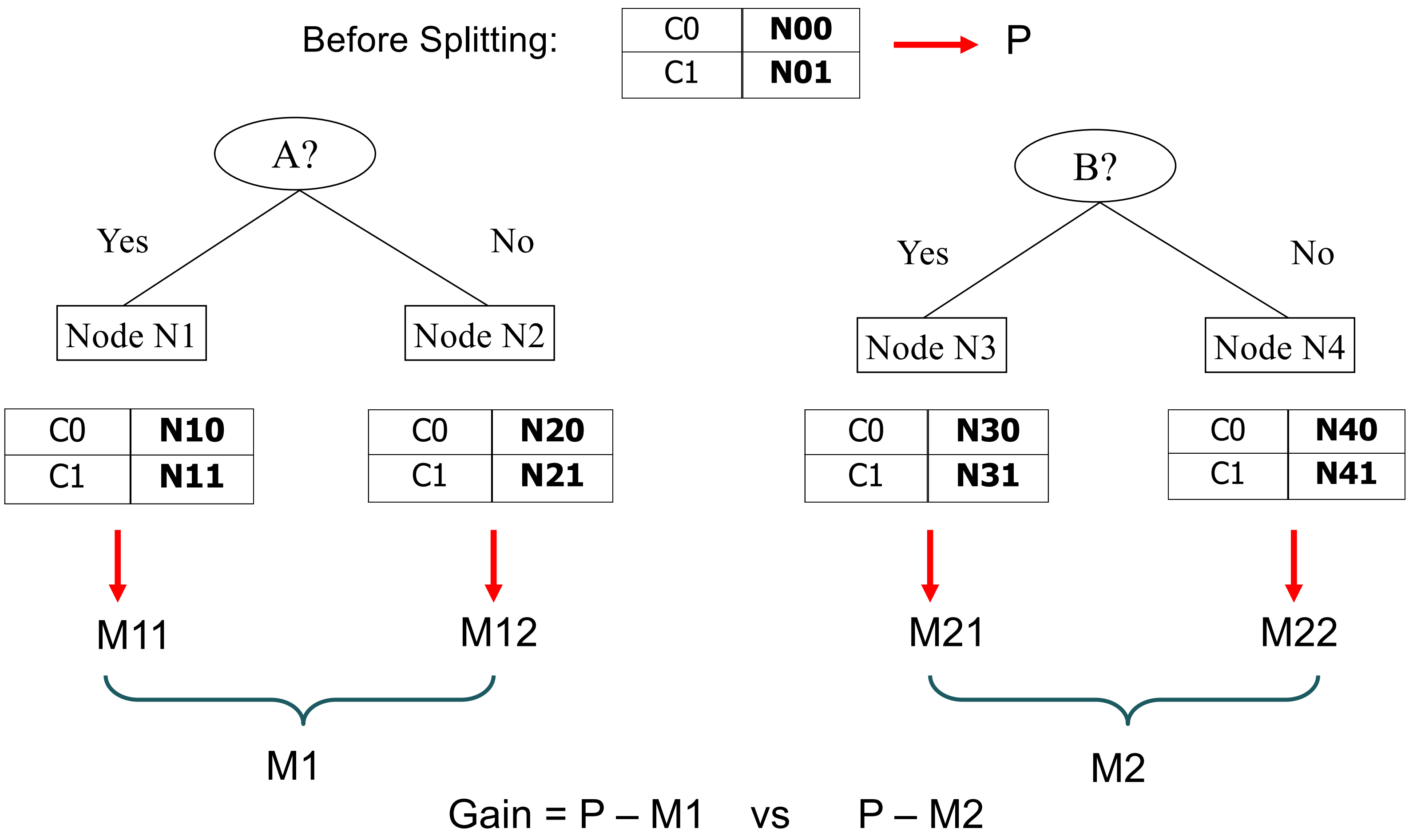
Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - Compute the average impurity of the children (M)
3. Choose the attribute test condition that produces the highest gain

$$\text{Gain} = P - M$$

or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split



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

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	

Computing Gini Index of a Single Node

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

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

Computing Gini Index for a Collection of Nodes

- When a node p is split into k partitions (children)

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

where, ni = number of records at child i,

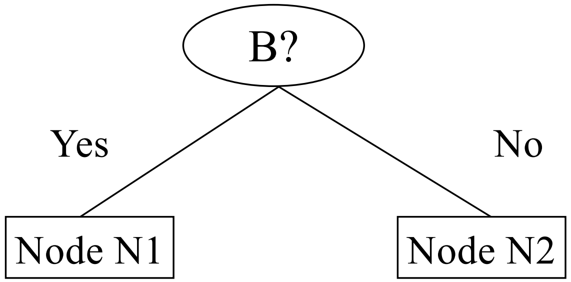
n = number of records at parent node p.

- Choose the attribute that minimizes weighted average Gini index of the children

- Gini index is used in decision tree algorithms such as CART, SLIQ, SPRINT

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	

Gini(N1)
$= 1 - (5/6)^2 - (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 - (2/6)^2 - (4/6)^2$
$= 0.444$

	N1	**N2**
C1	**5**	**2**
C2	**1**	**4**
Gini=0.361		

Gini(Children)
$= 6/12 * 0.278 +$
$6/12 * 0.444$
$= 0.361$

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	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

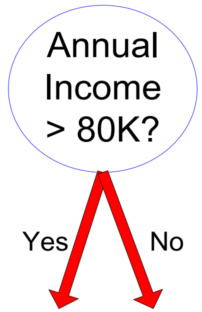
	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

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.

ID	Home Owner	Marital Status	Annual Income	Defaulted
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		
	Annual 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	

Measure of Impurity: Entropy

- Entropy at a given node t:

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

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

  - 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 quite similar to the GINI index computations

Computing Entropy of a Single Node

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

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

Computing Information Gain After Splitting

- Information Gain:

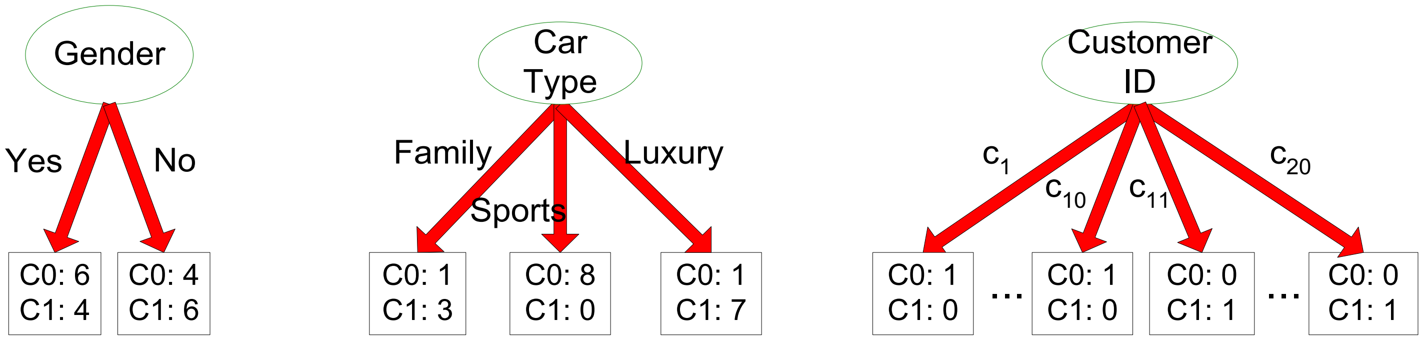
$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;
n_i is number of records in partition i

- Choose the split that achieves most reduction (maximizes GAIN)
- Used in the ID3 and C4.5 decision tree algorithms

Problems with Information Gain

- Info Gain tends to prefer splits that result in large number of partitions, each being small but pure



- Customer ID has highest information gain because entropy for all the children is zero

Gain Ratio

- 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
  - ni 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 algorithm
- Designed to overcome the disadvantage of Information Gain

Measure of Impurity: Classification Error

- Classification error at a node t :

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

  - 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

Computing Error of a Single Node

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

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

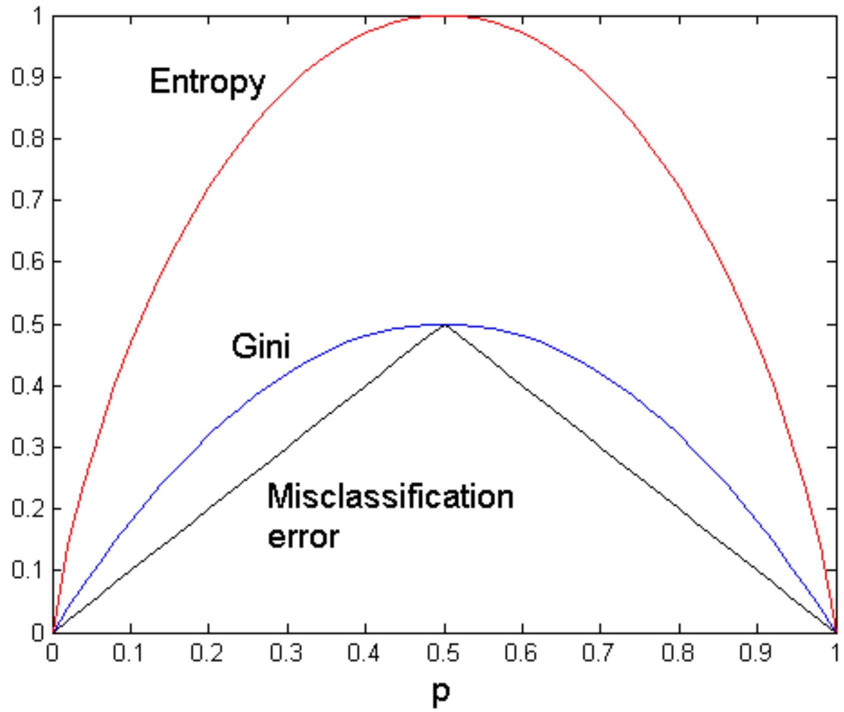
C1	**2**
C2	**4**

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

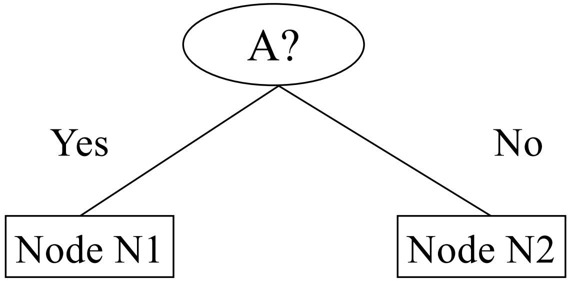
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Impurity Measures

For a 2-class problem:



Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

Gini(N1)
$= 1 - (3/3)^2 - (0/3)^2$
$= 0$

Gini(N2)
$= 1 - (4/7)^2 - (3/7)^2$
$= 0.489$

	N1	**N2**
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

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

Gini improves but error remains the same!!

Evaluation of Classification

- Confusion Matrix:

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

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy

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

- Most widely-used metric:

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

Problem with Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If a model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - This is misleading because the model does not detect any class 1 example
  - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

Alternative Measures

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

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

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

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

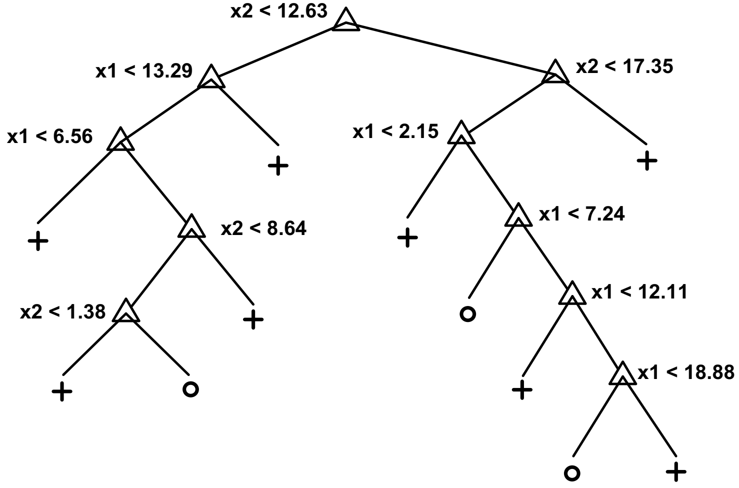
Classification Errors

- Training errors (apparent errors)
  - Errors committed on the training set

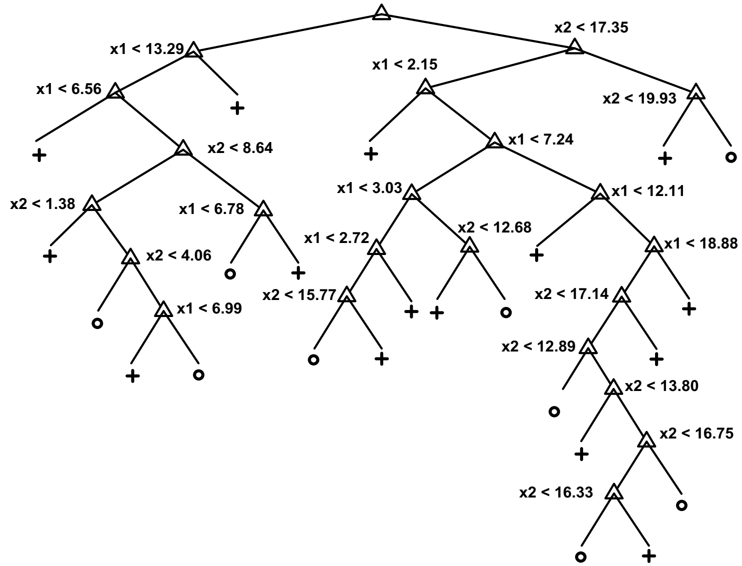
- Test errors
  - Errors committed on the test set

- Generalization errors
  - Expected error of a model over random selection of records from same distribution

Decision Tree



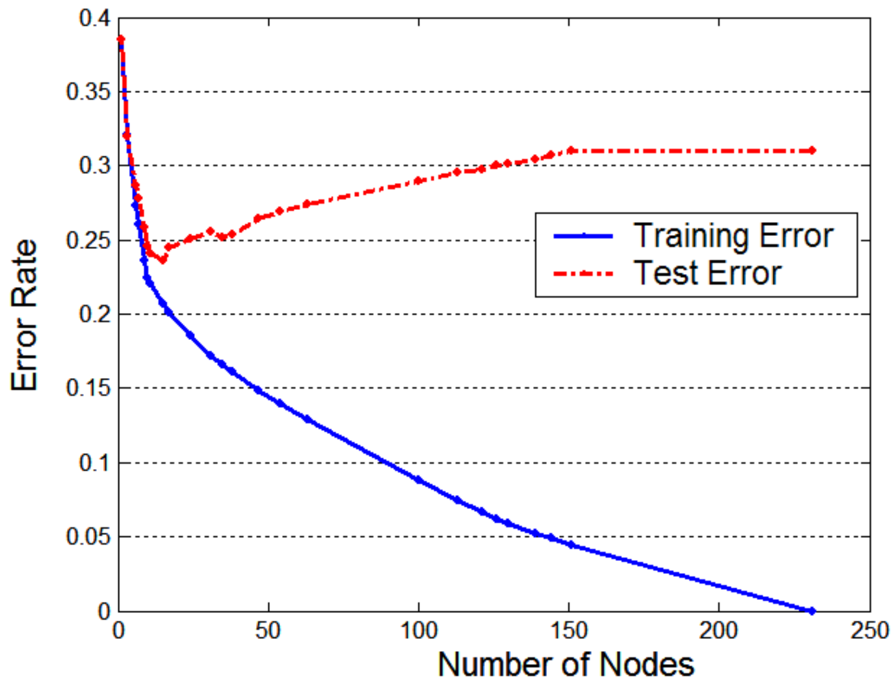
Decision Tree with 11 leaf nodes



Decision Tree with 24 leaf nodes

Which tree is better?

Model Overfitting



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

Overfitting: when model is too complex, training error is small but test error is large

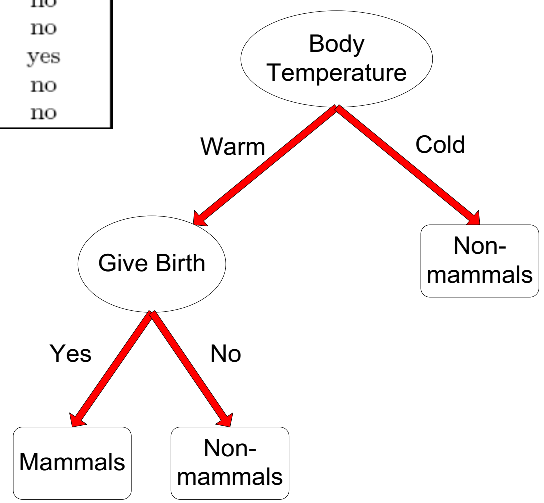
Mammal Classification Problem

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Mammal
human	warm-blooded	hair	yes	no	no	yes	no	yes
python	cold-blooded	scales	no	no	no	no	yes	no
salmon	cold-blooded	scales	no	yes	no	no	no	no
whale	warm-blooded	hair	yes	yes	no	no	no	yes
frog	cold-blooded	none	no	semi	no	yes	yes	no
komodo dragon	cold-blooded	scales	no	no	no	yes	no	no
bat	warm-blooded	hair	yes	no	yes	yes	yes	yes
pigeon	warm-blooded	feathers	no	no	yes	yes	no	no
cat	warm-blooded	fur	yes	no	no	yes	no	yes
leopard shark	cold-blooded	scales	yes	yes	no	no	no	no
turtle	cold-blooded	scales	no	semi	no	yes	no	no
penguin	warm-blooded	feathers	no	semi	no	yes	no	no
porcupine	warm-blooded	quills	yes	no	no	yes	yes	yes
eel	cold-blooded	scales	no	yes	no	no	no	no
salamander	cold-blooded	none	no	semi	no	yes	yes	no

Training Set

Decision Tree Model

training error = 0%

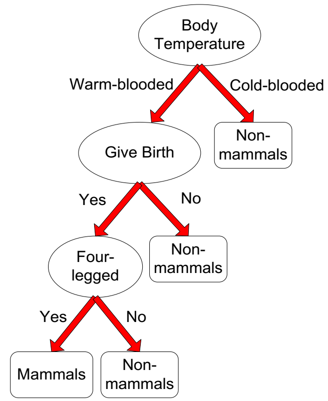


Effect of Noise

Example: Mammal Classification problem

Training Set:

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no



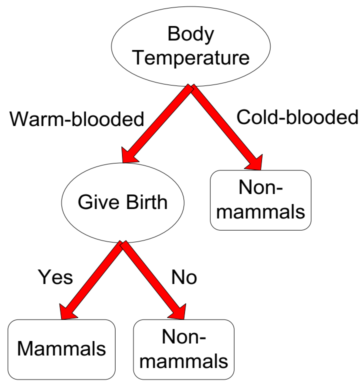
Model M1:

train err = 0%,

test err = 30%

Test Set:

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no



Model M2:

train err = 20%,

test err = 10%

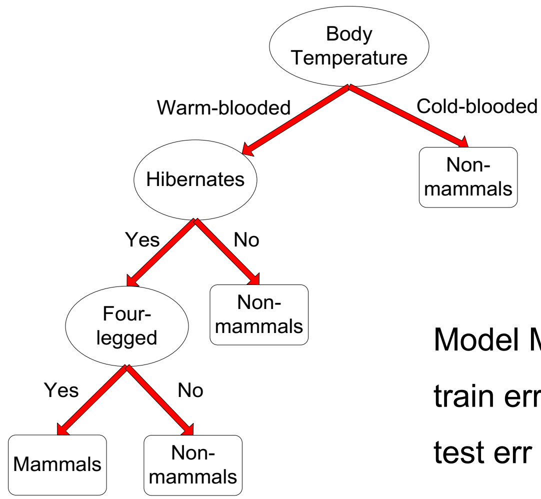
Lack of Representative Samples

Training Set:

Name	Body Temperature	Four-legged	Hibernates	Class Label
salamander	cold-blooded	yes	yes	no
guppy	cold-blooded	no	no	no
eagle	warm-blooded	no	no	no
poorwill	warm-blooded	no	yes	no
platypus	warm-blooded	yes	yes	yes

Test Set:

Name	Body Temperature	Four-legged	Hibernates	Class Label
human	warm-blooded	no	no	yes
pigeon	warm-blooded	no	no	no
elephant	warm-blooded	yes	no	yes
leopard shark	cold-blooded	no	no	no
turtle	cold-blooded	yes	no	no
penguin	cold-blooded	no	no	no
eel	cold-blooded	no	no	no
dolphin	warm-blooded	no	no	yes
spiny anteater	warm-blooded	yes	yes	yes
gila monster	cold-blooded	yes	yes	no



Model M3:

train err = 0%,

test err = 30%

Lack of training records at the leaf nodes for making reliable classification

Notes on Overfitting

- Overfitting results in decision trees that are <u>more complex</u> than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating generalization errors

Using Validation Set

- Divide training data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set

- Drawback:
  - Less data available for training

- Thanks!

Let's enjoy playing with data!