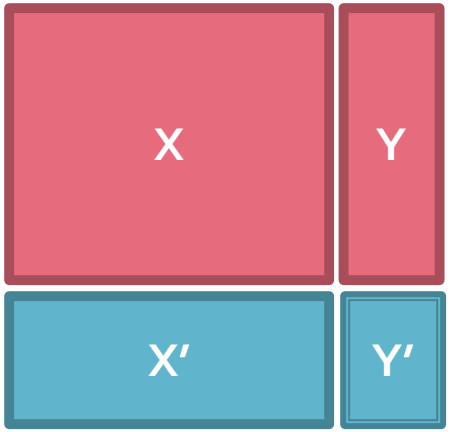
Classification (Part 1)

Some slides adapted from Stanford data mining course, "Introduction to Data Mining " by Kumar etc, and UIC data mining course

Supervised Learning

- **Data is labeled:**
  - Have many pairs **{(x, y)}**
    - **x** … vector of binary, categorical, real valued features
    - **y** … class ({+1, -1}, or a real number)
- **Where *y* can be:**
  - **Real number**: Regression
  - **Categorical**: Classification
  - Complex object:
    - Ranking of items, etc.
- **Task of prediction:**
  estimate a function **f(x)** so that ***y = f(x)***



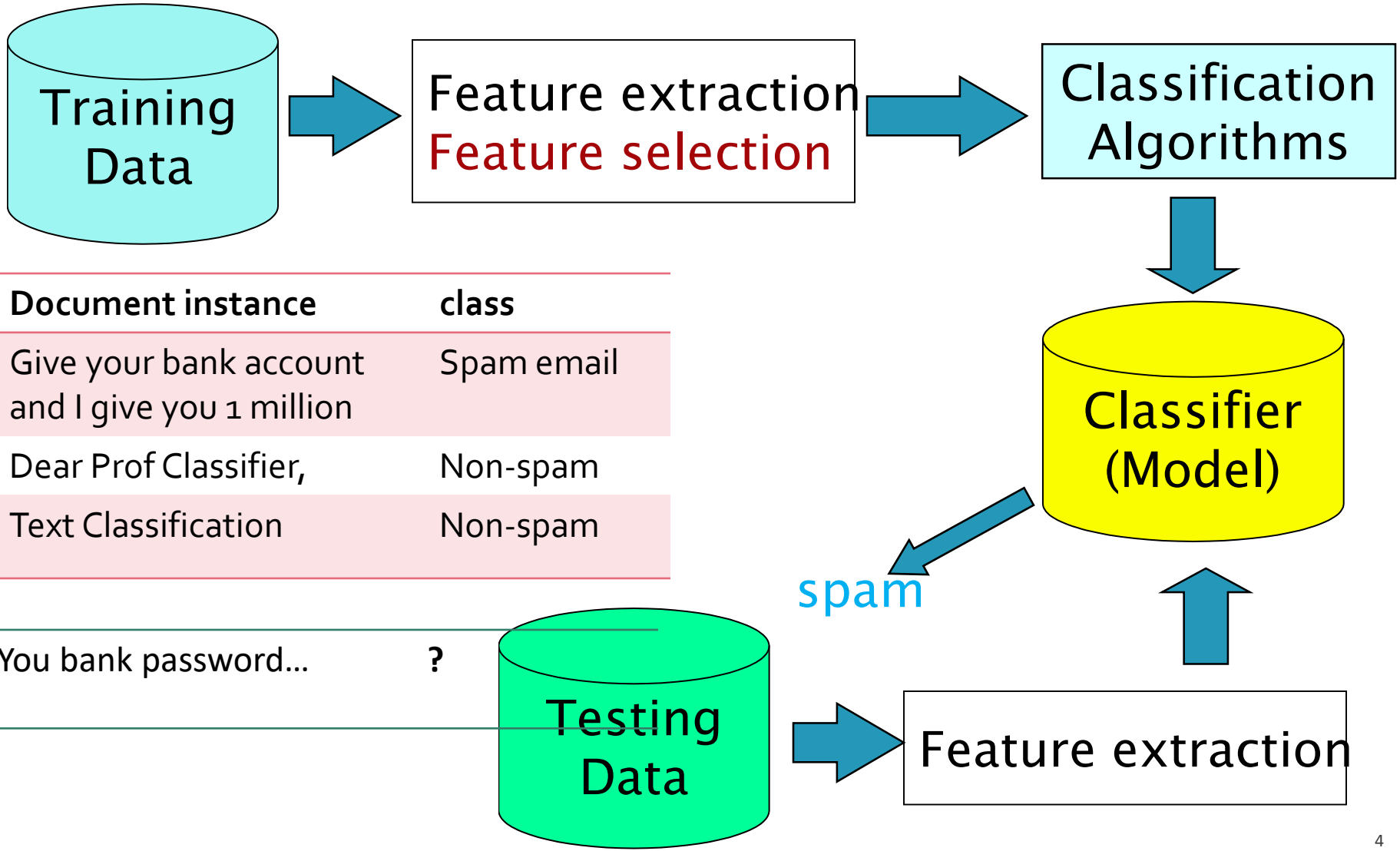
Training and test set

Estimate *y = f(x)* on X,Y.
Hope that the same *f(x)*
also works on unseen X', 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 x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

Classification Framework



Document instance	class
Give your bank account and I give you 1 million	Spam email
Dear Prof Classifier,	Non-spam
Text Classification	Non-spam
You bank password…	?

Classification Methods

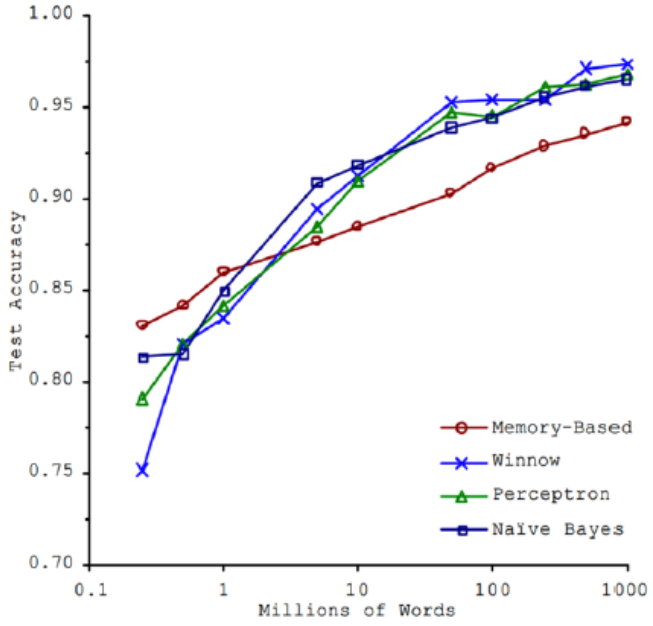
- Decision Tree based Methods
- Rule-based Methods
- Memory-based or Instance-based Methods
- Naïve Bayes Classifiers
- Support Vector Machines
- Neural Networks , Deep Neural Nets

(https://www.deeplearningbook.org/ for your reference)

- Ensemble Classification

Why large scale ML?

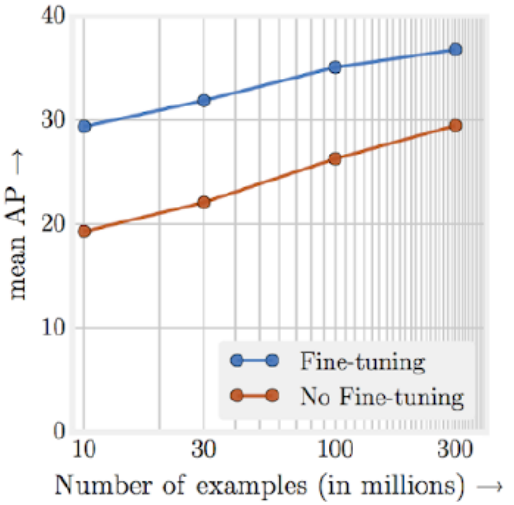
- **Brawn or Brains?**
  - In 2001, Microsoft researchers ran a test to evaluate 4 of different approaches to ML-based language translation
- **Findings:**
  - **Size of the dataset** used to train the model **mattered more** than the model itself
  - As the dataset grew large, performance difference between the models became small



Banko, M. and Brill, E. (2001), "Scaling to Very Very Large Corpora for Natural Language Disambiguation"

Why large scale ML?

- **The Unreasonable Effectiveness of Data**
  - In 2017, Google revisited the same type of experiment with the latest Deep Learning models in computer vision
- **Findings:**
  - Performance increases logarithmically **based on volume of training data**
  - Complexity of modern ML models (i.e., deep neural nets) allows for even **further performance gains**



- **Large datasets + large ML models => amazing results!!**

"Revisiting Unreasonable Effectiveness of Data in Deep Learning Era": https://arxiv.org/abs/1707.02968

Why Worry About Non-Deep Models?

A few reasons why this is important:

- Classical tasks in NLP and Vision are getting commoditized (you take pretrained model and fine tune it), but there are many other unique ML tasks.
- Deep models are often hard to scale and require lots and lots of data. Traditional models allow you to encode prior knowledge better and give you more control.
- Personally, if I am working on a well understood problem I'd use deep learning, but if I am the first person to work on a new problem/classifier I'd use techniques we'll discuss here.

Outline

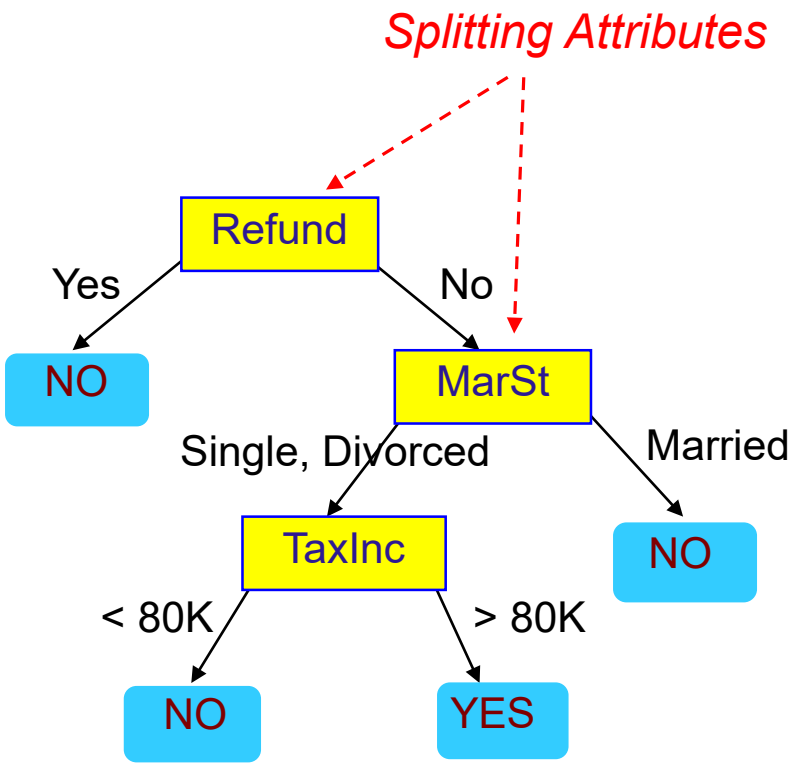
- Classification Techniques
  - Decision Tree
  - Classification based on association rules (CBA)
  - Ensemble classifier
  - Overfitting
  - Classification evaluation

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

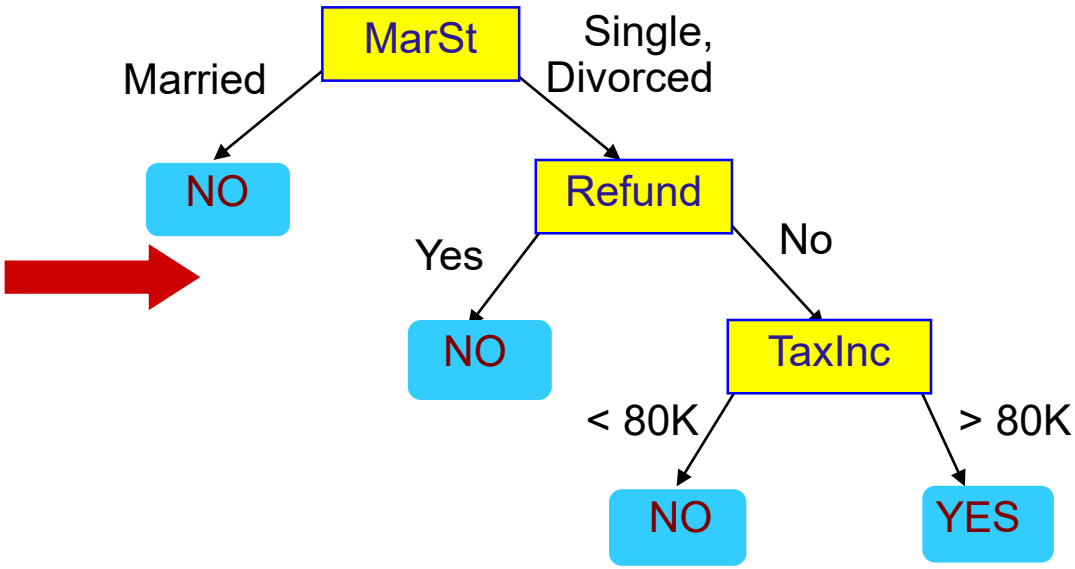


Model: Decision Tree

Another Example of Decision Tree

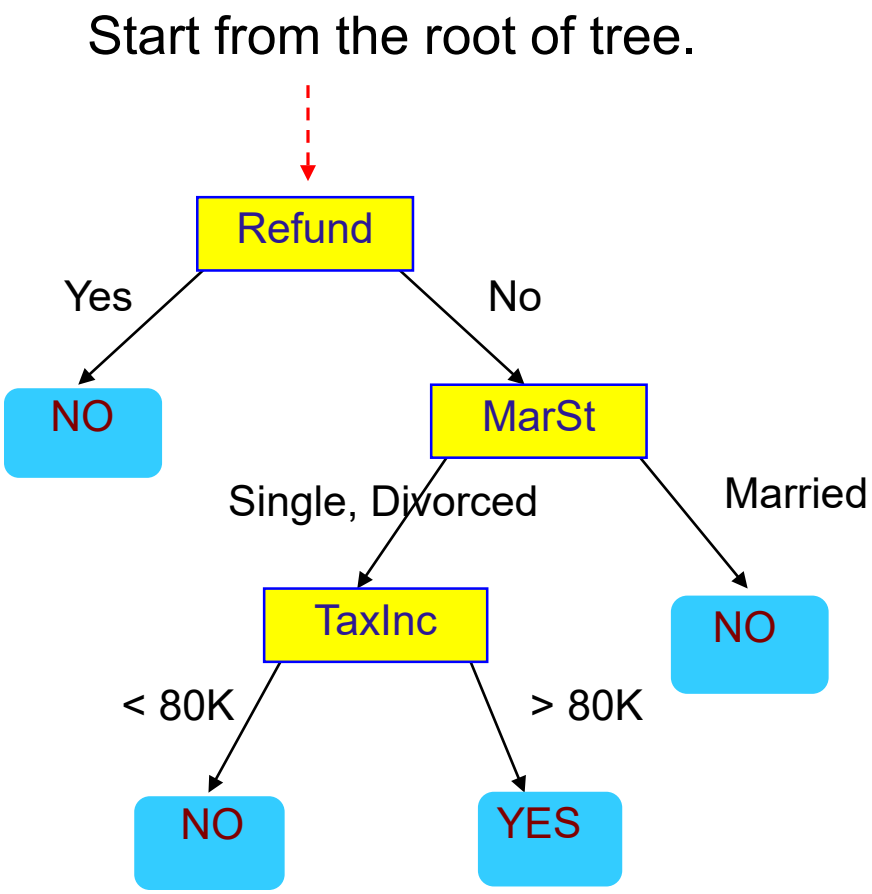
	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**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

Training Data



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

Apply Model to Test Data



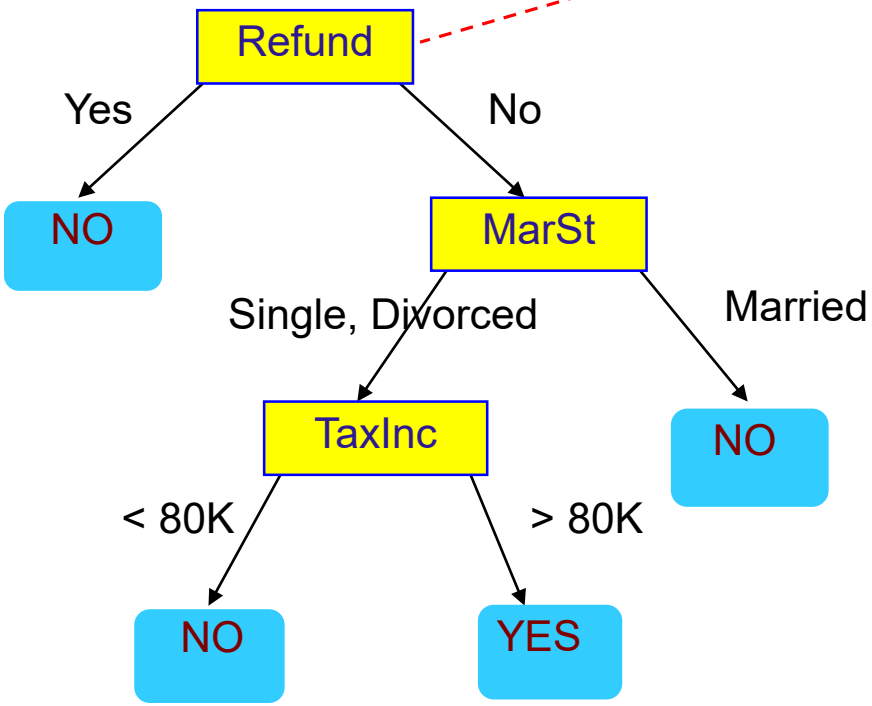
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

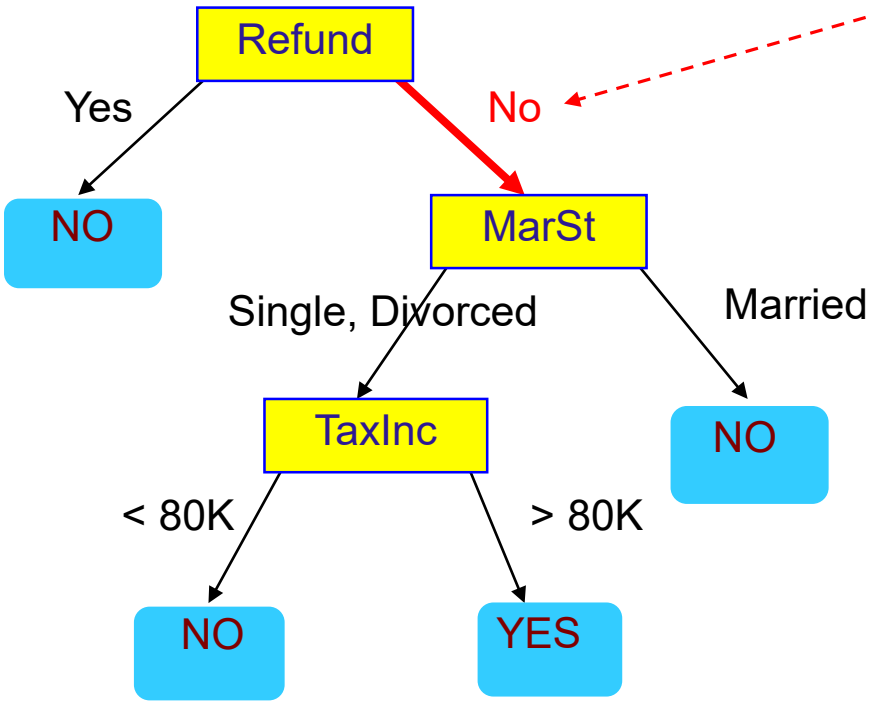
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

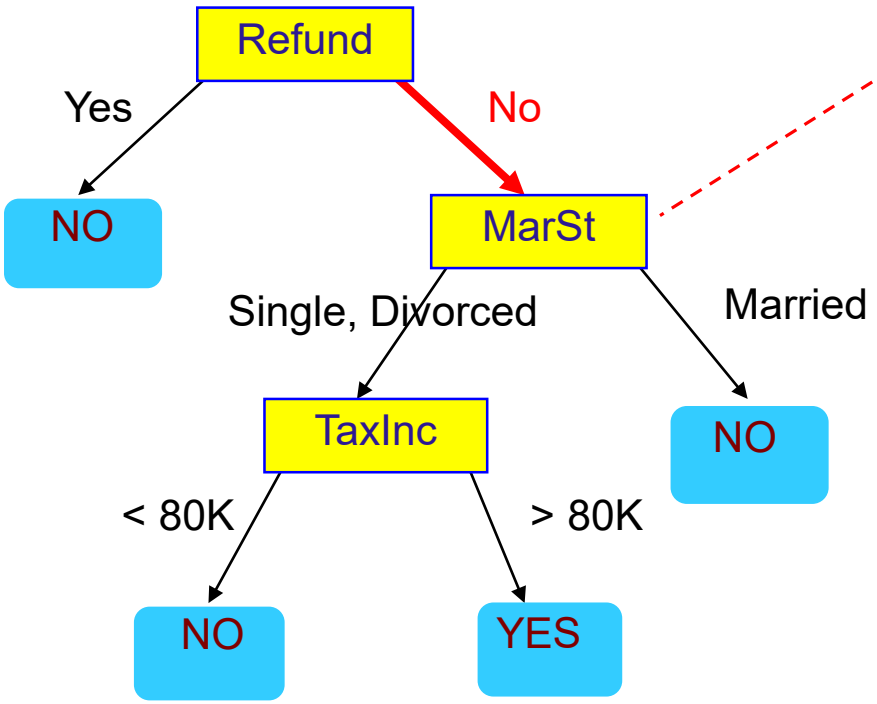
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

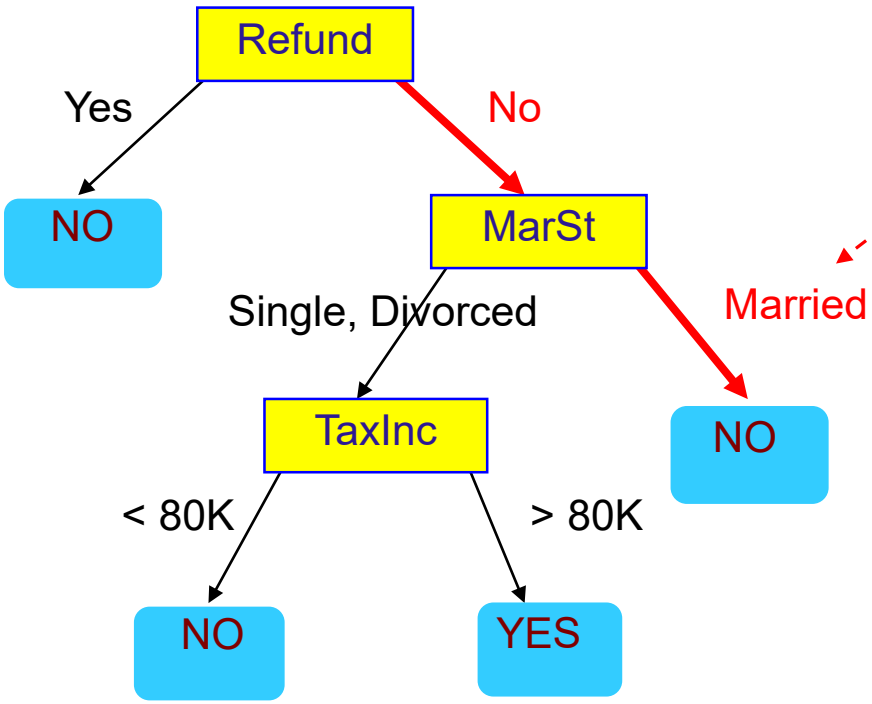
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



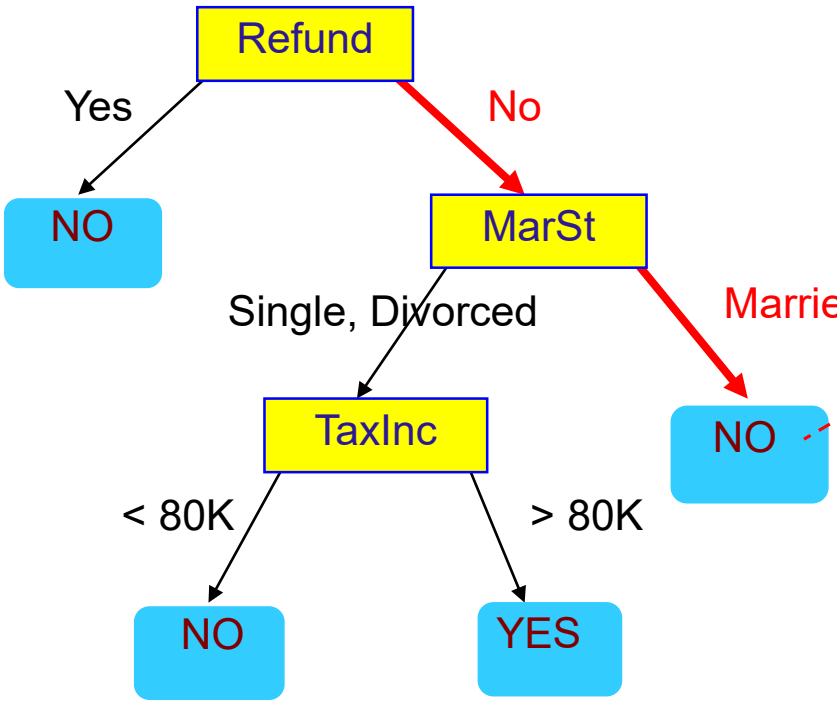
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

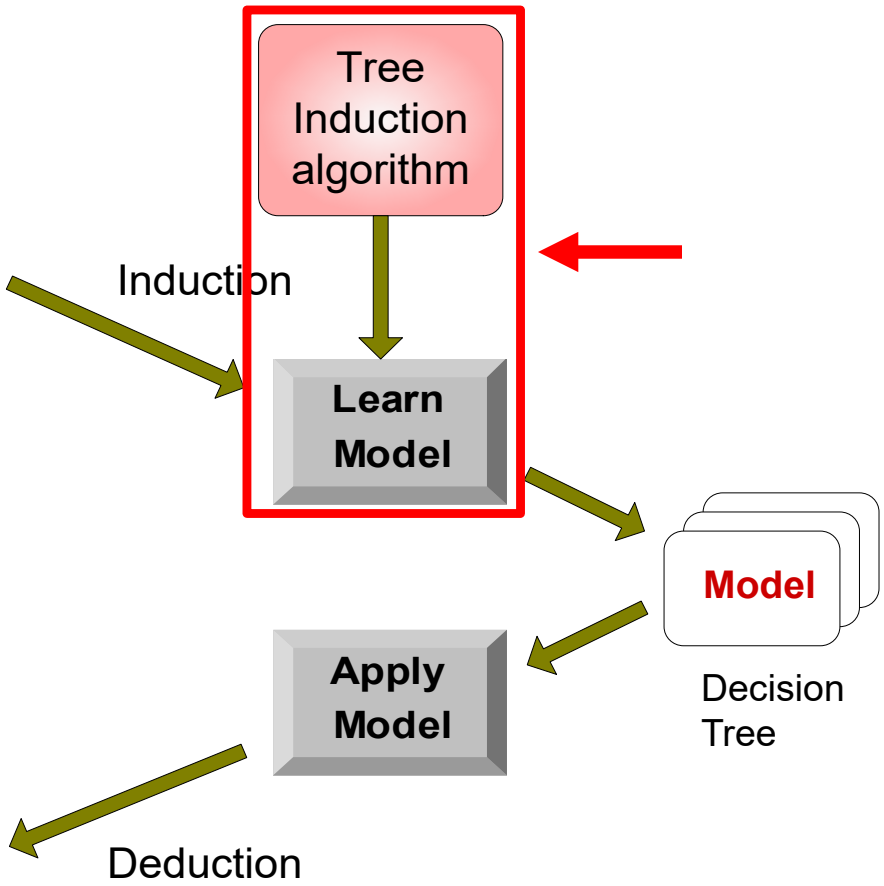
Decision Tree Classification Task

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



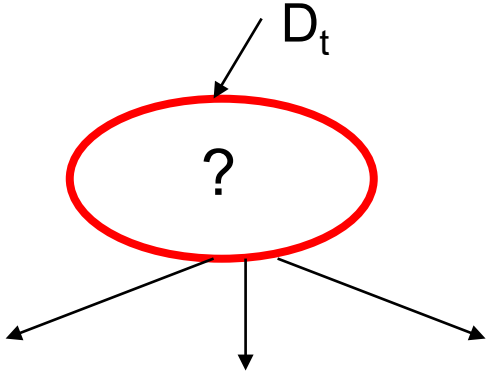
Decision Tree Induction

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

General Structure of Hunt's Algorithm

- Let 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 above 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



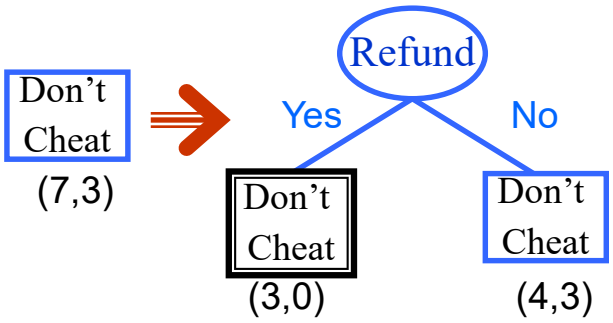
Hunt's Algorithm

Don't Cheat

(7,3)

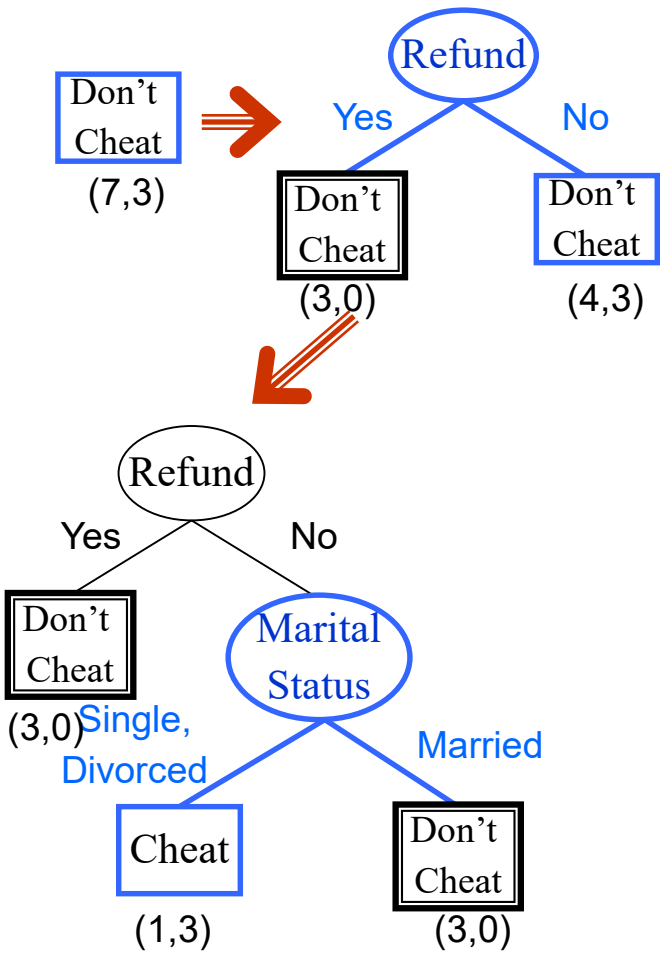
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm



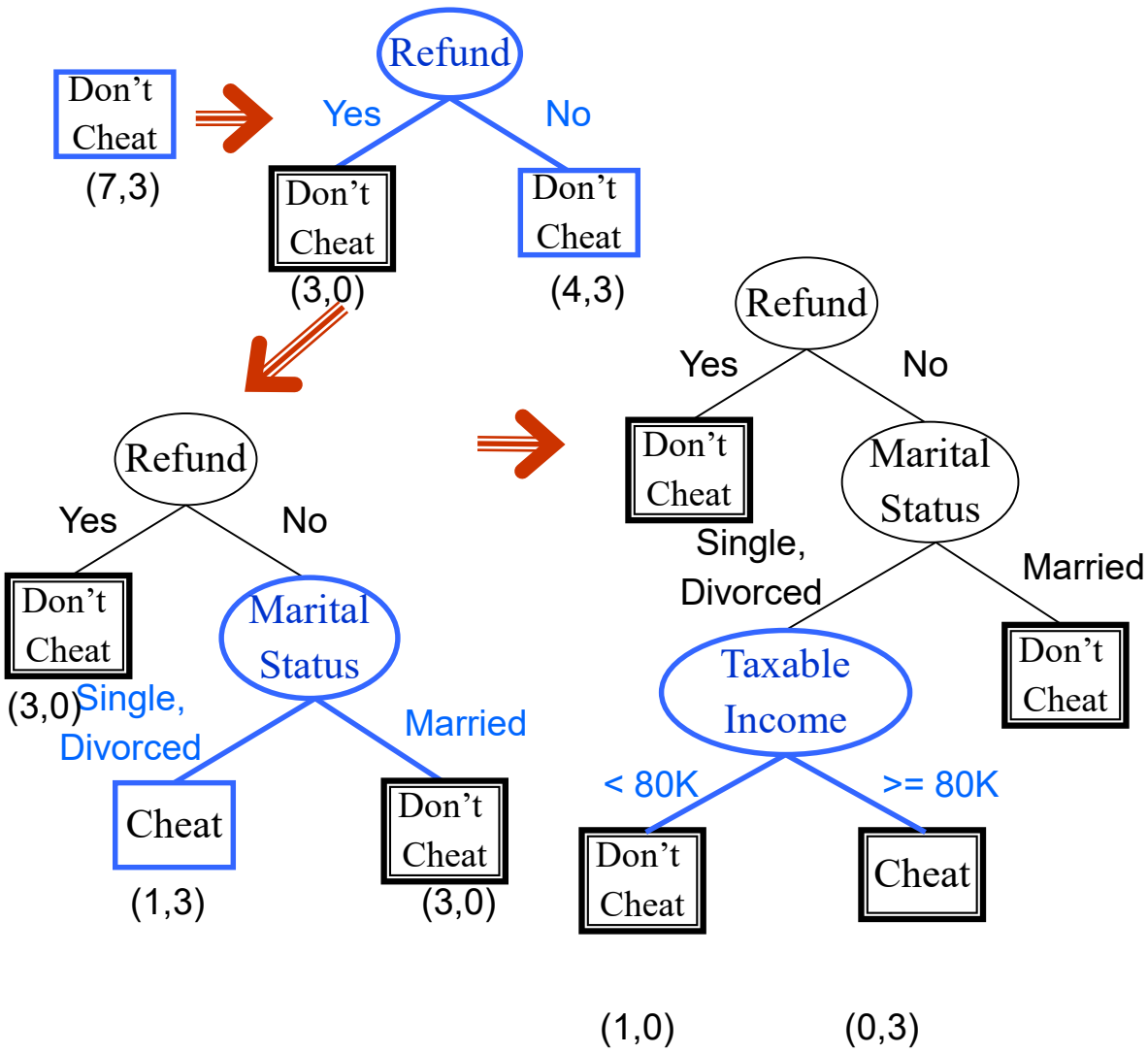
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

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

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

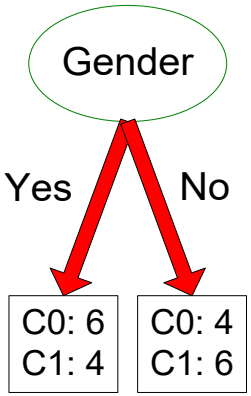
Design Issues of Decision Tree Induction

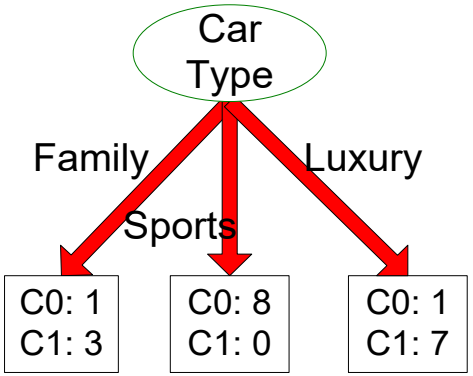
- The idea is to use greedy strategy.
  - Split the records based on **an attribute test** that optimizes certain criterion.
- Issues
  - Determine how to **split** the records
    - Which attribute to split and how to determine the best split?
  - Determine when to **stop splitting**

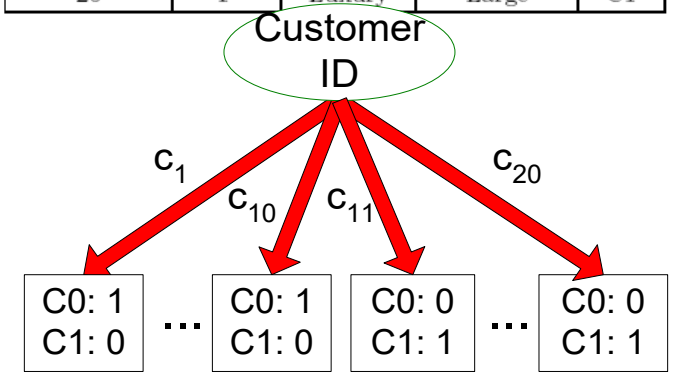
How to determine the Best Split

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

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

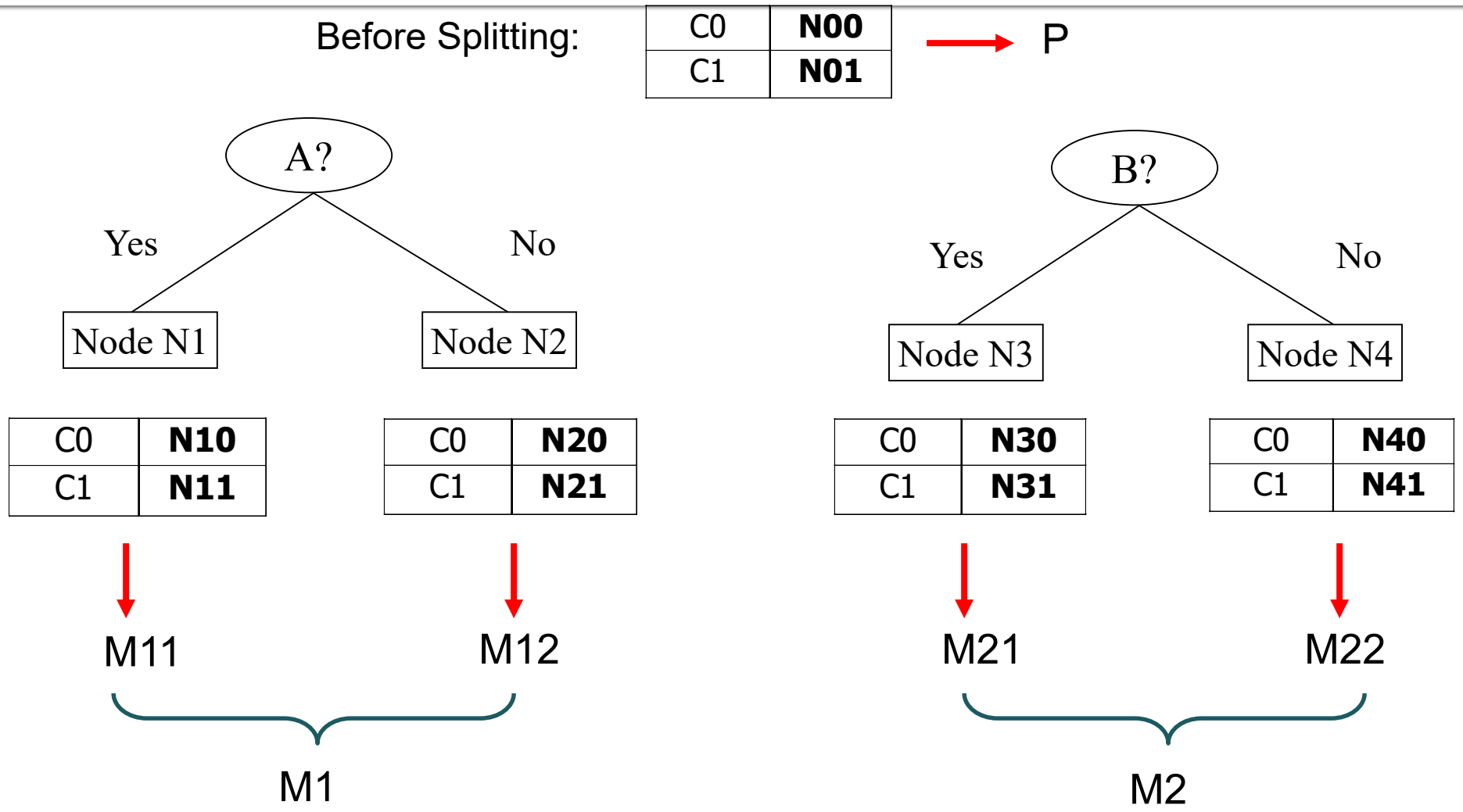
Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of child nodes
3. Choose the attribute and the split that produces the highest gain. *Note an attribute may have multiple ways to split*

Gain = P - M

or equivalently, lowest impurity measure after splitting (M)

Finding the Best Split



Measures of Node Impurity

- Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the probability of class $\boldsymbol{i}$ at node t, and $\boldsymbol{c}$ is the total number of classes

- Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

Computing GINI Measure

- Gini Index for a given node $\boldsymbol{t}$

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $\boldsymbol{p_i(t)}$ is the probability of class $\boldsymbol{i}$ at node $\boldsymbol{t}$, and $\boldsymbol{c}$ is the total number of classes

  - Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
  - Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification

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

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

For 2-class problem

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

$P_{(C1)}(t) = 0/6 = 0$ $P_{(C2)}(t) = 6/6 = 1$

Gini = $1 - P_{(C1)}(t)^2 - P_{(C2)}(t)^2 = 1 - 0 - 1 = 0$

C1	**1**
C2	**5**

$P_{(C1)}(t) = 1/6$ $P_{(C2)}(t) = 5/6$

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

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

$P_{(C1)}(t) = 2/6$ $P_{(C2)}(t) = 4/6$

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

Computing Gini Index for a Collection of Nodes

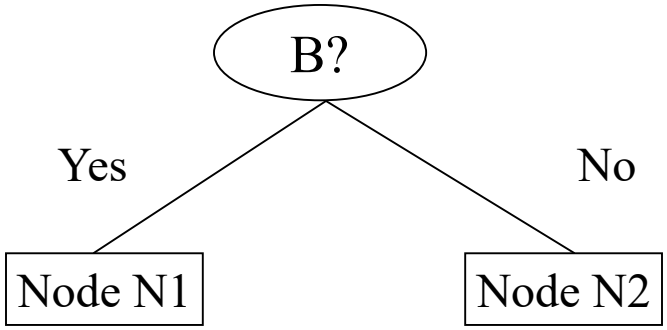
- In general, when a node p is split into k partitions (children)

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

where, n_i = number of records at child i,
n = number of records at parent node p.

Binary Attributes: Computing GINI Index

- Binary Attributes: Splits into two partitions (child nodes)
- Effect of Weighing partitions:
  - Larger and purer partitions are sought



	Parent
C1	**7**
C2	**5**
Gini = 0.486	

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		

Weighted Gini of N1 N2
= 6/12 * 0.278 +
  6/12 * 0.444
= 0.361

Gain = 0.486 – 0.361 = 0.125

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	

Which of these is the best?

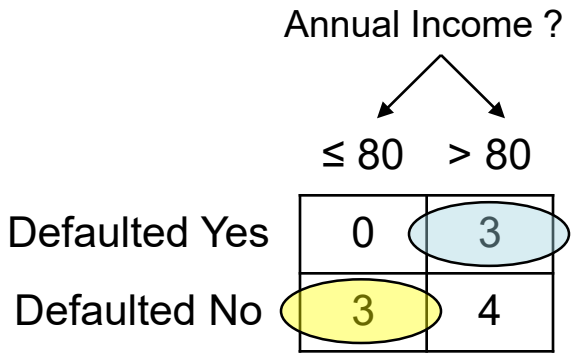
Continuous attributes: Computing Gini Index

- Two ways of handling continuous attributes
  - Discretization to form an (ordinal) categorical attribute, then use the method in last slide
    - Discretization will be introduced in later week
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more computationally intensive

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Choices for the splitting value
  - Each distinct value is a possible splitting value
- 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.

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	

Design Issues of Decision Tree Induction

- The idea is to use greedy strategy.
  - Split the records based on **an attribute test** that optimizes certain criterion.
- Issues
  - Determine how to **split** the records
    - Which attribute to split and how to determine the best split?
  - Determine when to **stop splitting**

Stopping Criteria for Tree Induction

- Many different heuristic options
- Three methods:
  - Stop expanding a node when all the records in the leaf belong to the same class, or mostly belong to one class
  - When number of examples in the leaf is too small. For example, < 100
  - Stop expanding a node when all the records have similar attribute values

Measure of Impurity: Entropy (for reference & self-study)

- Entropy at a given node $\boldsymbol{t}$

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

Where $\boldsymbol{p_i(t)}$ is the probability of class $\boldsymbol{i}$ at node $\boldsymbol{t}$, and $\boldsymbol{c}$ is the total number of classes

  - Maximum of $\log_2 c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
  - Minimum of 0 when all records belong to one class, implying most beneficial situation for classification

- Entropy based computations are quite similar to the GINI index computations

Computing Entropy of a Single Node (for reference & self-study)

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

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

$P_{(C1)}(t) = 0/6 = 0$ $P_{(C2)}(t) = 6/6 = 1$

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

$P_{(C1)}(t) = 1/6$ $P_{(C2)}(t) = 5/6$

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

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

$P_{(C1)}(t) = 2/6$ $P_{(C2)}(t) = 4/6$

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Computing Information Gain After Splitting(for reference & self-study)

- Information Gain:

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

Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

  - Choose the split that achieves most reduction (maximizes GAIN)

  - Information gain is the mutual information between the class variable and the splitting variable

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

Outline

- Decision tree
- Classification based on association rules (CBA)
- Ensemble Classifiers
- Overfitting
- Classification evaluation

Rule-Based Classifier

- Classify records by using a collection of "if…then…" rules
- Rule: $(Condition) \rightarrow y$
  - where
    - *Condition* is a conjunction of tests on attributes
    - *y* is the class label
  - Examples of classification rules:
    - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Evade=No

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Application of Rule-Based Classifier

- A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

Building Classification Rules

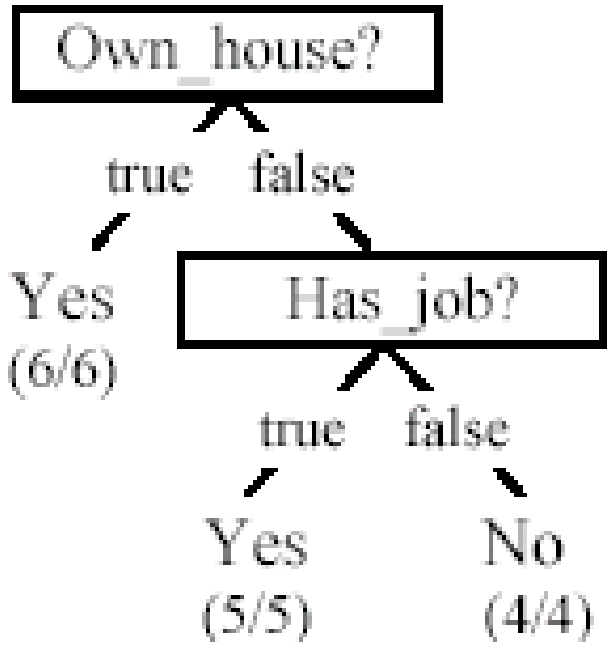
- Direct Method:
  - Extract rules directly from data
  - Examples: RIPPER, CN2, Holte's 1R, CBA
- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, etc).

Decision tree vs. rules

- **The decision tree below generates the following 3 rules.**

Own_house = true → Class =Yes [sup=6/15, conf=6/6]
Own_house = false, Has_job = true → Class=Yes [sup=5/15, conf=5/5]
Own_house = false, Has_job = false → Class=No [sup=4/15, conf=4/4]

- But there are many other rules that are not found by the decision tree



CBA: Class Association Rules

- **Association rules:** have no fixed target, but we can fix a target. target attribute: Class attribute
- Class association rules (CAR): has a target class attribute. E.g.,

  Own_house = true $\rightarrow$ Class =Yes [sup=6/15, conf=6/6]

  - CARs can obviously be used for classification.

Note that we use ratio for support in this part

Class Association Rules

Age = young, Has_job = true → Class=Yes [sup=2/15, conf=2/2]
Age = young, Has_job = false → Class=No [sup=3/15, conf=3/3]
Credit_Rating = fair → Class=No [sup=4/15, conf=4/4]
Credit_Rating = good → Class=Yes [sup=5/15, conf=5/6]

and many more, if we use minsup = 2/15 = 13.3% and minconf = 80%.

- CAR mining finds all of them.
- In many cases, rules not in the decision tree may perform classification better.

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Considerations in CAR mining

- Set different minimum class supports
  - Deal with imbalanced class distribution, e.g., some class is rare, 98% negative and 2% positive.
  - We can set the minsup(positive) = 0.2% and minsup(negative) = 2%.
- Rule pruning may be performed.
  - Pruning off less useful rules.
  - E.g., A rule a, b, c → class1 is pruned if its confidence is 60%, but a subset rule a, b → class1 has a confidence 70%.
    - a, b → class1 must have a higher support than a, b, c → class1

Example of pruning rules

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

r_1: <$(\{(A, e), (B, p)\}, 3), \big((C, y), 2\big)$>
Confidence = 2/3 = 67%
Error rate of = 33%

r_1^-: <$(\{A, e\}, 4), \big((C, y), 3\big)$>
Confidence = ¾ = 75%
Error rate of = 25%

r_1^- is a subset of r_1 and
r_1^- has a higher confidence (lower Error rate) than r_1

$\Rightarrow$

Prune r_1

Classification and Association Rule Mining

- There are many ways to build classifiers using CARs. Several existing systems available.
- Strongest rules: After CARs are mined, do nothing.
  - For each test case, we simply choose the most confident rule that covers the test case to classify it.
  - Or, using a combination of rules.
- Selecting a subset of Rules to build CBA
  - To be explained .
  - Reference: Bing Liu, Wynne Hsu, Yiming Ma:Integrating Classification and Association Rule Mining. KDD 1998: 80-86

CBA Algorithm

A CBA classifier L is of the form:

$L = <r_1, r_2, \ldots, r_k, \textit{default-class}>$

CBA algorithm consists of two parts:

1. Rule Generator (CBA-RG):
   - Generate CARs for each class.
   - A simple extension of Apriori algorithm. We have a tutorial question on it.
2. Classifier builder (CBA-CB): select a subset of rules

CBA: Rules are sorted first

Definition: Given two rules, r_i and r_j, $r_i \succ r_j$ (also called r_i precedes r_j or r_i has a higher **precedence** than r_j) if

- the confidence of r_i is greater than that of r_j, or
- their confidences are the same, but the support of r_i is greater than that of r_j

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- $CARs$ after pruning:

r_1: $A = e \rightarrow y$ $sup = 30\%$ $conf = 75\%$

r_2: $A = g \rightarrow n$ $sup = 30\%$ $conf = 60\%$

r_3: $B = p \rightarrow y$ $sup = 20\%$ $conf = 66\%$

r_4: $B = q \rightarrow y$ $sup = 30\%$ $conf = 60\%$

r_5: $B = w \rightarrow n$ $sup = 20\%$ $conf = 100\%$

r_6: $A = g, B = q \rightarrow y$ $sup = 20\%$ $conf = 66\%$

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- $CARs$ after pruning:

r_1:	$A = e \rightarrow y$	$sup = 30\%$	$conf = 75\%$
r_2:	$A = g \rightarrow n$	$sup = 30\%$	$conf = 60\%$
r_3:	$B = p \rightarrow y$	$sup = 20\%$	$conf = 66\%$
r_4:	$B = q \rightarrow y$	$sup = 30\%$	$conf = 60\%$
r_5:	$B = w \rightarrow n$	$sup = 20\%$	$conf = 100\%$
r_6:	$A = g, B = q \rightarrow y$	$sup = 20\%$	$conf = 66\%$

- Sort CARs ($\succ$Precedence):

# of attributes in left side smaller

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

CBA-CB: M1 Algorithm

CBA (R,D) Algorithm: R CARs, D training data

1. *R* = sort(*R*); //soring is done based on precedence definition
2. *RuleList* = ∅
3. **for** each rule *r* in *R* in sequence **do**
4. **If** D is not ∅ AND r classifies at least one tuple in D correctly **then**
5. delete all the training tuples covered by *r* from *D*;
6. Add *r* at the end of *RuleList*;
7. Add the majority class in the uncovered training data as the default class at the end of *RuleList*
8. compute the total number of errors of *RuleList*;
9. Find the first rule *p* in *RuleList* with the lowest total number of errors and drop all the rules after *p* in *RuleList;* add default class;

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ($\succ$Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: \quad B = w \rightarrow n$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ($\succ$Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: \quad B = w \rightarrow n$$

$$r_1: \quad A = e \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ($\succ$Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8 :

$$r_5: \quad B = w \rightarrow n$$

$$r_1: \quad A = e \rightarrow y$$

$$r_3: \quad B = p \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ($\succ$Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

$$r_5: \quad B = w \rightarrow n$$

$$r_1: \quad A = e \rightarrow y$$

$$r_6: \quad A = g, B = q \rightarrow y$$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ($\succ$Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8 :

r_5: $B = w \rightarrow n$

r_1: $A = e \rightarrow y$

r_6: $A = g, B = q \rightarrow y$

r_2: $A = g \rightarrow n$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2
2	-	-	-	-	-

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Sorted CARs ($\succ$Precedence):

$$r_5 \succ r_1 \succ r_3 \succ r_6 \succ r_2 \succ r_4$$

- Algorithm Line 3-8:

r_5: $B = w \rightarrow n$ $\quad$ r_6: $A = g, B = q \rightarrow y$

r_1: $A = e \rightarrow y$ $\quad$ r_4: $B = q \rightarrow y$

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2
2	-	-	-	-	-
4	-	-	-	-	-

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- C:

r_5: $B = w \rightarrow n$ $\quad$ r_6: $A = g, B = q \rightarrow y$

r_1: $A = e \rightarrow y$ $\quad$ ~~r_4: $B = q \rightarrow y$~~

- Algorithm Line 9:

  Find the first rule *p* in *RuleList* with the lowest total number of errors ....

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2
2	-	-	-	-	-
4	-	-	-	-	-

Example: CBA-CB (M1)

- Dataset:

Attribute A	Attribute B	Class C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

- Algorithm Line 9: default class

r	#covCases	#cCov	#wCov	defClass	#error
5	2	2	0	y	0+3=3
1	4	3	1	y/n	1+2=3
3	-	-	-	-	-
6	3	2	1	n	2+0=2

$$\Rightarrow C: \begin{cases} r_5: & B = w \rightarrow n \\ r_1: & A = e \rightarrow y \\ r_6: & A = g, B = q \rightarrow y \end{cases}$$

&

Default Class

n

Additional information on CBA algorithm

- This algorithm is CBA-CB:M1 algorithm in the original paper, which is inefficient
- CBA in the original paper has a more efficient algorithm (quite sophisticated) that scans the data at most two times.

Decision tree vs. CARs

- Association mining require discrete attributes. Decision tree learning uses both discrete and continuous attributes.
  - CAR mining requires continuous attributes discretized. There are several such algorithms (to be covered in this course).
- However, decision tree is
  - not constrained by minsup or minconf, and may find rules with very low support or confidence.

Summary

- Classification Techniques
  - Decision Tree
    - Understand the algorithm
  - Rule-Based Classifier
    - Understand the algorithm
- Next:
  - Ensemble classifiers
  - Overfitting
  - Classification evaluation