Inspire…Educate…Transform.

Association rules, Apriori algorithm, Decision trees

Dr. Manish Gupta

Sr. Mentor – Academics, INSOFE

Adapted from CS512 slides of Prof. Jiawei Han at UIUC.

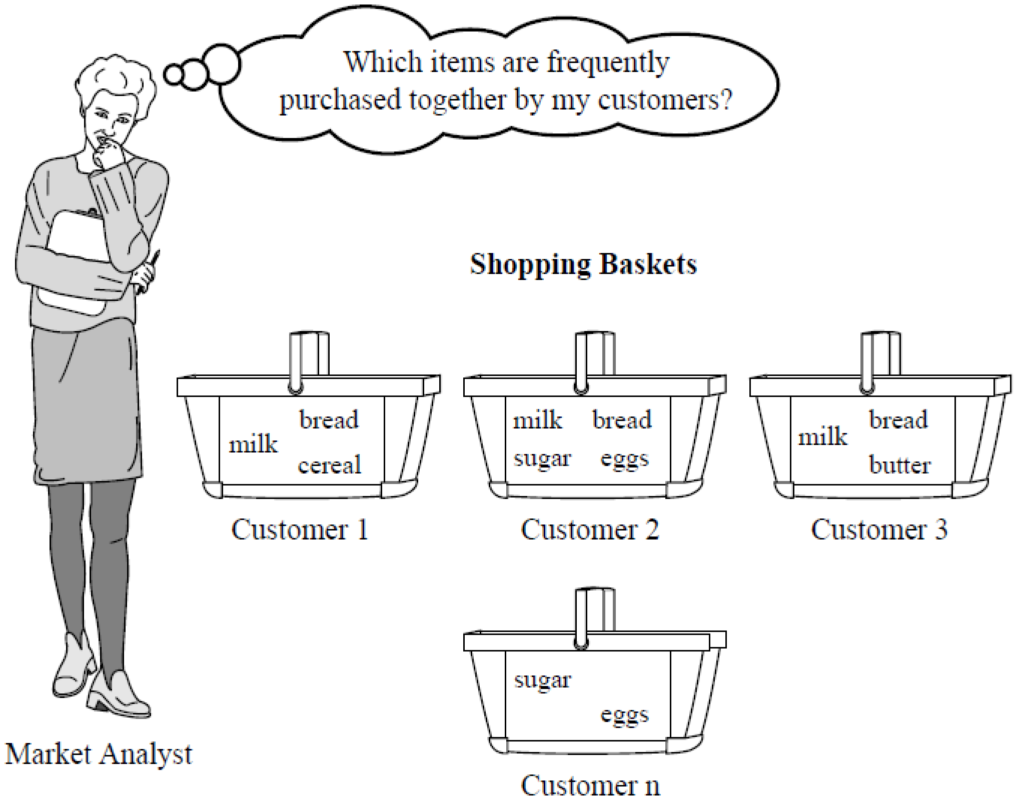
Agenda

- Association rules

What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— bread and butters?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Market Basket Analysis



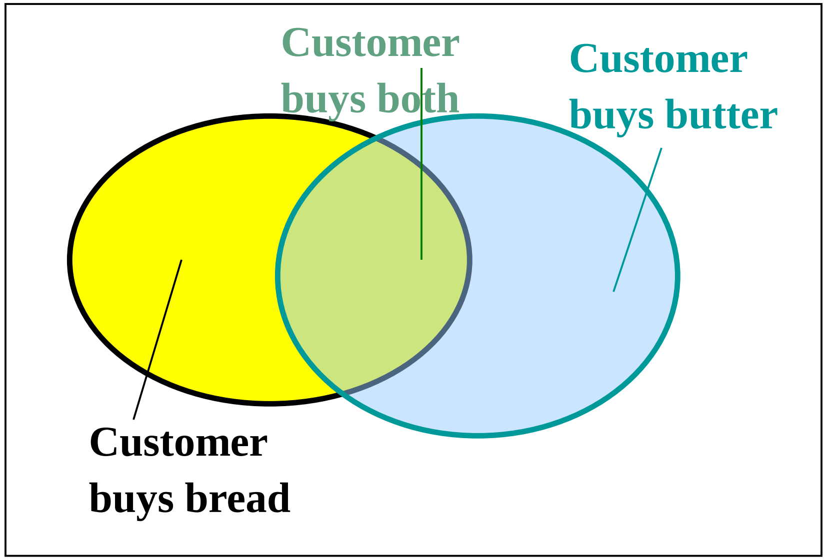
computer ⇒ *antivirus software* [*support* = 2%, *confidence* = 60%]

Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns

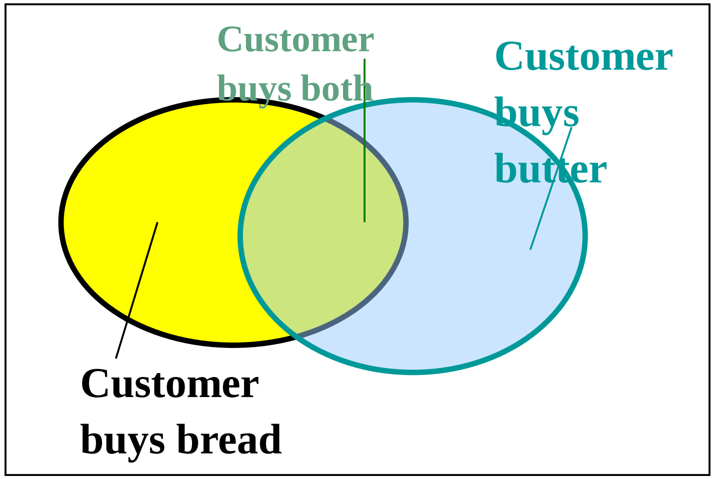
Tid	Items bought
10	bread, Nuts, butter
20	bread, Coffee, butter
30	bread, butter, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, butter, Eggs, Milk



- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

Basic Concepts: Association Rules

Tid	Items bought
10	bread, Nuts, butter
20	bread, Coffee, butter
30	bread, butter, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, butter, Eggs, Milk



- Find all the rules *X* → *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c,* conditional probability that a transaction having X also contains *Y. P(Y|X)*

Let minsup = 50%, minconf = 50%

Freq. Pat.: bread:3, Nuts:3, butter:4, Eggs:3, {bread, butter}:3

- Association rules:
  - *bread* → *butter* (60%, 100%)
  - *butter* → *bread* (60%, 75%)

CSE 7306c

Agenda

- Association rules
- Apriori

The Downward Closure Property and Scalable Mining Methods

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{bread, butter, nuts}** is frequent, so is **{bread, butter}**
  - i.e., every transaction having {bread, butter, nuts} also contains {bread, butter}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length candidate (k+1)-itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent candidates can be generated, else iterate

CSE 7306c

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan → C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

→ L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan ← C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

← L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan → L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items};

for (k = 1; L_k !=$\emptyset$; k++) **do begin**

  C_{k+1} = candidates generated from L_k;

  for each transaction t in database do

    increment the count of all candidates in C_{k+1} that are contained in t

  L_{k+1} = candidates in C_{k+1} with min_support

  end

return $\cup_k L_k$;

Implementation of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning by subset testing
    - *acde* is removed because *ade* is not in L_3
  - C_4 = {*abcd*}

Generating Candidates and Subset Testing

procedure apriori_gen(L_{k-1}:frequent $(k-1)$-itemsets)

(1) **for each** itemset $l_1 \in L_{k-1}$

(2) **for each** itemset $l_2 \in L_{k-1}$

(3) **if** $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2])$
$\wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ **then** {

(4) $c = l_1 \bowtie l_2$; // join step: generate candidates

(5) **if** has_infrequent_subset(c, L_{k-1}) **then**

(6) **delete** c; // prune step: remove unfruitful candidate

(7) **else add** c **to** C_k;

(8) }

(9) **return** C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
L_{k-1}: frequent $(k-1)$-itemsets); // use prior knowledge

(1) **for each** $(k-1)$-subset s **of** c

(2) **if** $s \notin L_{k-1}$ **then**

(3) **return** TRUE;

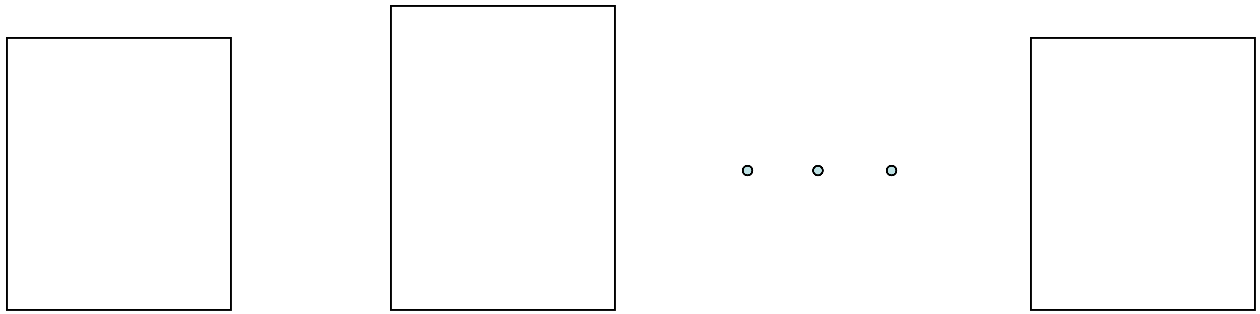
(4) **return** FALSE;

Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

Partition: Scan Database Only Twice

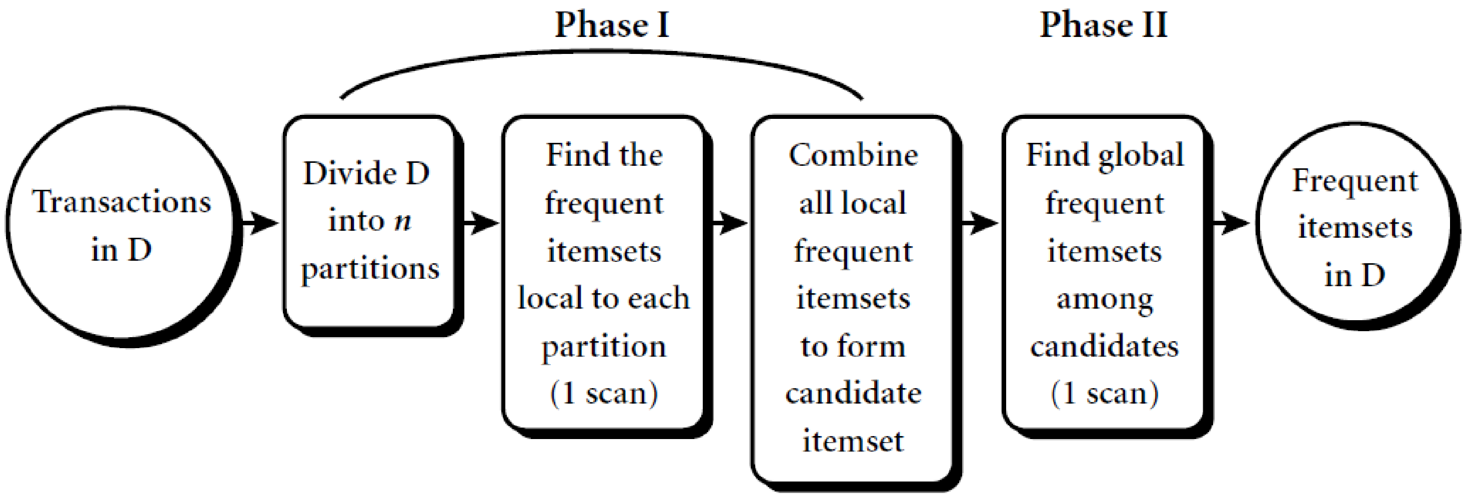
- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*



DB_1 + DB_2 + … + DB_k = DB

$sup_1(i) < \sigma DB_1$ $\quad$ $sup_2(i) < \sigma DB_2$ $\quad$ $sup_k(i) < \sigma DB_k$ $\quad$ $sup(i) < \sigma DB$

Partition: Scan Database Only Twice



- Partition size and the number of partitions are set so that each partition can fit into main memory and therefore be read only once in each phase.

Agenda

- Association rules
- Apriori
- Decision trees

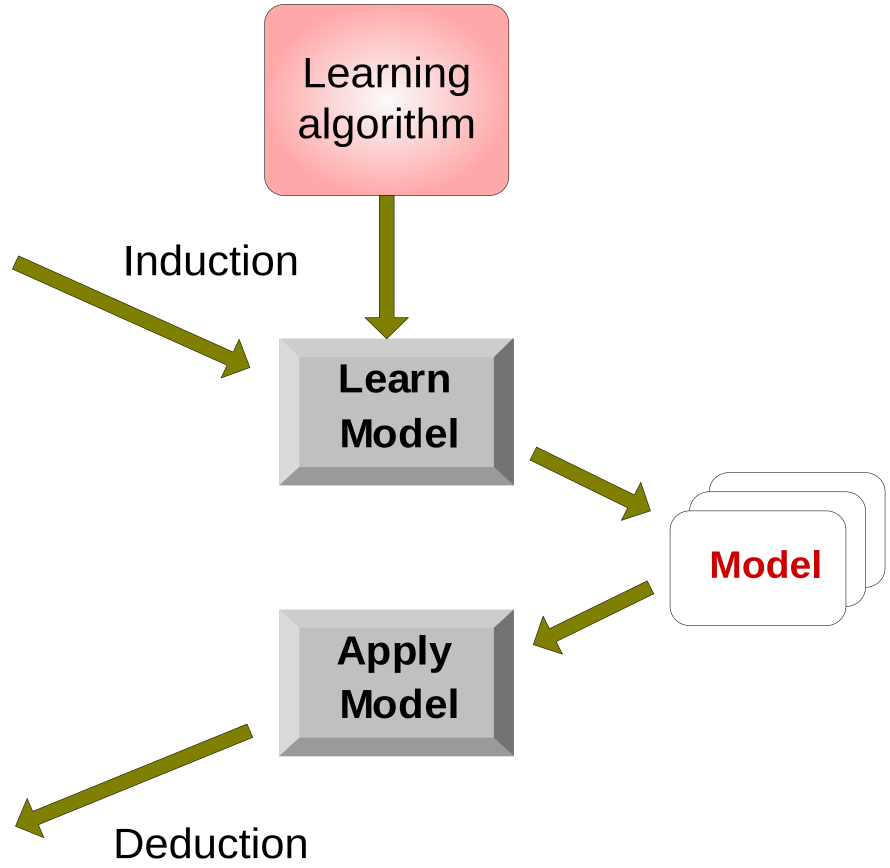
Classification: Introduction

- Given a collection of records (training set)
- Each record contains a set of attributes, one of the attributes is the class
- Find a model for class attribute as a function of the values of other attributes
- Goal: previously unseen records should be assigned a class as accurately as possible
- A test set is used to determine the accuracy of the model
  - Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it

Classification: Pictorial View

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



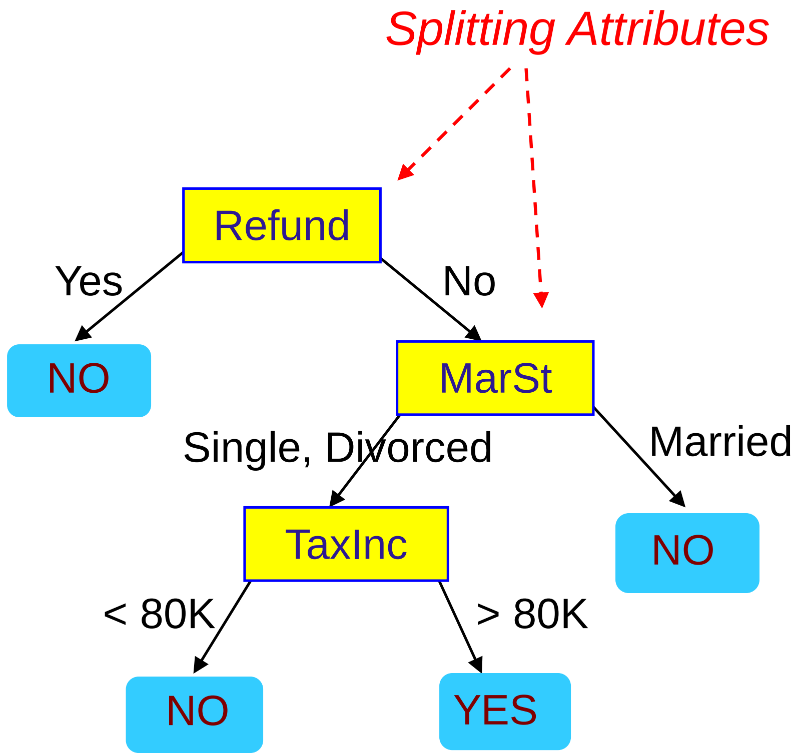
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Decision Tree Example

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (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	No	Single	90K	Yes

Training Data

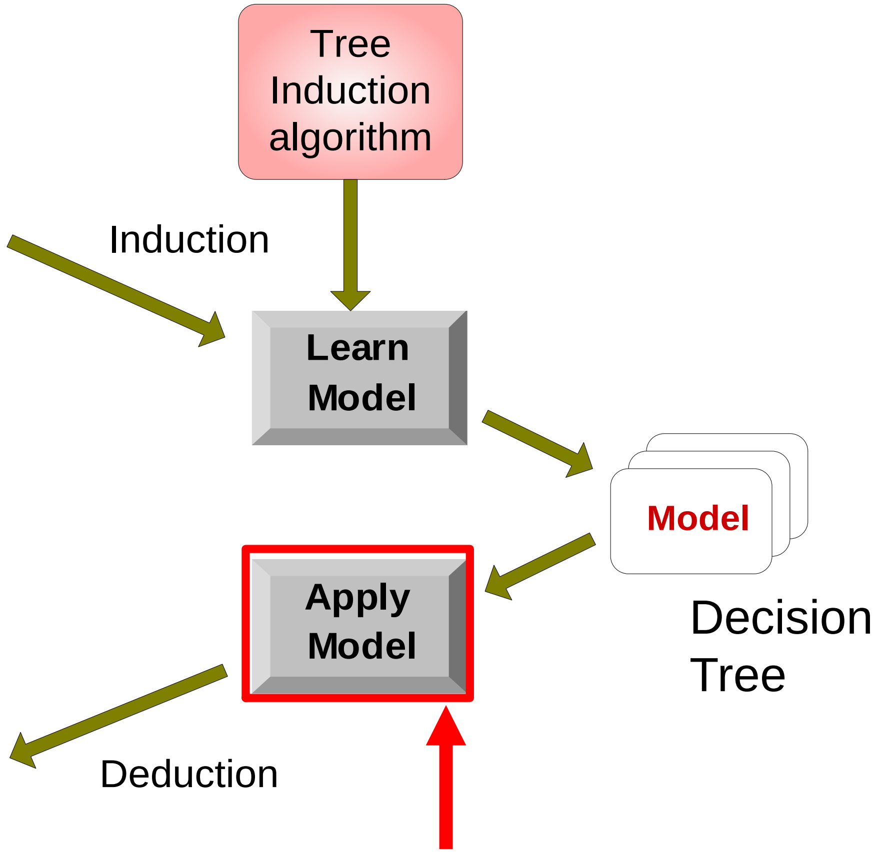


Model: Decision Tree

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

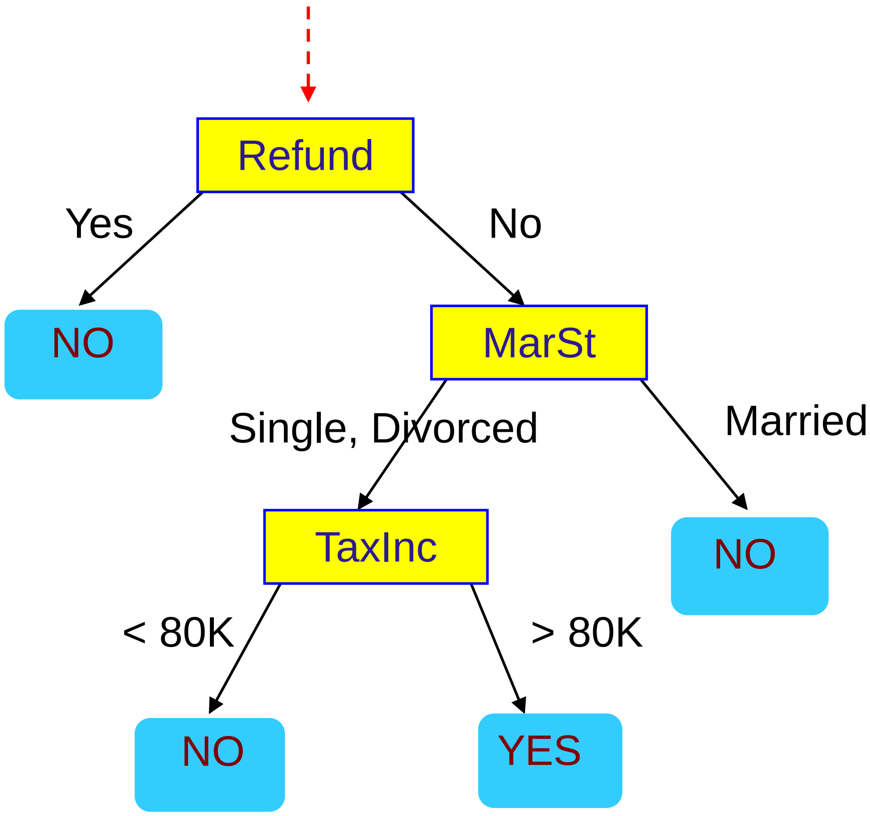


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Apply Model to Test Data

Start from the root of tree.



Test Data

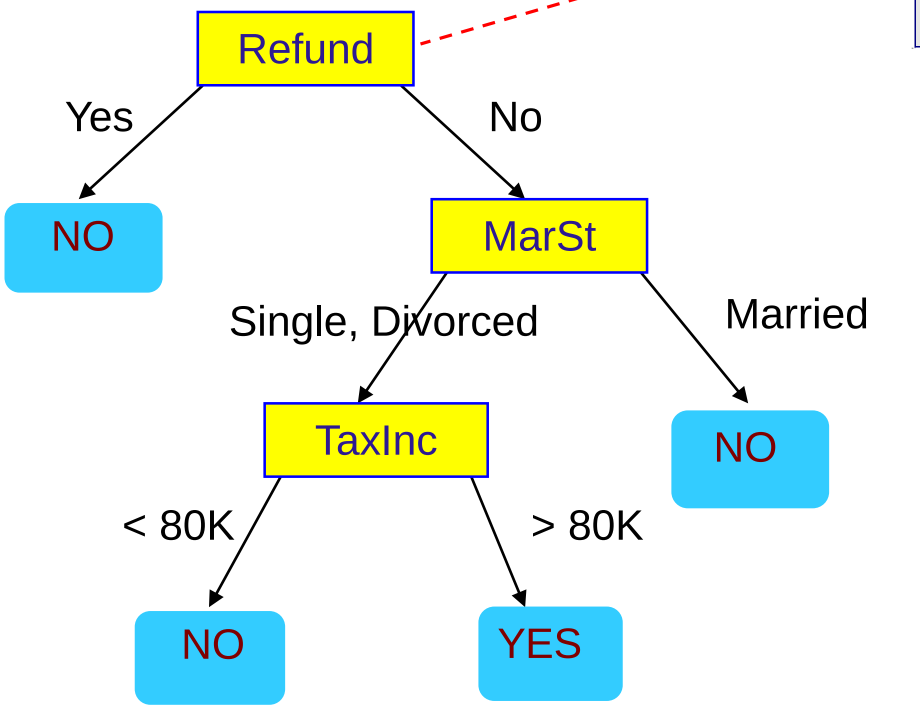
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

CSE 7306c

Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

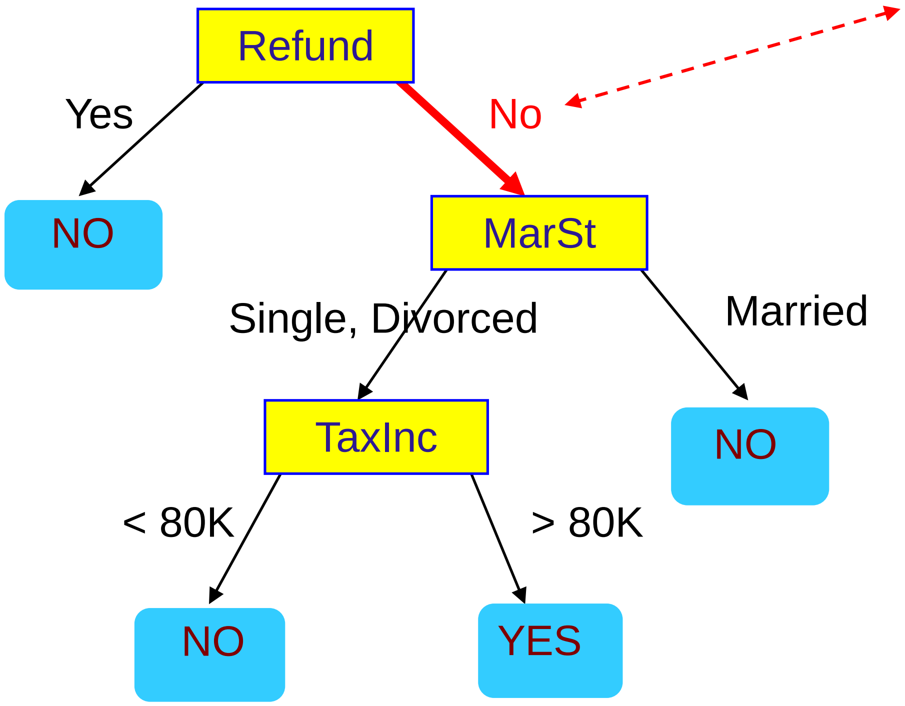


CSE 7306c

Apply Model to Test Data

Test Data

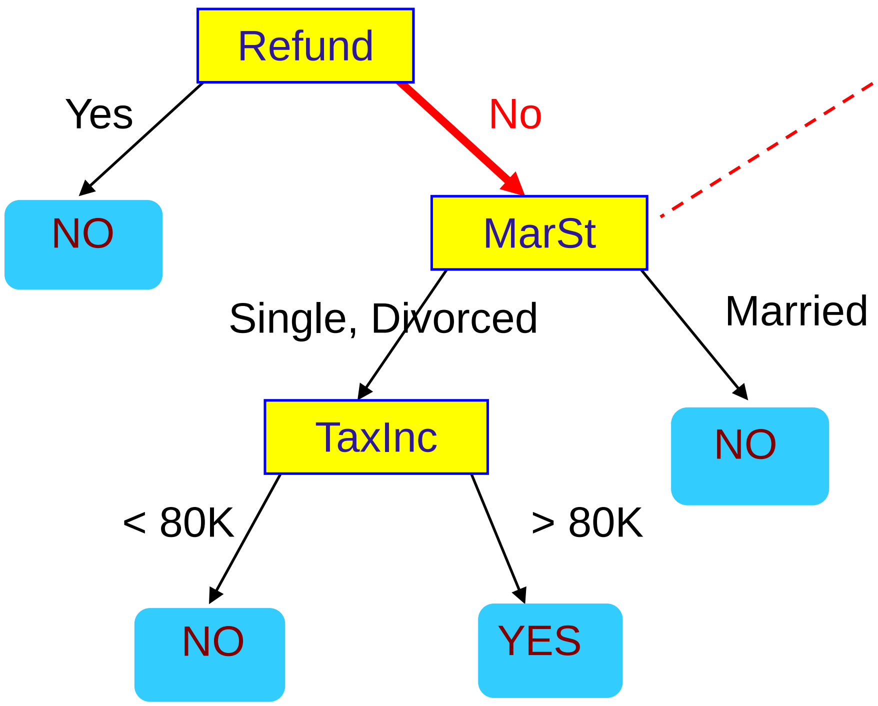
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

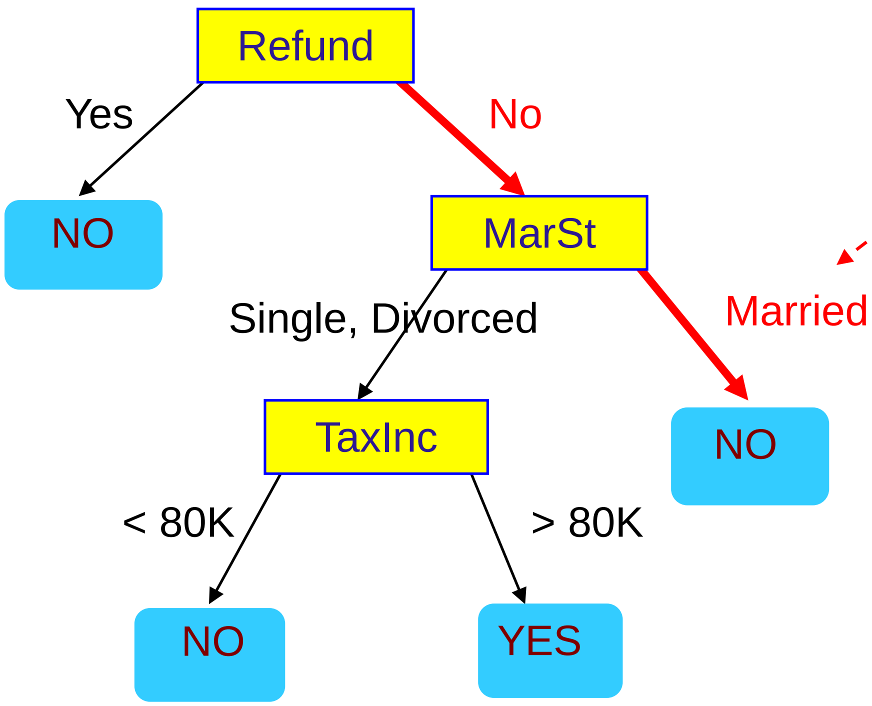
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

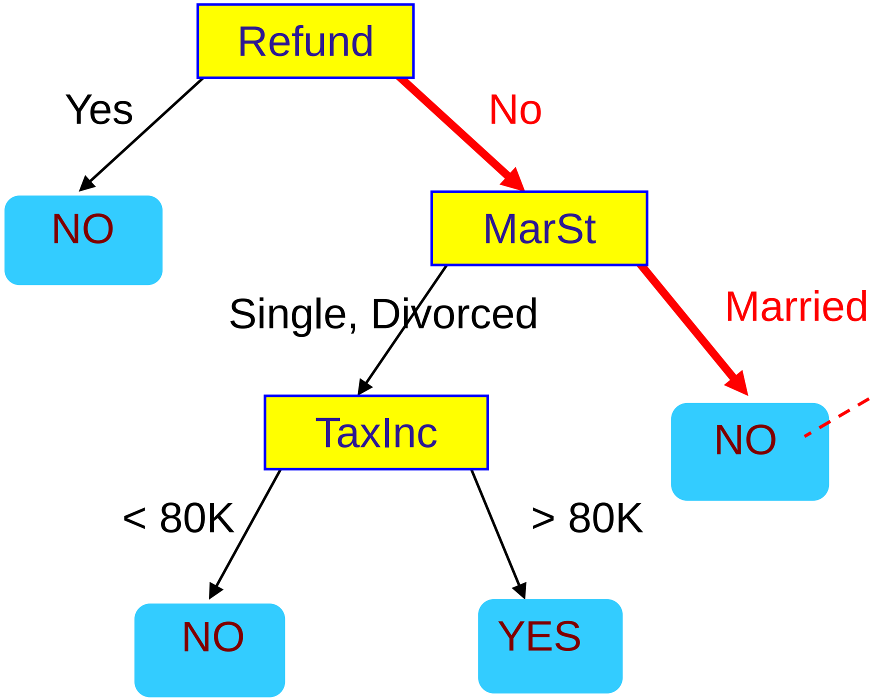
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

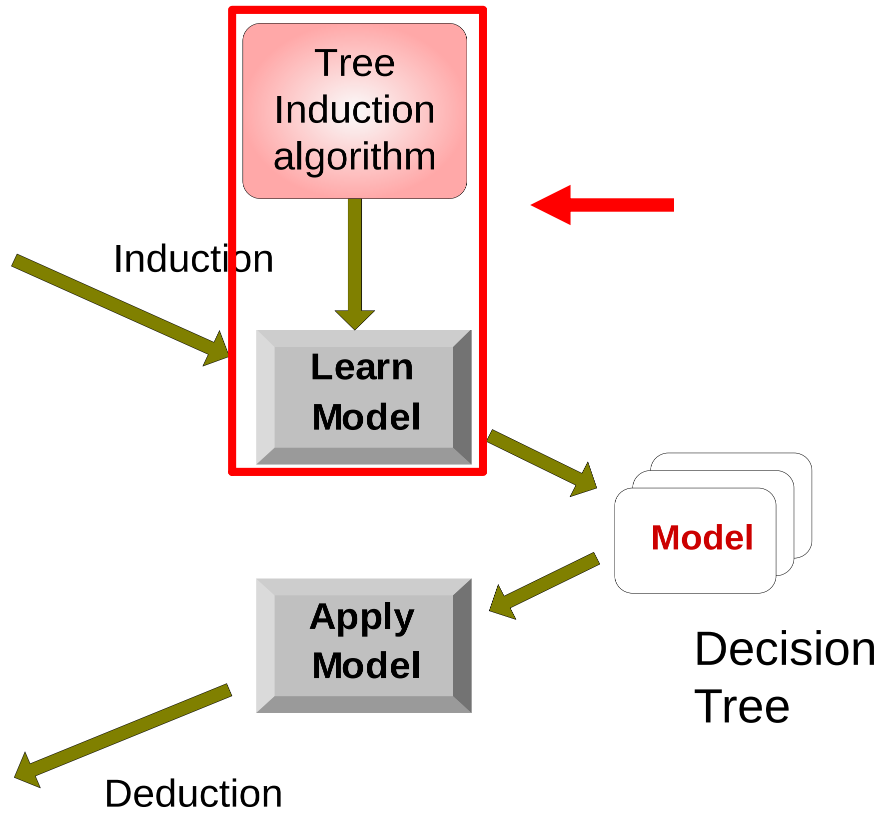


Assign Cheat to “No”

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



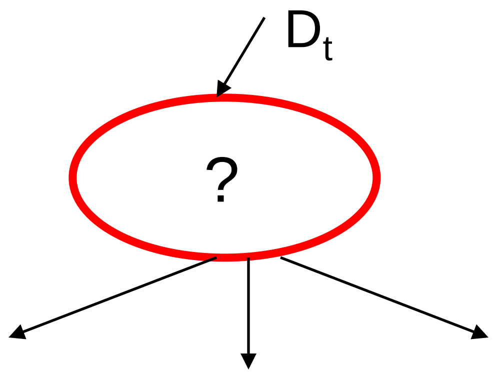
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

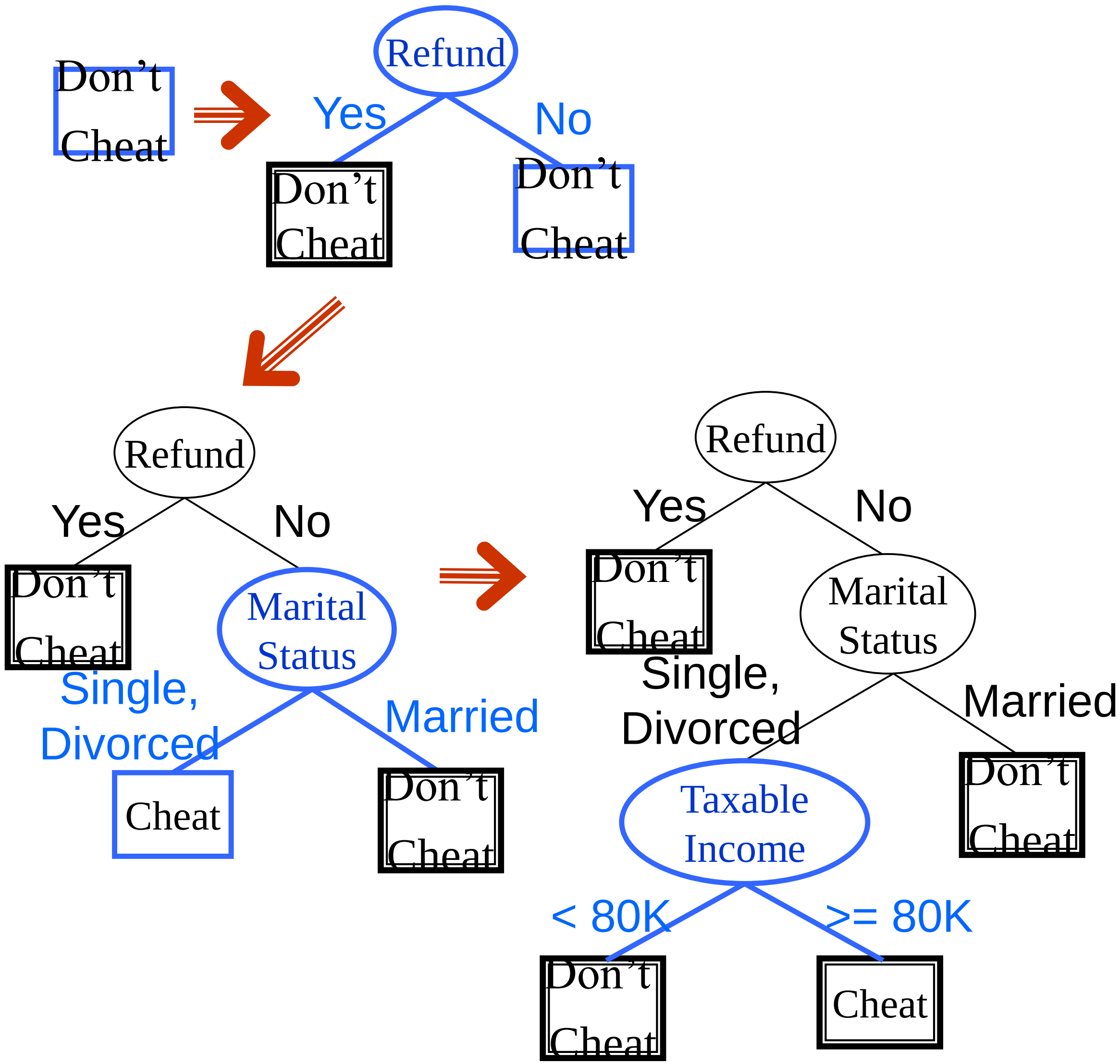
General Structure of Hunt's Algorithm

- 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.

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

CSE 7306c

Tree Induction

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

Tree Induction

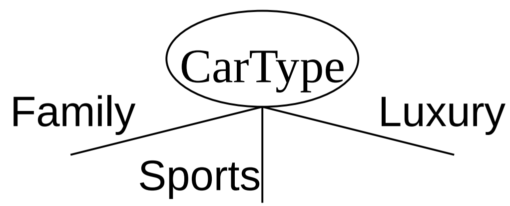
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

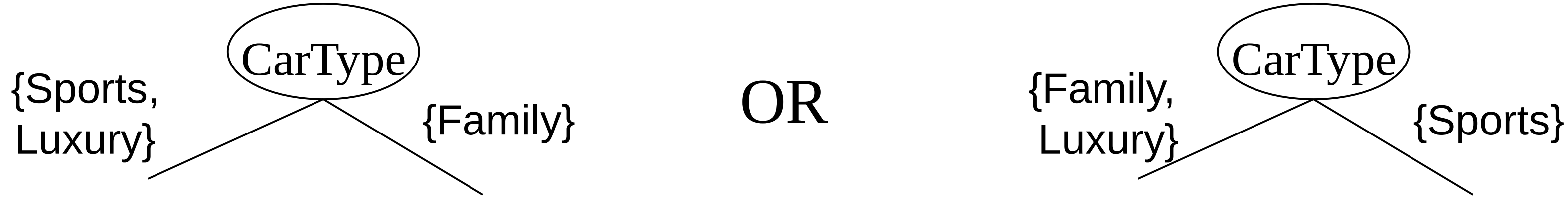
- Depends on attribute types
  - Nominal (gender, race, etc)
  - Continuous (height of people in this room)
- Depends on the number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

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



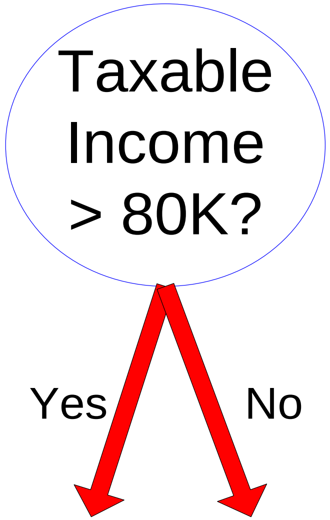
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



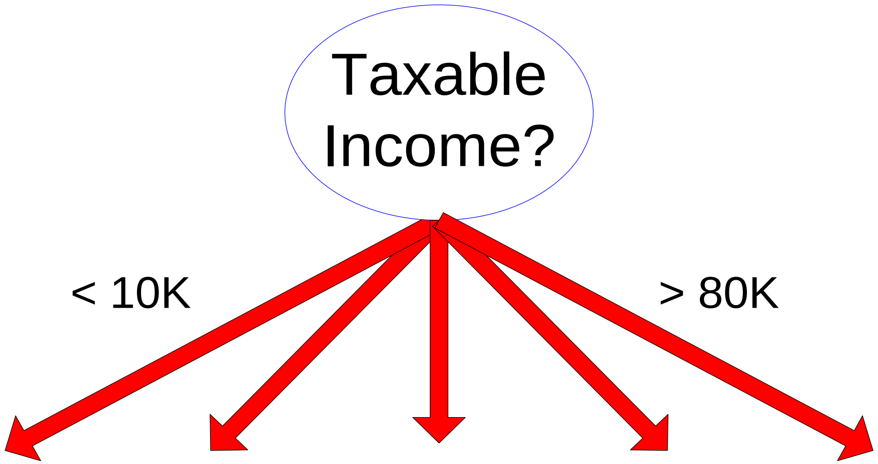
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes



(i) Binary split



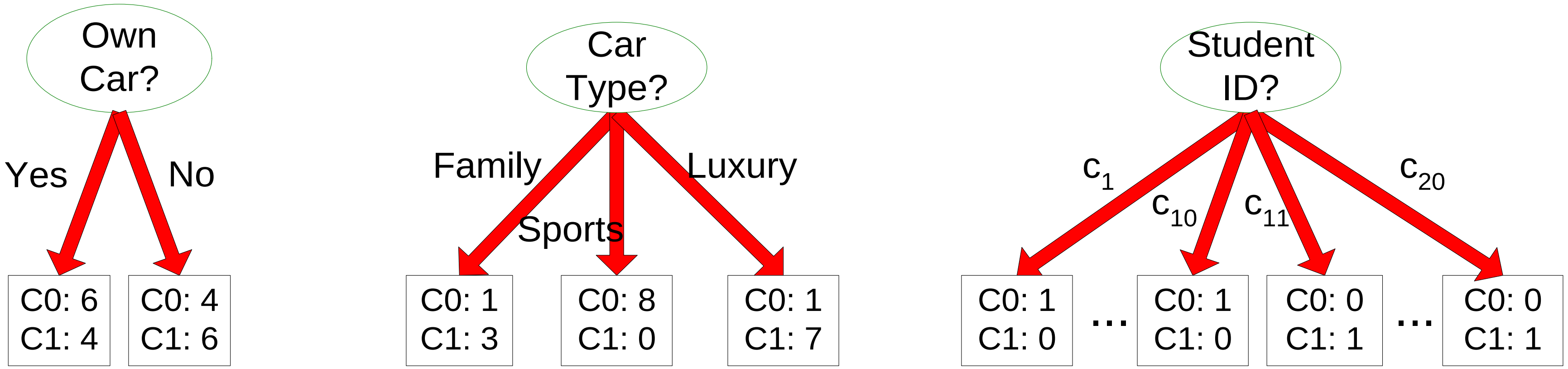
(ii) Multi-way split

Tree Induction

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

How to determine the Best Split

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



Which test condition is the best?

How to determine the Best Split

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

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

Measures of Node Impurity

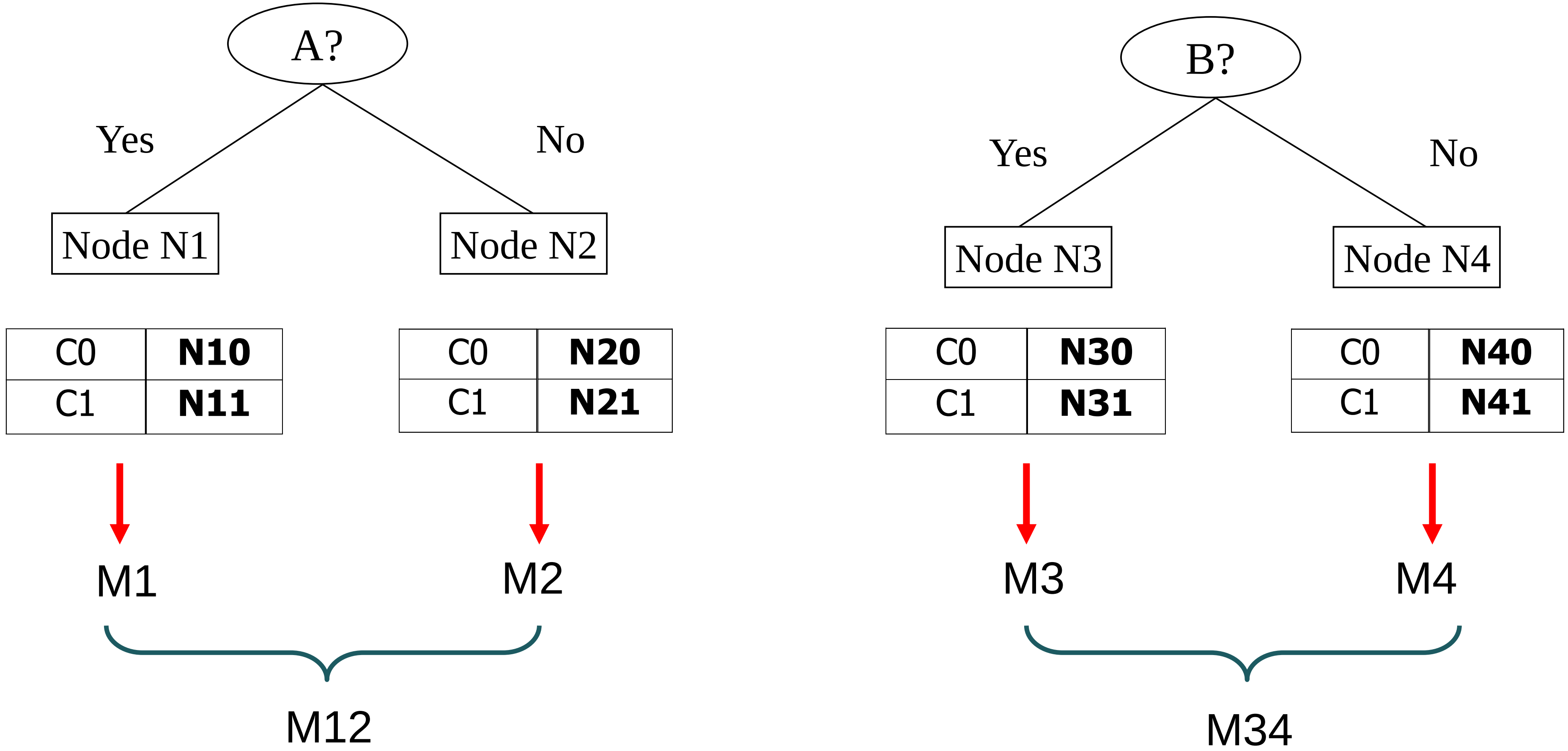
- Gini Index
- Information Gain
- Gain Ratio

How to Find the Best Split

Before Splitting:

C0	**N00**
C1	**N01**

→ M0



Node N1:

C0	**N10**
C1	**N11**

Node N2:

C0	**N20**
C1	**N21**

Node N3:

C0	**N30**
C1	**N31**

Node N4:

C0	**N40**
C1	**N41**

Gain = M0 – M12 vs M0 – M34

Measure of Impurity: GINI

- Gini Index for a given node t :

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

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

- Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

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

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

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

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

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

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

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

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

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

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

Splitting Based on GINI

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

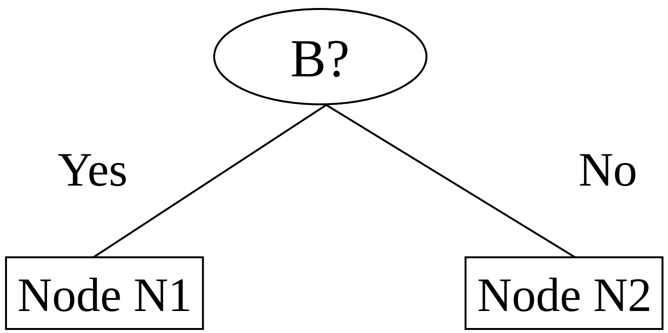
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



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

Gini(N1)
$= 1 – (5/7)^2 – (2/7)^2$
$= 0.4082$

Gini(N2)
$= 1 – (1/5)^2 – (4/5)^2$
$= 0.32$

	N1	**N2**
C1	**5**	**1**
C2	**2**	**4**
Gini=0.371		

Gini(Children)
$= 7/12 * 0.408 +$
$5/12 * 0.32$
$= 0.371$

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

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

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

- – Measures homogeneity of a node.
  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- – Entropy based computations are similar to the GINI index computations

CSE 7306c

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 0 – 1 log 1 = – 0 – 0 = 0

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

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

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

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

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

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

Splitting Based on INFO

- Information Gain:

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

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

  - Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Sample Problem

Model	Engine	SC/Turbo	Weight	Fuel Eco	Fast
Prius	small	no	average	good	no
Civic	small	no	light	average	no
WRX STI	small	yes	average	bad	yes
M3	medium	no	heavy	bad	yes
RS4	large	no	average	bad	yes
GTI	medium	no	light	bad	no
XJR	large	yes	heavy	bad	no
S500	large	no	heavy	bad	no
911	medium	yes	light	bad	yes
Corvette	large	no	average	bad	yes
Insight	small	no	light	good	no
RSX	small	no	average	average	no
IS350	medium	no	heavy	bad	no
MR2	small	yes	average	average	no
E320	medium	no	heavy	bad	no

Attribute "Model" can be tossed out, since its always unique, and it doesn't help our result.

Classification Entropy

- Calculating Classification Entropy of the entire dataset
- $I_E = -(5/15)\log_2(5/15) - (10/15)\log_2(10/15)$
- Must calculate Information Gain (IG) of remaining attributes to determine the root node.

Information Gain

- Engine: 6 small, 5 medium, 4 large
- 3 values for attribute engine, so we need 3 entropy calculations

small: 5 no, 1 yes	$I_{small} = -(5/6)\log_2(5/6)-(1/6)\log_2(1/6) = \sim 0.65$
medium: 3 no, 2 yes	$I_{medium} = -(3/5)\log_2(3/5)-(2/5)\log_2(2/5) = \sim 0.97$
large: 2 no, 2 yes	$I_{large} = 1$ (evenly distributed subset)

$$\mathbf{IG_{Engine} = IE(S) - [(6/15)*I_{small} + (5/15)*I_{medium} + (4/15)*I_{large}]}$$

$$\mathbf{IG_{Engine} = 0.971 - 0.85 = 0.121}$$

CSE 7306c

Information Gain

- SC/Turbo: 4 yes, 11 no
- 2 values for attribute SC/Turbo, so we need 2 entropy calculations

yes: 2 yes, 2 no	I_{turbo} = 1 (evenly distributed subset)
no: 3 yes, 8 no	$I_{noturbo} = -(3/11)\log_2(3/11)-(8/11)\log_2(8/11) = \sim 0.84$

$$IG_{turbo} = IE(S) - [(4/15)*I_{turbo} + (11/15)*I_{noturbo}]$$

IG_{turbo} = **0.971 – 0.886 = 0.085**

CSE 7306c

Information Gain

- Weight: 6 Average, 4 Light, 5 Heavy
- 3 values for attribute weight, so we need 3 entropy calculations

average: 3 no, 3 yes	$I_{average} = 1$ (evenly distributed subset)
light: 3 no, 1 yes	$I_{light} = -(3/4)\log_2(3/4)-(1/4)\log_2(1/4) = \sim 0.81$
heavy: 4 no, 1 yes	$I_{heavy} = -(4/5)\log_2(4/5)-(1/5)\log_2(1/5) = \sim 0.72$

$$\mathbf{IG_{Weight} = IE(S) - [(6/15)*I_{average} + (4/15)*I_{light} + (5/15)*I_{heavy}]}$$

IG_{Weight} = **0.971 – 0.856 = 0.115**

CSE 7306c

Information Gain

- Fuel Economy: 2 good, 3 average, 10 bad
- 3 values for attribute Fuel Eco, so we need 3 entropy calculations

good: 0 yes, 2 no	I_{good} = 0 (no variability)
average: 0 yes, 3 no	$I_{average}$ = 0 (no variability)
bad: 5 yes, 5 no	I_{bad} = 1 (evenly distributed subset)

We can omit calculations for good and average since they always end up not fast.

$\mathbf{IG_{FuelEco} = IE(S) - [(10/15)*I_{bad}]}$

$IG_{FuelEco}$ = **0.971 – 0.667 = 0.304**

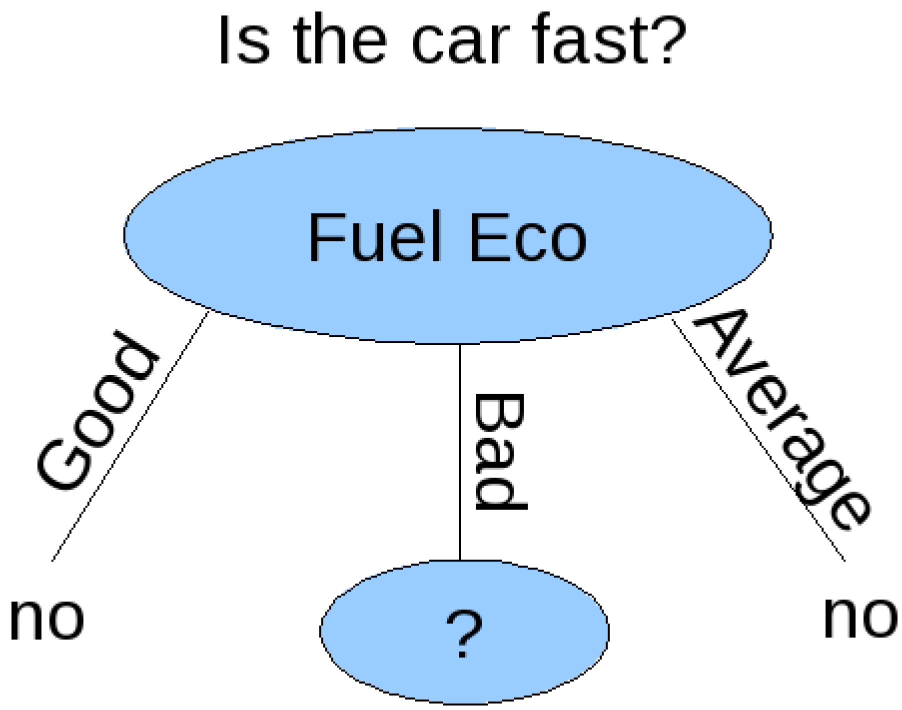
Choosing the Root Node

IG_{Engine}	0.121
IG_{turbo}	0.085
IG_{Weight}	0.115
$IG_{FuelEco}$	0.304

Our best pick is Fuel Eco, and we can immediately predict the car is not fast when fuel economy is good or average.

CSE 7306c

Root of Decision Tree



Finding Next Level Split

- Since we selected the Fuel Eco attribute for our Root Node, it is removed from the table for future calculations.

Engine	SC/Turbo	Weight	Fast
small	yes	average	yes
medium	no	heavy	yes
large	no	average	yes
medium	no	light	no
large	yes	heavy	no
large	no	heavy	no
medium	yes	light	yes
large	no	average	yes
medium	no	heavy	no
medium	no	heavy	no

- Calculating for Entropy I_E(Fuel Eco) we get 1, since we have 5 yes and 5 no.

Finding Next Level Split

- Engine: 1 small, 5 medium, 4 large
- 3 values for attribute engine, so we need 3 entropy calculations

small: 1 yes, 0 no	I_{small} = 0 (no variability)
medium: 2 yes, 3 no	$I_{medium} = -(2/5)\log_2(2/5)-(3/5)\log_2(3/5) = \sim 0.97$
large: 2 no, 2 yes	I_{large} = 1 (evenly distributed subset)

$$\mathbf{IG_{Engine} = IE(S_{FuelEco}) - (5/10)*I_{medium} + (4/10)*I_{large}]}$$

$$IG_{Engine} = \mathbf{1 - 0.885 = 0.115}$$

CSE 7306c

Finding Next Level Split

- SC/Turbo: 3 yes, 7 no
- 2 values for attribute SC/Turbo, so we need 2 entropy calculations

yes: 2 yes, 1 no	$I_{turbo} = -(2/3)\log_2(2/3)-(1/3)\log_2(1/3) = \sim 0.84$
no: 3 yes, 4 no	$I_{noturbo} = -(3/7)\log_2(3/7)-(4/7)\log_2(4/7) = \sim 0.84$

$$\mathbf{IG_{turbo} = IE(S_{FuelEco}) - [(3/10)*I_{turbo} + (7/10)*I_{noturbo}]}$$

$$IG_{turbo} = \mathbf{1 - 0.965 = 0.035}$$

Finding Next Level Split

- Weight: 3 average, 5 heavy, 2 light
- 3 values for attribute weight, so we need 3 entropy calculations

average: 3 yes, 0 no	$I_{average}$ = 0 (no variability)
heavy: 1 yes, 4 no	$I_{heavy} = -(1/5)\log_2(1/5)-(4/5)\log_2(4/5) = \sim 0.72$
light: 1 yes, 1 no	I_{light} = 1 (evenly distributed subset)

$$\mathbf{IG_{Engine} = IE(S_{Fuel\ Eco}) - [(5/10)*I_{heavy}+(2/10)*I_{light}]}$$

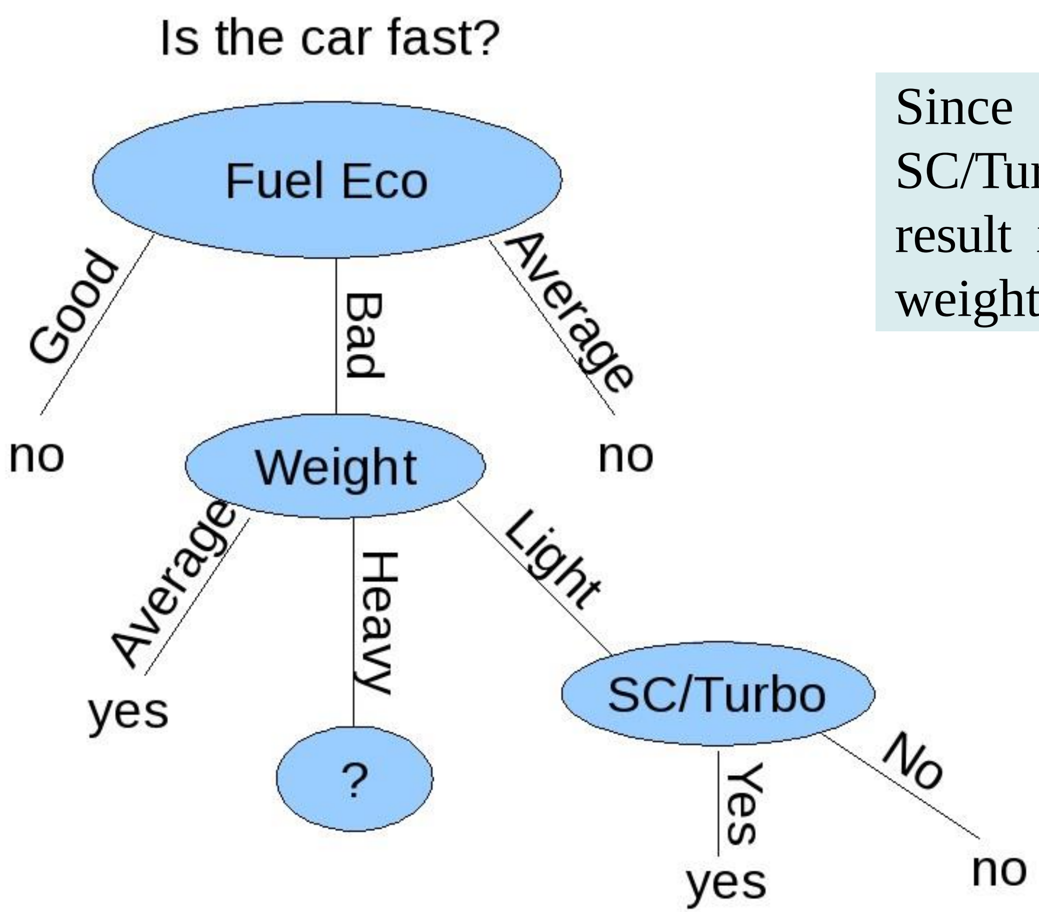
$$IG_{Engine} = \mathbf{1 - 0.561 = 0.439}$$

Choosing the Level 2 Node

IG_{Engine}	0.115
IG_{turbo}	0.035
IG_{Weight}	0.439

Weight has the highest gain, and is thus the best choice.

Decision Tree Now



Since there are only two items for SC/Turbo where Weight = Light, and the result is consistent, we can simplify the weight = Light path.

Final Updated Table

Engine	SC/Turbo	Fast
medium	no	yes
large	yes	no
large	no	no
medium	no	no
medium	no	no

Except Engine, all Attributes have been used. However, all cars with large engines in this table are not fast. But, the algorithm terminates here.

CSE 7306c

Splitting Based on INFO

- Gain Ratio:

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

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

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

Handling Numeric Attributes

- Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- temperature < 71.5: yes/4, no/2
- temperature $\geq$ 71.5: yes/5, no/3
- Info([4,2],[5,3])

  = 6/14 info([4,2]) + 8/14 info([5,3])

  = 0.939 bits

- Place split points halfway between values
- Can evaluate all split points in one pass!

Avoid repeated sorting!

- Sort instances by the values of the numeric attribute
  - Time complexity for sorting: O (n log n)
- Q. Does this have to be repeated at each node of the tree?
- A: No! Sort order for children can be derived from sort order for parent
  - Time complexity of derivation: O (n)
  - Drawback: need to create and store an array of sorted indices for each numeric attribute

More speed up!

- Entropy only needs to be evaluated between points of different classes (Fayyad & Irani, 1992)

value	64	65	68	69	70	71	72	72	75	75	80	81	83	85
class	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

X

- Potential optimal breakpoints

- Breakpoints between values of the same class cannot be optimal

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

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
  - Min number of nodes per leaf
  - Max depth of tree
  - No further correlation between attributes and class label

Ref: Basic Concepts of Frequent Pattern Mining

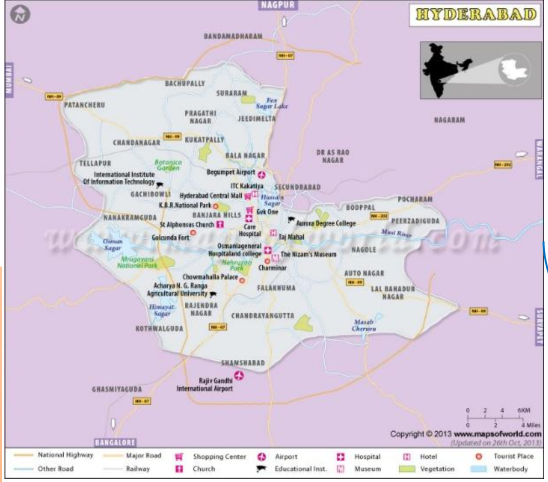
- (Association Rules) R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93.
- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- (Closed-pattern) N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95

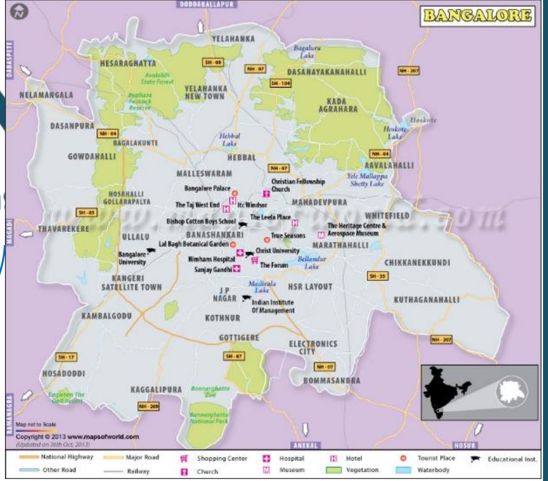
Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

Further Reading Decision trees

- Tools
  - Weka: http://www.cs.waikato.ac.nz/ml/weka/
- Chapter 3: Tom Mitchell. Machine Learning Book
- Chapter 6: Data Mining Concepts and Techniques. Jiawei Han and Micheline Kamber. Second edition
- Research papers
  - Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers
  - Friedman, J. H. (1999). *Stochastic gradient boosting.* Stanford University. http://www.sciencedirect.com/science/article/pii/S0167947301000652
- Survey paper: http://rd.springer.com/article/10.1023/A:1009744630224





HYDERABAD

Office and Classrooms

Plot 63/A, Floors 1&2, Road # 13, Film Nagar,
Jubilee Hills, Hyderabad - 500 033
+91-9701685511 (Individuals)
+91-9618483483 (Corporates)

BENGALURU

Office

Incubex, #728, Grace Platina, 4th Floor, CMH Road,
Indira Nagar, 1st Stage, Bengaluru – 560038
+91-9502334561 (Individuals)
+91-9502799088 (Corporates)

Classroom

KnowledgeHut Solutions Pvt. Ltd., Reliable Plaza,
Jakkasandra Main Road, Teacher's Colony, 14th Main
Road, Sector – 5, HSR Layout, Bengaluru - 560102

Social Media

Web: http://www.insofe.edu.in
Facebook: https://www.facebook.com/insofe
Twitter: https://twitter.com/Insofeedu
YouTube: http://www.youtube.com/InsofeVideos
SlideShare: http://www.slideshare.net/INSOFE
LinkedIn: http://www.linkedin.com/company/international-school-of-engineering