Inspire…Educate…Transform.

# Decision Tree Learning

**Dr. Kishore Reddy Konda**

Mentor, INSOFE

# Decision Tree Learning

**The Task**

- Given: collection of examples (x, f(x))
- Return: a function *h* (*hypothesis*) that approximates *f*
- *h is a* ***decision tree***

**Input:** an object or situation described by a set of attributes (or features)

**Output:** a "decision" – the predicted output value for the input

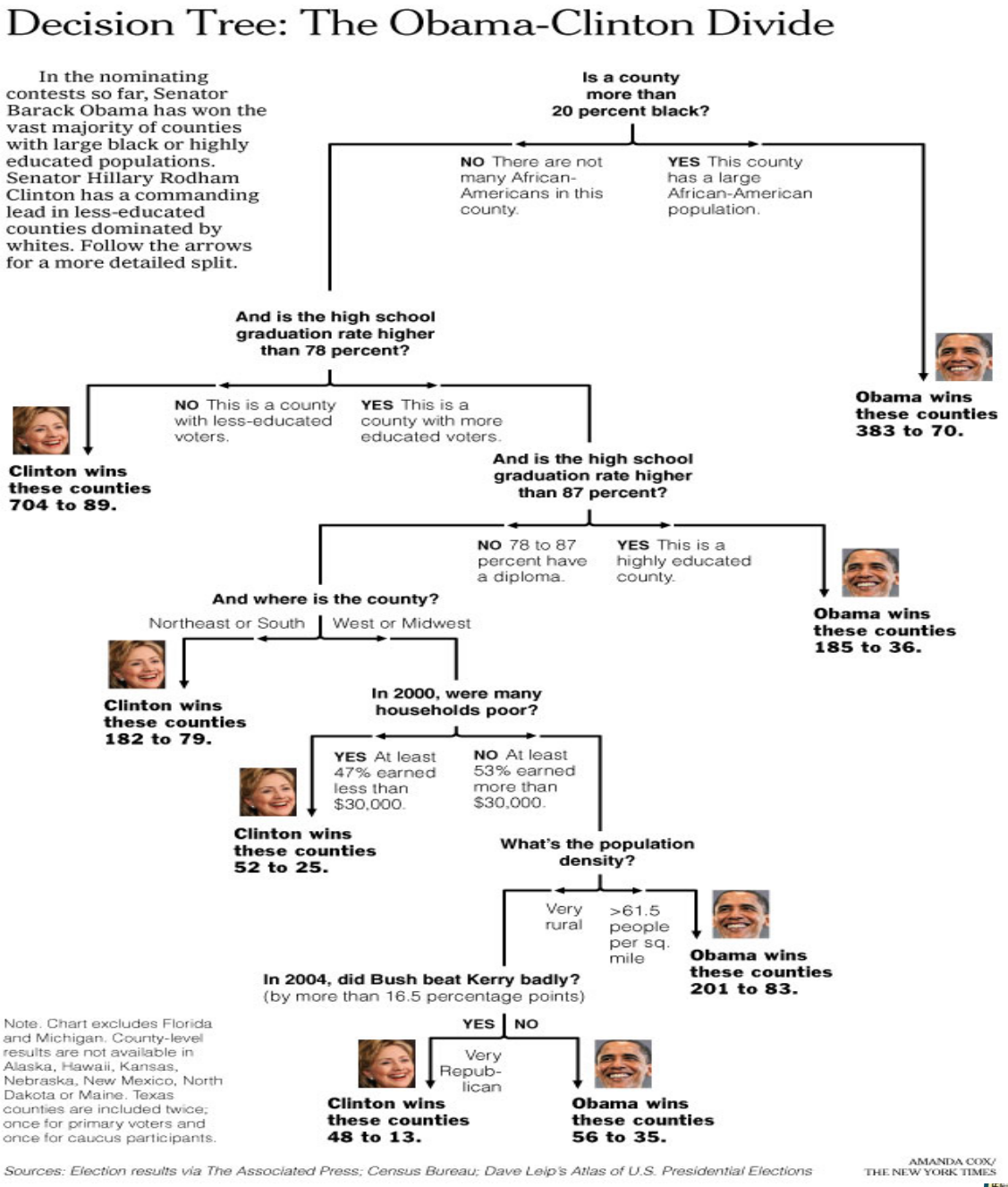
The input attributes and the outputs can be discrete or continuous.

# Learning how US counties voted?

**New York Times Article**
**April 16, 2008**

## Decision Tree Model

- A sequence of tests
- Representation very natural for humans to interpret
- Style of many “How To” manuals and trouble-shooting procedures.




CSE 7306c

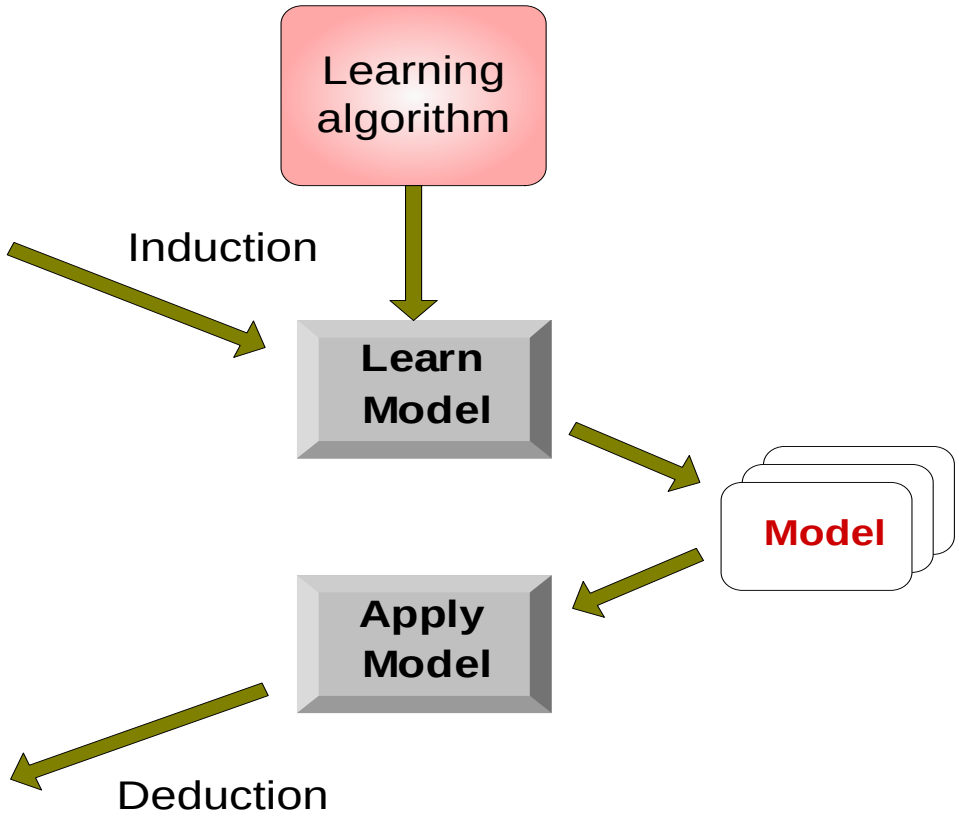
# Decision Tree Learning – Pictorial View

| 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

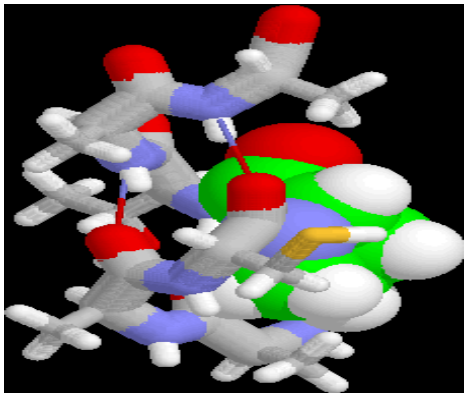
| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# Sample Applications

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc

# Classification vs. Clustering

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set

- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

# Classification vs. Regression

- Supervised Learning
  - Classification
    - Response Variable (class) has only two categories
    - Response Variable (class) has multiple categories
    - E.g., CORRECT/WRONG (sometimes expressed as 0,1)
  - Regression
    - Response Variable is continuous
    - E.g., Stock price of MSFT tomorrow

# Evaluation Metrics for Classification

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix

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

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

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

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

# Limitation of Accuracy

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

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

# Cost-Sensitive Metrics for Classification

| Cost Matrix | PREDICTED CLASS | | |
|---|---|---|---|
| | C(i\|j) | + | - |
| ACTUAL CLASS | + | -1 | 100 |
| | - | 1 | 0 |

| Model $M_1$ | PREDICTED CLASS | | |
|---|---|---|---|
| | | + | - |
| ACTUAL CLASS | + | 150 | 40 |
| | - | 60 | 250 |

Accuracy = 80%

Cost = 3910

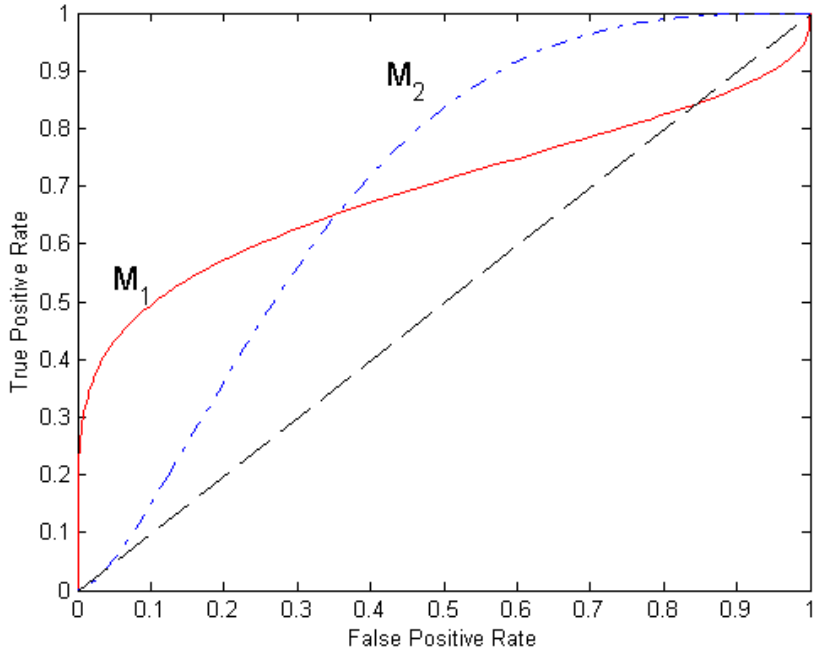
| Model $M_2$ | PREDICTED CLASS | | |
|---|---|---|---|
| | | + | - |
| ACTUAL CLASS | + | 250 | 45 |
| | - | 5 | 200 |

Accuracy = 90%

Cost = 4255

# ROC (Receiver Operating Characteristic) Curve

- ROC curve plots TP (on the y-axis) against FP (on the x-axis)



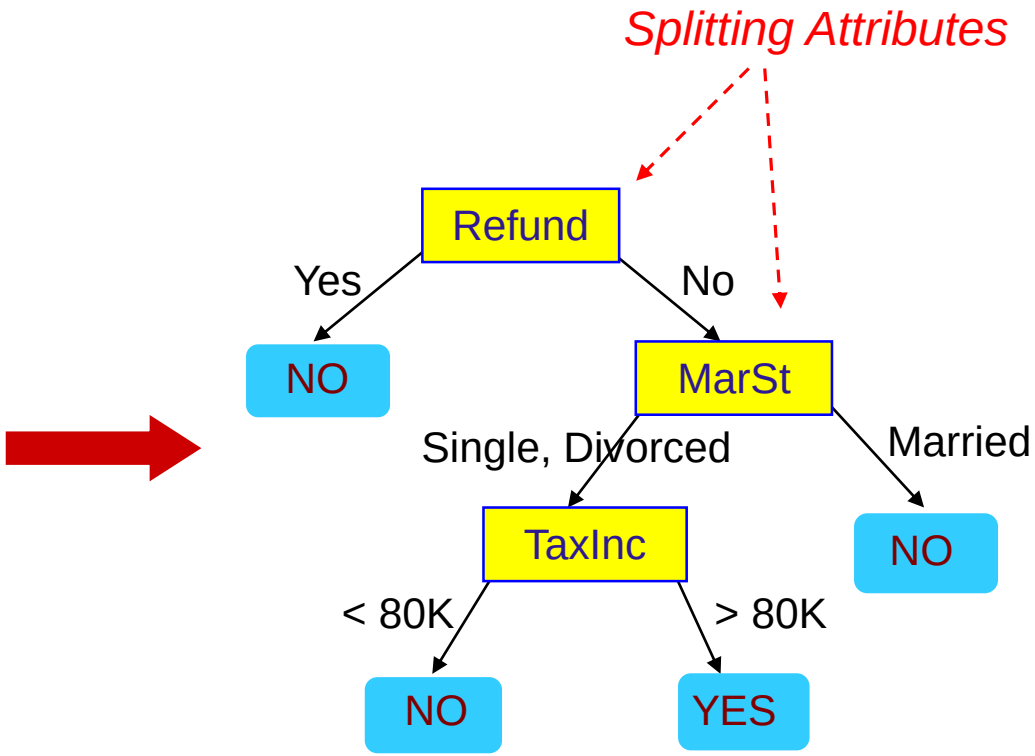

- $M_1$ is better for small FPR
- $M_2$ is better for large FPR
- Area Under the ROC curve
  - Ideal: Area = 1
  - Random guess = 0.5

- **Constructing an ROC Curve**
  - Rank the test tuples in decreasing order of P(class==+|X)
  - Apply threshold at each unique value of P(class==+|X)
  - Count the number of TP, FP, TN, FN at each threshold
  - TP rate, TPR = TP/(TP+FN)
  - FP rate, FPR = FP/(FP + TN)

# Decision Tree for Classification – An 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

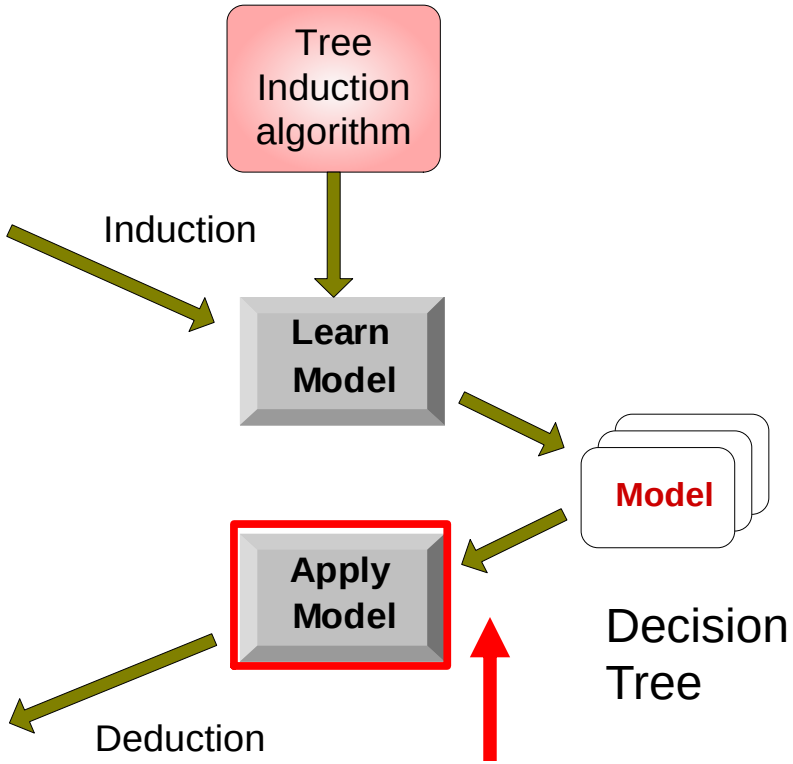
# Decision Tree (DT) for Classification – An Example

| 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

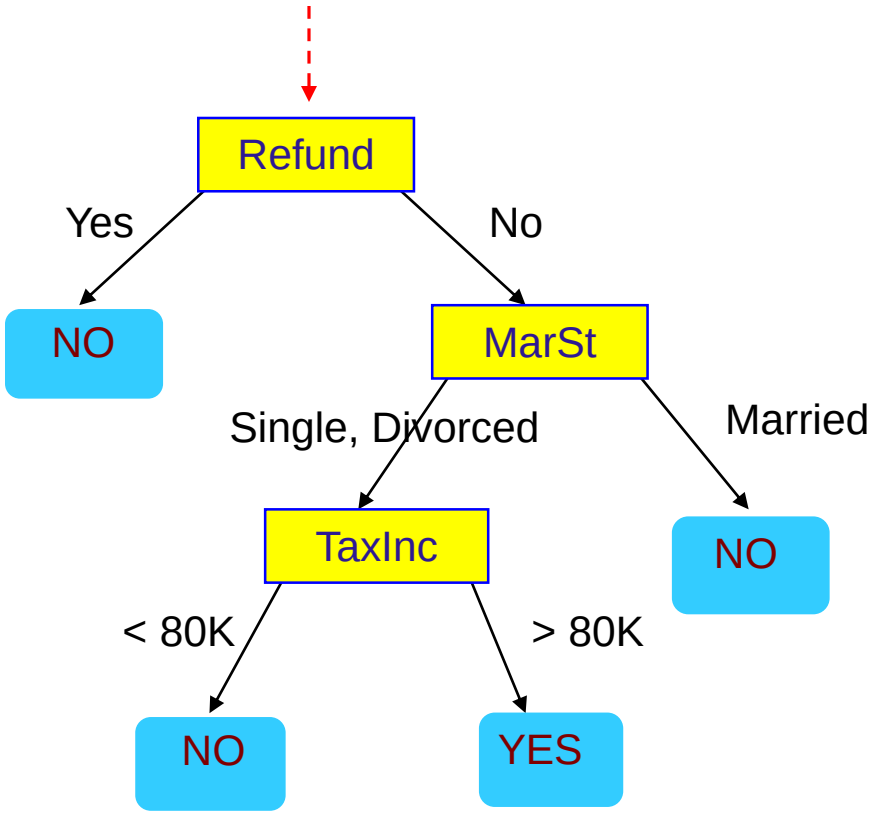
| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# Apply DT Model to Test Data

Start from the root of tree.



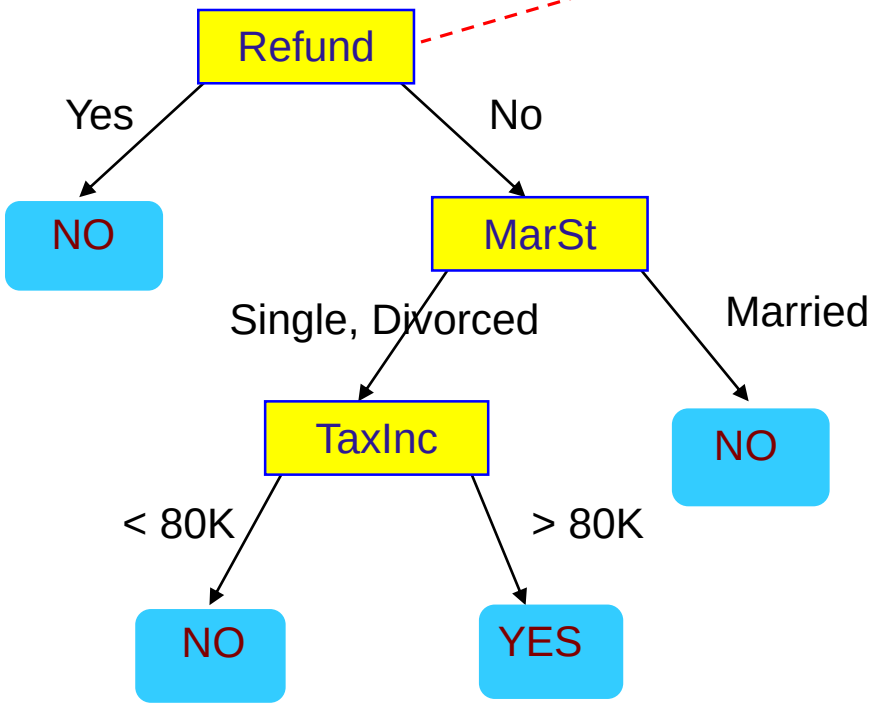

Test Data

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

# Apply DT Model to Test Data

Test Data

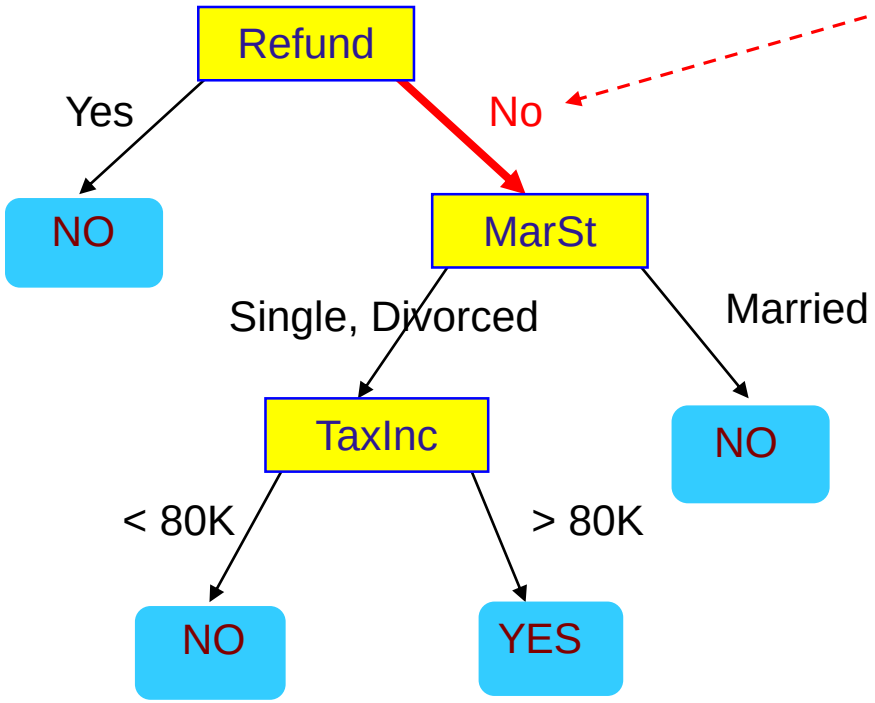
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply DT Model to Test Data

Test Data

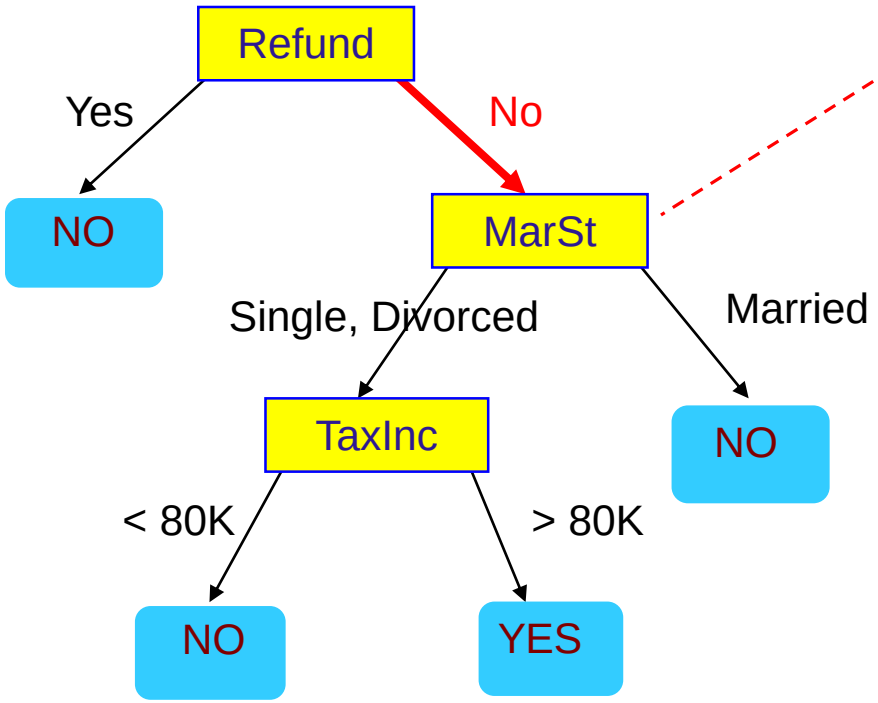
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply DT Model to Test Data

Test Data

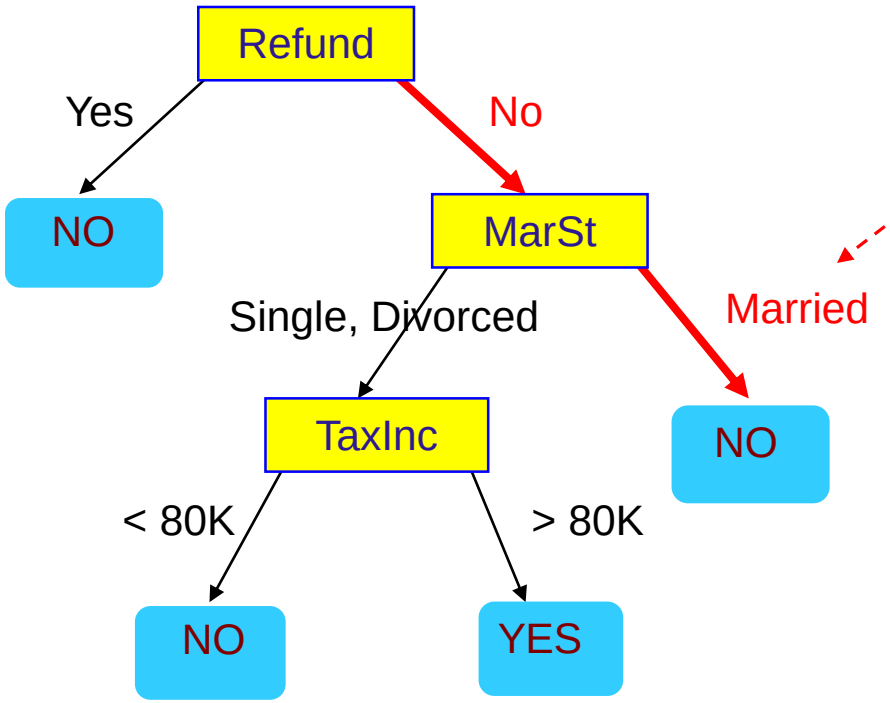
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



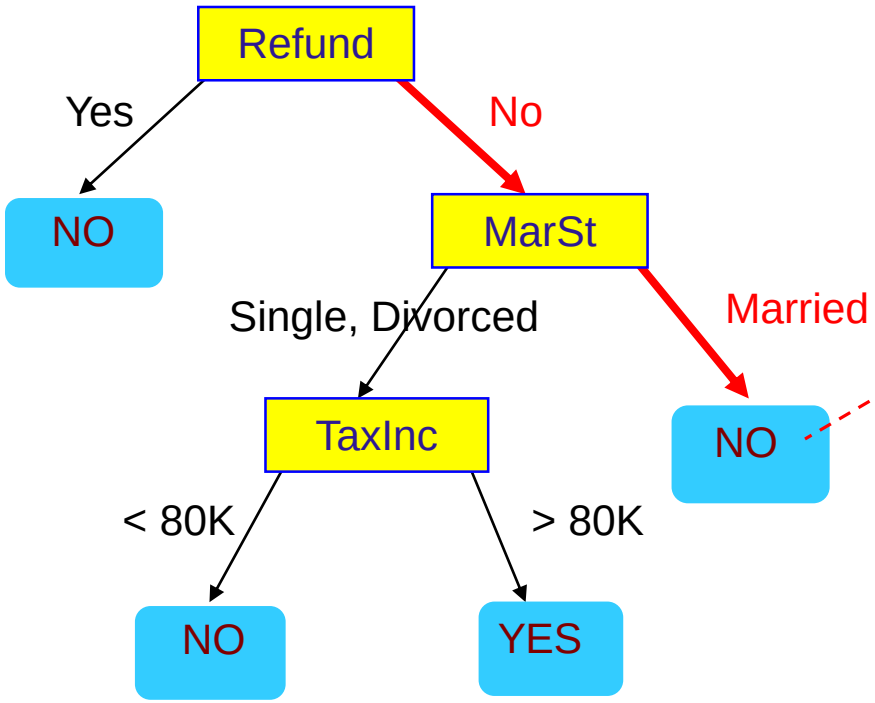
# Apply DT Model to Test Data

Test Data

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



# Apply DT Model to Test Data




Test Data

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

Assign Cheat to “No”

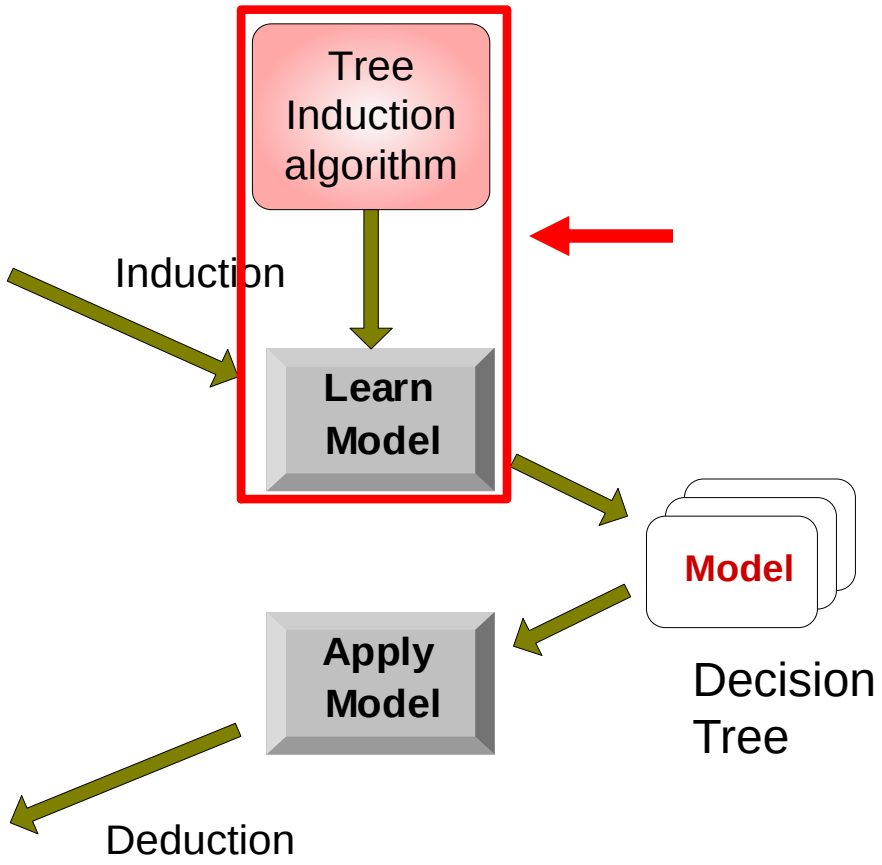
# Decision Tree Learning Task

| 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 | ? |

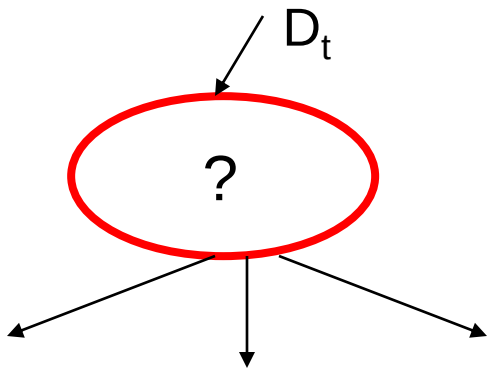
Test Set



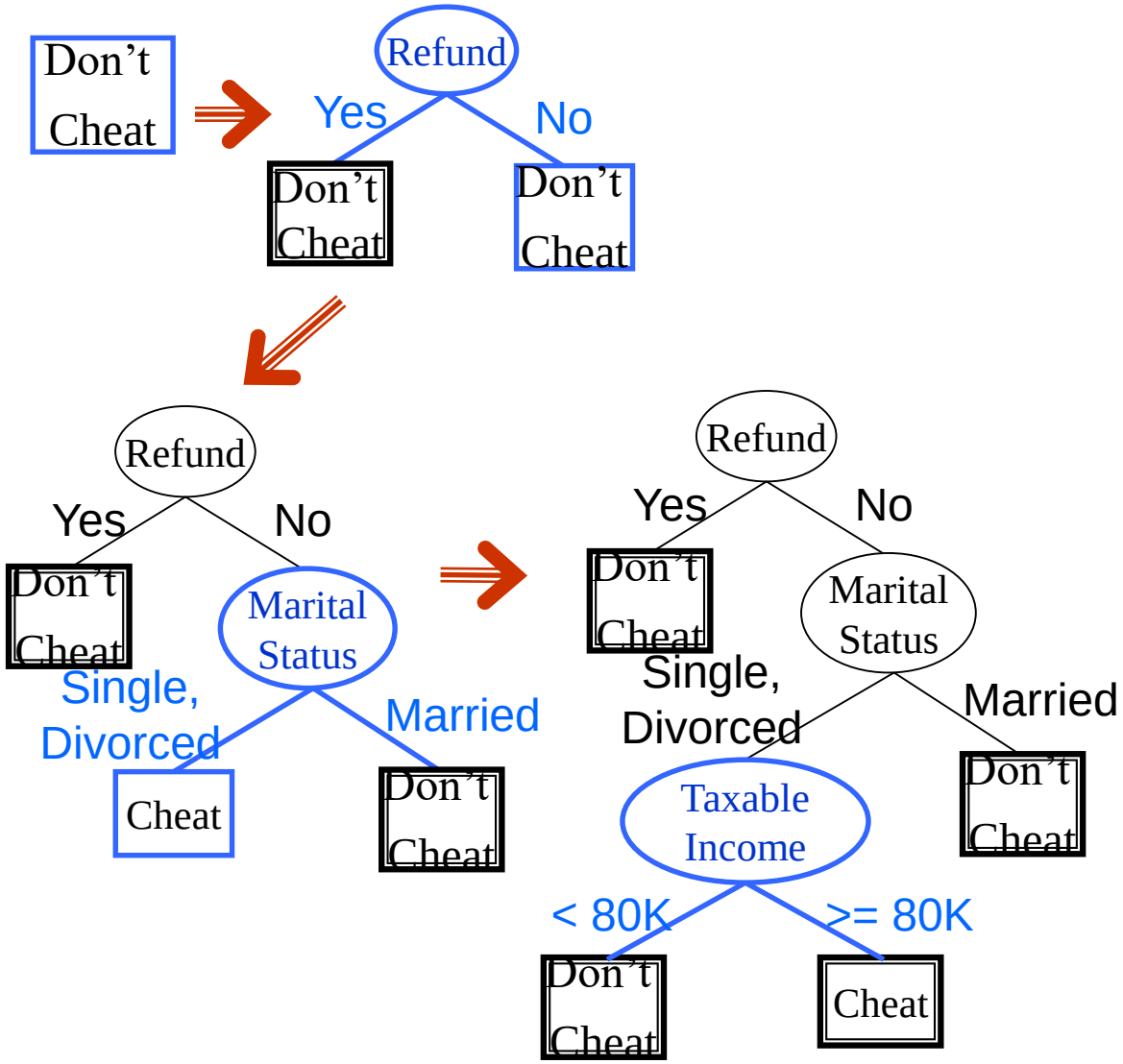
# General Structure of DT Learning

- Greedy Top-Down Algorithm – 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 |

# Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes certain ***criterion of "class purity"***
- 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 of "class purity"***
- 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

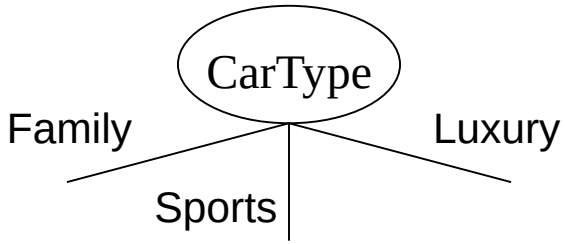
# How to specify test condition?

- Depends on attribute types
  - Nominal (gender, race, etc)
  - Continuous (height of people in this room)

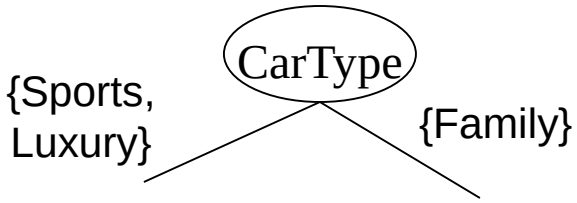
- Depends on the number of ways to split
  - 2-way split
  - Multi-way split

# Splitting Nominal Attributes

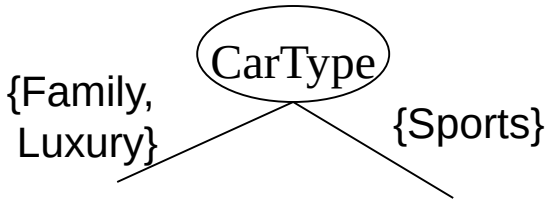
- Multi-way split: Use as many partitions as distinct values.




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



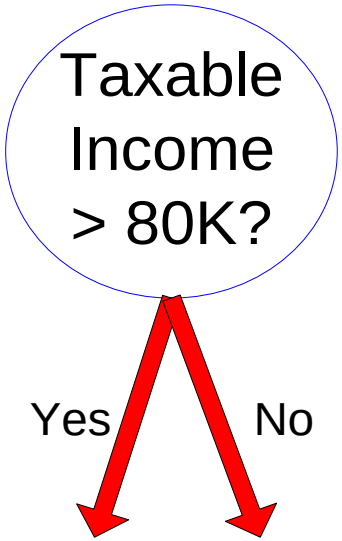

OR



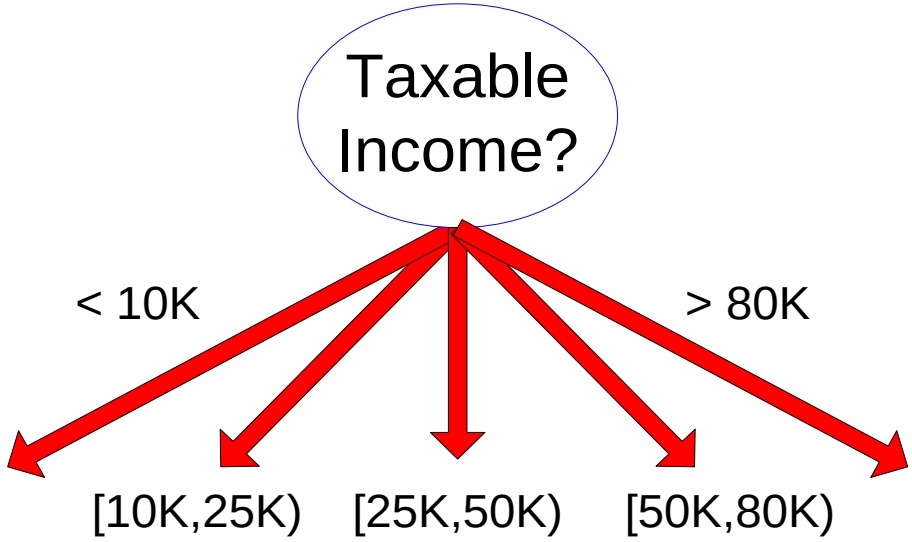
# Splitting Continuous Attributes

- 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 Continuous Attributes




(i) Binary split



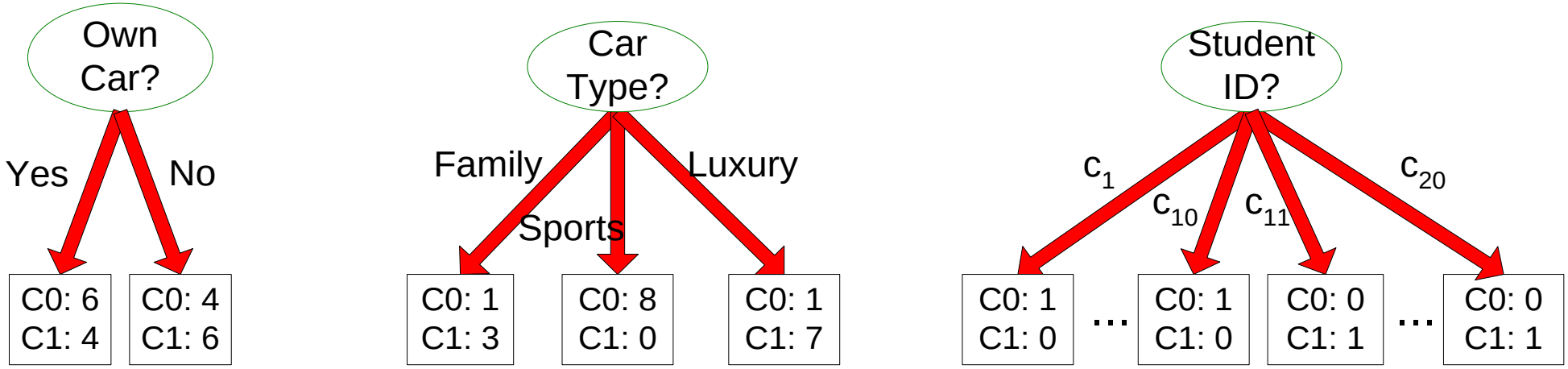

(ii) Multi-way split

# Tree Induction

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

# How to determine the best split?

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



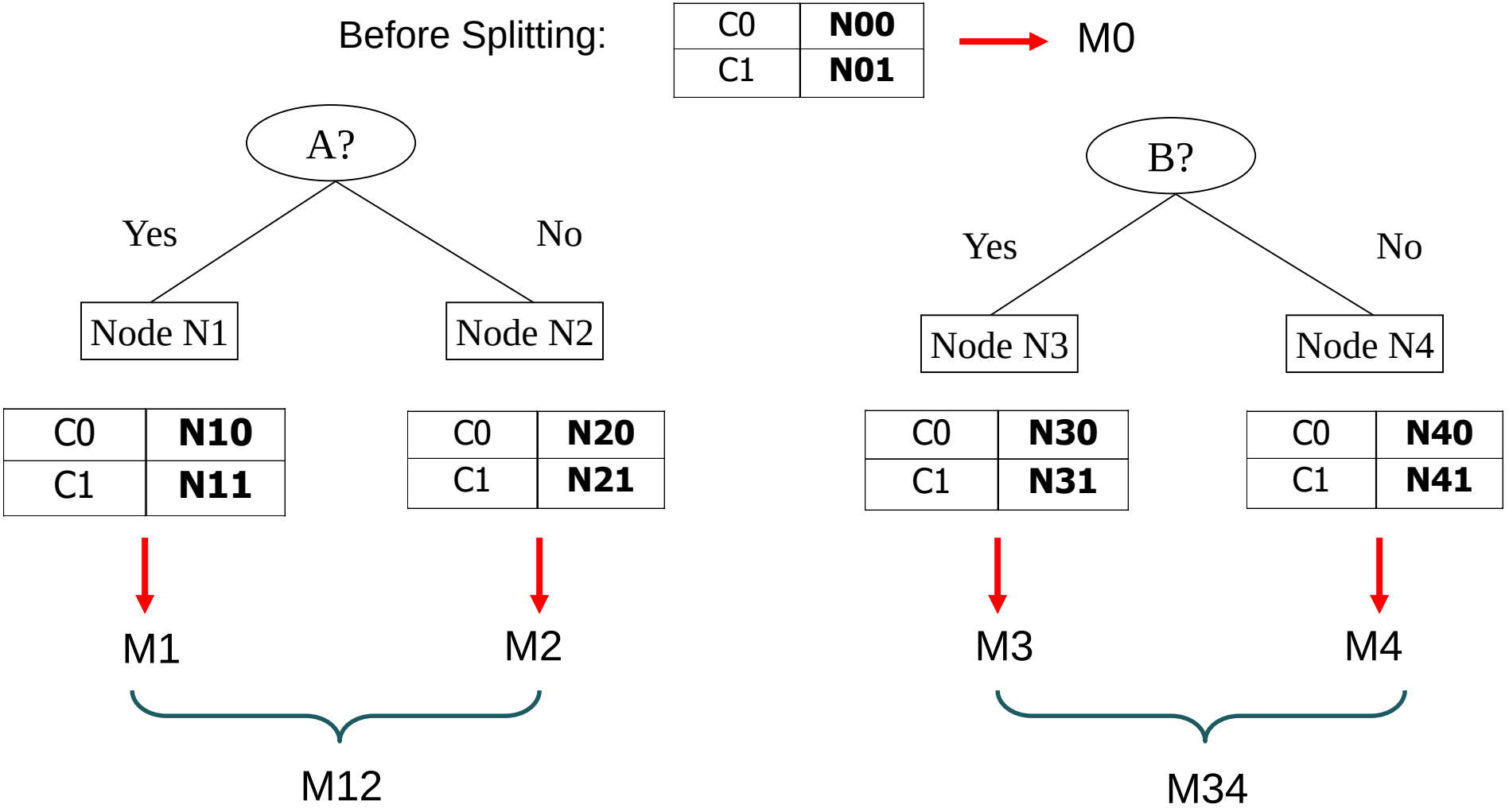

Which test condition is the best?

# How to determine the best split?

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

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

# How to determine the best split?




Gain = M0 – M12 vs M0 – M34

# ID3 – DT Learning Algorithm

- Iterative Dichotomizer (ID) 3
- Proposed by Ross Quinlan in 1979
- Based on Hunt's algorithm – uses a top-down greedy approach for DT learning
- Uses Shanon's Entropy for measuring node impurity
- Information Gain (IG) is used to select the best attribute for splitting

# Shannon's Entropy

- Entropy at a given node t:

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

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

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

# Shannon's Entropy

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

| C1 | **0** |
|---|---|
| C2 | **6** |

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

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

| C1 | **1** |
|---|---|
| C2 | **5** |

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

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

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

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

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

# Information Gain (IG)

- 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)

# ID3 Algorithm

1) Establish the Target Classification Attribute (in Table R)

2) Compute Classification Entropy

3) For each attribute in R, calculate IG using the classification attribute.

4) Select attribute with the ***highest IG to be the next Node*** in the tree (starting from the Root node)

5) Remove the Node Attribute, creating reduced table $R_S$

6) Repeat steps 3-5 ***until all attributes have been used***, or the ***same classification value remains for all rows*** in the reduced table.

# A 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.

# A Sample Problem - Classification Entropy

- Calculating Classification Entropy of the entire dataset

- $I_E = -(6/15)\log_2(6/15) - (9/15)\log_2(9/15) = \sim 0.971$

- Must calculate Information Gain (IG) of remaining attributes to determine the root node.

# A Sample Problem – 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) |

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

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

# A Sample Problem – 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**

# A Sample Problem – 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 | $I_{heavy} = -(4/5)\log_2(4/5)-(1/5)\log_2(1/5) = \sim 0.72$ |

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

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

# A Sample Problem – 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.**

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

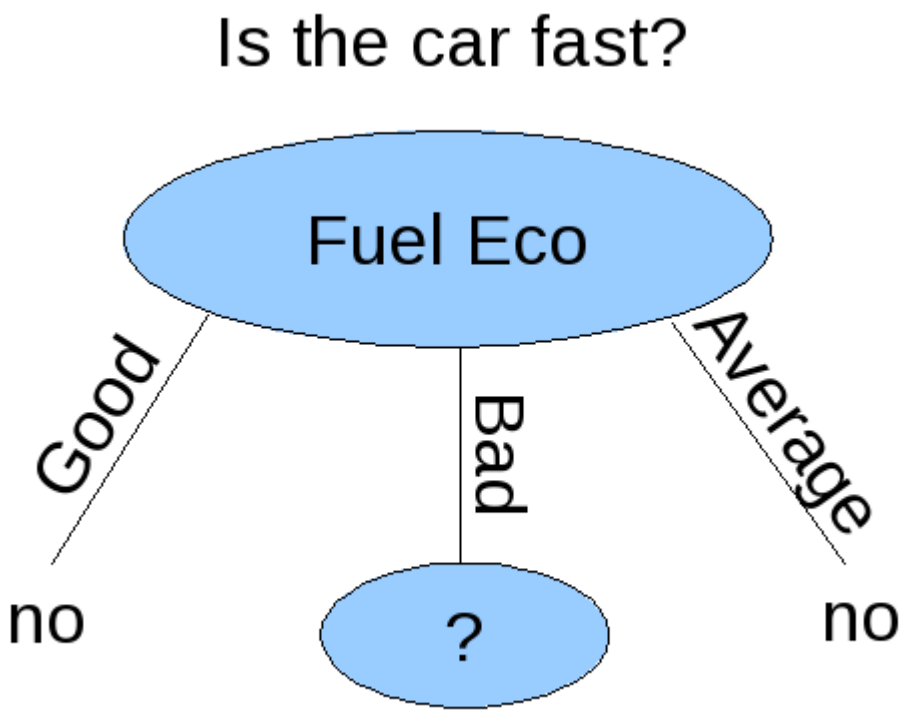
$IG_{FuelEco}$ = **0.971 – 0.667 = 0.304**

# A Sample Problem – 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.

# A Sample Problem – Root of Decision Tree



# A Sample Problem – 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.

# A Sample Problem – 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) |

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

$IG_{Engine}$ = **1 – 0.885 = 0.115**

# A Sample Problem – 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 | $I_{noturbo} = -(3/7)\log_2(3/7)-(4/7)\log_2(4/7) = \sim 0.84$ |

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

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

# A Sample Problem – 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) |

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

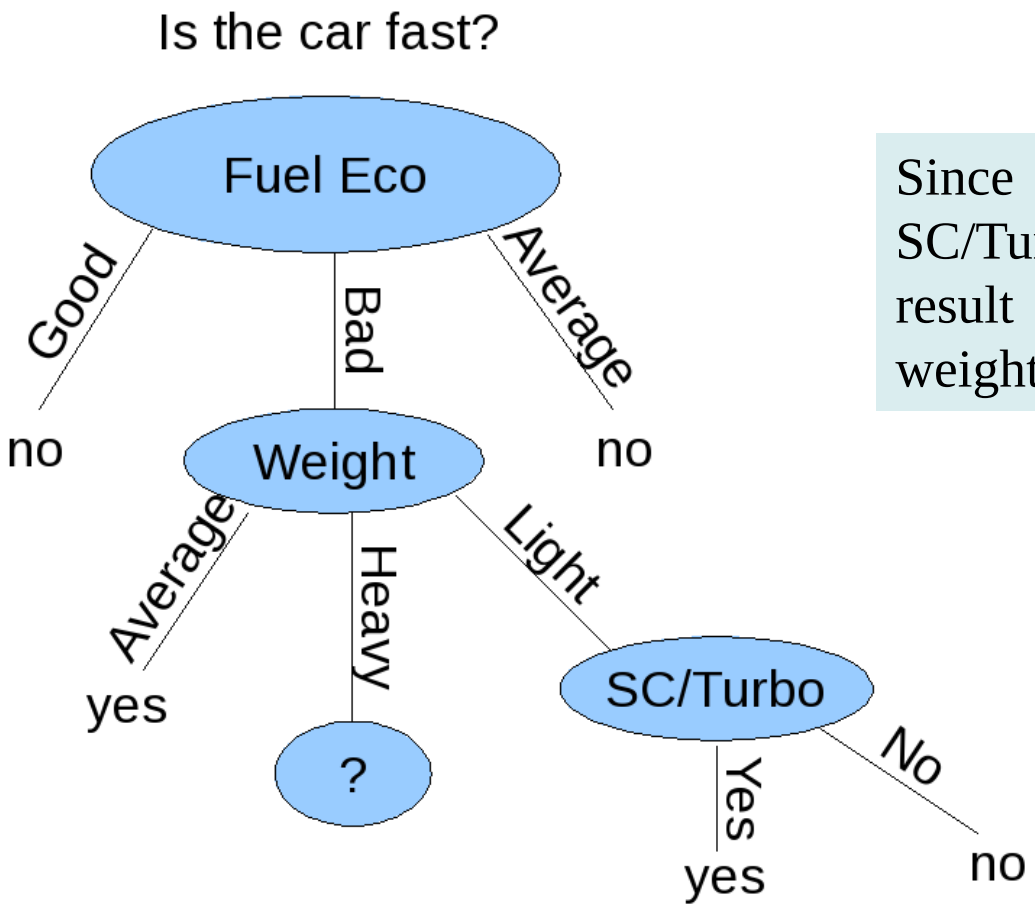
$$IG_{Engine} = \mathbf{1 - 0.561 = 0.439}$$

# A Sample Problem - 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.

# A Sample Problem – Decision Tree




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

# A Sample Problem – 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.

# ID3 - Shortcomings

- ID3 attempts to learn the shortest decision tree from the learning data which may not be always the best!
- Requires learning to have completely consistent patterns with no uncertainty or missing values allowed
- IG for unique keys such as “Employee ID” will be very high
  - Need to fix IG to avoid such trivial splits

# C4.5 History

- ID3, CHAID – 1960s
- C4.5 innovations (Quinlan):
  - Fix the Information Gain function to prevent trivial splits
  - Permit numeric attributes
  - Deal sensibly with missing values
  - Pruning to deal with for noisy data
- C4.5 - one of best-known and most widely-used learning algorithms
  - Last research version: C4.8, implemented in Weka as J4.8 (Java)
  - Commercial successor: C5.0 (available from Rulequest)

# Splitting based on GainRATIO

Gain Ratio:

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

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

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!
- 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 |

  - E.g. 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!

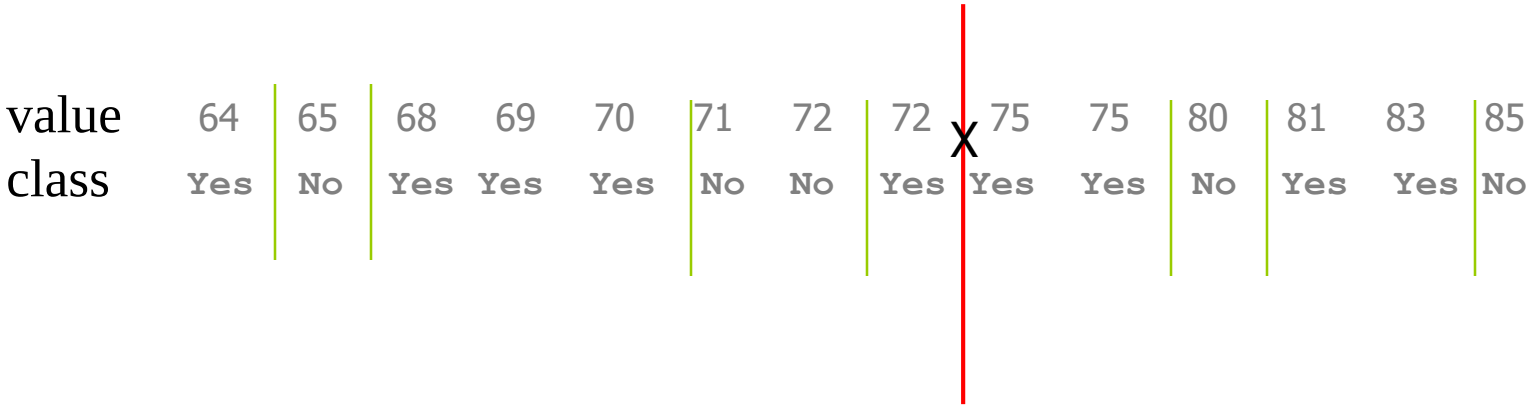
witten & eibe

# 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)




Potential optimal breakpoints

Breakpoints between values of the same class cannot be optimal

# Binary *vs.* Multi-way splits

- Splitting (multi-way) on a nominal attribute exhausts all information in that attribute
  - Nominal attribute is tested (at most) once on any path in the tree
- Not so for binary splits on numeric attributes!
  - Numeric attribute may be tested several times along a path in the tree
- Disadvantage: tree is hard to read
- Remedy:
  - pre-discretize numeric attributes, *or*
  - use multi-way splits instead of binary ones

# Handling Missing Values

- Missing value denoted "?" in C4.X
- Simple Idea: Treat missing as a separate value
- Q: When this is not appropriate?
- A: When values are missing due to different reasons
  - Example 1: Gene expression could be missing when it is very high or very low
  - Example 2: Field **IsPregnant**=missing for a male patient should be treated differently (no) than for a female patient of age 25 (unknown)

# Missing Values - Advanced

- Split instances with missing values into pieces
  - A piece going down a branch receives a weight proportional to the popularity of the branch
  - Weights sum to 1
- Info gain works with fractional instances
  - Use sums of weights instead of counts
- During classification, split the instance into pieces in the same way
  - Merge probability distribution using weights

# Missing Values - Example

| Attribute1 | Attribute2 | Attribute3 | Class |
|---|---|---|---|
| A | 70 | True | CLASS1 |
| A | 90 | True | CLASS2 |
| A | 85 | False | CLASS2 |
| A | 95 | False | CLASS2 |
| A | 70 | False | CLASS1 |
| ? | 90 | True | CLASS1 |
| B | 78 | False | CLASS1 |
| B | 65 | True | CLASS1 |
| B | 75 | False | CLASS1 |
| C | 80 | True | CLASS2 |
| C | 70 | True | CLASS2 |
| C | 80 | False | CLASS1 |
| C | 80 | False | CLASS1 |
| C | 96 | False | CLASS1 |

# Missing Values - Example

T1: (attribute1 = A)

| Att. 2 | Att.3 | Class | w |
|---|---|---|---|
| 70 | True | C1 | 1 |
| 90 | True | C2 | 1 |
| 85 | False | C2 | 1 |
| 95 | False | C2 | 1 |
| 70 | False | C1 | 1 |
| *90* | *True* | *C1* | *5/13* |

T1: (attribute1 = B)

| Att.2 | Att.3 | Class | w |
|---|---|---|---|
| *90* | *True* | *C1* | *3/13* |
| 78 | False | C1 | 1 |
| 65 | True | C1 | 1 |
| 75 | False | C1 | 1 |

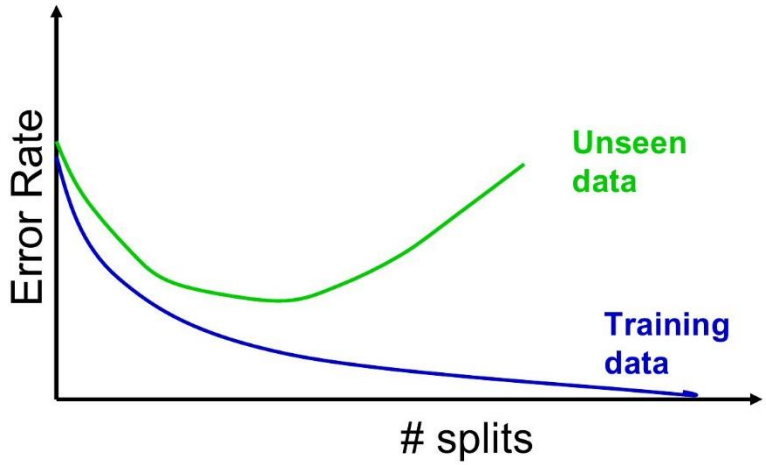
T1: (attribute1 = C)

| Att. 2 | Att.3 | Class | w |
|---|---|---|---|
| 80 | True | C2 | 1 |
| 70 | True | C2 | 1 |
| 80 | False | C1 | 1 |
| 80 | False | C1 | 1 |
| 96 | False | C1 | 1 |
| *90* | *True* | *C1* | *5/13* |

# Missing Values - Example

```
If  Attribute1 = A  Then
      If  Attribute2 <= 70 Then
            Classification = CLASS1  (2.0 / 0);
      else
            Classification = CLASS2  (3.4 / 0.4);
elseif  Attribute1 = B  Then
      Classification = CLASS1  (3.2 / 0);
elseif  Attribute1 = C  Then
      If Attribute3 = true Then
            Classification = CLASS2  (2.4 / 0);
      else
            Classification = CLASS1  (3.0 / 0).
```

# Pruning




- **Goal:** Prevent overfitting to noise in the data
- Two strategies for "pruning" the decision tree:
  - *Post-pruning:* Take a fully-grown decision tree and discard unreliable parts
  - *Pre-pruning:* Stop growing a branch when information becomes unreliable
- Post-pruning preferred in practice because pre-pruning can "stop too early"

# Pre-pruning

- Based on statistical significance test
  - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
- Most popular test: *chi-squared test (CHAID)*
- ID3 uses chi-squared test in addition to information gain
  - Only statistically significant attributes were allowed to be selected by information gain procedure

# Early Stopping Problem

| | a | b | class |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 |
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 |

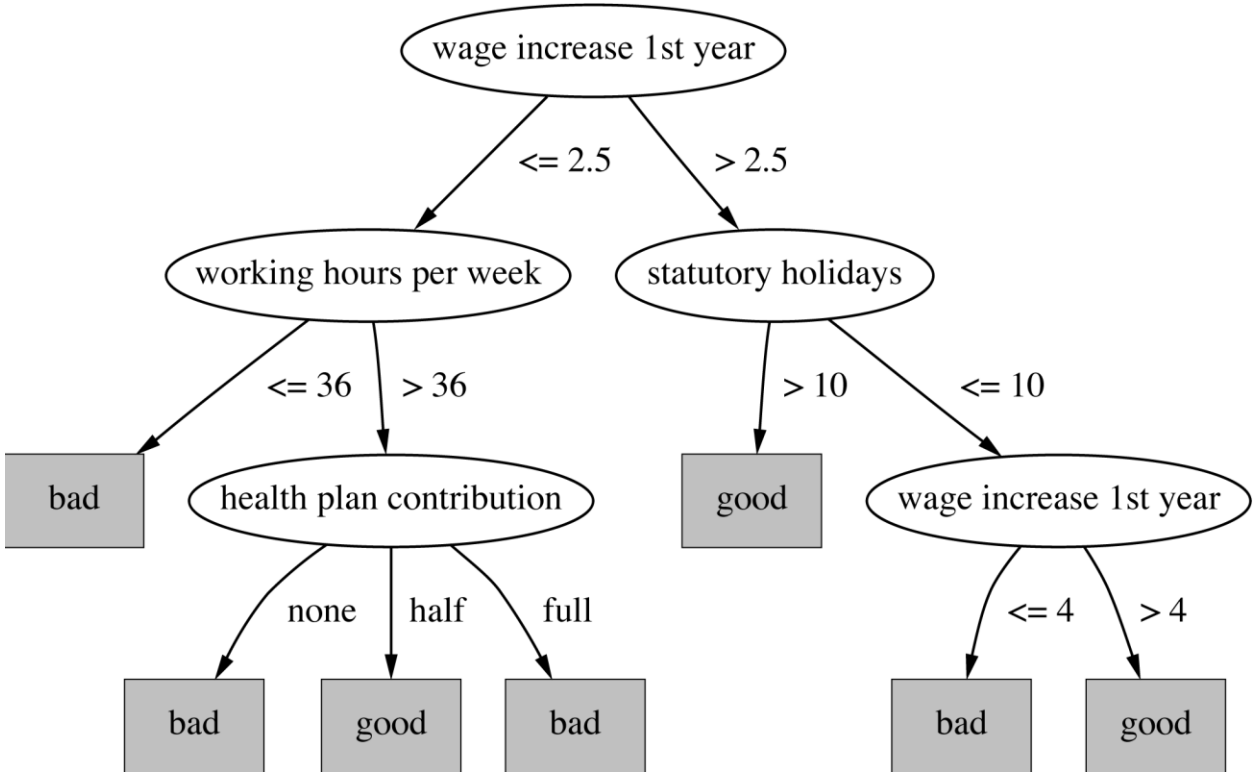
- Pre-pruning may stop the growth process prematurely: *early stopping*
- Classic example: XOR/Parity-problem
  - No *individual* attribute exhibits any significant association to the class
  - Structure is only visible in fully expanded tree
  - Pre-pruning won't expand the root node
- But: XOR-type problems rare in practice
- And: Pre-pruning faster than Post-pruning

# Post-pruning

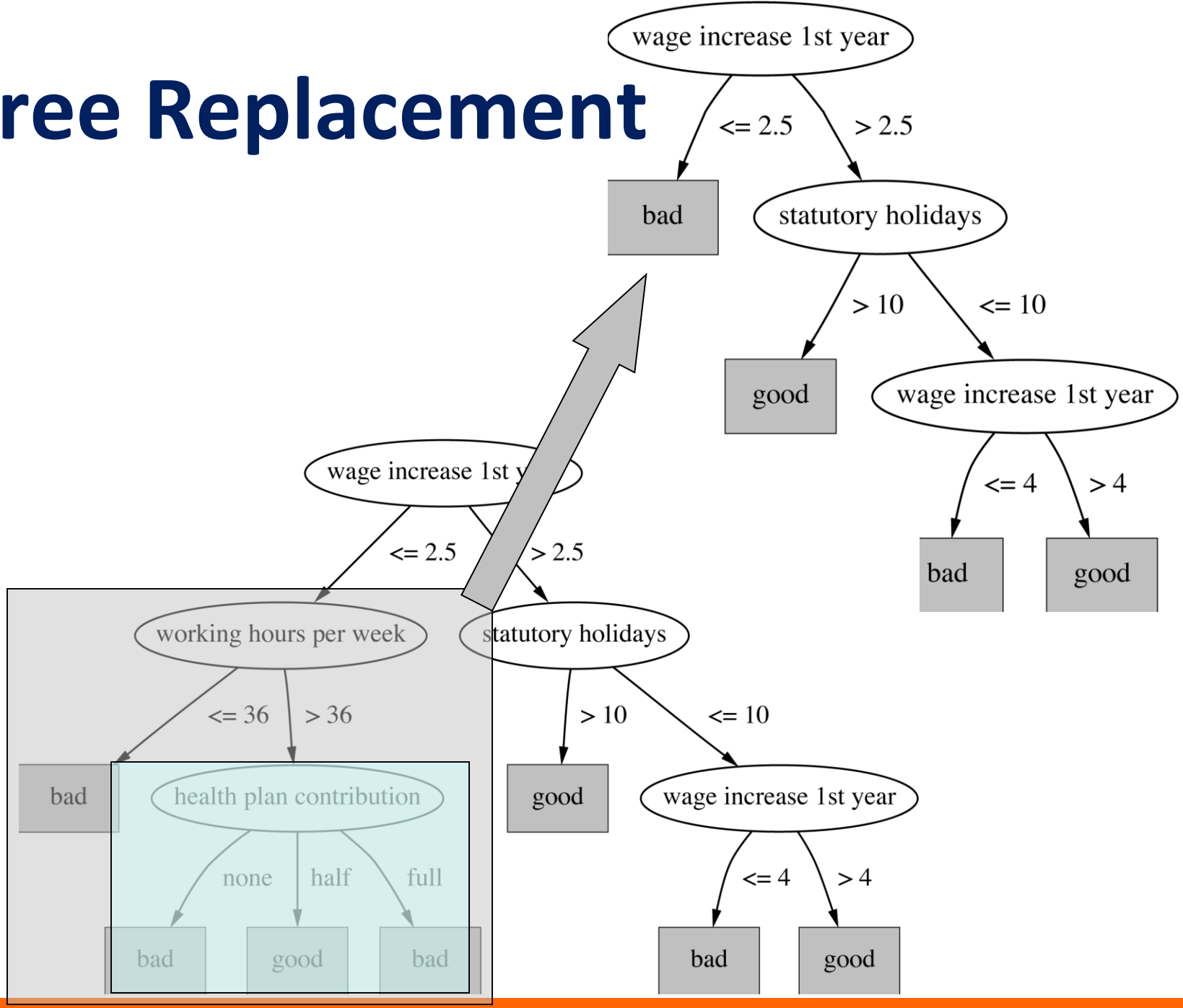
- First, build full tree
- Then, prune it
  - Fully-grown tree shows all attribute interactions
- Problem: Some subtrees might be due to chance effects
- Two pruning operations:
  1. *Subtree Replacement*
  2. *Subtree Raising*
- Evaluation Strategies:
  - Error Estimation
  - MDL Principle

# Subtree Replacement

- *Bottom-up Strategy*
- Consider replacing a tree only after considering all its subtrees
- Example: Labor negotiations

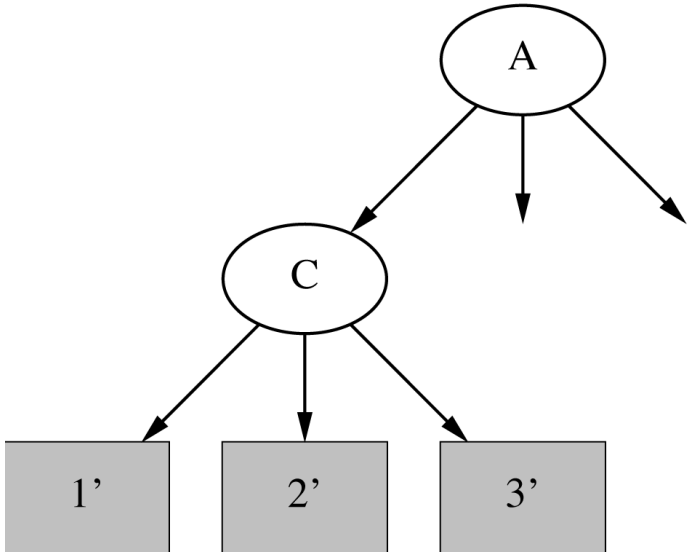


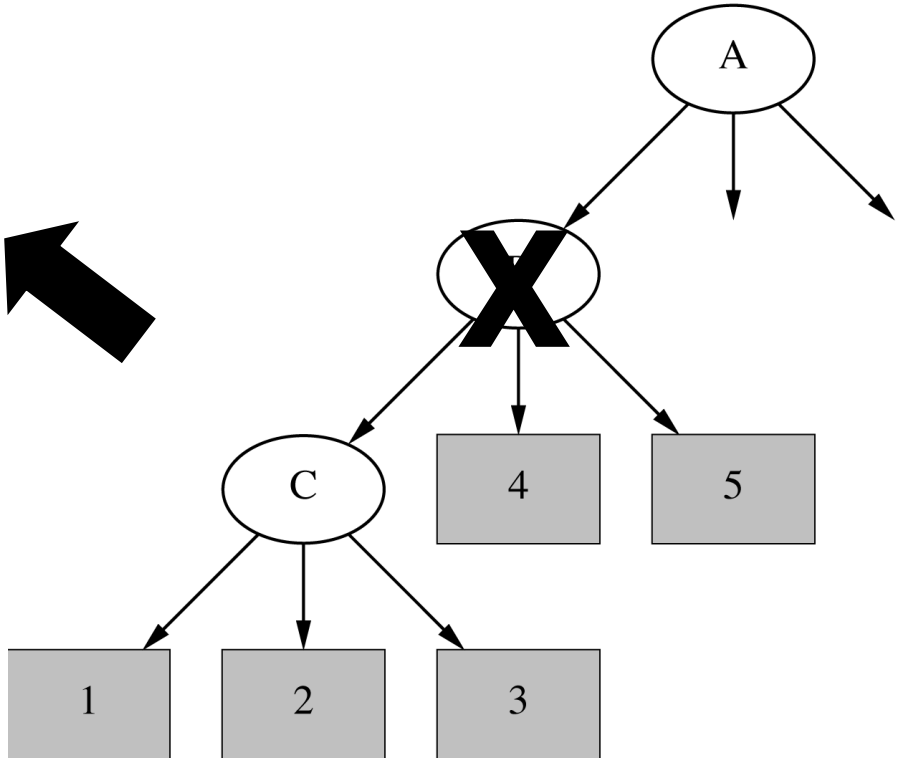
# Subtree Replacement



# *Subtree Raising

- Delete node
- Redistribute instances
- Slower than subtree replacement *(Worthwhile?)*



# Estimating Error Rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
  *Q: Why it would result in very little pruning?*
- Use hold-out set for pruning
  ("reduced-error pruning")
- C4.5's method
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Standard Bernoulli-process-based method
  - Shaky statistical assumptions (based on training data)

# MDL Principle

$$CC(T) = Err(T) + a\ L(T)$$

$CC(T)$ = Cost complexity of a tree

$Err(T)$ = Proportion of misclassified records

$\alpha$ = Penalty factor attached to tree size (set by user)

- Among trees of given size, choose the one with lowest CC
- Do this for each size of tree

# C4.5 – Generating Rules from Tree

- Simple way: one rule for each leaf
- C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
  - Can produce duplicate rules
  - Check for this at the end
- Then
  - Look at each class in turn
  - Consider the rules for that class
  - Find a "good" subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data

# What is CART?

- Classification And Regression Trees
- Developed by Breiman, Friedman, Olshen, Stone in early 80's.
  - Introduced tree-based modeling into the statistical mainstream
  - Rigorous approach involving cross-validation to select the optimal tree
- One of many tree-based modeling techniques.
  - CART -- the classic
  - CHAID
  - C5.0
  - Software package variants (SAS, S-Plus, R…)
  - Note: the "rpart" package in "R" is freely available

# Impurity Measure of CART: GINI

- Gini Index for a given node t :

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

(NOTE: *p( j | 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

- 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,

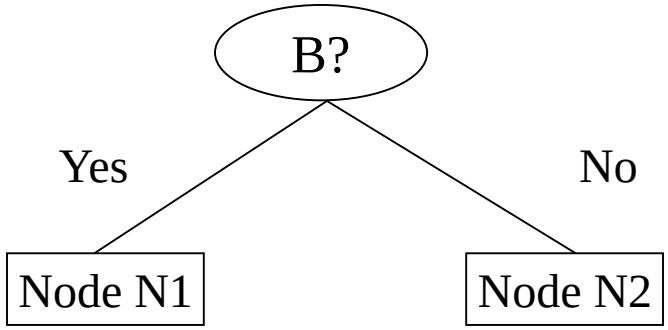
n = number of records at node p.

The "twoing" rule strikes a balance between *purity* and creating roughly *equal-sized nodes*

*Note: "twoing" is available in Salford Systems' CART but not in the "rpart" package in R.*

# 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

# Regression Trees

- Tree-based modeling for ***continuous* target variable**
    - Most intuitively appropriate method for loss ratio analysis
- Find split that produces greatest separation in

$$\sum[y - E(y)]^2$$

- Find nodes with minimal *within variance*
    - Therefore greatest *between variance*
- Every record in a node is assigned the same y
    - Model is a *step function*

# Regression Trees (Contd..)

$$F(x) = \sum_{i=1}^{M} c_m I(x \in R_m)$$

$\{R_m\}_1^M$ are subregions of the input variable space, and $x$ is a vector of input variables.

- *Examples:* $\{x_9 < 15.2\}$, $\{9 <= x_{300} < 786 \ \& \ color = red\}$

$c_m$ are the estimated values of the outcome (y) in region $R_m$

CART tries to minimize

- $e(T) = \sum_{i=1}^{N} \left[ y_i - \sum_{m=1}^{M} c_m I(x \in R_m) \right]^2$
- *with respect to* $c_m$ *and* $R_m$

# Regression Trees (Contd..)

Can calculate contribution of split to decreasing objective $e(T)$ by

$$e_m = \frac{1}{N} \sum_{x_i \in R_m} (y_i - \bar{y}_m)^2$$

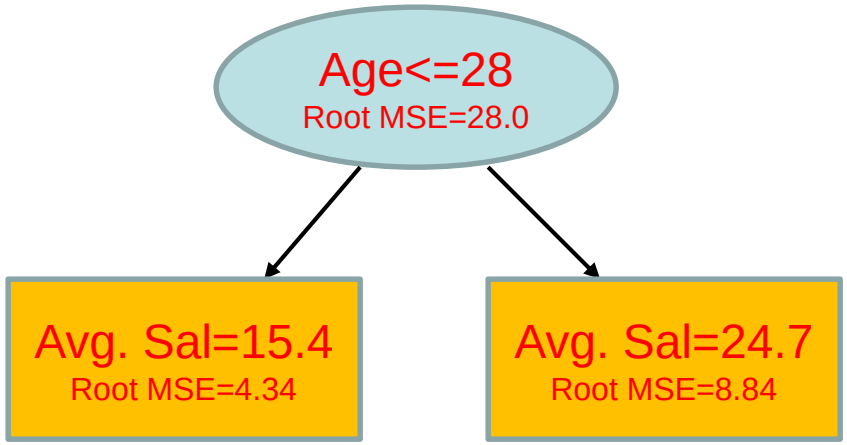
$$Imp_m = e_m - e_{ml} - e_{mr}$$

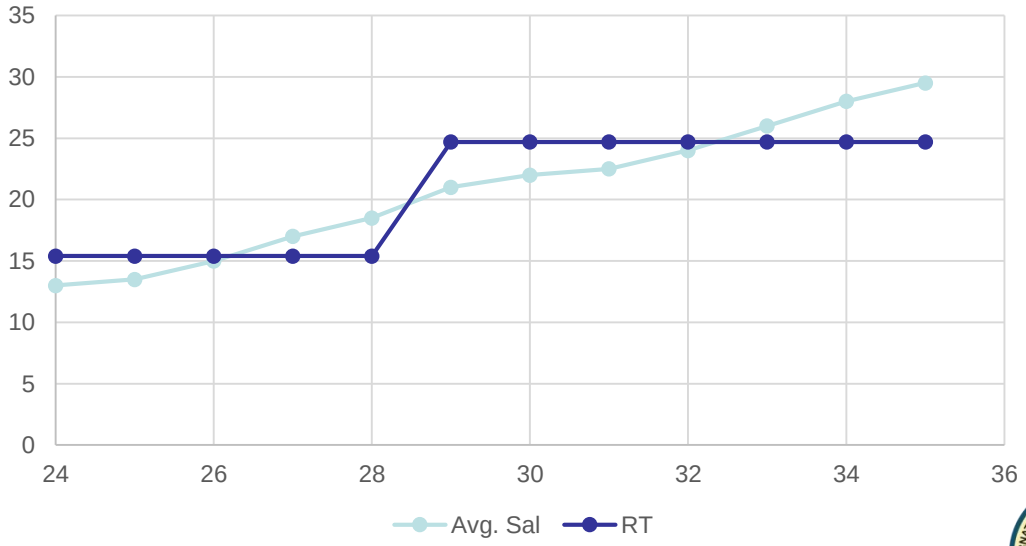
If $Imp_m \geq cp$ then accept the split, otherwise make $m$ a terminal node

# A Toy Example

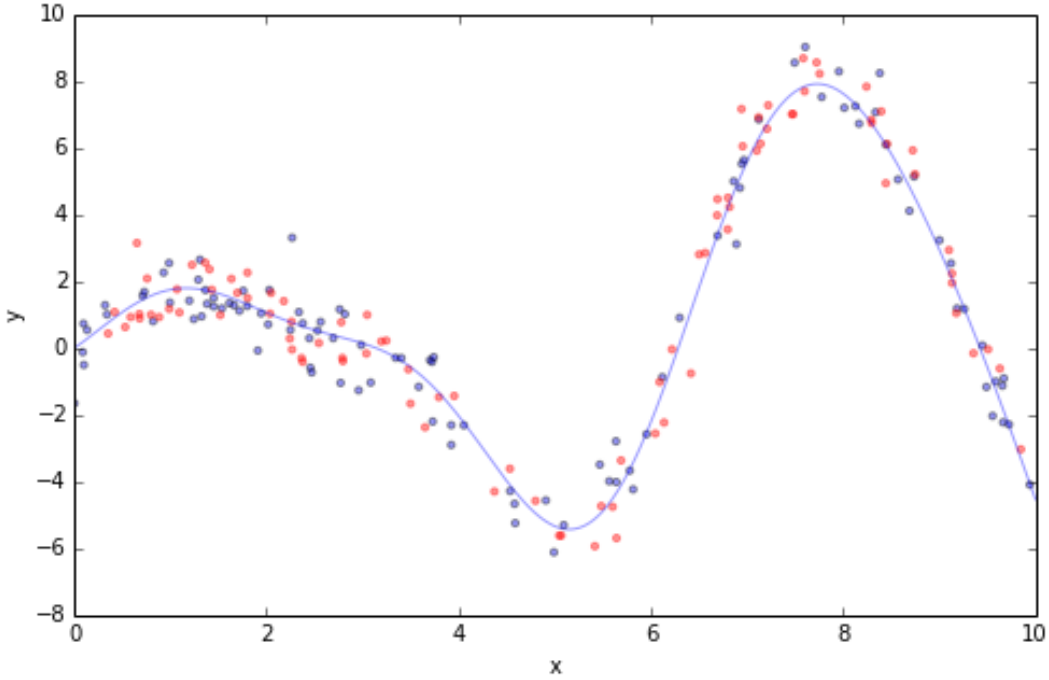
| Age | Avg. Sal (in Ks) |
|---|---|
| 24 | 13 |
| 25 | 13.5 |
| 26 | 15 |
| 27 | 17 |
| 28 | 18.5 |
| 29 | 21 |
| 30 | 22 |
| 31 | 22.5 |
| 32 | 24 |
| 33 | 26 |
| 34 | 28 |
| 35 | 29.5 |

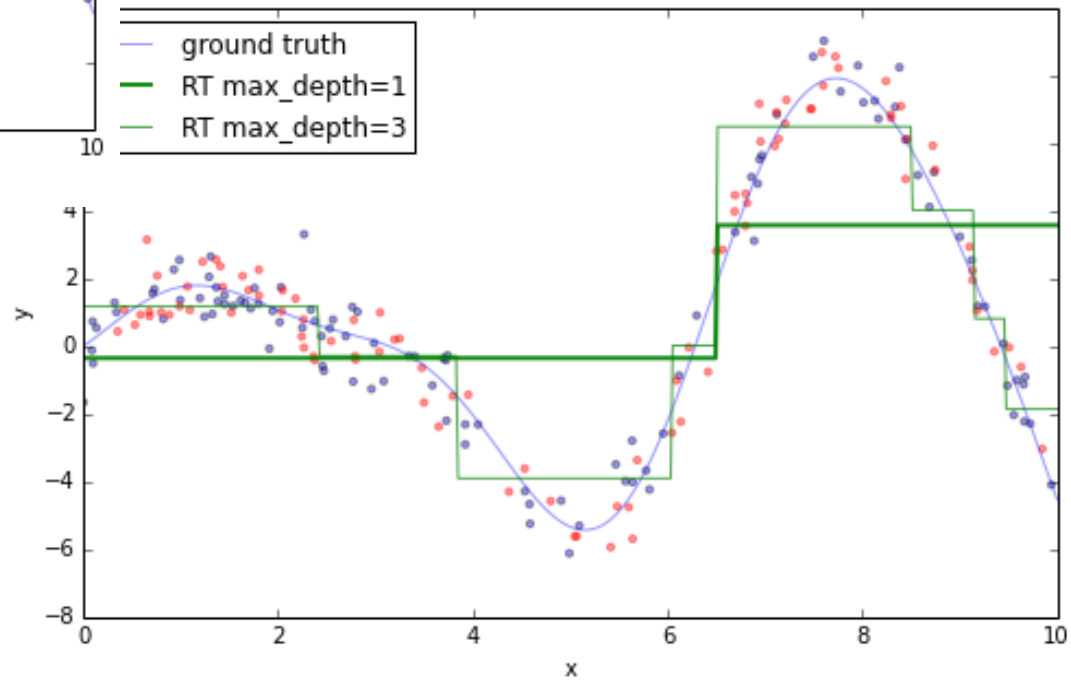
| Split Cond | Root Node MSE | L MSE | R MSE | Gain |
|---|---|---|---|---|
| Age<=24 | 28.05556 | 0 | 24.52066 | 3.534894 |
| Age<=25 | 28.05556 | 0.0625 | 19.8525 | 8.140556 |
| Age<=26 | 28.05556 | 0.722222 | 15.38889 | 11.94444 |
| Age<=27 | 28.05556 | 2.421875 | 11.96484 | 13.66884 |
| **Age<=28** | **28.05556** | **4.34** | **8.846939** | **14.86862** |
| Age<=29 | 28.05556 | 7.972222 | 7.638889 | 12.44444 |
| Age<=30 | 28.05556 | 10.76531 | 6.5 | 10.79025 |
| Age<=31 | 28.05556 | 12.55859 | 4.296875 | 11.20009 |
| Age<=32 | 28.05556 | 14.94444 | 2.055556 | 11.05556 |
| Age<=33 | 28.05556 | 18.5125 | 0.5625 | 8.980556 |
| Age<=34 | 28.05556 | 23.15702 | 0 | 4.898531 |



# Regression Trees (Contd..)



# Classification Trees vs. Regression Trees

- Splitting Criteria:
  - Gini, Entropy, Twoing
- Goodness of fit measure:
  - misclassification rates
- Prior probabilities and misclassification costs
  - available as model "tuning parameters"

- Splitting Criterion:
  - sum of squared errors
- Goodness of fit:
  - same measure!
  - sum of squared errors
- No priors or misclassification costs…
  - … just let it run

# CART Advantages

- Nonparametric (no probabilistic assumptions)
- Automatically performs variable selection
- Uses any combination of continuous/discrete variables
  - Very nice feature: ability to automatically bin massively categorical variables into a few categories.
    - zip code, business class, make/model…
- Discovers "interactions" among variables
  - Good for "rules" search
  - Hybrid GLM-CART models

# CART Advantages

- CART handles missing values automatically
    - Using "surrogate splits"
- Invariant to monotonic transformations of predictive variable
- Not sensitive to outliers in *predictive* variables
  - Unlike regression
- Great way to explore, visualize data

# CART Disadvantages

- The model is a *step function*, not a continuous score
  - So if a tree has 10 nodes, y can only take on 10 possible values.
  - MARS improves this.
- Might take a large tree to get good lift
  - But then hard to interpret
  - Data gets chopped thinner at each split
- Instability of model structure
  - Correlated variables ➔ random data fluctuations could result in entirely different trees.
- CART does a poor job of modeling *linear structure*

# Summary

- Decision Trees are very intuitive and explainable models
- Works well in practice for many applications
- Issues while learning Decision Trees
  - Splits – binary, multi-way
  - Split criteria – entropy, info gain, gini, …
  - Missing value treatment
  - Pruning
  - Rule extraction from trees
- Both C4.5 and CART are robust tools

# References


- Mitchell, Tom M. *Machine Learning*. McGraw-Hill, 1997. pp. 55–58
- Jiawei Han And Micheline Kamber, *Data Mining Concept and Techniques*
- Andrew Moore, *Decision Trees Tutorial*, Auton Lab, http://www.autonlab.org/tutorials/dtree.html
- T. Hastie, R. Tibshirani and J. Friedman. *Elements of Statistical Learning*, Springer, 2009.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.
- J.R. Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

# International School of Engineering

Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals: +91-9502334561/63 or 040-65743991

For Corporates: +91-9618483483

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