# CS 584
# Data Mining

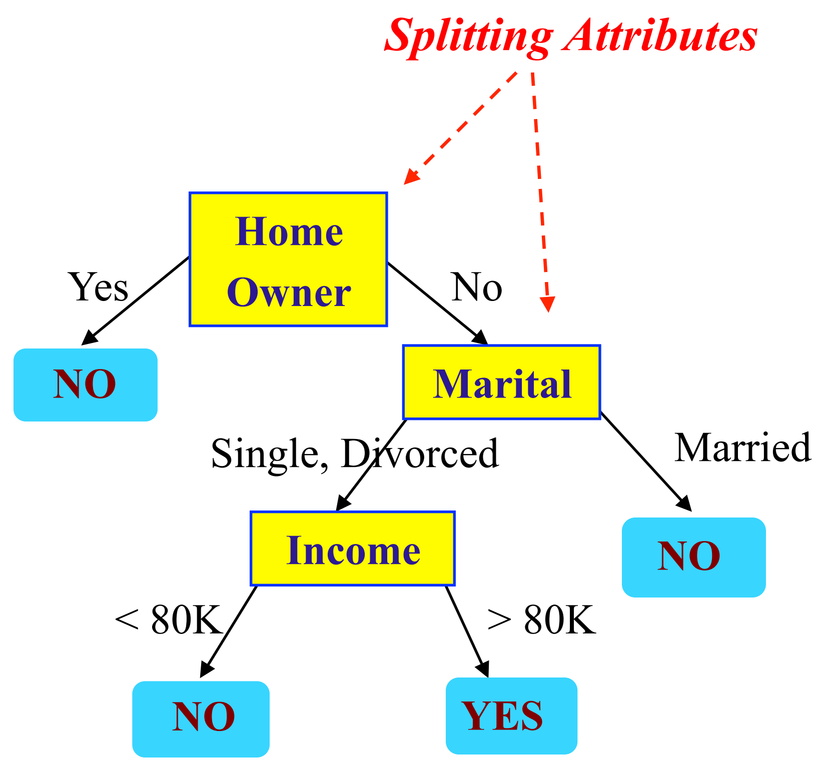
Classification 2

# Example of a Decision Tree

categorical categorical continuous class

| *Tid* | 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 |

**Training Data**

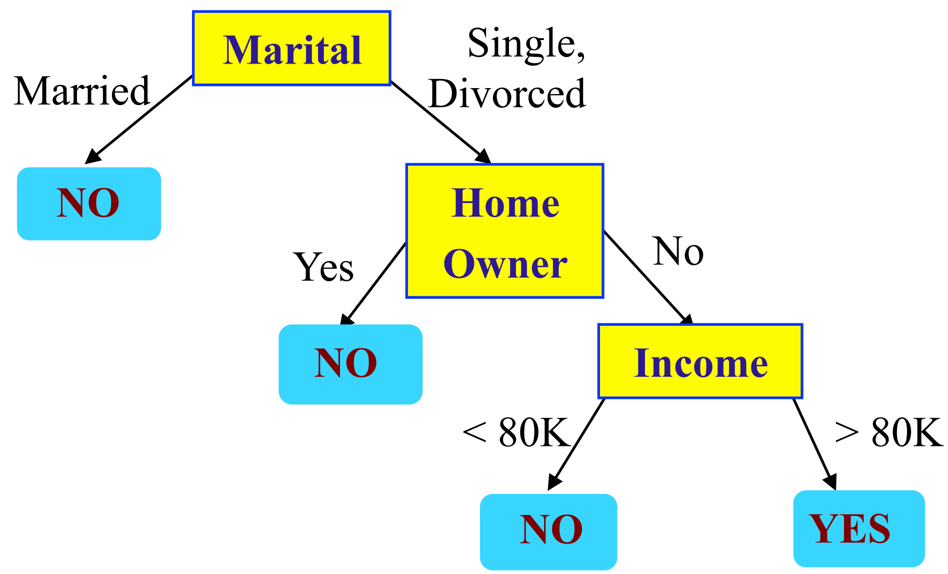



**Model: Decision Tree**

# Another Example of Decision Tree

categorical categorical continuous class

| *Tid* | 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 |




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

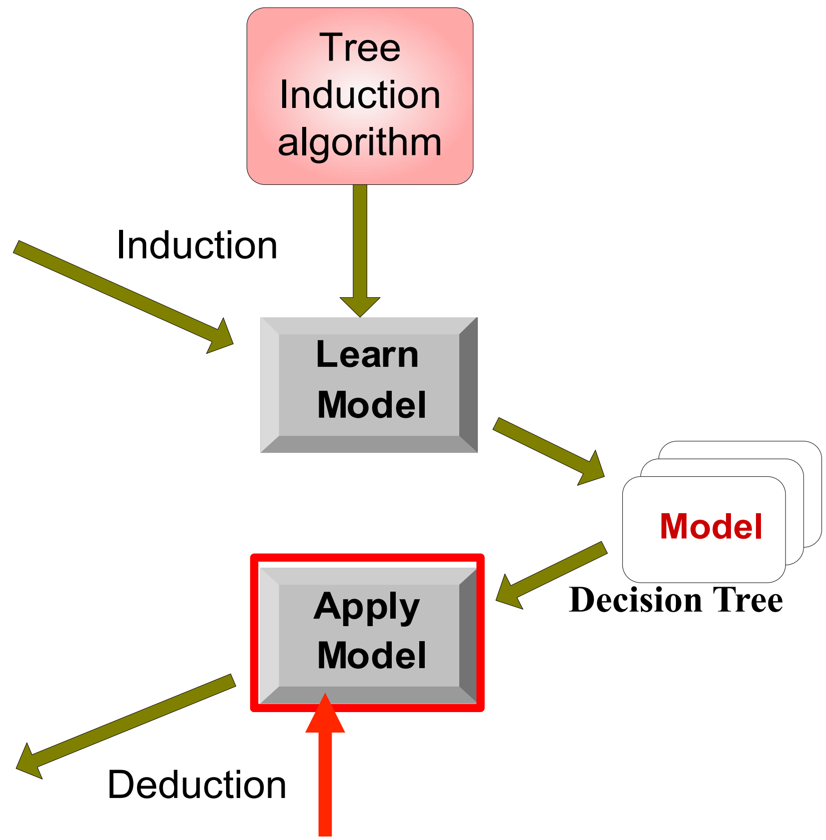
# Decision Tree Classification Task

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

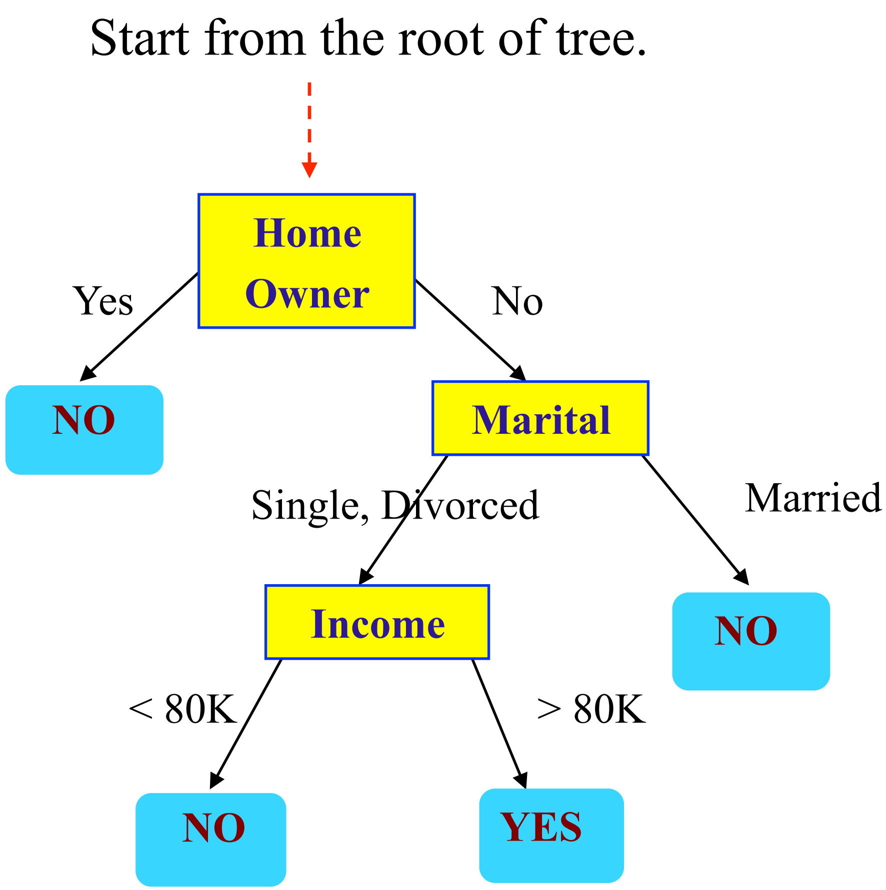
Training Set

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

Test Set



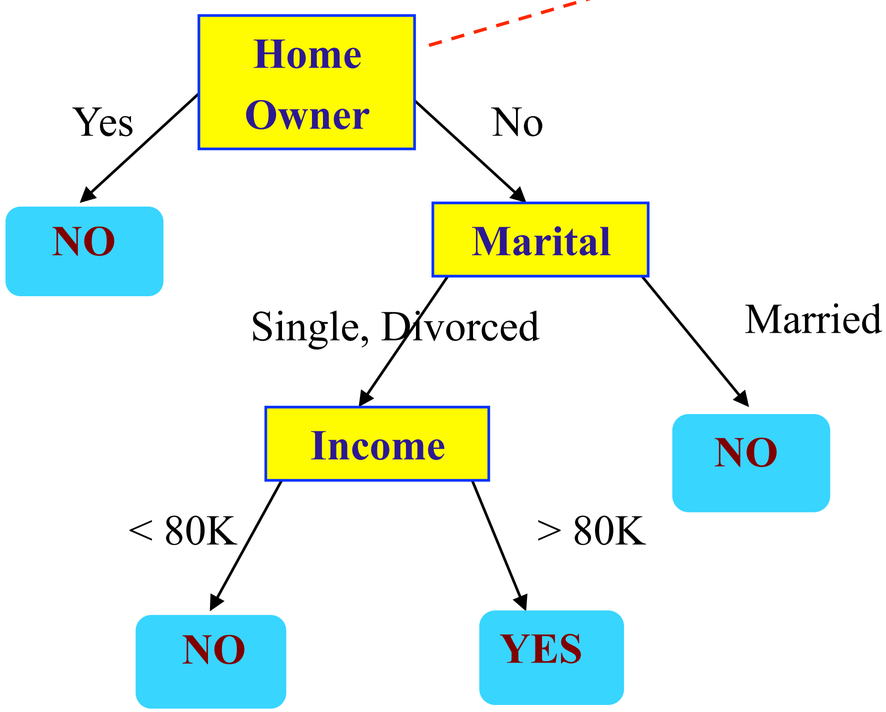
# Apply Model to Test Data




**Test Data**

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

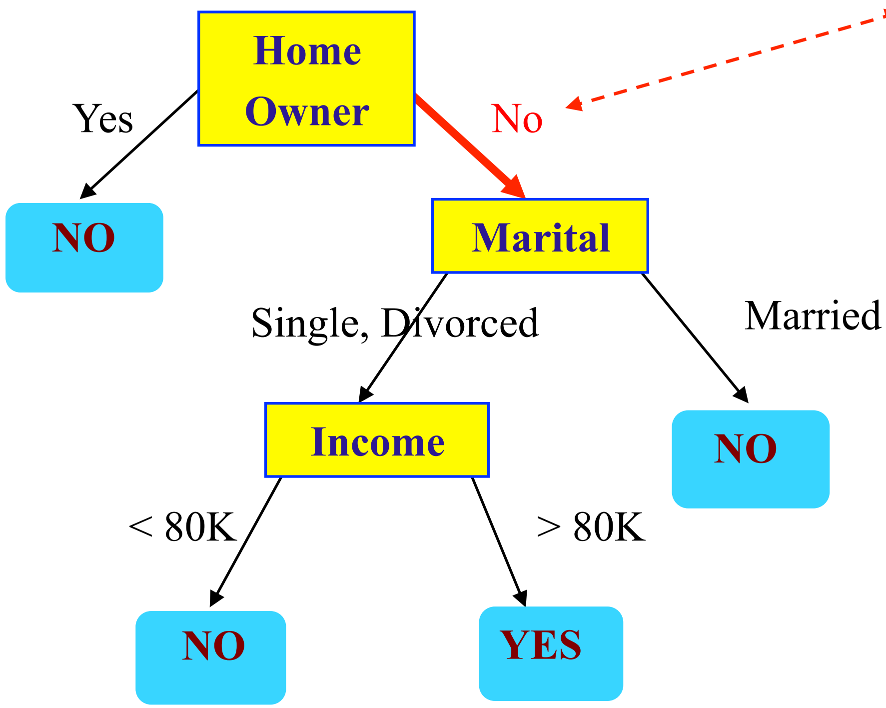
# Apply Model to Test Data




**Test Data**

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

# Apply Model to Test Data



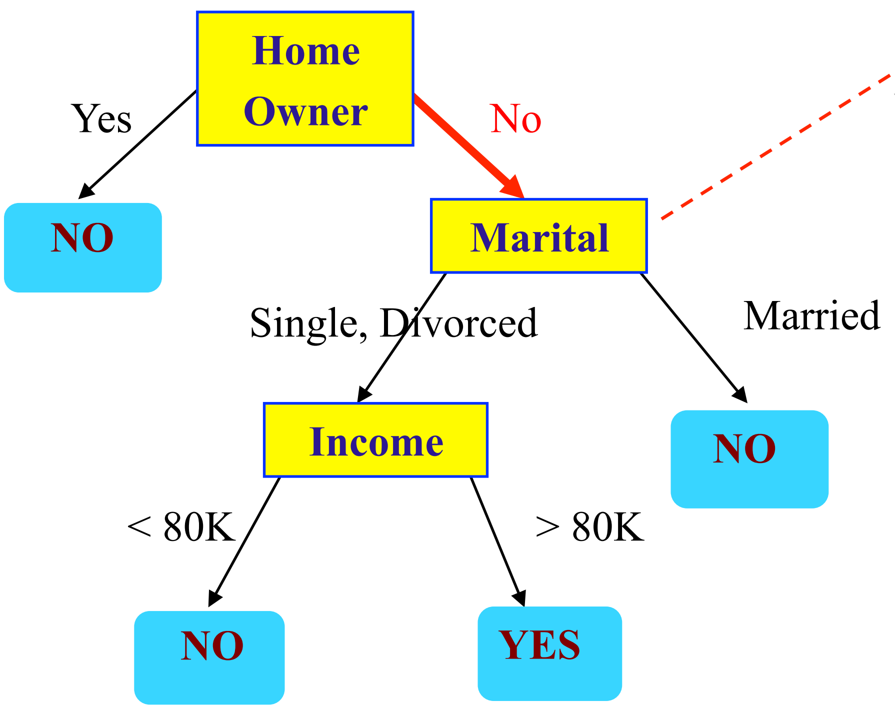

**Test Data**

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

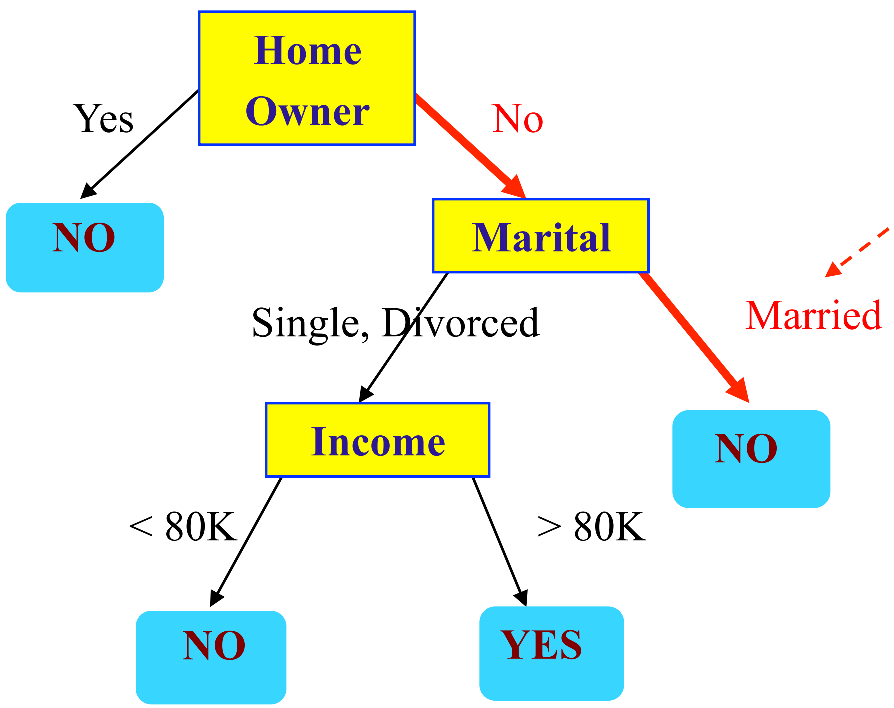
# Apply Model to Test Data

**Test Data**

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



# Apply Model to Test Data



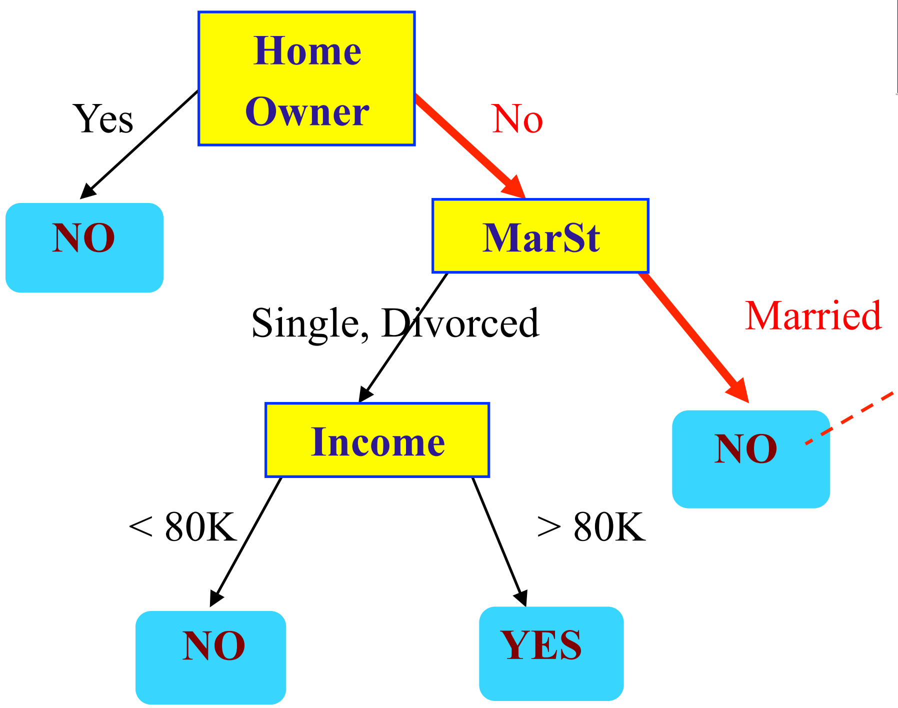

**Test Data**

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

# Apply Model to Test Data

**Test Data**

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

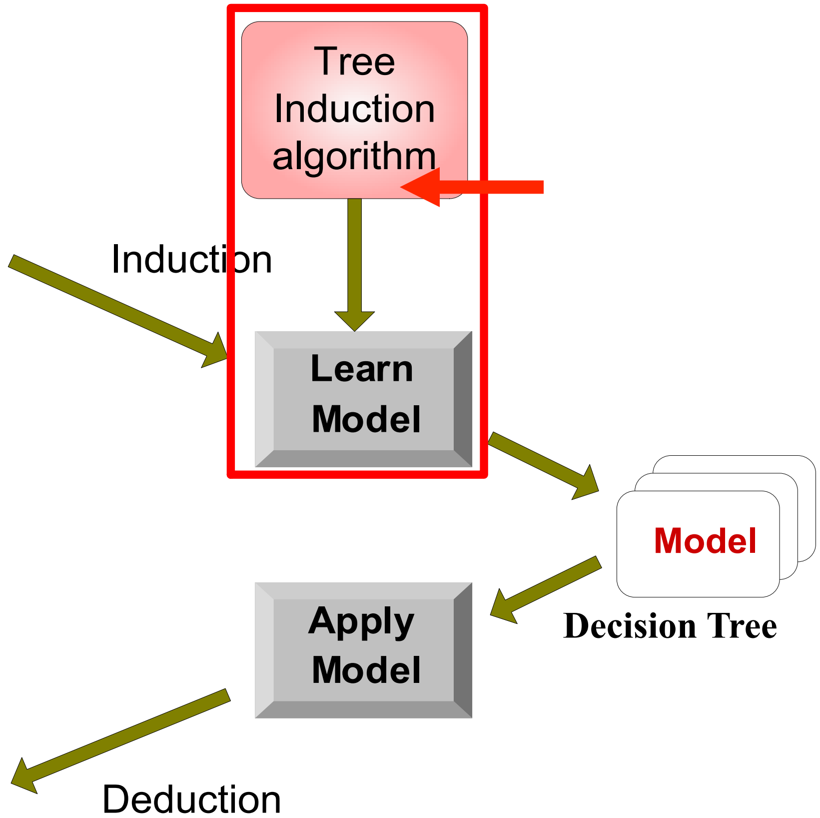



Assign Defaulted to “No”

# Decision Tree Classification Task

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

Training Set




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

Test Set

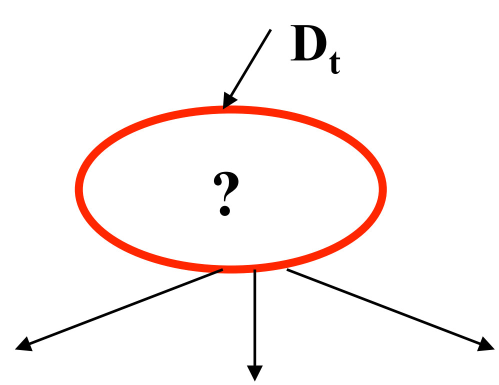
# Decision Tree Induction

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

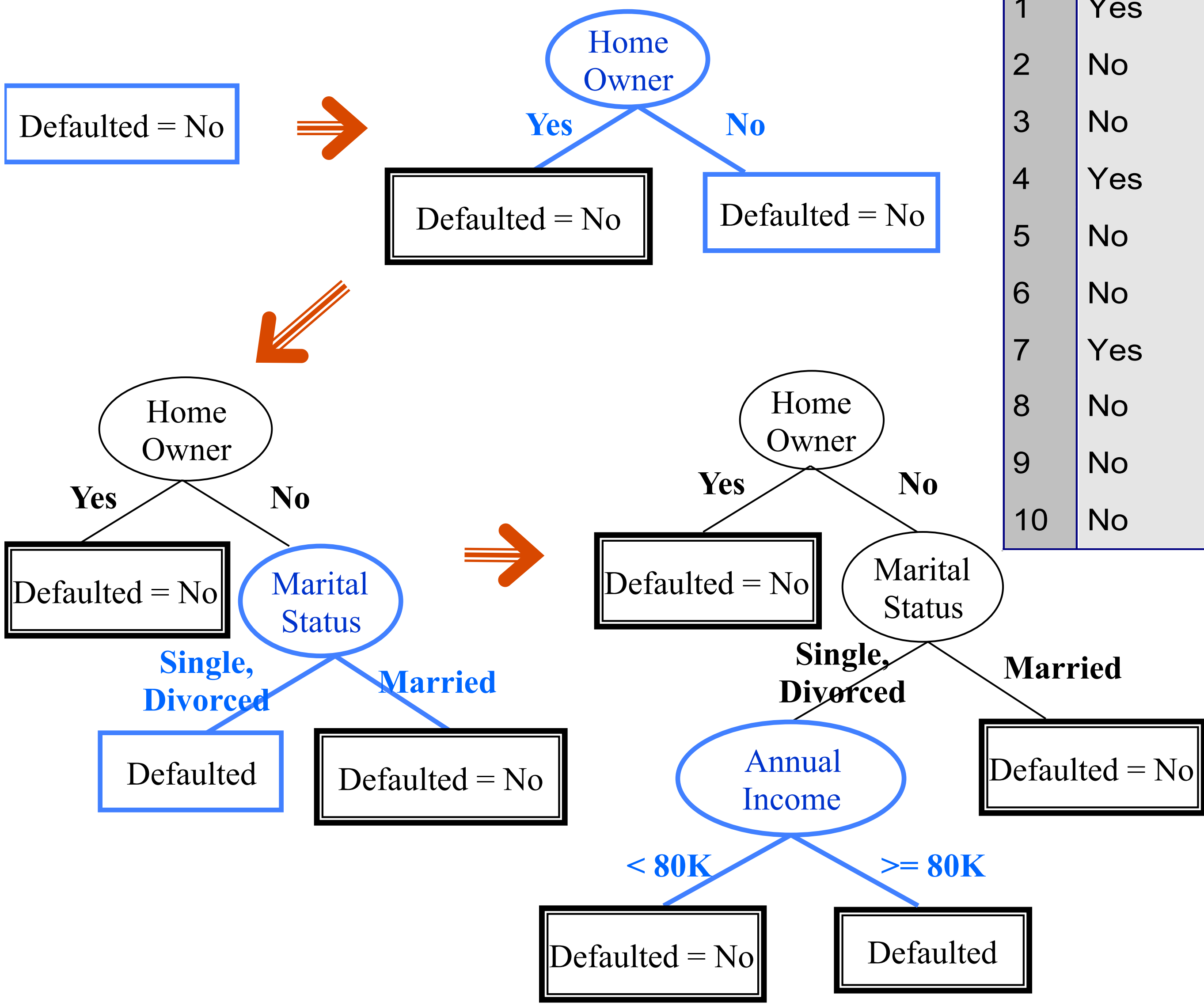
# General Structure of Hunt's Algorithm

- Let $D_t$ be the set of training records that reach a node t
- General Procedure:
  - If $D_t$ contains records that belong the same class $y_t$, then t is a leaf node labeled as $y_t$
  - If $D_t$ is an empty set, then t is a leaf node labeled by the default class, $y_d$
  - 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* | **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** |



# Hunt's Algorithm




| *Tid* | 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 |

# 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

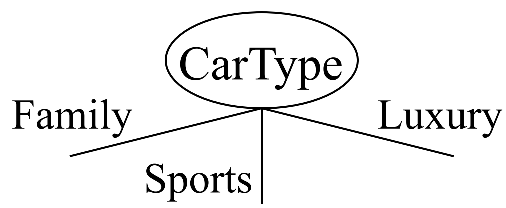
# How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

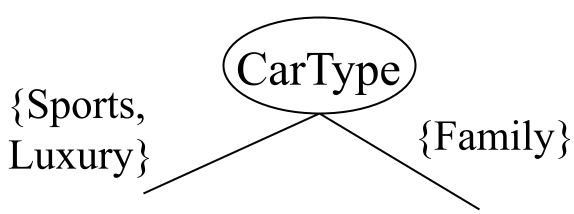
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

# Splitting Based on Nominal Attributes

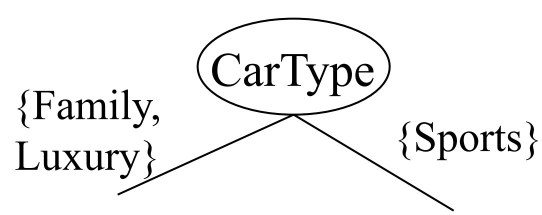
- Multi-way split: Use as many partitions as distinct values.




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

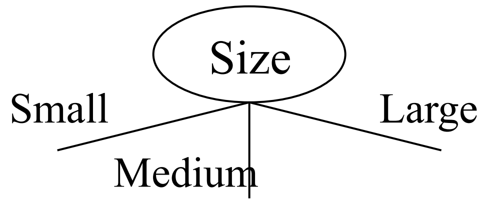



OR

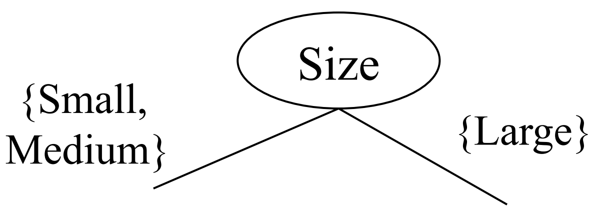


# Splitting Based on Ordinal Attributes

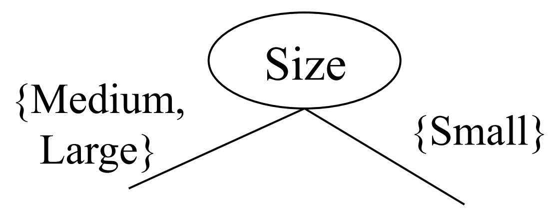
- Multi-way split: Use as many partitions as distinct values.



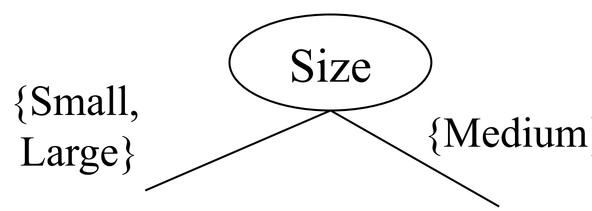

- Binary split: Divides values into two subsets. Need to find optimal partitioning.




OR



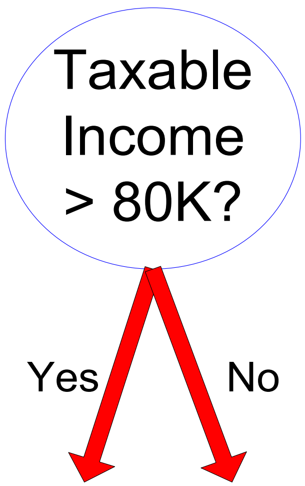

- What about this split?



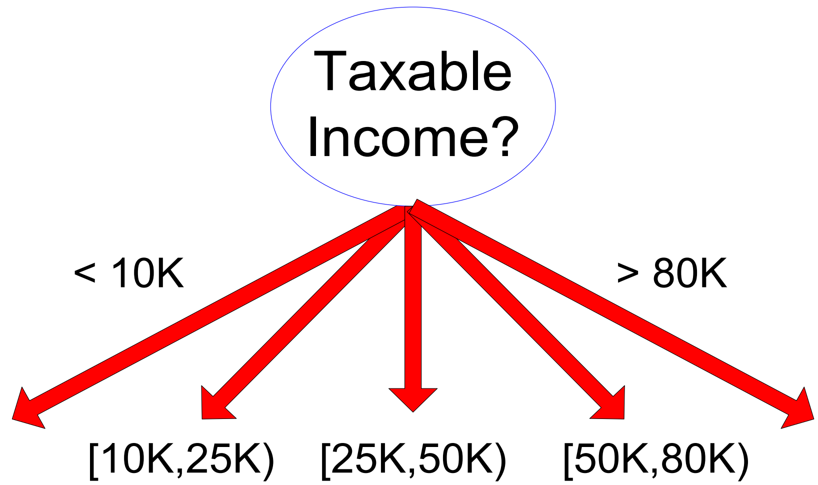
# Splitting Based on Continuous Attributes

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

# Splitting Based on Continuous Attributes




(i) Binary split




(ii) Multi-way split

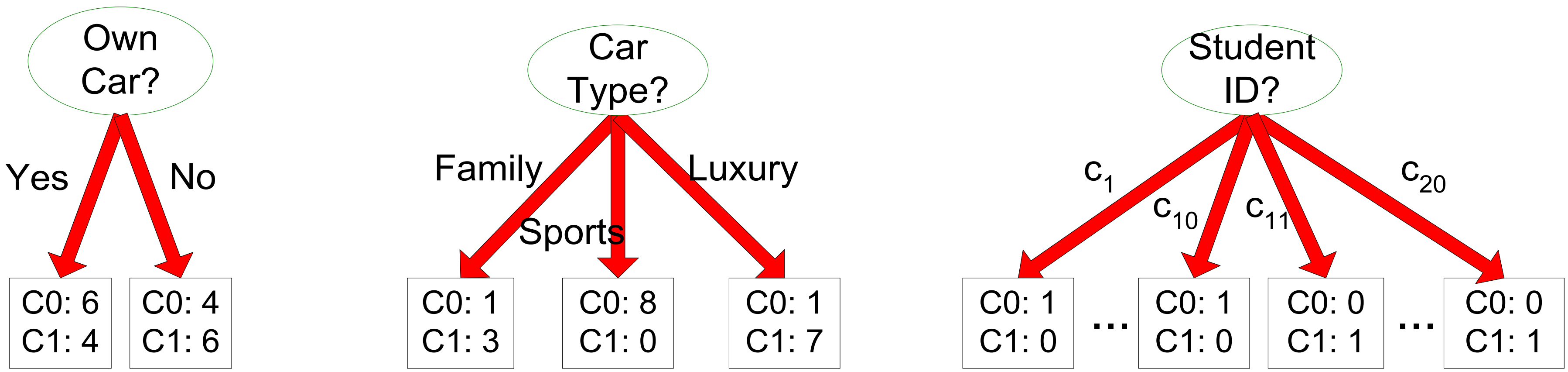
# Tree Induction

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

# Measures of Node Impurity

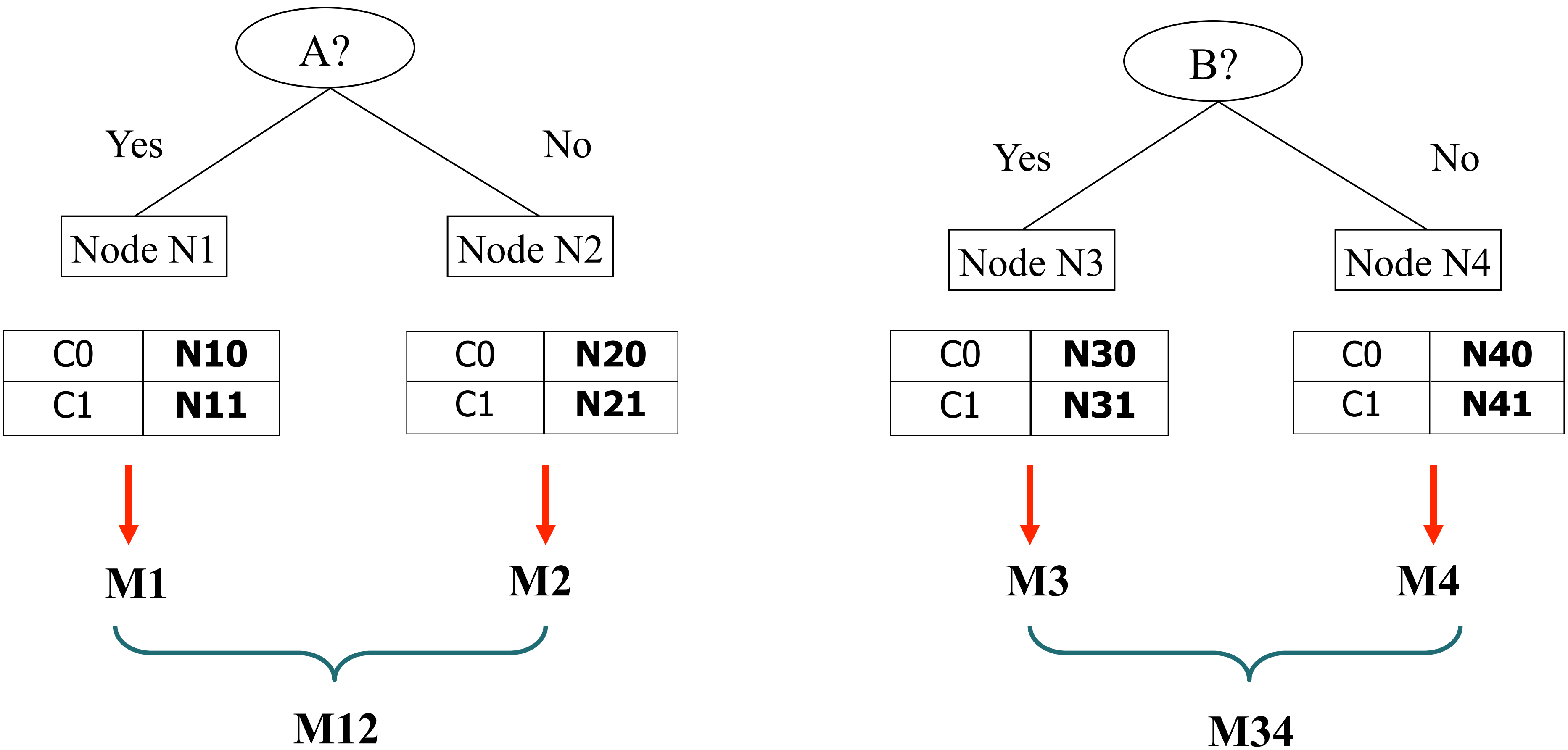
- Gini Index
- Entropy
- Misclassification error

# How to Find the Best Split

**Before Splitting:**

| | |
|---|---|
| C0 | **N00** |
| C1 | **N01** |

→ **M0**




**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 | 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,

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

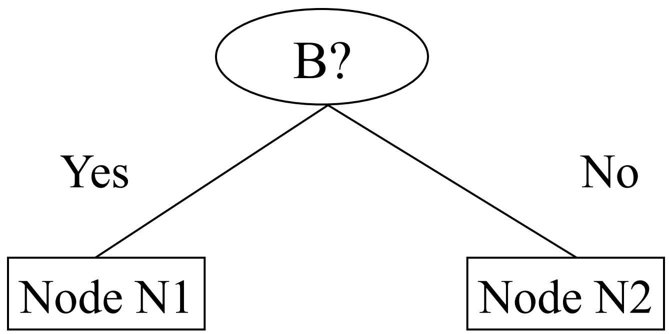
where, $n_i$ = number of records at child i,

$n$ = 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)**
**$= 1 - (5/7)^2 - (2/7)^2$**
**$= 0.408$**

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

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

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

# 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 | 2 | 1 |
| C2 | 4 | 1 | 1 |
| Gini | ? | | |

Two-way split
(find best partition of values)

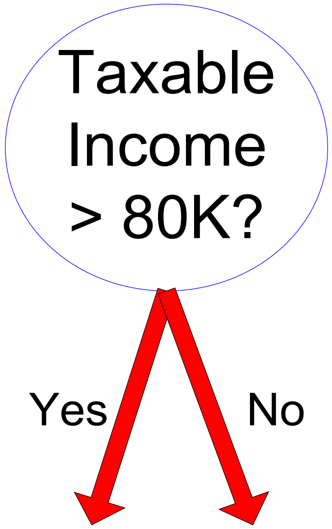
| | CarType | |
|---|---|---|
| | {Sports, Luxury} | {Family} |
| C1 | 3 | 1 |
| C2 | 2 | 4 |
| Gini | ? | |

| | CarType | |
|---|---|---|
| | {Sports} | {Family, Luxury} |
| C1 | 2 | 2 |
| C2 | 1 | 5 |
| Gini | ? | |

# Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

| *Tid* | 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

| Defaulted | No | No | No | Yes | Yes | Yes | No | No | No | No |
|---|---|---|---|---|---|---|---|---|---|---|
| | 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 | |

# Alternative Splitting Criteria based on INFO

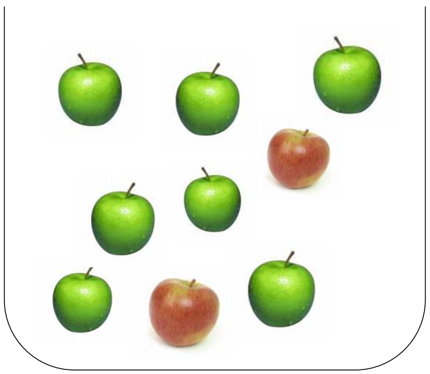
- Entropy at a given node t:

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

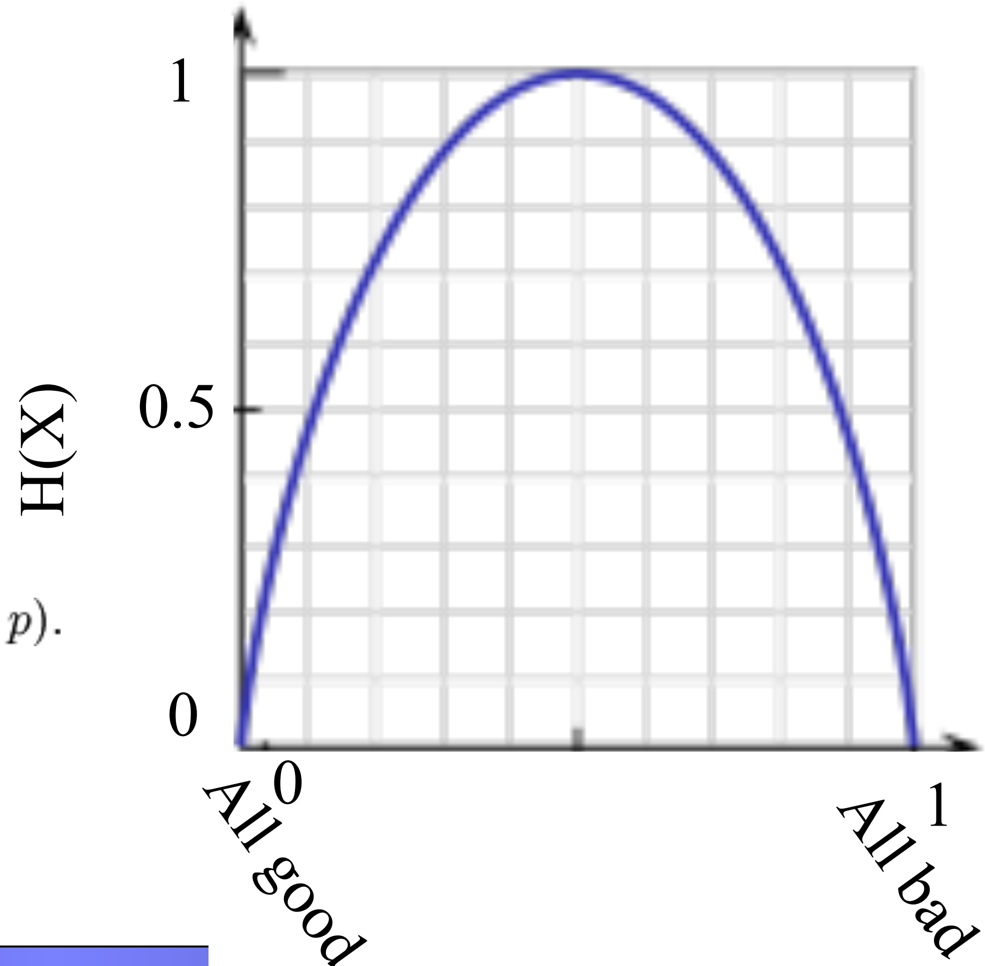
(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  – 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
  – Entropy based computations are similar to the GINI index computations

# Entropy



I have a box of apples…




Pr($X$ = good) = $p$
then Pr($X$ = bad) = 1 − $p$
the entropy of $X$ is given by

$$H(X) = H_b(p) = -p\log p - (1-p)\log(1-p).$$

binary entropy function attains its maximum value when $p$ = 0.5

# 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) $\log_2$ (5/6) = 0.65**

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

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

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

# Splitting Based on INFO...

- Information Gain:

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

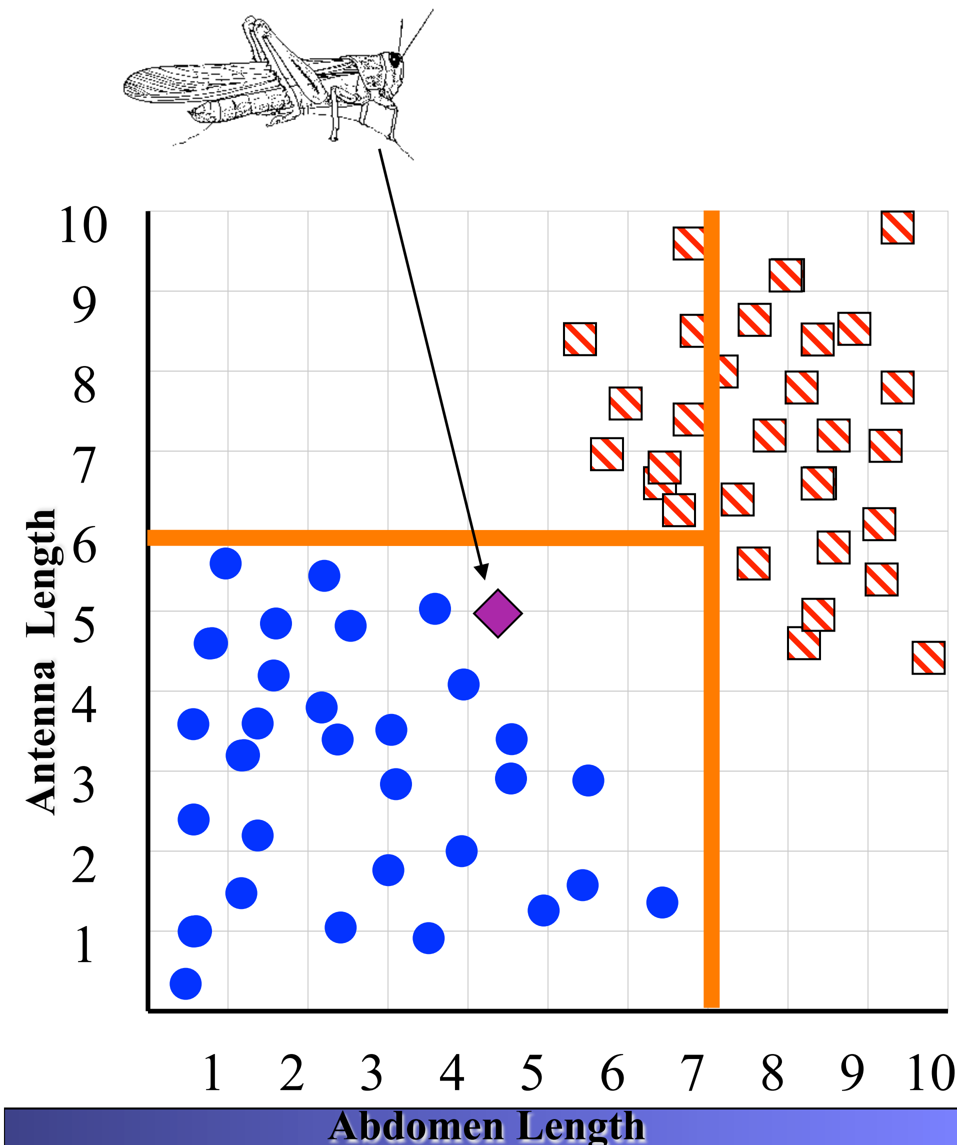
Parent Node, p is split into k partitions;

$n_i$ is number of records in partition i

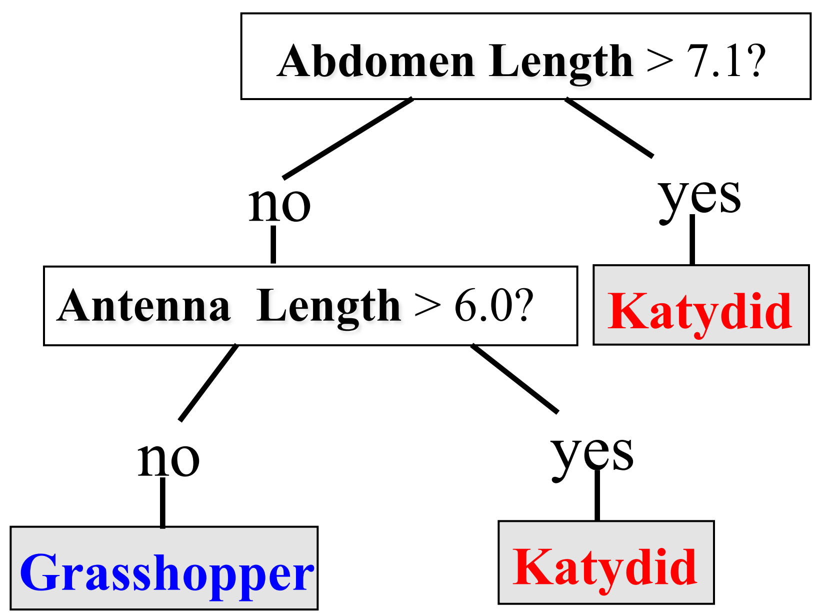
$$\Longrightarrow \quad Gain(split) = E(Parent\ set) - \sum E(all\ child\ sets)$$

- 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.

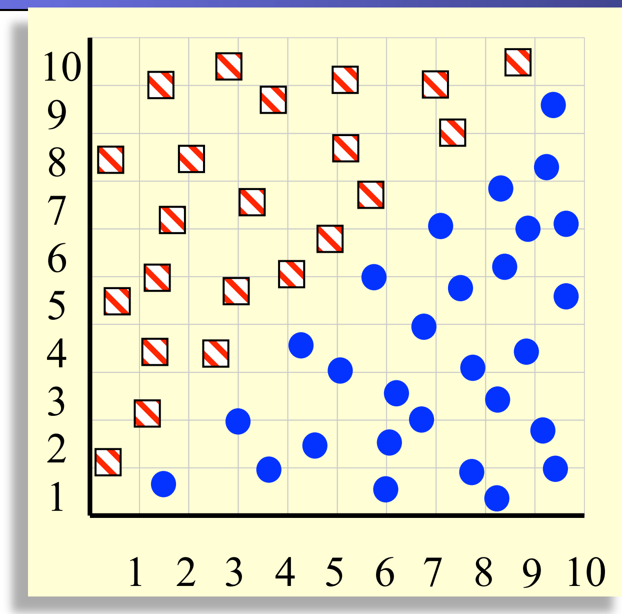
# Back To Our Insect Problem

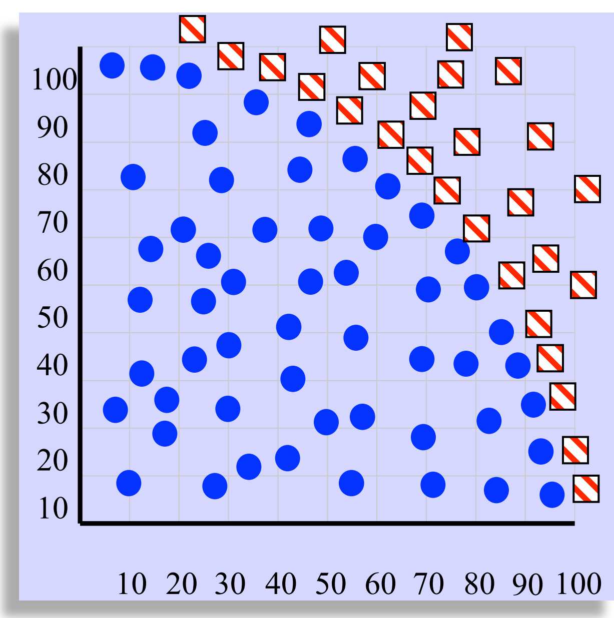


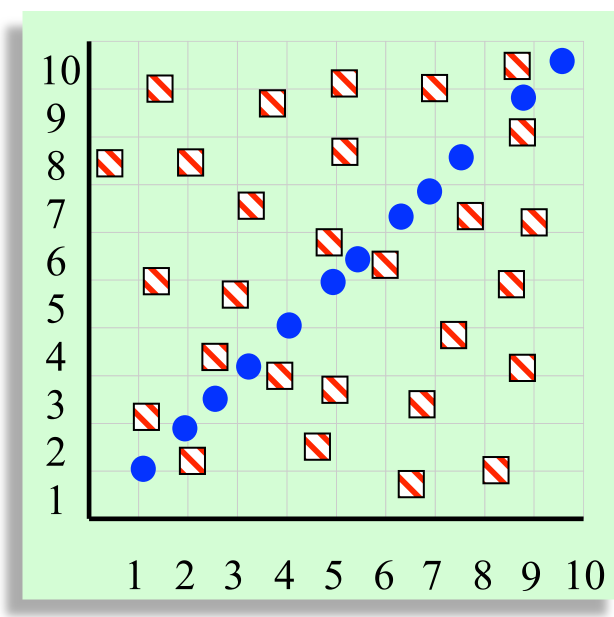

Ross Quinlan



Which of the “Pigeon Problems” can be solved by a Decision Tree?

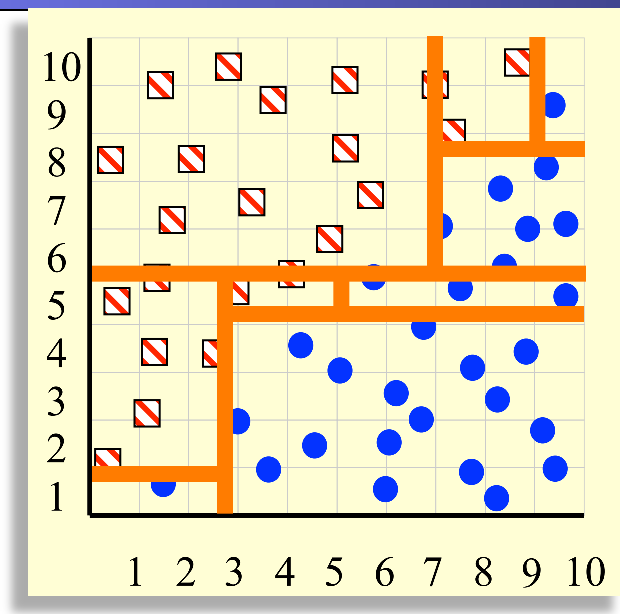


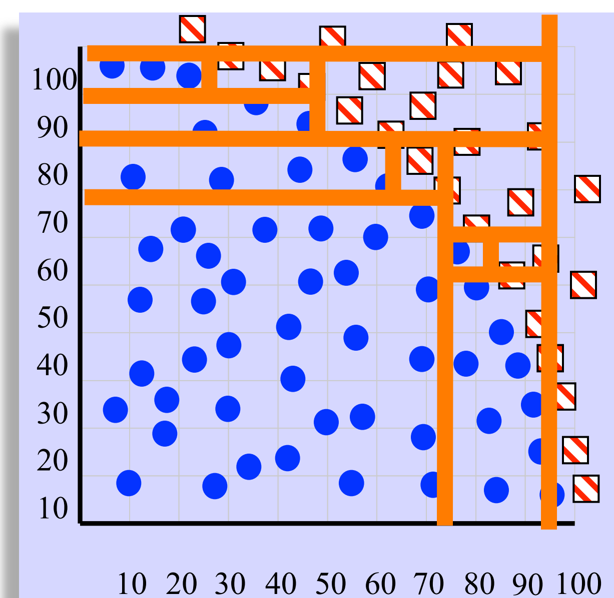



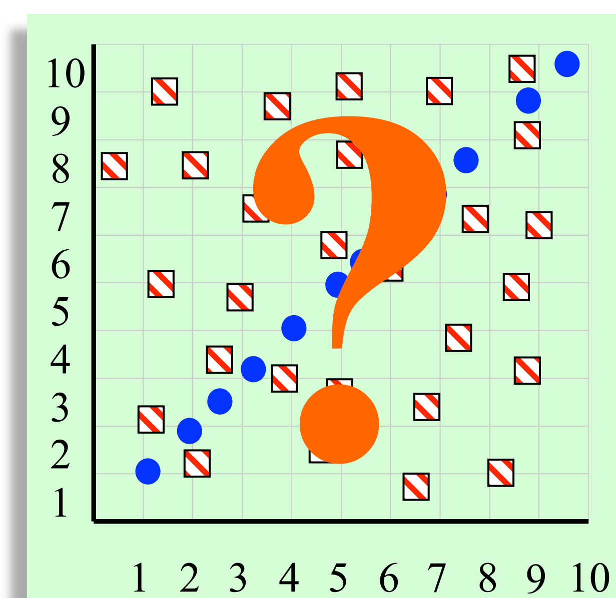
Which of the "Pigeon Problems" can be solved by a Decision Tree?

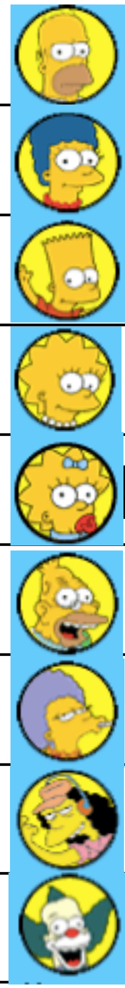
Deep Bushy Tree
Useless
Deep Bushy Tree




The Decision Tree has a hard time with correlated attributes

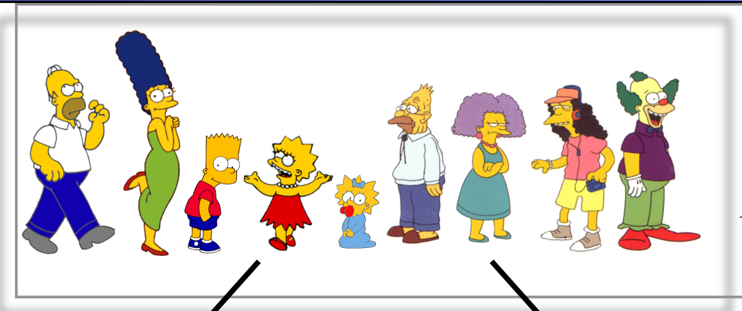




| Person | Hair Length | Weight | Age | Class |
|---|---|---|---|---|
| Homer | 0” | 250 | 36 | M |
| Marge | 10” | 150 | 34 | F |
| Bart | 2” | 90 | 10 | M |
| Lisa | 6” | 78 | 8 | F |
| Maggie | 4” | 20 | 1 | F |
| Abe | 1” | 170 | 70 | M |
| Selma | 8” | 160 | 41 | F |
| Otto | 10” | 180 | 38 | M |
| Krusty | 6” | 200 | 45 | M |

| | | | | |
|---|---|---|---|---|
|  Comic | 8” | 290 | 38 | ? |



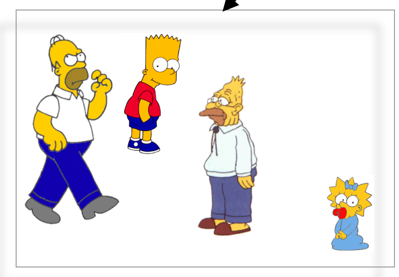
$$Entropy(t) = -\sum_j p(j\,|\,t)\log p(j\,|\,t)$$

$$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9) = \mathbf{0.9911}$$

yes

no

Hair Length <= 5?



Let us try splitting on *Hair length*

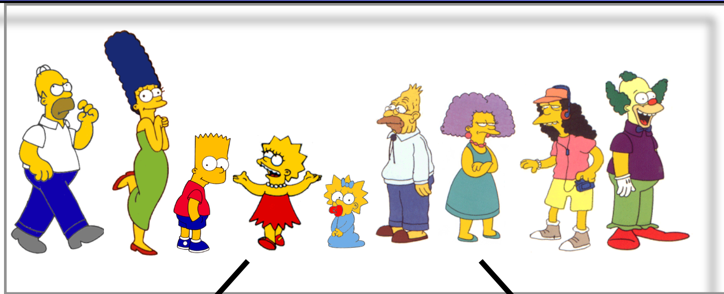
$$Entropy(1\mathbf{F},3\mathbf{M}) = -(1/4)\log_2(1/4) - (3/4)\log_2(3/4) = \mathbf{0.8113}$$

$$Entropy(3\mathbf{F},2\mathbf{M}) = -(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = \mathbf{0.9710}$$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

*Gain*(Hair Length <= 5) = **0.9911** – (4/9 * **0.8113** + 5/9 * **0.9710** ) = **0.0911**

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



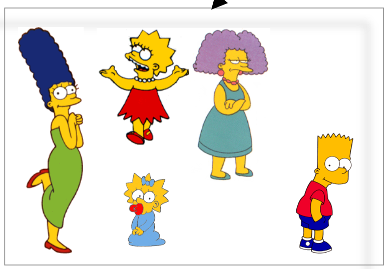
$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$
$= \mathbf{0.9911}$

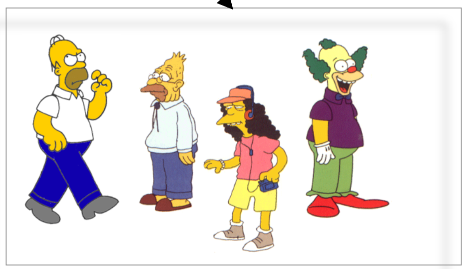
yes

no

Weight <= 160?

Let us try splitting on *Weight*





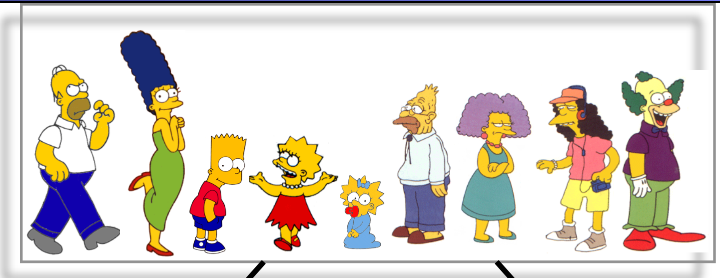
$Entropy(4\mathbf{F},1\mathbf{M}) = -(4/5)\log_2(4/5) - (1/5)\log_2(1/5)$
$= \mathbf{0.7219}$

$Entropy(0\mathbf{F},4\mathbf{M}) = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4)$
$= \mathbf{0}$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

*Gain*(Weight <= 160) = **0.9911** – (5/9 * **0.7219** + 4/9 * **0** ) = **0.5900**

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

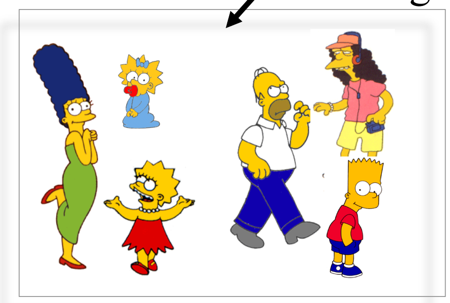


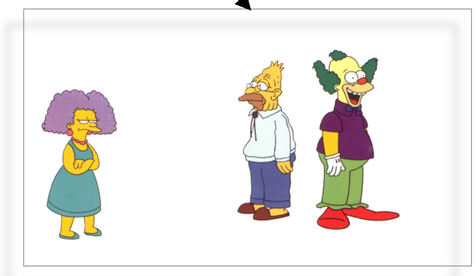
$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$
$= \mathbf{0.9911}$

yes

no

age <= 40?





Let us try splitting on *Age*

$Entropy(3\mathbf{F},3\mathbf{M}) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6)$
$= \mathbf{1}$

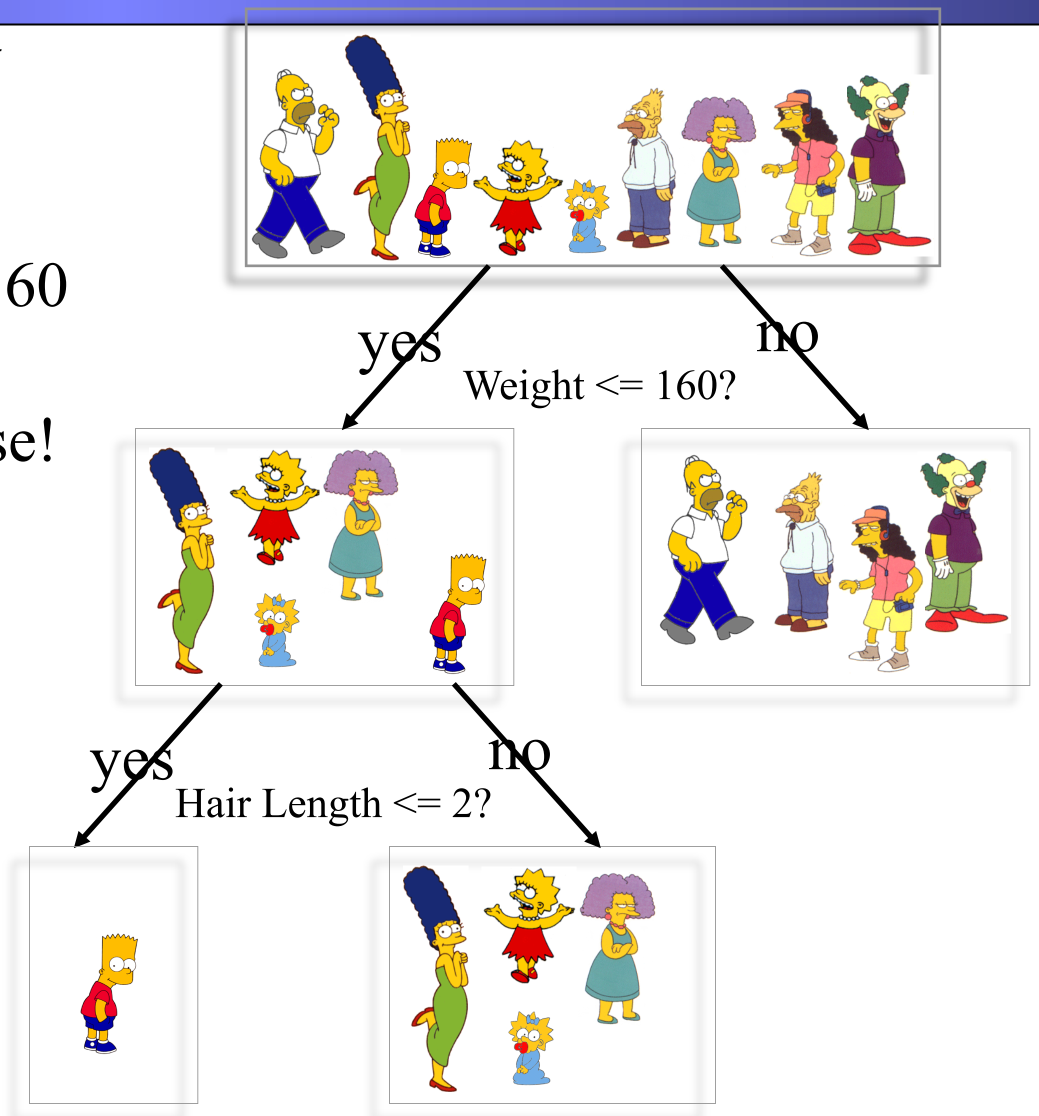
$Entropy(1\mathbf{F},2\mathbf{M}) = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= \mathbf{0.9183}$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$Gain(\text{Age} <= 40) = \mathbf{0.9911} - (6/9 * \mathbf{1} + 3/9 * \mathbf{0.9183}) = \mathbf{0.0183}$

Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified… So we simply recurse!

This time we find that we can split on *Hair length,* and we are done!




We'll talk more about stopping criteria later.

# Splitting Based on INFO...

- 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!
  - Used in C4.5
  - Designed to overcome the disadvantage of Information Gain

# Splitting Criteria based on Classification Error

- Classification error at a node t :

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

- Measures misclassification error made by a node.
  - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

# Examples for Computing Error

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

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

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

**Error = 1 – max (0, 1) = 1 – 1 = 0**

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

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

**Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6**

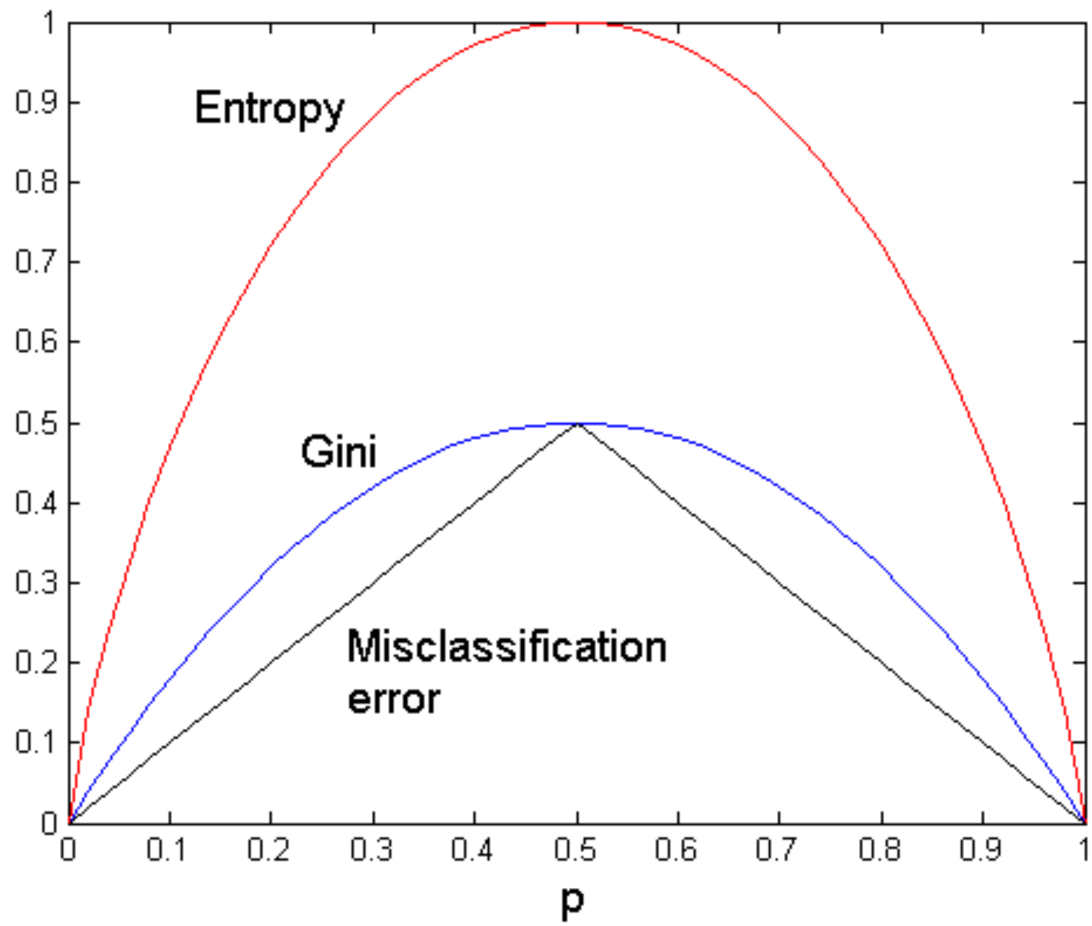
| | |
|---|---|
| C1 | **2** |
| C2 | **4** |

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

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

# Comparison among Splitting Criteria

**For a 2-class problem:**




**P refers to the fraction of records that belong to one of the two classes**

# 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

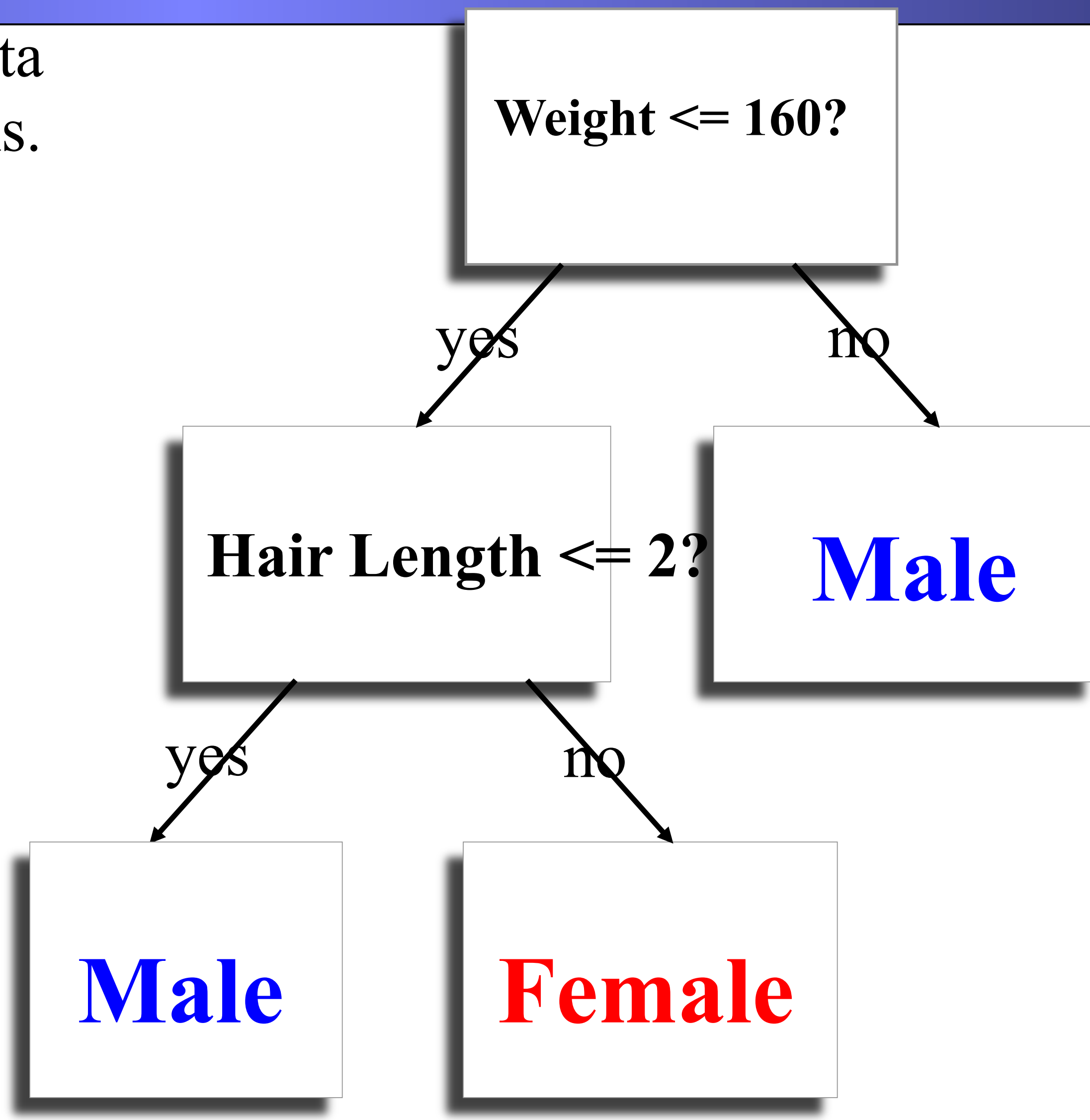
# Stopping Criteria for Tree Induction

- 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 (to be discussed later)

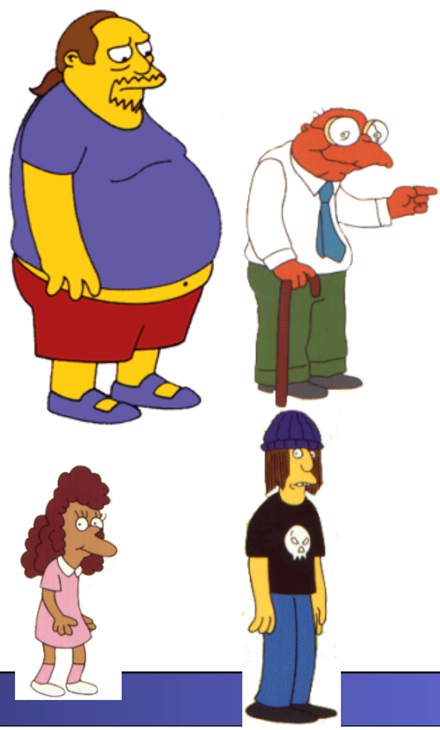
# Decision Tree Based Classification

- 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

We don’t need to keep the data around, just the test conditions.

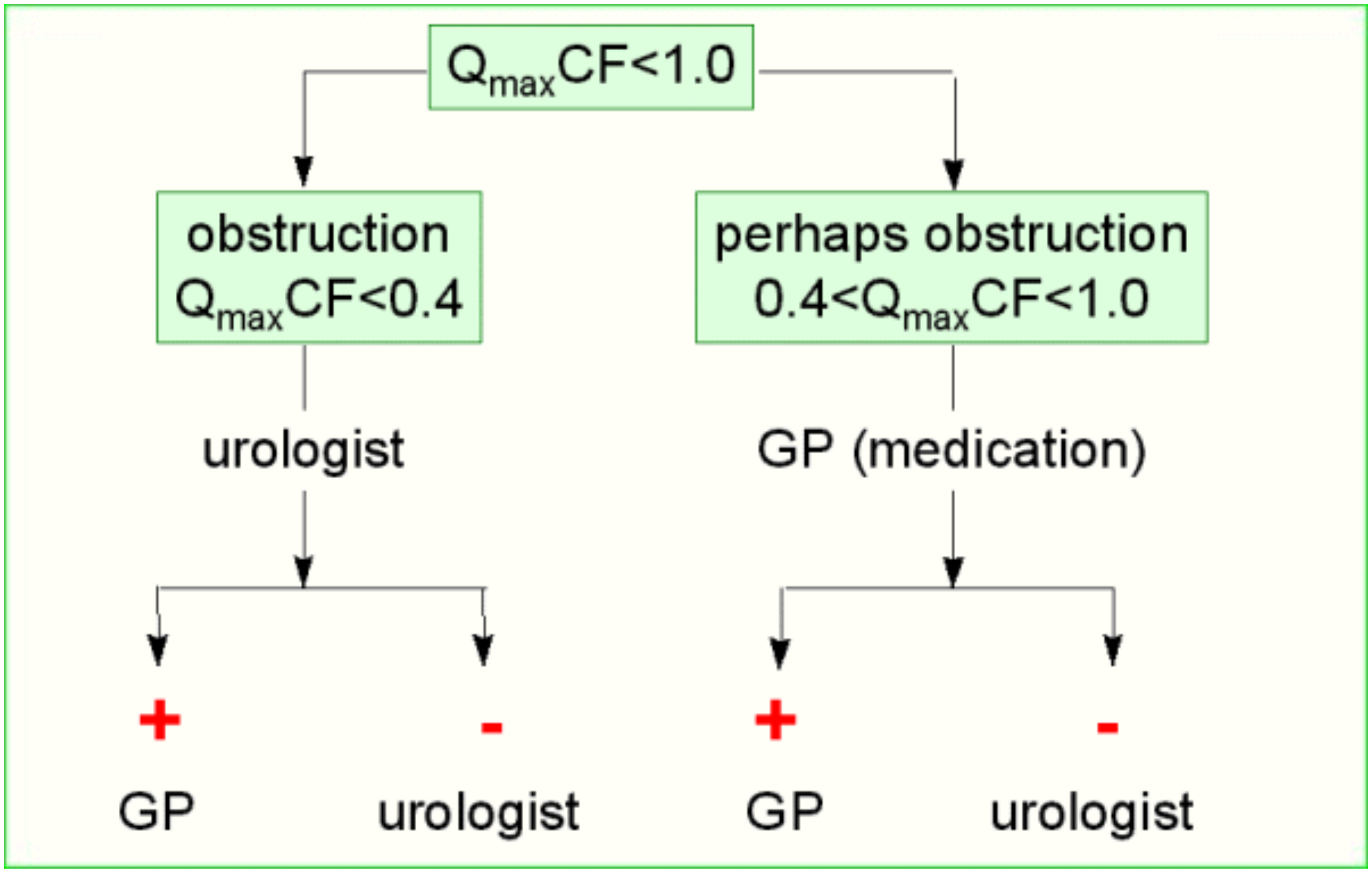



How would these people be classified?

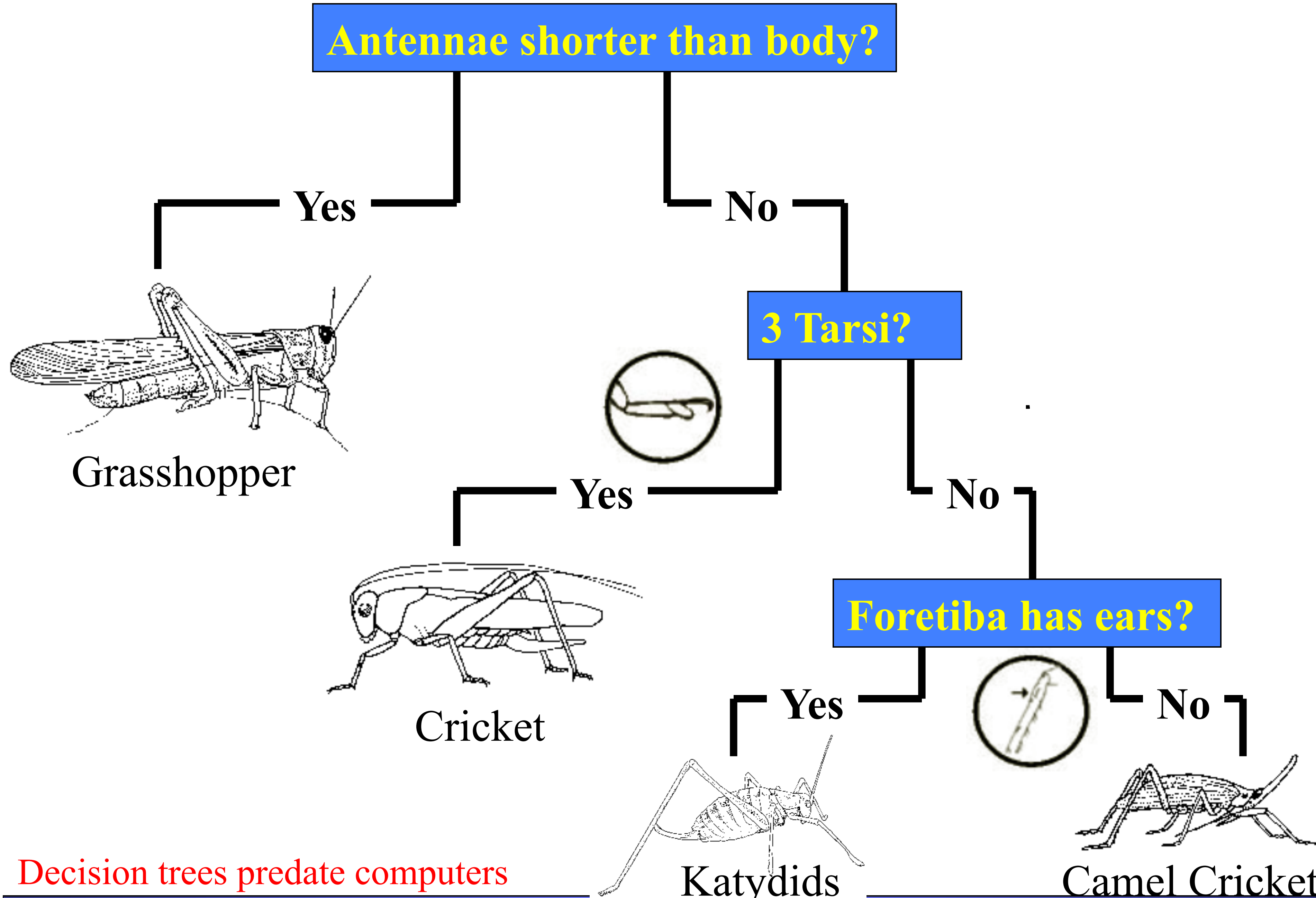


# Once we have learned the decision tree, we don't even need a computer!

This decision tree is attached to a medical machine, and is designed to help nurses make decisions about what type of doctor to call.




Decision tree for a typical shared-care setting applying the system for the diagnosis of prostatic obstructions.




Decision trees predate computers

# Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

# Practical Issues of Classification

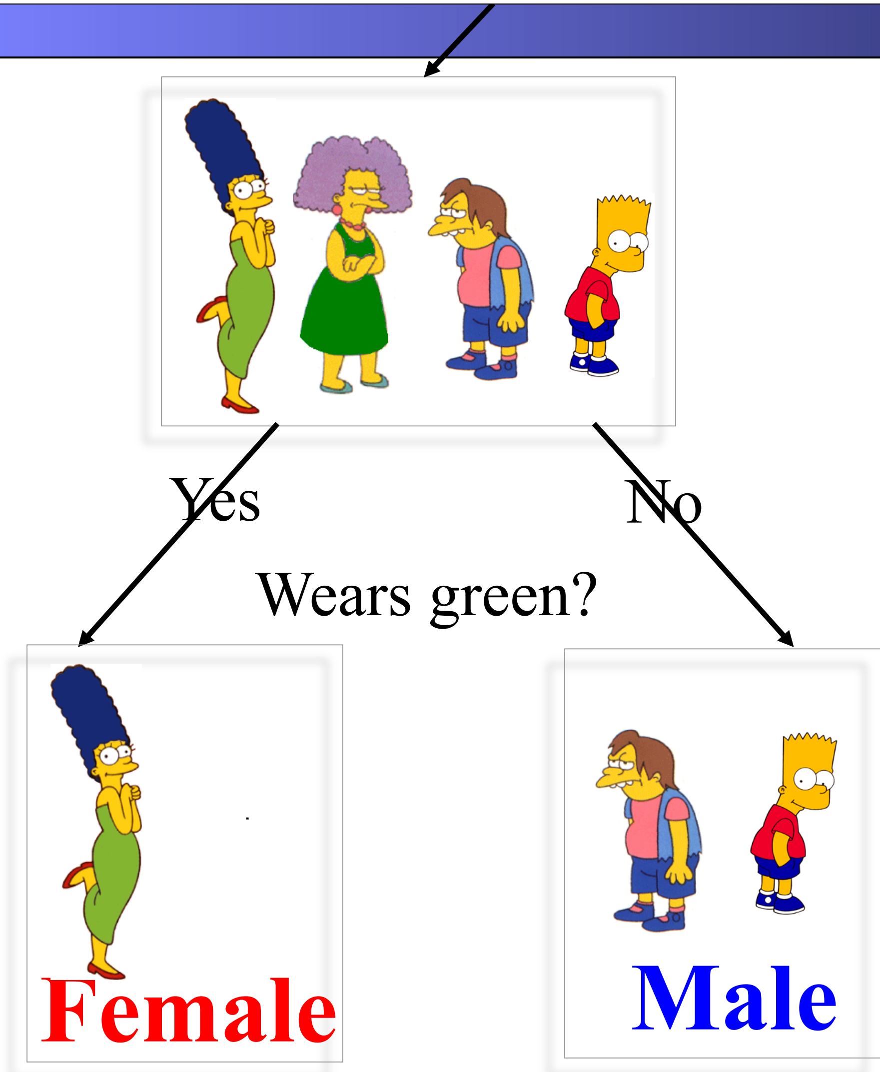
- Underfitting and Overfitting

- Missing Values

- Costs of Classification

The previous examples we have seen were performed on small datasets. However with small datasets there is a great danger of overfitting the data…

When you have few data points, there are many possible splitting rules that perfectly classify the data, but will not generalize to future datasets.

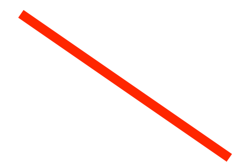



For example, the rule "Wears green?" perfectly classifies the data, so does "Mother's name is Jacqueline?", so does "Has blue shoes"…

Suppose we need to solve a classification problem

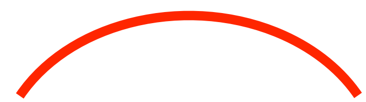
We are not sure if we should use the..

- Simple linear classifier



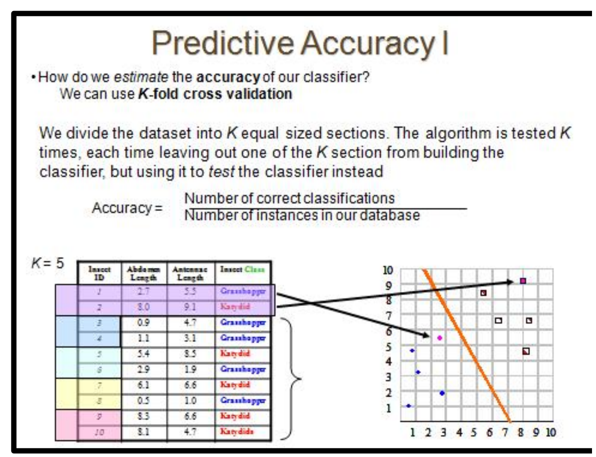
or the

- Simple quadratic classifier

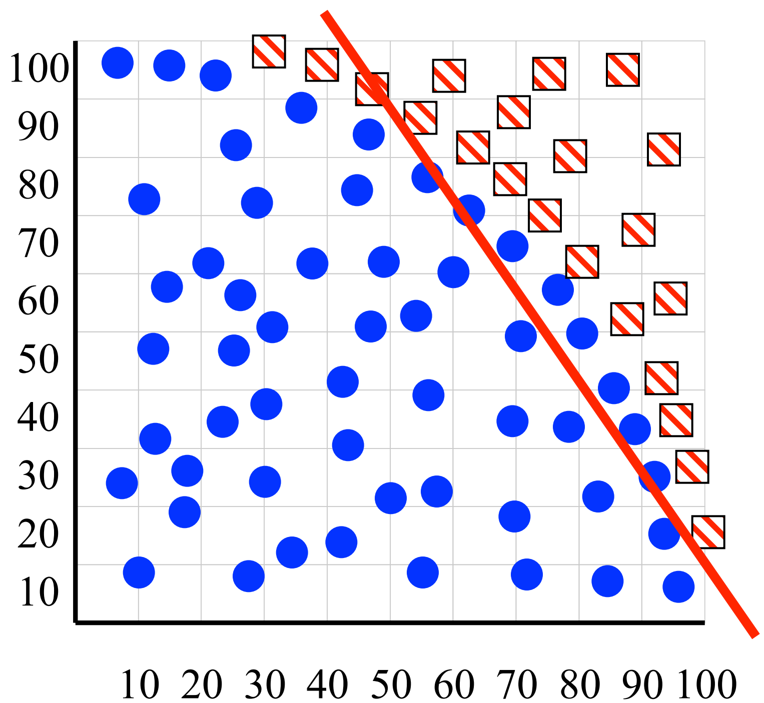


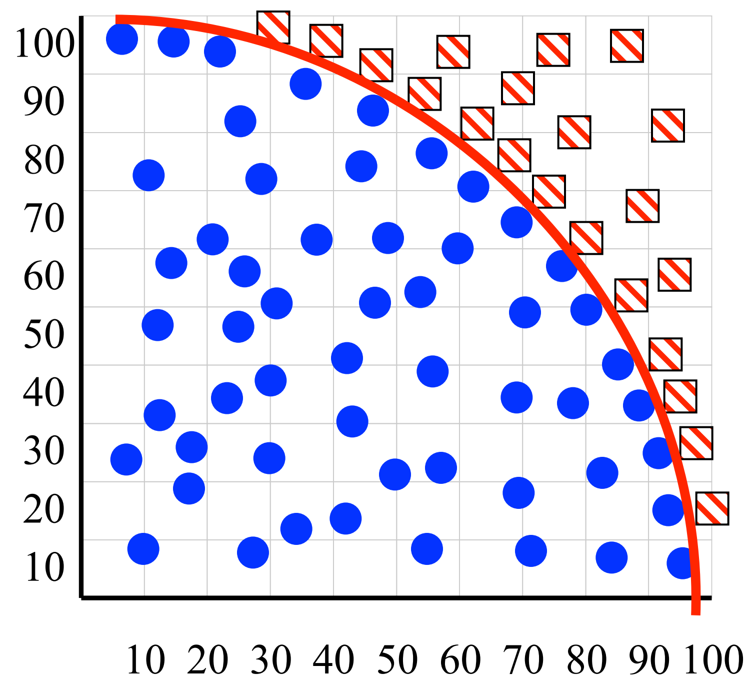
How do we decide which to use?

We do cross validation (discussed later) and choose the best one.

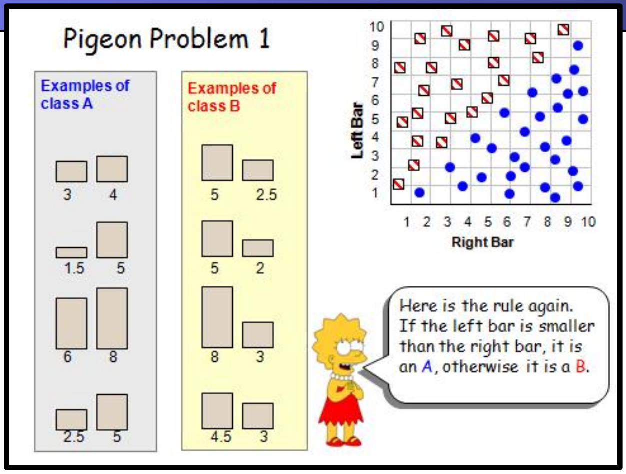


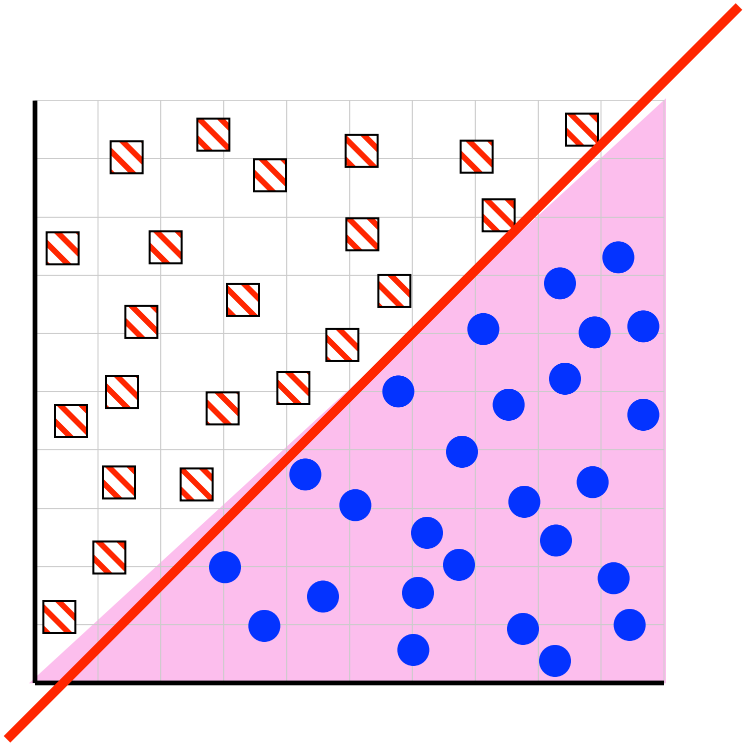
• Simple linear classifier gets 81% accuracy
• Simple quadratic classifier gets 99% accuracy

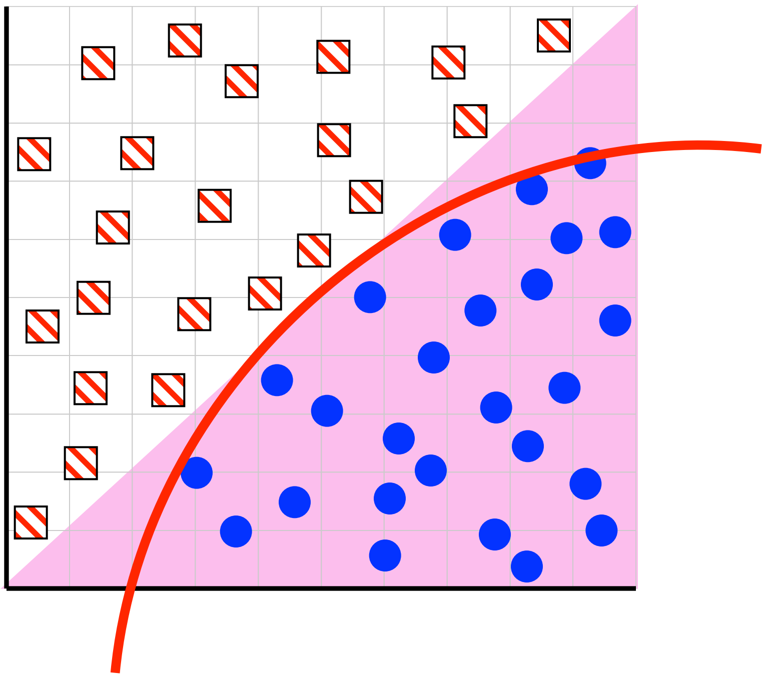


- Simple linear classifier gets 96% accuracy
- Simple quadratic classifier 97% accuracy

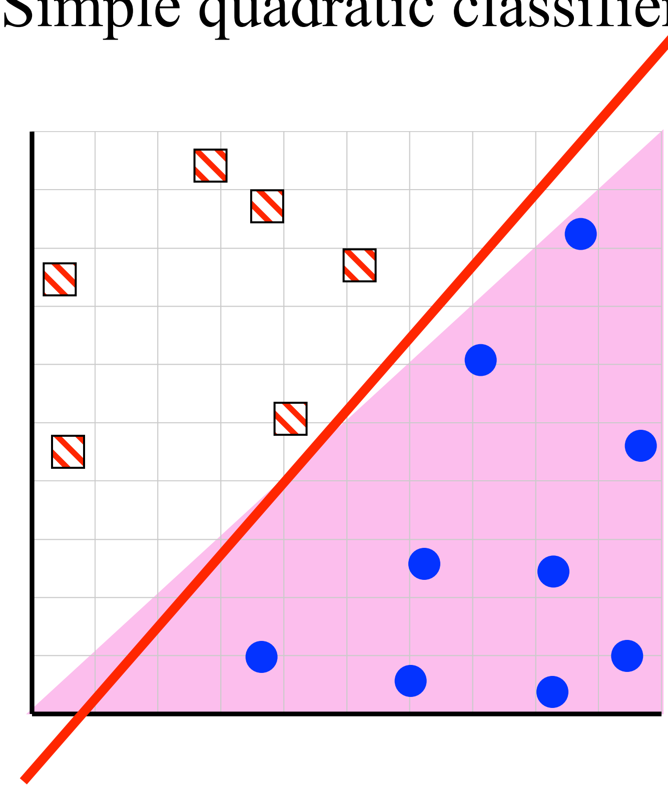


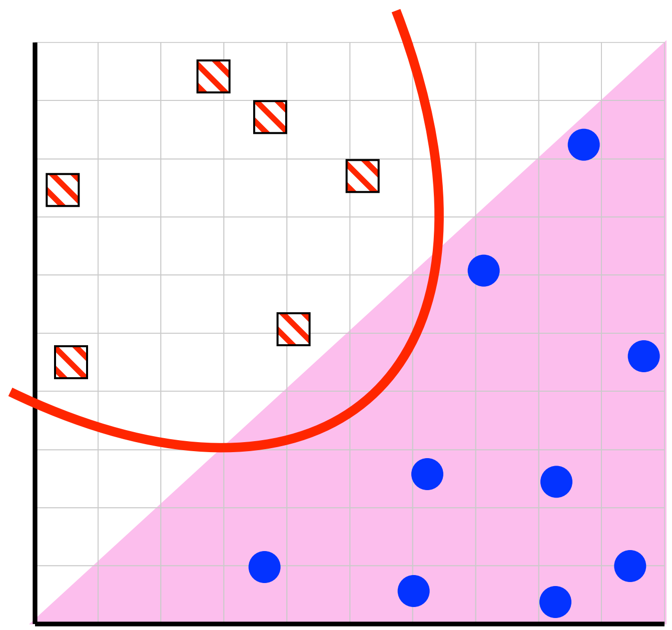





This problem is greatly exacerbated by having too little data

- Simple linear classifier gets 90% accuracy
- Simple quadratic classifier 95% accuracy

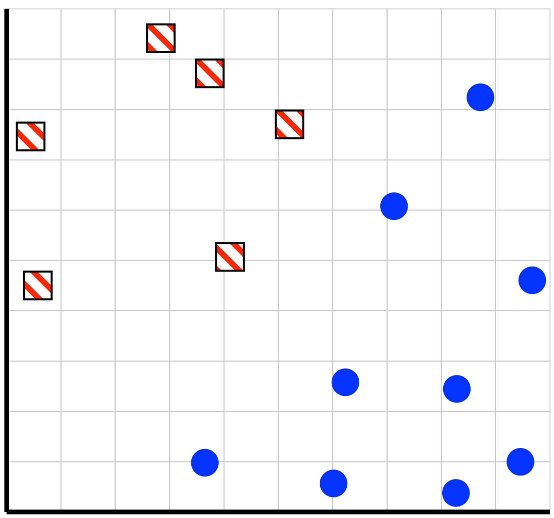




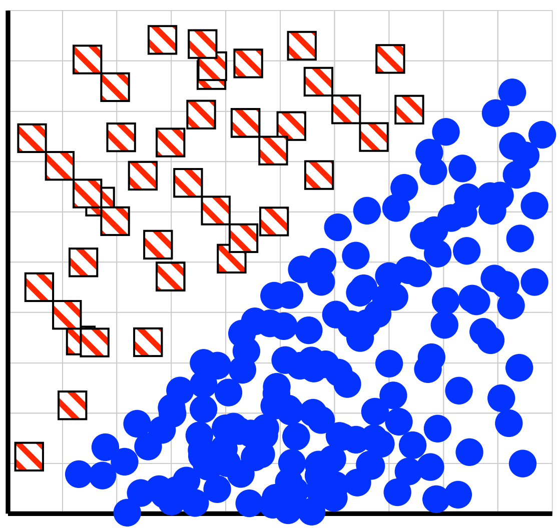
What happens as we have more and more training examples?

The accuracy for all models goes up!
The chance of making a mistake goes down
The cost of the mistake (if made) goes down

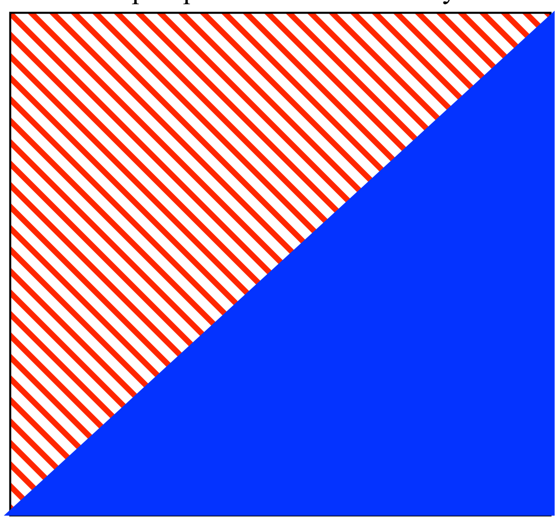
- Simple linear 70% accuracy
- Simple quadratic 90% accuracy



- Simple linear 90% accuracy
- Simple quadratic 95% accuracy



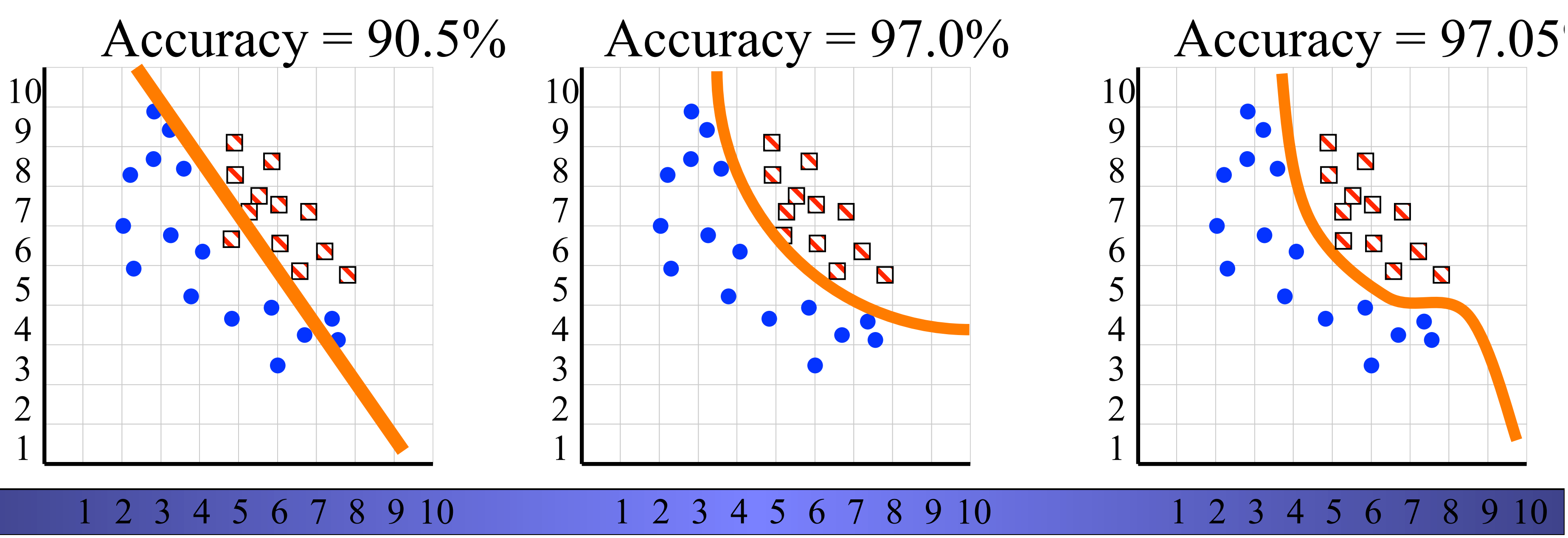
- Simple linear 99% accuracy
- Simple quadratic 99% accuracy



# One Solution: Charge Penalty for complex models

- For example, for the simple {polynomial} classifier, we could charge 1% for every increase in the degree of the polynomial

  - Simple linear classifier gets 90.5% accuracy, minus 0, equals 90.5%
  - Simple quadratic classifier 97.0% accuracy, minus 1, equals 96.0%
  - Simple cubic classifier 97.05% accuracy, minus 2, equals 95.05%

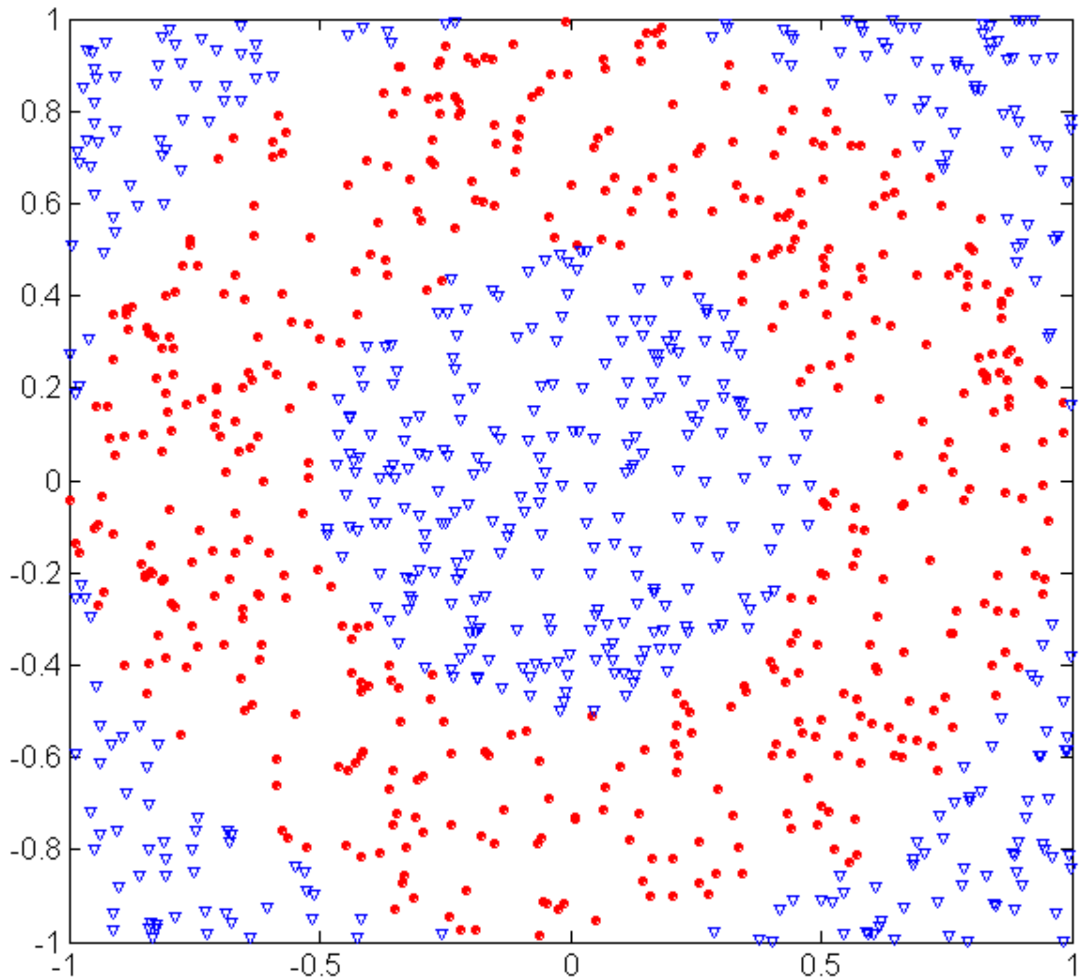


# One Solution: Charge Penalty for complex models

• For example, for the simple {polynomial} classifier, we could charge 1% for every increase in the degree of the polynomial.

• There are more principled ways to charge penalties
• In particular, there is a technique called **Minimum Description Length** (MDL)

# Underfitting and Overfitting (Example)




**500 circular and 500 triangular data points.**

**Circular points:**

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

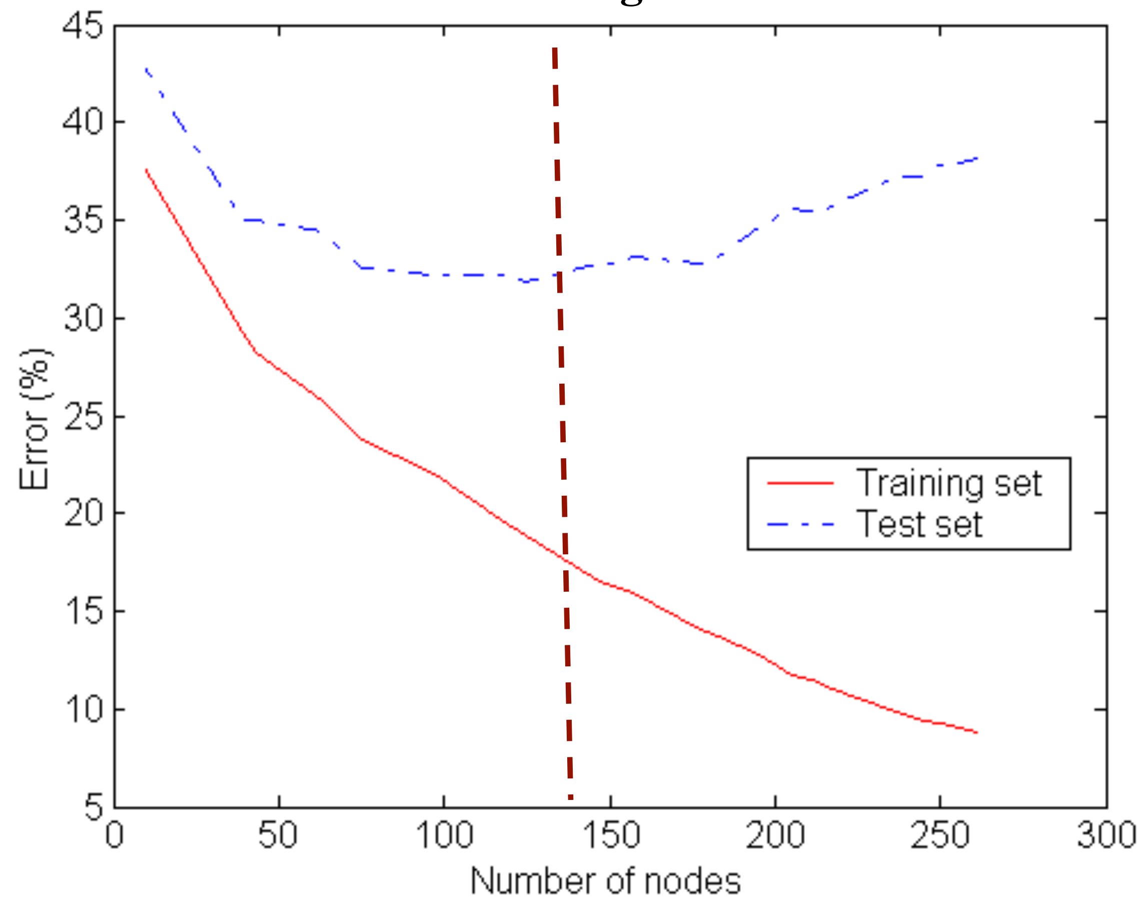
**Triangular points:**

**$\mathrm{sqrt}(x_1^2+x_2^2) > 0.5$ or**

**$\mathrm{sqrt}(x_1^2+x_2^2) < 1$**

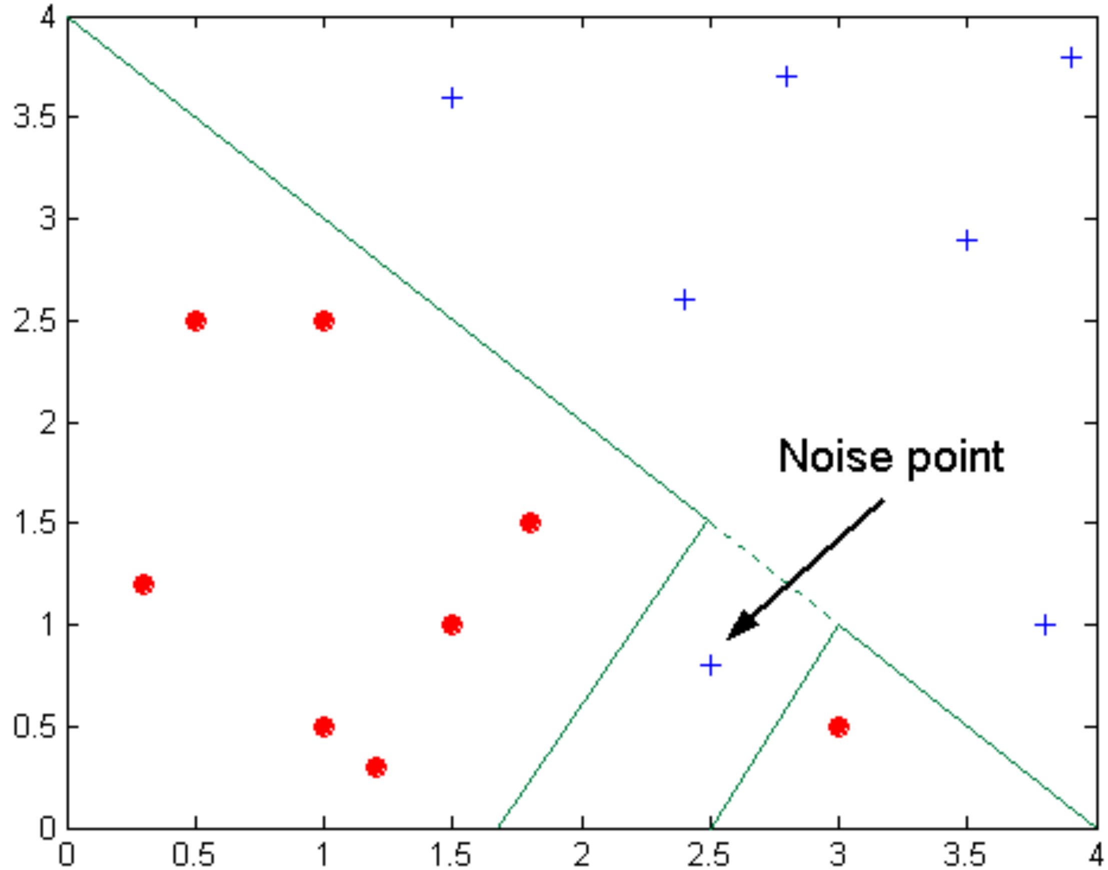
# The Fitting Curve: Overfitting vs. Underfitting

**Overfitting**



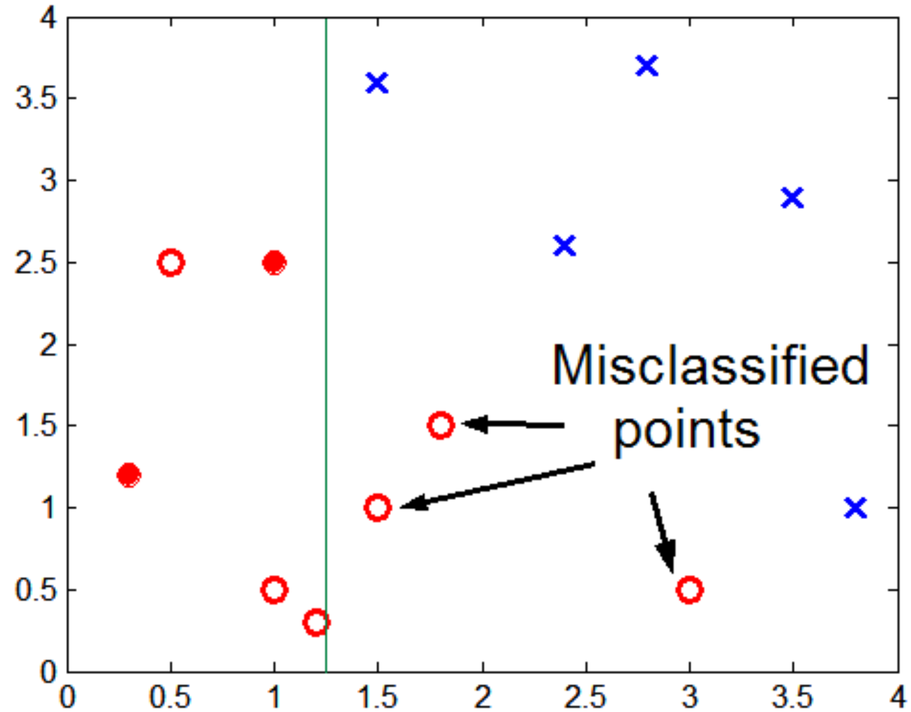

**Underfitting**: when model is too simple, both training and test errors are large

# Overfitting due to Noise




**Decision boundary is distorted by noise point**

# Overfitting due to Insufficient Examples




**Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region**

**- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task**

# Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

# Estimating Generalization Errors

- Re-substitution errors: error on training ($\Sigma$ e(t) )
- Generalization errors: error on testing ($\Sigma$ e' (t))

# Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
- For complex models, there is a greater chance that it was fitted accidentally by errors in data
- Therefore, one should include model complexity when evaluating a model

# How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using $\chi^2$ test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

# How to Address Overfitting…

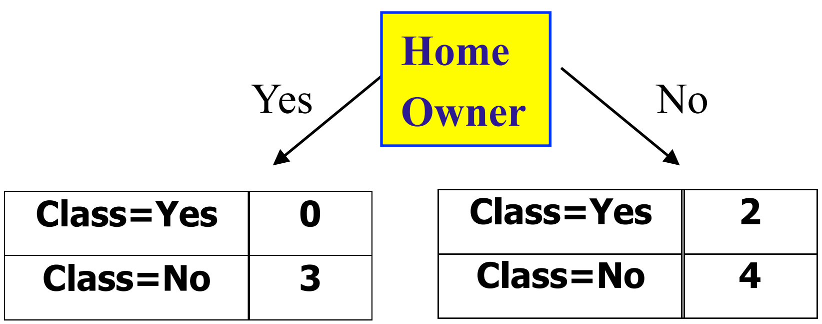
- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree

# Handling Missing Attribute Values

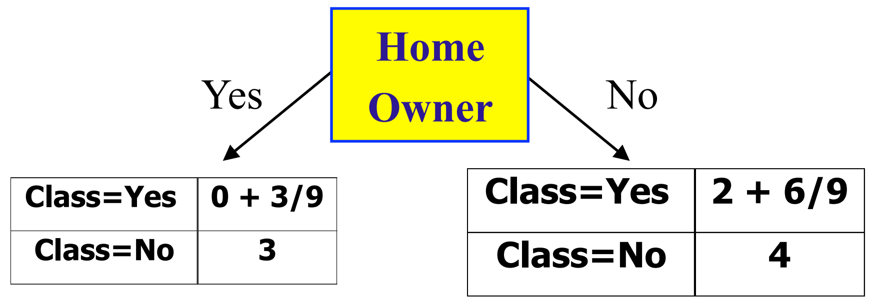
- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

# Distribute Instances

| *Tid* | Home Owner | Marital Status | Annual Income | Class |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |




| *Tid* | Home Owner | Marital Status | Annual Income | Class |
|---|---|---|---|---|
| 10 | ? | Single | 90K | Yes |




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

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

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

# Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

# Data Fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

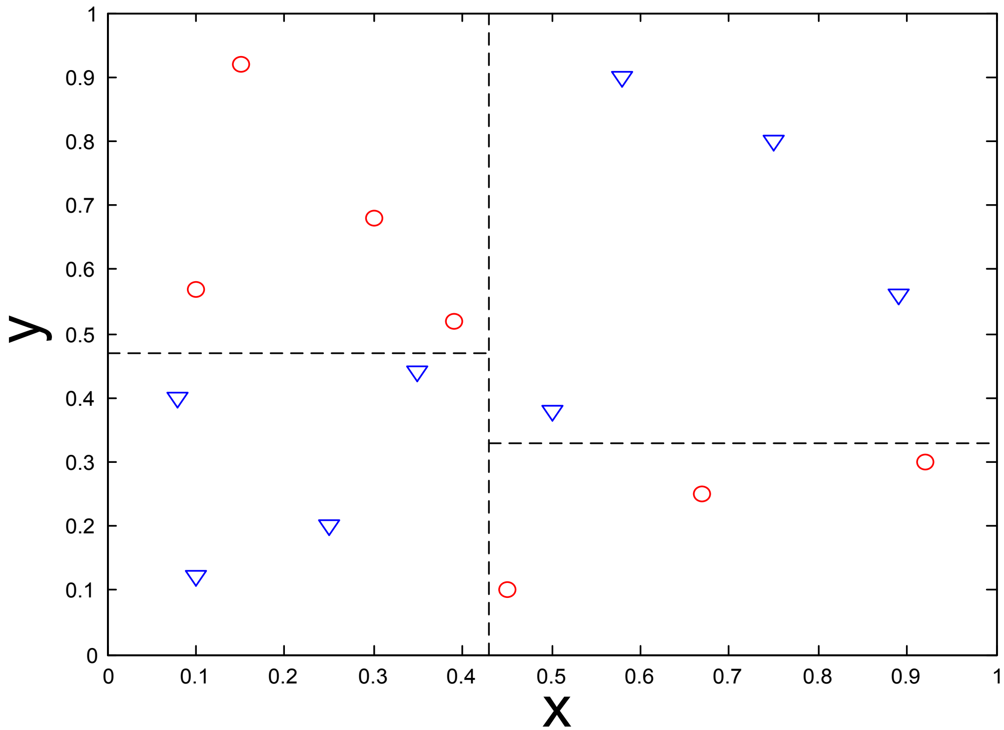
# Search Strategy

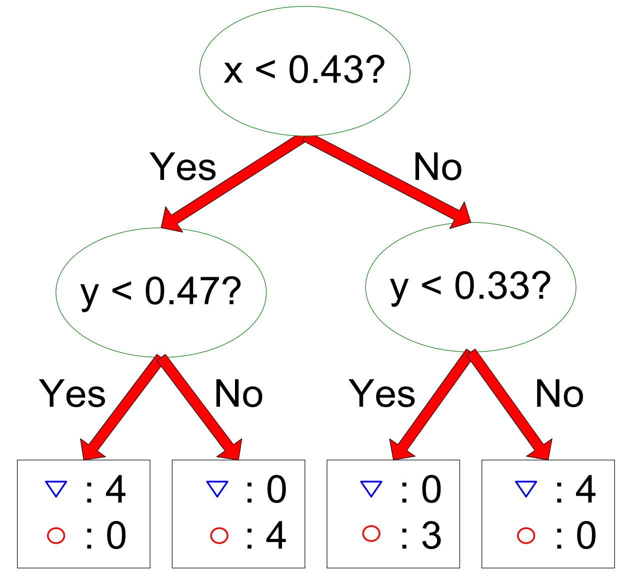
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

# Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at a time

# Decision Boundary

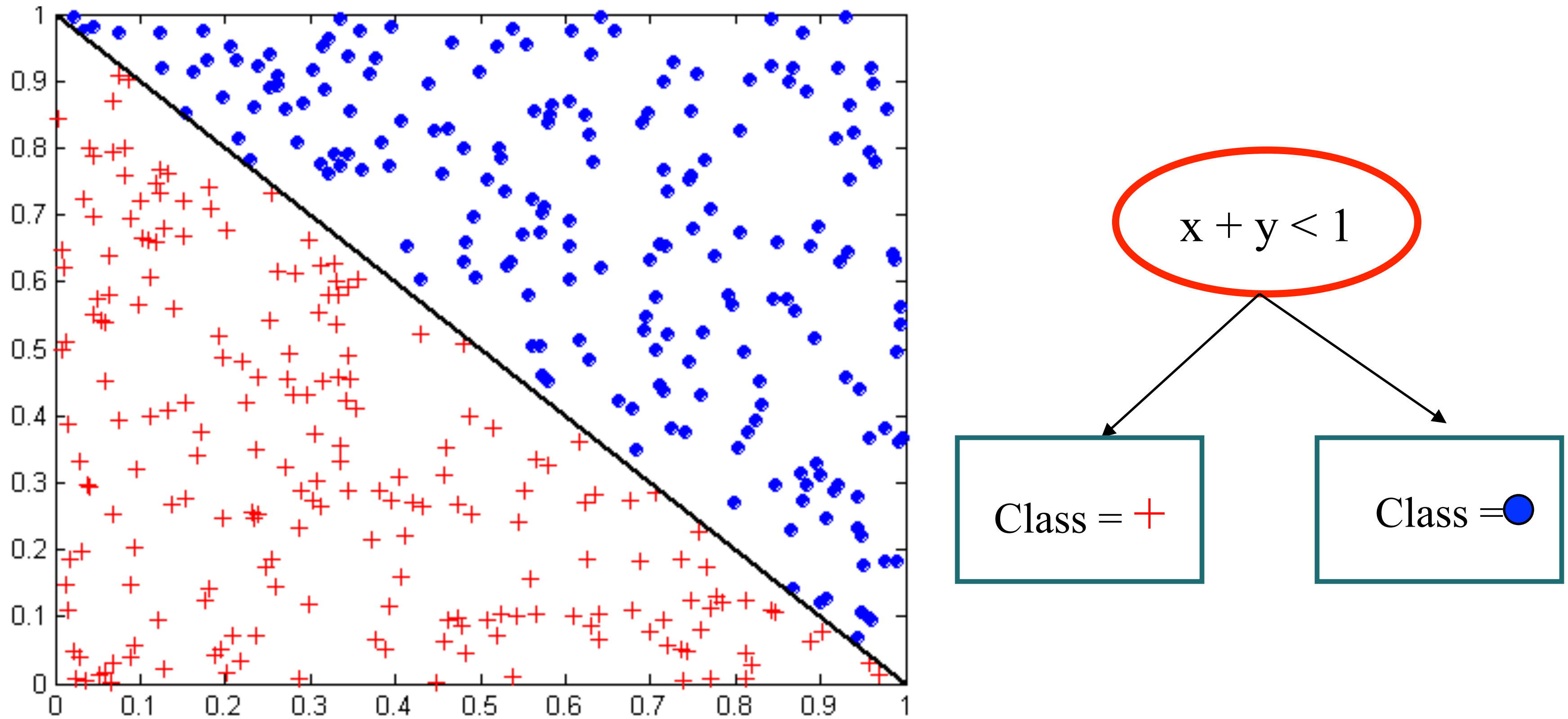



- **Border line between two neighboring regions of different classes is known as decision boundary**

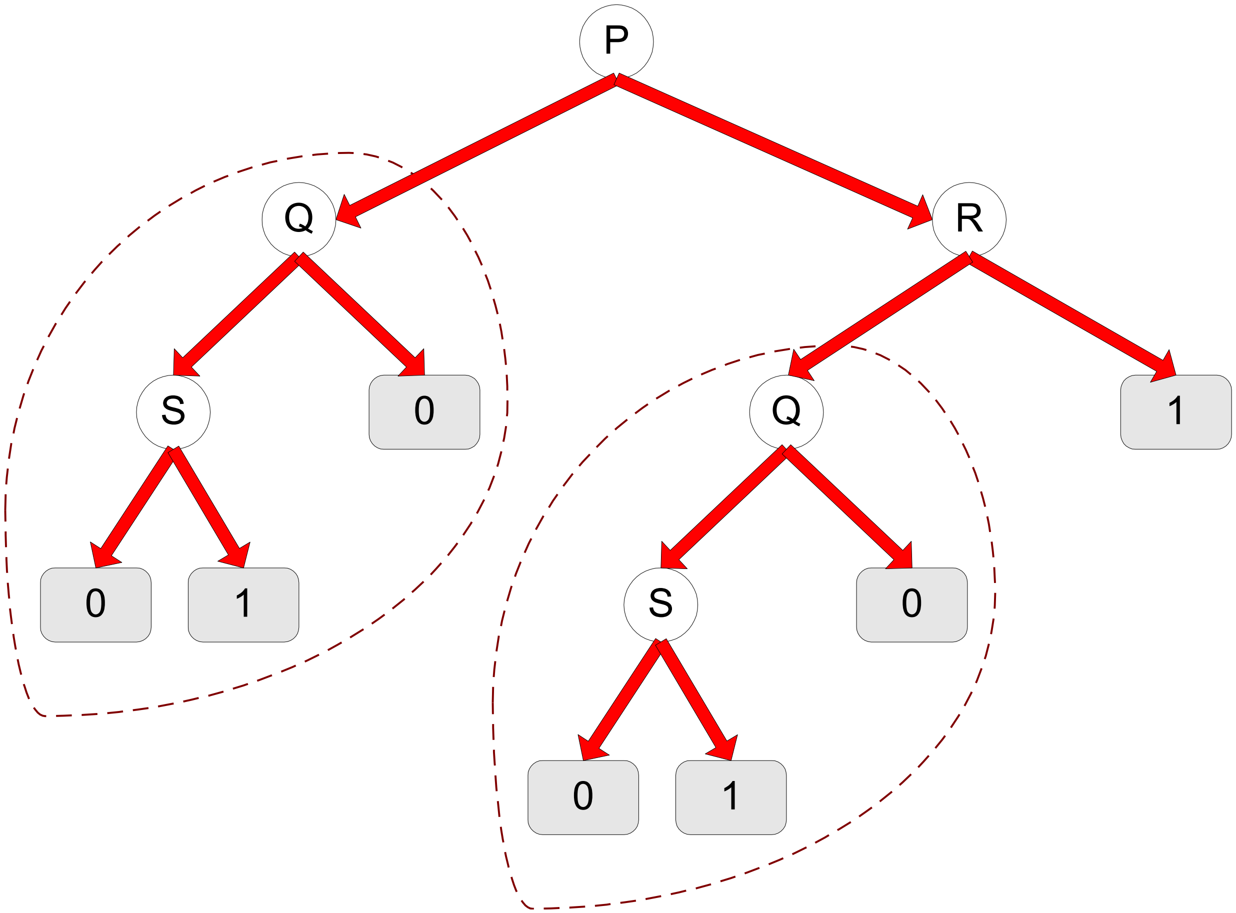
- **Decision boundary is parallel to axes because test condition involves a single attribute at-a-time**

# Oblique Decision Trees




- **Test condition may involve multiple attributes**
- **More expressive representation**
- **Finding optimal test condition is computationally expensive**

# Tree Replication




- **Same subtree appears in multiple branches**