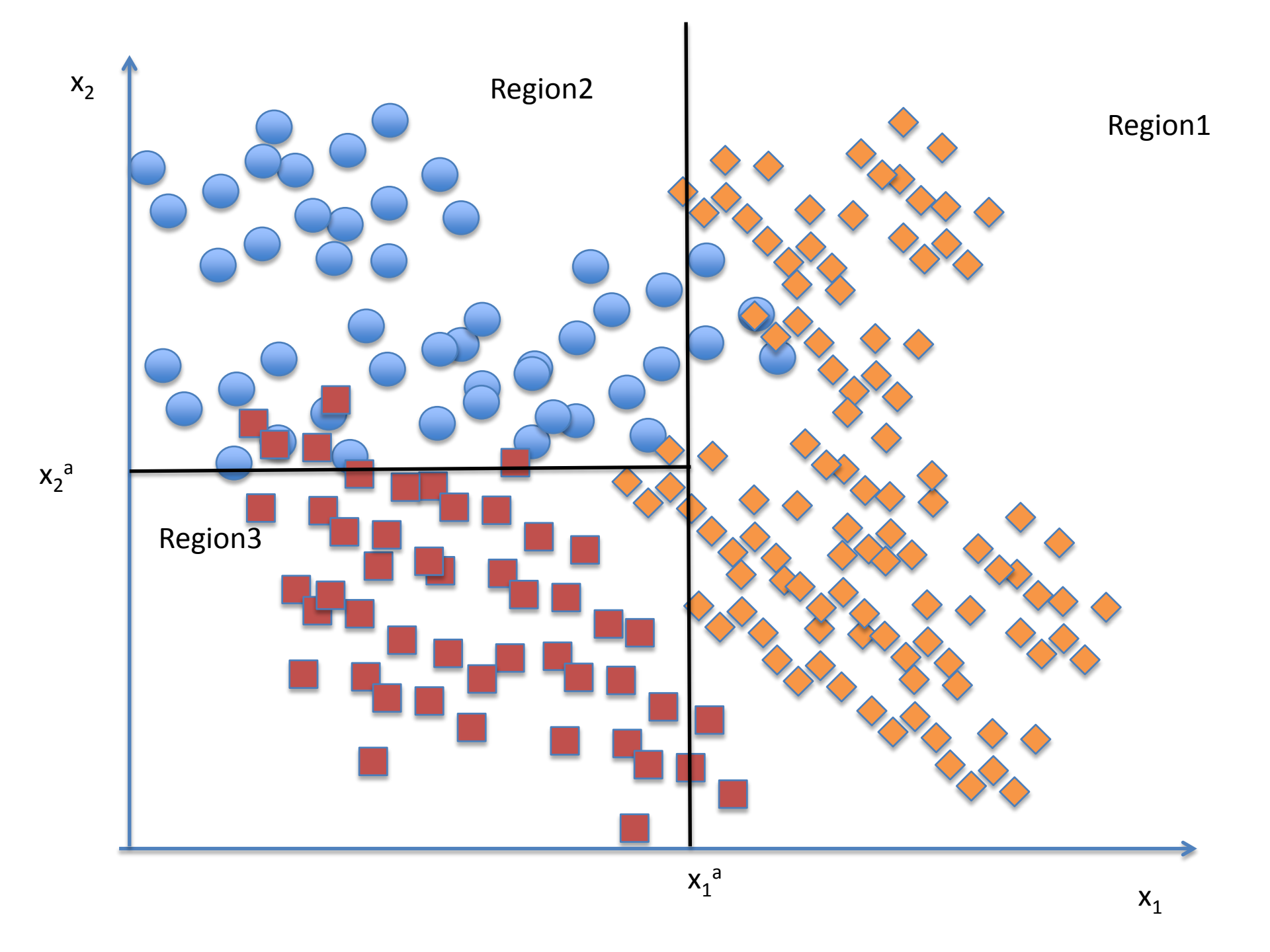
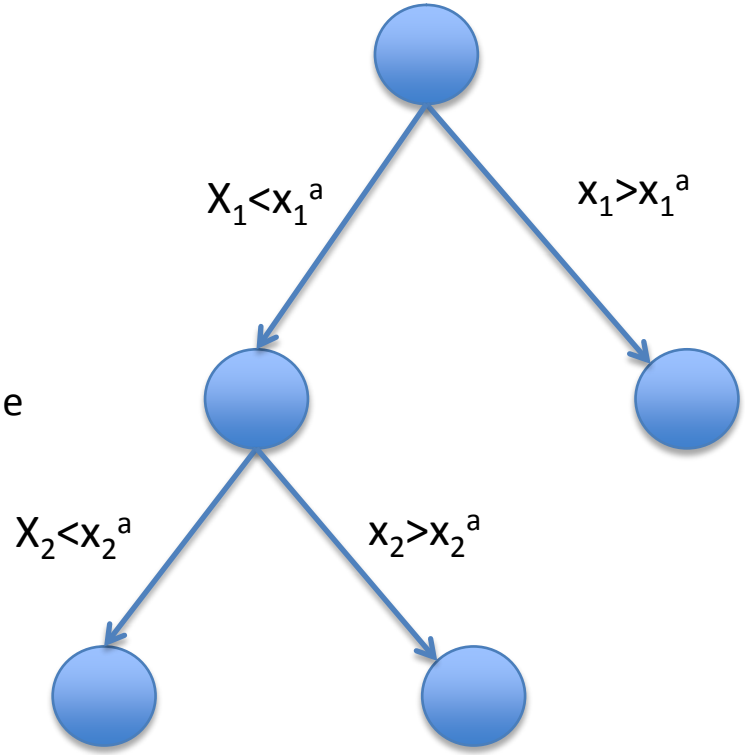
Tree-based methods


x_2
Region2
Region1
x_2^a
Region3
x_1^a
x_1



Left-hand side of the plane

Right-hand side of the plane

Lower Left-hand side of the plane

Upper Left-hand side of the plane

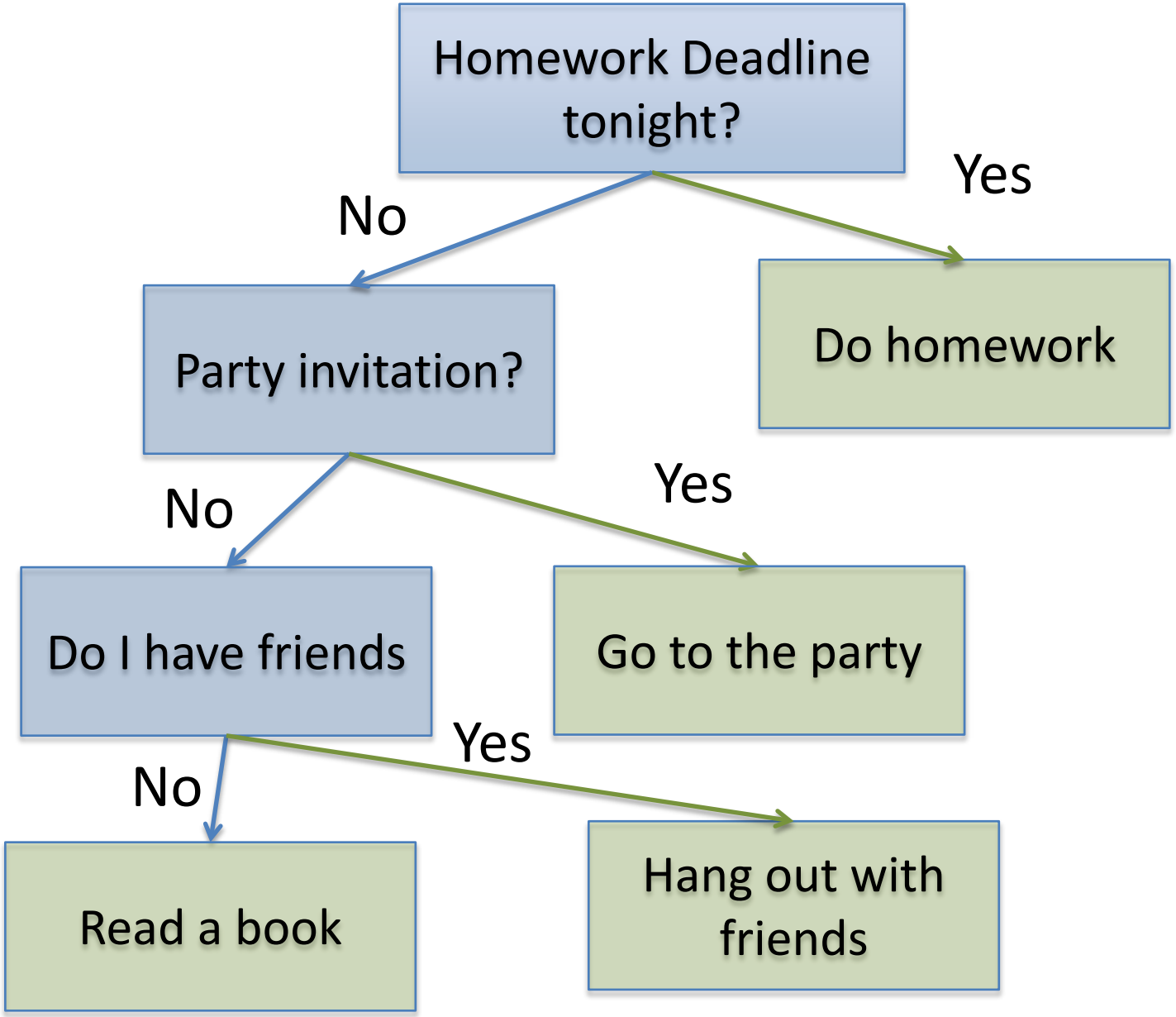
Basic Idea

Segment the predictor space into sub-regions and we learn from the training set the value to predict as the mean or mode or median of the respond variable of the training examples that are in that segment.

Why Trees?

What would you do tonight? Decide amongst the following:

- Finish homework
- Go to a party
- Read a book
- Hang out with friends


Homework Deadline tonight?
No
Yes
Party invitation?
Do homework
No
Yes
Do I have friends
Go to the party
No
Yes
Read a book
Hang out with friends

Why Trees?

We split the predictor space as branches of a tree and therefore these methods are called decision tree methods

Function Approximation

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$

Input: Training examples of unknown target function f

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

Output: Hypothesis $h \in H$ that best approximates f

Based on slide by Tom Mitchell

Sample Dataset

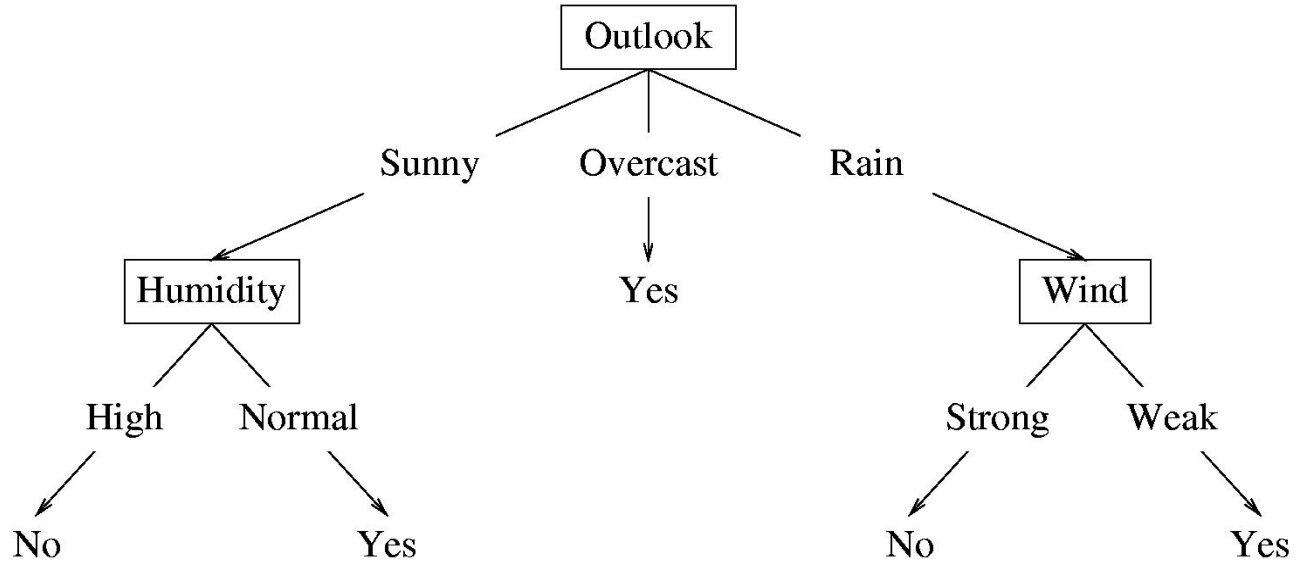
- Columns denote features X_i
- Rows denote labeled instances $\langle \boldsymbol{x}_i, y_i \rangle$
- Class label denotes whether a tennis game was played

$\langle \boldsymbol{x}_i, y_i \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision Tree

- A possible decision tree for the data:

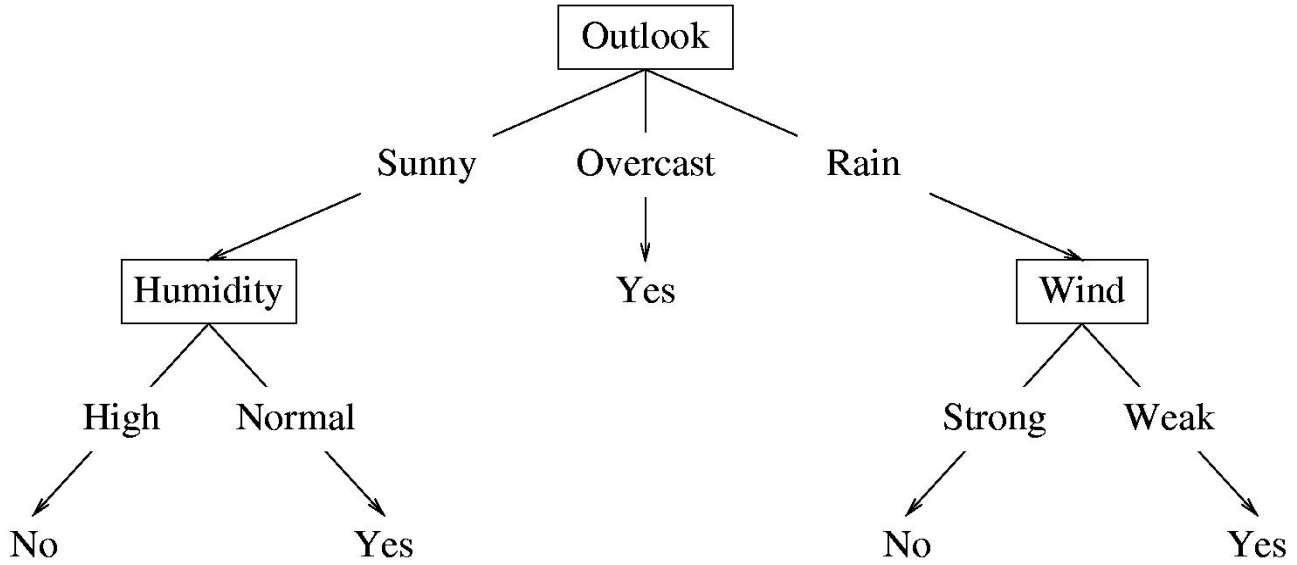


- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y \mid \boldsymbol{x} \in \text{leaf})$)

Based on slide by Tom Mitchell

Decision Tree

- A possible decision tree for the data:

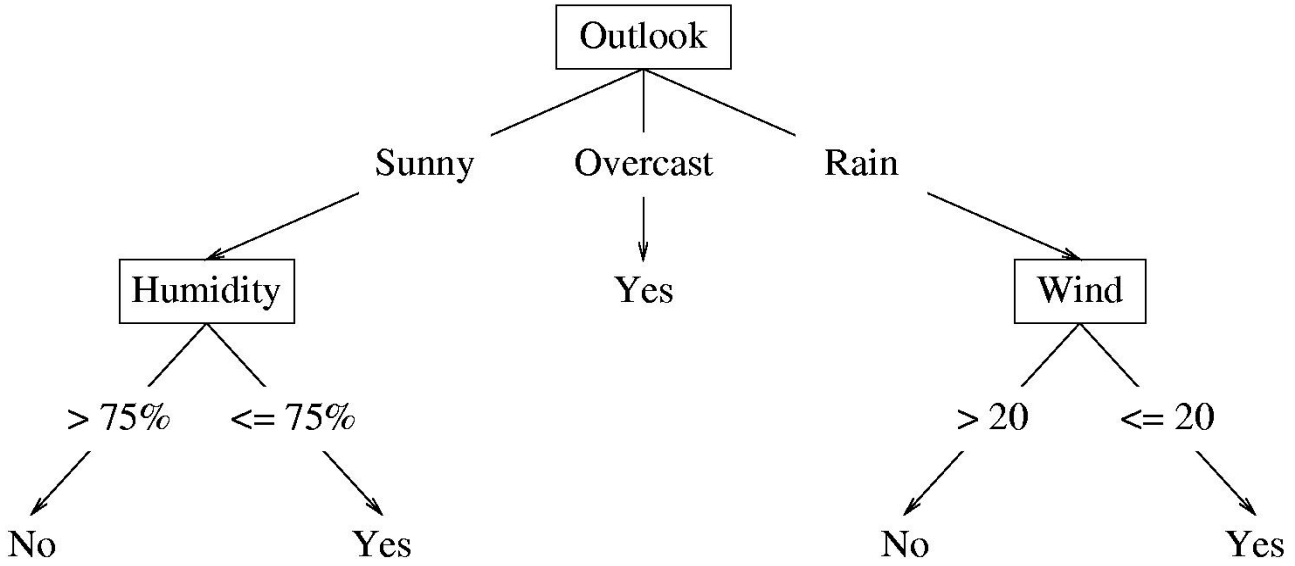


- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Based on slide by Tom Mitchell

Decision Tree

- If features are continuous, internal nodes can test the value of a feature against a threshold



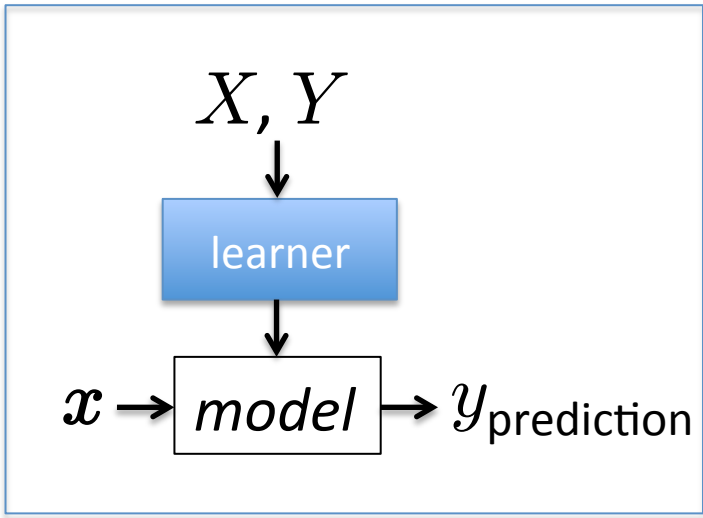
Stages of (Batch) Machine Learning

Given: labeled training data $X, Y = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n}$

- Assumes each $\boldsymbol{x}_i \sim \mathcal{D}(\mathcal{X})$ with $y_i = f_{target}(\boldsymbol{x}_i)$

Train the model:

model ← *classifier*.train(X, Y)



Apply the model to new data:

- Given: new unlabeled instance $\boldsymbol{x} \sim \mathcal{D}(\mathcal{X})$

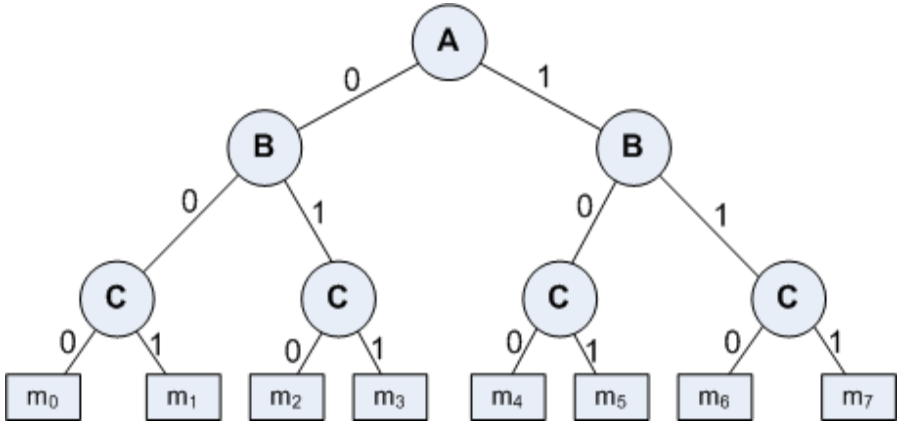
$y_{\text{prediction}}$ ← *model*.predict($\boldsymbol{x}$)

Basic Concept

- A Decision Tree is an important data structure known to solve many classification problems

Example: Binary Decision Tree

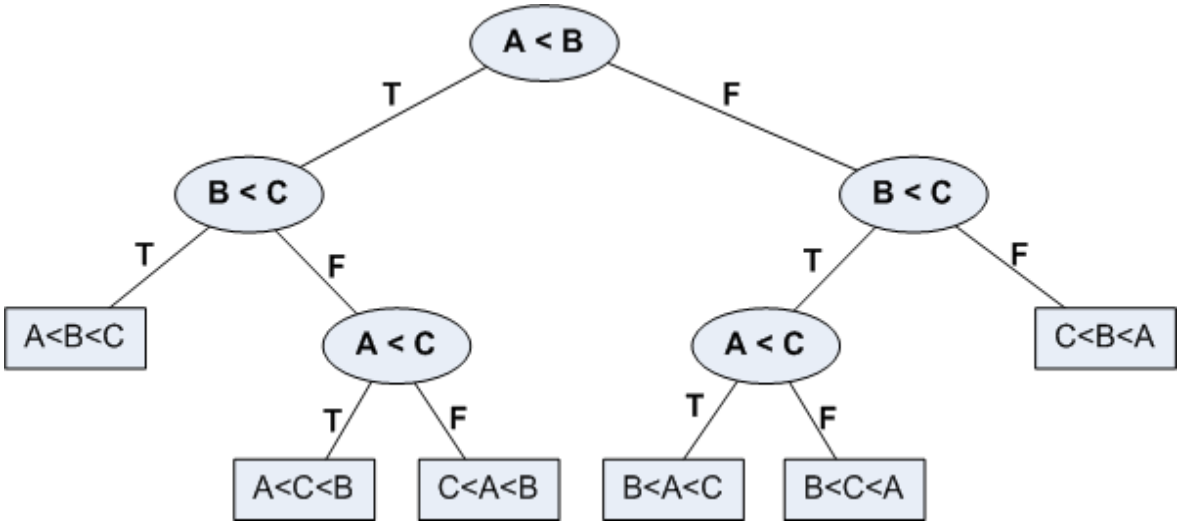
A	*B*	*C*	*f*
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



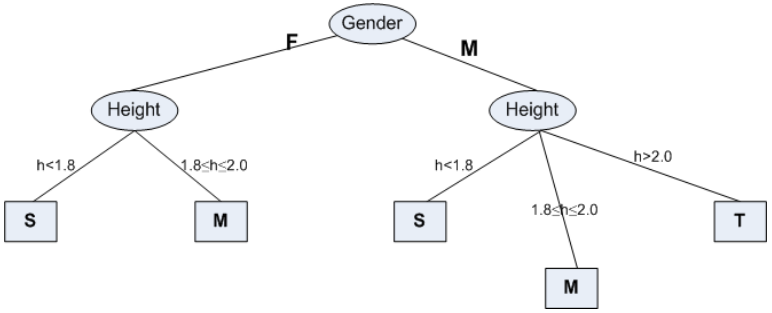
Basic Concept

- In the last Example, we have considered a decsion tree where values of any attribute if binary only. Decision tree is also possible where attributes are of continuous data type

Example: Decision Tree with numeric data



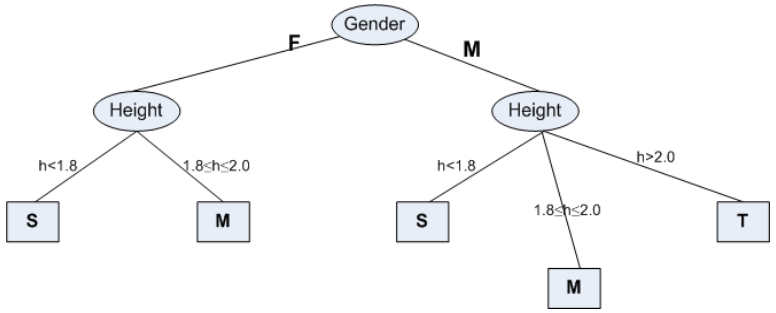
Some Characteristics



- Decision tree may be *n*-ary, $n \geq 2$.
- There is a special node called root node.
- All nodes drawn with circle (ellipse) are called internal nodes.
- All nodes drawn with rectangle boxes are called terminal nodes or leaf nodes.
- Edges of a node represent the outcome for a value of the node.
- In a path, a node with same label is never repeated.
- Decision tree is not unique, as different ordering of internal nodes can give different decision tree.

Decision Tree and Classification Task

- Decision tree helps us to classify data.
  - Internal nodes are some attribute
  - Edges are the values of attributes
  - External nodes are the outcome of classification
- Such a classification is, in fact, made by posing questions starting from the root node to each terminal node.



Decision Tree and Classification Task

Example 9.3 : Vertebrate Classification

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	**Mammal**
Python	Cold	scales	no	no	no	no	yes	**Reptile**
Salmon	Cold	scales	no	yes	no	no	no	**Fish**
Whale	Warm	hair	yes	yes	no	no	no	**Mammal**
Frog	Cold	none	no	semi	no	yes	yes	**Amphibian**
Komodo	Cold	scales	no	no	no	yes	no	**Reptile**
Bat	Warm	hair	yes	no	yes	yes	yes	**Mammal**
Pigeon	Warm	feathers	no	no	yes	yes	no	**Bird**
Cat	Warm	fur	yes	no	no	yes	no	**Mammal**
Leopard	Cold	scales	yes	yes	no	no	no	**Fish**
Turtle	Cold	scales	no	semi	no	yes	no	**Reptile**
Penguin	Warm	feathers	no	semi	no	yes	no	**Bird**
Porcupine	Warm	quills	yes	no	no	yes	yes	**Mammal**
Eel	Cold	scales	no	yes	no	no	no	**Fish**
Salamander	Cold	none	no	semi	no	yes	yes	**Amphibian**

What are the class label of Dragon and Shark?

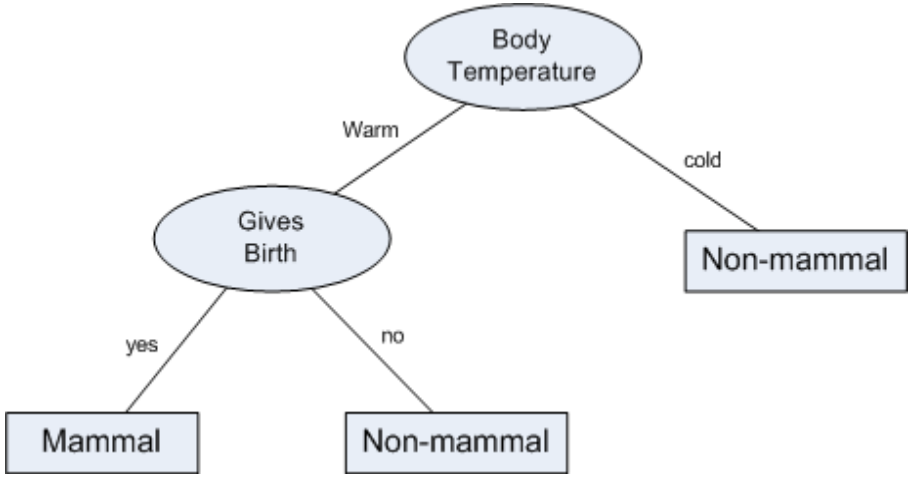
Decision Tree and Classification Task

Example 9.3 : Vertebrate Classification

- Suppose, a new species is discovered as follows.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Gila Monster	cold	scale	no	no	no	yes	yes	?

- Decision Tree that can be inducted based on the data (in Example 9.3) is as follows.



Definition of Decision Tree

Definition 9.1: **Decision Tree**

Given a database $D = \{t_1, t_2, \ldots\ldots, t_n\}$, where t_i denotes a tuple, which is defined by a set of attribute $A = \{A_1, A_2, \ldots\ldots, A_m\}$. Also, given a set of classes $C = \{c_1, c_2, \ldots\ldots, c_k\}$.

A decision tree $\boldsymbol{T}$ is a tree associated with D that has the following properties:

- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

Building Decision Tree

- In principle, there are exponentially many decision tree that can be constructed from a given database (also called training data).
  - Some of the tree may not be optimum
  - Some of them may give inaccurate result
- Two approaches are known
  - **Greedy strategy**
    - A top-down recursive divide-and-conquer
  - **Modification of greedy strategy**
    - ID3
    - C4.5
    - CART, etc.

Built Decision Tree Algorithm

- **Algorithm BuildDT**
- Input: D : Training data set
- Output: T : Decision tree

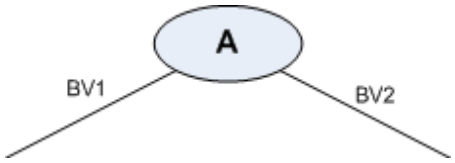
Steps

1. If all tuples in D belongs to the same class C_j
   Add a leaf node labeled as C_j
   Return *// Termination condition*
2. **Select** an attribute A_i (so that it is not selected twice in the same branch)
3. **Partition** $D = \{ D_1, D_2, \ldots, D_p\}$ based on p different values of A_i in D
4. For each $D_k \in D$
   Create a node and add an edge between D and D_k with label as the A_i's attribute value in D_k
5. For each $D_k \in D$
   BuildTD(D_k) *// Recursive call*
6. Stop

Node Splitting in **BuildDT** Algorithm

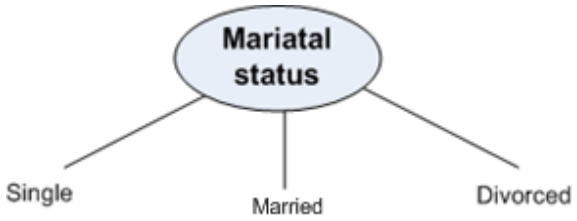
- BuildDT algorithm must provides a method for expressing an attribute test condition and corresponding outcome for different attribute type

- **Case: Binary attribute**
  - This is the simplest case of node splitting
  - The test condition for a binary attribute generates only two outcomes



Node Splitting in **BuildDT** Algorithm

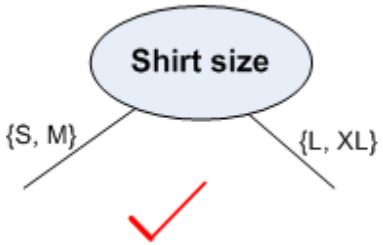
- **Case: Nominal attribute (no order among its values)**
  - Since a nominal attribute can have many values, its test condition can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: Outcome depends on the number of distinct values for the corresponding attribute

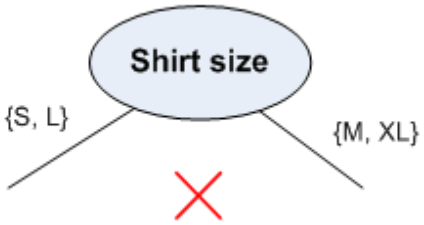


  - Binary splitting by grouping attribute values

Node Splitting in **BuildDT** Algorithm

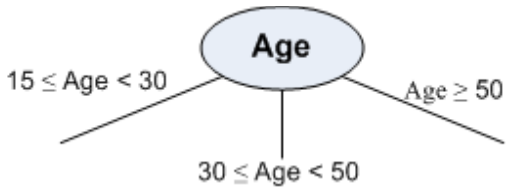
- **Case: Ordinal attribute (order among its values)**
  - It also can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: It is same as in the case of nominal attribute
  - Binary splitting attribute values should be grouped maintaining the order property of the attribute values





Node Splitting in **BuildDT** Algorithm

- **Case: Numerical attribute**
  - For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
    - Binary outcome: $A > v$ or $A \leq v$
      - In this case, decision tree induction must consider all possible split positions
    - Range query : $v_i \leq A < v_{i+1}$ for $i = 1, 2, \ldots, q$ (if q number of ranges are chosen)
      - Here, q should be decided a priori



  - For a numeric attribute, decision tree induction is a combinatorial optimization problem

Illustration : **BuildDT** Algorithm

Example 9.4: Illustration of BuildDT Algorithm

– Consider a training data set as shown.

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Attributes:

Gender = {Male(M), Female (F)} // Binary attribute
Height = {1.5, …, 2.5} // Continuous attribute

Class = {Short (S), Medium (M), Tall (T)}

Given a person, we are to test in which class s/he belongs

Illustration : **BuildDT** Algorithm

- To build a decision tree, we can select an attribute in two different orderings: <Gender, Height> or <Height, Gender>
- Further, for each ordering, we can choose different ways of splitting
- Different instances are shown in the following.
- **Approach 1 : <Gender, Height>**

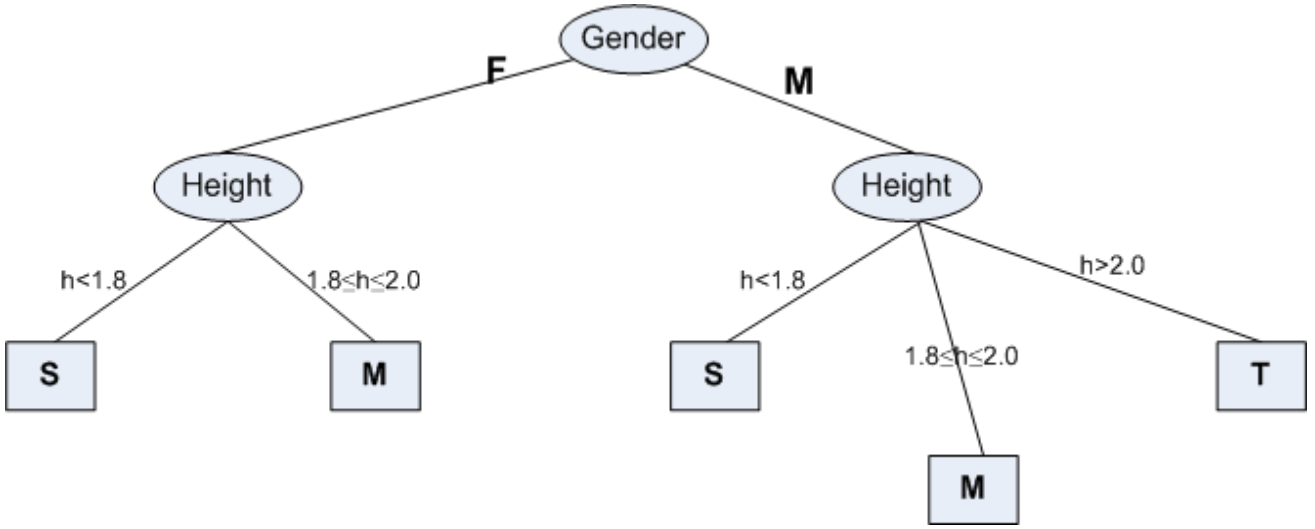


Illustration : **BuildDT** Algorithm

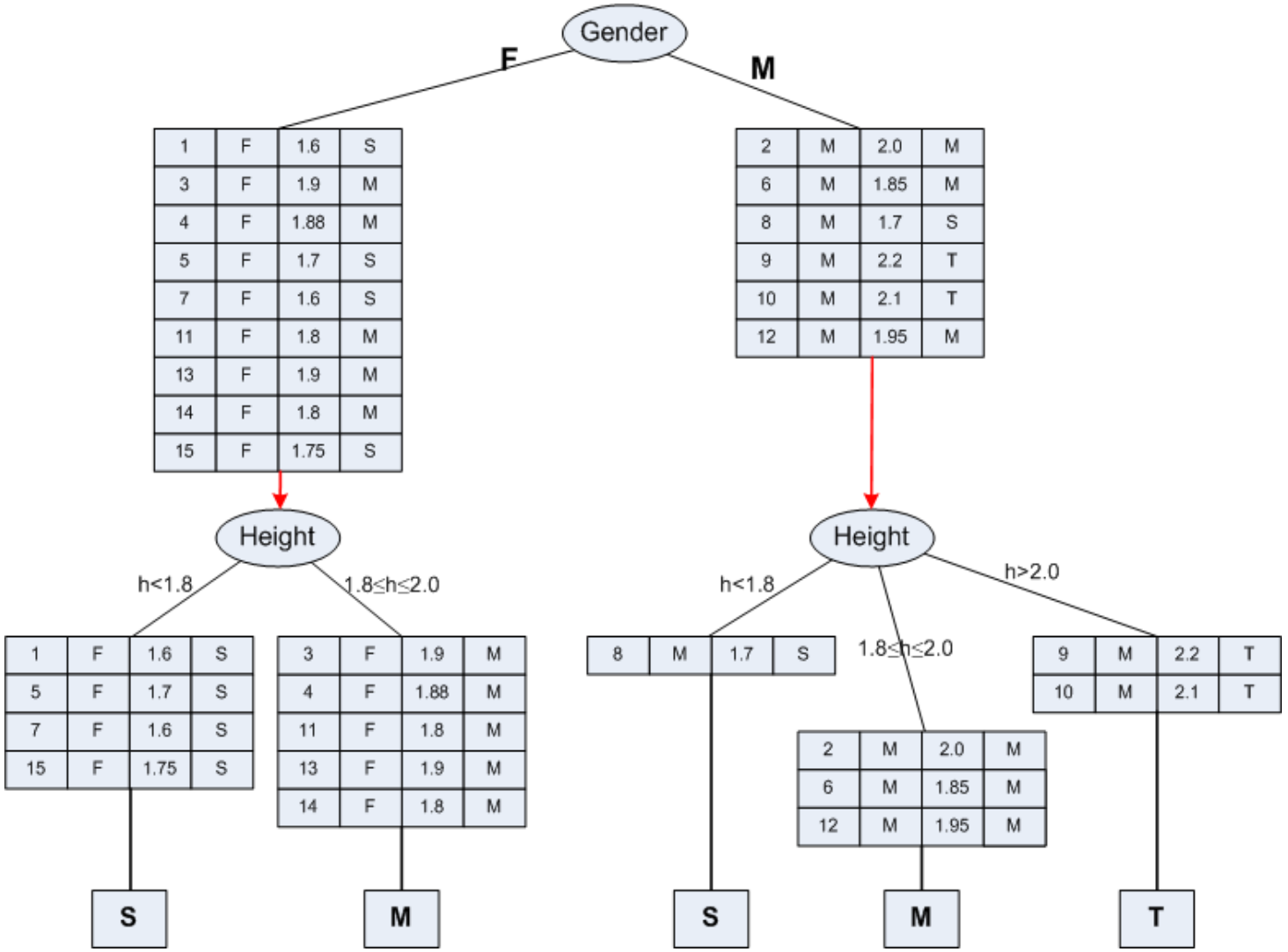
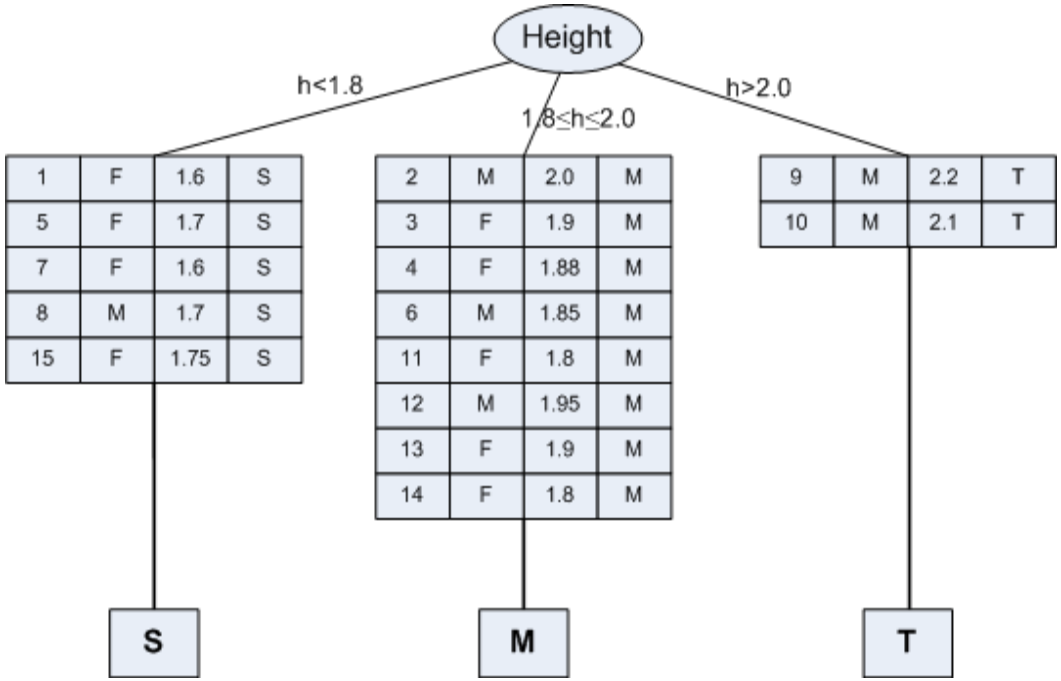
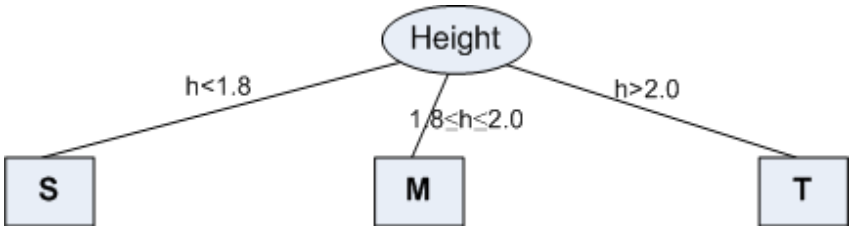


Illustration : **BuildDT** Algorithm

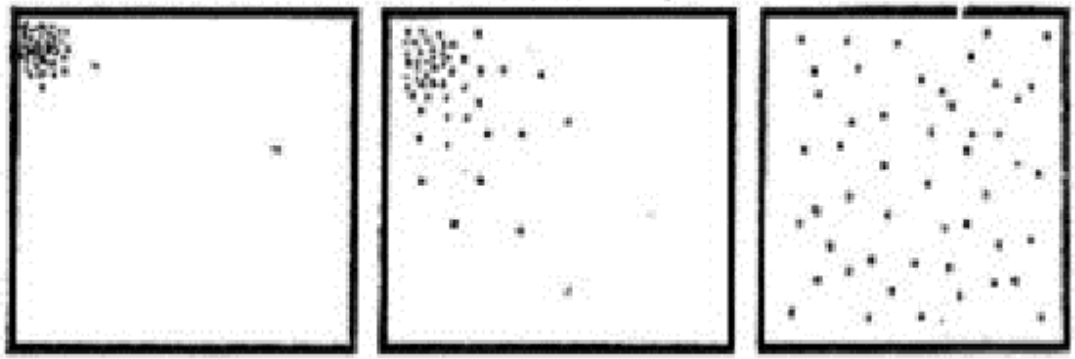
- **Approach 2 : <Height, Gender>**





Concept of Entropy

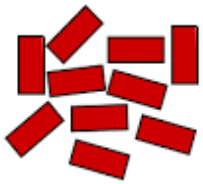
Concept of Entropy



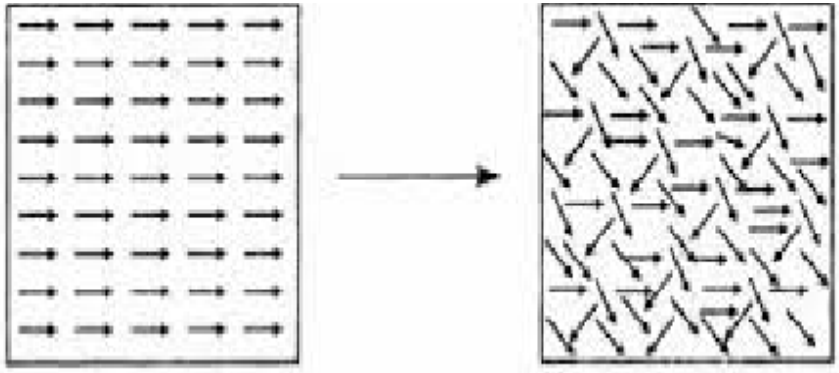
If a point represents a gas molecule, then which system has the more entropy?

How to measure? $\Delta S = \frac{\Delta Q}{T}$?

More **ordered**
less **entropy**
entropy



Less ordered
higher



More organized or
ordered (less **probable**)

Less organized or
disordered (**more** probable)

Entropy in Information Theory

- The entropy concept in information theory first time coined by Claude Shannon (1850).
- The first time it was used to measure the "information content" in messages.
- According to his concept of entropy, presently entropy is widely being used as a way of representing messages for efficient transmission by Telecommunication Systems.

Entropy of a Training Set

- If there are k classes $c_1, c_2 \ldots\ldots, c_k$ and p_i for $i = 1\ to\ k$ denotes the number of occurrences of classes c_i divided by the total number of instances (i.e., the frequency of occurrence of c_i) in the training set, then entropy of the training set is denoted by

$$E = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Here, E is measured in "bits" of information.

Note:

- The above formula should be summed over the non-empty classes only, that is, classes for which $p_i \neq 0$
- E is always a positive quantity
- E takes it's minimum value (zero) if and only if all the instances have the same class (i.e., the training set with only **one** non-empty class, for which the probability 1).
- Entropy takes its maximum value when the instances are equally distributed among k possible classes. In this case, the maximum value of E is $log_2 k$.

Note:

- The entropy of a training set implies the number of yes/no questions, on the average, needed to determine an unknown test to be classified.
- It is very crucial to decide the series of questions about the value of a set of attribute, which collectively determine the classification. Sometimes it may take one question, sometimes many more.
- Decision tree induction helps us to ask such a series of questions. In other words, we can utilize entropy concept to build a better decision tree.

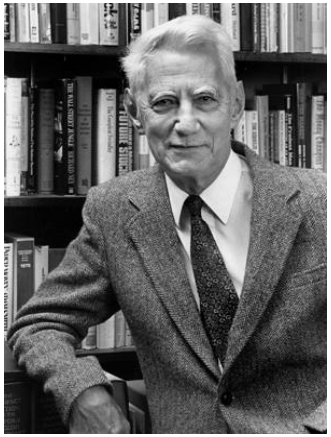
How entropy can be used to build a decision tree is our next topic of discussion.

Entropy

- Quantifies the amount of uncertainty associated with a specific probability distribution
- The higher the entropy, the less confident we are in the outcome
- Definition

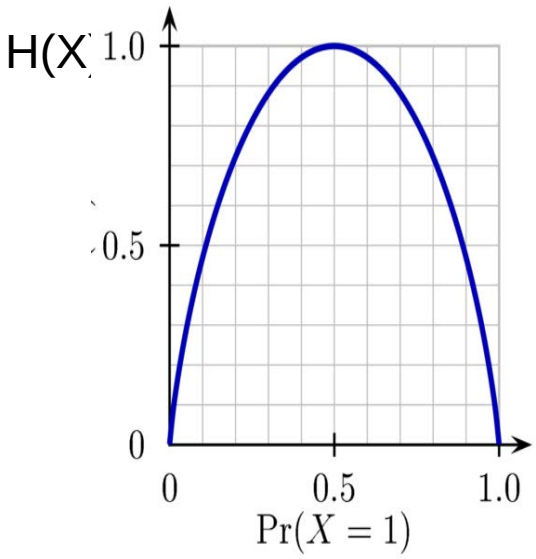
$$H(X) = \sum_x p(X = x) \log_2 \frac{1}{p(X = x)}$$

$$H(X) = -\sum_x p(X = x) \log_2 p(X = x)$$



Claude Shannon (1916 – 2001), most of the work was done in Bell labs

Entropy



- Definition

$$H(X) = -\sum_{x} p(X = x) \log_2 p(X = x)$$

- So, if P(X=1) = 1 then

$$H(X) = -p(x=1)\log_2 p(X=1) - p(x=0)\log_2 p(X=0)$$
$$= -1\log 1 - 0\log 0 = 0$$

- If P(X=1) = .5 then

$$H(X) = -p(x=1)\log_2 p(X=1) - p(x=0)\log_2 p(X=0)$$
$$= -.5\log_2 .5 - .5\log_2 .5 = -\log_2 .5 = 1$$