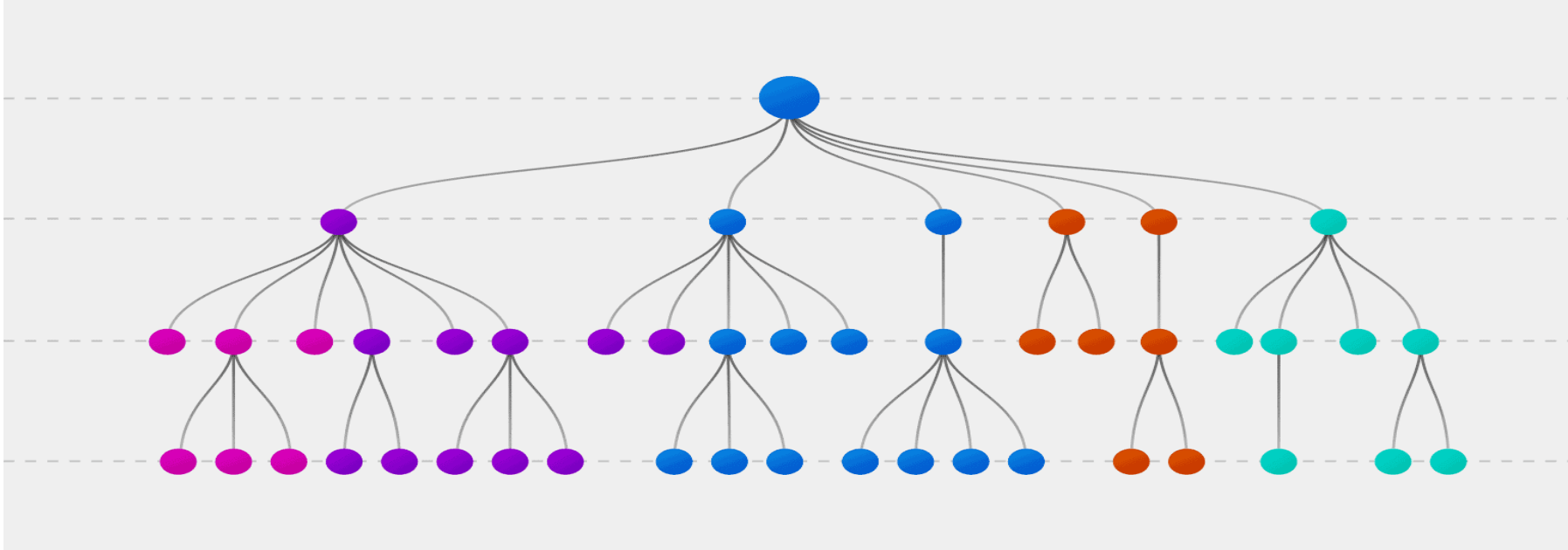
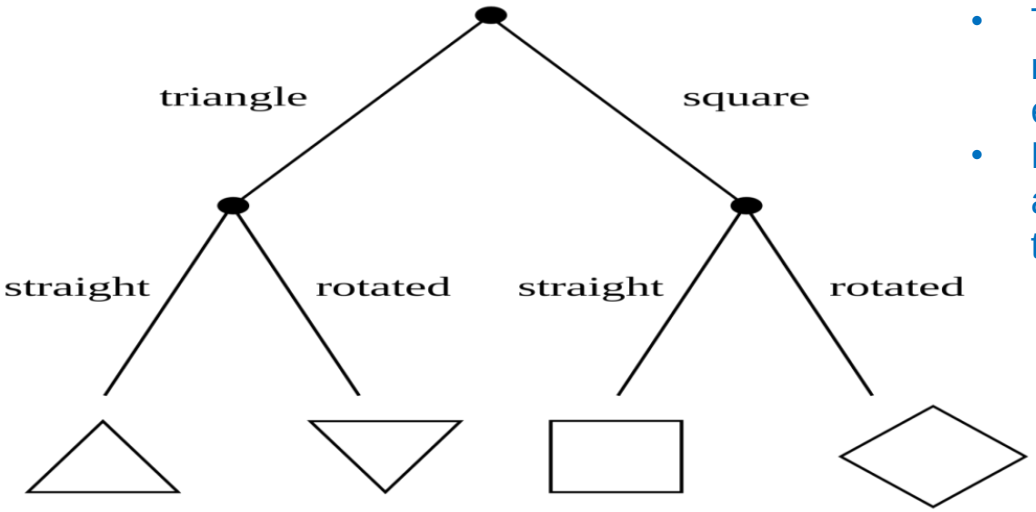
# Decision Trees



**Dr. Dinesh Kumar Vishwakarma**
Professor,
Department of Information Technology,
Delhi Technological University, Delhi-110042
**dinesh@dtu.ac.in**
http://www.dtu.ac.in/Web/Departments/InformationTechnology/faculty/dkvishwakarma.php

# Introduction

- A decision tree is a support tool with a tree-like structure that models probable outcomes, cost of resources, utilities, and possible consequences.
- It provides a way to present algorithms with conditional control statements.
- It include branches that represent decision-making steps that can lead to a favorable result.




- The flowchart structure includes internal nodes that represent tests or attributes at each stage.
- Every branch stands for an outcome for the attributes, while the path from the leaf to the root represents rules for classification.

# Applications

- Assessing prospective growth opportunities
- Using demographic data to find prospective clients
- Marketing
- Retention of Customers
- Diagnosis of Diseases and Ailments
- Detection of Frauds

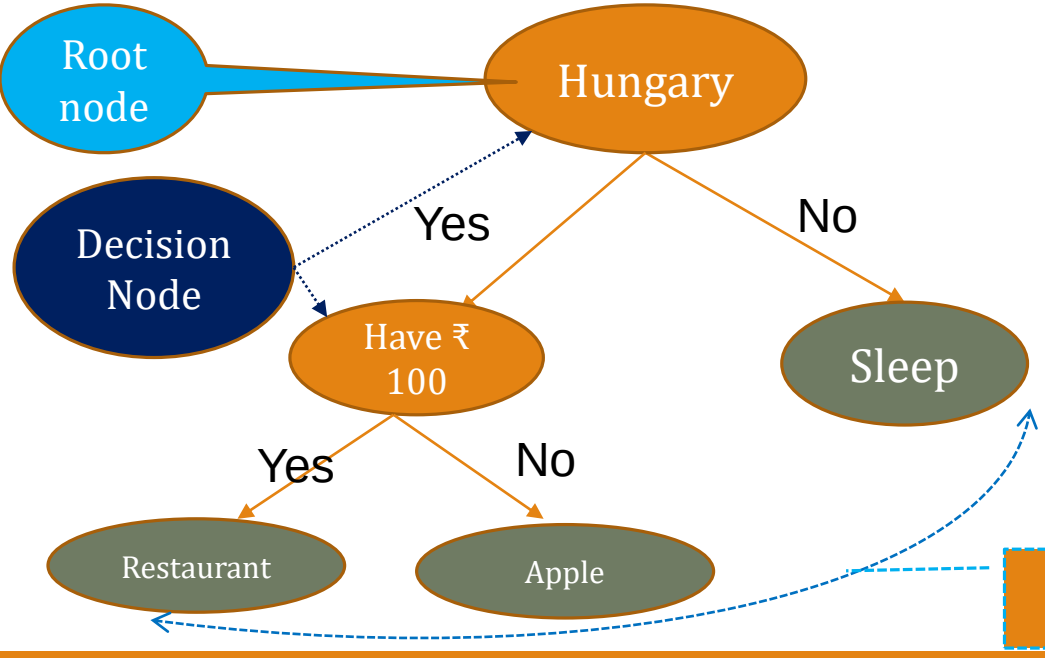
# Advantages

- **Easy to read and interpret**
  - The o/p are easy to read and interpret without requiring statistical knowledge
- **Easy to prepare**
  - take less effort for data preparation
- **Less data cleaning required**
  - There is less data cleaning required once the variables have been created. In cases of missing values and outliers have less significance on the decision tree's data.

# Decision Tree: DT

- Graphical representation of all possible solutions.
- Decisions are based on some conditions.
- Decision made can be easily explained.

E.g. **Call Centre**

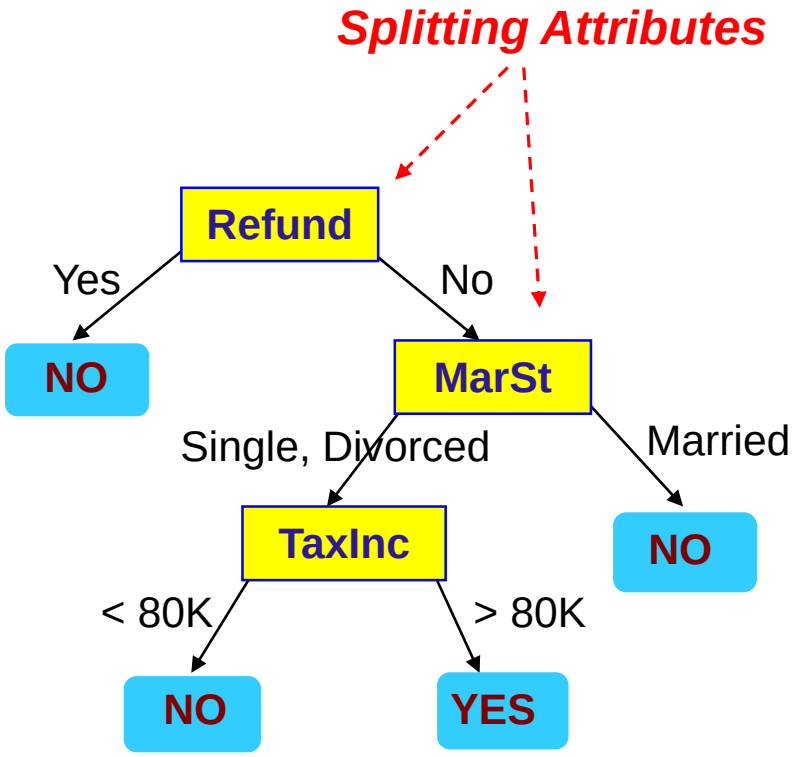



A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes)

# Decision Tree: DT

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

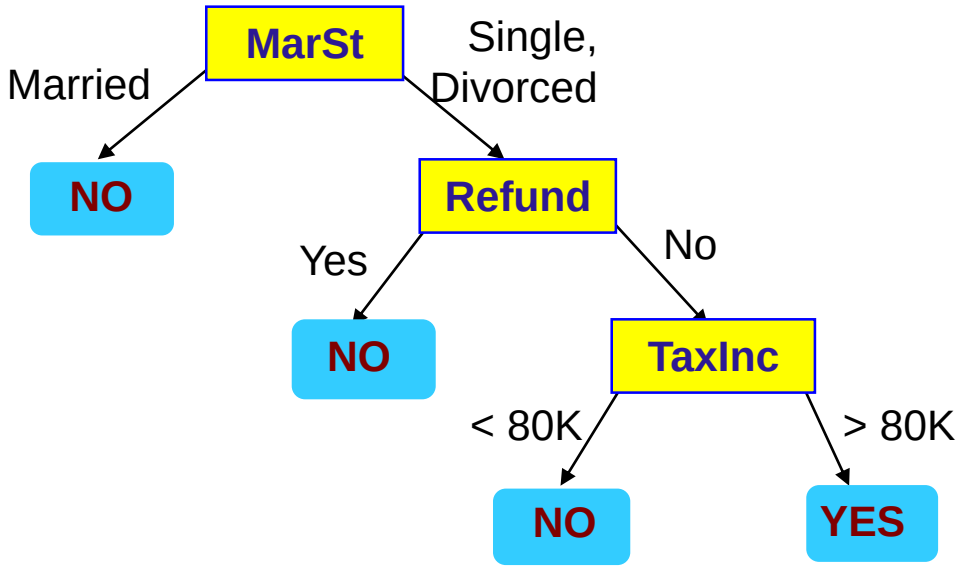
Training Data




Model: Decision Tree

# Another Example of Decision Tree

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




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

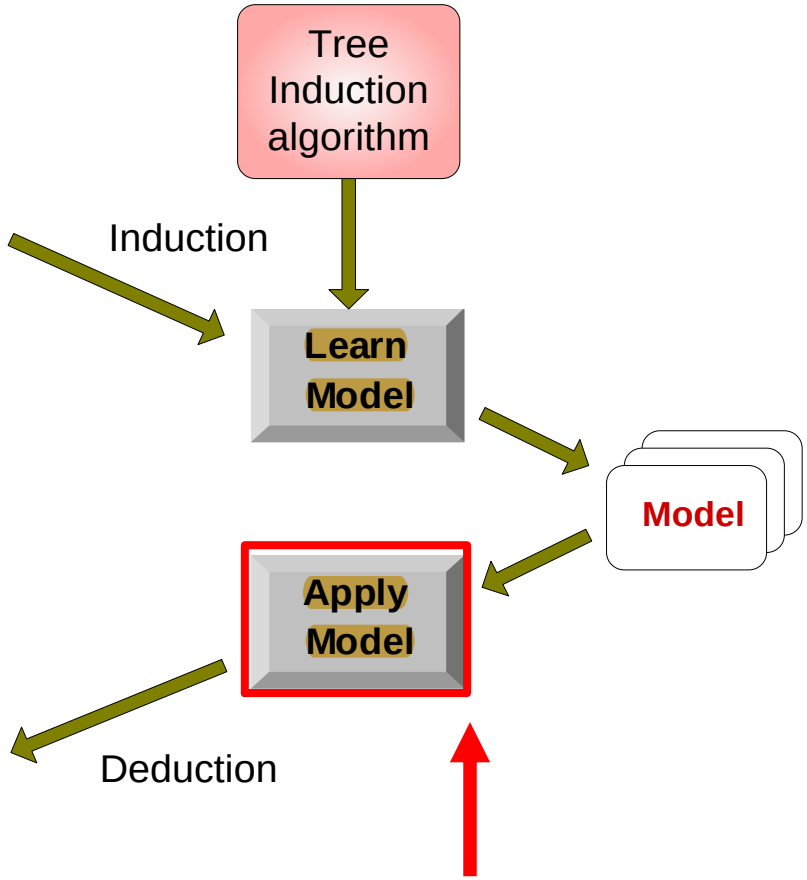
# Decision Tree Classification Task

| 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 |

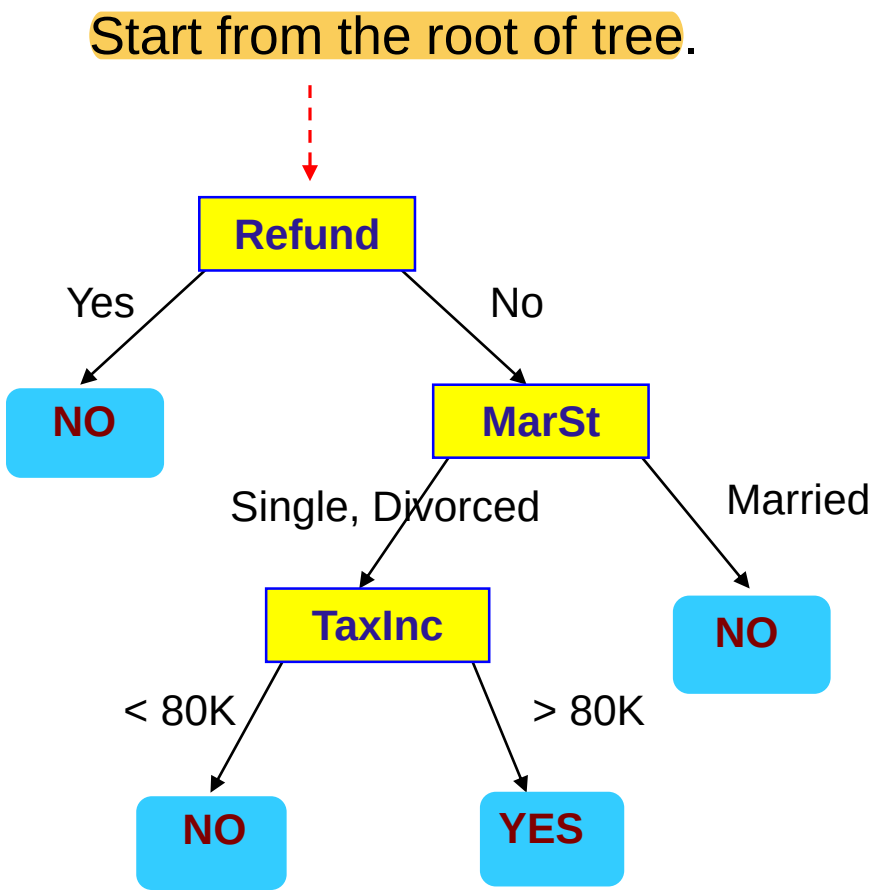
Training Set

| 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



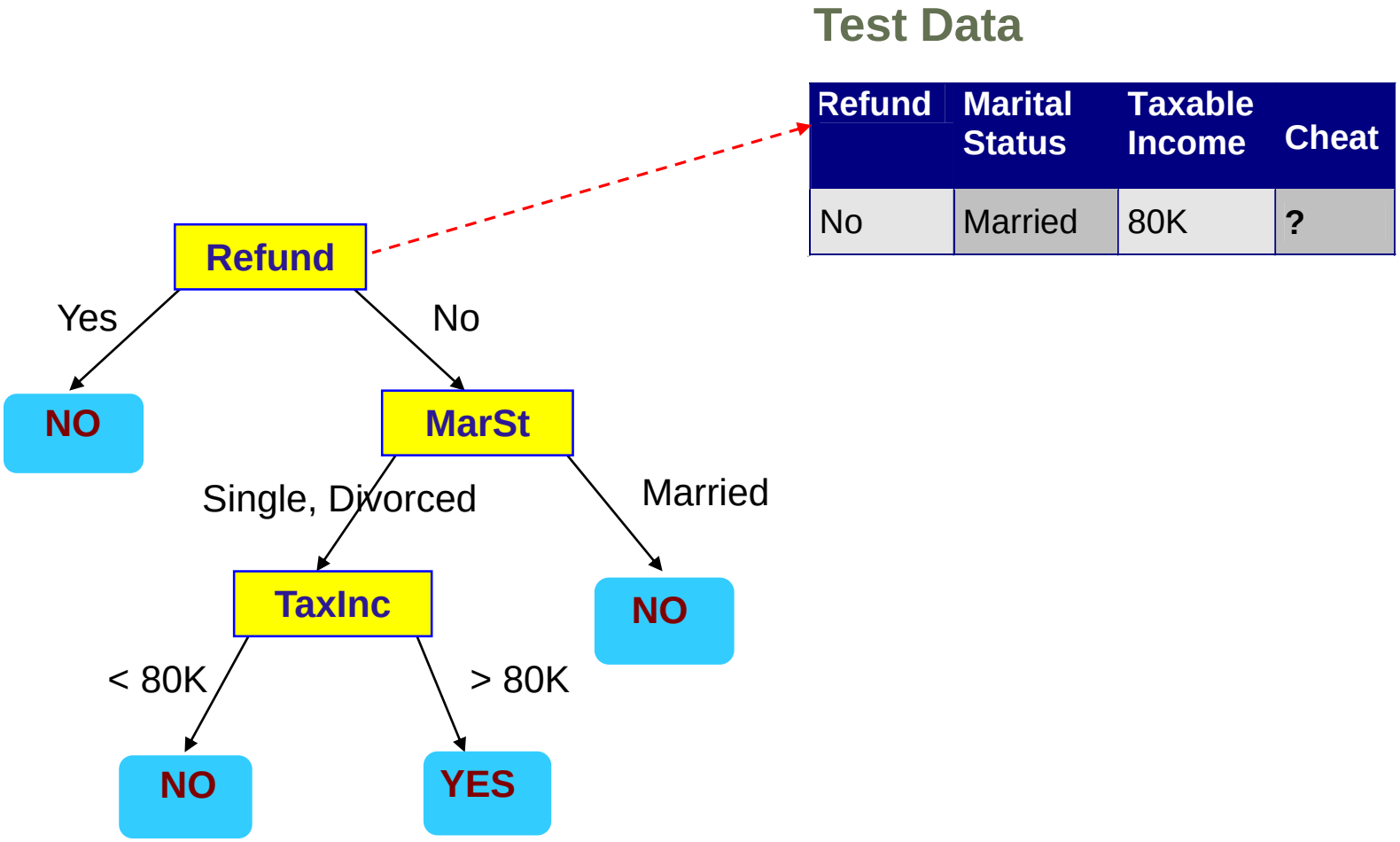
# Apply Model to Test Data



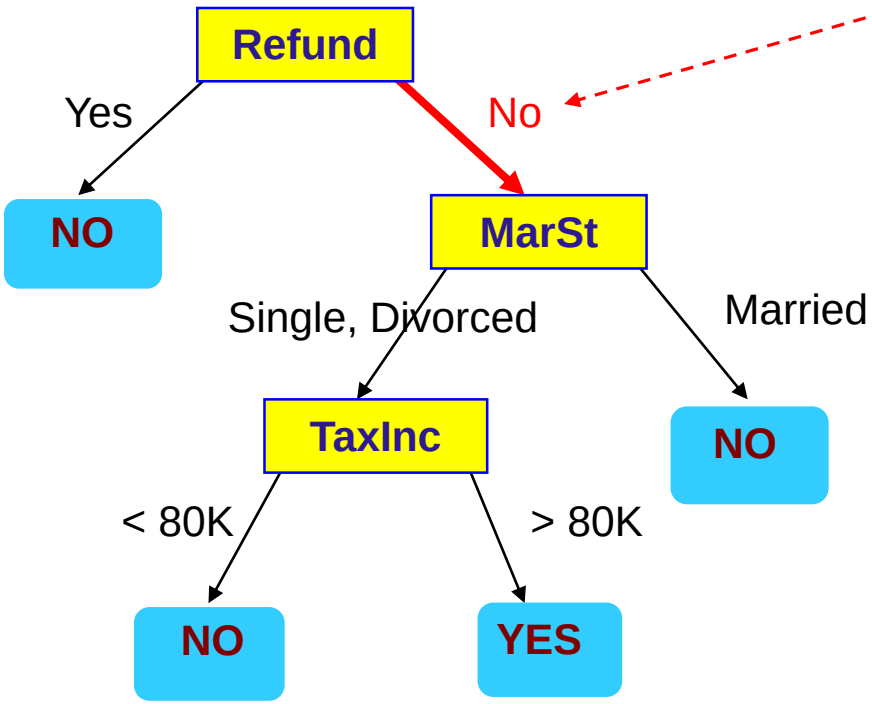

**Test Data**

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

# Apply Model to Test Data



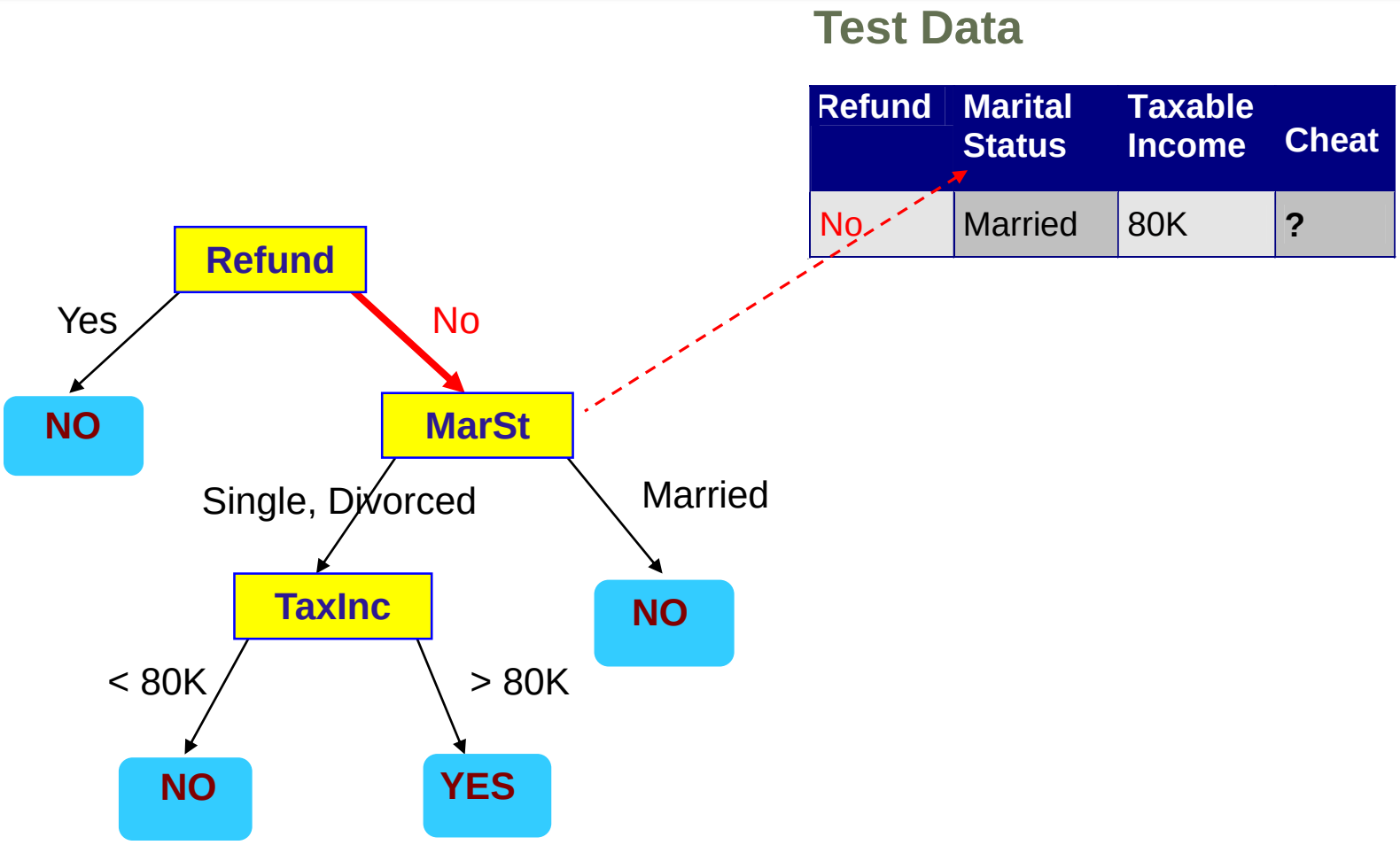
# Apply Model to Test Data



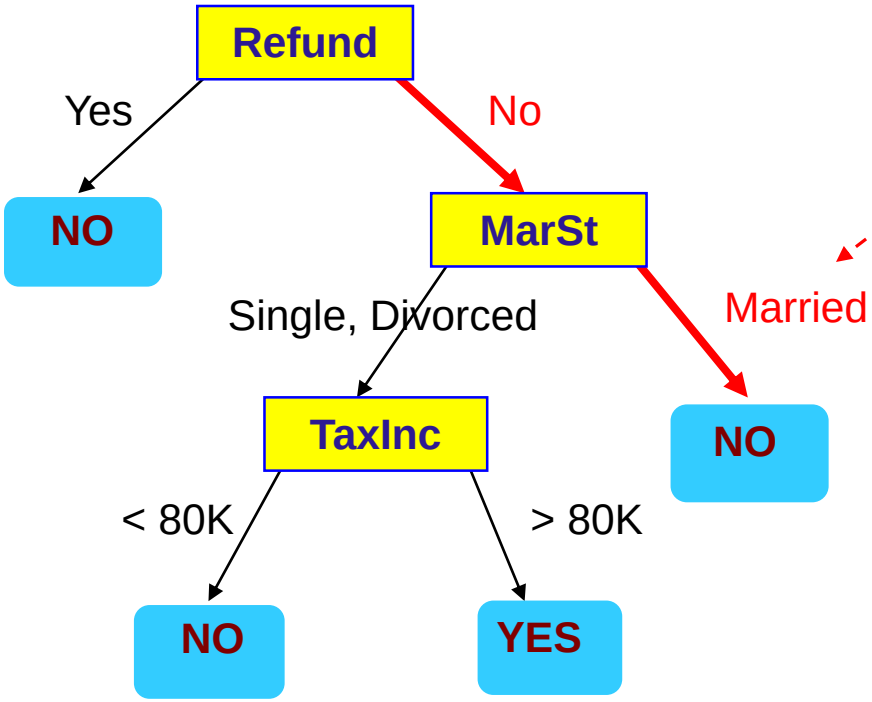

Test Data

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

# Apply Model to Test Data



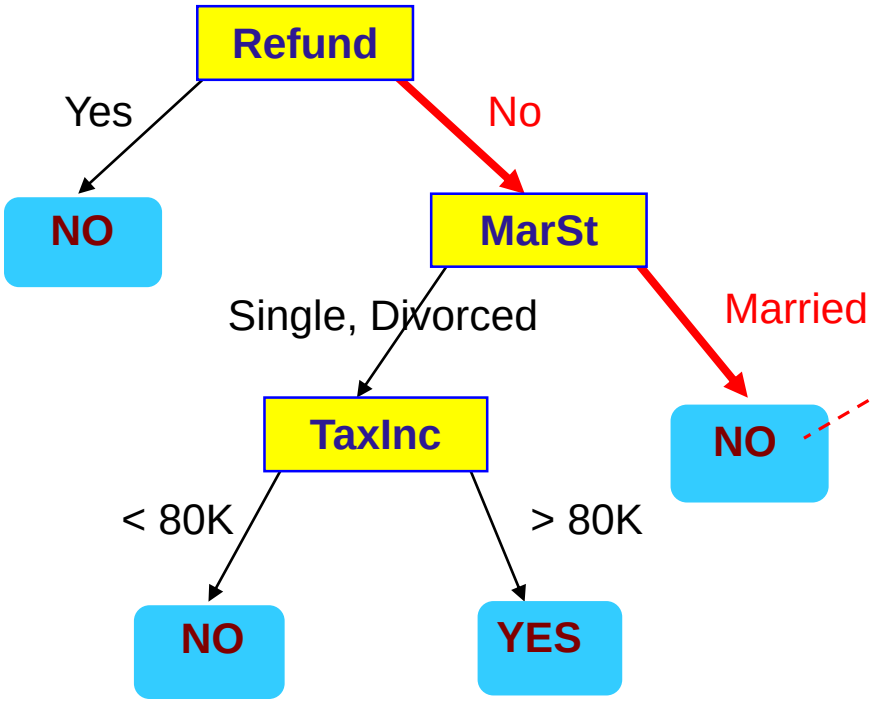
# Apply Model to Test Data




Test Data

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

# Apply Model to Test Data




**Test Data**

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

Assign Cheat to “No”

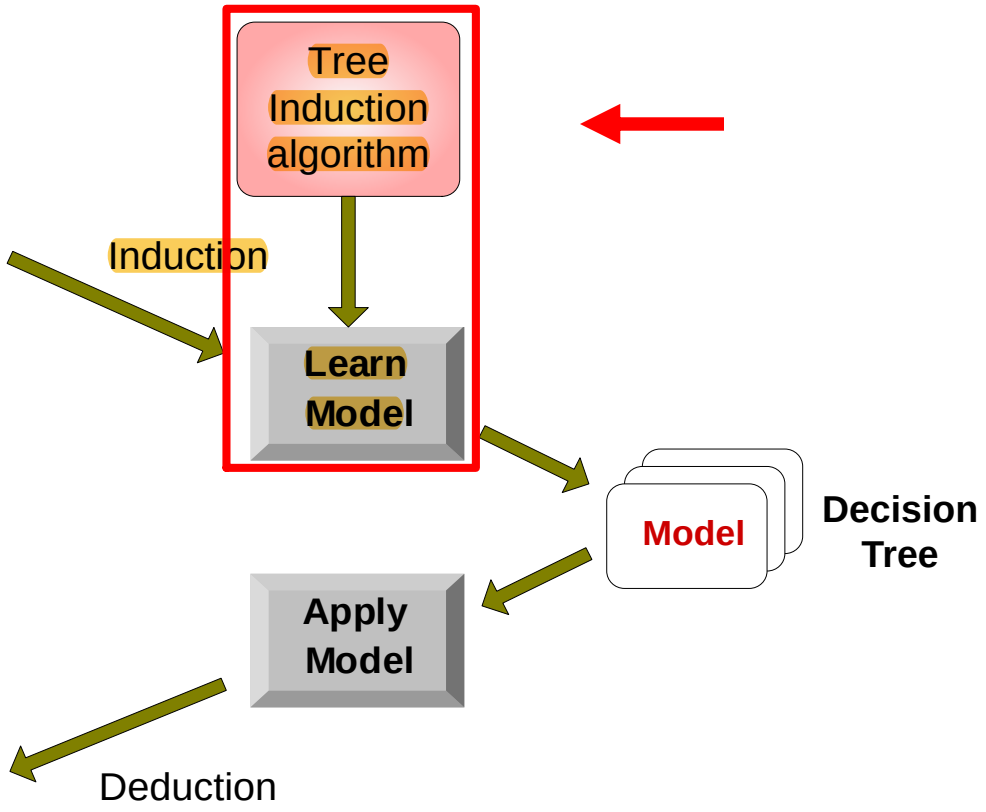
# Decision Tree Classification Task

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

Training Set

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

Test Set



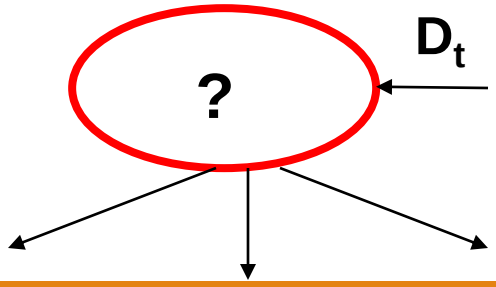
# Decision Tree Induction

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

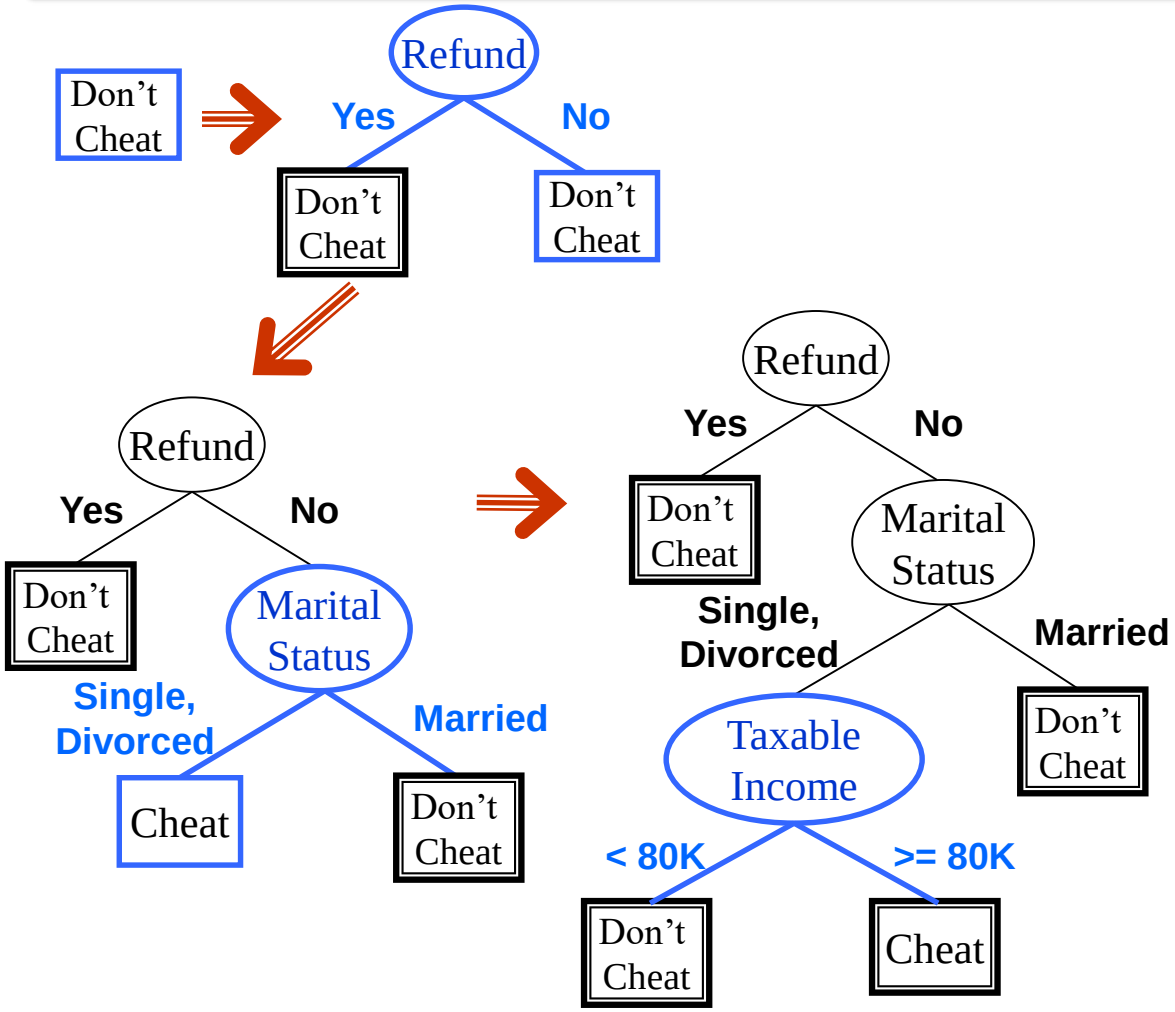
# Hunt's Algorithm

- 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* | 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.
- 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

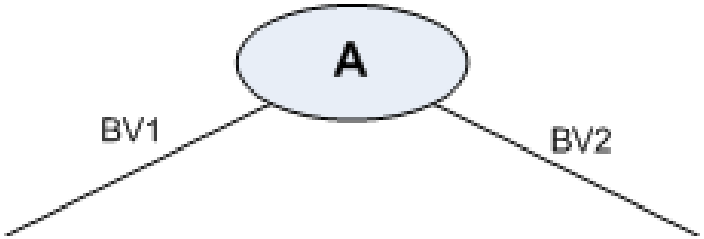
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

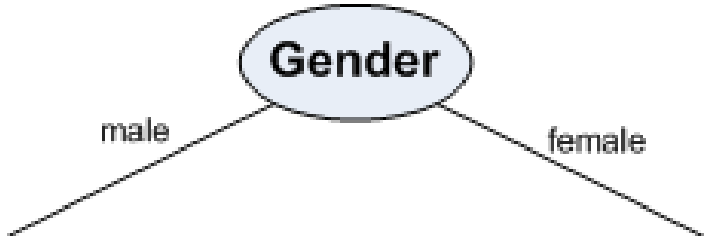
# How to Specify Test Condition?

- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

# Splitting Based on Binary Attributes

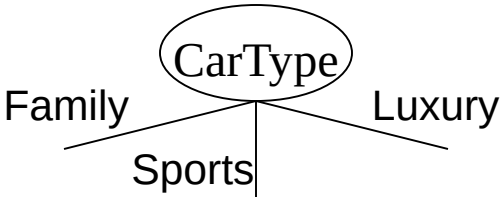
- BuildDT algorithm must provides a method for expressing an attribute test condition and corresponding outcome for different attribute type
- **Case: Binary attribute**
  - This is the simplest case of node splitting
  - The test condition for a binary attribute generates only two outcomes



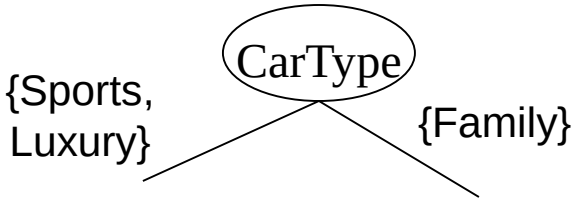



# Splitting Based on Nominal Attributes

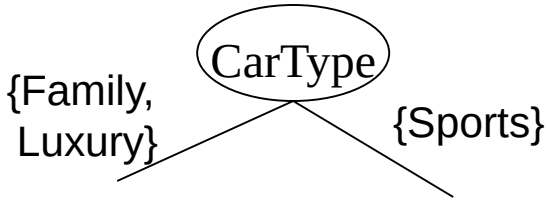
- Multi-way split: Use as many partitions as distinct values.




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

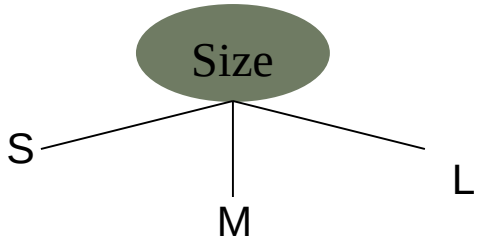



OR

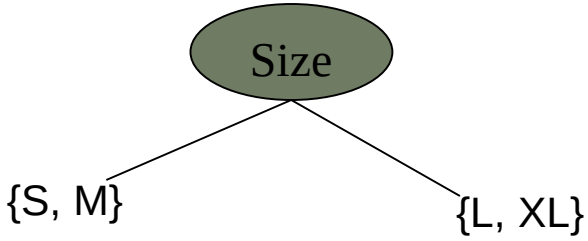


# Splitting Based on Ordinal Attributes

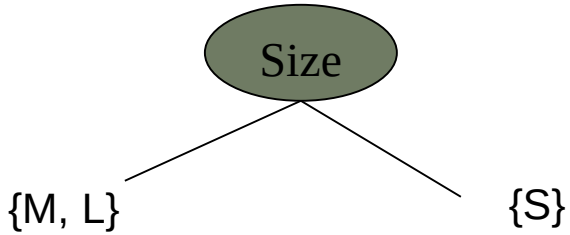
- Multi-way split: Use as many partitions as distinct values.
  - Small (S), Medium (M),
  - Large (L), Extra Large (XL)




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




OR

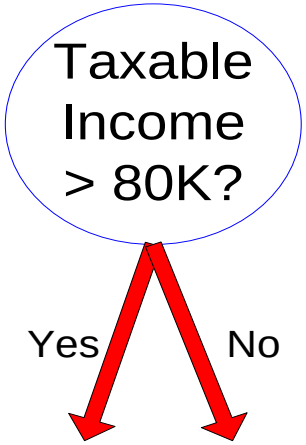


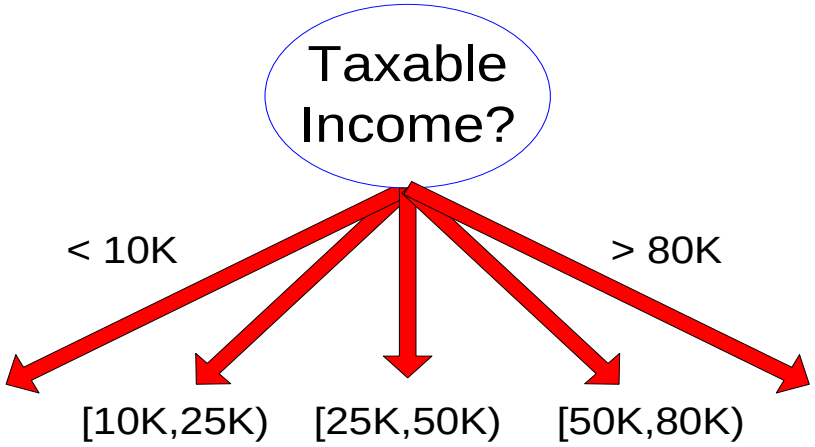

- What about this split?

# 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

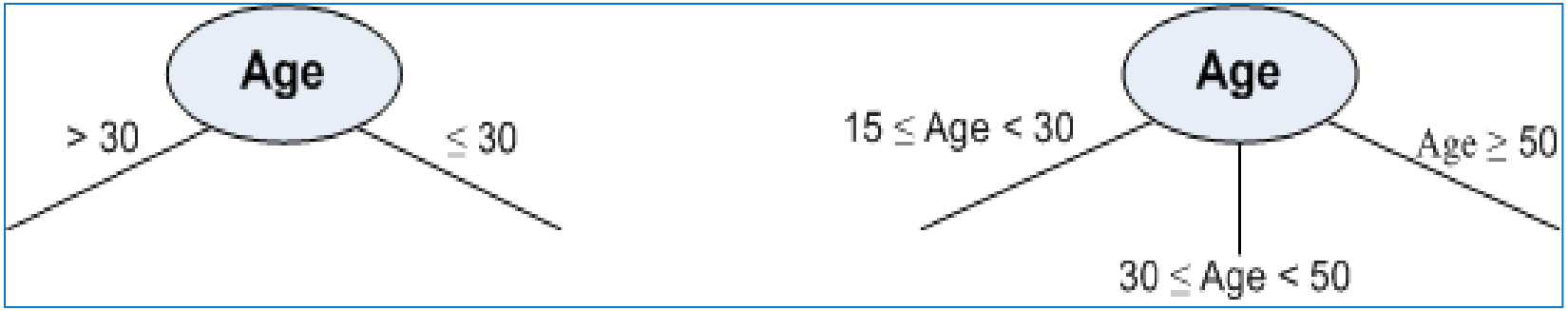
# Splitting Based on Continuous Attributes…




(i) Binary split

(ii) Multi-way split



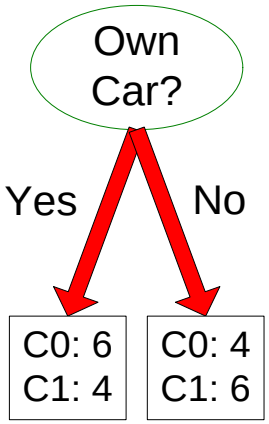
# Tree Induction

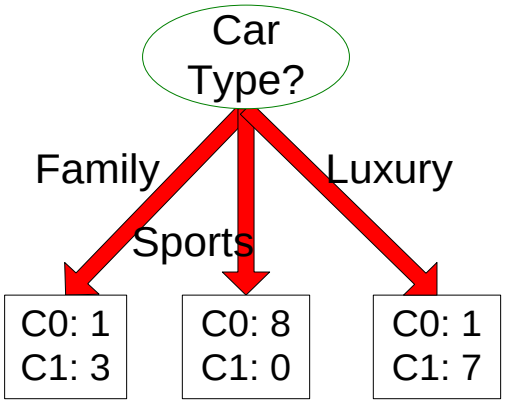
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

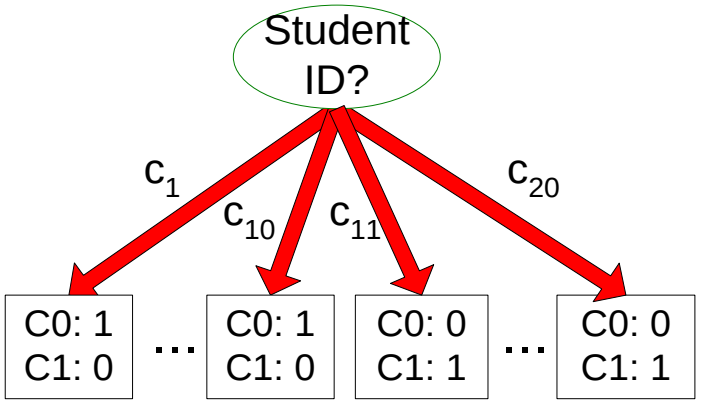
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

# 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

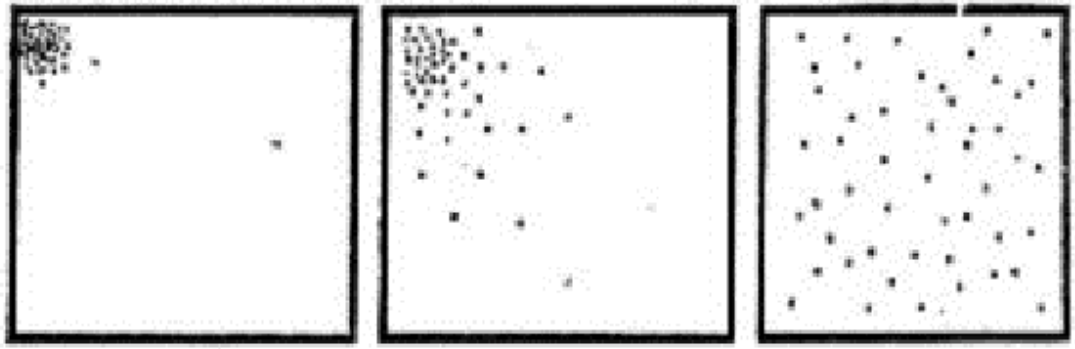
- Greedy approach:
  - Nodes with homogeneous class distribution are preferred.
- Need a measure of node impurity:

| 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

- There are three popular way to measure impurity is:
  - Information Gain
  - Gain Ratio
  - Gini Index

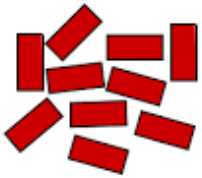
# Concept of Entropy



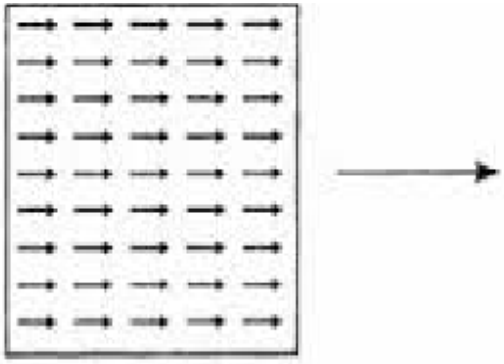
If a point represents a gas molecule, then which system has the more entropy?

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

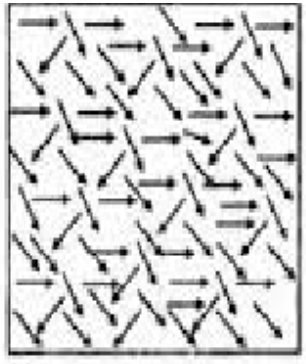
More **ordered**
less **entropy**



**Less** ordered
**higher** entropy



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



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

# An Open Challenge!

| Roll No. | Assignment | Project | Mid-Sem | End-Sem |
|---|---|---|---|---|
| 12BT3FP06 | 89 | 99 | 56 | 91 |
| 10IM30013 | 95 | 98 | 55 | 93 |
| 12CE31005 | 98 | 96 | 58 | 97 |
| 12EC35015 | 93 | 95 | 54 | 99 |
| 12GG2005 | 90 | 91 | 53 | 98 |
| 12MI33006 | 91 | 93 | 57 | 97 |
| 13AG36001 | 96 | 94 | 58 | 95 |
| 13EE10009 | 92 | 96 | 56 | 96 |
| 13MA20012 | 88 | 98 | 59 | 96 |
| 14CS30017 | 94 | 90 | 60 | 94 |
| 14ME10067 | 90 | 92 | 58 | 95 |
| 14MT10038 | 99 | 89 | 55 | 93 |

| Roll No. | Assignment | Project | Mid-Sem | End-Sem |
|---|---|---|---|---|
| 12BT3FP06 | 19 | 59 | 16 | 71 |
| 10IM30013 | 37 | 38 | 25 | 83 |
| 12CE31005 | 38 | 16 | 48 | 97 |
| 12EC35015 | 23 | 95 | 54 | 19 |
| 12GG2005 | 40 | 71 | 43 | 28 |
| 12MI33006 | 61 | 93 | 47 | 97 |
| 13AG36001 | 26 | 64 | 48 | 75 |
| 13EE10009 | 92 | 46 | 56 | 56 |
| 13MA20012 | 88 | 58 | 59 | 66 |
| 14CS30017 | 74 | 20 | 60 | 44 |
| 14ME10067 | 50 | 42 | 38 | 35 |
| 14MT10038 | 29 | 69 | 25 | 33 |

Two sheets showing the tabulation of marks obtained in a course are shown.

Which tabulation of marks shows the “good” performance of the class?
How you can measure the same?

# Entropy and its Meaning

- Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.
- At a later stage, with the growth of Information Technology, entropy becomes an important concept in Information Theory.
- To deal with the classification job, entropy is an important concept, which is considered as
  - an information-theoretic measure of the "uncertainty" contained in a training data
    - due to the presence of more than one classes.

# 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.

# Measure of Information Content

**Example 1**

a) Guessing a birthday of your classmate

It is with uncertainty $\sim \frac{1}{365}$

Whereas guessing the day of his/her birthday is $\frac{1}{7}$.

This uncertainty, we may say varies between 0 to 1, both inclusive.

b) As another example, a question related to event with eventuality (or impossibility) will be answered with 0 or 1 uncertainty.

- Does sun rises in the East? (answer is with 0 uncertainty)
- Will mother give birth to male baby? (answer is with ½ uncertainty)
- Is there a planet like earth in the galaxy? (answer is with an extreme uncertainty)

# Entropy Calculations

- Consider a dataset $\mathcal{D}$, which has various samples as $s^{(1)}, s^{(2)}, \ldots\ldots\ldots s^{(N)}$.
- The classes of dataset is $y_1, y_2$ (Binary Classification). It can be taken k classes too. (in general)
- Selecting a random sample from the dataset, which may belongs to class $y_q$.
- The probability of this selection is $P_q = \frac{freq(y_q, \mathcal{D})}{|\mathcal{D}|}$
- Where $freq(y_q, \mathcal{D})$ is the number of patterns in $\mathcal{D}$ that belongs to $y_q$, $|\mathcal{D}|$ is the total number of samples in $\mathcal{D}$.

# Entropy Calculations…

- The expected information required to classify a pattern in $\mathcal{D}$ is $Info(\mathcal{D}) = -\sum_{q=1}^{k} p_q log_2^{(p_q)}$. M=2 for binary classification.
- A log of base 2 is used because information is encoded into bits.
- $Info(\mathcal{D})$ is average amount of information required to identify the class label of a sample in $\mathcal{D}$.
- $Info(\mathcal{D})$ can also be expressed as entropy as $E(\mathcal{D}) = -\sum_{q=1}^{k} p_q log_2^{(p_q)}$.
- Here, $E(\mathcal{D})$ 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 $log_2 k$.

# Entropy of a Training Set

**Consider a dataset of** OTPH as shown in the following table with total 24 instances in it.

| Age | Eye sight | Astigmatic | Use Type | Class |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 3 |
| 1 | 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 1 | 3 |
| 1 | 1 | 2 | 2 | 1 |
| 1 | 2 | 1 | 1 | 3 |
| 1 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 3 |
| 1 | 2 | 2 | 2 | 1 |
| 2 | 1 | 1 | 1 | 3 |
| 2 | 1 | 1 | 2 | 2 |
| 2 | 1 | 2 | 1 | 3 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 1 | 3 |
| 2 | 2 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 3 |
| 2 | 2 | 2 | 2 | 3 |
| 3 | 1 | 1 | 1 | 3 |
| 3 | 1 | 1 | 2 | 3 |
| 3 | 1 | 2 | 1 | 3 |
| 3 | 1 | 2 | 2 | 3 |
| 3 | 2 | 1 | 1 | 3 |
| 3 | 2 | 1 | 2 | 3 |
| 3 | 2 | 2 | 1 | 3 |
| 3 | 2 | 2 | 2 | 3 |

A coded forms for all values of attributes are used to avoid the cluttering in the table.

# Entropy of a training set…

Specification of the attributes are as follows.

| Age | Eye Sight | Astigmatic | Use Type |
|---|---|---|---|
| 1: Young | 1: Myopia | 1: No | 1: Frequent |
| 2: Middle-aged | 2: Hypermetropia | 2: Yes | 2: Less |
| 3: Old | | | |

**Class: 1: Contact Lens 2:Normal glass 3: Nothing**

In the OPTH database, there are 3 classes and 4 instances with class 1, 5 instances with class 2 and 15 instances with class 3. Hence, entropy $E$ of the database is:

$$E = -\frac{4}{24}\log_2\frac{4}{24} - \frac{5}{24}\log_2\frac{5}{24} - \frac{15}{24}\log_2\frac{15}{24} = 1.3261$$

# Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.
- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.
- Different algorithms have been proposed to take a good control over
  1. Choosing the best attribute to be splitted, and
  2. Splitting criteria
- Several algorithms have been proposed for the above tasks. In this lecture, we shall limit our discussions into three important of them
  - **ID3**
  - **C 4.5**
  - **CART**

# Algorithm ID3

# ID3: Decision Tree Induction Algorithms

- Quinlan [1986] introduced the ID3, a popular short form of **Iterative Dichotomizer 3** for decision trees from a set of training data.

- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.

- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

# Algorithm ID3

- In ID3, entropy is used to measure how informative a node is.
  - It is observed that splitting on any attribute has the property that average entropy of the resulting training subsets will be less than or equal to that of the previous training set.
- ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
  - The attribute with the largest value of information gain is chosen as the splitting attribute and
  - it partitions into a number of smaller training sets based on the distinct values of attribute under split.

# Defining Information Gain

- We consider the following symbols and terminologies to define information gain, which is denoted as α.
- $D \equiv$ denotes the training set at any instant
- $|D| \equiv$ denotes the size of the training set $D$
- $E(D) \equiv$ denotes the entropy of the training set $D$
- The entropy of the training set $D$

$$E(D) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

  - where the training set $D$ has $c_1, c_2, \ldots, c_k$, the $k$ number of distinct classes and
  - $p_i$, $0 < p_i \leq 1$ is the probability that an arbitrary tuple in $D$ belongs to class $c_i$ (i *= 1, 2, … , k).*
  - $p_i$ can be calculated as

$$p_i = \frac{|C_{i,D}|}{|D|}$$

  - where $C_{i,D}$ is the set of tuples of class $c_i$ in $D$.

# Defining Information Gain…

- Suppose, we want to partition $D$ on some attribute $A$ having $m$ distinct values $\{a_1, a_2, \dots, a_m\}$.
- Attribute $A$ can be considered to split $D$ into $m$ partitions $\{D_1, D_2, \dots, D_m\}$, where $D_j$ $(j = 1,2, \dots ,m)$ contains those tuples in $D$ that have outcome $a_j$ of $A$.
- The **weighted entropy** denoted as $E_A(D)$ for all partitions of $D$ with respect to $A$ is given by:
  - $E_A(D) = \sum_{j=1}^{m} \frac{|D_j|}{|D|} E(D_j)$
- Here, the term $\frac{|D_j|}{|D|}$ denotes the weight of the $j$-th training set.
- More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from $D$ based on the splitting of $A$.

# Defining Information Gain…

- Our objective is to take *A* on splitting to produce an exact classification (also called pure), that is, all tuples belong to one class.
- However, it is quite likely that the partitions is impure, that is, they contain tuples from two or more classes.
- In that sense, $E_A(D)$ is a measure of impurities (or purity). A lesser value of $E_A(D)$ implying more power the partitions are.
- **Information gain, $\alpha(A, D)$** of the training set *D* splitting on the attribute *A* is given by
  - $\alpha(A, D) = E(D) - E_A(D)$
- In other words, $\alpha(A, D)$ gives us an estimation how much would be gained by splitting on *A*. The attribute *A* with the highest value of $\alpha$ should be chosen as the splitting attribute for *D*.

# Compute Information Gain

| *Tid* | Refund | Marital Status | Taxable 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 |
| 10 | ? | Single | 90K | Yes |

Missing value

**Before Splitting:**

$$E(\mathcal{D}) = -\frac{3}{10} log\left(\frac{3}{10}\right) - \frac{7}{10} log\left(\frac{7}{10}\right) = 0.8813$$

| | Class = Yes | Class = No |
|---|---|---|
| Refund=Yes | 0 | 3 |
| Refund=No | 2 | 4 |
| Refund=? | 1 | 0 |

**Split on Refund (A):**

$$E(Refund = Yes) = 0$$

$$E(Refund = No) = \frac{2}{6} log\left(\frac{2}{6}\right) - \frac{4}{6} log\left(\frac{4}{6}\right) = 0.9183$$

$$E_A(\mathcal{D}) = \frac{3}{10} \times 0 + \frac{6}{10} \times 0.9183 = .5509$$

$$\alpha(A, \mathcal{D}) = .8813 - 0.5509 = .33032$$

# Example

- **Information gain on splitting OPTH**
- Training set: $D_1$ (Age = 1)

| Age (x1) | Eye-sight (x2) | Astigmatism (x3) | Use type (x4) | Class (y) |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 3 |
| 1 | 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 1 | 3 |
| 1 | 1 | 2 | 2 | 1 |
| 1 | 2 | 1 | 1 | 3 |
| 1 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 3 |
| 1 | 2 | 2 | 2 | 1 |

$$E(D_1) = -\frac{2}{8} log_2 \quad ( \quad \frac{2}{8} \quad ) - \frac{2}{8} log_2 (\frac{2}{8}) - \frac{4}{8} log_2 (\frac{4}{8}) = \mathbf{1.5}$$

$$E_{Age}(D_1) = \quad \frac{8}{24} \quad \times \quad 1.5 \quad = \mathbf{0.5000}$$

# Example…

Training set: $D_2$(Age = 2)

| Age | Eye-sight | Astigmatism | Use type | Class |
|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 3 |
| 2 | 1 | 1 | 2 | 2 |
| 2 | 1 | 2 | 1 | 3 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 1 | 3 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 1 | 3 |
| 2 | 2 | 2 | 2 | 3 |

$$E(D_2) = -\frac{1}{8}log_2(\frac{1}{8}) - \frac{2}{8}log_2(\frac{2}{8}) - \frac{5}{8}log_2(\frac{5}{8}) = \mathbf{1.2988}$$

$$E_{Age}(D_2) = \frac{8}{24} \times 1.2988 = \mathbf{0.4329}$$

# Example…

Training set: $D_3$(Age = 3)

| Age | Eye-sight | Astigmatism | Use type | Class |
|---|---|---|---|---|
| 3 | 1 | 1 | 1 | 3 |
| 3 | 1 | 1 | 2 | 3 |
| 3 | 1 | 2 | 1 | 3 |
| 3 | 1 | 2 | 2 | 1 |
| 3 | 2 | 1 | 1 | 3 |
| 3 | 2 | 1 | 2 | 2 |
| 3 | 2 | 2 | 1 | 3 |
| 3 | 2 | 2 | 2 | 3 |

$$E(D_3) = -\frac{1}{8}\log\left(\frac{1}{8}\right) - \frac{1}{8}\log\left(\frac{1}{8}\right) - \frac{6}{8}\log\left(\frac{6}{8}\right) = 1.0613$$

$$E_{Age}(D_3) = \frac{8}{24} \times 1.0613 = 0.3504$$

$\alpha\ (Age, D)$ = 1.3261 – (0.5000 + 0.4329 + 0.3504) = **0.0394**

# Information Gains for Different Attributes

- In the same way, we can calculate the information gains, when splitting the OPTH database on Eye-sight, Astigmatic and Use Type. The results are summarized below.
- Splitting attribute: Age

$$\alpha(Age, OPTH) = 0.0394$$

- Splitting attribute: Eye-sight

$$\alpha(Eye - sight, OPTH) = 0.0395$$

- Splitting attribute: Astigmatic

$$\alpha(\text{Astigmatic}, OPTH) = 0.3770$$

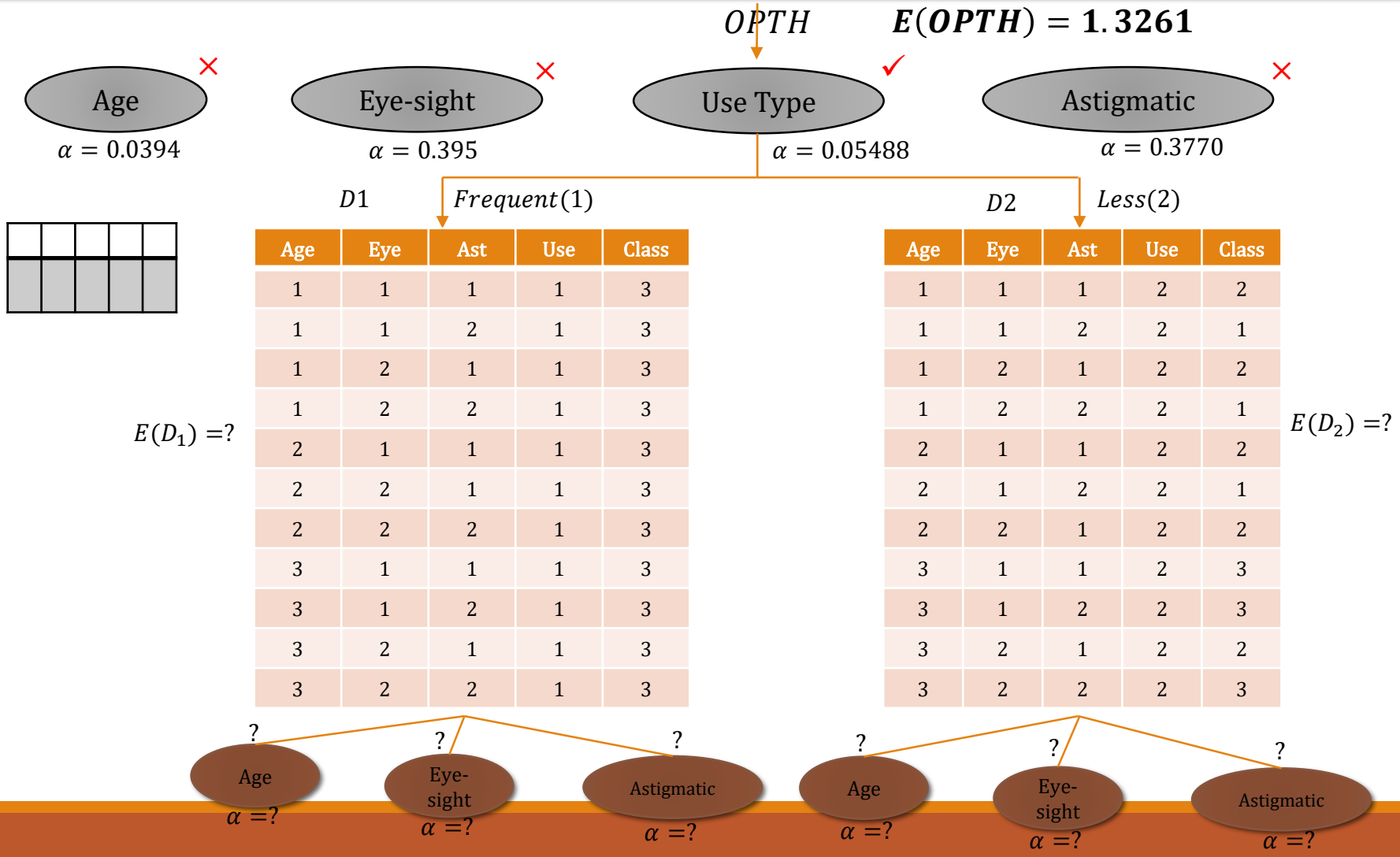
- Splitting attribute: Use Type

$$\alpha(\text{Use Type}, OPTH) = 0.5488$$

# Decision Tree Induction : ID3 Way

- The ID3 strategy of attribute selection is to choose to split on the attribute that gives the greatest reduction in the weighted average entropy
  - The one that maximizes the value of information gain
- In the example with OPTH database, the larger values of information gain is $\alpha(\text{Use Type}, OPTH) = 0.5488$
  - Hence, the attribute should be chosen for splitting is "Use Type".
- The process of splitting on nodes is repeated for each branch of the evolving decision tree, and the final tree, which would look like is shown in the following slide and calculation is left for practice.

# Decision Tree Induction : ID3 Way




| Age | Eye | Ast | Use | Class |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 3 |
| 1 | 1 | 2 | 1 | 3 |
| 1 | 2 | 1 | 1 | 3 |
| 1 | 2 | 2 | 1 | 3 |
| 2 | 1 | 1 | 1 | 3 |
| 2 | 2 | 1 | 1 | 3 |
| 2 | 2 | 2 | 1 | 3 |
| 3 | 1 | 1 | 1 | 3 |
| 3 | 1 | 2 | 1 | 3 |
| 3 | 2 | 1 | 1 | 3 |
| 3 | 2 | 2 | 1 | 3 |

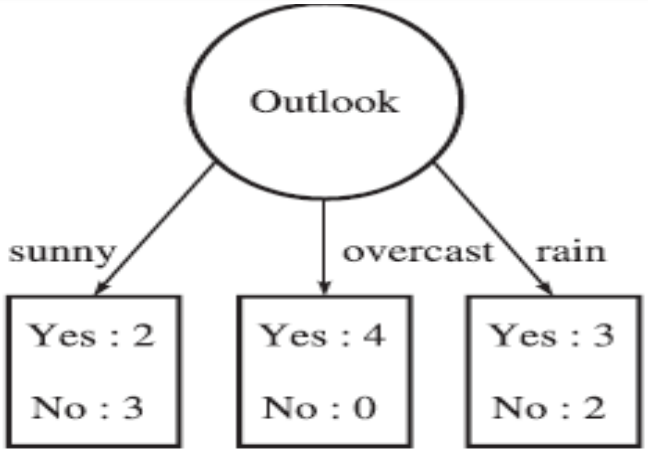
| Age | Eye | Ast | Use | Class |
|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 2 |
| 1 | 1 | 2 | 2 | 1 |
| 1 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 3 | 1 | 1 | 2 | 3 |
| 3 | 1 | 2 | 2 | 3 |
| 3 | 2 | 1 | 2 | 2 |
| 3 | 2 | 2 | 2 | 3 |

# Example 3

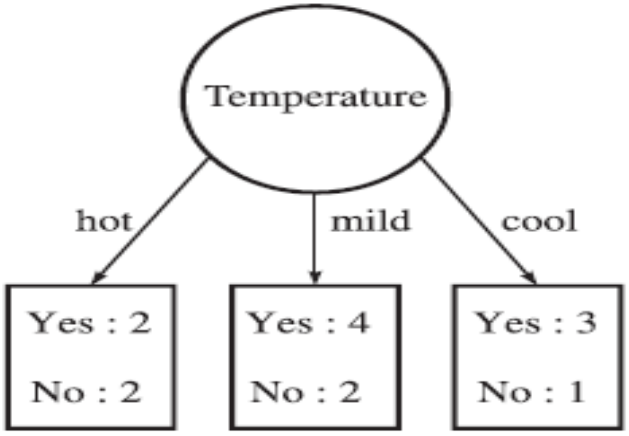
- The input variables are:
  - $x_1 =$ *outlook* ; $x_2 =$ *Temperature*
  - $x_3 =$ *humidity;* $x_4 =$ *wind*
  - Target variable $y =$ *Play Tennis*.
  - Target function to be learnt as:
    $\hat{y}: S \rightarrow [0, 1]$

| Instances | Outlook $x_1$ | Temperature $x_2$ | Humidity $x_3$ | Wind $x_4$ | Play Tennis $y$ |
|---|---|---|---|---|---|
| $s^{(1)}$ | Sunny | Hot | High | Weak | No |
| $s^{(2)}$ | Sunny | Hot | High | Strong | No |
| $s^{(3)}$ | Overcast | Hot | High | Weak | Yes |
| $s^{(4)}$ | Rain | Mild | High | Weak | Yes |
| $s^{(5)}$ | Rain | Cool | Normal | Weak | Yes |
| $s^{(6)}$ | Rain | Cool | Normal | Strong | No |
| $s^{(7)}$ | Overcast | Cool | Normal | Strong | Yes |
| $s^{(8)}$ | Sunny | Mild | High | Weak | No |
| $s^{(9)}$ | Sunny | Cool | Normal | Weak | Yes |
| $s^{(10)}$ | Rain | Mild | Normal | Weak | Yes |
| $s^{(11)}$ | Sunny | Mild | Normal | Strong | Yes |
| $s^{(12)}$ | Overcast | Mild | High | Strong | Yes |
| $s^{(13)}$ | Overcast | Hot | Normal | Weak | Yes |
| $s^{(14)}$ | Rain | Mild | High | Strong | No |

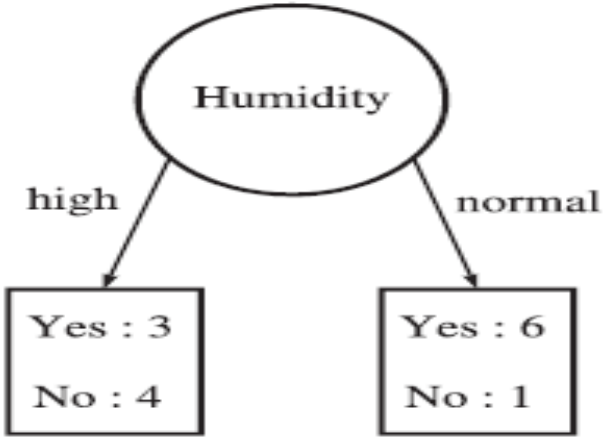
# Example 3…



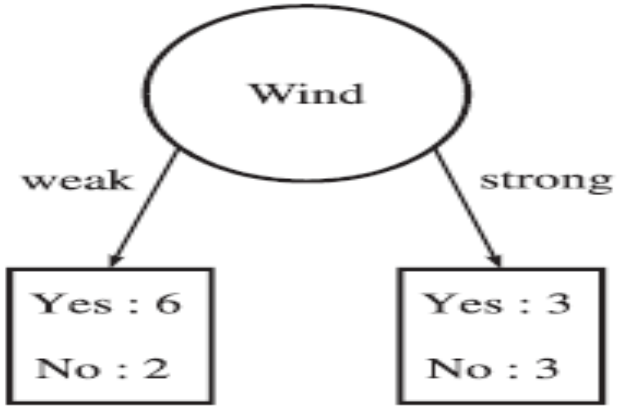

(a)




(b)



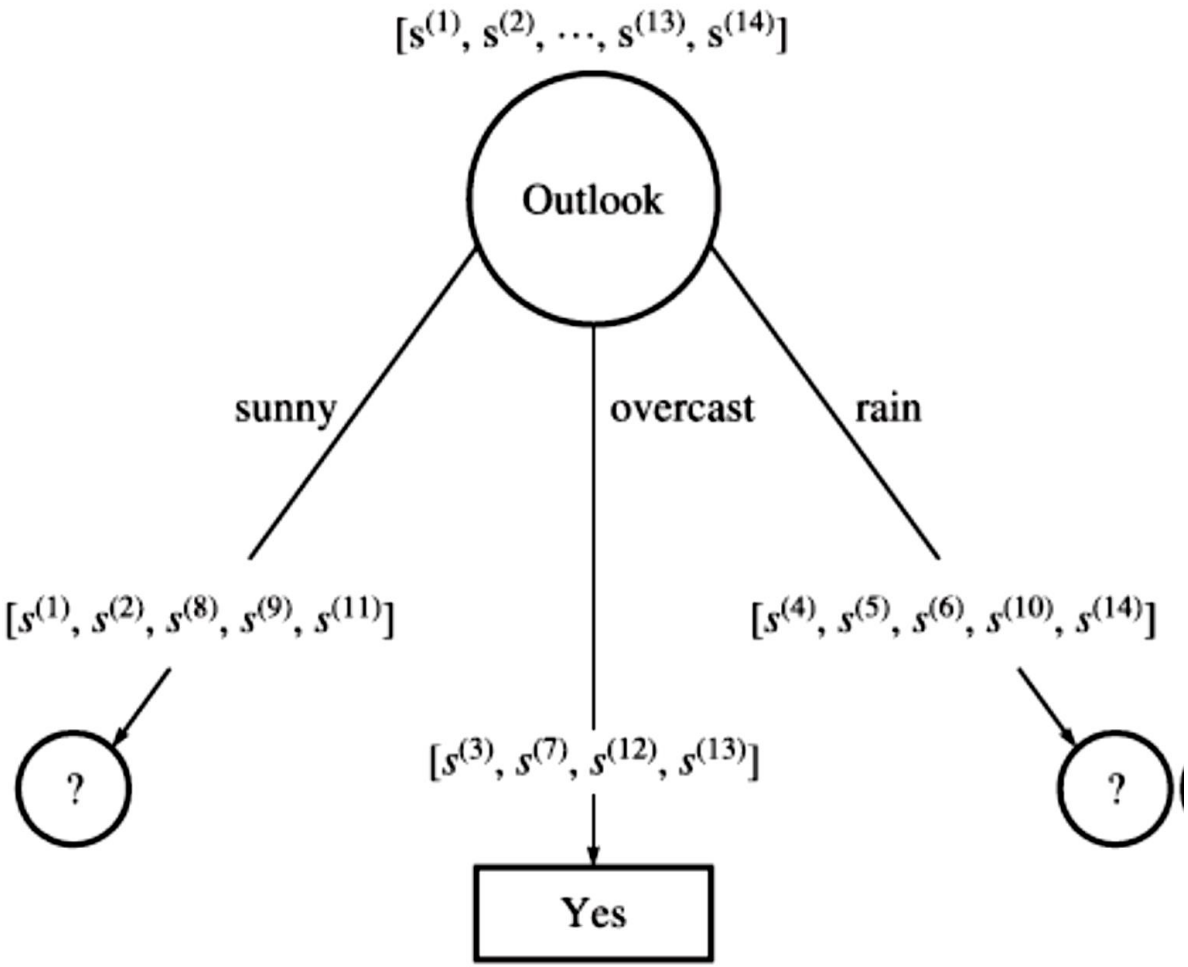

(c)




(d)

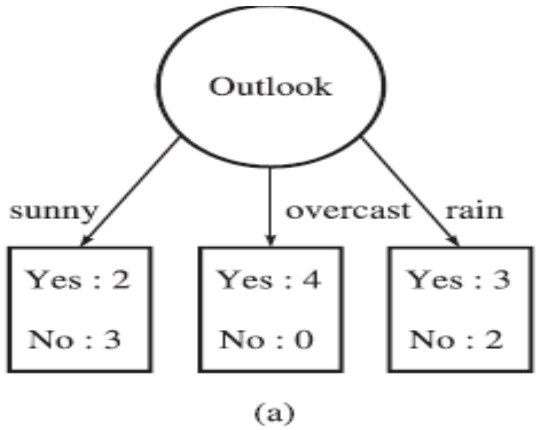
# Example 3…

- A partially learned decision tree
- There are three possibility to select further attribute **Temperature, Humidity, and Wind.**
- **Intuitively, "Humidity" is the best to choose**



# Example 3: Information Gain

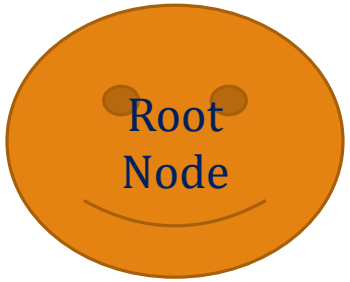
- Information of dataset is computed as:
- $Info(D) = E(D) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94\ bits.$




- $E(D_1) = -\frac{2}{5}\log\left(\frac{2}{5}\right) - \frac{3}{5}\log\left(\frac{3}{5}\right) = 0.97$
- $E(D_2) = 0$
- $E(D_3) = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.97$
- $E_{outlook}(D) = \frac{5}{14} \times 0.97 + \frac{5}{14} \times .97 = 0.693 = \mathbf{Info(D, x_1)}$
- **Similarly,** $E_{Temperature}(D) = 0.911, E_{Humidity}(D) = 0.788,$ **,** $E_{Wind}(D) = 0.892$

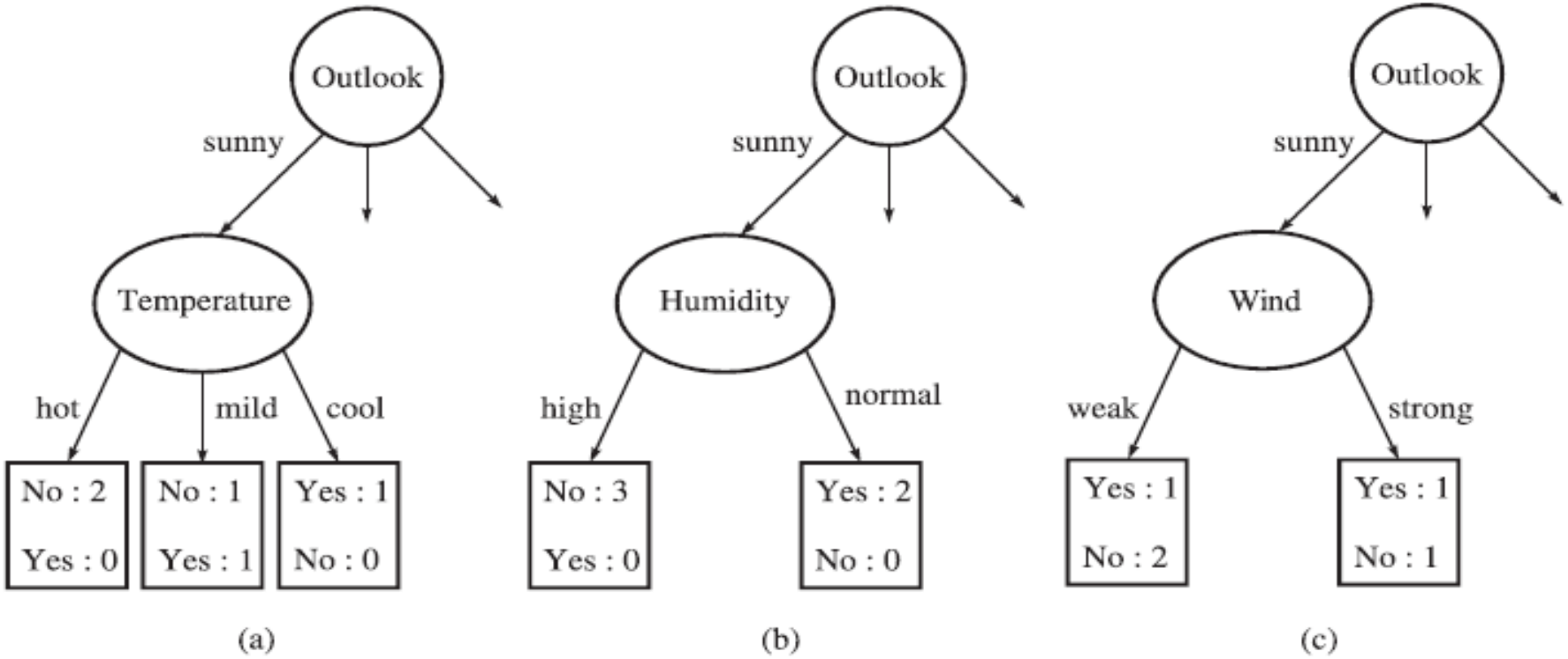
# Example 3: Information Gain…

- $Gain(D, x_j) = E(D) - E(D, x_j)$
- $Gain(D, x_1) = .94 - 0.693 = 0.247$: **Outlook**
- $Gain(D, x_2) = .029$ : **Temperature**
- $Gain(D, x_3) = .152$ : **Humidity**
- $Gain(D, x_4) = .048$ : **Wind**

# Example 3: Information Gain…

- Same strategy is applied recursively to each subset of training instances.



# Example 3: Information Gain…

- Further branching at the node reached when outlook is sunny.
- The information gain at daughter node are:
- ***Temperature: Hot-02, Mild-02, Cool-01***

$$Info(Hot) = -\frac{2}{2}\log\left(\frac{2}{2}\right) - \frac{0}{2}\log\left(\frac{0}{2}\right) = 0.0$$

$$Info(Mild) = -\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right) = 0.5$$

$$Info(Cool) = -\frac{1}{1}\log\left(\frac{1}{1}\right) - \frac{0}{1}\log\left(\frac{0}{1}\right) = 0.0$$

$$Info(Ds, Temp) = \frac{2}{5} * 0.0 + \frac{2}{5} * 0.5 + \frac{1}{5} * 0.0 = 0.4$$

$$Info(Ds) = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.97095$$

$$Gain(Ds, Temp) = Info(Ds) - Info(Ds, Temp) = 0.97095 - 0.4 = 0.571$$

# Example 3: Information Gain…

- Similarly, for Humidity and Wind
- ***Humidity: High-03 (N-3, Y-0), Normal-02(N-0, Y-2)***

$$Info(High) = -\frac{3}{3}\log\left(\frac{3}{3}\right) - \frac{0}{3}\log\left(\frac{0}{3}\right) = 0.0$$

$$Info(Normal) = -\frac{2}{2}\log\left(\frac{2}{2}\right) - \frac{0}{2}\log\left(\frac{0}{2}\right) = 0.0$$

$$Info(Ds, Humidity) = \frac{3}{5} * 0.0 + \frac{2}{5} * 0.0 = 0.0$$

$$Info(Ds) = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.97095$$

$$Gain(Ds, Humidity) = .97095 - 0.0 = 0.971$$

- ***Wind: Weak-03(Y-1, N-2), Strong-02 (Y-1, N-1)***

$$Info(Week) = -\frac{1}{3}\log\left(\frac{1}{3}\right) - \frac{2}{3}\log\left(\frac{2}{3}\right) = .9183$$

$$Info(Strong) = -\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right) = 0.5$$

$$Info(Ds, Wind) = \frac{3}{5} * 0.9183 + \frac{2}{5} * 0.5 = 0.74978$$

$$Gain(Ds, Wind) = .97095 - 0.74978 = 0.221$$

We choose "Humidity", since it has highest gain. No further splitting, reached as leaf node

# Facts of Information Gain

- Information gain $G(\mathcal{D}, x_i)$ measures the expected reduction in entropy, caused by portioning the patterns in dataset $\mathcal{D}$.
- It gave good result and there are several data mining software's where it is used.
- It has problem when attribute has large possible values, which give rise in multiway splitting with daughter node.
- It has strong bias in favour of attributes with large number of values. The attribute with large number of values will get selected at root itself and may lead to all leaf nodes, resulting in to a too simple hypothesis model unable to capture the structure of the data.

# C4.5
# Gain Ratio

# Gain Ratio

- It is successor of ID3, uses an extension of information gain, known as gain ratio, which attempts to overcome the bias in information gain.
- it uses a normalization of information gain using "split information" and analogous term of information gain $Info(\mathcal{D}, x_j)$ is used to defined as $SplitInfo(\mathcal{D}, x_j)$.
- $SplitInfo(\mathcal{D}, x_j) = -\sum_{l=1}^{d_j} \frac{|\mathcal{D}_l|}{|\mathcal{D}|} log_2^{\frac{|\mathcal{D}_l|}{|\mathcal{D}|}}$
- The gain ratio is defined as $GainRatio(\mathcal{D}, x_j) = \frac{Gain(\mathcal{D}, x_j)}{SplitInfo(\mathcal{D}, x_j)}$
- The attributes with maximum gain is selected as splitting node
  - Designed to overcome the disadvantage of Information Gain

# Dataset

| Instances | Outlook $x_1$ | Temperature $x_2$ | Humidity $x_3$ | Wind $x_4$ | Play Tennis $y$ |
|---|---|---|---|---|---|
| $s^{(1)}$ | Sunny | Hot | High | Weak | No |
| $s^{(2)}$ | Sunny | Hot | High | Strong | No |
| $s^{(3)}$ | Overcast | Hot | High | Weak | Yes |
| $s^{(4)}$ | Rain | Mild | High | Weak | Yes |
| $s^{(5)}$ | Rain | Cool | Normal | Weak | Yes |
| $s^{(6)}$ | Rain | Cool | Normal | Strong | No |
| $s^{(7)}$ | Overcast | Cool | Normal | Strong | Yes |
| $s^{(8)}$ | Sunny | Mild | High | Weak | No |
| $s^{(9)}$ | Sunny | Cool | Normal | Weak | Yes |
| $s^{(10)}$ | Rain | Mild | Normal | Weak | Yes |
| $s^{(11)}$ | Sunny | Mild | Normal | Strong | Yes |
| $s^{(12)}$ | Overcast | Mild | High | Strong | Yes |
| $s^{(13)}$ | Overcast | Hot | Normal | Weak | Yes |
| $s^{(14)}$ | Rain | Mild | High | Strong | No |

# Example :Gain Ratio

- Considering the same example of weather data, $x_i = Outlook$ splits the dataset into three subsets of size 5, 4, and 5.
- The *SplitInfo* is given by

$$SplitInfo(\mathcal{D}, x_1) = -\sum_{l=1}^{3} \frac{|\mathcal{D}_l|}{|\mathcal{D}|} \log_2 \frac{|\mathcal{D}_l|}{|\mathcal{D}|}$$

$$= -\frac{5}{14} \log_2 \frac{5}{14} - \frac{4}{14} \log_2 \frac{4}{14} - \frac{5}{14} \log_2 \frac{5}{14}$$

$$= 1.577$$

# Example :Gain Ratio…

- Normalize the information gain by dividing by the ***split info*** value to get *gain ratio*.

$$GainRatio(\mathcal{D}, x_1) = \frac{Gain(\mathcal{D}, x_1)}{SplitInfo(\mathcal{D}, x_1)}$$

$$= \frac{0.247}{1.577} = 0.156$$

# Example :Gain Ratio…

- Similarly, it can be calculated for others attributes such as:

$$
\begin{aligned}
&\textit{Outlook} && : \textit{Gain}(\mathcal{D}, x_1) = 0.247, \textit{SplitInfo}(\mathcal{D}, x_1) = 1.577, \textit{GainRatio}(\mathcal{D}, x_1) = 0.156\\
&\textit{Temperature} && : \textit{Gain}(\mathcal{D}, x_2) = 0.029, \textit{SplitInfo}(\mathcal{D}, x_2) = 1.362, \textit{GainRatio}(\mathcal{D}, x_2) = 0.019\\
&\textit{Humidity} && : \textit{Gain}(\mathcal{D}, x_3) = 0.152, \textit{SplitInfo}(\mathcal{D}, x_3) = 1.000, \textit{GainRatio}(\mathcal{D}, x_3) = 0.152\\
&\textit{Wind} && : \textit{Gain}(\mathcal{D}, x_4) = 0.048, \textit{SplitInfo}(\mathcal{D}, x_4) = 0.985, \textit{GainRatio}(\mathcal{D}, x_4) = 0.049
\end{aligned}
$$

- Outlook still comes out on top but Humidity is much closers contender because it split the data into two subsets instead three.

# CART
# GINI INDEX

CLASSIFICATION AND REGRESSION TREE

# GINI Index

- The Gini Index **facilitates the bigger distributions** so easy to implement whereas the Information Gain **favors lesser distributions** having small count with multiple specific values.
- The method of the Gini Index is **used by CART algorithms**, in contrast to it, Information Gain is **used in ID3, C4.5 algorithms**.
- Gini index **operates on the categorical target variables** in terms of "success" or "failure" and **performs only binary split**, in opposite to that Information Gain **computes the difference between entropy before and after the split** and indicates the impurity in classes of elements.

# GINI Index…

- It is also popular splitting criterion, which is named as Gini in the name of Italian **statistician** and ***economist*** **Corrado Gini**.
- GINI index is defined as
  - $GINI(\mathcal{D}) = 1 - \sum_{q=1}^{M} P_q^2$
- Where $P_q$ is the probability that a tuple in $\mathcal{D}$ belongs to class $y_q$, and is estimated by
  - $P_q = \frac{freq(y_d, \mathcal{D})}{|\mathcal{D}|}$
- Gini index considers a binary split for each attribute.

# GINI Index…

- Let us first consider the case where $x_j$ is continuous-valued attribute having $d_j$ distinct values$v_{lx_j}; l = 1, 2 \ldots . d_j$.
- It is common to take mid-point between each pair of (sorted) adjacent values as a possible split-point.
- The point giving the **minimum Gini index** for the attribute $x_j$ is taken as its split-point.

# GINI Index…

- For a possible split-point of $x_j$, $\mathcal{D}_1$ is the number of tuples in $\mathcal{D}$ satisfying $x_j \leq split - point$, and $\mathcal{D}_2$ is the set of tuples satisfying $x_1 > split - point$.
- The reduction in impurity that would be incurred by a binary split on $x_j$ is:
- $\Delta Gini(x_j) = Gini(\mathcal{D}) - Gini(\mathcal{D}, x_j)$
- $Gini(\mathcal{D}, x_j) = \frac{|\mathcal{D}_1|}{\mathcal{D}} Gini(\mathcal{D}_1) + \frac{|\mathcal{D}_2|}{\mathcal{D}} Gini(\mathcal{D}_2)$
- The attribute that maximizes the reduction in impurity is selected as the splitting attribute.
- The attribute that has Minimum GINI index.

# Measure of Impurity: GINI

- Gini Index for a given 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** | |

$$GINI(\mathcal{D}) = 1 - \sum_{q=1}^{M} P_q^2$$

# Examples for computing GINI

$$GINI(\mathcal{D}) = 1 - \sum_{q=1}^{M} P_q^2$$

| | |
|---|---|
| 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: *Classification And Regression Trees*
- When a node p is split into k partitions (children), the quality of split is computed as,

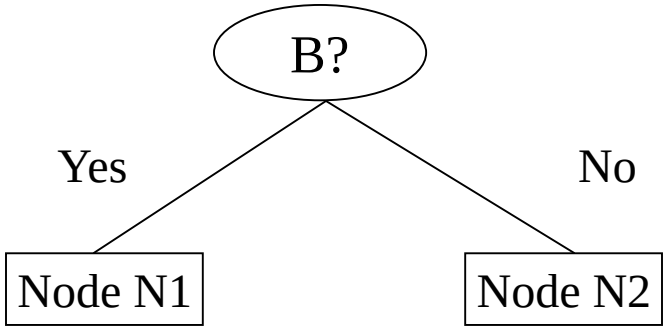
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

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 – (5/6)² – (2/6)²**
**= 0.194**

**Gini(N2)**
**= 1 – (1/6)² – (4/6)²**
**= 0.528**

| | N1 | N2 |
|---|---|---|
| C1 | **5** | **1** |
| C2 | **2** | **4** |
| **Gini=0.333** | | |

**Gini(Children)**
**= 7/12 * 0.194 + 5/12 * 0.528**
**= 0.333**

# 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 | 0.393 | | |

Two-way split
(find best partition of values)

| | CarType | |
|---|---|---|
| | {Sports, Luxury} | {Family} |
| C1 | 3 | 1 |
| C2 | 2 | 4 |
| Gini | 0.400 | |

| | CarType | |
|---|---|---|
| | {Sports} | {Family, Luxury} |
| C1 | 2 | 2 |
| C2 | 1 | 5 |
| Gini | 0.419 | |

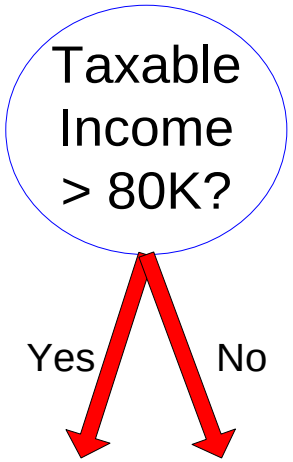
# Example

| Past Trend | Open Interest | Trading Volume | Return |
|---|---|---|---|
| Positive | Low | High | Up |
| Negative | High | Low | Down |
| Positive | Low | High | Up |
| Positive | High | High | Up |
| Negative | Low | High | Down |
| Positive | Low | Low | Down |
| Negative | High | High | Down |
| Negative | Low | High | Down |
| Positive | Low | Low | Down |
| Positive | High | High | Up |

# 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* | 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 |



# Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

| Cheat | No | No | No | Yes | Yes | Yes | No | No | No | No |
|---|---|---|---|---|---|---|---|---|---|---|
| Taxable 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 |

# Splitting Criteria based on Classification Error

- Classification error at a node t :

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

- 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**

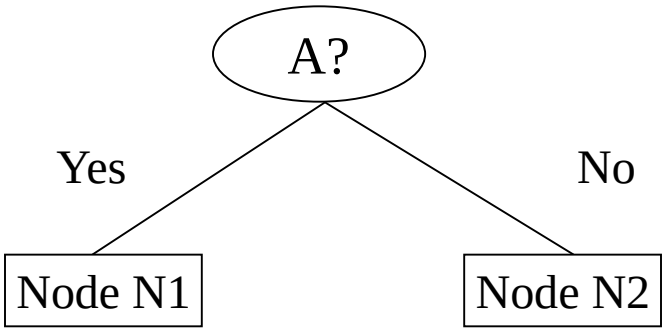
**Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6**

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

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

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

# Misclassification Error vs Gini




| | Parent |
|---|---|
| C1 | **7** |
| C2 | **3** |
| **Gini = 0.42** | |

**Gini(N1)**
**= 1 – (3/3)² – (0/3)²**
**= 0**

**Gini(N2)**
**= 1 – (4/7)² – (3/7)²**
**= 0.489**

| | N1 | N2 |
|---|---|---|
| C1 | **3** | **4** |
| C2 | **0** | **3** |

**Gini(Children)**
**= 3/10 * 0**
**+ 7/10 * 0.489**
**= 0.342**

# Example: GINI Index

$$GINI(\mathcal{D}) = 1 - \sum_{q=1}^{M} P_q^2$$

| Instances | Outlook $x_1$ | Temperature $x_2$ | Humidity $x_3$ | Wind $x_4$ | Play Tennis $y$ |
|---|---|---|---|---|---|
| $s^{(1)}$ | Sunny | Hot | High | Weak | No |
| $s^{(2)}$ | Sunny | Hot | High | Strong | No |
| $s^{(3)}$ | Overcast | Hot | High | Weak | Yes |
| $s^{(4)}$ | Rain | Mild | High | Weak | Yes |
| $s^{(5)}$ | Rain | Cool | Normal | Weak | Yes |
| $s^{(6)}$ | Rain | Cool | Normal | Strong | No |
| $s^{(7)}$ | Overcast | Cool | Normal | Strong | Yes |
| $s^{(8)}$ | Sunny | Mild | High | Weak | No |
| $s^{(9)}$ | Sunny | Cool | Normal | Weak | Yes |
| $s^{(10)}$ | Rain | Mild | Normal | Weak | Yes |
| $s^{(11)}$ | Sunny | Mild | Normal | Strong | Yes |
| $s^{(12)}$ | Overcast | Mild | High | Strong | Yes |
| $s^{(13)}$ | Overcast | Hot | Normal | Weak | Yes |
| $s^{(14)}$ | Rain | Mild | High | Strong | No |

# Example: GINI Index…

- First attribute is "Outlook"

| **Outlook** | *Yes* | *No* | *Number of instances* |
|---|---|---|---|
| **Sunny** | 2 | 3 | 5 |
| **Overcast** | 4 | 0 | 4 |
| **Rain** | 3 | 2 | 5 |

Gini(Outlook=Sunny) = $1 – (2/5)^2 – (3/5)^2 = 1 – 0.16 – 0.36 = 0.48$
Gini(Outlook=Overcast) = $1 – (4/4)^2 – (0/4)^2 = 0$
Gini(Outlook=Rain) = $1 – (3/5)^2 – (2/5)^2 = 1 – 0.36 – 0.16 = 0.48$

Then, we will calculate weighted sum of Gini indexes for outlook feature.
**Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171 = 0.342**

# Example: GINI Index…

- Second attribute is “Temperature”

| **Temperature** | *Yes* | *No* | Number of instances |
|---|---|---|---|
| **Hot** | 2 | 2 | 4 |
| **Cool** | 3 | 1 | 4 |
| **Mild** | 4 | 2 | 6 |

Gini(Temp=Hot) = $1 – (2/4)^2 – (2/4)^2 = 0.5$
Gini(Temp=Cool) = $1 – (3/4)^2 – (1/4)^2 = 1 – 0.5625 – 0.0625 = 0.375$
Gini(Temp=Mild) = $1 – (4/6)^2 – (2/6)^2 = 1 – 0.444 – 0.111 = 0.445$

**We’ll calculate weighted sum of gini index for temperature feature**
**Gini(Temp) = (4/14) x 0.5 + (4/14) x 0.375 + (6/14) x 0.445 = 0.142 + 0.107 + 0.190 = 0.439**

# Example: GINI Index…

- Third attribute is "Humidity"

| **Humidity** | *Yes* | *No* | Number of instances |
|---|---|---|---|
| **High** | 3 | 4 | 7 |
| **Normal** | 6 | 1 | 7 |

Gini(Humidity=High) = $1 – (3/7)^2 – (4/7)^2 = 1 – 0.183 – 0.326 = 0.489$
Gini(Humidity=Normal) = $1 – (6/7)^2 – (1/7)^2 = 1 – 0.734 – 0.02 = 0.244$

**Weighted sum for humidity feature will be calculated next**
**Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367**

# Example: GINI Index…

- Fourth attribute is "**Wind**"

| **Wind** | *Yes* | *No* | Number of instances |
|---|---|---|---|
| Weak | 6 | 2 | 8 |
| Strong | 3 | 3 | 6 |

Gini(Wind=Weak) = $1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.062 = 0.375$
Gini(Wind=Strong) = $1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$

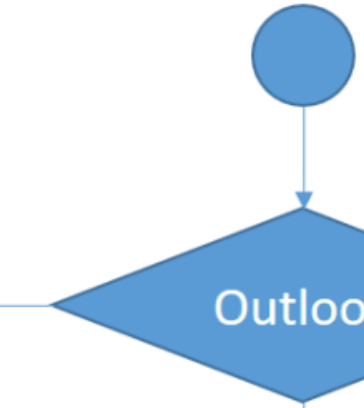
**Gini(Wind) = (8/14) x 0.375 + (6/14) x 0.5 = 0.428**

# Example: GINI Index...

- Gini Index of all attributes are as:

| Feature | Gini index |
|---|---|
| **Outlook** | **0.342** |
| Temperature | 0.439 |
| **Humidity** | **0.367** |
| Wind | 0.428 |

# Decision Tree at First Split



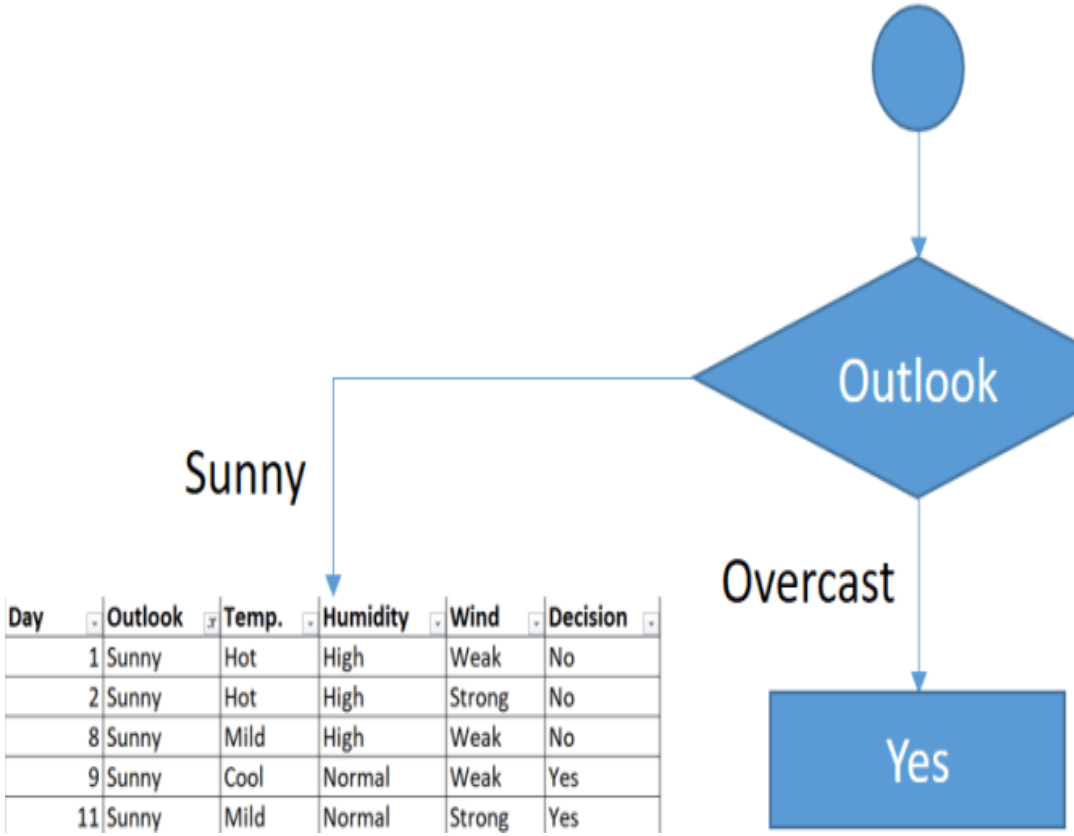

**In this tree, overcast has only yes. Hence, it reaches at leaf node**

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 8 | Sunny | Mild | High | Weak | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 6 | Rain | Cool | Normal | Strong | No |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 14 | Rain | Mild | High | Strong | No |

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 3 | Overcast | Hot | High | Weak | Yes |
| 7 | Overcast | Cool | Normal | Strong | Yes |
| 12 | Overcast | Mild | High | Strong | Yes |
| 13 | Overcast | Hot | Normal | Weak | Yes |

# Decision Tree at First Split




| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 8 | Sunny | Mild | High | Weak | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 6 | Rain | Cool | Normal | Strong | No |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 14 | Rain | Mild | High | Strong | No |

**In this tree, overcast has only yes. Hence, it reaches at leaf node**

# DT: GINI Index for sub dataset

- Same procedure is applied for the sub datasets.
- The sub dataset for "**sunny**" outlook. We need to find the **GINI index** scores for temperature, humidity and wind features, respectively.

| Day | Outlook | **Temp.** | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 8 | Sunny | Mild | High | Weak | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |

# DT: GINI Index for sub dataset

- Gini of temperature for "**sunny**" outlook.

| **Temperature** | Yes | No | Number of instances |
|---|---|---|---|
| Hot | 0 | 2 | 2 |
| Cool | 1 | 0 | 1 |
| Mild | 1 | 1 | 2 |

Gini(Outlook=Sunny and Temp.=Hot) = $1 - (0/2)^2 - (2/2)^2 = 0$
Gini(Outlook=Sunny and Temp.=Cool) = $1 - (1/1)^2 - (0/1)^2 = 0$
Gini(Outlook=Sunny and Temp.=Mild) = $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$

**Gini(Outlook=Sunny and Temp.) = (2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2**

# DT: GINI Index for sub dataset

- Gini of **humidity** for sunny outlook.

| Humidity | Yes | No | Number of instances |
|---|---|---|---|
| High | 0 | 3 | 3 |
| Normal | 2 | 0 | 2 |

Gini(Outlook=Sunny and Humidity=High) = $1 - (0/3)^2 - (3/3)^2 = 0$
Gini(Outlook=Sunny and Humidity=Normal) = $1 - (2/2)^2 - (0/2)^2 = 0$

**Gini(Outlook=Sunny and Humidity) = (3/5)x0 + (2/5)x0 = 0**

# DT: GINI Index for sub dataset

- Gini of **wind** for sunny outlook.

| Wind | Yes | No | Number of instances |
|---|---|---|---|
| Weak | 1 | 2 | 3 |
| Strong | 1 | 1 | 2 |

Gini(Outlook=Sunny and Wind=Weak) = $1 - (1/3)^2 - (2/3)^2 = 0.266$
Gini(Outlook=Sunny and Wind=Strong) = $1- (1/2)^2 - (1/2)^2 = 0.2$

**Gini(Outlook=Sunny and Wind) = (3/5)x0.266 + (2/5)x0.2 = 0.466**

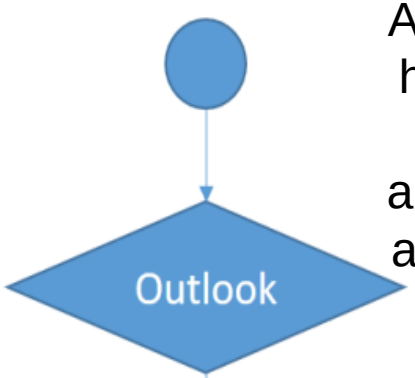
# DT: GINI Index for sub dataset

- Decision for **sunny** outlook.

| Feature | Gini index |
|---|---|
| Temperature | 0.2 |
| Humidity | 0 |
| Wind | 0.466 |

We'll put humidity check at the extension of sunny outlook.

# DT: Second Split

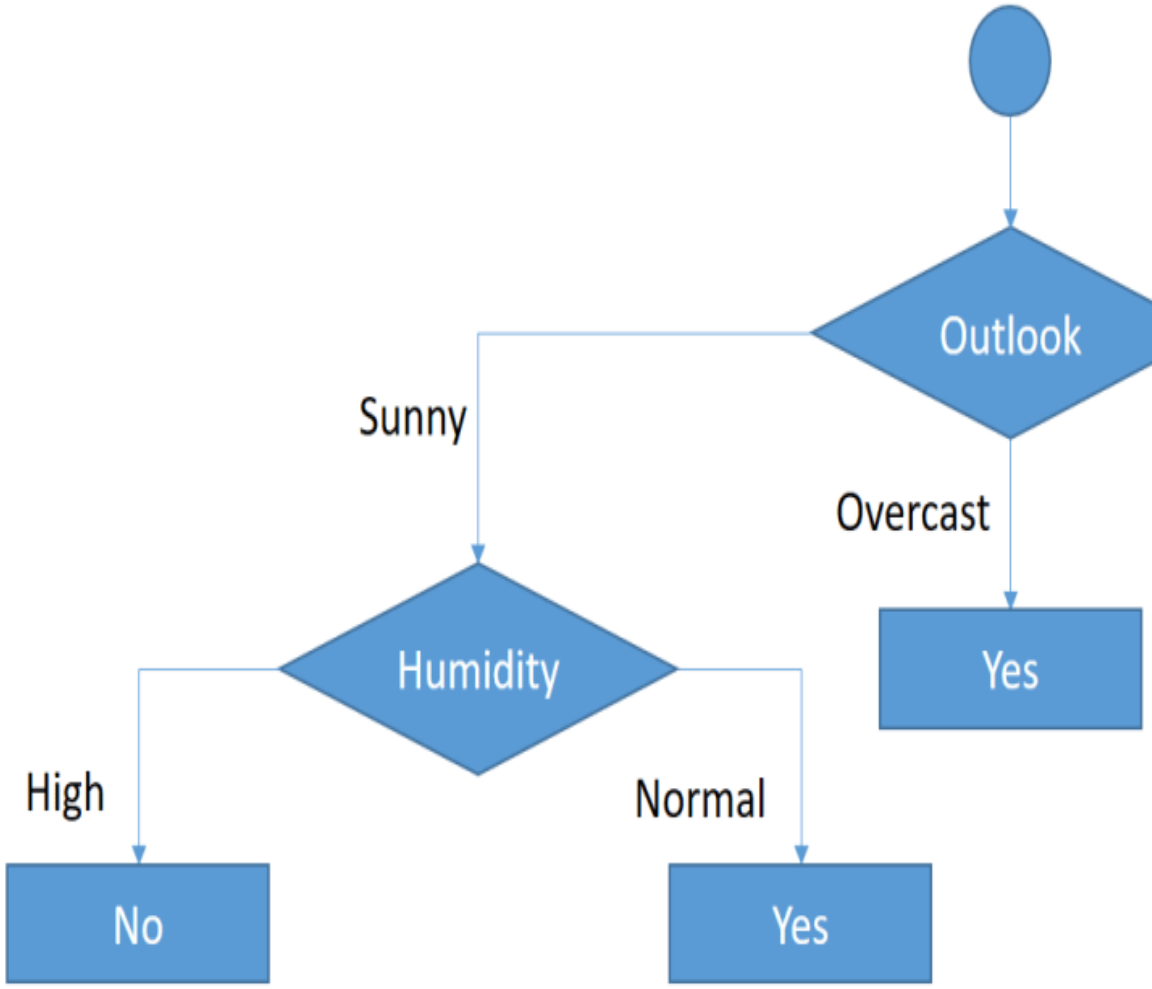



As seen, decision is always no for high humidity and sunny outlook. On the other hand, decision will always be yes for normal humidity and sunny outlook. This branch is over.

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 8 | Sunny | Mild | High | Weak | No |

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Strong | Yes |

# Decision Tree: Second Split




| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 6 | Rain | Cool | Normal | Strong | No |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 14 | Rain | Mild | High | Strong | No |

# DT: Rain_Outlook

- **Rain outlook**

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 6 | Rain | Cool | Normal | Strong | No |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 14 | Rain | Mild | High | Strong | No |

We'll calculate Gini index scores for temperature, humidity and wind features when outlook is rain.

# DT: Rain_Outlook…

- Gini of **temperature** for ***rain*** **outlook**

| Temperature | Yes | No | Number of instances |
|---|---|---|---|
| Cool | 1 | 1 | 2 |
| Mild | 2 | 1 | 3 |

Gini(Outlook=Rain and Temp.=Cool) = $1 - (1/2)^2 - (1/2)^2 = 0.5$
Gini(Outlook=Rain and Temp.=Mild) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

**Gini(Outlook=Rain and Temp.) = (2/5)x0.5 + (3/5)x0.444 = 0.466**

# DT: Rain_Outlook…

- Gini of **humidity** for ***rain*** **outlook**

| Humidity | Yes | No | Number of instances |
|---|---|---|---|
| High | 1 | 1 | 2 |
| Normal | 2 | 1 | 3 |

Gini(Outlook=Rain and Humidity=High) = $1 - (1/2)^2 - (1/2)^2 = 0.5$
Gini(Outlook=Rain and Humidity=Normal) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

**Gini(Outlook=Rain and Humidity) = (2/5)x0.5 + (3/5)x0.444 = 0.466**

# DT: Rain_Outlook…

- Gini of **wind** for ***rain*** **outlook**

| Wind | Yes | No | Number of instances |
|---|---|---|---|
| Weak | 3 | 0 | 3 |
| Strong | 0 | 2 | 2 |

Gini(Outlook=Rain and Wind=Weak) = $1 – (3/3)^2 – (0/3)^2 = 0$
Gini(Outlook=Rain and Wind=Strong) = $1 – (0/2)^2 – (2/2)^2 = 0$

**Gini(Outlook=Rain and Wind) = (3/5)x0 + (2/5)x0 = 0**

# DT: Rain_Outlook…

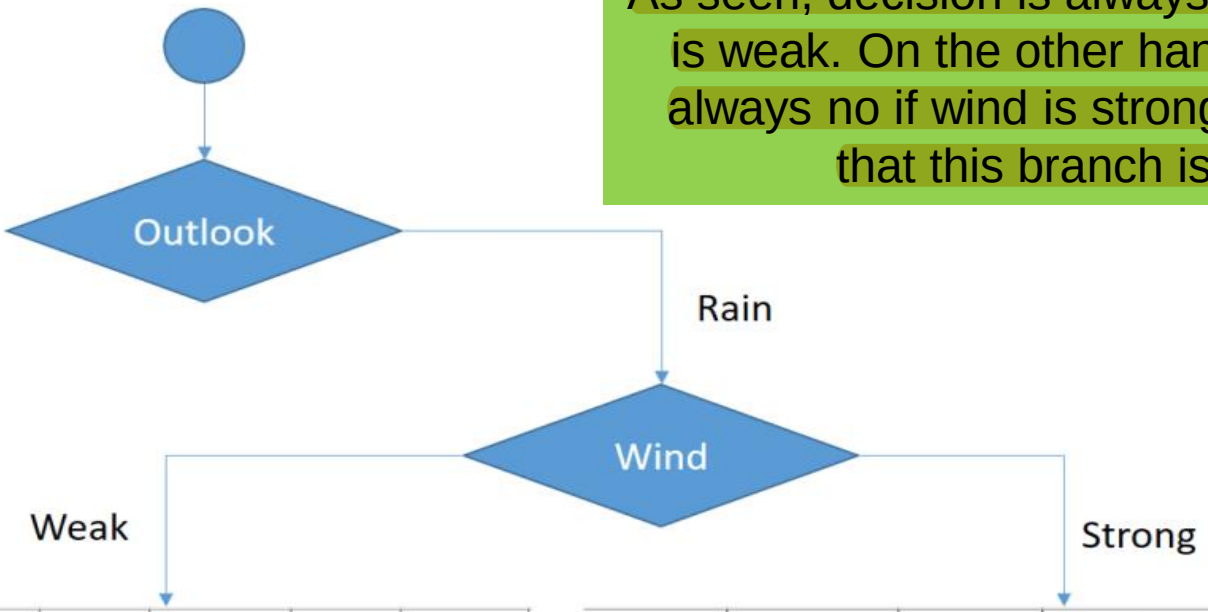
- **Decision for rain outlook**
  - The winner is wind feature for rain outlook because it has the minimum Gini index score in features.

| Feature | Gini index |
|---|---|
| Temperature | 0.466 |
| Humidity | 0.466 |
| Wind | 0 |

# DT: at Third Split

- Put the wind feature for rain outlook branch and monitor the new sub data sets.

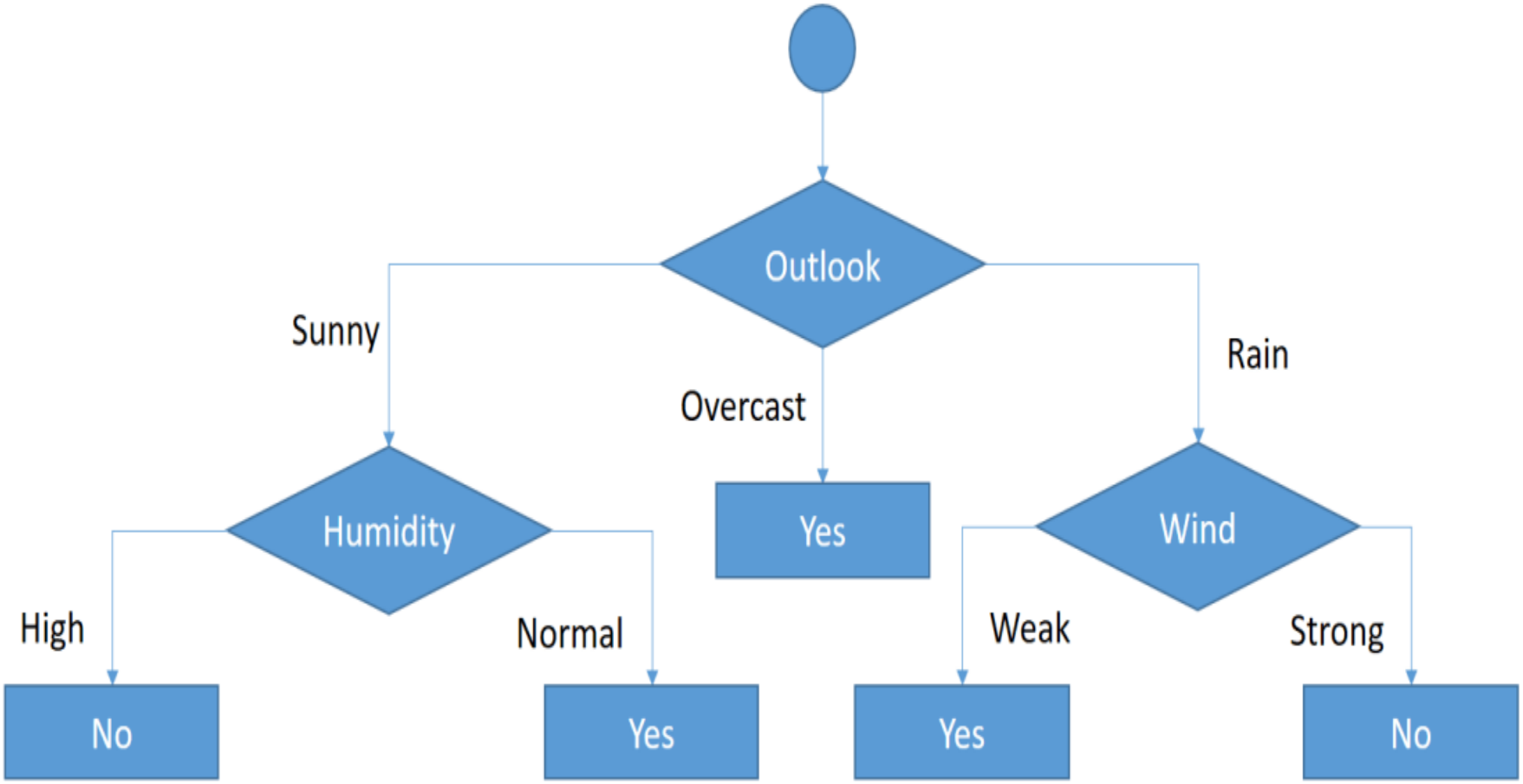
As seen, decision is always yes when wind is weak. On the other hand, decision is always no if wind is strong. This means that this branch is over.




| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 10 | Rain | Mild | Normal | Weak | Yes |

| Day | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| 6 | Rain | Cool | Normal | Strong | No |
| 14 | Rain | Mild | High | Strong | No |

# Decision Tree: Final



# 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.

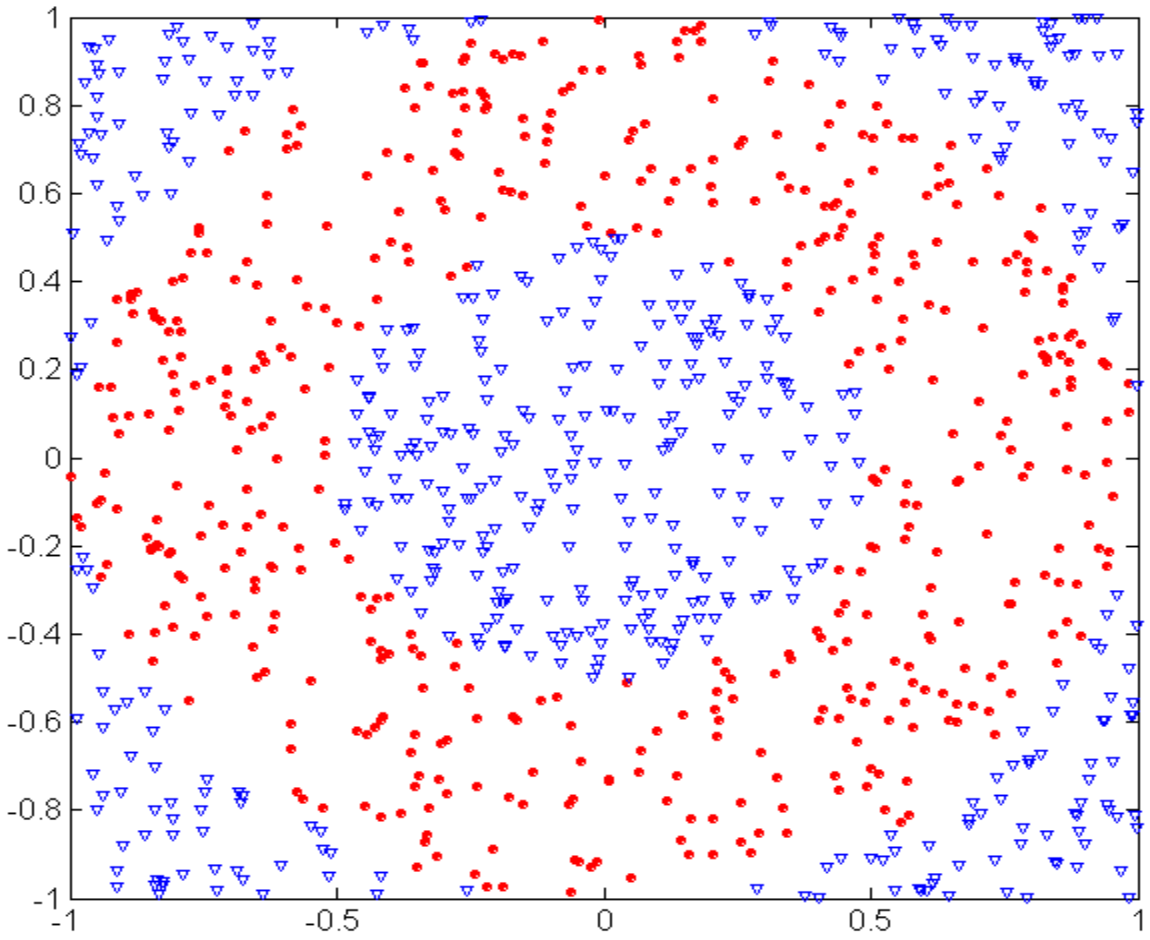
# Decision Tree Based Classification

- **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

# Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

# Underfitting and Overfitting (Example)




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

**Circular points:**

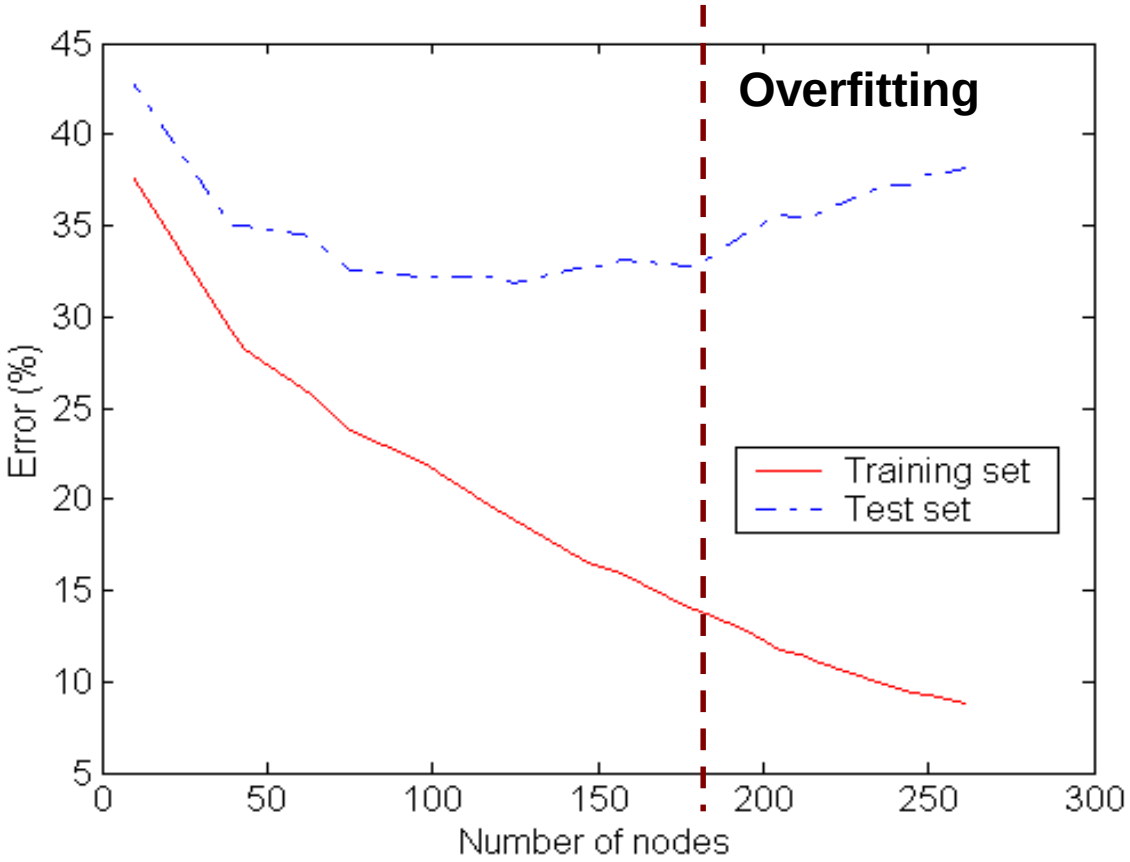
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

**Triangular points:**

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

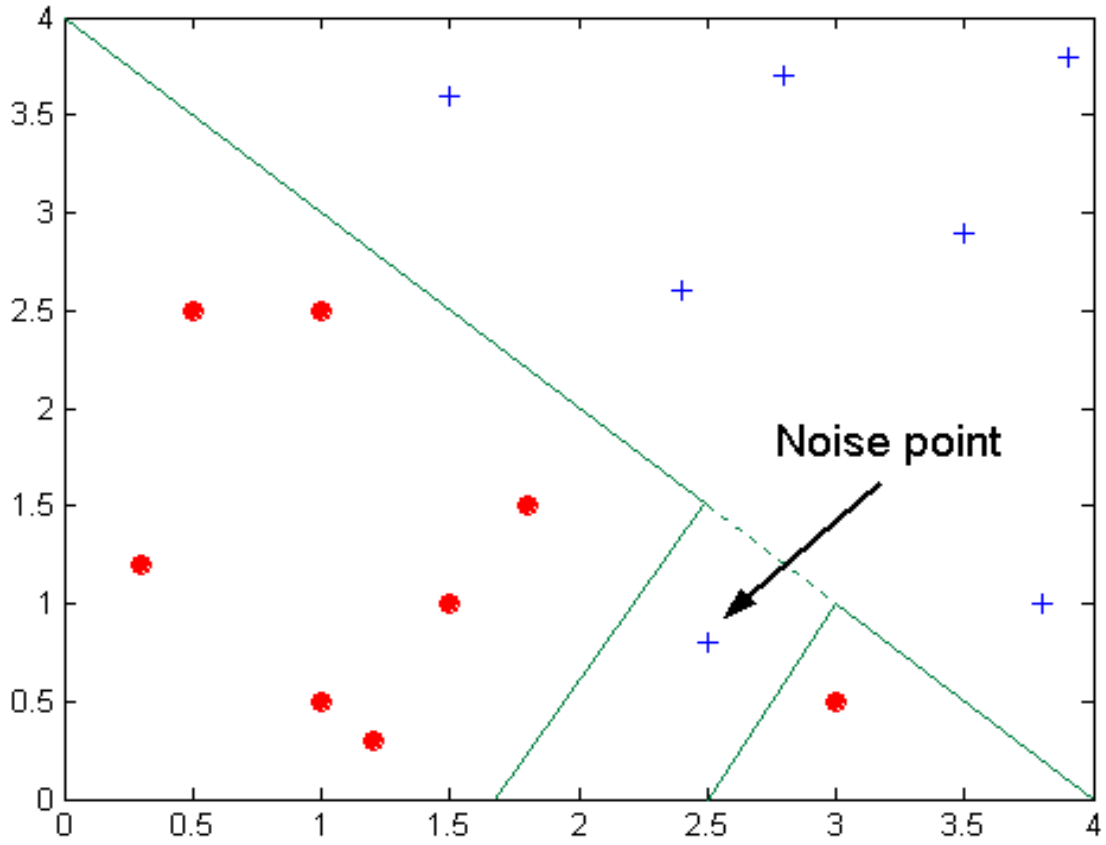
$\text{sqrt}(x_1^2+x_2^2) < 1$

# Underfitting and Overfitting



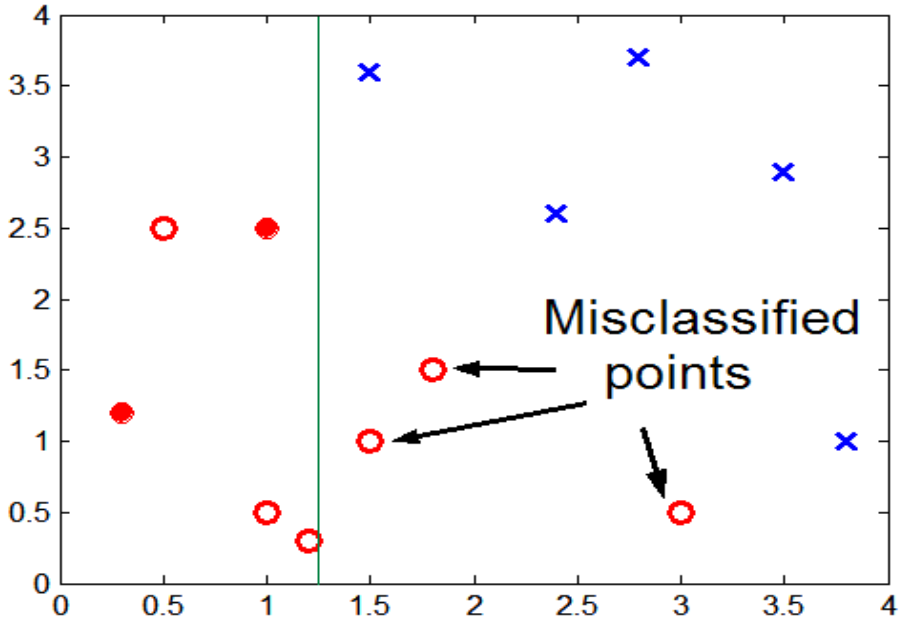

**Underfitting**: when model is too simple, both training and test errors are large

# 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

# Occam's Razor

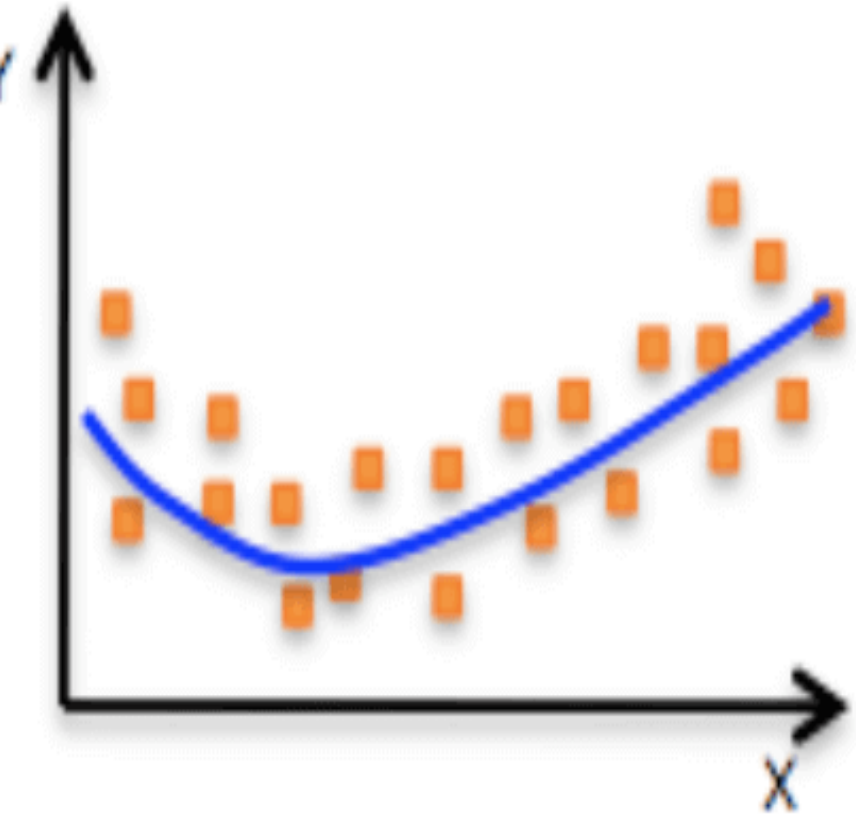
- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.

- For complex models, there is a greater chance that it was fitted accidentally by errors in data.

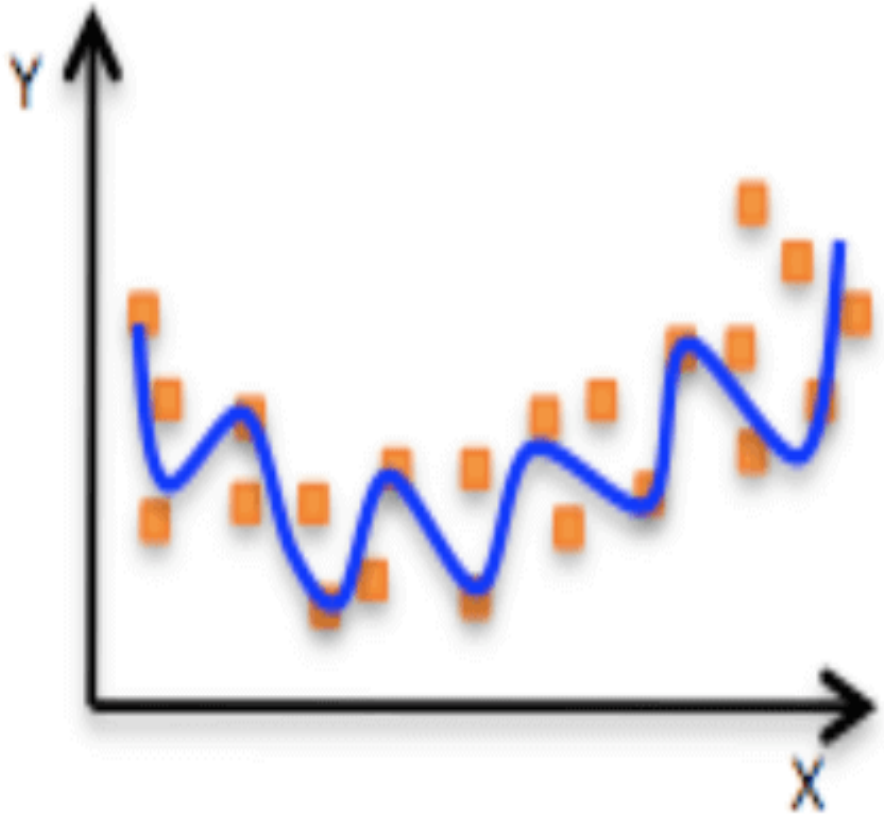
- Therefore, one should include model complexity when evaluating a model.

# 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).




Just right!




overfitting