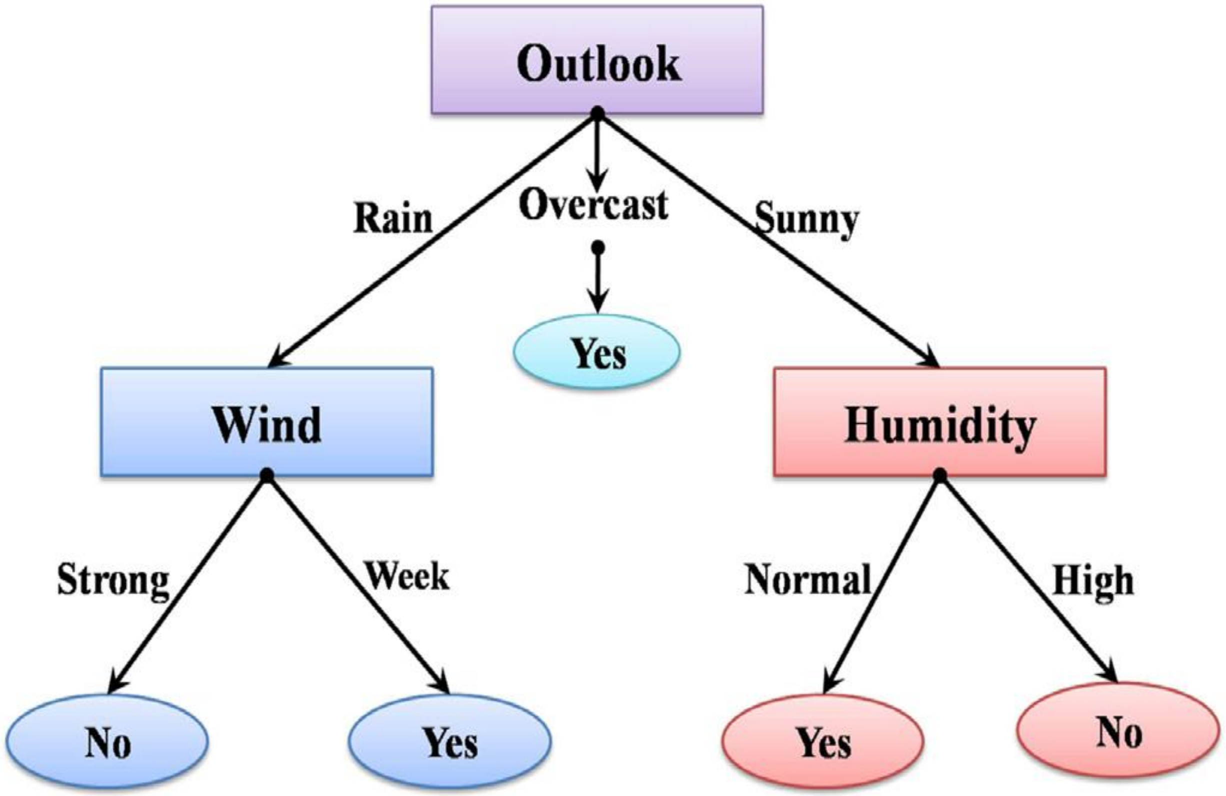
Machine Learning
Decision Tree

Dr. Pratyay Kuila

Dept. of Computer Science & Engineering
NIT Sikkim, Ravangla-737139

Decision Tree

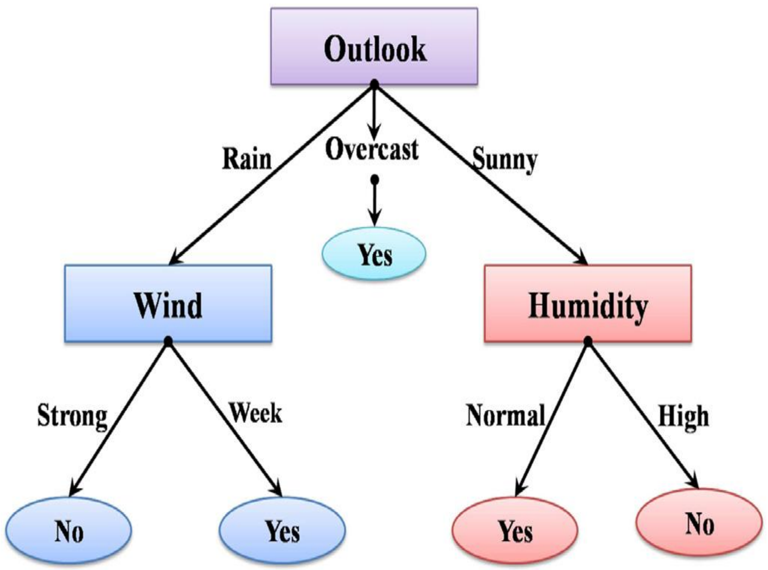


- ❖ *Decision tree* is a classifier.
- ❖ It classifies instances by sorting them down into a form of a tree.
- ❖ The tree is traversed from the root to some leaf node based on some instances.
- ❖ After such traversal, the leaf node provides the classification of the instances.

Decision Tree

Table 1: Training Data for *PlayTennis*.

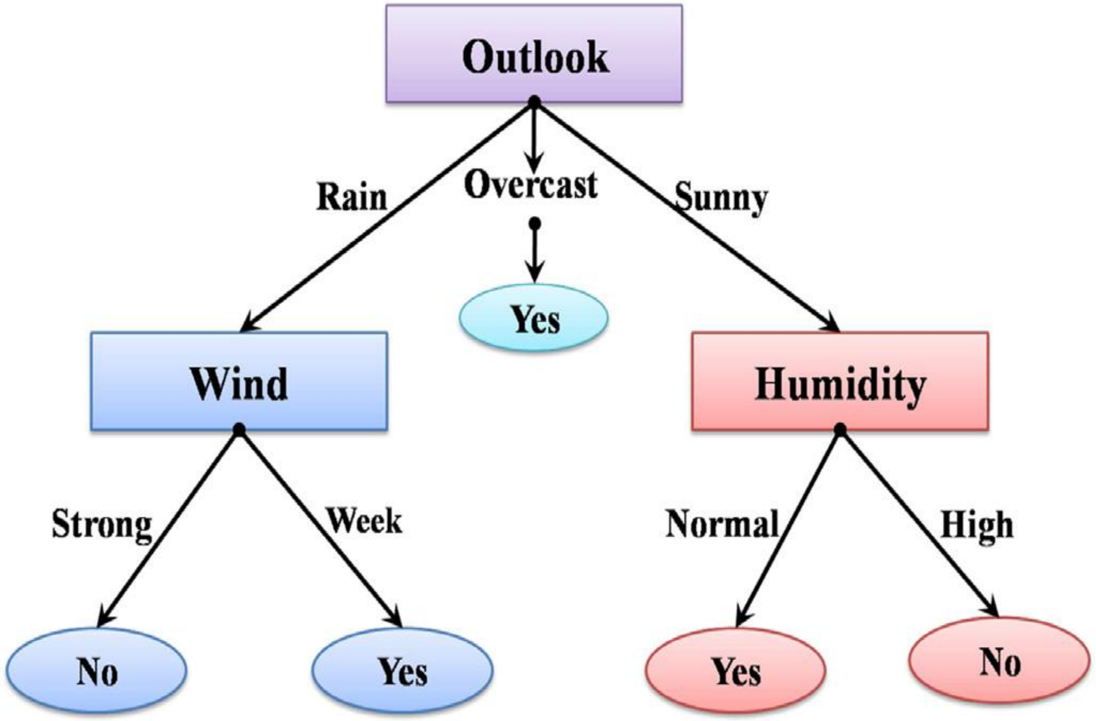
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D_1	*Sunny*	*Hot*	*High*	*Weak*	*No*
D_2	*Sunny*	*Hot*	*High*	*Strong*	*No*
D_3	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
D_4	*Rain*	*Mild*	*High*	*Weak*	*Yes*
D_5	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
D_6	*Rain*	*Cool*	*Normal*	*Strong*	*No*
D_7	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
D_8	*Sunny*	*Mild*	*High*	*Weak*	*No*
D_9	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
D_{10}	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
D_{11}	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
D_{12}	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
D_{13}	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
D_{14}	*Rain*	*Mild*	*High*	*Strong*	*No*



Decision Tree

- Decision trees represent a *disjunction of conjunctions of constraints* on the attribute values of instances.
- For example, the decision tree shown in the figure corresponds to the following expression:

$$(\boldsymbol{Outlook = Sunny} \wedge \boldsymbol{Humidity = Normal})$$
$$\vee$$
$$(\boldsymbol{Outlook = Overcast})$$
$$\vee$$
$$(\boldsymbol{Outlook = Rain} \wedge \boldsymbol{Wind = Weak})$$



Decision Tree

Decision Tree Construction Algorithms: Generally, following three algorithms are used to construct decision tree.

- **ID3:** J. Ross Quinlan proposed this decision tree algorithm (during late 1970's and early 1980's). It was the **third procedure** in the series of **'interactive dichotomizer'** processes.

- **C4.5:** Later, Quinlan's C4.5 (ID3's successor) becomes the benchmark with which newer supervised learning algorithms are often compared.

- **CART:** In 1984, CART (**C**lassification **A**nd **R**egression **T**ree) is proposed.

Decision Tree: Entropy

- In the case of a decision tree for classification, namely, a *classification tree*, the goodness of a split is quantified by an *impurity measure*.
- A split is *pure* if after the split, for all branches, all the instances choosing a branch belong to the same class.
- One possible function to measure impurity is *entropy*.

$$Entropy(S) = -\sum_{b=1}^{K} P_b \log_2 P_b$$

where, K be the number of classes and $0\log 0 \equiv 0$. The range of entropy is $(0, \log_2 K)$.

- Entropy in information theory specifies the minimum number of bits needed to encode the class code of an instance.

Decision Tree: Entropy

Entropy of a set *S*: the entropy of *S* relative to this Boolean classification ($K = 2$) is,

$$Entropy(S) = -\sum_{b=1}^{2} P_b \log P_b = -P_{(-)} \log_2 P_{(-)} - P_{(+)} \log_2 P_{(+)}$$

where, $P_{(+)}$ and $P_{(-)}$ are the proportion of positive and negative examples in *S*.

Example: Suppose *S* is a collection of 14 instances of some Boolean class (e.g., *Yes* and *No*), including 9 positive and 5 negative instances. Then the entropy of *S* is:

$$Entropy([9+,5-]) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

Decision Tree: Entropy

Entropy of a set *S*:

- The entropy is 0 if all members belong to the same class. For example, if all members are positive ($P_{(+)} = 1$), then $P_{(-)} = 0$, and

$$Entropy(S) = -1\times\log_2(1) - 0\times\log_2(0) = -1\times 0 - 0\times\log_2(0) = 0.$$

- The entropy is 1 when the data contains an equal number of positive and negative examples. For example, if half members are positive ($P_{(+)} = ½$), then $P_{(-)} = ½$, and

$$Entropy(S) = -½\log_2(½) - ½\log_2(½) = -1\times\log_2(½) = 1.$$

- If the collection contains unequal numbers of positive and negative examples ($K = 2$), then the entropy is between 0 and $\log_2 K$, i.e., (0, 1).

Decision Tree: Entropy

***Exercise*:** Calculate entropy of these two datasets.

$S^{(i)}$	a_1	a_2	*Class*
1	x	y	+
2	x	u	+
3	z	u	-
4	z	v	+
5	x	v	-
6	z	y	-
7	z	u	+

$S^{(i)}$	a_1	a_2	a_3	a_4	*Class*
1	x	u	n	e	c_1
2	x	u	p	f	c_1
3	x	u	n	g	c_3
4	y	u	n	e	c_3
5	y	v	n	f	c_2
6	x	v	n	e	c_1
7	x	u	p	e	c_2
8	y	v	m	f	c_1
9	x	u	n	f	c_1
10	x	w	p	f	c_1
11	y	w	n	f	c_2
12	x	w	n	g	c_2

Decision Tree: Entropy

- Entropy is not the only possible measure. For a two-class problem, $\varphi(p, 1-p)$ is a nonnegative function measuring the *impurity* of a split if it satisfies the following properties:

> - **$\varphi(½, ½) \geq \varphi(p, 1-p)$, for any $p \in [0, 1]$.**
> - **$\varphi(0, 1) = \varphi(1, 0) = 0$.**
> - **$\varphi(p, 1-p)$ is increasing in p on $[0, ½]$ and decreasing in p on $[½, 1]$.**

- Examples:

> 1. ***Entropy***: $\varphi(p,1-p) = -p\log_2 p - (1-p)\log_2(1-p)$
> 2. ***Gini index***: $\varphi(p,1-p) = 2p(1-p)$
> 3. ***Misclassification error***: $\varphi(p,1-p) = 1 - max(p, 1-p)$

ID3 Algorithm

- Which attribute is the Best Classifier or root of the tree? What is a good quantitative measure of the worth of an attribute?
- We will use statistical properties, like *Entropy* and *Information gain*.
- A pure node has maximum homogeneity, i.e., all *Yes* or *No*.
- An impure node has heterogeneity which is maximum when *Yes* and *No* are in equal number.
- ***Entropy*** is used to measure the (im) purity of an arbitrary collection of examples.
- Therefore, highest Entropy indicates maximum heterogeneity, i.e., *Yes* and *No* are in equal number.

ID3 Algorithm

Entropy of a set S with respect to some attribute x_j:

- Suppose we select an attribute x_j (say *Outlook*) to partition S.
- Let x_j has distinct value, $\{v_{1j}, v_{2j}, v_{3j}, \ldots, v_{dj}\}$ and thereby x_j can be used to split S into$\{s_1, s_2, \ldots, s_d\}$, where s_i contains the instances with value v_{ij}.
- The entropy with respect to x_j can be calculated as follow.

$$Entropy(S, x_j) = \sum_{i=1}^{d} \frac{|s_i|}{|S|} Entropy(s_i)$$

Example: From the Table 1:

x_1= *Outlook*, the corresponding values are: $\{v_{11}=Sunny, v_{21}=Overcast, v_{31}=Rain\}$
x_2= *Temperature* , the corresponding values are: $\{v_{12}=Hot, v_{22}=Mild, v_{32}=Cool\}$
x_3= *Humidity*, the corresponding values are: $\{v_{13}=High, v_{23}=Normal\}$
x_4= *Wind*, the corresponding values are: $\{v_{14}=Weak, v_{24}=Strong\}$.

ID3 Algorithm

Table 1: Training Data for *PlayTennis*.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D_1	*Sunny*	*Hot*	*High*	*Weak*	*No*
D_2	*Sunny*	*Hot*	*High*	*Strong*	*No*
D_3	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
D_4	*Rain*	*Mild*	*High*	*Weak*	*Yes*
D_5	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
D_6	*Rain*	*Cool*	*Normal*	*Strong*	*No*
D_7	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
D_8	*Sunny*	*Mild*	*High*	*Weak*	*No*
D_9	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
D_{10}	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
D_{11}	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
D_{12}	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
D_{13}	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
D_{14}	*Rain*	*Mild*	*High*	*Strong*	*No*

Consider the attribute x_1=*Outlook*, where,

- Five (5) instances belong to v_{11}=*Sunny* (out of which 2 Yes and 3 No),
- Four (4) belong to v_{21}=*Overcast* (all of them belong to *Yes* class),
- Five (5) belong to v_{31}=*Rain* (three belong to *Yes* and two belong to *No*).

ID3 Algorithm

Entropy of a set S with respect to some attribute x_j:

We have,

x_1= Outlook {v_{11}=*Sunny*, v_{21}=*Overcast*, v_{31}=*Rain*}
x_2= Temperature {v_{12}=*Hot*, v_{22}=*Mild*, v_{32}=*Cool*}
x_3= Humidity {v_{13}=*High*, v_{23}=*Normal*}
x_4= Wind {v_{14}=*Weak*, v_{24}=*Strong*}.

$$Entropy(S, x_1) = \sum_{i=1}^{3} \frac{|s_i|}{|S|} Entropy(s_i) = \frac{|s_1|}{|S|} Entropy(s_1) + \frac{|s_2|}{|S|} Entropy(s_2) + \frac{|s_3|}{|S|} Entropy(s_3)$$

Where,

$$Entropy(s_1) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$$

// 5 *Sunny* (out of which 2 *Yes* and 3 *No*)

$$Entropy(s_2) = 0$$

// 4 *Overcast* (all of them belong to *Yes*)

$$Entropy(s_3) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$$

// 5 *Rain* (3 *Yes* and 2 *No*)

$$\text{Therefore, } Entropy(S, x_1) = \frac{5}{14} \times 0.97 + \frac{4}{14} \times 0.0 + \frac{5}{14} \times 0.97 = 0.693$$

Similarly, $Entropy(S, x_2) = 0.911$ (for *Temperature*),
$Entropy(S, x_3) = 0.788$ (for *Humidity*),
$Entropy(S, x_4) = 0.892$ (for *Wind*).

ID3 Algorithm

Information Gain of an attribute:

- The ***information gain*** or ***entropy reduction*** is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

The *information gain* of an attribute x_i is defined as: $Gain\ (S, x_i) = Entropy\ (S) - Entropy\ (S, x_i)$

Therefore, $Gain(S, x_1) = Entropy(S) - Entropy(S, x_1)$

$= 0.94 - 0.693 = 0.247$ //for *Outlook*

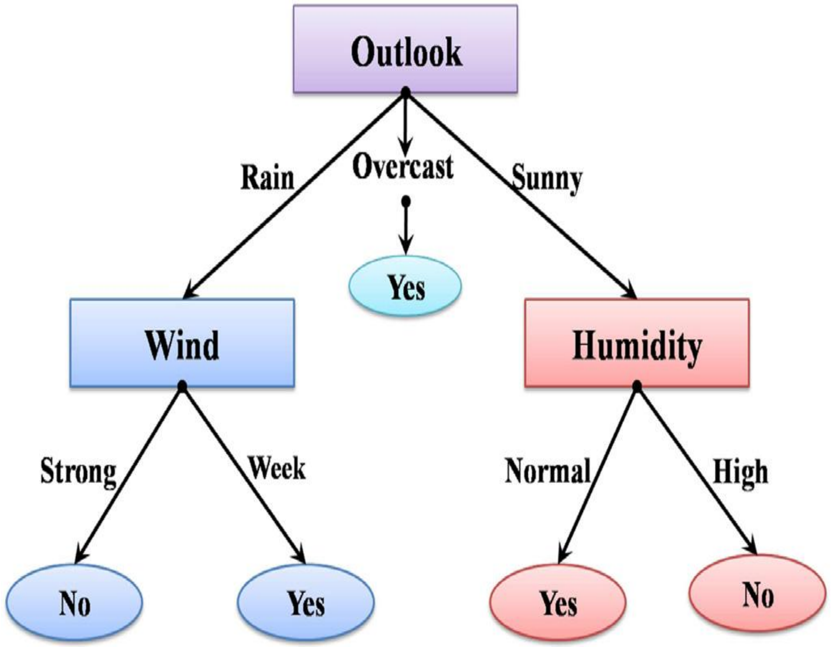
Similarly, $Gain(S, x_2) = Entropy(S) - Entropy(S, x_2) = 0.029$ // for *Temperature*

$Gain\ (S, x_3) = Entropy(S) - Entropy(S, x_3) = 0.152$ // for *Humidity*

$Gain\ (S, x_4) = Entropy(S) - Entropy(S, x_4) = 0.048$ // for *Wind*

ID3 Algorithm

- ID3 determines the information gain for each candidate attribute (i.e., ***Outlook* (x_1)**, ***Temperature* (x_2)**, ***Humidity* (x_3)**, and ***Wind* (x_4)**), then selects the one with highest information gain.



- The ***Outlook*** attribute provides the best prediction of the target attribute, ***PlayTennis***. Therefore, ***Outlook*** is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., ***Sunny, Overcast,*** and ***Rain***).

- Note that every example for which ***Outlook*** = ***Overcast*** is also a positive example of ***PlayTennis***. Therefore, this node of the tree becomes a leaf node with the classification ***PlayTennis*** = ***Yes***. In contrast, the descendants corresponding to ***Outlook*** = ***Sunny*** and ***Outlook*** = ***Rain*** still have nonzero entropy, and the decision tree will be further elaborated below these nodes.

ID3 Algorithm

- The process of selecting a new attribute and partitioning the training examples is now repeated for each descendant node, this time using only the training examples associated with that node.
- *Attributes that have been incorporated higher in the tree are excluded*, so that any given attribute can appear at most once along any path through the tree.
- This process continues for each new leaf node until either of two conditions is met:
  1) every attribute has already been included along this path through the tree, or
  2) the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

ID3 Algorithm

[D1, D2, ..., D14]
(9+,5-)

Here, $Gain(S, x_1) = \mathbf{0.247}$ // for *Outlook*
$Gain(S, x_2) = 0.029$ // for *Temperature*
$Gain(S, x_3) = 0.152$ // for *Humidity*
$Gain(S, x_4) = 0.048$ // for *Wind*

- Here, maximum Gain is for *Outlook*.
- Therefore, we have to break the tree based on the attribute *Outlook*.

ID3 Algorithm

Outlook

Sunny

Day	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D_1	Hot	High	Weak	No
D_2	Hot	High	Strong	No
D_8	Mild	High	Weak	No
D_9	Cool	Normal	Weak	Yes
D_{11}	Mild	Normal	Strong	Yes

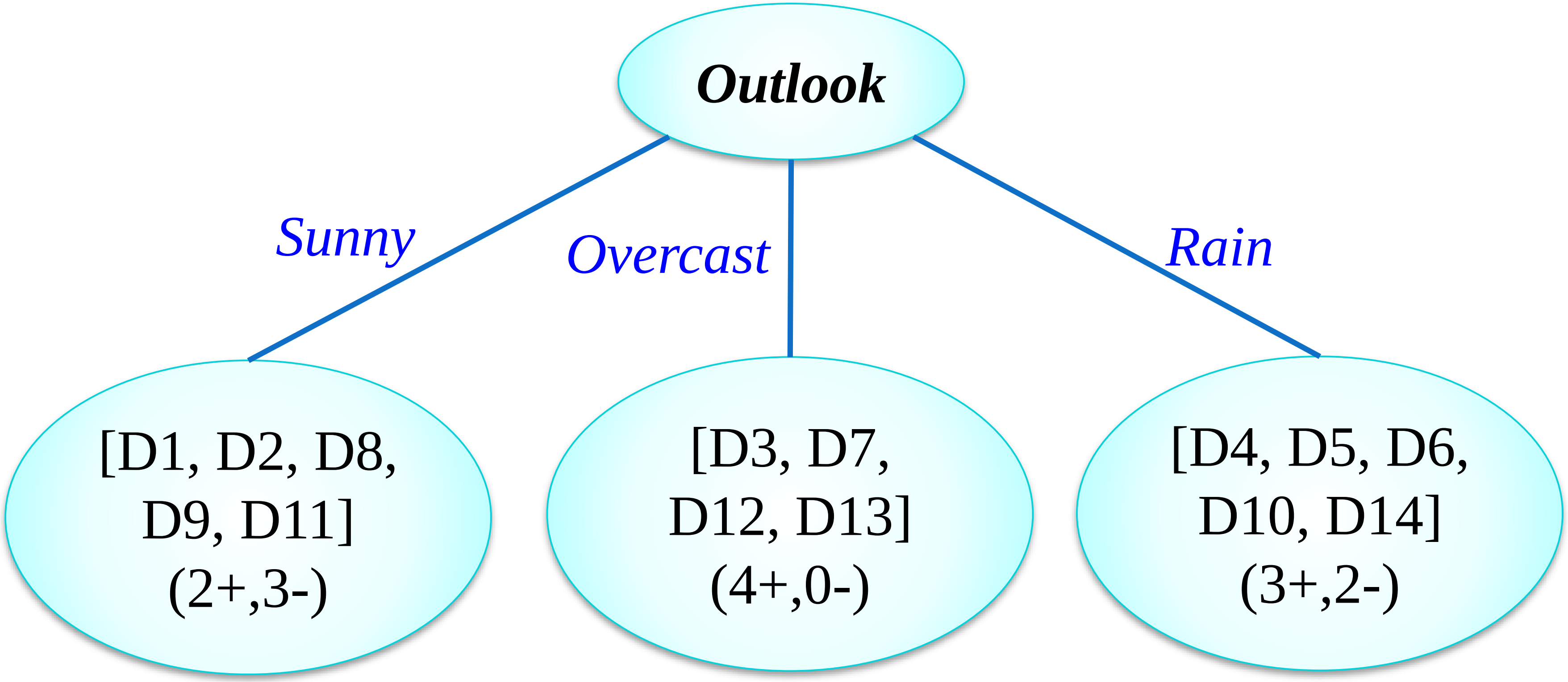
Overcast

Day	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D_3	Hot	High	Weak	Yes
D_7	Cool	Normal	Strong	Yes
D_{12}	Mild	High	Strong	Yes
D_{13}	Hot	Normal	Weak	Yes

Rain

Day	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D_4	Mild	High	Weak	Yes
D_5	Cool	Normal	Weak	Yes
D_6	Cool	Normal	Strong	No
D_{10}	Mild	Normal	Weak	Yes
D_{14}	Mild	High	Strong	No

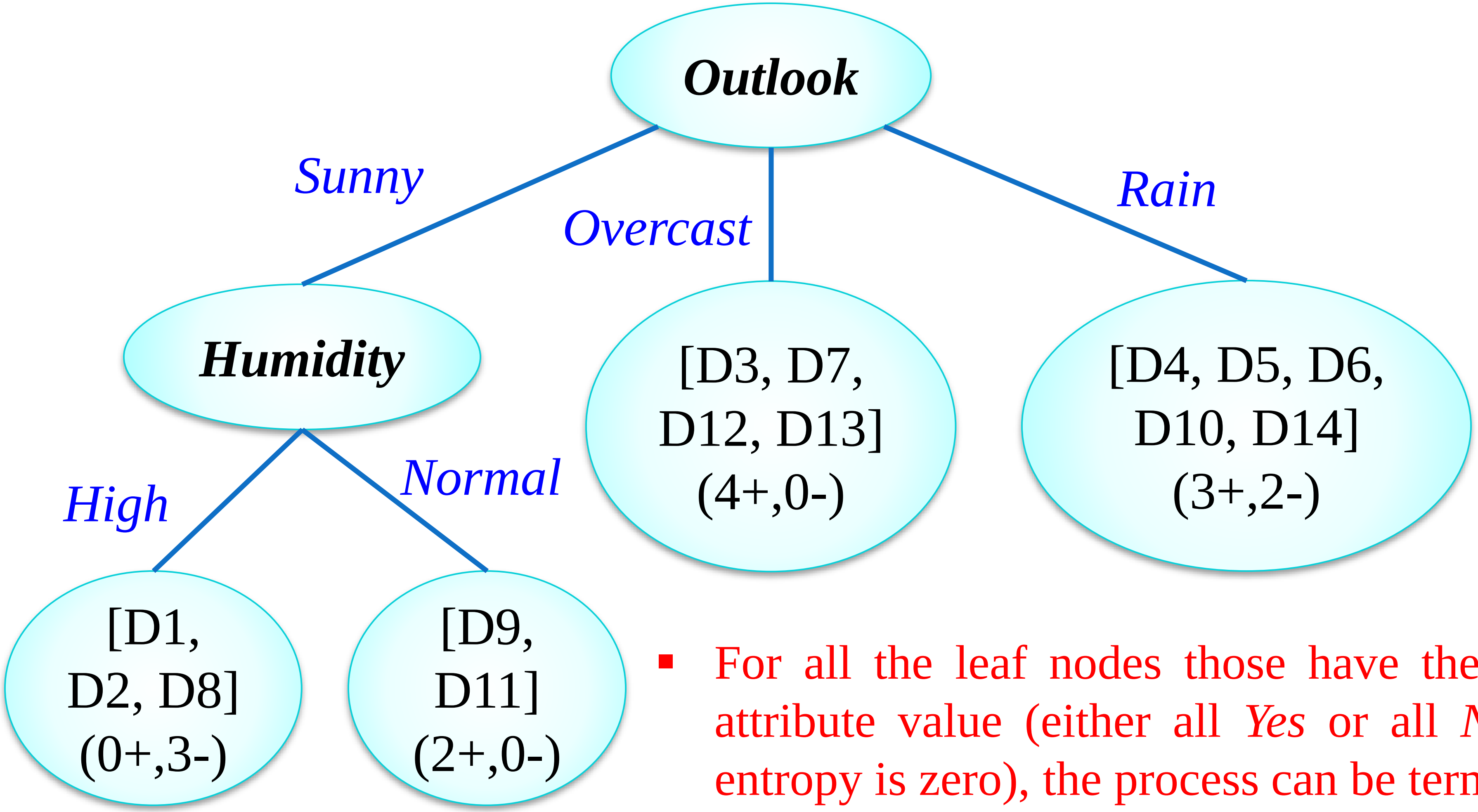
ID3 Algorithm



- Next, consider the node for *Sunny*. Which attribute??
  - *Gain*(*Sunny*, *Humidity*) = 0.970 – (3/5)0 – (2/5) 0 = **0.97**
  - *Gain*(*Sunny*, *Temperature*) = 0.57
  - *Gain*(*Sunny*, *Wind*) = 0.019

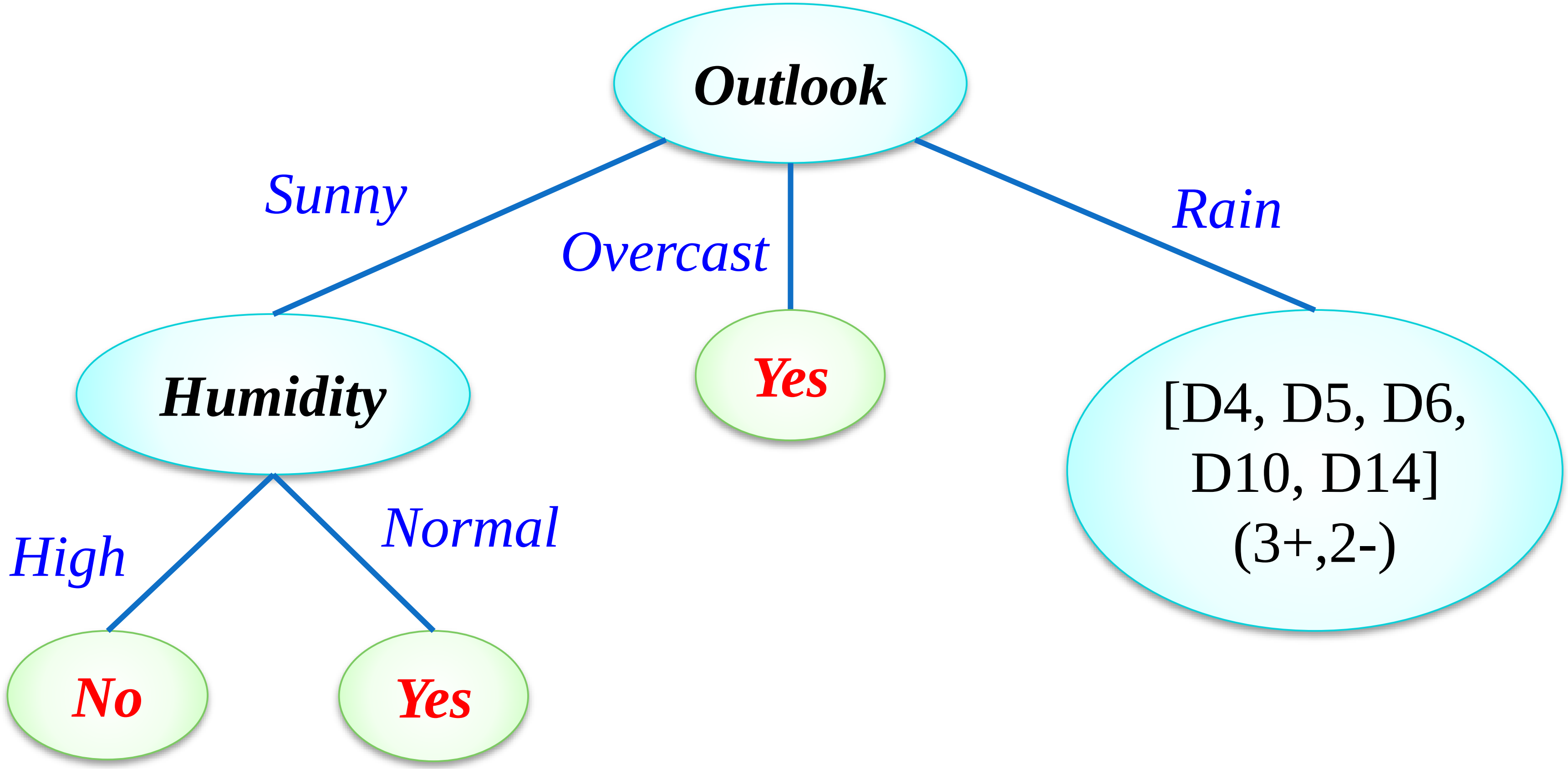
Split based on *Humidity*!!

ID3 Algorithm



- For all the leaf nodes those have the same target attribute value (either all *Yes* or all *No*, i.e., their entropy is zero), the process can be terminated.
- The target class for such nodes is the **majority** class.

ID3 Algorithm



- Next, consider the node towards *Rain*. Which attribute??
- You have to follow the same process.

ID3 Algorithm

Algorithm 1 : ID3 Algorithm

Input:(1)Classification data, S.

1: Calculate P_i and $Entropy(S)$. ▷ P_i be the probability of i^{th} class.
2: Compute $MaxGain(S, A_j) = max_{i \in S} Gain(S, A_i)$.
3: Branch for attribute A_j. ▷ As, A_j has maximum $Gain$.
4: Update S.
5: **if** $Entropy(S) = 0$ or no attribute remains to split **then**
6:     Terminate this branch.
7: **end if**
8: **if** All branches terminated **then**
9:     Terminate the Algorithm.
10: **else**
11:     Go to Line 1.
12: **end if**
13: Output: Decision tree

ID3 Algorithm

- There is a natural bias in the *information gain* measure that favors the attributes with many values over those with few values. As an extreme example, consider the attribute Date, which has a very large number of possible values (e.g., March 4, 1979).
- Date alone perfectly predicts the target attribute over the training data. It would have the highest information gain than any of the attributes. Thus, it would be selected as the decision attribute for the root node of the tree and lead to a (quite broad) tree of depth one, which perfectly classifies the training data.
- It is not a useful predictor despite the fact that it perfectly separates the training data.
- One way to avoid this difficulty is to select decision attributes based on some measure other than information gain. One alternative measure that has been used successfully is the *gain ratio* as used in C4.5 Algorithm, a successor and refinement of ID3.

C4.5 Algorithm

- The *Gain* (in ID3) has a strong bias in favor of the attributes with large number of values.
- Such attributes will get selected at root itself and may lead to all leaf nodes, resulting in too simple hypothesis model unable to compute the structure of the data.
- C4.5 is a successor of ID3. C4.5 applies a kind of normalization to information gain using a split information value defined as:

$$SplitInfo(S, x_j) = -\sum_{i=1}^{d} \frac{|s_i|}{|S|} \log_2 \frac{|s_i|}{|S|}$$

The gain ratio is defined as: $GainRatio(S, x_j) = \dfrac{Gain(S, x_j)}{SplitInfo(S, x_j)}$

C4.5 Algorithm

Returning to the *PlayTennis* table, x_1=*Outlook*, x_2=*Temperature*, x_3=*Humidity* and x_4=*Wind*. *Outlook* splits the dataset into three subsets of size 5, 4 and 5 and thus the *SplitInfo* can be calculated as:

$$SplitInfo(S,x_1) = -\sum_{i=1}^{3}\frac{|s_i|}{|S|}\log_2\frac{|s_i|}{|S|} = -\frac{5}{14}\log\frac{5}{14} - \frac{4}{14}\log\frac{4}{14} - \frac{5}{14}\log\frac{5}{14} = 1.577$$

Therefore, the *GainRatio* can be calculated as,

$$GainRatio(S,x_1) = \frac{Gain(S,x_1)}{SplitInfo(S,x_1)} = \frac{0.247}{1.577} = 0.156$$

The overall calculation can be summarized as follow:

$$\begin{array}{ll} OutLook & : Gain(S,x_1) = 0.247,\ \ SplitInfo(S,x_1) = 1.577,\ \ GainRatio(S,x_1) = 0.156 \\ Temperature & : Gain(S,x_2) = 0.029,\ \ SplitInfo(S,x_2) = 1.362,\ \ GainRatio(S,x_2) = 0.019 \\ Humidity & : Gain(S,x_3) = 0.152,\ \ SplitInfo(S,x_3) = 1.00,\ \ GainRatio(S,x_3) = 0.152 \\ Wind & : Gain(S,x_4) = 0.048,\ \ SplitInfo(S,x_4) = 0.985,\ \ GainRatio(S,x_4) = 0.049 \end{array}$$

C4.5 Algorithm

- The attribute with the maximum *GainRatio* is selected as splitting attribute.
- *Outlook* still comes out on top, but *Humidity* now is a much closer contender because it splits the data into two subsets instead of three.
- Similar to ID3 Algorithm, C4.5 also split the data set based on impurity measure. C4.5 uses *GainRatio* which is in contrast to *Gain* in ID3.
- One practical issue that arises in using *GainRatio* is that the denominator can be zero or very small when $s_i \approx S$ for some s_i.
- To avoid such problem, we can adopt some heuristic such as first calculating the *Gain* of each attribute, then applying the *GainRatio* test only considering those attributes with above average *Gain*.

CART Algorithm

- Another popular splitting criterion is named Gini. It is being used in CART.

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2$$

Where, $p_i = \frac{freq(y_i, S)}{|S|}$

where C be the number of class, p_i is the probability of an instance belongs to the class y_i.

- As example, Table 1 has two classes, *Yes* and *No*. p_1 be the probability of an instance belongs to class *Yes* and p_2 be the probability of an instance belongs to class *No*.
- Therefore, $p_1 = \frac{9}{14}$ and $p_2 = \frac{5}{14}$.

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2 = 1 - (\tfrac{9}{14})^2 - (\tfrac{5}{14})^2 = 1 - 0.413 - 0.127 = 0.45918$$

CART Algorithm

- Gini index considers a *binary split* for each attribute.
- Consider the case where x_j is continuous-valued attribute having d_j distinct values.
- We have to split the value and 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 is taken as final split-point. The Gini index of an attribute is calculated as:

$$\text{Ginini Index } \Delta Gini(x_j) = Gini(S) - Gini(S, x_j)$$

- Where, *Gini*(*S*) can be calculated as discussed in the previous slide. *Gini*(*S*, x_j) can be calculated as :

$$Gini(S, x_j) = \frac{|s_1|}{|S|} Gini(s_1) + \frac{|s_2|}{|S|} Gini(s_2)$$

CART Algorithm

- The attribute that maximizes the reduction in impurity (or equivalently, has the minimum Gini index) is selected as the splitting attribute.

- Where, for a possible split of x_j, s_1 is the number of tuples in S satisfying $x_j \leq$ *split-point* and s_2 is the number of tuples satisfying $x_j >$ *split-point*.

- Then one of these two parts (s_1 and s_2) is divided in a similar manner by choosing a variable again and a split-value for the variable.

- The process is continued till we get a *pure* leaf node.

CART Algorithm

- Consider the case where, x_j is a categorical attribute, example *Outlook* with categories {*Sunny*, *Overcast*, *Rain*}.
- To determine the best binary split, we examine all possible subsets that can be formed using the categories of *Outlook*: {*Sunny, Overcast, Rain*}, {*Sunny, Overcast*}, {*Sunny, Rain*}, {*Overcast, Rain*}, {*Sunny*}, {*Overcast*}, {*Rain*}, {}.
- We exclude the power set {*Sunny*, *Overcast*, *Rain*} and the empty set {} from consideration since, conceptually, they do not represent a split.
- Therefore, there are $(2^d - 2)/2$ possible ways to form two partitions of dataset S, based on the binary splits on x_j having d categorical values.
- Each of the possible binary splits is considered. **The split that gives the minimum Gini index is selected as splitting subset.**
- **The CART approach restricts the split to binary values for both continuous and categorical attributes. Thus, CART produces binary tree.**

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

$S^{(i)}$	***Grade***	***Interactiveness***	***Practical***	***Communication***	***Job Offer***
1	A	*Yes*	*Very Good*	*Good*	*Yes*
2	B	*No*	*Good*	*Moderate*	*Yes*
3	A	*No*	*Average*	*Poor*	*No*
4	C	*No*	*Average*	*Good*	*No*
5	B	*Yes*	*Good*	*Moderate*	*Yes*
6	A	*Yes*	*Good*	*Moderate*	*Yes*
7	C	*Yes*	*Good*	*Poor*	*No*
8	A	*No*	*Very Good*	*Good*	*Yes*
9	B	*Yes*	*Good*	*Good*	*Yes*
10	B	*Yes*	*Average*	*Good*	*Yes*

- Out of 10 instances, 7 *Yes* and 3 *No* class.

Therefore, $Gini(S) = 1 - \sum_{i=1}^{C} p_i^2 = 1 - (\frac{7}{10})^2 - (\frac{3}{10})^2 = 1 - 0.49 - 0.09 = 0.42$

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

- Compute Gini index for the categorical attribute *Grade*.

Grade (*X*)	*Job Offer: Yes*	*Job Offer: No*
A	3	1
B	4	0
C	0	2

➢ The possible subsets that can be formed: {*A*, *B*}, {*A*, *C*}, {*B*, *C*}, {*A*}, {*B*}, {*C*}.

➢ Now, we have to calculate the following Gini for all the possible splits as:

$$Gini(S, \{\{A, B\}, C\}) = \frac{|\{A, B\}|}{|S|}.Gini(\{A, B\}) + \frac{|\{C\}|}{|S|}.Gini(\{C\})$$

$$Gini(S, \{\{A, C\}, B\}) = \frac{|\{A, C\}|}{|S|}.Gini(\{A, C\}) + \frac{|\{B\}|}{|S|}.Gini(\{B\})$$

$$Gini(S, \{\{B, C\}, A\}) = \frac{|\{B, C\}|}{|S|}.Gini(\{B, C\}) + \frac{|\{A\}|}{|S|}.Gini(\{A\})$$

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

- Compute Gini index for the categorical attribute *Grade*.

Grade (X)	*Job Offer: Yes*	*Job Offer: No*
A	3	1
B	4	0
C	0	2

➢ The possible subsets that can be formed: {*A*, *B*}, {*A*, *C*}, {*B*, *C*}, {*A*}, {*B*}, {*C*}.

$$Gini(\{A\}) = 1 - (\tfrac{3}{4})^2 - (\tfrac{1}{4})^2 = 0.375$$

Similarly, $Gini(\{B\}) = 0$

$$Gini(\{C\}) = 1 - (\tfrac{0}{2})^2 - (\tfrac{2}{2})^2 = 0$$

$$Gini(\{A, B\}) = 1 - (\tfrac{7}{8})^2 - (\tfrac{1}{8})^2 = 1 - 0.7806 = 0.2194$$

$$Gini(\{A, C\}) = 1 - (\tfrac{3}{6})^2 - (\tfrac{3}{6})^2 = 0.5$$

$$Gini(\{B, C\}) = 1 - (\tfrac{4}{6})^2 - (\tfrac{2}{6})^2 = 0.445$$

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

- Compute Gini index for the categorical attribute *Grade*.

Grade (X)	*Job Offer: Yes*	*Job Offer: No*
A	3	1
B	4	0
C	0	2

➢ The possible subsets that can be formed: *{A, B}, {A, C}, {B, C}, {A}, {B}, {C}*.

$$Gini(\{A\}) = 1 - (\tfrac{3}{4})^2 - (\tfrac{1}{4})^2 = 0.375$$

Similarly, $Gini(\{B\}) = 0$

$$Gini(\{C\}) = 1 - (\tfrac{0}{2})^2 - (\tfrac{2}{2})^2 = 0$$

$$Gini(\{A,B\}) = 1 - (\tfrac{7}{8})^2 - (\tfrac{1}{8})^2 = 1 - 0.7806 = 0.2194$$

$$Gini(\{A,C\}) = 1 - (\tfrac{3}{6})^2 - (\tfrac{3}{6})^2 = 0.5$$

$$Gini(\{B,C\}) = 1 - (\tfrac{4}{6})^2 - (\tfrac{2}{6})^2 = 0.445$$

$$Gini(S,\{\{A,B\},C\}) = \frac{8}{10} \times 0.2194 + \frac{2}{10} \times 0 = 0.1755$$

$$Gini(S,\{\{A,C\},B\}) = \frac{6}{10} \times 0.5 + \frac{4}{10} \times 0 = 0.3$$

$$Gini(S,\{\{B,C\},A\}) = \frac{6}{10} \times 0.445 + \frac{4}{10} \times 0.375 = 0.417$$

The partition: {{*A*, *B*}, {*C*}} has the lowest Gini = 0.1755.

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

- Compute Gini index for the categorical attribute *Grade*.

Grade (X)	*Job Offer: Yes*	*Job Offer: No*
A	3	1
B	4	0
C	0	2

- The possible subsets that can be formed: $\{A, B\}, \{A, C\}, \{B, C\}, \{A\}, \{B\}, \{C\}$.

- The split: $\{\{A, B\}, \{C\}\}$ has the lowest Gini = 0.1755.
- Therefore, $\Delta Gini(\{\{A, B\}, C\}) = Gini(S) - 0.1755 = 0.42 - 0.1755 = 0.2445$
- Now, we have to repeat the same process for the remaining attributes.

Attribute (X_j)	$Gini(S)$	$Gini\ (S, X_j)$	$\Delta Gini(X_j)$	Split
Grade	0.42	0.1755	**0.2445**	$\{\{A, B\}, \{C\}\}$
Interactiveness		0.368	0.052	{Yes, No}
Practical		0.3054	0.1146	{{Very Good, Good}, Average}
Communication		0.1755	**0.2445**	{{Good, Moderate}, Poor}

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

- *Grade* and *Communication* have the highest Gini value. Let us choose *Grade* as root of the tree.

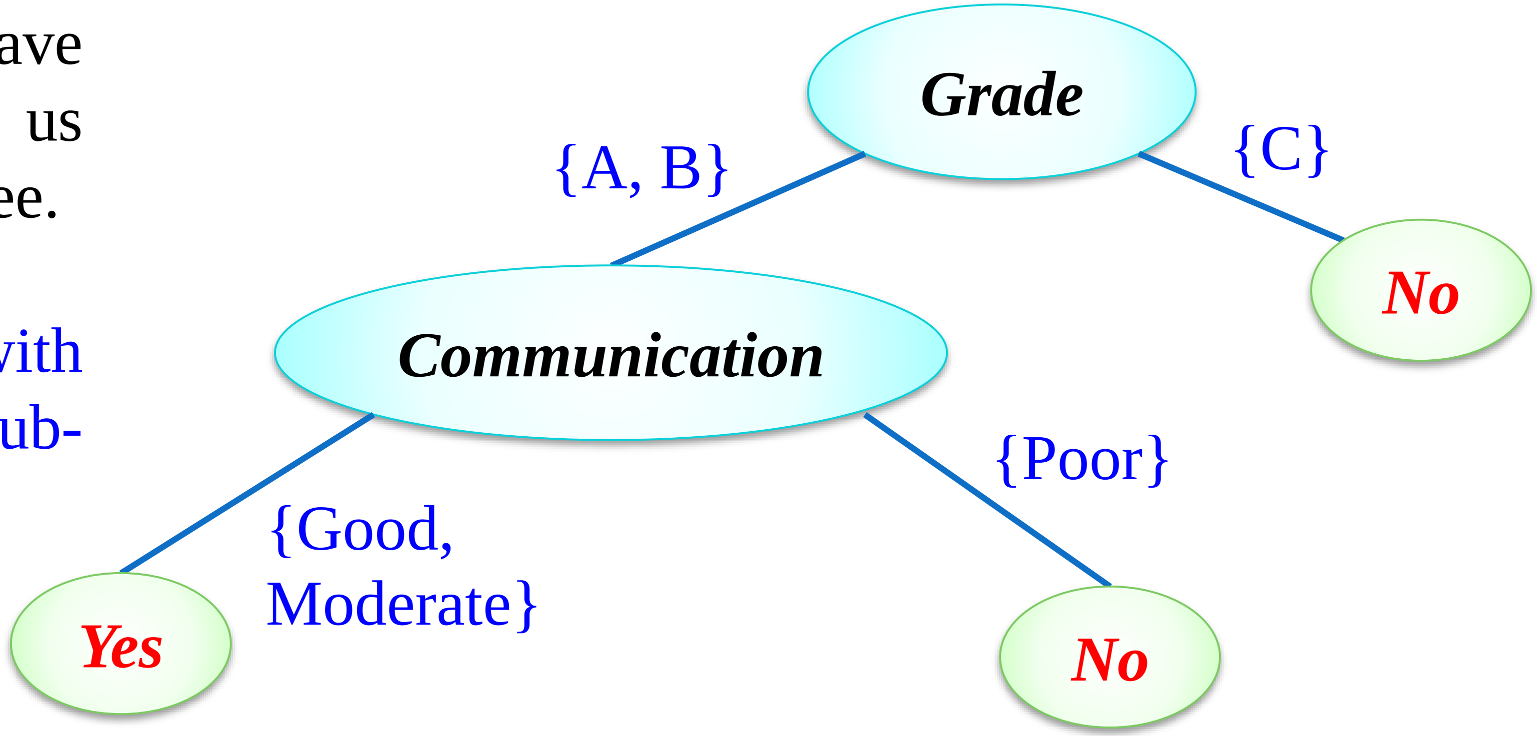
- Now we have to continue with the remaining dataset of left sub-tree.

$S^{(i)}$	***Interactiveness***	***Practical***	***Communication***	***Job Offer***
1	*Yes*	*Very Good*	*Good*	*Yes*
2	*No*	*Good*	*Moderate*	*Yes*
3	*No*	*Average*	*Poor*	*No*
5	*Yes*	*Good*	*Moderate*	*Yes*
6	*Yes*	*Good*	*Moderate*	*Yes*
8	*No*	*Very Good*	*Good*	*Yes*
9	*Yes*	*Good*	*Good*	*Yes*
10	*Yes*	*Average*	*Good*	*Yes*

CART Algorithm

Exercise: Consider the following datasets. Build a decision tree using CART algorithm.

- *Grade* and *Communication* have the highest Gini value. Let us choose *Grade* as root of the tree.
- Now we have to continue with the remaining dataset of left sub-tree.
- Next, *Communication* is selected as the root of the left sub-tree.



- *Final Tree using CART*

Incorporating Continuous-Valued Attributes

- Our initial definition of ID3 and C4.5 is restricted to attributes that take on a discrete set of values.
- This restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree.
- Dynamically define new discrete valued attributes that partition the continuous attribute value into a discrete set of intervals.
- For an attribute *A* that is continuous-valued, the algorithm can dynamically create a new Boolean attribute *A*, that is true if $A < c$ and false otherwise.
- The only question is how to select the best value for the threshold *c*.

Incorporating Continuous-Valued Attributes

- As an example, consider the continuous-valued attribute ***Temperature*** and its corresponding target attribute ***PlayTennis*** as follow.

Temperature	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

- What threshold-based boolean attribute should be defined based on Temperature? Clearly, we would like to pick a threshold, *c*, that produces the greatest information gain.
- First sorting the examples, then identify adjacent examples that differ in their target classification. Now, we can generate a set of candidate thresholds midway between the corresponding values.
- These candidate thresholds can then be evaluated by computing the information gain associated with each.

Incorporating Continuous-Valued Attributes

Temperature	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

- In the current example, there are two candidate thresholds, corresponding to the values of Temperature at which the value of *PlayTennis* changes: (48 + 60)/2, and (80 + 90)/2. The information gain can then be computed for each of the candidate attributes, *Temperature*>54 and *Temperature*>85, and the best can be selected (*Temperature*>85).

- One of the main problem with this selection criterion is that it is relatively expensive. It must be evaluated *N*-1 times for each attribute (assuming *N* samples have distinct values). Machine learning algorithms are designed for large set of training data, so is typically very large.

Incorporating Continuous-Valued Attributes

- For each continuous valued attribute x_j, we select the best cut-point T_{xj} from its range of values by evaluating every candidate cut point in the range of values.

- The examples are first sorted by increasing value of the attribute x_j, and mid-point between each successive pair of values in the sorted sequence is evaluated as potential cut-point.

- A *cut-point* T_{xj} for x_j will partition the patterns into two subsets satisfying the conditions $V_{xj} \leq T_{xj}$ and $V_{xj} > T_{xj}$ respectively, thereby creating a binary discretization.

Incorporating Continuous-Valued Attributes

- **Cut-Point Optimization:** Fayyad and Irani have proved that regardless of how many classes there are (binary or multiclass problems) and how they are distributed, the cut-point will always occur on the boundary between two classes.
  Example: Consider the temperature data (total 14 instances).

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

- There are 11 possible candidate cut-points (64, 65, 68, 69, 70, 71, 72, 75, 80, 81, 83).

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

- As per the cut point optimization result, the number of optimized candidate cut points is 8 and these are{64, 65, 70, 71, 72, 75, 80, 83}.

Incorporating Continuous-Valued Attributes

- As per the cut point optimization result, the number of optimized candidate cut points is 8 and these are {64, 65, 70, 71, 72, 75, 80, 83}.

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

- Total 14 instances with 9 *Yes* and 5 *No*.

- Now we have to calculate entropy for each candidate cut-points one by one.
- Start from c = 64. One instance is equal or less than c = 64. 13 instances are greater than 64.
- Entropy can be calculated as:

$$Entropy(S, c = 64) = \frac{|s_1|}{|S|}Entropy(s_1) + \frac{|s_2|}{|S|}Entropy(s_2)$$

$$\text{where, } |S| = 14, |s_1| = 13 \text{ and } |s_2| = 1.$$

$$Entropy(s_1) = -\frac{8}{13}\log_2\frac{8}{13} - \frac{5}{13}\log_2\frac{5}{13} = 0.9612$$

$$Entropy(s_2) = -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$$

$$\text{Therefore, } Entropy(S, c = 64) = \frac{13}{13} \times 0.9612 + \frac{1}{14} \times 0 = 0.8925$$

Incorporating Continuous-Valued Attributes

- As per the cut point optimization result, the number of optimized candidate cut points is 8 and these are {64, 65, 70, 71, 72, 75, 80, 83}.

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

- Total 14 instances with 9 *Yes* and 5 *No*.

  - Now we have to calculate entropy for each candidate cut-points one by one.
  - Similarly, *Entropy*(*S*, *c* = 83) can also be calculated as follows.

$$Entropy(S, c = 83) = \frac{13}{14}\left(-\frac{9}{13}\log_2\frac{9}{13} - \frac{4}{13}\log_2\frac{4}{13}\right) + \frac{1}{14}\times 0,$$

$$= \frac{13}{14}(0.3672 + 0.5232) = 0.8268$$

- After calculating entropy for all other cut-points, it can be observed that *Entropy*(*S*, *c* = 83) = 0.8268 is the minimum than all other cut-points. Hence, maximum information gain.
- Therefore, *c* = 83 be the final cut point.

Decision Tree

Features	ID3	C4.5	CART
Impurity Measure	Entropy, Information Gain	Entropy, Information Gain, SplitInfo, GainRatio	Gini
Degree of Tree	Any degree. Depends on the number of categorical data of the attribute.	Any degree. Depends on the number of categorical data of the attribute.	Binary tree
Data set	Suitable for categorical attributes	Suitable for categorical attributes	Continuous or nominal categorical attributes
Disadvantages	The *information gain* favors the attributes with many values over those with few values.	May not be suitable for the data set with many continuous attributes.	May not be suitable for the data set with many categorical attributes.

Decision Tree

- The simple strategies face difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that ***overfit the training examples.***

  Definition (Overfit): Given a hypothesis space H, a hypothesis $h \in H$ is said to *overfit* the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

- *Overfitting* is a significant practical difficulty for decision tree learning and many other learning methods.
- The decision tree uses a hierarchical approach for supervised learning.
- The decision tree is a non-linear classifier.

Regression Tree

- A *regression tree* is constructed in almost the same manner as a classification tree, except that the impurity measure that is appropriate for classification is replaced by a measure appropriate for regression. In regression, the goodness of a split is measured by the mean square error from the estimated value.

- A regression tree can only generate as many distinct output values as there are leaf nodes in the tree.

Assignments

Assignment DTR1: Give decision trees to represent the following Boolean functions:

a) $A \wedge \neg B$ b) $A \vee [B \wedge C]$ c) A *XOR* B

d) $[A \wedge B] \vee [C \wedge D]$ e) $[A \wedge \neg B] \vee \neg C$ f) $\neg A \vee [B$ *XOR* $C]$

Assignment DTR2: Consider the following set of training examples and

a) What is the entropy of this collection of training examples?

b) What is the information gain of a_1 and a_2 relative to these training examples?

c) Build the decision tree using ID3 algorithm.

Instance	a_1	a_2	*Classification*
1	x	y	+
2	x	u	+
3	z	u	-
4	z	v	+
5	x	v	-
6	z	y	-

Assignments

Assignment DTR3: Consider the following set of training examples and,

a) What is the entropy of this collection of training examples?

b) Build the decision tree using ID3 algorithm.

$S^{(i)}$	a_1	a_2	a_3	a_4	*Class*
1	*x*	*u*	*n*	*e*	+
2	*x*	*u*	*p*	*f*	+
3	*x*	*u*	*n*	*g*	+
4	*y*	*u*	*n*	*e*	+
5	*y*	*v*	*n*	*f*	-
6	*x*	*v*	*n*	*e*	+
7	*x*	*u*	*p*	*e*	-
8	*y*	*v*	*m*	*f*	+
9	*x*	*u*	*n*	*f*	+
10	*x*	*w*	*p*	*f*	+
11	*y*	*w*	*n*	*f*	-
12	*x*	*w*	*n*	*g*	+

Assignments

Assignment DTR4: Consider the following training example. Apply ID3, C4.5, and CART and show the tree.

Day	*Outlook*	*Temperature* (ºF)	*Humidity* (%)	*Wind*	*PlayTennis*
D_1	*Sunny*	85	85	*Weak*	*No*
D_2	*Sunny*	80	90	*Strong*	*No*
D_3	*Overcast*	83	86	*Weak*	*Yes*
D_4	*Rain*	70	96	*Weak*	*Yes*
D_5	*Rain*	68	80	*Weak*	*Yes*
D_6	*Rain*	65	70	*Strong*	*No*
D_7	*Overcast*	64	65	*Strong*	*Yes*
D_8	*Sunny*	72	95	*Weak*	*No*
D_9	*Sunny*	69	70	*Weak*	*Yes*
D_{10}	*Rain*	75	80	*Weak*	*Yes*
D_{11}	*Sunny*	75	70	*Strong*	*Yes*
D_{12}	*Overcast*	72	90	*Strong*	*Yes*
D_{13}	*Overcast*	81	75	*Weak*	*Yes*
D_{14}	*Rain*	71	91	*Strong*	*No*

<u>Books:</u>

1. “Machine Learning” by Tom Mitchell, McGraw Hill.
2. “Applied Machine Learning” by M.Gopal, McGraw Hill.