Machine Learning
ICT-4261

By-
Dr. Jesmin Akhter
Professor
Institute of Information Technology
Jahangirnagar University

Contents

The course will mainly cover the following topics:

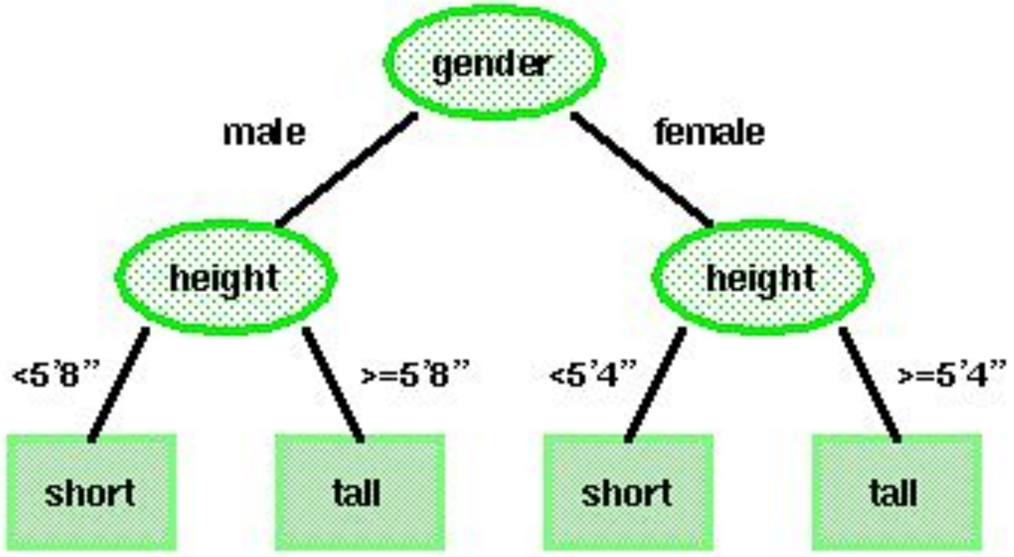
- ✔ A Gentle Introduction to Machine Learning
- ✔ Important Elements in Machine Learning
- ✔ Linear Regression
- ✔ Logistic Regression
- ✔ Naive Bayes
- ✔ Support Vector Machines
- ✔ Decision Trees and Ensemble Learning
- ✔ Clustering Fundamentals
- ✔ Hierarchical Clustering
- ✔ Neural Networks and Deep Learning
- ✔ Unsupervised Learning

Outline

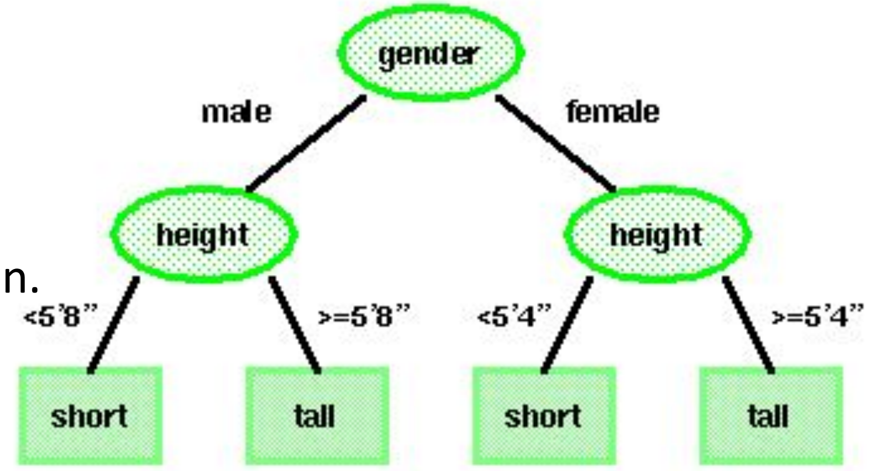
✔ Decision Trees

Decision Trees

- ✔ A *decision tree* is an approach to predictive analysis that can help you make decisions.
- ✔ A decision tree consists of nodes and leaves, with each leaf denoting a class.
- ✔ For example Classes (tall or short) are the outputs of the tree.
- ✔ Attributes (gender and height) are a set of features that describe the data.
- ✔ The input data consists of values of the different attributes. Using these attribute values, the decision tree generates a class as the output for each input data.



Basic Principles

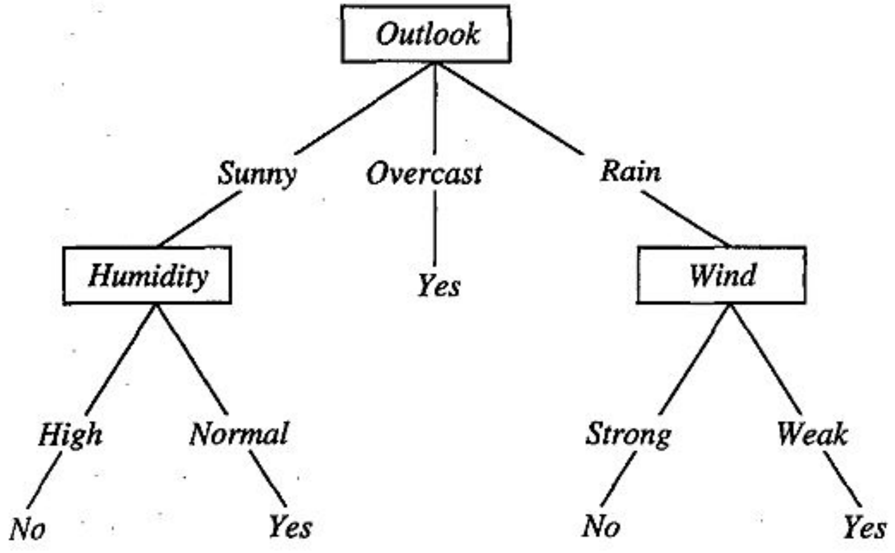


- ✔ The top, or first node, is called the root node.
- ✔ The last level of nodes are the leaf nodes and contain the final classification.
- ✔ The intermediate nodes are the descendant or "hidden" layers.
- ✔ Binary trees, are the most popular type of tree.
  However, M-ary trees (M branches at each node) are possible.
- ✔ Decision trees attempt to classify a pattern through a sequence of questions. For example, attributes such as gender and height can be used to classify people as short or tall. But the best threshold for height is gender dependent. In a binary tree, by convention if the answer to a question is "yes", the left branch is selected.
- ✔ Key questions include how to grow the tree, how to stop growing, and how to prune the tree to increase generalization.

DECISION TREE REPRESENTATION

- ✔ A decision tree can contain categorical data (YES/NO) as well as numeric data.
- ✔ Figure illustrates a typical learned decision tree. This **decision tree classifies** Saturday mornings according to **whether they are suitable for playing tennis**.
- ✔ For example, the instance (Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong) would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance (i.e., the tree predicts that PlayTennis = no).
- ✔ The decision tree shown in Figure corresponds to the expression

$$
\begin{aligned}
& (Outlook = Sunny \wedge Humidity = Normal) \\
\vee \quad & (Outlook = Overcast) \\
\vee \quad & (Outlook = Rain \wedge Wind = Weak)
\end{aligned}
$$



How do Decision Trees work?

- ✔ The decision of making strategic splits heavily affects a tree’s accuracy. The decision criteria are different for classification and regression trees.
- ✔ Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes.
- ✔ The decision tree splits the nodes on all available variables
- ✔ The algorithm selection is also based on the type of target variables. Some algorithms used in Decision Trees:
- ✔ **ID3** → (extension of D3)
  C4.5 → (successor of ID3)
  CART → (Classification And Regression Tree)
  CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
  MARS → (multivariate adaptive regression splines)

How do Decision Trees work?

- ✔ The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking. A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment.
- ✔ **Steps in ID3 algorithm:**
- ✔ It begins with the original set S as the root node.
- ✔ On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates **Entropy(H)** and **Information gain(IG)** of this attribute.
- ✔ It then selects the attribute which has the smallest Entropy or Largest Information gain.
- ✔ The set S is then split by the selected attribute to produce a subset of the data.
- ✔ The algorithm continues to recur on each subset, considering only attributes never selected before.

Attribute Selection Measures

- ✔ If the dataset consists of **N** attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy.
- ✔ For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some *criteria* like :
  - — **Entropy**,
  - — **Information gain,**
  - — **Gini index,**
  - — **Gain Ratio,**
  - — **Reduction in Variance**
  - — **Chi-Square**
- ✔ These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order
  - – Attribute with a high value(in case of information gain) is placed at the root.
  - – While using Information Gain as a criterion, we assume attributes to be categorical, and
  - – For the Gini index, attributes are assumed to be continuous.

Entropy

- ✔ Measures the level of impurity in a group of examples. Impurity is the degree of randomness; it tells how random our data is. if the target attribute can take on n different values, then the entropy of S relative to this n-wise classification is defined as:

$$Entropy = -\sum_{i=1}^{n} p_i log_b p_i$$

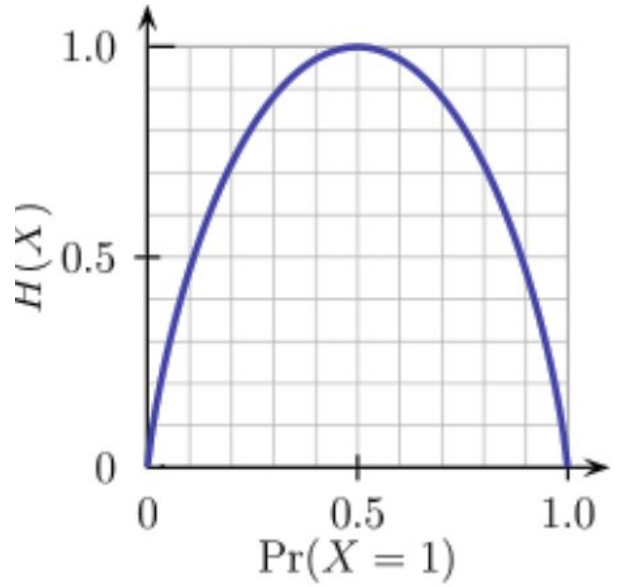
Where p_i is the proportion of S belonging to class i. Note the logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits.

- ✔ Now we can rewrite:

$$\textit{Entropy} = -(p_1 * \log_2(p_1) + p_2 * \log_2(p_2) + .. + p_n * \log_2(p_n))$$

Entropy

- ✔ Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.



- ✔ From the above graph, it is quite evident that the entropy H(X) is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance if perfectly determining the outcome.

Entropy-Example 01

- ✔ A **pure sub-split** means that either you should be getting "yes", or you should be getting "no".
- ✔ Suppose a *feature* as 8 "yes" and 4 "no" initially, after the first split the left node *gets 5 'yes' and 2 'no'* whereas right node *gets 3 'yes' and 2 'no'.*
- ✔ We see here the split is not pure, why? Because we can still see some negative classes in both the nodes. In order to make a decision tree, we need to calculate the impurity of each split, and when the purity is 100%, we make it as a leaf node.
- ✔ To check the impurity of feature 2 and feature 3 we will take the help for Entropy formula.

$$\Rightarrow -\left(\frac{5}{7}\right)log_2\left(\frac{5}{7}\right) - \left(\frac{2}{7}\right)log_2\left(\frac{2}{7}\right)$$

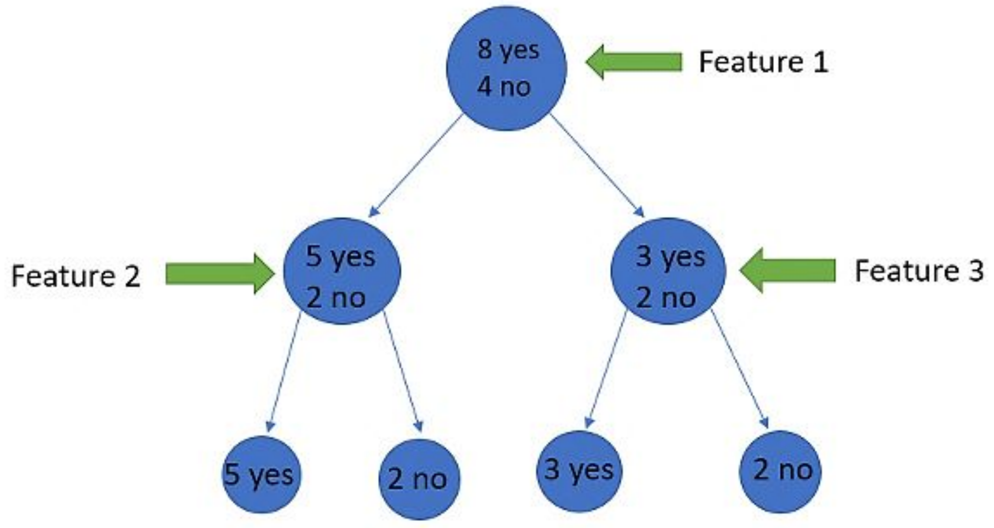
$$\Rightarrow -(0.71 * -0.49) - (0.28 * -1.83)$$

$$\Rightarrow -(-0.34) - (-0.51)$$

$$\Rightarrow 0.34 + 0.51$$

$$\Rightarrow 0.85$$

$$\log_2(x) = \ln(x) \div \ln(2)$$



Entropy-Example 01

- ✔ For feature 3,

$$\Rightarrow -\left(\frac{3}{5}\right)log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)log_2\left(\frac{2}{5}\right)$$

$$\Rightarrow -(0.6*-0.73)-(0.4*-1.32)$$

$$\Rightarrow -(-0.438)-(-0.528)$$

$$\Rightarrow 0.438+0.528$$

$$\Rightarrow 0.966$$

- ✔ We can clearly see from the tree itself that left node has low entropy or more purity than right node since left node has a greater number of "yes" and it is easy to decide here.

Information gain

- ✔ Information gain or **IG** is a statistical property that measures how well a given attribute separates the training examples according to their target classification.
- ✔ Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.
- ✔ Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node.
- ✔ Information gain is used in both classification and regression decision trees. **In classification, entropy is used as a measure of impurity, while in regression, variance is used as a measure of impurity**

Information gain

- ✔ Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

- ✔ In a much simpler way, we can conclude that:

$$Information\ Gain = Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

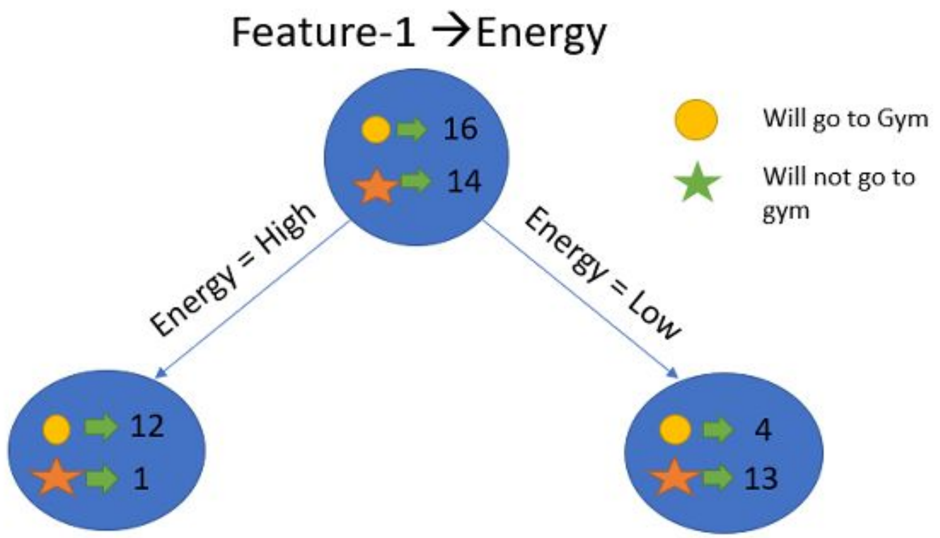
- ✔ Where "before" is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

Information gain

- ✔ It is just entropy of the full dataset – entropy of the dataset given some feature.
- ✔ To understand this better let's consider an example:
  - Suppose our entire population has a total of 30 instances. The dataset is to predict whether the person will go to the gym or not. Let's say 16 people go to the gym and 14 people don't
  - Now we have two features to predict whether he/she will go to the gym or not.
  - Feature 1 is **"Energy"** which takes two values *"high" and "low"*
  - Feature 2 is **"Motivation"** which takes 3 values *"No motivation", "Neutral" and "Highly motivated".*

Information gain-Example 01

- ✔ Let's see how our decision tree will be made using these 2 features. We'll use information gain to decide which feature should be the root node and which feature should be placed after the split.
- ✔



Let's calculate the entropy

$$E(Parent) = -\left(\frac{16}{30}\right)\log_2\left(\frac{16}{30}\right) - \left(\frac{14}{30}\right)\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Parent|Energy = \text{"high"}) = -\left(\frac{12}{13}\right)\log_2\left(\frac{12}{13}\right) - \left(\frac{1}{13}\right)\log_2\left(\frac{1}{13}\right) \approx 0.39$$

$$E(Parent|Energy = \text{"low"}) = -\left(\frac{4}{17}\right)\log_2\left(\frac{4}{17}\right) - \left(\frac{13}{17}\right)\log_2\left(\frac{13}{17}\right) \approx 0.79$$

Information gain-Example 01

- ✔ To see the weighted average of entropy of each node we will do as follows:

$$E(Parent|Energy) = \frac{13}{30} * 0.39 + \frac{17}{30} * 0.79 = 0.62$$

- ✔ Now we have the value of E(Parent) and E(Parent|Energy), information gain will be:

$$\begin{aligned} Information\ Gain &= E(parent) - E(parent|energy) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$

- ✔ Our parent entropy was near 0.99 and after looking at this value of information gain, we can say that the entropy of the dataset will decrease by 0.37 if we make "Energy" as our root node.
- ✔ Similarly, we will do this with the other feature "Motivation" and calculate its information gain.

Information gain-Example 01

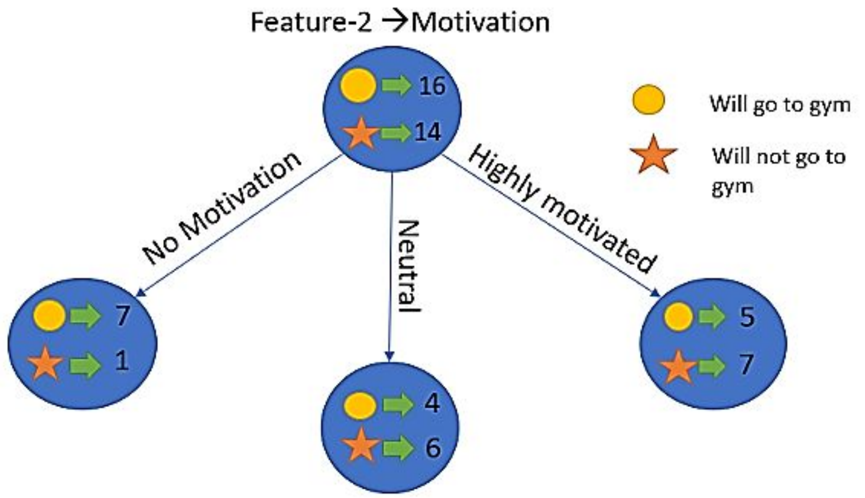
- ✔ Let's calculate the entropy here:

$$E(Parent) = 0.99$$

$$E\left(Parent|Motivation = \text{"No motivation"}\right) = -\left(\frac{7}{8}\right)log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) = 0.54$$

$$E(Parent|Motivation = \text{"Neutral"}) = -\left(\frac{4}{10}\right)log_2\left(\frac{4}{10}\right) - (\frac{6}{10})\log_2\left(\frac{6}{10}\right) = 0.97$$

$$E(Parent|Motivation = \text{"Highly motivated"}) = -\left(\frac{5}{12}\right)log_2\left(\frac{5}{12}\right) - (\frac{7}{12})\log_2\left(\frac{7}{12}\right) = 0.98$$



- ✔ To see the weighted average of entropy of each node we will do as follows:

$$E(Parent|Motivation) = \frac{8}{30} * 0.54 + \frac{10}{30} * 0.97 + \frac{12}{30} * 0.98 = 0.86$$

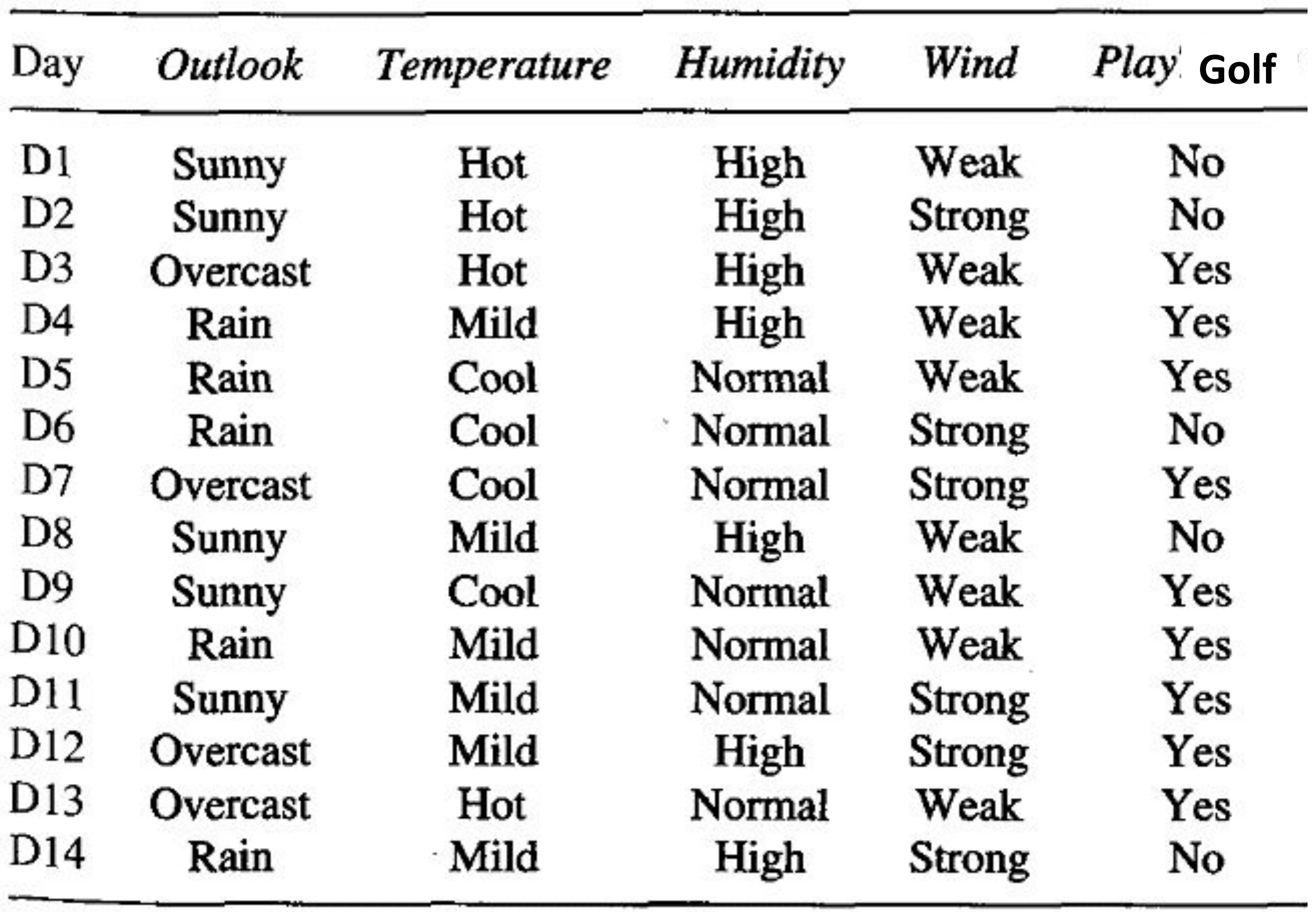
- ✔ Now we have the value of E(Parent) and E(Parent|Motivation), information gain will be:

$$\begin{aligned} Information\ Gain &= E(Parent) - E(Parent|Motivation) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

Information gain-Example 01

- ✔ We now see that the “Energy” feature gives more reduction which is 0.37 than the “Motivation” feature. Hence we will select the feature which has the highest information gain and then split the node based on that feature.
- ✔ In this example “Energy” will be our root node and we’ll do the same for sub-nodes. Here we can see that when the energy is “high” the entropy is low and hence we can say a person will definitely go to the gym if he has high energy, but what if the energy is low? We will again split the node based on the new feature which is “Motivation”.

Entropy-Example 02



Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*Play* Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TABLE 3.2
Training examples for the target concept *Play* Golf.

Entropy-Example 02

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - $(0.36 \log_2 0.36)$ - $(0.64 \log_2 0.64)$

= 0.94

✔ Entropy for multiple attributes is represented as:

		Play Golf		
		Yes	No	
Outlook	Sunny	2	3	5
	Overcast	4	0	4
	Rainy	3	2	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

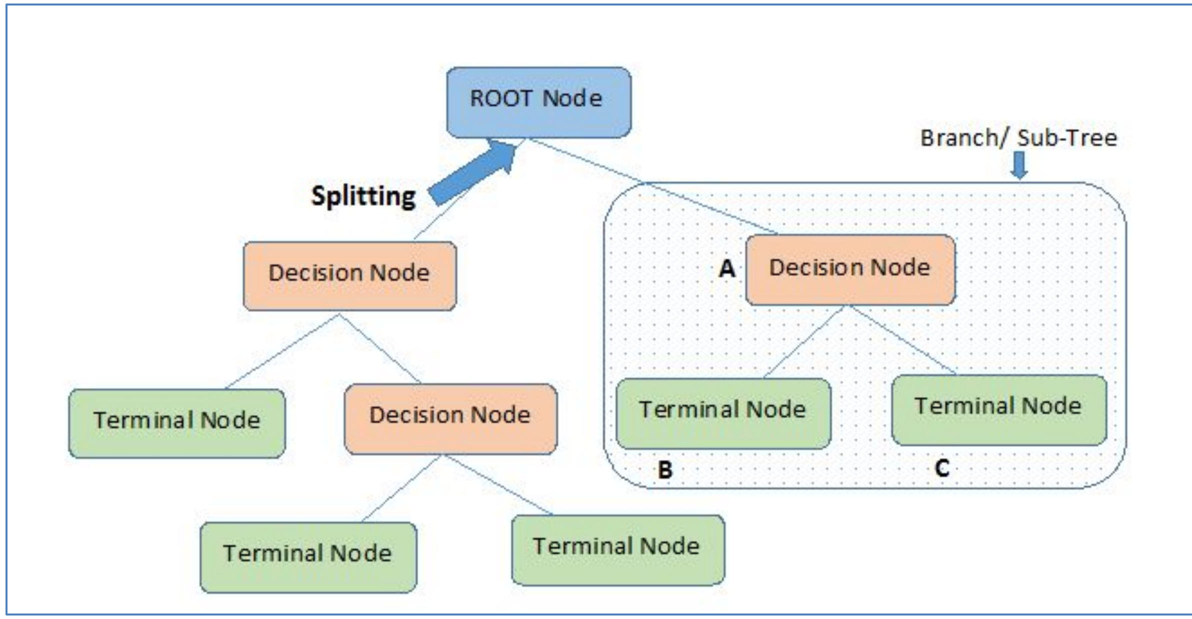
= 0.693

Entropy

- ✔ Always remember that the higher the Entropy, the lower will be the purity and the higher will be the impurity.
- ✔ As the goal of machine learning is to decrease the uncertainty or impurity in the dataset, here by using the entropy we are getting the impurity of a particular node, we don’t know if the parent entropy or the entropy of a particular node has decreased or not.
- ✔ For this, we bring a new metric called “Information gain” which tells us how much the parent entropy has decreased after splitting it with some feature.

Important Terminology

- ✔ **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- ✔ **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- ✔ **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
- ✔ **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
- ✔ **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
- ✔ **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.

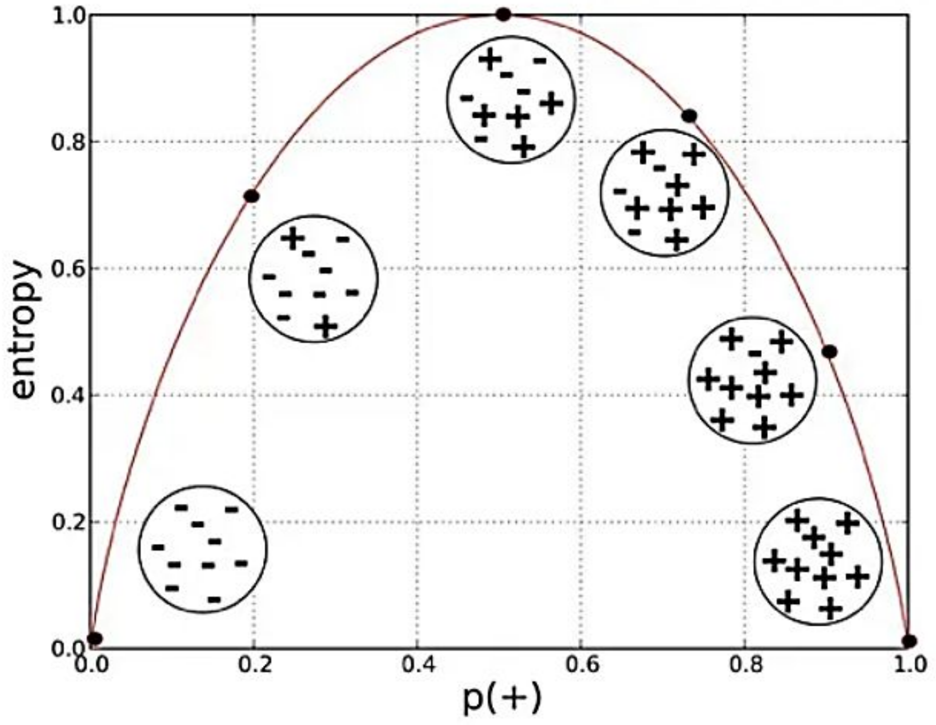


Decision Tree Algorithm

- ✔ The decision tree algorithm can be used for solving **regression and classification problems**
- ✔ The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from training data.
- ✔ In Decision Trees, for predicting a class label for a record we start from the **root** of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Decision Tree Algorithm

✔ ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

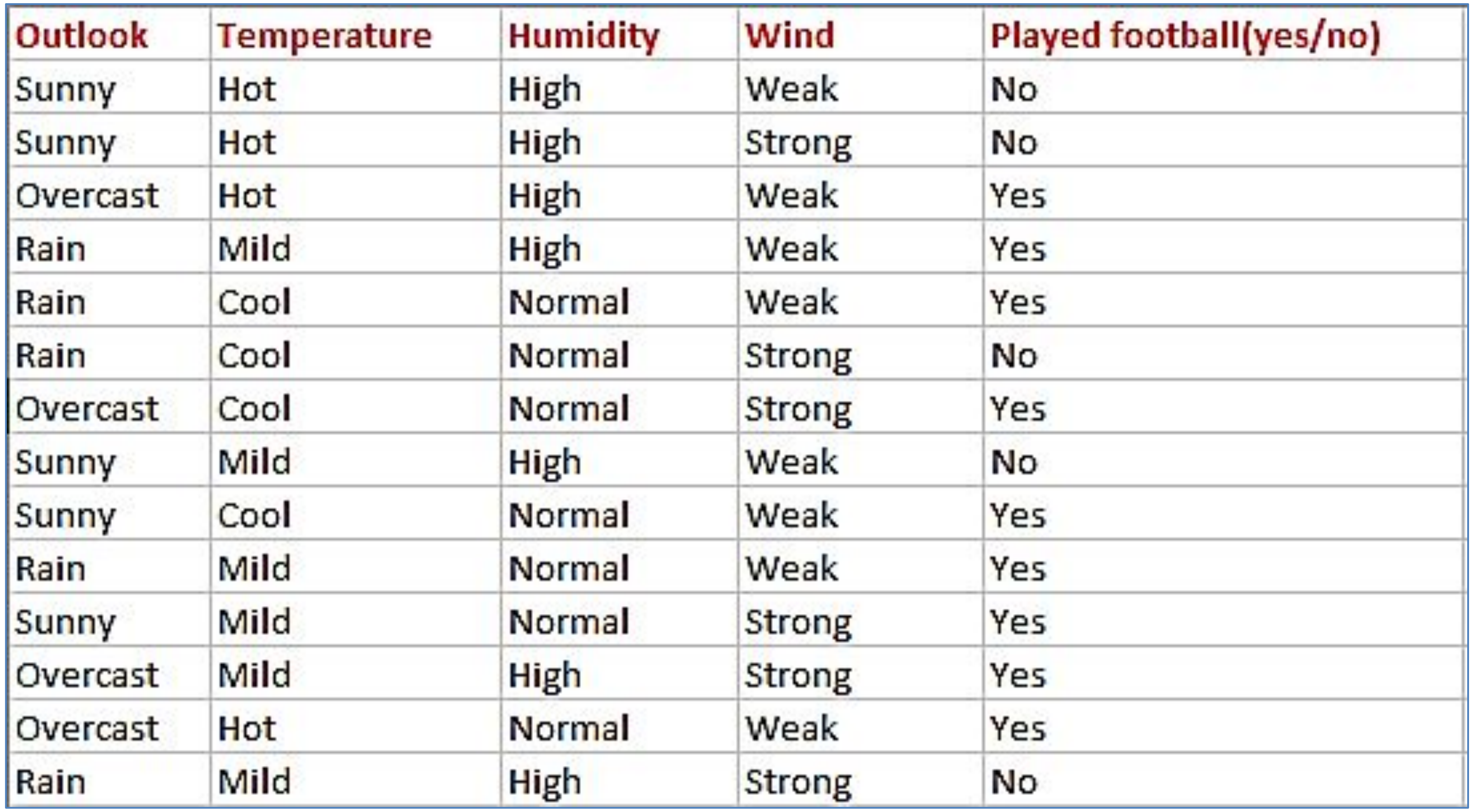


$H(X) = -\Sigma (pi * \log2\ pi)$

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step. Invented by Ross Quinlan, ID3 uses a top-down greedy approach to build a decision tree.

Classification using the ID3 algorithm

- ✔ Consider a dataset based on which we will determine whether to play football or



Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

- ✔ Here There are four independent variables to determine the dependent variable.
- ✔ The independent variables are Outlook, Temperature, Humidity, and Wind.
- ✔ The dependent variable is whether to play football or not.

Calculations

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i)$$

Here
9 samples are Yes
5 samples are No

- **Step 1**
  - *Find the entropy of the class variable*
    - $H(S) \leftarrow \begin{Bmatrix} \mathbf{9} \leftarrow \boldsymbol{Yes} \\ \mathbf{5} \leftarrow \boldsymbol{No} \end{Bmatrix} = -P(Yes) log_2 P(Yes) - \boldsymbol{P(No) log_2 P(No)}$

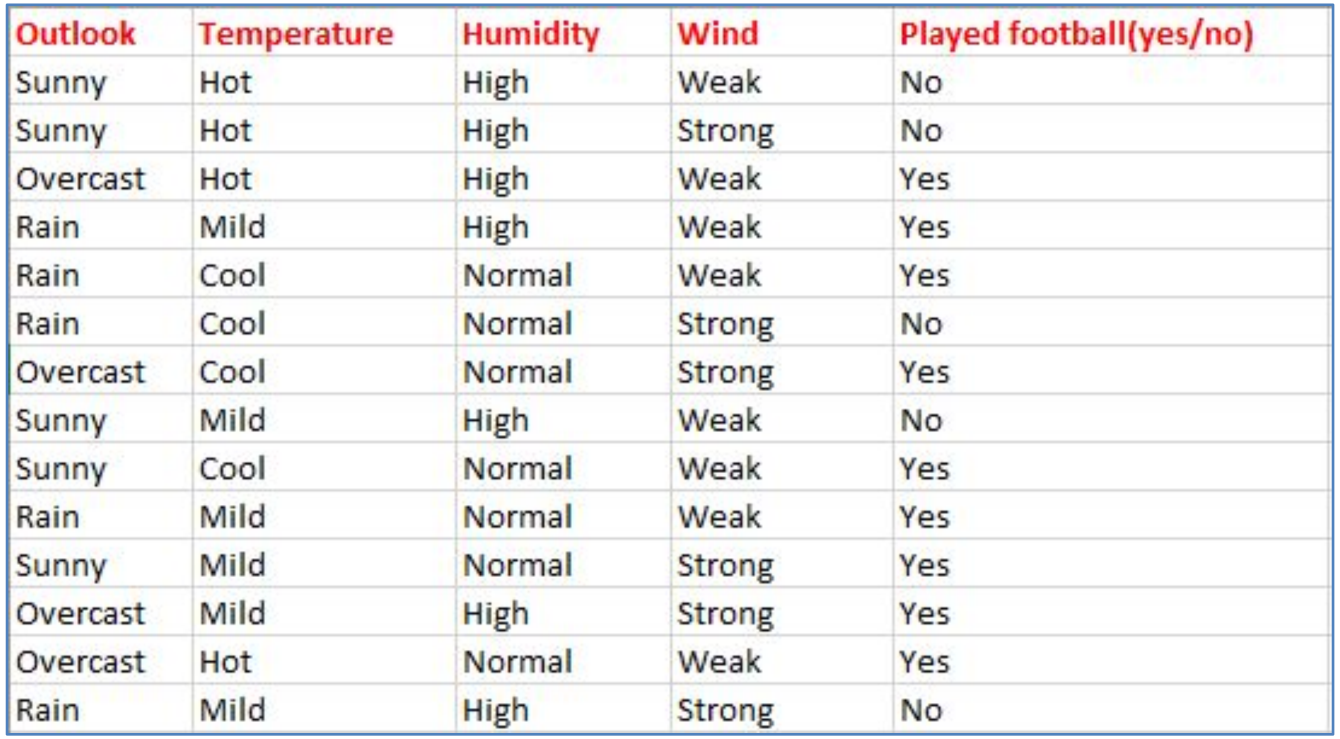
$$= -\frac{9}{14} log_2\left(\frac{9}{14}\right) - \frac{5}{14} log_2\left(\frac{5}{14}\right)$$

$$= 0.94$$

Step 2: Calculate average weighted entropy for each attribute

✔ For Outlook: Values(Outlook)= Sunny, Overcast, Rain

- Entropy (S_{Sunny}) ← $\begin{Bmatrix} 2 \leftarrow Yes \\ 3 \leftarrow No \end{Bmatrix} = -\frac{2}{5} log_2\left(\frac{2}{5}\right) - \frac{3}{5} log_2\left(\frac{3}{5}\right) = 0.971$
- Entropy ($S_{Overcast}$) ← $\begin{Bmatrix} 4 \leftarrow Yes \\ 0 \leftarrow No \end{Bmatrix} = 0$
- Entropy (S_{Rain}) ← $\begin{Bmatrix} 3 \leftarrow Yes \\ 2 \leftarrow No \end{Bmatrix} = 0.971$

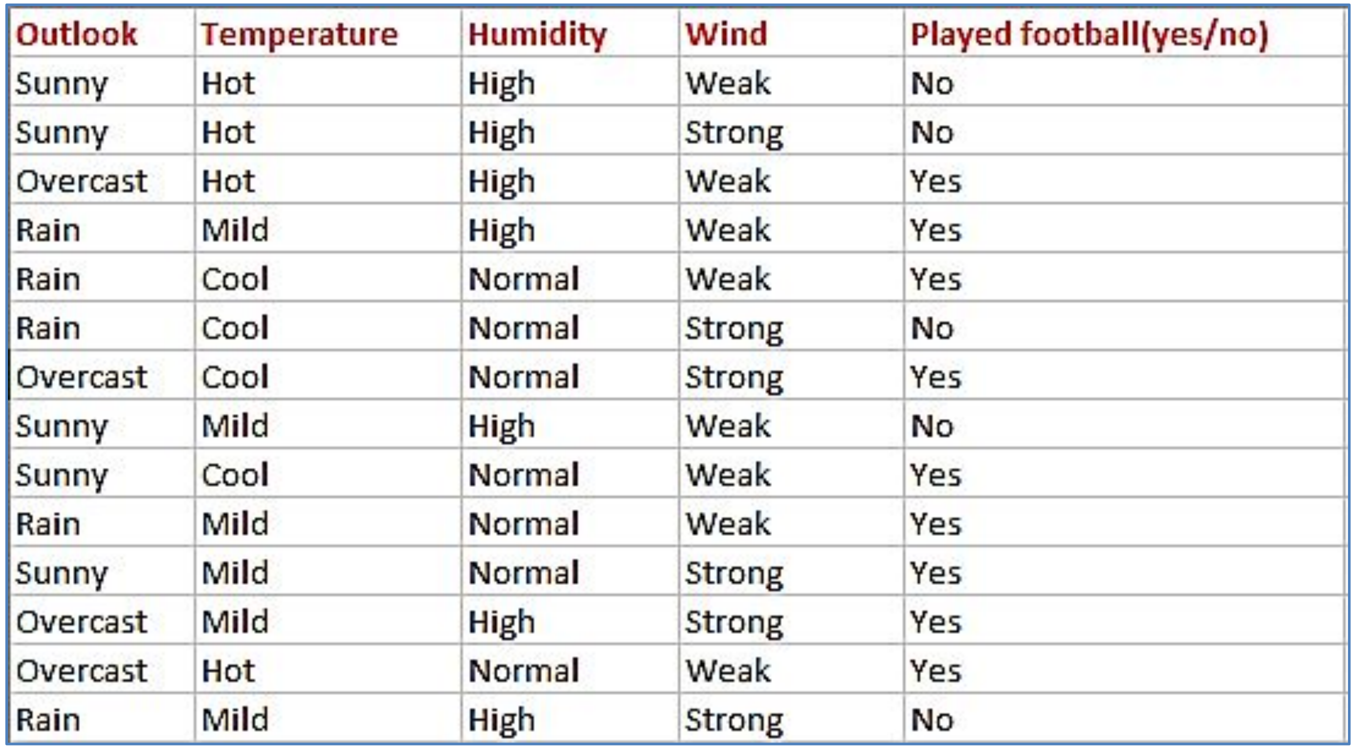


Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 2: Calculate average weighted entropy for each attribute (Cont…)

✔ *For Temperature: Values(Temperature)= Hot, Mild, Cool*

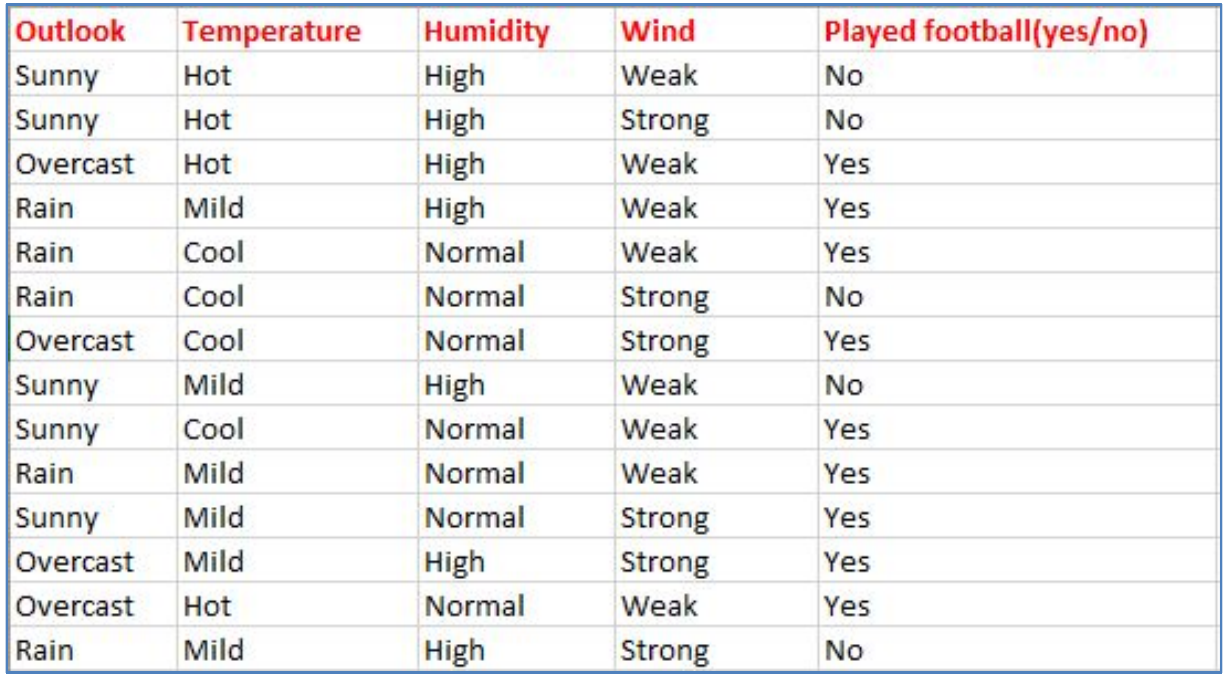
- **Entropy (S_{Hot}) ←** $\begin{Bmatrix} 2 \leftarrow Yes \\ 2 \leftarrow No \end{Bmatrix} = -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{2}{5}log_2\left(\frac{2}{5}\right) = 1$
- **Entropy (S_{Mild}) ←** $\begin{Bmatrix} 4 \leftarrow Yes \\ 2 \leftarrow No \end{Bmatrix} = 0.9183$
- **Entropy (S_{Cool}) ←** $\begin{Bmatrix} 3 \leftarrow Yes \\ 1 \leftarrow No \end{Bmatrix} = 0.8113$



Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 2: Calculate average weighted entropy for each attribute (Cont…)

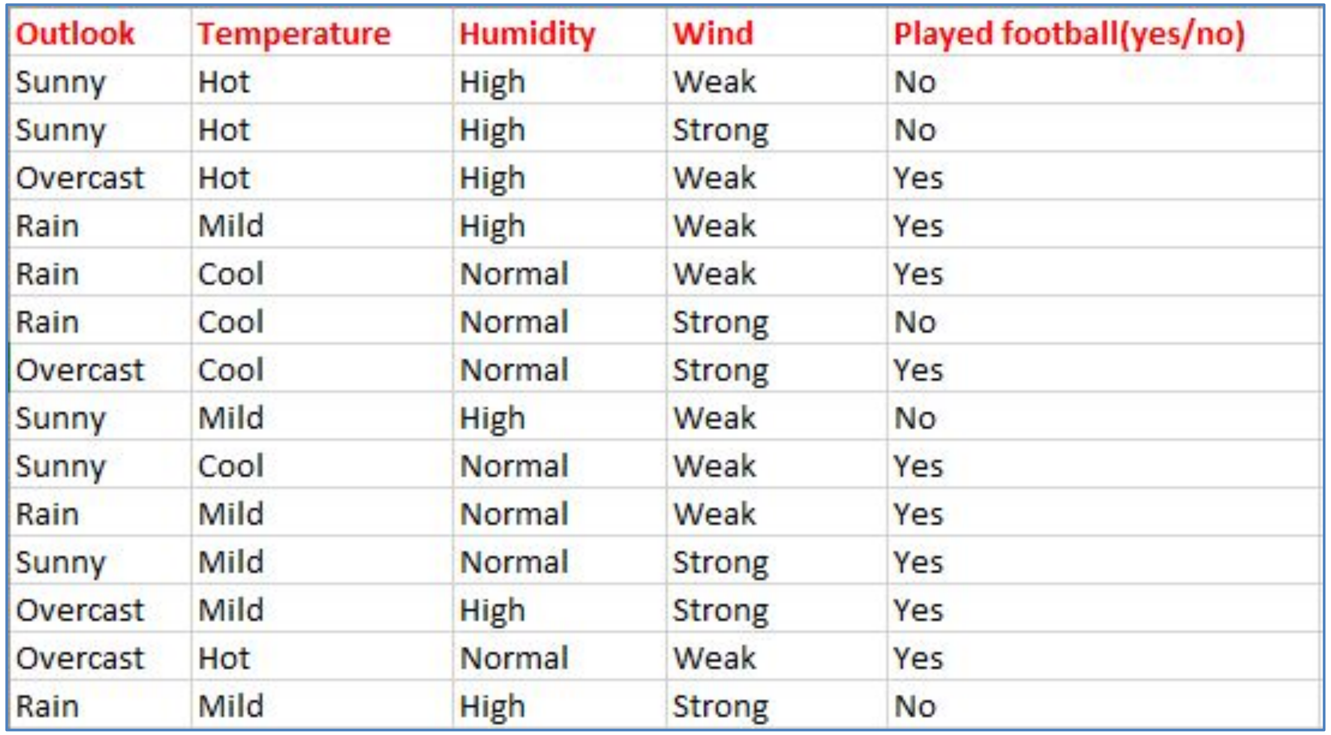
- For Humidity: Values(Humidity)= High, Normal
  - Entropy (S_{High}) ← $\begin{cases} 3 \leftarrow Yes \\ 4 \leftarrow No \end{cases} = -\frac{3}{7} log_2\left(\frac{6}{7}\right) - \frac{2}{7} log_2\left(\frac{4}{7}\right) = 0.9852$
  - Entropy (S_{Normal}) ← $\begin{cases} 6 \leftarrow Yes \\ 1 \leftarrow No \end{cases} = 0.5916$



Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 2: Calculate average weighted entropy for each attribute (Cont…)

- *For Wind: Values(Wind)= Weak, Strong*
  - Entropy (S_{Weak}) ← $\begin{Bmatrix} 6 \leftarrow \boldsymbol{Yes} \\ 2 \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{0.8113}$
  - Entropy (S_{Strong}) ← $\begin{Bmatrix} 3 \leftarrow \boldsymbol{Yes} \\ 3 \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{1}$



Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step3: Find the information gain: Outlook

- **Entropy (S_{Sunny}) ←** $\begin{Bmatrix} 2 \leftarrow Yes \\ 3 \leftarrow No \end{Bmatrix} = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971$
- **Entropy ($S_{Overcast}$) ←** $\begin{Bmatrix} 4 \leftarrow Yes \\ 0 \leftarrow No \end{Bmatrix} = 0$
- **Entropy (S_{Rain}) ←** $\begin{Bmatrix} 3 \leftarrow Yes \\ 2 \leftarrow No \end{Bmatrix} = 0.971$

- $H(S) \leftarrow \begin{Bmatrix} 9 \leftarrow Yes \\ 5 \leftarrow No \end{Bmatrix} = -P(Yes)\log_2 P(Yes) - \boldsymbol{P(No)\log_2 P(No)}$

$$= -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

$$= 0.94$$

Information gain is the difference between parent entropy and average weighted entropy we found above.

✔ $$Gain(S, Outlook) = Entropy(S) - \sum_{v \in (Sunny, Overcast, Rain)} \frac{|S_v|}{|S|} Entropy(S_v)$$

➔ Gain(S,Outlook)= H(s) $- \frac{5}{14}Entropy(Sunny) - \frac{4}{14}Entropy(Overcast) - \frac{5}{14}Entropy(Rain)$

➔ Gain(S,Outlook)= 0.94 $- \left(\frac{5}{14} \times 0.971\right) - \left(\frac{4}{14} \times 0.0\right) - \left(\frac{5}{14} \times 0.971\right)$

➔ **Gain(S,Outlook)=0.2464**

Gain: Temperature

- **Entropy (S_{Hot}) ←** $\begin{Bmatrix} \mathbf{2} \leftarrow \boldsymbol{Yes} \\ \mathbf{2} \leftarrow \boldsymbol{No} \end{Bmatrix} = -\frac{2}{5} \boldsymbol{log_2}\left(\frac{2}{5}\right) - \frac{2}{5} \boldsymbol{log_2}\left(\frac{2}{5}\right) = \mathbf{1}$
- **Entropy (S_{Mild}) ←** $\begin{Bmatrix} \mathbf{4} \leftarrow \boldsymbol{Yes} \\ \mathbf{2} \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{0.9183}$
- **Entropy (S_{Cool}) ←** $\begin{Bmatrix} \mathbf{3} \leftarrow \boldsymbol{Yes} \\ \mathbf{1} \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{0.8113}$

- $H(S) \leftarrow \begin{Bmatrix} \mathbf{9} \leftarrow \boldsymbol{Yes} \\ \mathbf{5} \leftarrow \boldsymbol{No} \end{Bmatrix} = -P(Yes)log_2P(Yes) - \boldsymbol{P(No)log_2P(No)}$

$$= -\frac{9}{14} log_2\left(\frac{9}{14}\right) - \frac{5}{14} log_2\left(\frac{5}{14}\right)$$

$$= 0.94$$

$$Gain(S, Temperure) = Entropy(S) - \sum_{v \in (Hot, Mild, Cool)} \frac{|S_v|}{|S|} Entropy(S_v)$$

→ Gain($S, Temperure$)= H(s) − $\frac{4}{14} Entropy(Hot) - \frac{6}{14} Entropy(Mild) - \frac{4}{14} Entropy(Cool)$

Gain($S, Temperure$)=0.0289 $or\ 0.0292$

Gain: Humidity

- Entropy (S_{High}) ← $\begin{Bmatrix} \mathbf{3} \leftarrow \boldsymbol{Yes} \\ \mathbf{4} \leftarrow \boldsymbol{No} \end{Bmatrix} = -\frac{3}{7}\boldsymbol{log_2}\left(\frac{6}{7}\right) - \frac{2}{7}\boldsymbol{log_2}\left(\frac{4}{7}\right) = \mathbf{0.9852}$

- Entropy (S_{Normal}) ← $\begin{Bmatrix} \mathbf{6} \leftarrow \boldsymbol{Yes} \\ \mathbf{1} \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{0.5916}$

- $H(S) \leftarrow \begin{Bmatrix} \mathbf{9} \leftarrow \boldsymbol{Yes} \\ \mathbf{5} \leftarrow \boldsymbol{No} \end{Bmatrix} = -P(Yes)log_2P(Yes) - \boldsymbol{P(No)log_2P(No)}$

$$= -\frac{9}{14}log_2\left(\frac{9}{14}\right) - \frac{5}{14}log_2\left(\frac{5}{14}\right)$$

$$= 0.94$$

$$Gain(S, Humidity) = Entropy(S) - \sum_{v \in (High, Normal)} \frac{|S_v|}{|S|} Entropy(S_v)$$

→ Gain(

→ Gain(S,Humidity)=**0.1516 or 0.1519**

Gain: Wind

- **Entropy (S$_{\text{Weak}}$) ←** $\begin{Bmatrix} \mathbf{6} \leftarrow \boldsymbol{Yes} \\ \mathbf{2} \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{0.8113}$
- **Entropy (S$_{\text{Strong}}$) ←** $\begin{Bmatrix} \mathbf{3} \leftarrow \boldsymbol{Yes} \\ \mathbf{3} \leftarrow \boldsymbol{No} \end{Bmatrix} = \mathbf{1}$

- $H(S) \leftarrow \begin{Bmatrix} \mathbf{9} \leftarrow \boldsymbol{Yes} \\ \mathbf{5} \leftarrow \boldsymbol{No} \end{Bmatrix} = -P(Yes)log_2P(Yes) - \boldsymbol{P(No)log_2P(No)}$

$$= -\frac{9}{14}log_2\left(\frac{9}{14}\right) - \frac{5}{14}log_2\left(\frac{5}{14}\right)$$

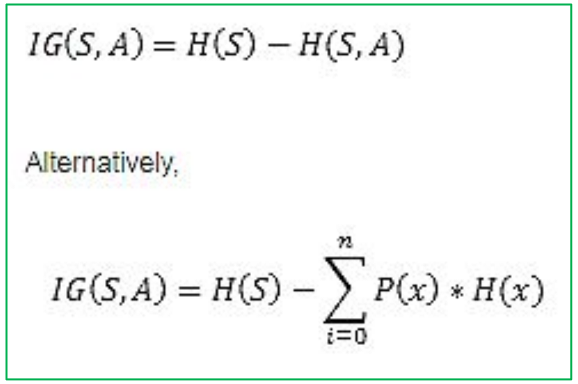
$$= 0.94$$

$$Gain(S,Wind) = Entropy(S) - \sum_{v\in(Weak,Strong)} \frac{|S_v|}{|S|} Entropy(S_v)$$

⇒ Gain(S,Weak)= H(s)⇒ **Gain(S,Wind)=0.0478**

Information gain

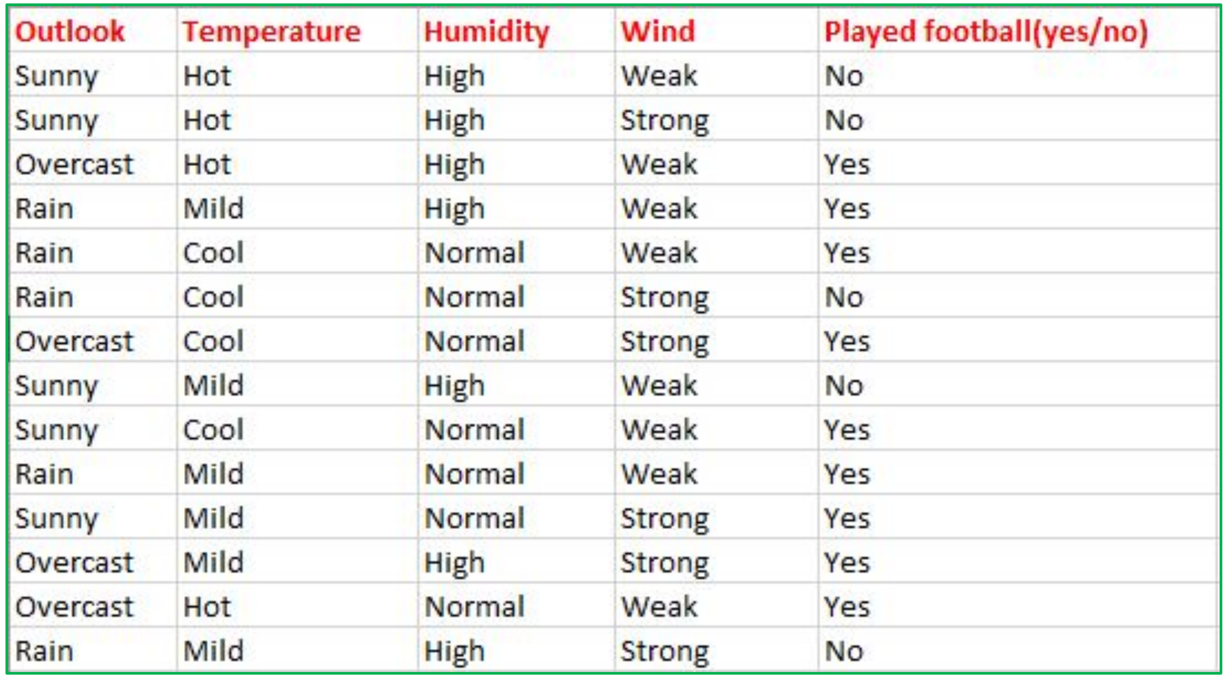
- IG(S, outlook) = 0.94 - 0.693 = 0.247
- IG(S, Temperature) = 0.940 - 0.911 = 0.029
- IG(S, Humidity) = 0.940 - 0.788 = 0.152
- IG(S, Windy) = 0.940 - 0.8932 = 0.048



$$IG(S,A) = H(S) - H(S,A)$$

Alternatively,

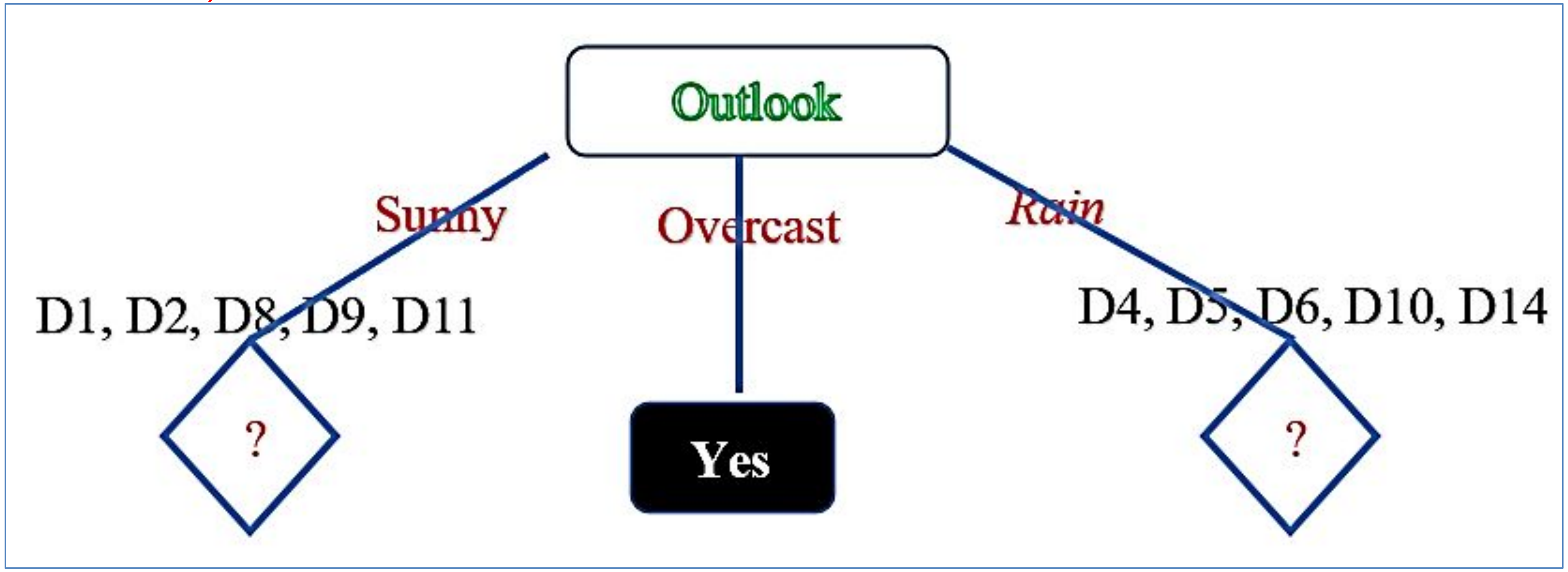
$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

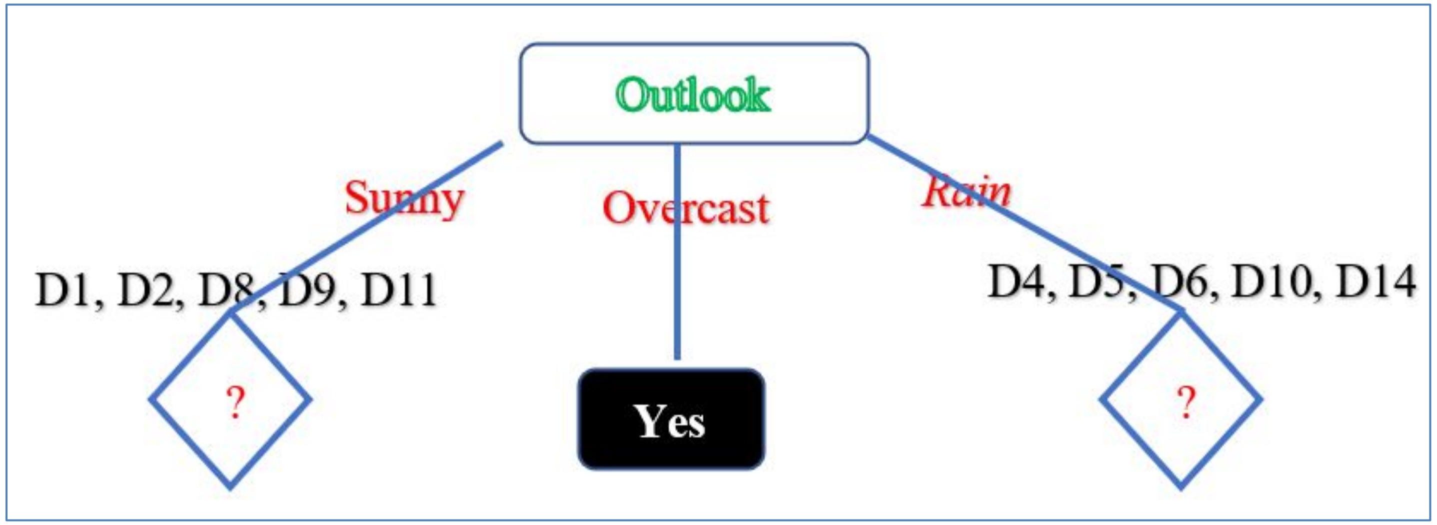


Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Finding Root Node

- ✔ *Since Outlook has the highest gain so, Outlook is the root node. And it has three child nodes (Sunny, Overcast, Rain)*





Data	Outlook	Temperature	Humidity	Windy	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Data	Outlook	Temperature	Humidity	Windy	Play
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

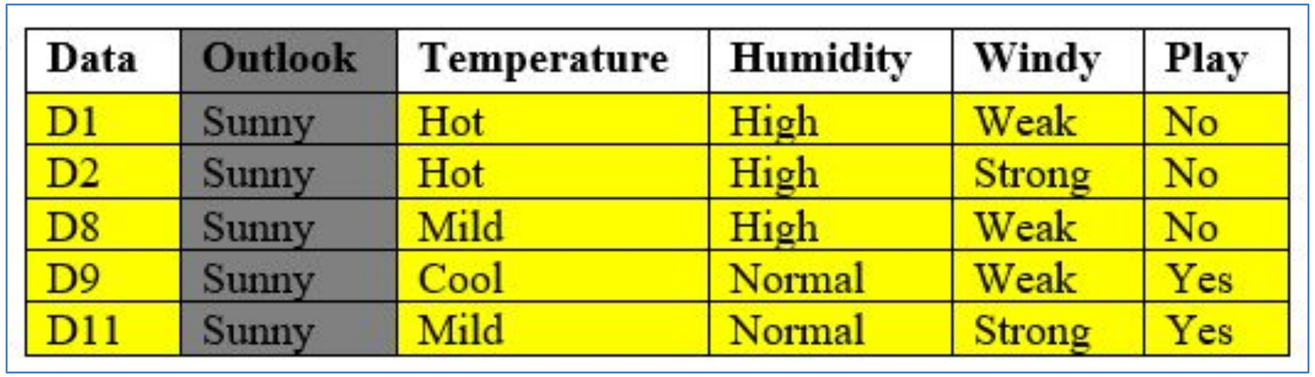
Data	Outlook	Temperature	Humidity	Windy	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Calculate IG of Temperature

✓ $H(S) \leftarrow \begin{Bmatrix} \mathbf{2} \leftarrow \boldsymbol{Yes} \\ \mathbf{3} \leftarrow \boldsymbol{No} \end{Bmatrix} = -\boldsymbol{P(Yes)log_2P(Yes)} -$

$$= -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right)$$

$$= 0.971$$



Data	Outlook	Temperature	Humidity	Windy	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

✓ *For Temperature:*

Values(Temperature)= Hot, Mild, Cool

✓ Entropy (S_{Hot}) $\leftarrow \begin{Bmatrix} 0 \leftarrow Yes \\ 2 \leftarrow No \end{Bmatrix} = -\frac{0}{2}log_2\left(\frac{0}{2}\right) - \frac{2}{2}log_2\left(\frac{2}{2}\right) = 0$

✓ Entropy (S_{Mild}) $\leftarrow \begin{Bmatrix} 1 \leftarrow Yes \\ 1 \leftarrow No \end{Bmatrix}$ =1

✓ Entropy (S_{Cool}) $\leftarrow \begin{Bmatrix} 1 \leftarrow Yes \\ 0 \leftarrow No \end{Bmatrix} = 0$

$$Gain(S, Temperature) = H(s) - \sum_{v \in (Hor, Mild, Cool)} \frac{|S_v|}{|S|} Entropy(S_v)$$

➩ Gain(S,Temperature)= H(s) $-\frac{2}{5}Entropy(Hot) - \frac{2}{5}Entropy(Mild) - \frac{1}{5}Entropy(Cool)$

➩ Gain(S,Temperature)= $0.971 - \frac{2}{5}0 - \frac{2}{5}1 - \frac{1}{5}0$

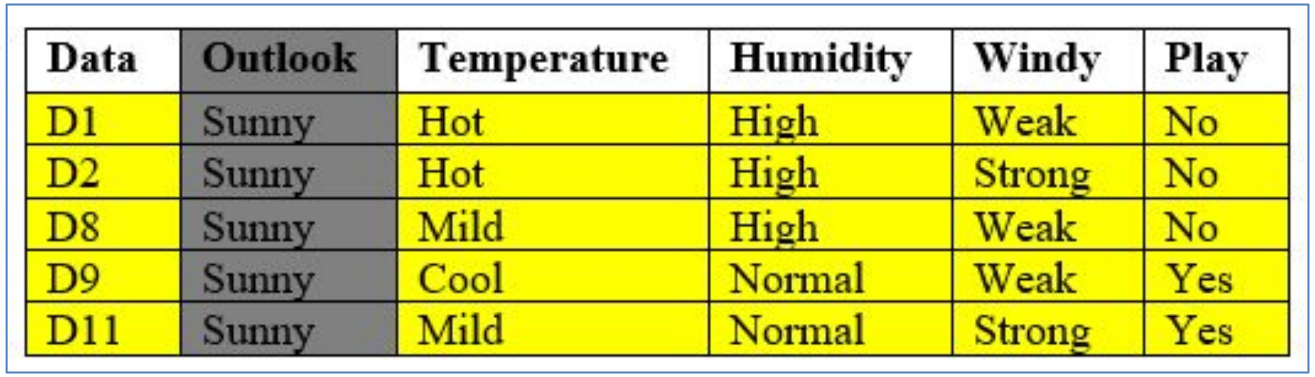
➩ **Gain(S,Temperature)=0.571**

Calculate IG of *Humidity*

- ✓ $H(S) \leftarrow \begin{Bmatrix} 2 \leftarrow Yes \\ 3 \leftarrow No \end{Bmatrix} = -P(Yes)log_2P(Yes) -$

$$= -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right)$$

$$= 0.971$$



Data	Outlook	Temperature	Humidity	Windy	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

- ✓ *For Humidity:*

Values(Humidity)= High, Normal

- ✓ Entropy (S_{High}) $\leftarrow \begin{Bmatrix} 0 \leftarrow Yes \\ 3 \leftarrow No \end{Bmatrix} = -\frac{0}{3}log_2\left(\frac{0}{3}\right) - \frac{3}{3}log_2\left(\frac{3}{3}\right) = 0$
- ✓ Entropy (S_{Normal}) $\leftarrow \begin{Bmatrix} 2 \leftarrow Yes \\ 0 \leftarrow No \end{Bmatrix} = 0$

$$Gain(S, Humidity) = H(s) - \sum_{v \in (High\ Normal)} \frac{|S_v|}{|S|} Entropy(S_v)$$

➔ Gain(S,Humidity)= H(s) $- \frac{3}{5} Entropy(High) - \frac{2}{5} Entropy(Normal)$

➔ Gain(S,Temperature)= 0.971 − 0

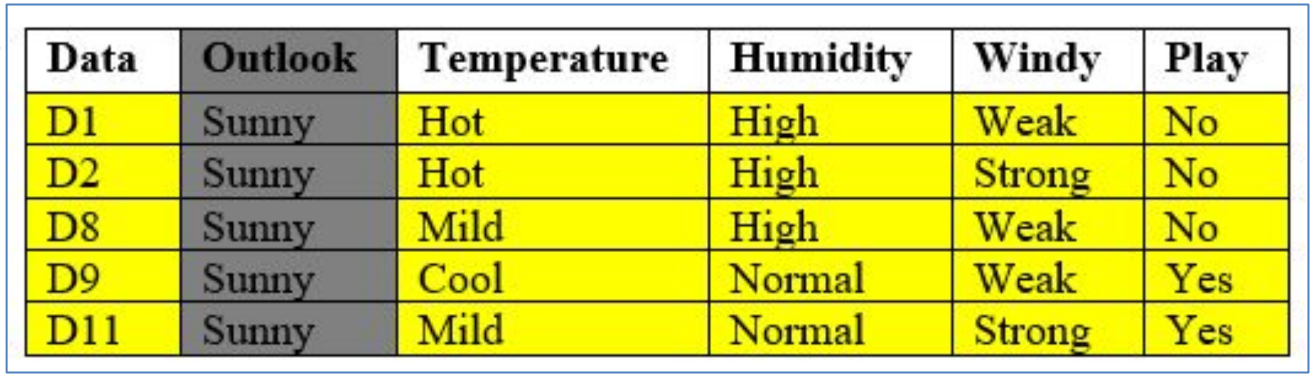
➔ **Gain(S,*Humidity*)=0.971**

Calculate IG of *Windy*

✔ $H(S) \leftarrow \begin{Bmatrix} 2 \leftarrow Yes \\ 3 \leftarrow No \end{Bmatrix} = -P(Yes)log_2P(Yes) -$

$$= -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right)$$

$$= 0.971$$



Data	Outlook	Temperature	Humidity	Windy	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

✓ *For Windy:*

Values(Windy)= Weak, Strong

✓ Entropy (S_{Weak}) $\leftarrow \begin{Bmatrix} 1 \leftarrow Yes \\ 2 \leftarrow No \end{Bmatrix} = -\frac{1}{3}log_2\left(\frac{1}{3}\right) - \frac{2}{3}log_2\left(\frac{2}{3}\right) = .5278$

✓ Entropy (S_{Strong} $\leftarrow \begin{Bmatrix} 1 \leftarrow Yes \\ 1 \leftarrow No \end{Bmatrix} = 1$

$$Gain(S, Humidity) = H(s) - \sum_{v \in (High\,Normal)} \frac{|S_v|}{|S|} Entropy(S_v)$$

➔ Gain(S,Humidity)= H(s) $- \frac{3}{5} Entropy(High) - \frac{2}{5} Entropy(Normal)$

➔ Gain(S,Temperature)= $0.971 - 0$

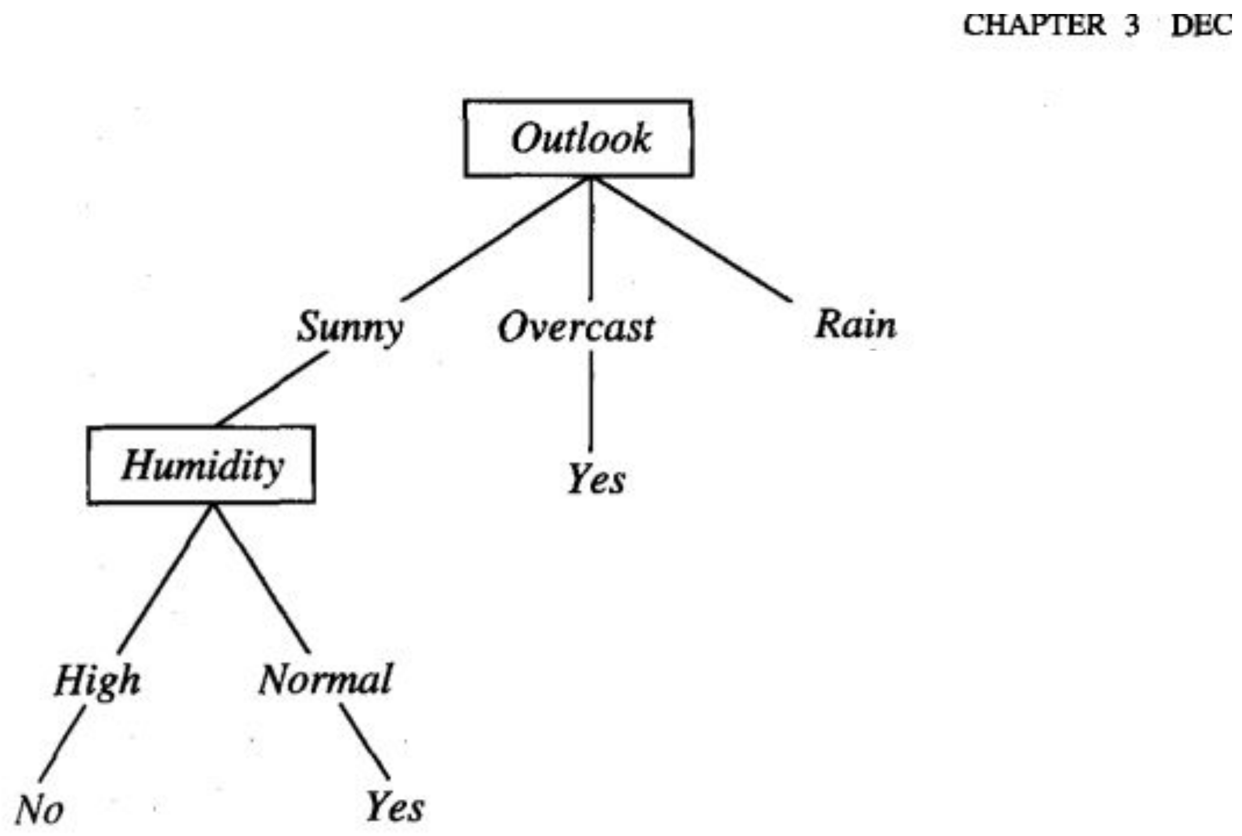
➔ **Gain(S,Windy)=0.019**

Information gain

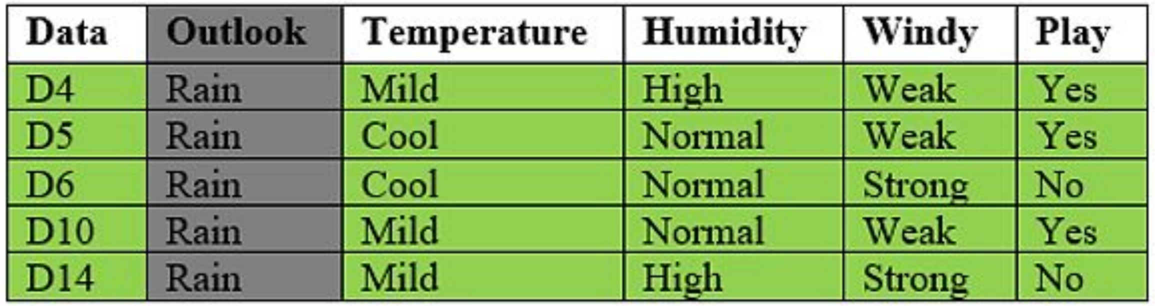
➔ Gain(S,Temperature)=0.571

Gain(S,<u>Humidity</u>)=0.971

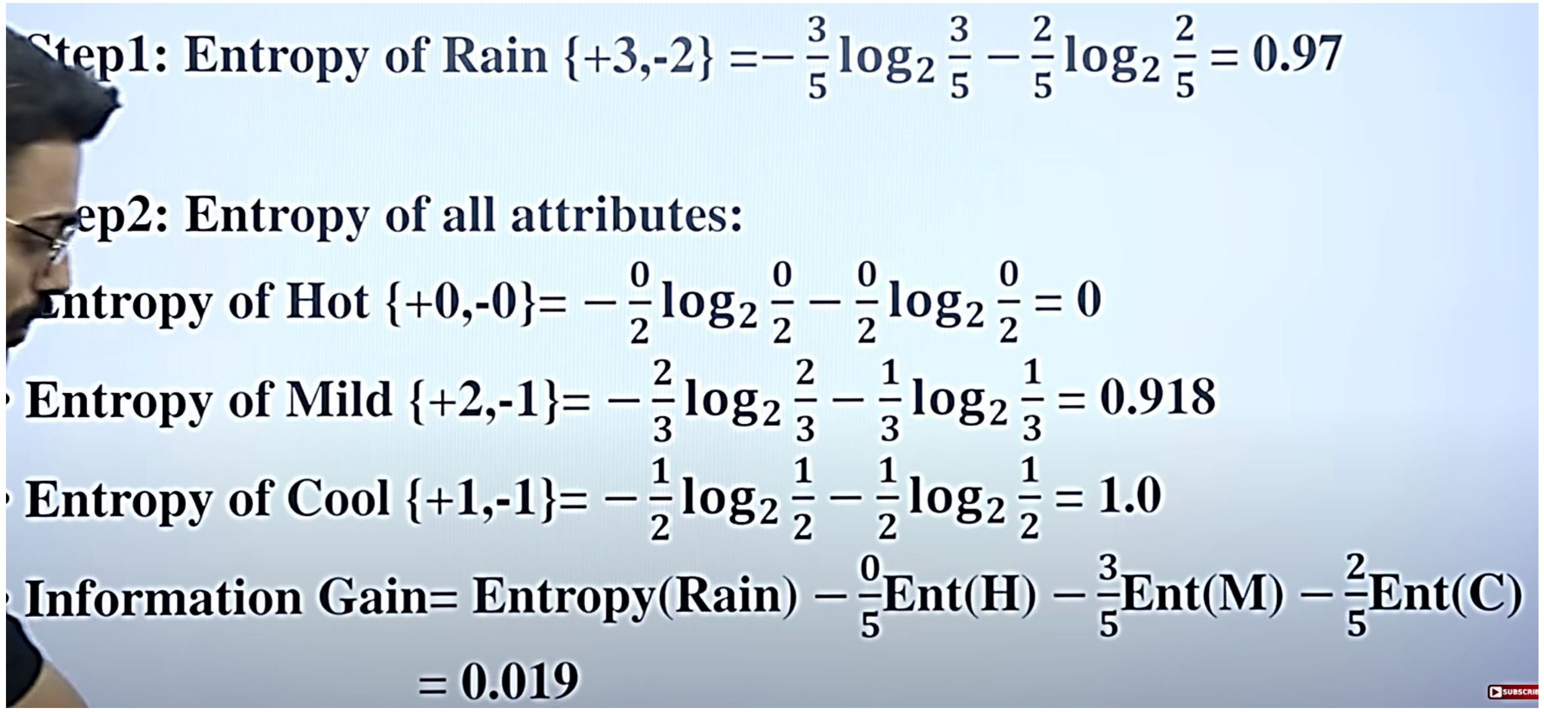
➔Gain(S,<u>Windy</u>)=0.019



Calculate IG of Temperature



Data	Outlook	Temperature	Humidity	Windy	Play
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



tep1: Entropy of Rain {+3,-2} $= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

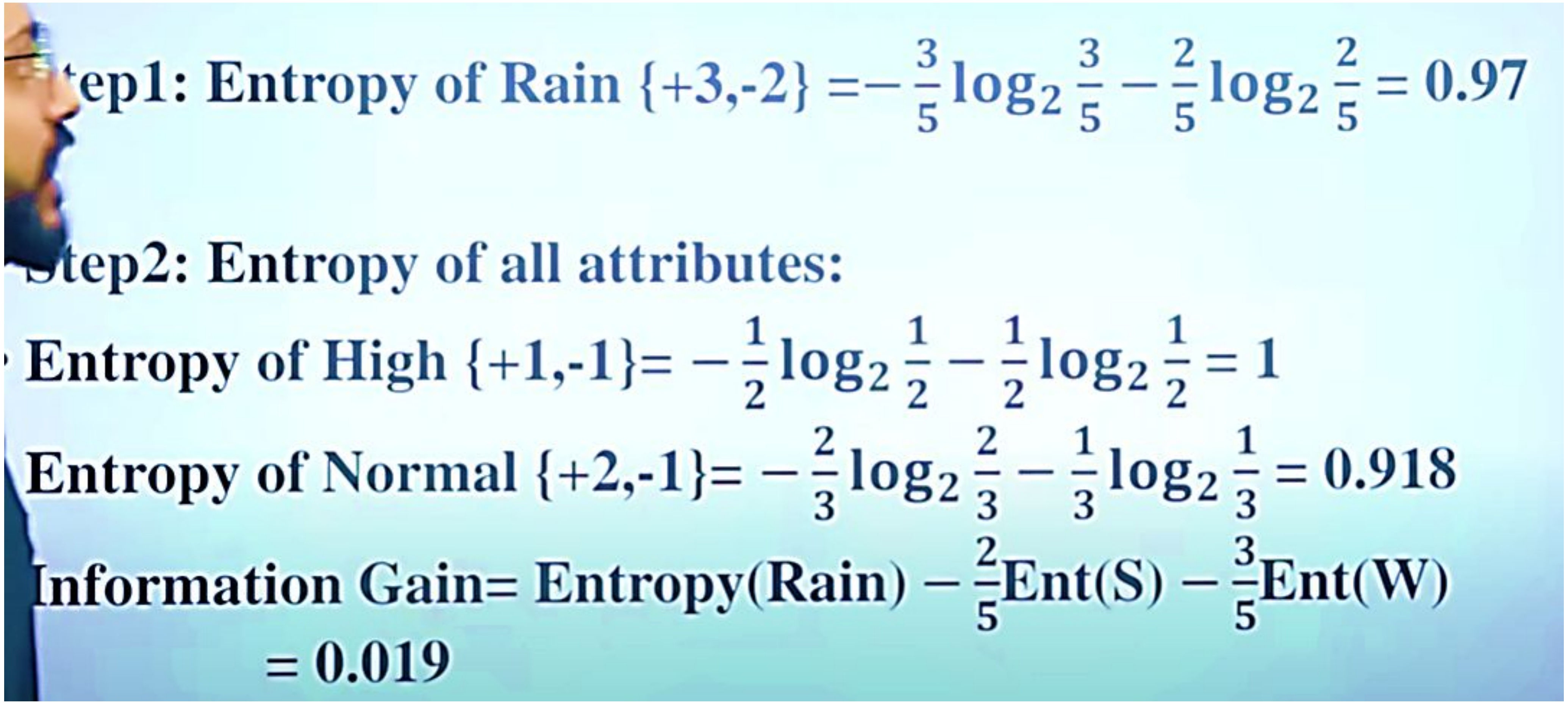
ep2: Entropy of all attributes:

ntropy of Hot {+0,-0}$= -\frac{0}{2}\log_2\frac{0}{2} - \frac{0}{2}\log_2\frac{0}{2} = 0$

- **Entropy of Mild {+2,-1}**$= -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$
- **Entropy of Cool {+1,-1}**$= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1.0$
- **Information Gain= Entropy(Rain)** $-\frac{0}{5}\text{Ent(H)} - \frac{3}{5}\text{Ent(M)} - \frac{2}{5}\text{Ent(C)}$

$= 0.019$

Calculate IG of *Humidity*



tep1: Entropy of Rain {+3,-2} $= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

tep2: Entropy of all attributes:

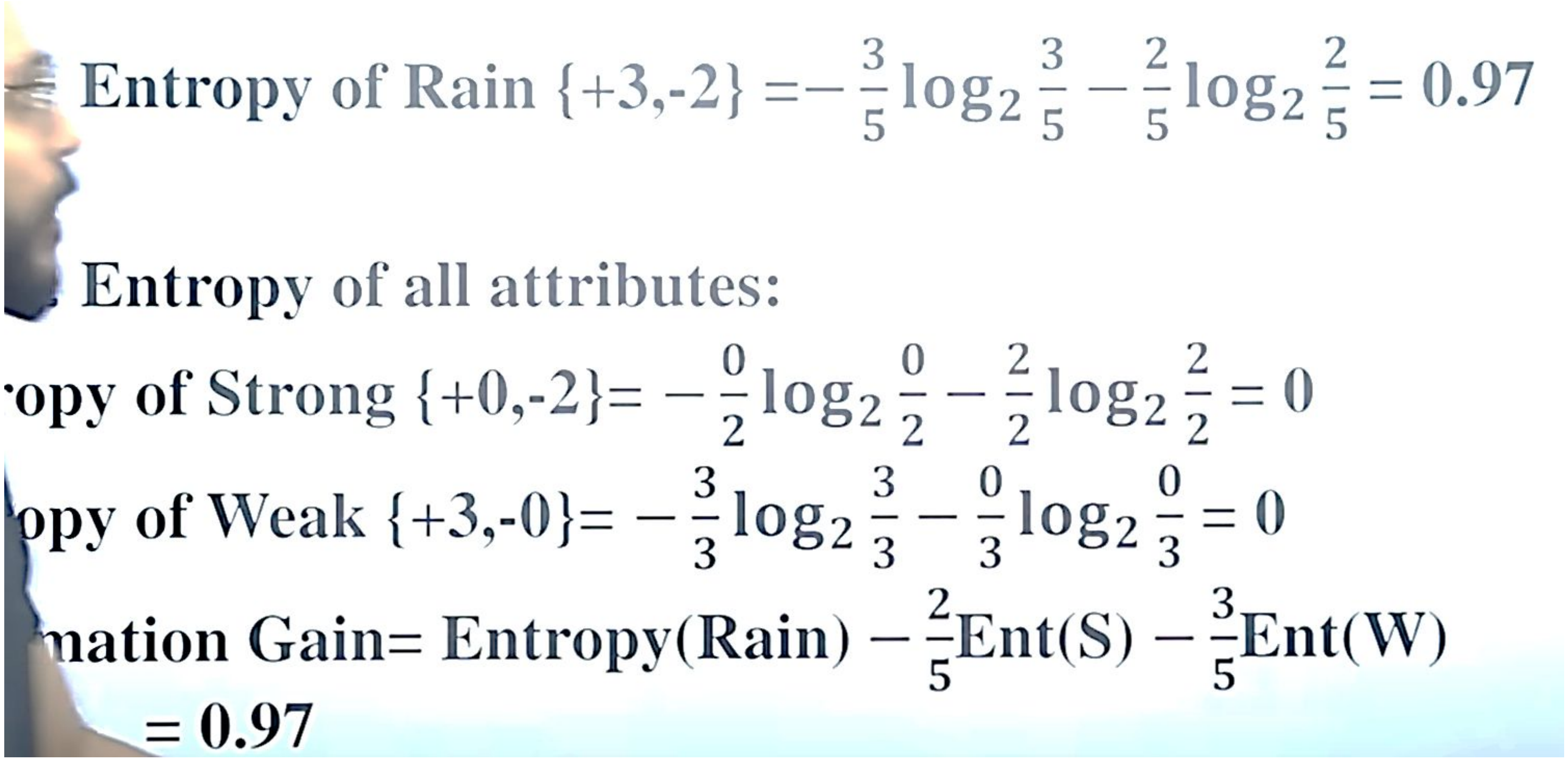
Entropy of High {+1,-1}$= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$

Entropy of Normal {+2,-1}$= -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$

Information Gain= Entropy(Rain) $- \frac{2}{5}$Ent(S) $- \frac{3}{5}$Ent(W)

$= 0.019$

Calculate IG of *Windy*



Entropy of Rain {+3,-2} $= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

Entropy of all attributes:

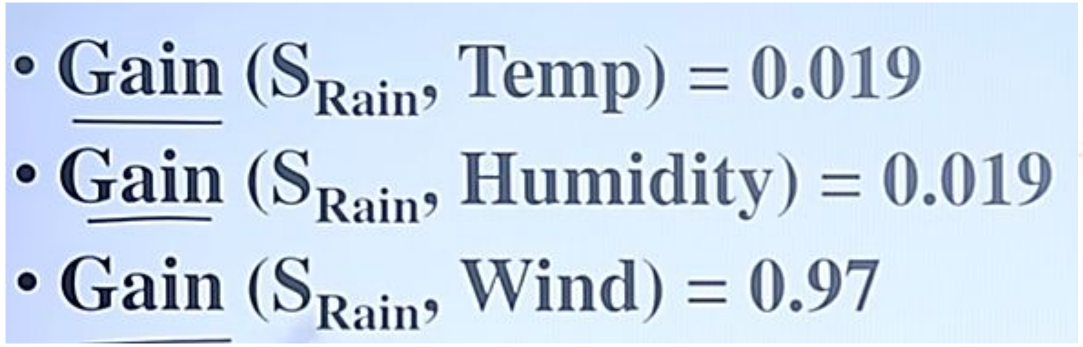
ropy of Strong {+0,-2}$= -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$

opy of Weak {+3,-0}$= -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3} = 0$

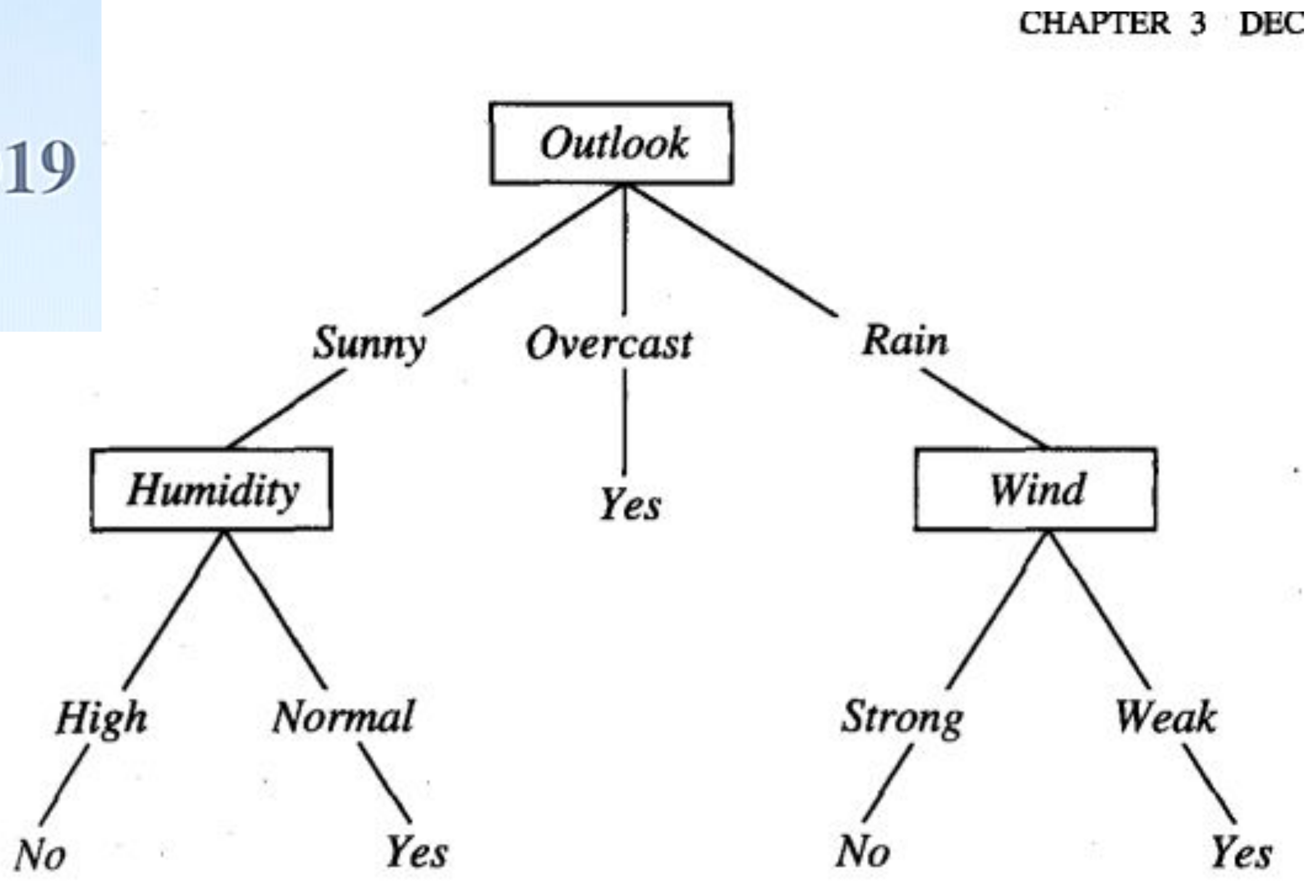
mation Gain= Entropy(Rain) $- \frac{2}{5}$Ent(S) $- \frac{3}{5}$Ent(W)

$= 0.97$

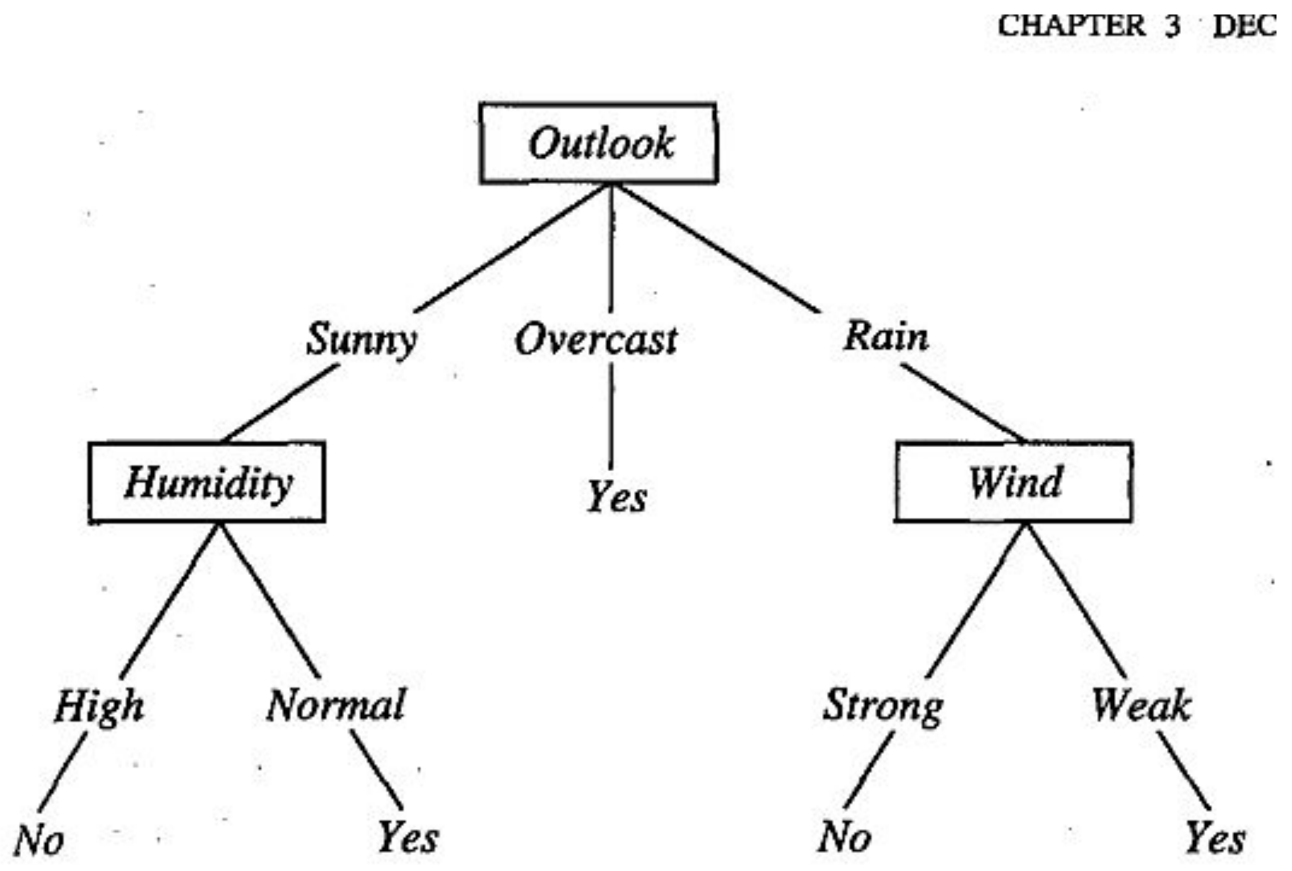
Information gain



- Gain (S_{Rain}, Temp) = 0.019
- Gain (S_{Rain}, Humidity) = 0.019
- Gain (S_{Rain}, Wind) = 0.97



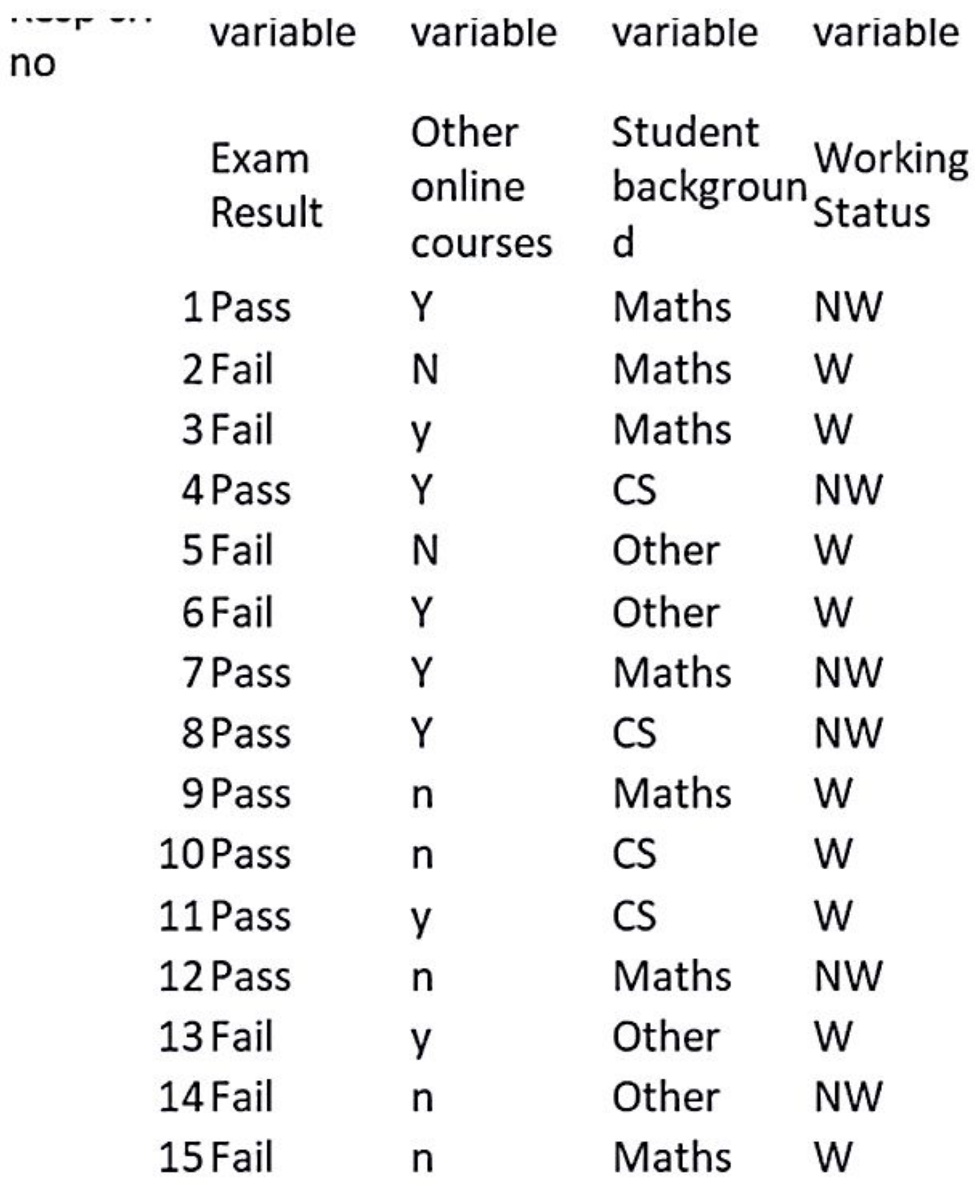
A decision tree for the concept PlayTennis



- ✔ An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, Yes or No).
- ✔ This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

Class Work

- ✔ Apply ID3 algorithm to build a decision tree for the given datasets



no	variable	variable	variable	variable
	Exam Result	Other online courses	Student background	Working Status
1	Pass	Y	Maths	NW
2	Fail	N	Maths	W
3	Fail	y	Maths	W
4	Pass	Y	CS	NW
5	Fail	N	Other	W
6	Fail	Y	Other	W
7	Pass	Y	Maths	NW
8	Pass	Y	CS	NW
9	Pass	n	Maths	W
10	Pass	n	CS	W
11	Pass	y	CS	W
12	Pass	n	Maths	NW
13	Fail	y	Other	W
14	Fail	n	Other	NW
15	Fail	n	Maths	W

CART Algorithm

- ✔ The CART (Classification and Regression Trees) algorithm is a decision tree learning algorithm that can be used for both classification and regression problems in machine learning. It works by recursively partitioning the training data into smaller subsets using binary splits.
- ✔ CART is a powerful and popular algorithm due to its ability to handle both categorical and continuous features,
- ✔ CART algorithm is developed by Leo Breiman and Jerome Friedman.
- ✔ It uses the Gini impurity measure for classification trees and the mean squared error for regression trees.
- ✔ CART algorithm always creates a binary tree, which means that each non-terminal node has two child nodes. This is in contrast with other tree-based methods, which may allow multiple child nodes

Gini Index

- ✔ Steps 1 Calculate the Gini impurity for the entire dataset. This is the impurity of the root node.
- ✔ Steps 2 For each input variable, calculate the Gini impurity for all possible split points. The split point that results in the minimum Gini impurity is chosen.
- ✔ Steps 3 The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
- ✔ Steps 4 Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
- ✔ Steps 5 The resulting tree is the decision tree.
- ✔ Gini impurity is a measure of the quality of a split in a decision tree algorithm. Given a sample, the Gini impurity measures the probability of a misclassification if a label is randomly chosen using the probability distribution of the branch.
- ✔ The Gini impurity of a node can be calculated as follows:
  - **Gini = $1 - \sum(p_i)^2$**
  - Here, p_i is the probability that a tuple in D belongs to the class C_i.
- ✔ The index reaches its minimum (0.0) when all the samples of a node are classified into a single category.

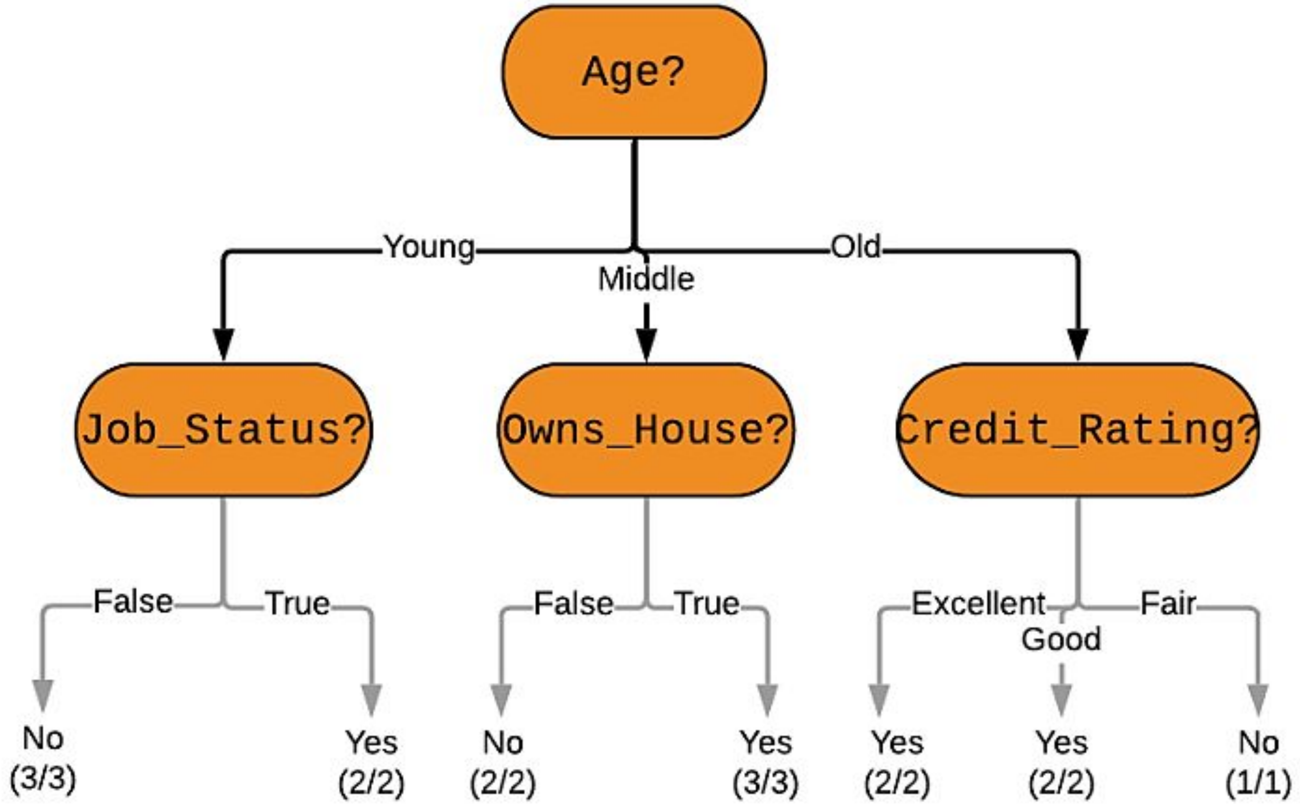
Example

✔ Here is the goal:
Use the model to classify future loan applications into one of these classes:

- Yes (approved) and
- No (not approved)

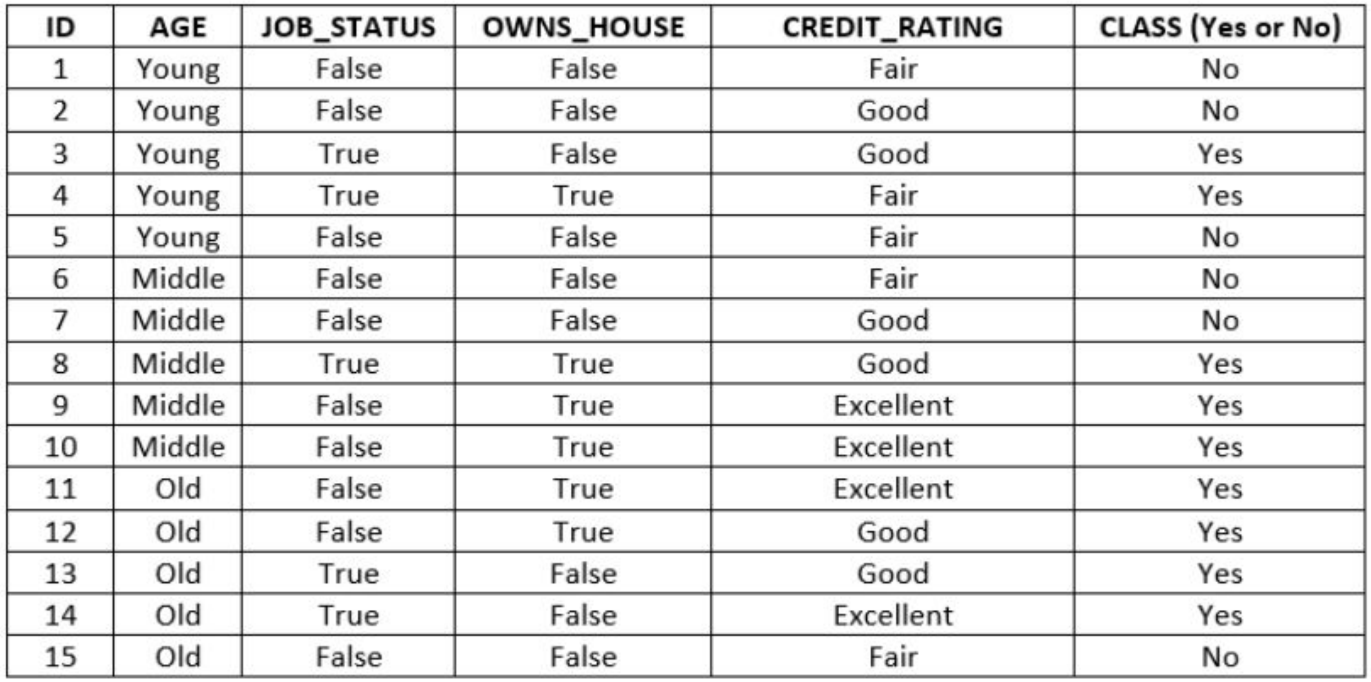
✔ Let us clasify the following applicant whose record looks like this:
Age: Young
Job_Status: False
Owns_House: False
Credit_Rating: Good

What would a decision tree look like for the above problem?
Here is one example of a decision tree -



Find Gini Index

Our Data: Loan Approval Prediction



ID	AGE	JOB_STATUS	OWNS_HOUSE	CREDIT_RATING	CLASS (Yes or No)
1	Young	False	False	Fair	No
2	Young	False	False	Good	No
3	Young	True	False	Good	Yes
4	Young	True	True	Fair	Yes
5	Young	False	False	Fair	No
6	Middle	False	False	Fair	No
7	Middle	False	False	Good	No
8	Middle	True	True	Good	Yes
9	Middle	False	True	Excellent	Yes
10	Middle	False	True	Excellent	Yes
11	Old	False	True	Excellent	Yes
12	Old	False	True	Good	Yes
13	Old	True	False	Good	Yes
14	Old	True	False	Excellent	Yes
15	Old	False	False	Fair	No

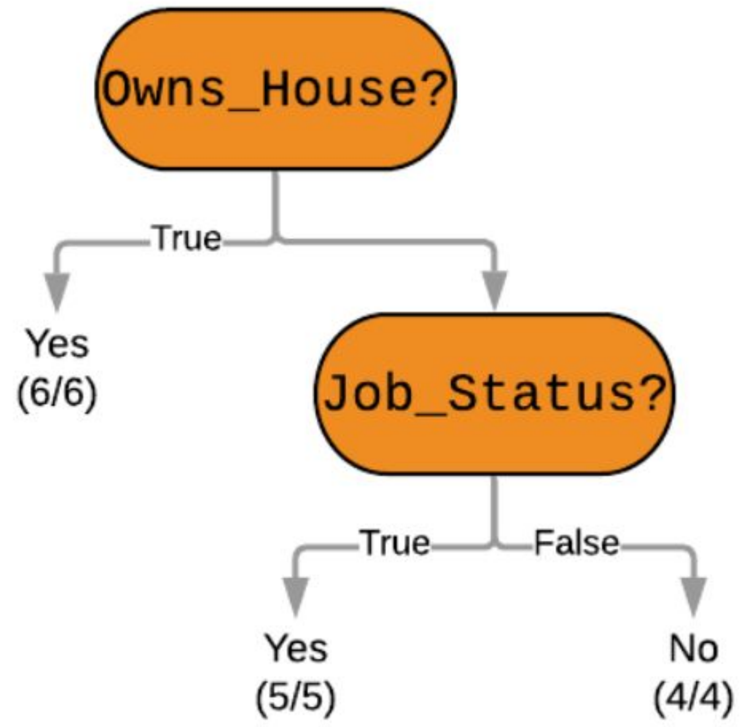
- ✔ **Step 1: Find Gini(D)**
  In our case, $\text{Gini}(D) = 1 - (9/15)^2 - (6/15)^2 = 0.48$
  [since we have 9 Yes and 6 No class values
  for the 15 total records]
- ✔ **Step 2: Find $\text{Gini}_k(D)$ for each attribute k**
  $\text{Gini}_k(D) = \sum|D_i|/|D| * \text{Gini}(D_i)$
  For $i = 1$ to n, where D_i is a partition of the dataset D
- ✔ So, for the attribute Owns_House:
  $\text{Gini}_{\text{Owns\_House}}(D) = (5/15) * \text{Gini}(D_{\text{Owns\_House = True}}) + (10/15) * \text{Gini}(D_{\text{Owns\_House = False}})$
- ✔ $= (5/15) * [1 - (5/5)^2 - (0/5)^2] + (10/15) * [1 - (3/10)^2 - (7/10)^2]$
- ✔ $= 0.28$
- ✔ And for the Age attribute:
  $\text{Gini}_{\text{Age}}(D) = (5/15) * \text{Gini}(D_{\text{Age = Young}}) + (5/15) * \text{Gini}(D_{\text{Age = Middle}}) + (5/15) * \text{Gini}(D_{\text{Age = Old}})$
  $= 0.33$
- ✔ Similarly, we can find the Gini Indices for the other attributes:
  $\text{Gini}_{\text{Credit\_Rating}}(D) = 0.634$
  $\text{Gini}_{\text{Job\_Status}}(D) = 0.32$
  (try calculating these yourself!)

Our Data: Loan Approval Prediction

ID	AGE	JOB_STATUS	OWNS_HOUSE	CREDIT_RATING	CLASS (Yes or No)
1	Young	False	False	Fair	No
2	Young	False	False	Good	No
3	Young	True	False	Good	Yes
4	Young	True	True	Fair	Yes
5	Young	False	False	Fair	No
6	Middle	False	False	Fair	No
7	Middle	False	False	Good	No
8	Middle	True	True	Good	Yes
9	Middle	False	True	Excellent	Yes
10	Middle	False	True	Excellent	Yes
11	Old	False	True	Excellent	Yes
12	Old	False	True	Good	Yes
13	Old	True	False	Good	Yes
14	Old	True	False	Excellent	Yes
15	Old	False	False	Fair	No

Partition at Minimum Gini Index Values

- ✔ **Step 3: Partition at Minimum Gini Index Values** As stated above, we need the root node of the decision tree to have the lowest possible Gini Index, and in our case that is the attribute Owns_House. The next attribute is Job_Status.
- ✔ This is how we ultimately arrive at this decision tree –
- ✔ And using this decision tree for our problem - we can see that the applicant does not own a house, and does not have a job. Unfortunately, his loan will not be approved.



Example.

Table থেকে loan approval এর decision tree বের করো using CART algorithm.

S-1 Gini for Root

$$\mathrm{Gini}(D) = 1-(9/15)^2-(6/15)^2$$

$$= 0.48$$

here yes - 9টি
NO - 6টি total - 15টি

S-2
Find $\mathrm{Gini}_k(D)$ for each attribute k.

$$\mathrm{Gini}_k(D) = \sum |D_i|/|D| * \mathrm{Gini}(D_i)$$

$i = 1$ to n, D_i is partition.

Own house

$$\mathrm{Gini}_{\text{own\_House}}(D) = (6/15) * \mathrm{Gini}(D_{\text{own\_House-True}})$$

$$+ (9/15) * \mathrm{Gini}(D_{\text{own-house-no}})$$

$$= (6/15) * \left[1-\left(\frac{6}{6}\right)^2-(0/6)^2\right]$$

$$+ \frac{9}{15} * \left(1-\left(\frac{3}{9}\right)^2-\left(\frac{6}{9}\right)^2\right]$$

$$= 0.26667 \approx 0.27$$

Age

$$\text{Gini}_{age}(D) = \left\{\frac{5}{15} \times \left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right)\right\} + \left\{\frac{5}{15}\left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right)\right\}$$

$$+ \left\{\frac{5}{15} \times \left(1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2\right)\right\}$$

$$= 0.42667$$

$$\text{Gini}_{Credit-rating}(D) = \frac{5}{15}\left(1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2\right)$$

$$+ \frac{6}{15}\left(1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2\right)$$

$$+ \frac{4}{15}\left(1 - \left(\frac{4}{4}\right)^2 - 0\right)$$

$$= 0.28444$$

$$\text{Gini}_{Job-status} = \overset{2}{\cancel{\frac{10}{15}}}_{3}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{5}{15}\left(1 - \left(\frac{5}{5}\right)^2 - \left(\frac{0}{5}\right)^2\right.$$

$= 0.32$

Job Status , —

False
False

iD	Age	Job-status	own-house	credit	class
1	young	False	False	Fair	NO
	young	False	False	Good	NO
	young	True	False	Good	yes
	young	False	False	Fair	NO
	middle	False	False	Fair	NO
	middle	False	False	Good	NO
	old	True	False	Good	yes
	old	True	False	excelent	yes
	old	False	False	Fair	NO

~~Job-Status~~ Age

$$= \frac{4}{9}\left(1-\left(\frac{3}{4}\right)^2-\left(\frac{1}{4}\right)^2\right)+\frac{2}{9}\left(1-\left(\frac{2}{2}\right)^2-\left(\frac{0}{2}\right)^2\right)$$

$$+\frac{3}{9}\left(1-\left(\frac{2}{3}\right)^2-\left(\frac{1}{3}\right)^2\right)$$

$$= 0'3148$$

$$G_{status} = \frac{6}{9}\left(1-\left(\frac{6}{6}\right)^2-\left(\frac{0}{6}\right)^2\right) + \frac{3}{9}\left(1-\left(\frac{3}{3}\right)^2-\left(\frac{0}{3}\right)^2\right)$$

$$= 0$$

$$G_{credit} = \frac{4}{9}\left(1-\left(\frac{4}{4}\right)^2-\left(\frac{0}{4}\right)^2\right) + \frac{4}{9}\left(1-\left(\frac{2}{4}\right)^2-\left(\frac{2}{4}\right)^2\right)$$

$$+ \frac{1}{9}\left(1-\left(\frac{1}{1}\right)^2-0\right)$$

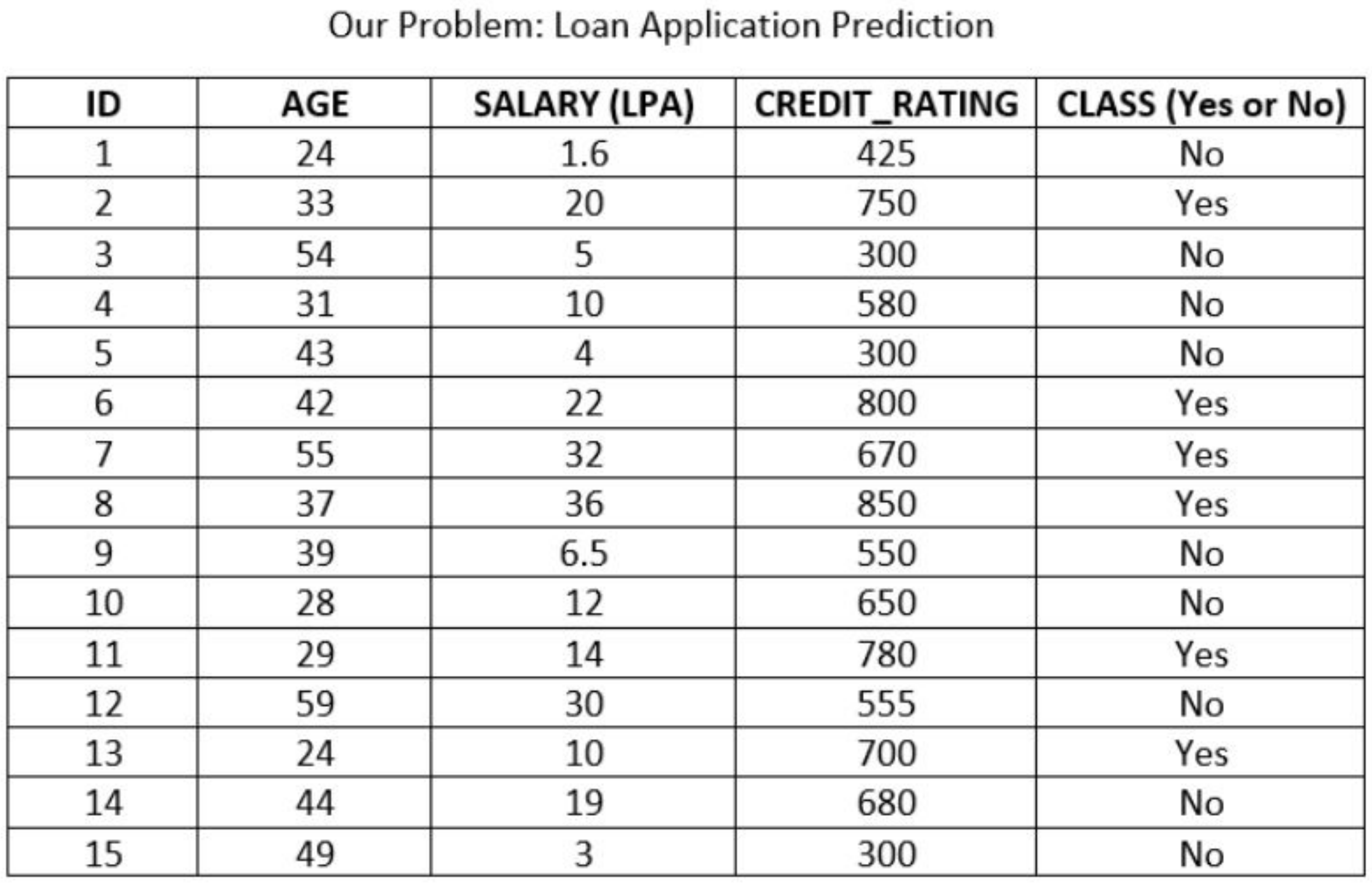
$$= 0.222$$

Prediction using CARTs

- ✔ We saw that decision trees can be classified into two types:
  Classification trees which are used to separate a dataset into different classes (generally used when we expect categorical classes). The other type are Regression Trees which are used when the class variable is continuous (or numerical).
- ✔ Our focus is to learn a target function that can be used to predict the values of a continueous class attribute, i.e. this is a prediction problem where we want to know if, given labelled training data, something falls into one class or another.
- ✔ Let us continue with the Loan Application approval problem, but add an attribute for each applicant - Salary. This is a continuous attribute ranging from zero (for applicants with no jobs), to near infinity. The age and credit rating attributes can also be represented with actual numerical values, instead of the categories they were before. Our new dataset can look like this:

Prediction using CARTs

- ✔ Now we can build a decision tree to predict loan approval on age, salary, credit rating, and various other variables. In this case, we are predicting values for the continuous variables. And the same CART algorithm we discussed above can be applied here. This is the biggest difference between CART and C4.5
- ✔ C4.5 cannot support numerical data and hence cannot be used for regression (prediction problems).



Our Problem: Loan Application Prediction

ID	AGE	SALARY (LPA)	CREDIT_RATING	CLASS (Yes or No)
1	24	1.6	425	No
2	33	20	750	Yes
3	54	5	300	No
4	31	10	580	No
5	43	4	300	No
6	42	22	800	Yes
7	55	32	670	Yes
8	37	36	850	Yes
9	39	6.5	550	No
10	28	12	650	No
11	29	14	780	Yes
12	59	30	555	No
13	24	10	700	Yes
14	44	19	680	No
15	49	3	300	No

Pruning: Getting an Optimal Decision tree

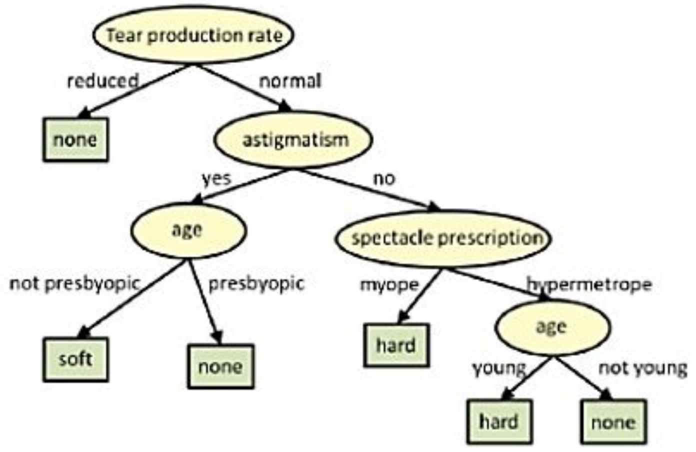
- ✔ Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.
- ✔ A too-large tree increases the risk of **overfitting**, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning.
- ✔ **How to avoid/counter Overfitting in Decision Trees?**
- ✔ The common problem with Decision trees, especially having a table full of columns, they fit a lot. Sometimes it looks like the tree memorized the training data set. If there is no limit set on a decision tree, it will give you 100% accuracy on the training data set because in the worse case it will end up making 1 leaf for each observation. Thus this affects the accuracy when predicting samples that are not part of the training set.
- ✔ Here are two ways to remove overfitting:
  - Pruning Decision Trees.
  - Random Forest

Pruning

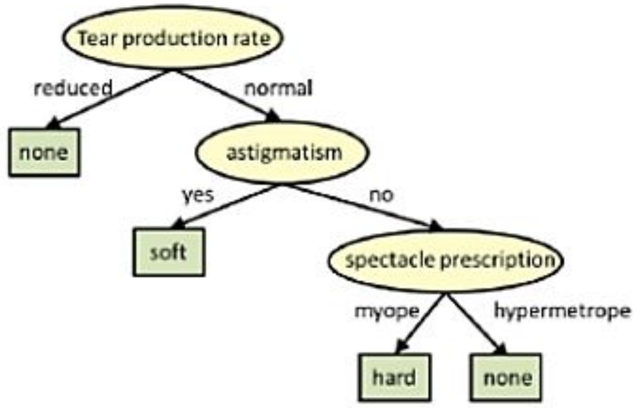
- ✔ Pruning is another method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. Additionally, it removes the branches which have very low importance.
- ✔ There are mainly 2 ways for pruning:
- ✔ **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance **while growing** the tree.
- ✔ **Post-pruning** – once our **tree is built to its depth**, we can start pruning the nodes based on their significance.

Pruning

- ✔ In **pruning**, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.



Original Tree



Pruned Tree

- ✔ In the above diagram, the 'Age' attribute in the left-hand side of the tree has been pruned as it has more importance on the right-hand side of the tree, hence removing overfitting.

Thank You