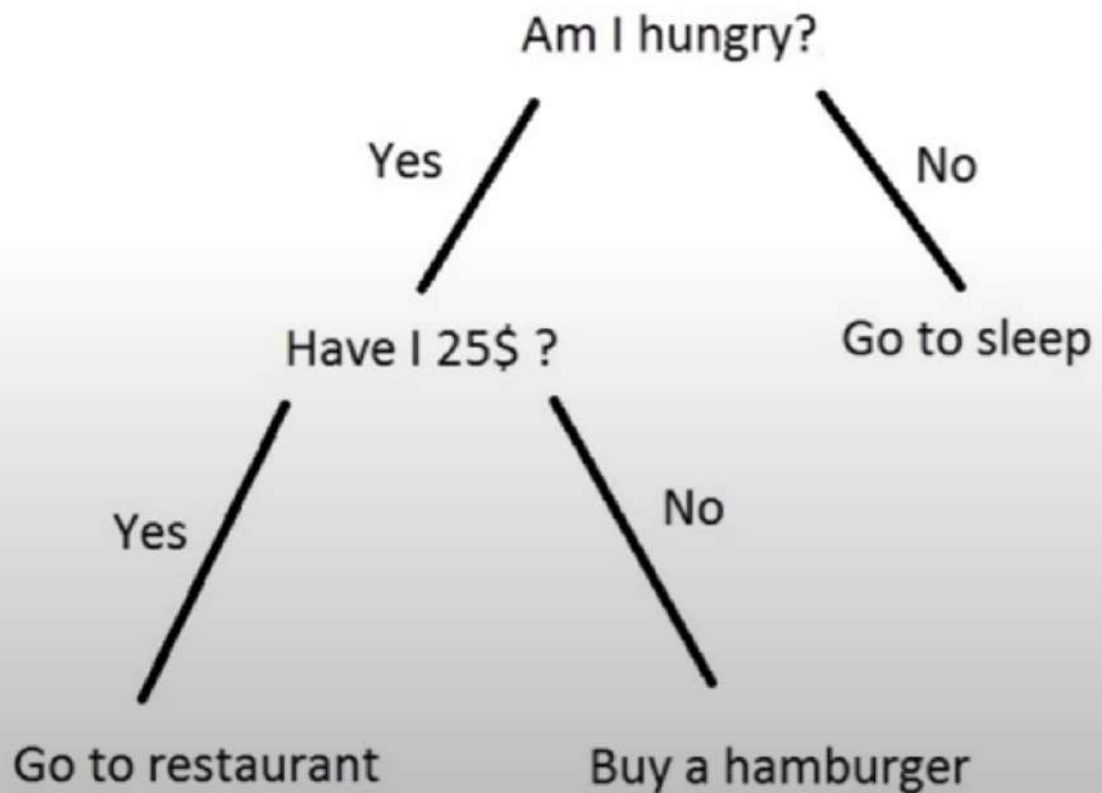
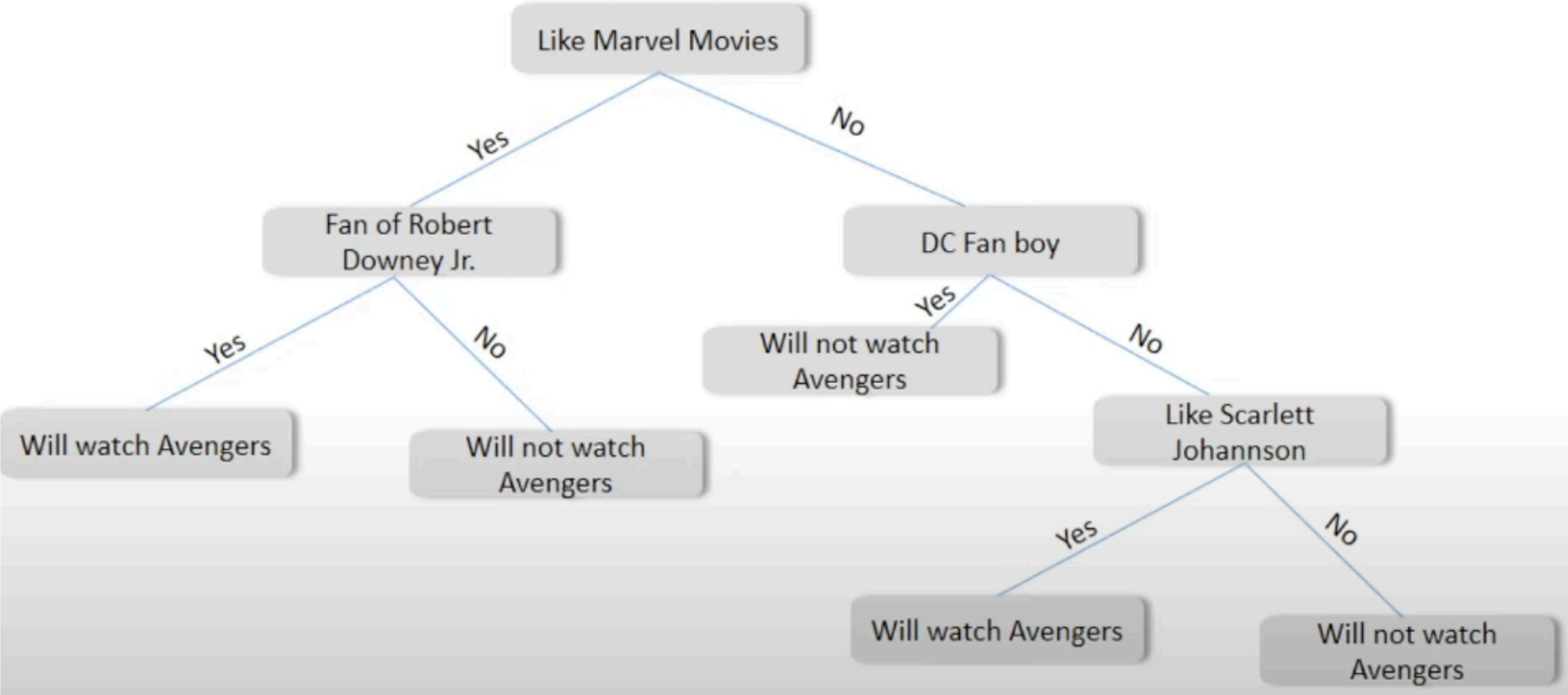
# Decision Tree

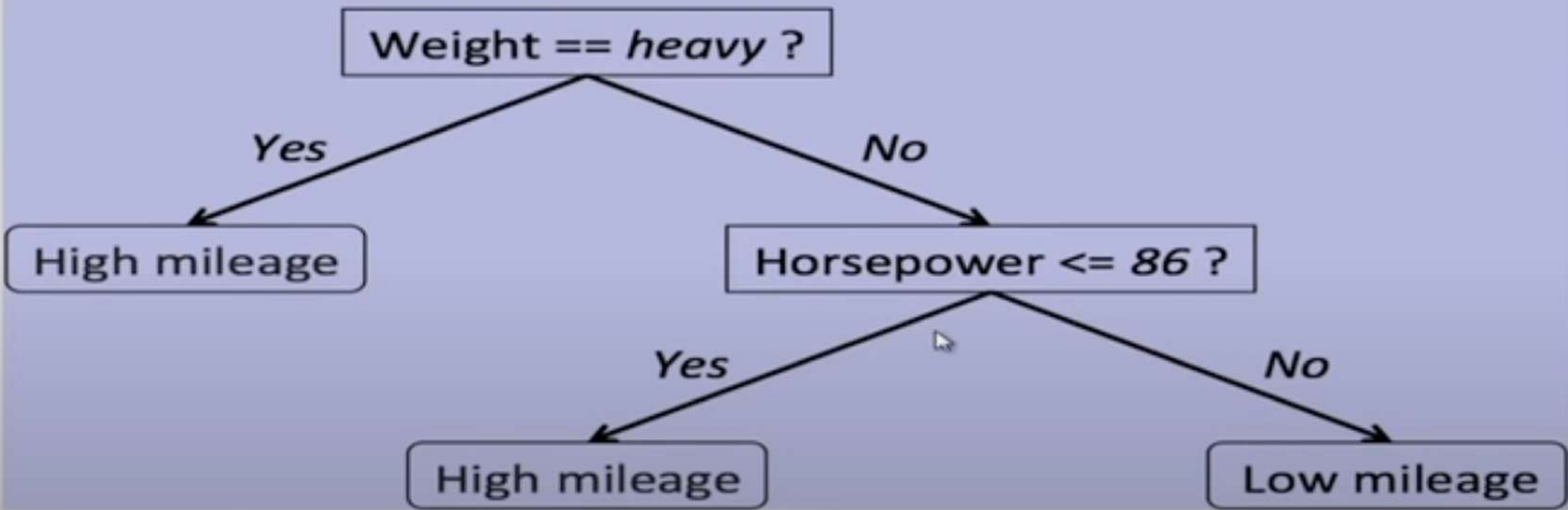
- Graphical representation of all the possible solutions to a decision
- Decisions are based on some conditions
- Decision made can be easily explained



## Decision Tree Algorithm is a supervised learning method used for both classification and regression



# Decision Tree Model
## for Car Mileage Prediction



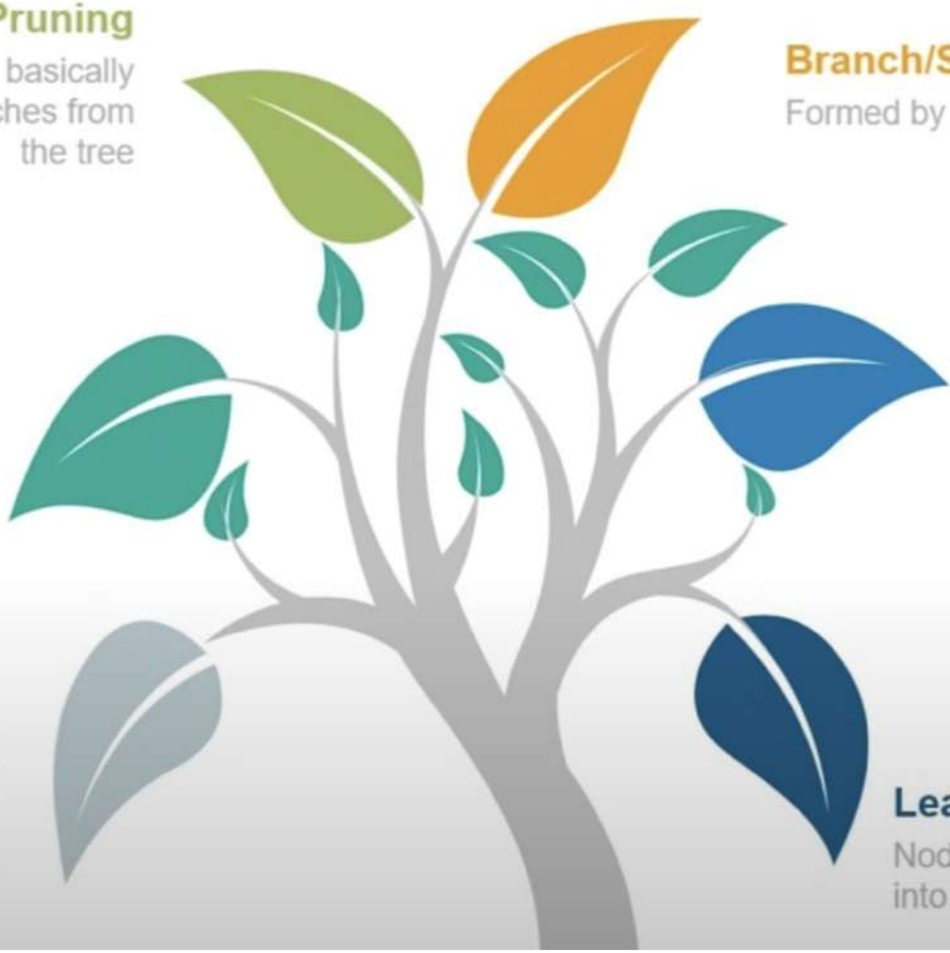
# Decision Tree Terminology

**Pruning**

Opposite of Splitting, basically removing unwanted branches from the tree

**Branch/SubTree**

Formed by splitting the tree/node



**Parent/Child Node**

Root node is the parent node and all the other nodes branched from it is known as child node

**Splitting**

Splitting is dividing the root node/sub node into different parts on the basis of some condition.

**Root Node**

It represents the entire  population or sample  and this further gets divided into two or more homogenous sets.

**Leaf Node**

Node cannot be further segregated into further nodes

# Issues

- Given some training examples, what decision tree should be generated?
- One proposal: prefer the smallest tree that is consistent with the data (Bias)
- Possible method:
  - search the space of decision trees for the smallest decision tree that fits the data

# Example Data

Training Examples:

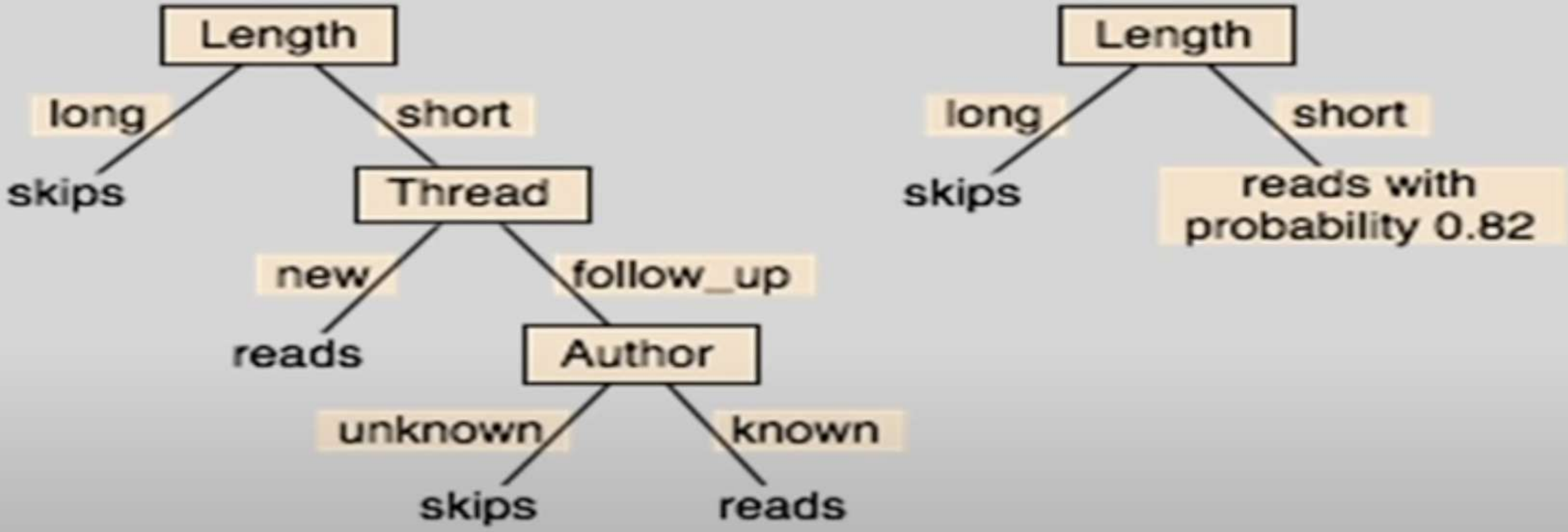
| | Action | Author | Thread | Length | Where |
|---|---|---|---|---|---|
| e1 | skips | known | new | long | Home |
| e2 | reads | unknown | new | short | Work |
| e3 | skips | unknown | old | long | Work |
| e4 | skips | known | old | long | home |
| e5 | reads | known | new | short | home |
| e6 | skips | known | old | long | work |

New Examples:

| | | | | | |
|---|---|---|---|---|---|
| e7 | ??? | known | new | short | work |
| e8 | ??? | unknown | new | short | work |

# Possible splits

# Two Example DTs

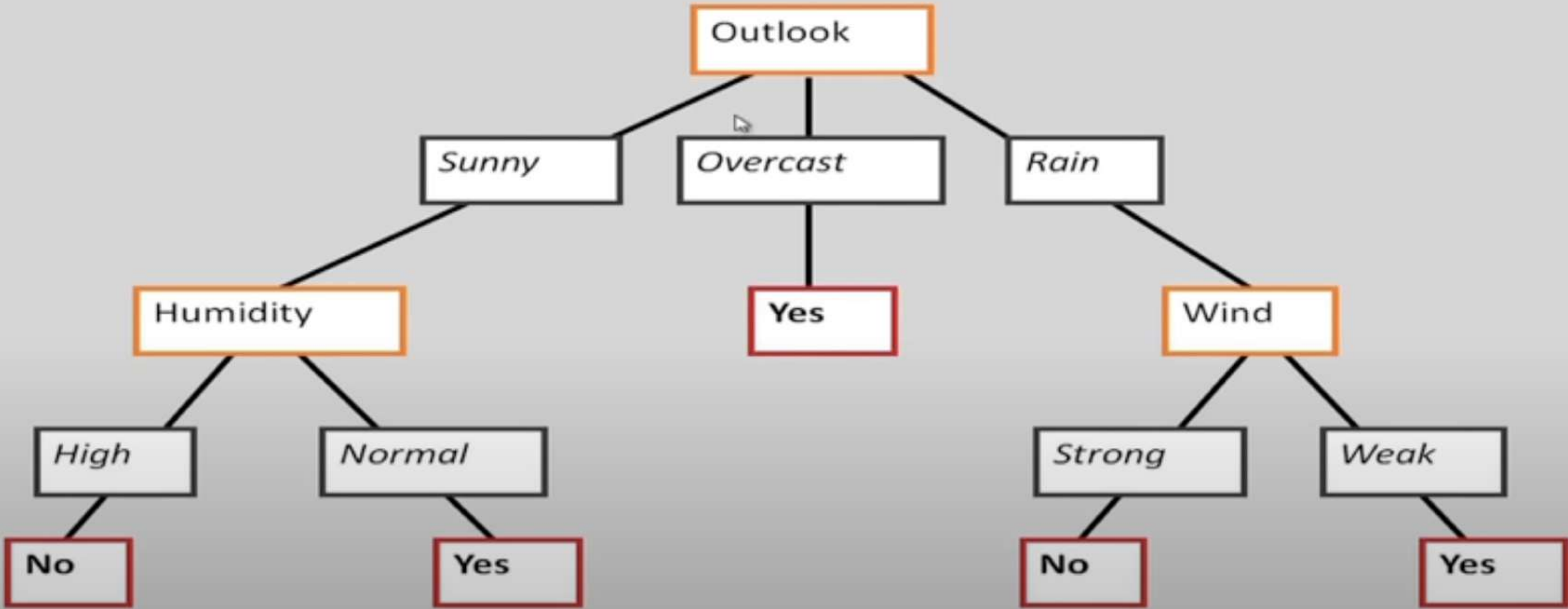


# Decision Tree for PlayTennis

- Attributes and their values:
  - Outlook: *Sunny, Overcast, Rain*
  - Humidity: *High, Normal*
  - Wind: *Strong, Weak*
  - Temperature: *Hot, Mild, Cool*

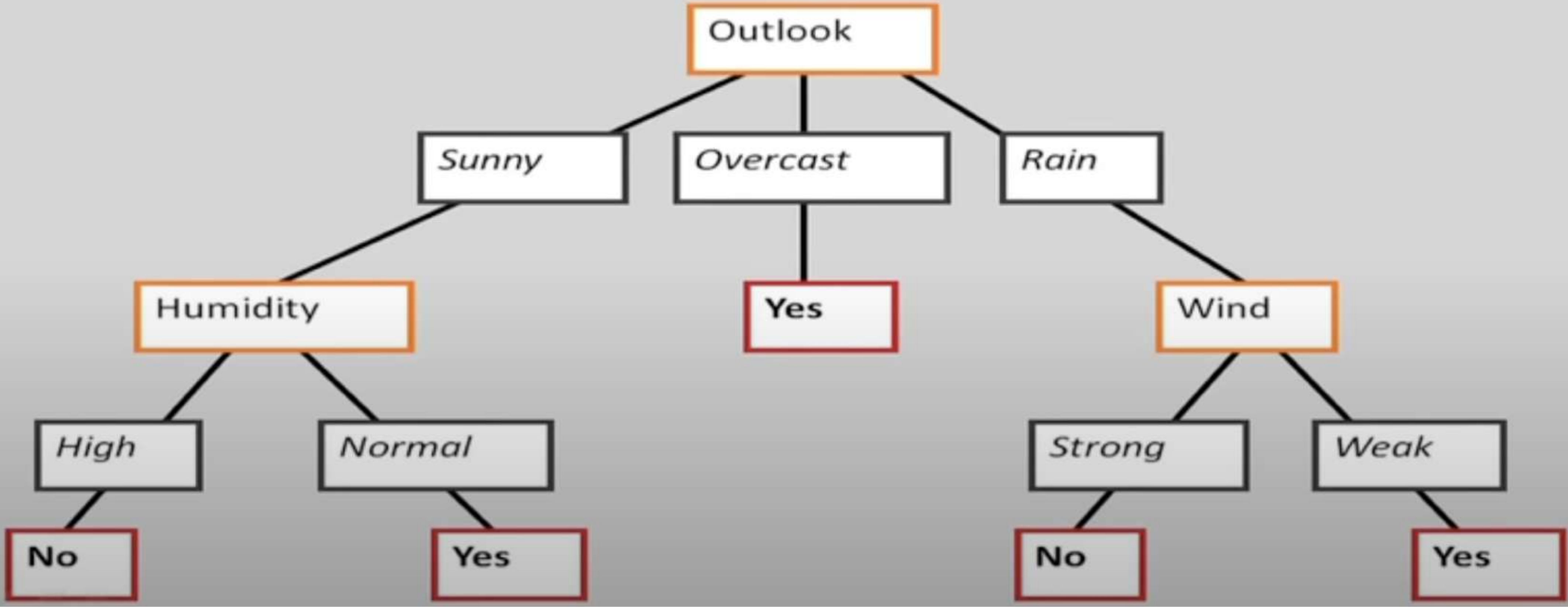
- Target concept - Play Tennis: *Yes, No*

# Decision Tree for PlayTennis

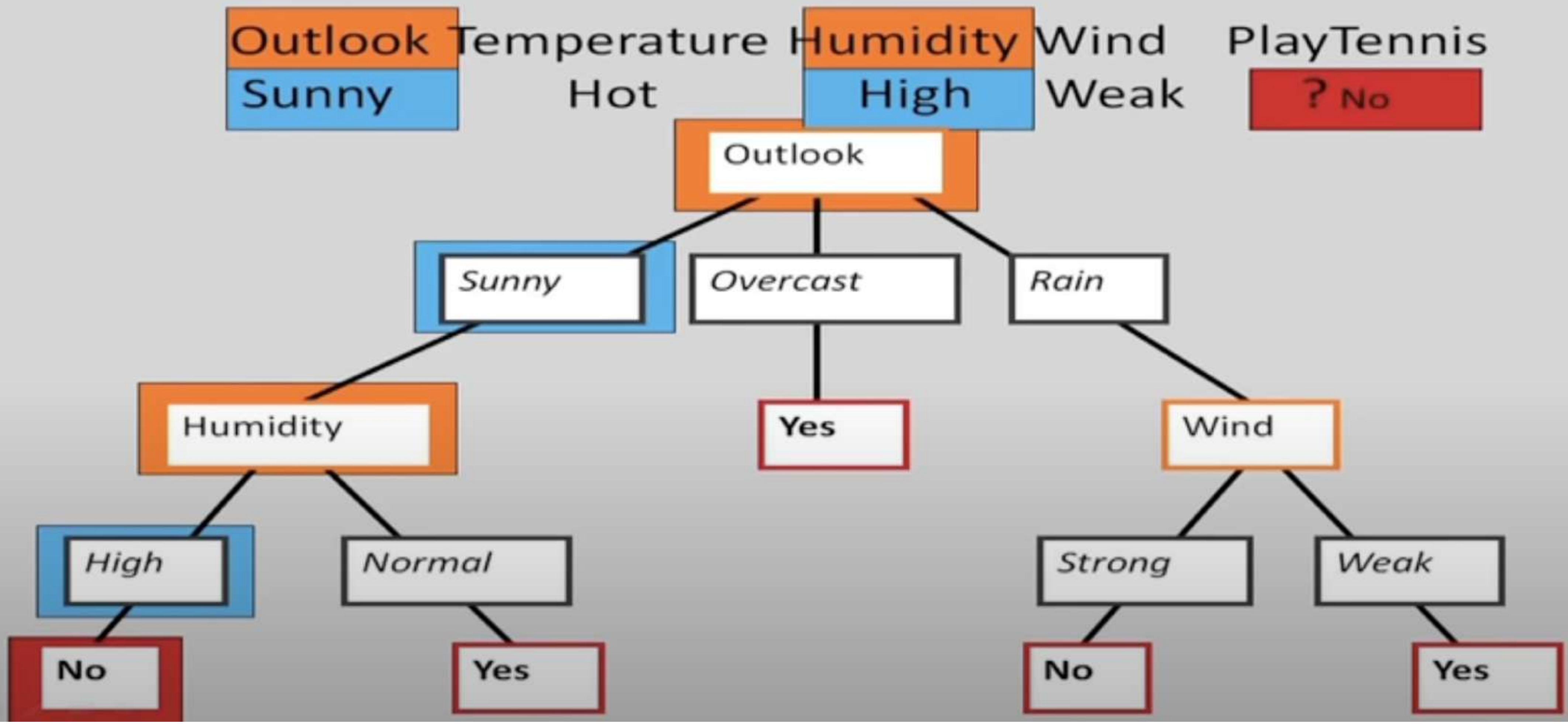


# Decision Tree for PlayTennis

| Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|
| Sunny | Hot | High | Weak | ? |

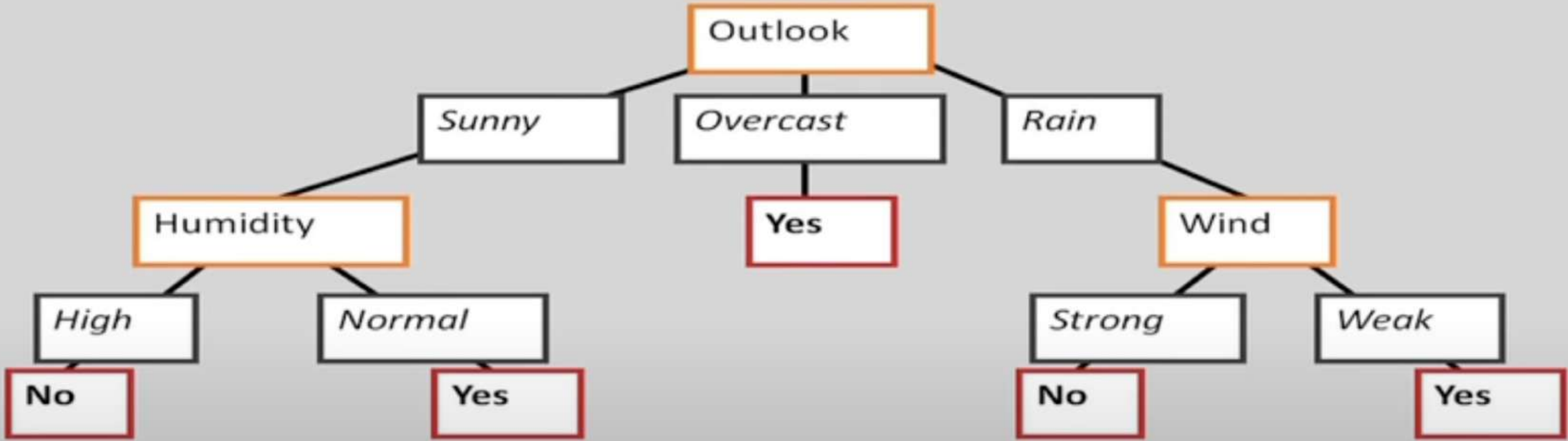


# Decision Tree for PlayTennis



# Decision Tree

decision trees represent disjunctions of conjunctions




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

# Searching for a good tree

- The space of decision trees is too big for systematic search.
- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values
- Choose a test (e.g. an input feature) to split on.
  - For each value of the test, build a subtree for those examples with this value for the test.

# Top-Down Induction of Decision Trees ID3

1. A $\leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

# Top-Down Induction of Decision Trees ID3

1. Which node to proceed with?

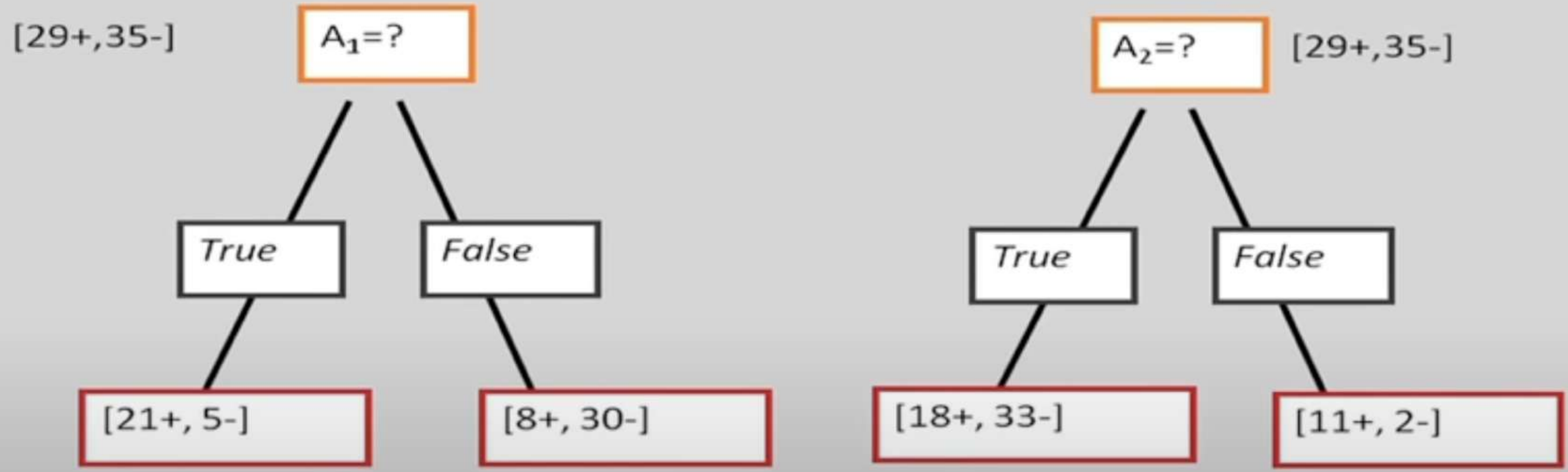
1. A $\leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

2. When to stop?

# Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split
- Which test to split on
  - split gives smallest error.
  - With multi-valued features
    - split on all values or
    - split values into half.

# Which Attribute is "best"?



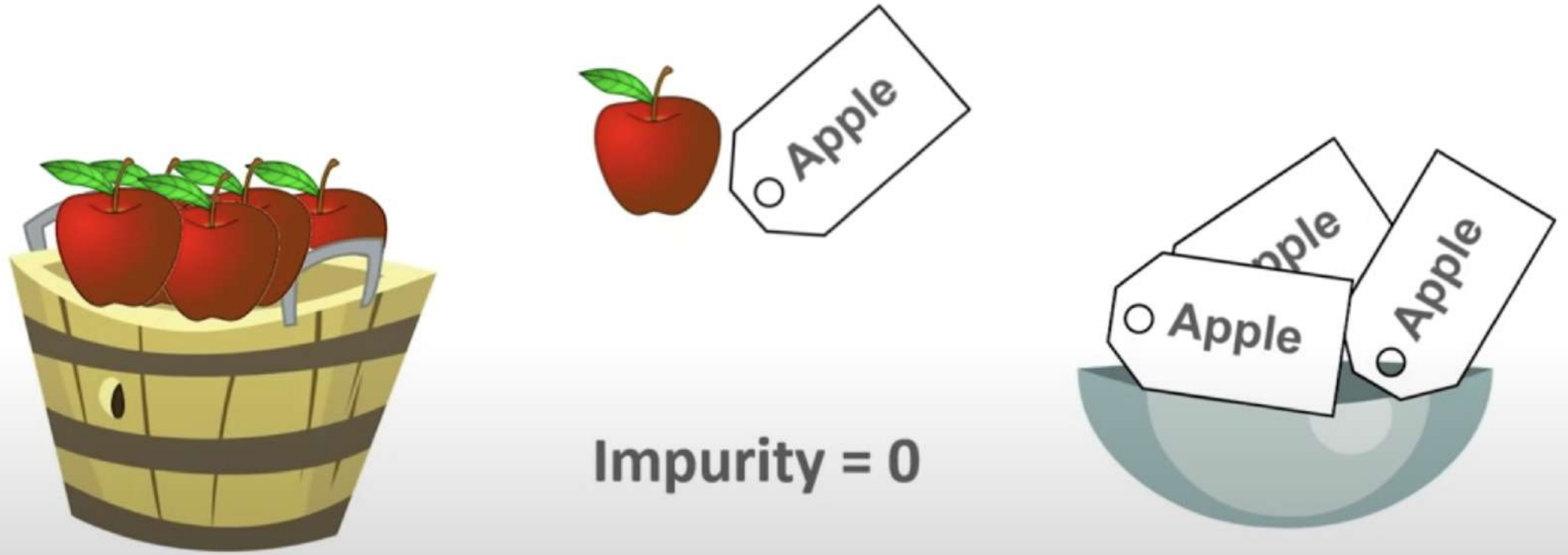
# Principled Criterion

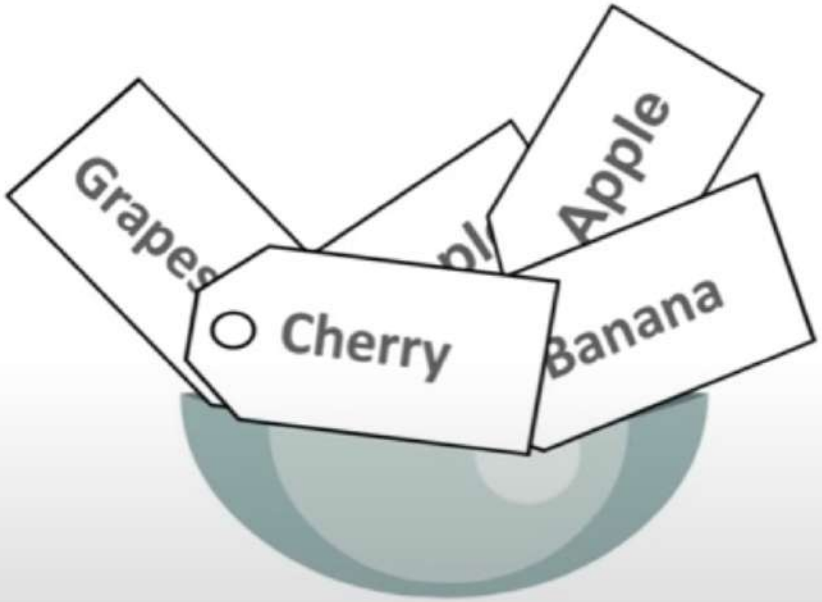
- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)

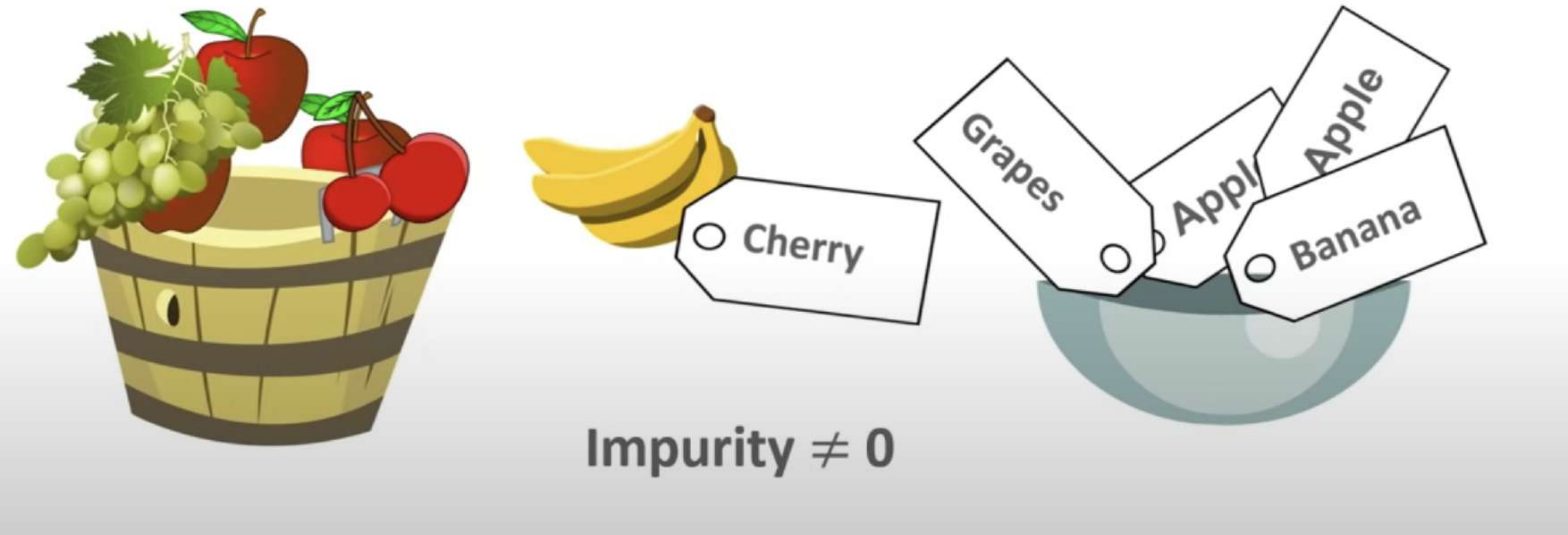
# Entropy

- A measure for
  - uncertainty
  - purity
  - information content
- Information theory: optimal length code assigns ($-\log_2 p$) bits to message having probability $p$
- $S$ is a sample of training examples
  - $p_+$ is the proportion of positive examples in $S$
  - $p_-$ is the proportion of negative examples in $S$
- Entropy of $S$: average optimal number of bits to encode information about certainty/uncertainty about $S$

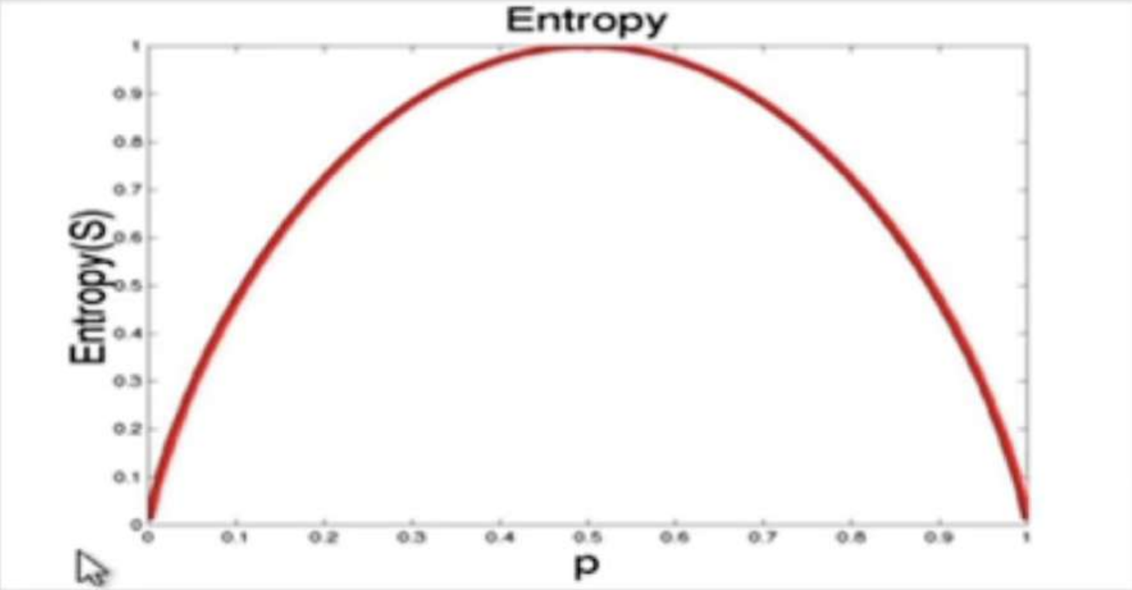
$$Entropy(S) = p_+(-\log_2 p_+) + p_-(-\log_2 p_-) = -p_+\log_2 p_+ - p_-\log_2 p_-$$


Apple
Apple
Apple
Impurity = 0


Grapes
Apple
Cherry
Banana


Cherry
Grapes
Appl
Apple
Banana
Impurity ≠ 0

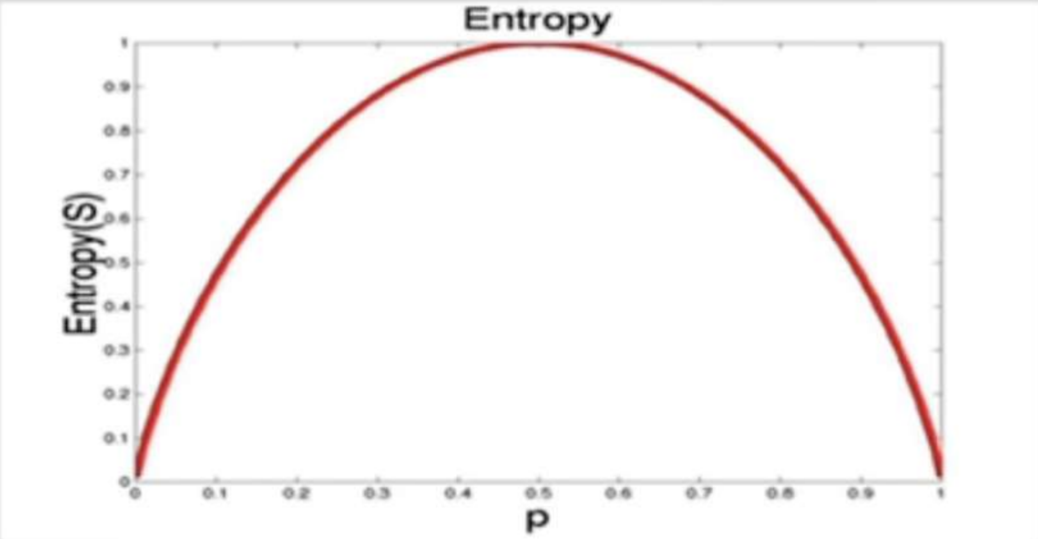
# Entropy




- S is a sample of training examples
- $p_+$ is the proportion of positive examples
- $p_-$ is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

# Entropy




- The entropy is 0 if the outcome is ``certain".
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

- S is a sample of training examples
- $p_+$ is the proportion of positive examples
- $p_-$ is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy(S)} = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

# Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$\text{Gain}(S,A)=\text{Entropy}(S) - \sum_{v\in \text{values}(A)} |S_v|/|S| \ \text{Entropy}(S_v)$$

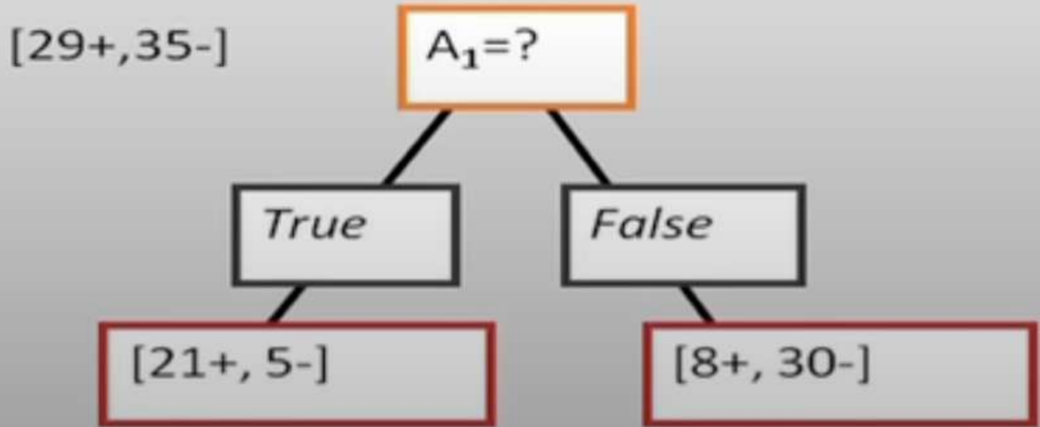
$$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$
$$= 0.99$$

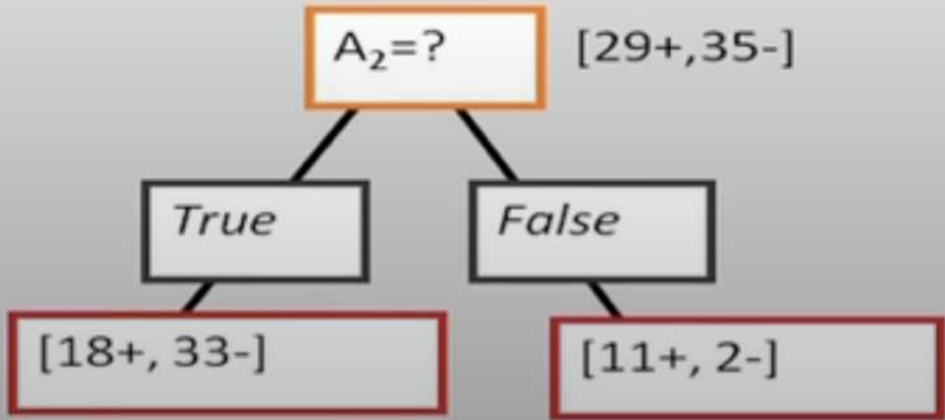
# Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} |S_v|/|S| \ \text{Entropy}(S_v)$$

$$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$
$$= 0.99$$



# Information Gain

Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain(S,$A_1$)=Entropy(S)

-26/64*Entropy([21+,5-])

-38/64*Entropy([8+,30-])

=0.27

Entropy([18+,33-]) = 0.94

Entropy([8+,30-]) = 0.62

Gain(S,$A_2$)=Entropy(S)

-51/64*Entropy([18+,33-])

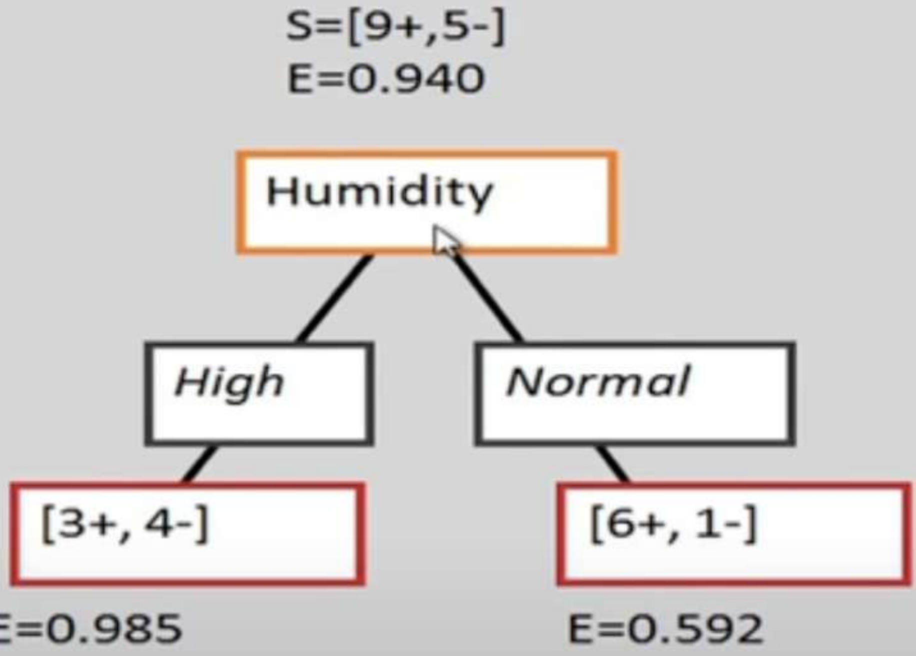
-13/64*Entropy([11+,2-])

=0.12

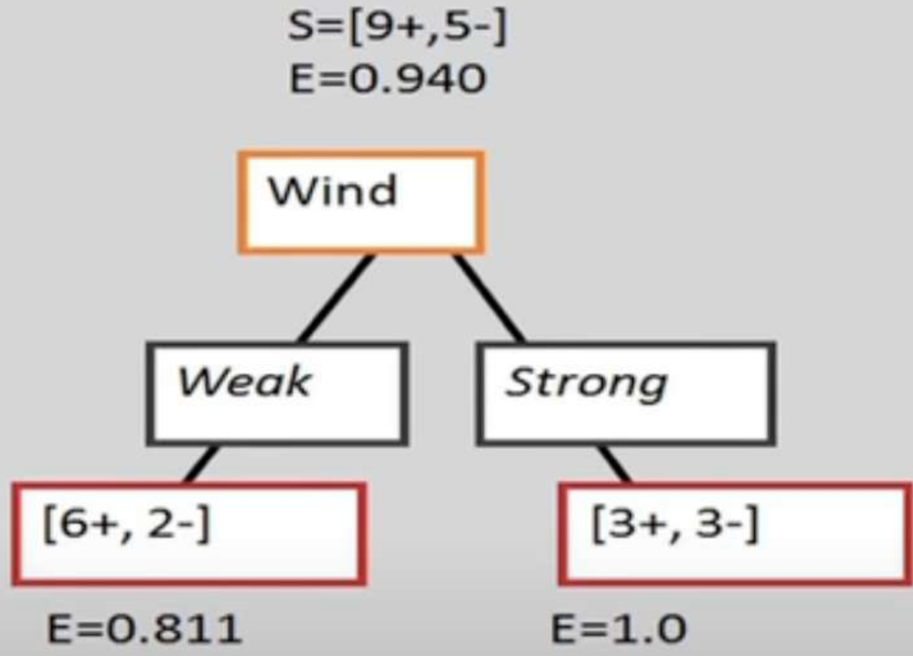
# Training Examples

| Day | Outlook | Temp | Humidity | Wind | Tennis? |
|---|---|---|---|---|---|
| *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* |

# Selecting the Next Attribute



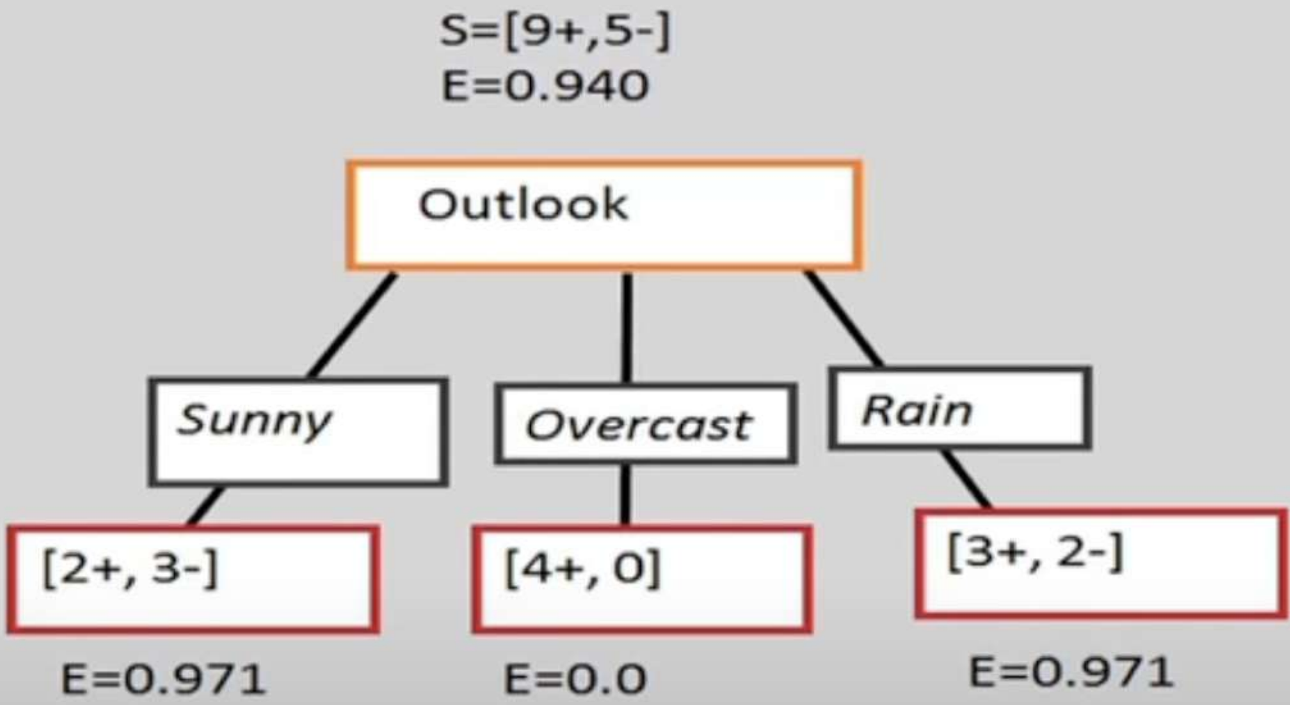

Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151




Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

# Selecting the Next Attribute




Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

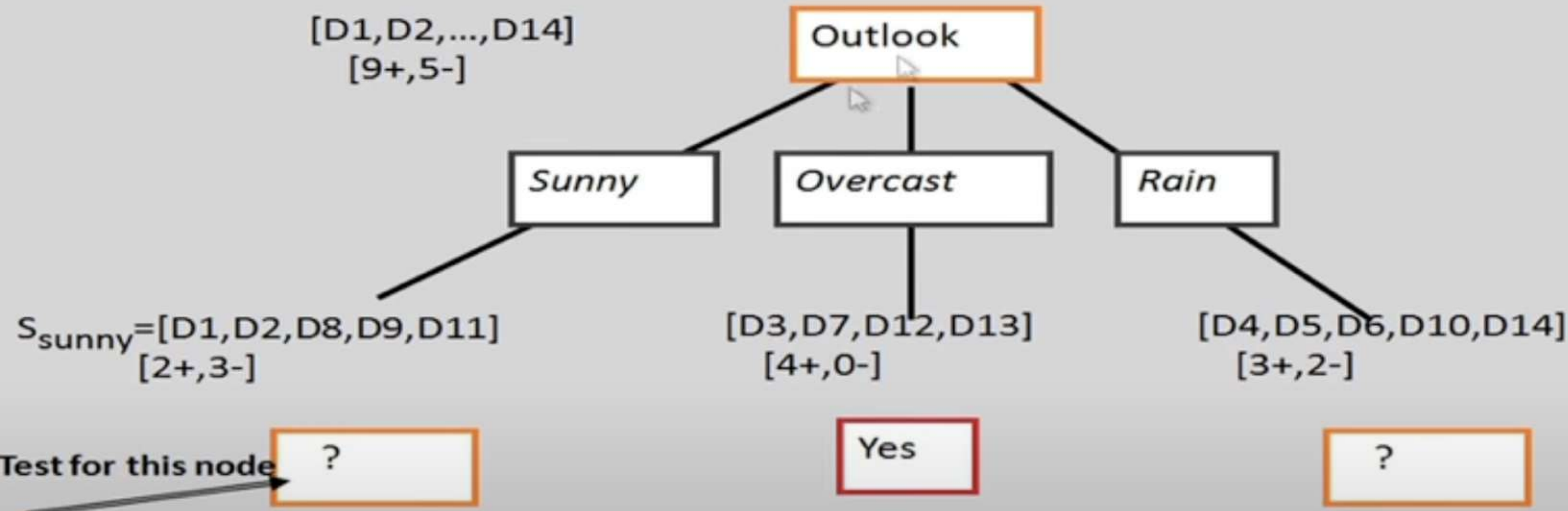
# Selecting the Next Attribute

The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

# ID3 Algorithm




$\text{Gain}(S_{sunny}, \text{Humidity}) = 0.970 - (3/5)0.0 - 2/5(0.0) = 0.970$

$\text{Gain}(S_{sunny}, \text{Temp.}) = 0.970 - (2/5)0.0 - 2/5(1.0) - (1/5)0.0 = 0.570$

$\text{Gain}(S_{sunny}, \text{Wind}) = 0.970 = -(2/5)1.0 - 3/5(0.918) = 0.019$

# Splitting Rule: GINI Index

- GINI Index
  - Measure of node impurity

$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

# How Does A Tree Decide Where To Split?

**Gini Index**

The measure of impurity (or purity) used in building decision tree in CART is Gini Index

**Information Gain**

The information gain is the decrease in entropy after a dataset is split on the basis of an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain

**Chi Square**

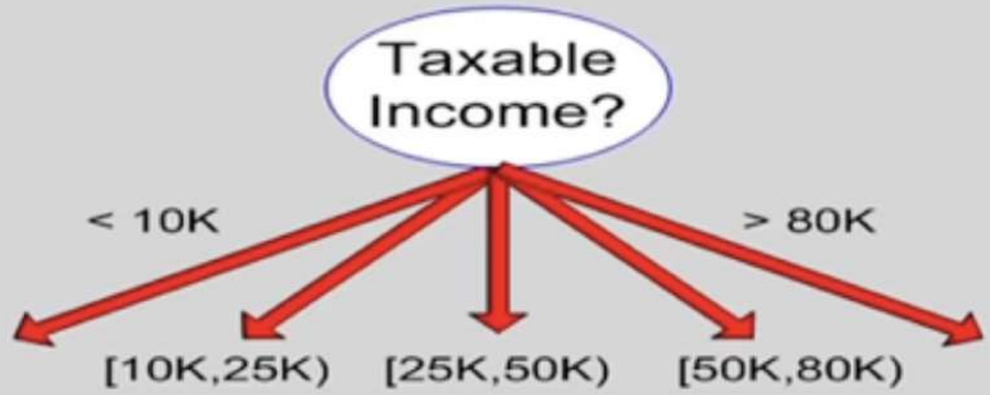
It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node

**Reduction in Variance**

Reduction in variance is an algorithm used for continuous target variables (regression problems). The split with lower variance is selected as the criteria to split the population

# Splitting Based on Continuous Attributes


(i) Binary split




(ii) Multi-way split

# Continuous Attribute – Binary Split

- For continuous attribute
  - Partition the continuous value of attribute A into a discrete set of intervals
  - Create a new boolean attribute $A_c$ , looking for a threshold c,

$$A_c = \begin{cases} true & \text{if } A_c < c \\ false & \text{otherwise} \end{cases}$$

How to choose c ?

- consider all possible splits and finds the best cut

## Random Forest

- Builds multiple decision trees and merges them together
- More accurate and stable prediction
- Random decision forests correct for decision trees' habit of overfitting to their training set
- Trained with the “bagging” method