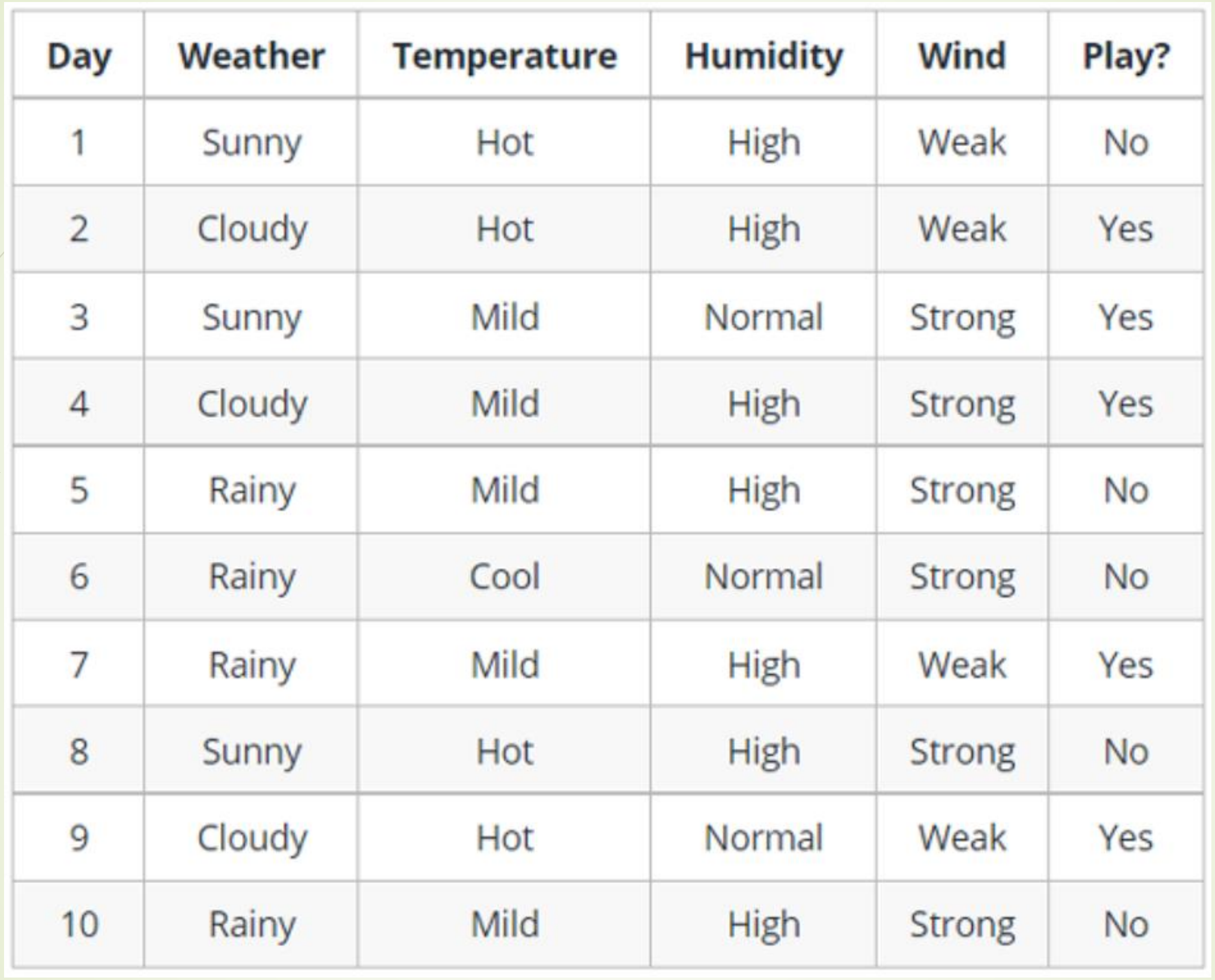
CAP 4630 – Decision Trees

Instructor: Aakash Kumar

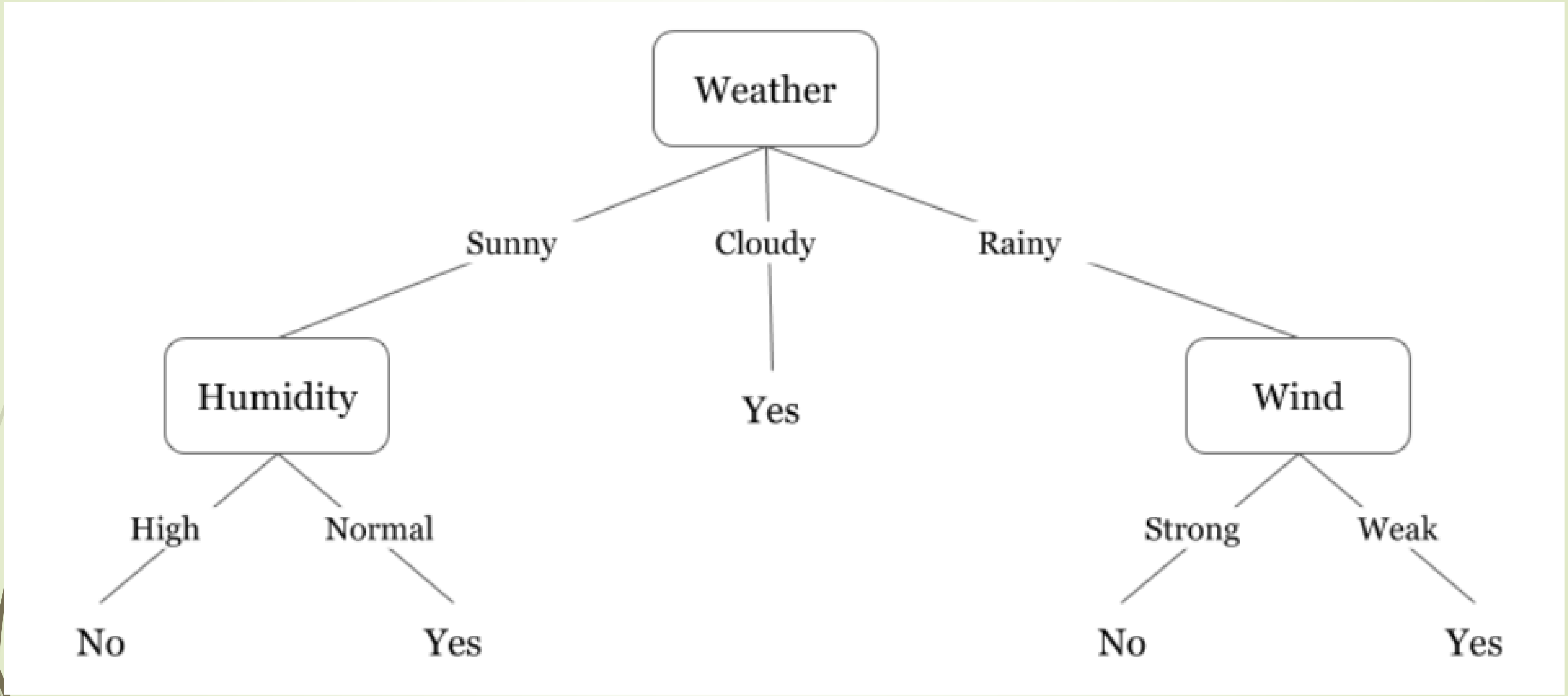
University of Central Florida

Introduction to Decision Trees

- **Decision Tree** is a simple yet powerful **supervised learning** algorithm used for classification tasks.
- It is structured like a tree, where:
  - Each **branch node** represents a decision or test based on feature values.
  - Each **leaf node** represents the final decision or classification.
- **Internal nodes** (also called test nodes) have outgoing edges representing different options.
- Leaf nodes (also called **terminal nodes**) represent the final outcome or classification



Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No


Weather
Sunny
Cloudy
Rainy
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

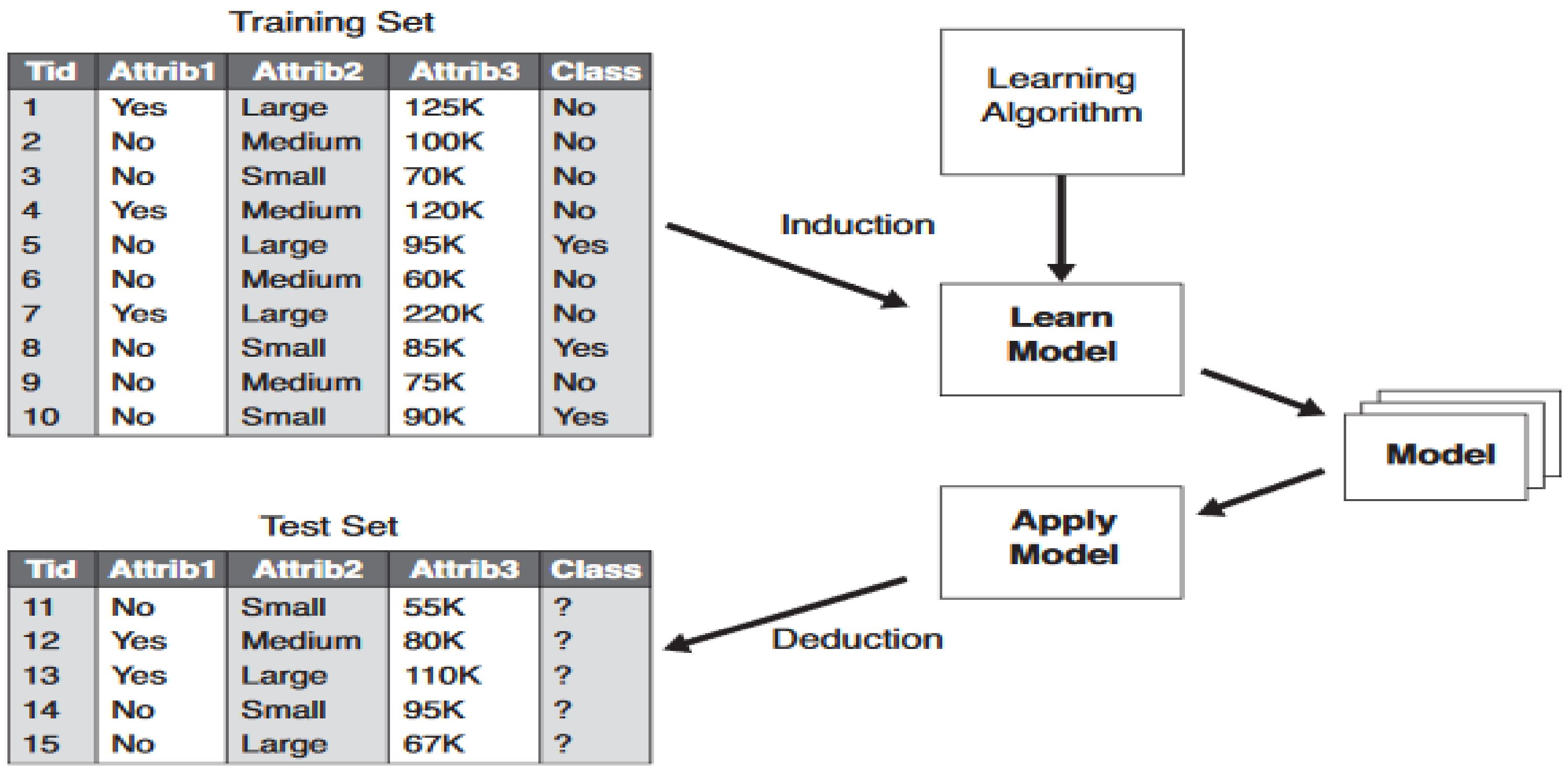
Construction of a Decision Tree

- Select the Best Root Attribute:
  - Test all attributes and choose the one that provides the best split to serve as the root node.
- Split the Dataset:
  - Divide the training set into subsets based on the root node's branches (according to attribute values).
- Test Remaining Attributes:
  - For each branch, test the remaining attributes to find the best fit for the next decision node (subsequent splits).
- Repeat the Process:
  - Continue this process recursively for each branch until all branches lead to a final classification (leaf nodes).

ID3 (Iterative Dichotomiser 3)

- **Inventor**: J. Ross Quinlan, 1975
- **Purpose**: Generates a decision tree from a given dataset using a **top-down, greedy** approach.
- **Method**:
  - At each node, it tests each attribute to determine the best split.
  - The algorithm selects the attribute that maximizes information gain (or minimizes entropy).
  - Outcome: The resulting tree is used to classify future data samples.

General Approach for Building a Classification Model



Training Set

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

Test Set

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

Figure 4.3. General approach for building a classification model.

Attributes				Classes
Gender	Car Ownership	Travel Cost	Income Level	Transportation
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

Algorithm for Building a Decision Tree (ID3)

- Calculate Entropy:
  - Compute the entropy of each attribute in the dataset to measure the uncertainty in class distribution.
- Split the Data:
  - Split the dataset into subsets using the attribute that results in the lowest entropy (or, equivalently, the highest information gain).
- Create a Node:
  - Create a decision tree node based on the attribute with the highest information gain.
- Recursive Process:
  - Repeat the process for each subset, using the remaining attributes until no further splits are possible.

Entropy

- Definition of Entropy:
  - Entropy measures the homogeneity or uncertainty within a dataset.
  - A completely homogeneous dataset (all samples belong to one class) has an entropy of 0.
  - A dataset that is equally divided among classes has an entropy of 1.

- Entropy Formula:

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p(i) \log_2 p(i)$$

Where:

- **S** = the dataset or sample.
- **p(i)** = the proportion of elements in class **i** within the dataset **S**.
- **n** = the total number of classes.
- $\sum_{i=1}^{n}$ = summation over all the classes in the dataset.

Entropy Example: 1

- Entropy Example 1
- Consider a set S with 14 examples:
  - 9 belong to class YES
  - 5 belong to class NO

$$E(S) = -p(+)\log_2 p(+) - p(-)\log_2 p(-)$$

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

$$\text{Entropy}(S) = 0.940$$

Entropy Example: 2

- **Disorder in Decision Making**
  - Suppose a group of friends is trying to decide which movie to watch together on Sunday.
  - There are 2 choices:
    - "Lucy" gets 4 votes
    - "Titanic" gets 5 votes.
  - Which movie do they watch now?
  - The votes are nearly equal, so it's hard to decide!

- **What Does This Mean?**
  - This situation represents disorder, where the votes are nearly split, making the decision unclear.
  - It would have been much easier if:
  - Lucy had 8 votes and Titanic had only 2.
- In a decision tree, we aim to reduce disorder by making decisions that create more certainty (e.g., more "yes" or "no" outcomes).

How Decision Trees Use Entropy

- Now that we understand what entropy is and how to calculate it, let's explore how it's applied in decision trees.
- Entropy measures the impurity or uncertainty of a node in the tree.
- Impurity reflects the level of randomness in the data at a node.
- A pure split occurs when the data is perfectly classified into one outcome (either all "yes" or all "no").

How to Build Decision Trees

Attribute Selection in Decision Trees

- The key to building a decision tree is selecting the best attribute from the list of features in the dataset for both the root and sub-nodes.
- This process is guided by a technique known as Attribute Selection Measure (ASM).
- ASM helps determine which attribute should be chosen at each node to split the data for better classification.
- There are two common techniques used in ASM:
  - **Information Gain**: Measures the reduction in entropy (uncertainty) after splitting the dataset based on an attribute.
  - Gini Index: Measures the impurity of a dataset, aiming to minimize class mix after each split.

Information Gain

- Definition:
  - Information gain measures the reduction in uncertainty (entropy) after splitting a dataset on a particular attribute. It helps decide which attribute should be selected as a decision node or root node in a decision tree.
- How it Works:
  - Information gain is calculated as the entropy of the full dataset minus the entropy of the dataset after splitting on a specific attribute.
- Formula for Information Gain:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum \left(\frac{|S_v|}{|S|} \right) \text{Entropy}(S_v)$$

Where:

- S_v = Subset of **S** for which attribute **A** has value **v**.
- $|S_v|$ = Number of elements in subset **S_v**.
- $|S|$ = Number of elements in the full dataset **S**.

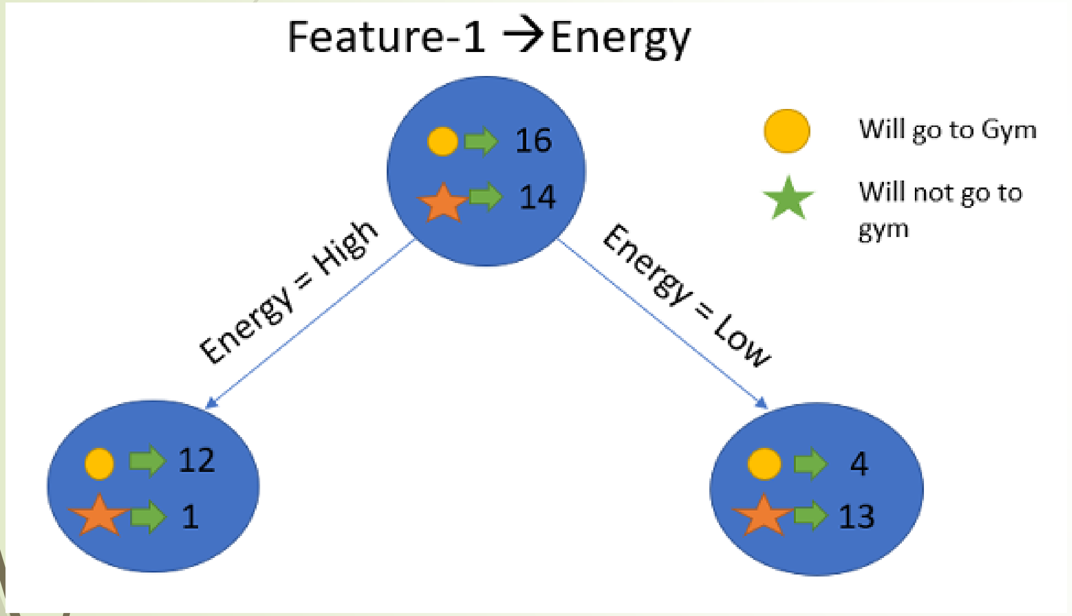
Procedure

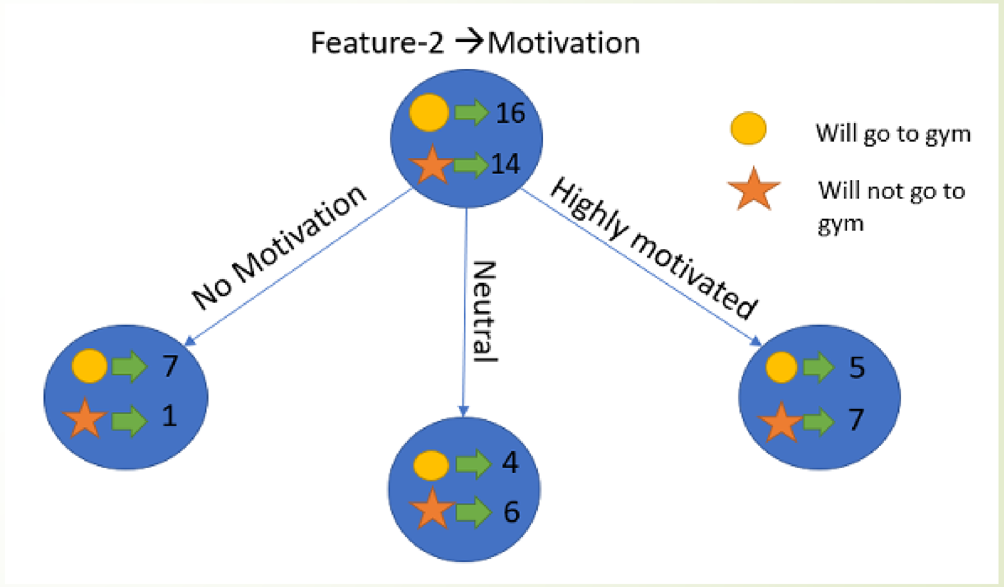
- Calculate Entropy of the Total Dataset:
  - Start by computing the overall entropy for the dataset.
- Split the Dataset on Attributes:
  - Divide the dataset based on each attribute.
- Calculate Entropy for Each Branch:
  - For each branch created by the split, calculate its individual entropy.
- Compute Total Entropy for the Split:
  - Multiply the entropy of each branch by the proportion of instances in that branch, then sum them up to get the total entropy after the split.
- Determine Information Gain:
  - Subtract the resulting entropy (after the split) from the initial entropy (before the split). This difference is the Information Gain.
- Select Attribute with Highest Information Gain:
  - The attribute that results in the largest Information Gain is chosen as the decision node.

Example - Population with 30 Instances

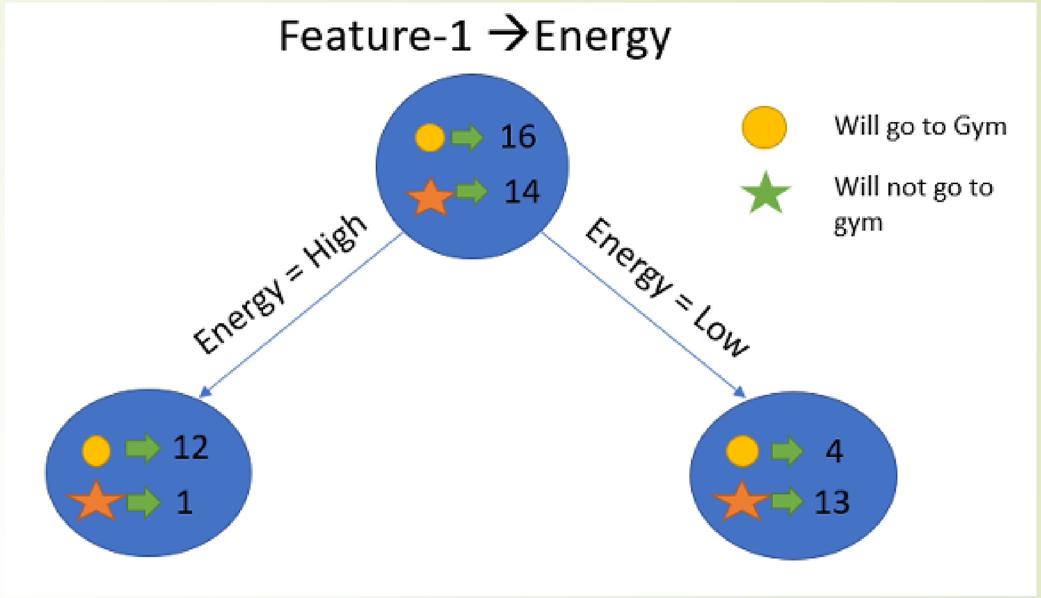
- Suppose the dataset is used to predict whether a person goes to the gym:
  - 16 people go to the gym.
  - 14 people don't go.
- We have two features to predict this:
  - Energy (high/low)
  - Motivation (No motivation/Neutral/Highly motivated)
- Parent Entropy Calculation:
- Class Distribution:
  - 16 people go to the gym (positive class).
  - 14 people do not go to the gym (negative class).

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


Feature-1 →Energy
16
14
Will go to Gym
Will not go to gym
Energy = High
Energy = Low
12
1
4
13


Feature-2 →Motivation
16
14
Will go to gym
Will not go to gym
No Motivation
Neutral
Highly motivated
7
1
4
6
5
7

Calculation for Energy:



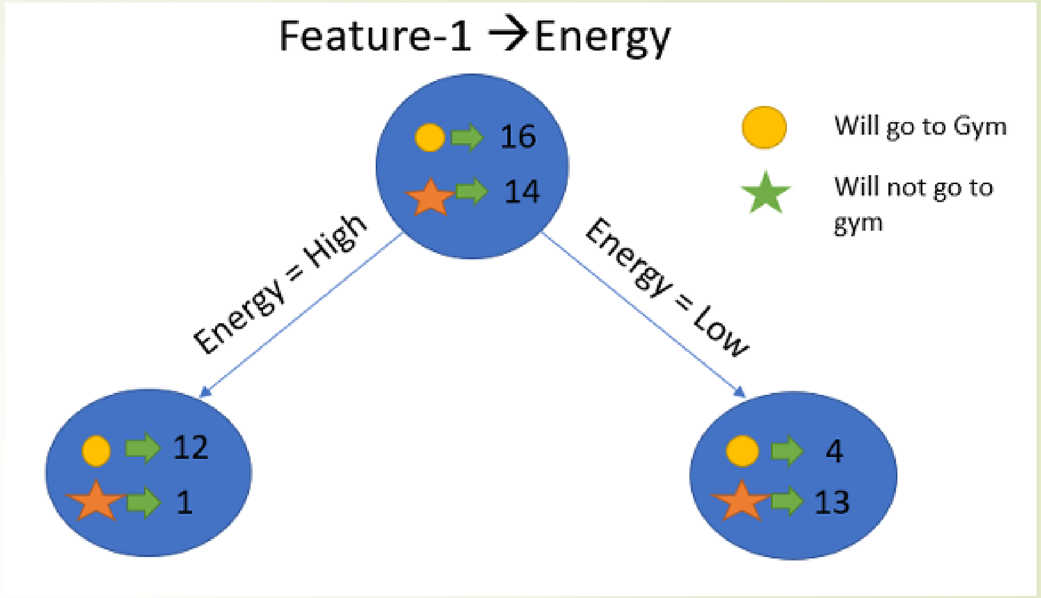
- Entropy for "Energy = High":
- Class Distribution:
  - 12 people go to the gym.
  - 1 person does not go to the gym.

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

- Entropy for "Energy = Low":
  - Class Distribution:
  - 4 people go to the gym.
  - 13 people do not go to the gym.

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

Calculation for Energy:



- Entropy for "Energy = High":

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

- Entropy for "Energy = Low":

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

- Weighted Average Entropy:

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

- Information Gain:

$$\text{Information Gain} = E(\text{Parent}) - E(\text{Parent}|\text{Energy})$$

$$\text{Information Gain} = 0.99 - 0.62 = 0.37$$

Calculation for Motivation:

- Entropy for "No Motivation":
  - Class Distribution: 7, 1

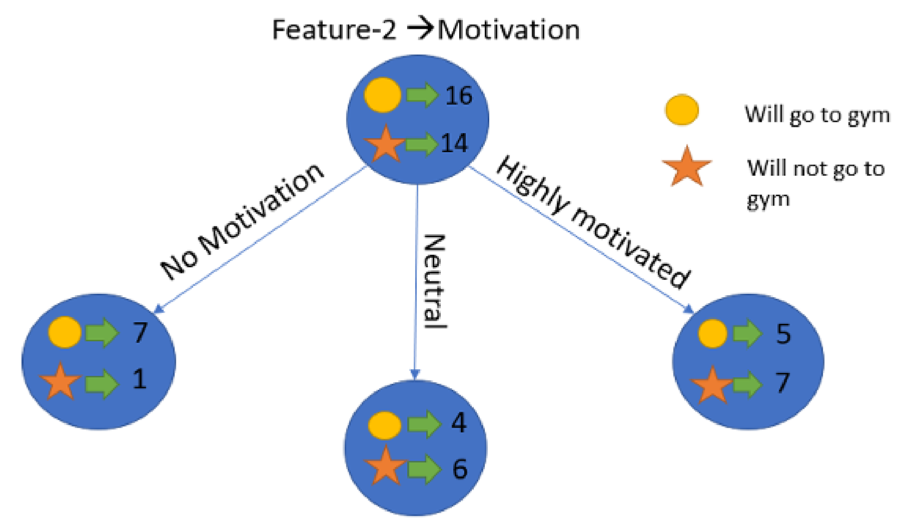
$$E(\text{Parent}|\text{Motivation} = \text{"No Motivation"}) = -\left(\frac{7}{8}\log_2\frac{7}{8}\right) - \left(\frac{1}{8}\log_2\frac{1}{8}\right) = 0.54$$

- Entropy for "Neutral":
  - Class Distribution: 4, 6

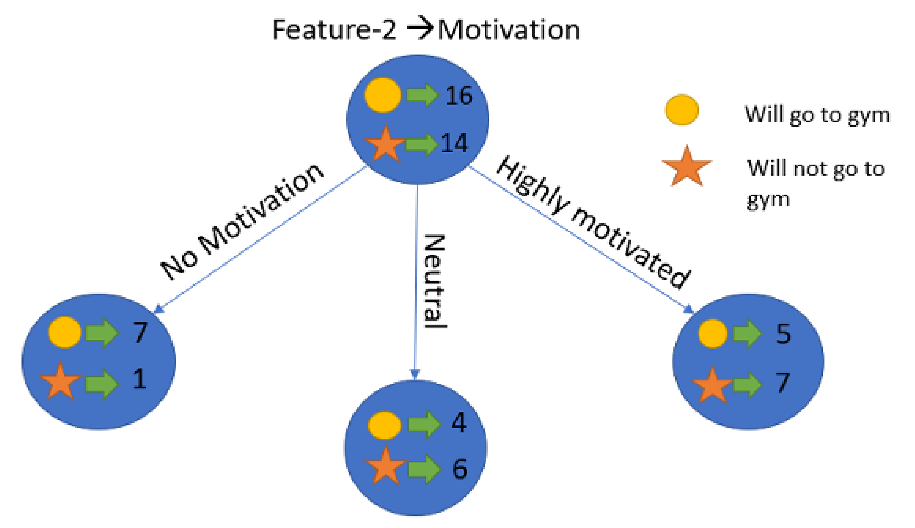
$$E(\text{Parent}|\text{Motivation} = \text{"Neutral"}) = -\left(\frac{4}{10}\log_2\frac{4}{10}\right) - \left(\frac{6}{10}\log_2\frac{6}{10}\right) = 0.97$$

- Entropy for "Highly Motivated":
  - Class Distribution: 5, 7

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



Calculation for Motivation:



- Entropy for "No Motivation":

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

- Entropy for "Neutral":

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

- Entropy for "Highly Motivated":

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

- Weighted Average Entropy:

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

- Information Gain:

$$\text{Information Gain} = E(\text{Parent}) - E(\text{Parent}|\text{Motivation}) = 0.99 - 0.86 = 0.13$$

Information Gain

- The **Energy** feature has an information gain of **0.37**, while the **Motivation** feature has an information gain of **0.13**.
- Since "**Energy**" provides a higher reduction in entropy, we will choose it as the feature for splitting at the root node.
- The feature with the **highest information gain** is used to split the node, and this process continues for sub-nodes.
- In this case, "**Energy**" is the root node, and the same method will be applied to determine the splits for sub-nodes.

Information Gain

- When the energy is "high", the entropy is low, indicating a higher likelihood that the person will go to the gym.
- However, if the energy is "low", we cannot make a clear decision yet.
- To further classify the outcome when energy is low, we will split the node again based on the next feature, which is "Motivation."

New Problem

Name	Gender	Car Ownership	Travel Cost	Income Level	Transportation
Abhi	Male	1	Standard	High	?
Pavi	Male	0	Cheap	Medium	?
Ammu	Female	1	Cheap	High	?

Attributes				Classes
Gender	**Car Ownership**	**Travel Cost**	**Income Level**	**Transportation**
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

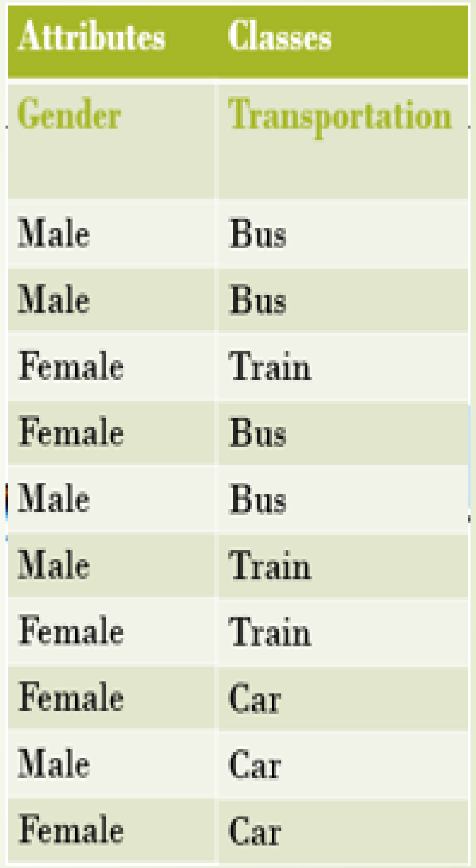
Calculate the entropy of the total dataset

- **First compute the Entropy of given training set.**

Probability
Bus : 4/10 = 0.4
Train: 3/10 = 0.3
Car:3/10 = 0.3

$$E(S) = -(0.4)\log_2(0.4) - (0.3)\log_2(0.3) - (0.3)\log_2(0.3) = 1.571$$

Attributes				Classes
Gender	Car Ownership	Travel Cost	Income Level	Transportation
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car



Attributes	Classes
Gender	Transportation
Male	Bus
Male	Bus
Female	Train
Female	Bus
Male	Bus
Male	Train
Female	Train
Female	Car
Male	Car
Female	Car

Split the dataset on ‘Gender’ attribute

Attributes	Classes
Gender	Transportatio n
Male	Bus
Male	Bus
Male	Bus
Male	Train
Male	Car

Probability
Bus : 3/5 = 0.6
Train: 1/5 = 0.2
Car:1/5 = 0.2

Attributes	Classes
Gender	Transportation
Female	Train
Female	Bus
Female	Train
Female	Car
Female	Car

Probability
Bus : 1/5 = 0.2
Train: 2/5 = 0.4
Car:2/5 = 0.4

$$Gain(S,A) = E(S) - I(S,A)$$

$$I(S,A) = 1.522 * (5/10) + 1.371 * (5/10)$$

$$Gain(S,A) = 1.571 - 1.447$$
$$= 0.12$$

$$E(S_v) = -0.6 \log_2 (0.6) - 0.2 \log_2 (0.2) - 0.2 \log_2 (0.2)$$
$$= 1.522$$

$$E(S_v) = -0.2 \log_2 (0.2) - 0.4 \log_2 (0.4) - 0.4 \log_2 (0.4)$$
$$= 1.371$$

Attributes				Classes
Gender	Car Ownership	Travel Cost	Income Level	Transportation
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

Attributes	Classes
Car Ownership	Transportation
0	Bus
1	Bus
1	Train
0	Bus
1	Bus
0	Train
1	Train
1	Car
2	Car
2	Car

Split the dataset on 'ownership' attribute

Attributes	Classes
Car Ownership	Transportation
0	Bus
0	Bus
0	Train

Probability
Bus : 2/3= 0.6
Train: 1/3 = 0.3
Car:0/3 = 0
Entropy = 0.918

Attributes	Classes
Car Ownership	Transportation
1	Bus
1	Train
1	Bus
1	Train
1	Car

Probability
Bus : 2/5 = 0.4
Train: 2/5 = 0.4
Car:1/5 = 0.2
Entropy = 1.522

Attributes	Classes
Car Ownership	Transportation
2	Car
2	Car

Probability
Bus : 0/2 = 0
Train: 0/2 = 0
Car:2/2 = 1
Entropy = 0

$$Gain(S,A) = E(S) - I(S,A)$$

$$I(S,A) = 0.918 * (3/10) + 1.522 * (5/10) + 0 * (2/10)$$

$$Gain(S,A) = 1.571 - 1.0364 = 0.534$$

Travel Cost	Transportation
Cheap	Bus
Cheap	Bus
Cheap	Train
Cheap	Bus
Cheap	Bus

Travel Cost	Transportation
Standard	Train
Standard	Train

Travel Cost	Transportation
Expensive	Car
Expensive	Car
Expensive	Car

- If we choose Travel Cost as splitting attribute,

- Entropy for Cheap = 0.722

Entropy for Cheap	=	0.722
Standard	=	0
Expensive	=	0
IG	=	1.21

- If we choose Income Level as splitting attribute,

-Entropy for Low	=	0
Medium	=	1.459
High	=	0
IG	=	0.695

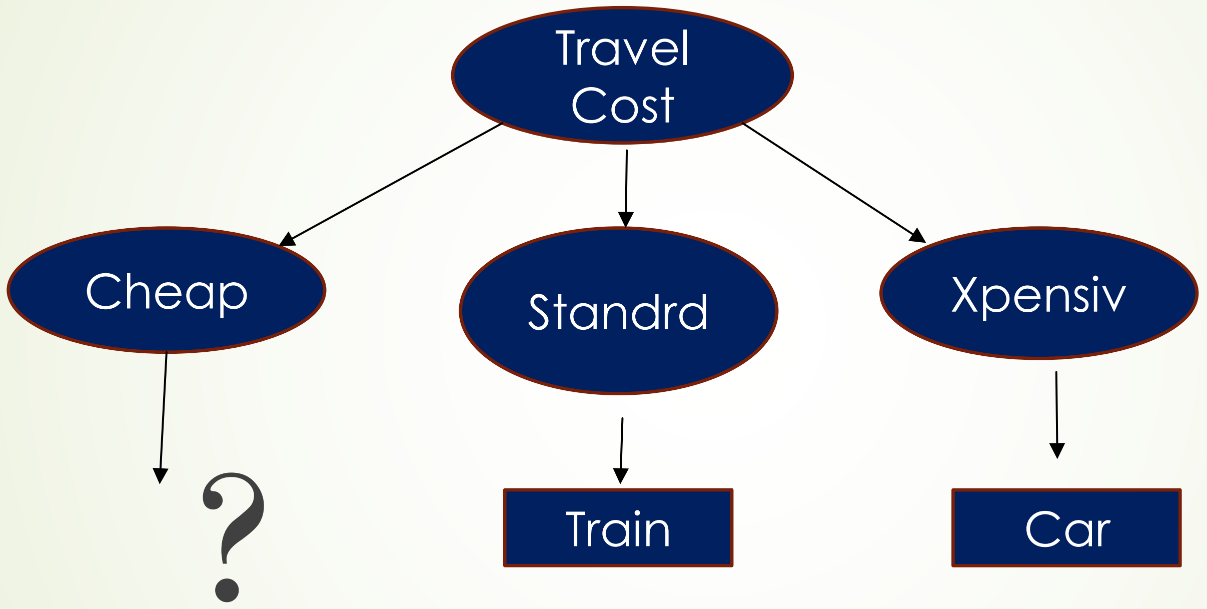
Attribute	Information Gain
Gender	0.125
Car Ownership	0.534
Travel Cost	1.21
Income Level	0.695

Attributes				Classes
Travel Cost	Gender	Car Ownership	Income Level	Transportation
Cheap	Male	0	Low	Bus
Cheap	Male	1	Medium	Bus
Cheap	Female	1	Medium	Train
Cheap	Female	0	Low	Bus
Cheap	Male	1	Medium	Bus

Attributes				Classes
Travel Cost	Gender	Car Ownership	Income Level	Transportation
Standard	Male	0	Medium	Train
Standard	Female	1	Medium	Train

Attributes				Classes
Travel Cost	Gender	Car Ownership	Income Level	Transportation
Expensive	Female	1	High	Car
Expensive	Male	2	Medium	Car
Expensive	Female	2	High	Car

Diagram : Decision Tree



Iteration on Subset of Training Set

Attributes				Classes
Travel Cost	**Gender**	**Car Ownership**	**Income Level**	**Transportation**
Cheap	Male	0	Low	Bus
Cheap	Male	1	Medium	Bus
Cheap	Female	1	Medium	Train
Cheap	Female	0	Low	Bus
Cheap	Male	1	Medium	Bus

Probability
Bus : 4/5 = 0.8
Train: 1/5 = 0.2
Car:0/5 = 0

$$E(S) = -(0.8)\log_2 (0.8) - (0.2)\log_2 (0.2) = 0.722$$

Attributes				Classes
Travel Cost	**Gender**	**Car Ownership**	**Income Level**	**Transportation**
Cheap	Male	0	Low	Bus
Cheap	Male	1	Medium	Bus
Cheap	Female	1	Medium	Train
Cheap	Female	0	Low	Bus
Cheap	Male	1	Medium	Bus

- If we choose **Gender** as splitting attribute,

-Entropy for Male = 0

Female = 1

IG = 0.322

- If we choose **Car Ownership** as splitting attribute,

-Entropy for 0 = 0

1 = 0.918

IG = 0.171

- If we choose **Income Level** as splitting attribute,

-Entropy for Low = 0

Medium = 0.918

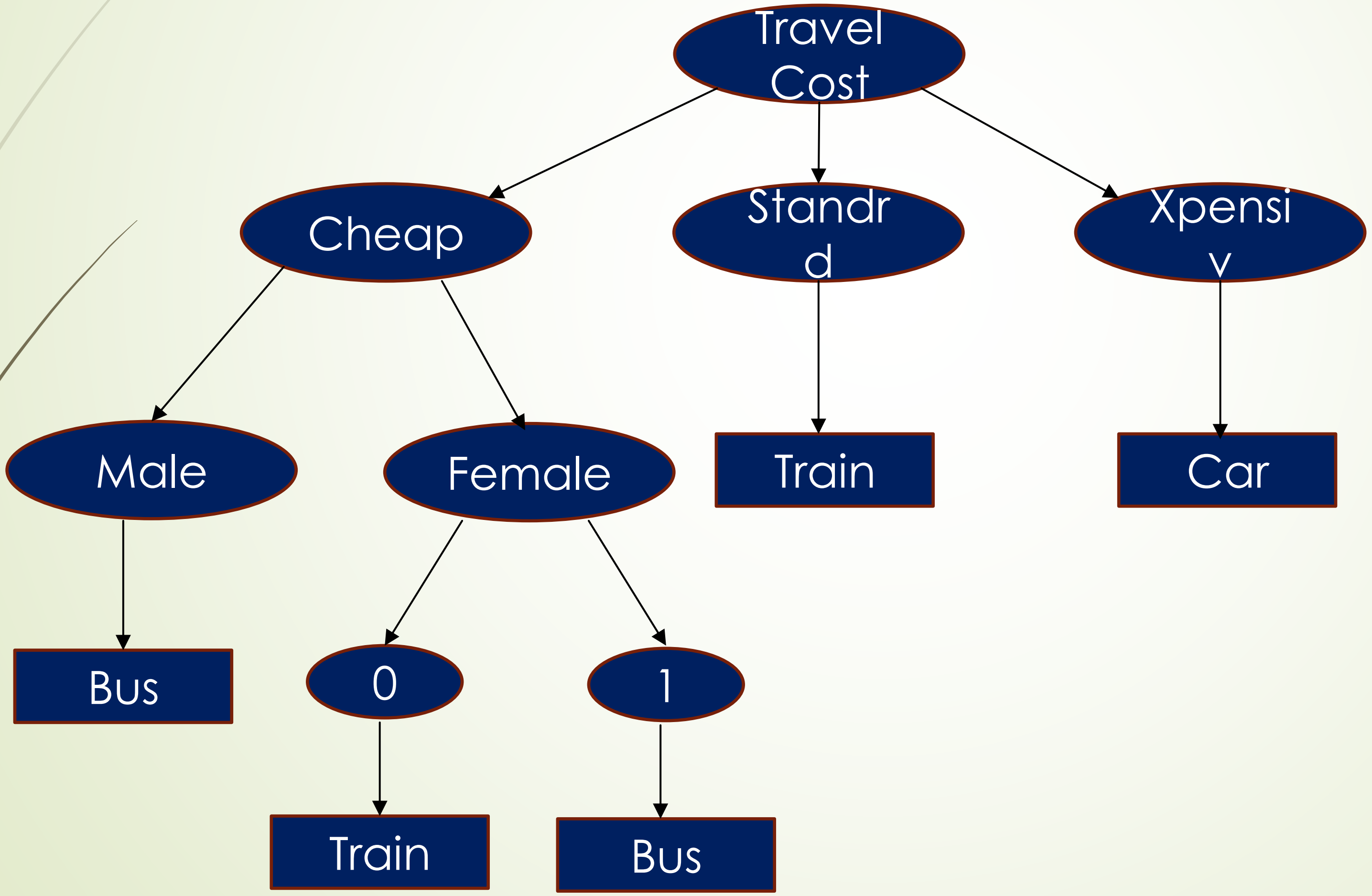
IG = 0.171

Attributes	Information Gain
Gender	0.322
Car Ownership	0.171
Income Level	0.171

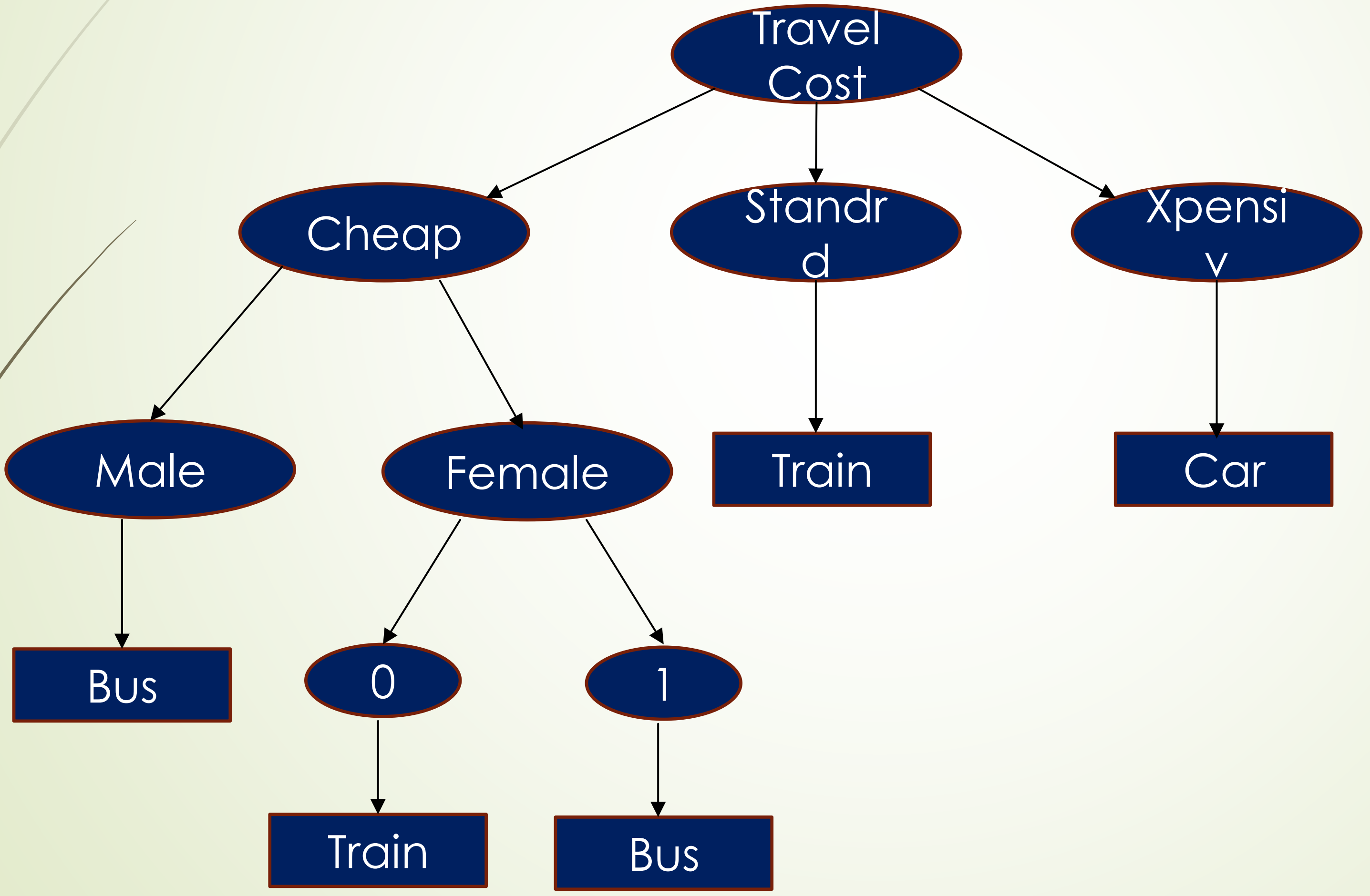
Attributes			Classes
Gender	Car Ownership	Income Level	Transportation
Male	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus

Attributes			Classes
Gender	Car Ownership	Income Level	Transportation
Female	1	Medium	Train
Female	0	Low	Bus

Diagram : Decision Tree



Name	Gender	Car Ownership	Travel Cost	Income Level	Transportatio n
Abhi	Male	1	Standard	High	?
Pavi	Male	0	Cheap	Medium	?
Ammu	Female	1	Cheap	High	?

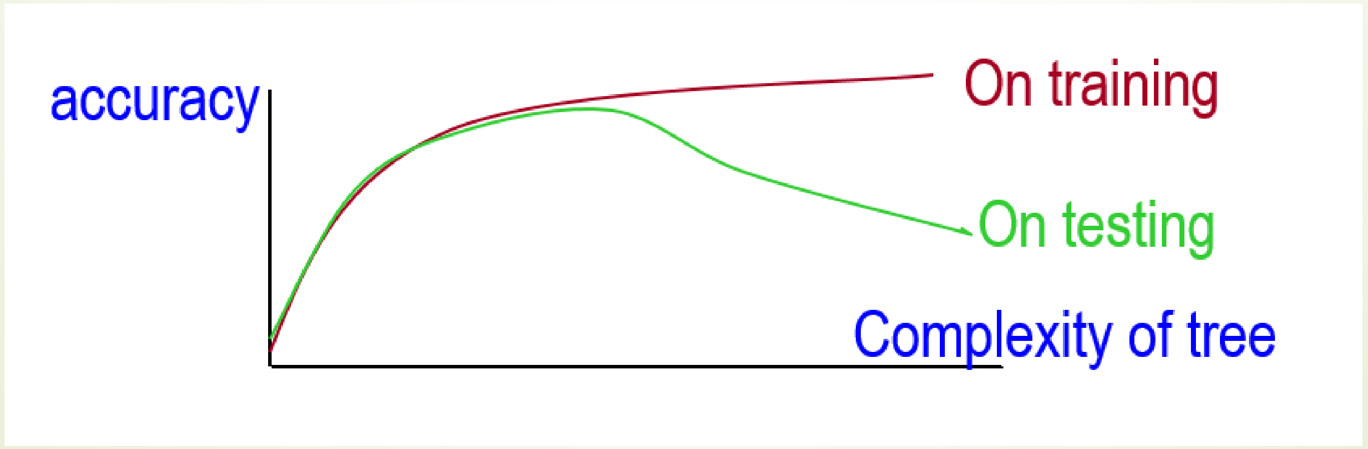


Solution to Our Problem :

Name	Gender	Car Ownership	Travel Cost	Income Level	Transportatio n
Abhi	Male	1	Standard	High	Train
Pavi	Male	0	Cheap	Medium	Bus
Ammu	Female	1	Cheap	High	Bus

Limitations of Decision Trees

- Overfitting:
  - A tree that perfectly classifies the training data may not generalize well to unseen data.
  - The tree may fit **noise** in the training data, leading to poor performance on test data.
  - The algorithm could be making decisions based on very **limited data**, resulting in overfitting.



Evaluation Methods for Decision Trees

- Two Basic Approaches:
  - Pre-pruning:
    - Stop growing the tree during construction when it's determined that there is not enough data to make reliable decisions.
  - Post-pruning:
    - First grow the entire tree, then remove nodes that lack sufficient evidence for making reliable predictions.
- Methods for Evaluating Subtrees to Prune:
  - Cross-validation:
    - Use a hold-out set to evaluate the utility of subtrees and determine if pruning will improve performance.
  - Statistical Testing:
    - Test whether the observed patterns are statistically significant or likely to have occurred by chance.
  - Minimum Description Length (MDL):
    - Compare the complexity of the tree (hypothesis) with the simplicity of remembering exceptions. If the tree is overly complex for the data it explains, prune it.

Gini Index

Gini Index

- **Definition**: A measure of impurity or purity used to evaluate splits in decision trees.
- **Goal**: Lower Gini index values indicate a purer node, meaning the node is dominated by samples from a single class.
- **Formula**:

$$Gini\ Index = 1 - \sum_j p_j^2$$

- where p_j is the probability of a sample belonging to class j.
- The Gini Index ranges between **0** (perfect purity) and **1** (maximum impurity).

Gini – Example

- Initial Dataset Distribution (Before Splitting):
  - There are 14 samples.
  - 9 belong to the "Play" class (Yes).
  - 5 belong to the "Not Play" class (No).
- We calculate the **Gini Index** of this parent node:

$$Gini_{parent} = 1 - (p_{Play}^2 + p_{Not\ Play}^2)$$

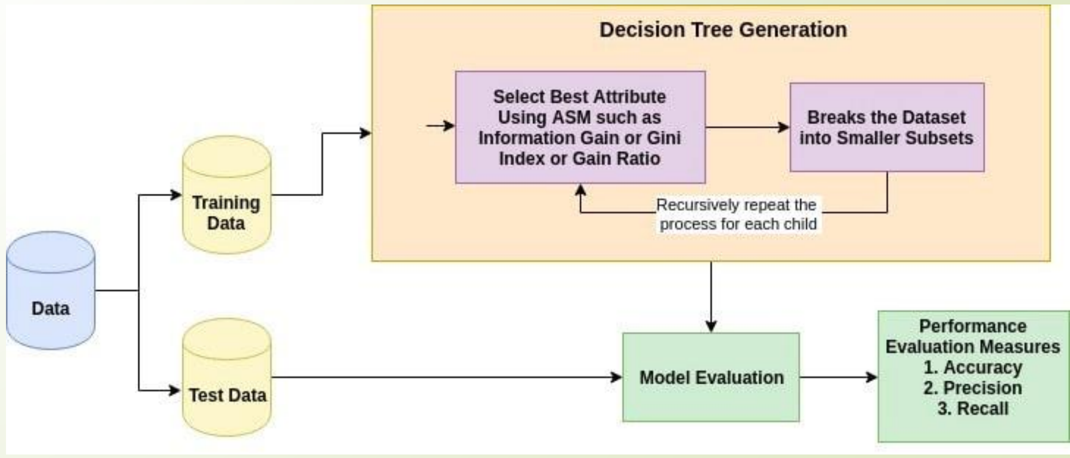
$$Gini_{parent} = 1 - \left(\left(\frac{9}{14}\right)^2 + \left(\frac{5}{14}\right)^2\right)$$

$$Gini_{parent} = 1 - (0.408 + 0.128) = 1 - 0.536 = 0.464$$

When to Stop Splitting

Tree Creation

- Select the Best Feature:
  - Use Attribute Selection Measures (ASM), such as Information Gain or Gini Index, to identify the most informative feature for splitting the records.
- Create a Decision Node:
  - The selected feature becomes a decision node, splitting the dataset into smaller subsets.
  - Repeat this process recursively until stopping criteria are met.


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

When to Stop Splitting

- Why Stop Splitting?
  - Splitting too much leads to overfitting, where the tree performs well on training data but poorly on unseen test data.
- Real-World Challenges:
  - Large datasets with many features often lead to large trees with many splits.
  - Building such trees takes time and can result in overfitting if no stopping criteria are applied.
- Overfitting Risk:
  - A perfectly trained tree may give high accuracy on the training data but low accuracy on test data (poor generalization).

When to Stop Splitting

- max_depth:
  - Controls the maximum depth of the tree. A higher value results in a more complex tree.
  - Increasing the depth reduces training error, but can cause overfitting, resulting in poor accuracy on test data.
- min_samples_split:
  - Defines the minimum number of data points (samples) required to split an internal node.
  - Helps prevent overfitting by limiting splits on nodes with few samples.
- min_samples_leaf:
  - Represents the minimum number of samples required in a leaf node.
  - Larger values reduce overfitting but might lead to underfitting if set too high.
- max_features:
  - Specifies the maximum number of features to consider when splitting a node.
  - Helps control complexity and prevents overfitting by limiting the number of features evaluated.

max_features:

- Definition:
  - max_features determines the maximum number of features the decision tree will evaluate when searching for the best possible split at each node.
  - Instead of considering all available features, the algorithm randomly selects a subset of features, and the best split is chosen only from that subset.
- Purpose:
  - By limiting the number of features considered at each split, max_features helps to reduce the complexity of the model and lowers the chance of overfitting.
  - It is particularly useful in ensemble methods like Random Forests, where individual decision trees are built by selecting random subsets of features, which increases model diversity and reduces correlation among the trees.
- Effect:
  - Lower values oconsider fewer features at each split, resulting in simpler trees and reduced risk of overfitting.
  - Higher values (or setting it to the total number of features) allow the tree to consider all available features for each split, which may result in overfitting if the dataset has many irrelevant features.
  - f max_features (e.g., square root or log of the number of total features.

Pruning

Pruning in Decision Trees

- Pre-pruning (Early Stopping):
  - The tree stops growing early by avoiding splits that are not statistically significant.
  - Nodes can be pruned during tree construction if the split doesn't improve the model much, preventing overfitting while growing the tree.

- Post-pruning (Tree Pruning):
  - The tree is first allowed to grow to its maximum depth.
  - Afterward, pruning is applied to remove nodes that add little to no predictive value, simplifying the tree and improving generalization.

Advantages of Decision Trees

- Easy to Understand:
  - The decision tree’s output is intuitive and easy to understand, even for individuals with non-analytical backgrounds.
  - It doesn't require deep statistical knowledge to interpret the results.
  - The graphical representation of a decision tree makes it easy to visualize relationships and verify hypotheses.
- Useful for Data Exploration:
  - Decision trees help quickly identify the most significant variables and understand relationships between them.
  - They can assist in creating new features that have stronger predictive power for the target variable.
- Minimal Data Cleaning:
  - Decision trees require less data preprocessing compared to many other models.

Disadvantages of Decision Trees

- Overfitting:
  - Overfitting is a common issue with decision trees, especially when the tree is allowed to grow too deep, capturing noise in the training data.
  - This problem can be mitigated by applying constraints such as limiting the tree's depth or pruning to remove branches that have little predictive power.
- Not Ideal for Continuous Variables:
  - Decision trees can struggle with continuous numerical variables, as they split them into discrete intervals, potentially leading to a loss of information.
  - This can make decision trees less effective compared to algorithms that handle continuous variables more smoothly (e.g., linear regression).

Random Forest

Introduction to Random Forest

- What is Random Forest?
  - Random Forest is a versatile machine learning method capable of handling both regression and classification tasks.
  - It is an ensemble learning method that combines multiple decision trees (often referred to as "weak learners") to create a stronger model.
  - Random forests are highly effective at handling issues such as dimensionality reduction, missing values, and outliers.
- Key Features:
  - Performs well on complex datasets with a large number of features and data points.
  - Robust to overfitting due to the averaging of multiple decision trees.
  - Works well even when the data has missing or noisy values.

How Random Forest Works

- How Random Forests Work:
  - A Random Forest consists of many decision trees. Each tree in the forest is trained on a random subset of the data (using bootstrap sampling) and a random subset of features at each split.
- Decision Trees vs. Random Forest:
  - Decision Trees: Make predictions by splitting the data on individual features in sequence, often prone to overfitting.
  - Random Forest: Combines the predictions of multiple decision trees. Each tree votes on the final prediction, and the most frequent vote (for classification) or the average prediction (for regression) is taken as the output.
- Why Use Random Forest?:
  - Reduces overfitting: Since individual trees can overfit to the training data, combining multiple trees leads to a more generalized model.
  - Voting Mechanism: For classification tasks, Random Forest uses a majority voting approach, where each tree contributes one vote for the final prediction.

Random Forest: Out-of-Bag (OOB) Error

- Bootstrap Sampling:
  - Random Forest uses bootstrap sampling to train each tree. It involves sampling the input data with replacement, meaning that some samples are selected multiple times, while others are left out.
- Out-of-Bag Samples:
  - About one-third of the data is not used in training for each tree. These unused samples are referred to as out-of-bag (OOB) samples.
- Out-of-Bag Error:
  - The OOB samples are used to estimate the model's performance. The error calculated on these samples is called the out-of-bag error.
- Reliable Estimate:
  - Research shows that OOB error is a reliable indicator of model performance, often comparable to the error estimate from a separate test set.

Advantages of Random Forest

- Versatile for Classification and Regression
  - Random Forest is effective for both classification and regression tasks.
  - It consistently delivers strong performance across different types of problems.
- Handling High-Dimensional Data
  - Capable of managing large datasets with many features.
  - Random Forest can identify the most important features, making it useful for dimensionality reduction.
- Handles Class Imbalances
  - Incorporates methods to balance errors in datasets where the target classes are imbalanced, improving model fairness and performance.

References

- https://realpython.com/logistic-regression-python/
- https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/