

**Machine Learning Lecture Series**

# **Machine Learning**

**CSE21816**

**UNIT: 02**

**Lecture: 02**

## **Course Instructor**



**Dr. Tamal Ghosh**

Associate Professor

Computer Science and Engineering, Adamas University

[tamal.ghosh1@adamasuniversity.ac.in](mailto:tamal.ghosh1@adamasuniversity.ac.in)

# Machine Learning –An Introduction

- The aim of machine learning is, making the computers to learn automatically by itself.
- A computer programming that can access data, and sometimes observe data (external experience) and use it for learning itself (self experience) to take decision is called as machine learning.
- There is no fixed definition for machine learning, and sometimes we get same definition for both Artificial Intelligence and Machine Learning.

# Data in Vector Space Representation

- *Machine Learning is the science of getting computers to learn and act like humans do, and*
  - *improve their learning over time in autonomous fashion,*
  - *by feeding them data and information in the form of*
  - *observations and real-world interactions.*

# Types of Machine Learning

- Supervised learning
- Semi supervised learning
- Unsupervised learning
- Reinforcement learning

# Supervised Learning

- There are two phases in supervised learning,
- **Phase-1 – Model Construction** : first train the machine with sample data which are **labelled** and error free data. i.e. some data are already tagged with correct data.
- The model is represented as classification rules, decision trees, or mathematical formulae
- **Phase-2 – Model Usage:** This phase will estimate the accuracy of model.
- The new set of data (Testing data) will be given to machine which are example data, then the machine analyze the training data and produces the output.

# Semi Supervised Learning

- This semi supervised learning will fall **between supervised and unsupervised learning**, and it also called as weak supervision.
- This learning method uses small amount of labeled data and large amount of unlabeled data.
- In this model, the learning (training) will taken place without having much of labeled training data.

# Unsupervised Learning

- This model doesn't require labeled data.
- This model learns to identify the patterns and trends or categorize data without labeling it.
- There is more volume of data available for unsupervised learning, because most data are unlabeled.

# Reinforcement Learning

- This model is not like the previous models.
- In reinforcement learning, the reward values are attached to different steps that the model is supposed to go through.
- Hence, the aim here is to accumulate more reward points possible and eventually get to an end goal.
- E.g. Video game.
- To enter into next step, that will earn a reward and taken to next level.

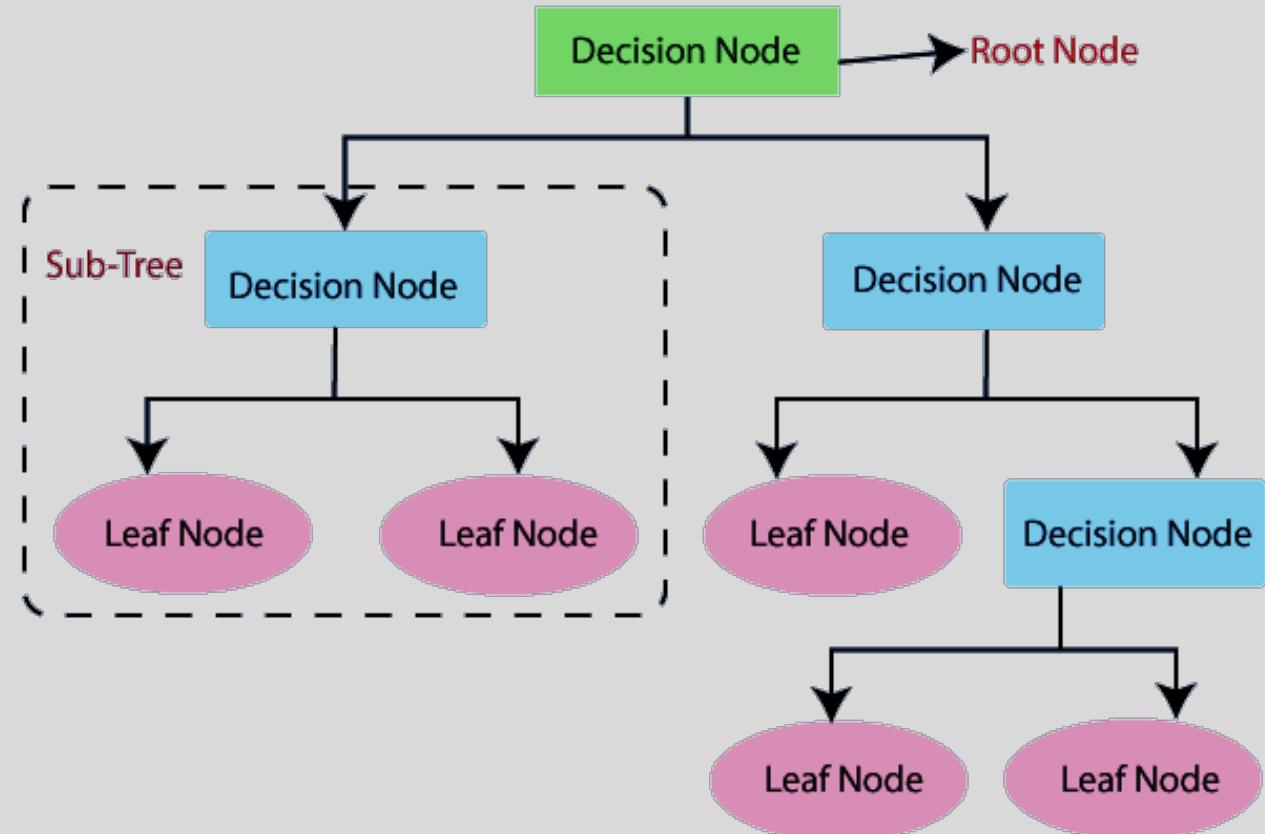
# **DECISION TREE**

# Decision Tree: Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.**
- The decision tree has two node types, which are the **Decision Node** and **the Leaf Node**. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed based on features of the given dataset.
- *It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.*

# Decision Tree: Classification Algorithm

- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- To build a tree, we use the **CART algorithm**, which stands for **Classification and Regression Tree algorithm**.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.
- The diagram at right explains the general structure of a decision tree:



# Why to use Decision Trees?

- There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:
- **Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.**
- **The logic behind the decision tree can be easily understood because it shows a tree-like structure.**

# Decision Tree Terminologies

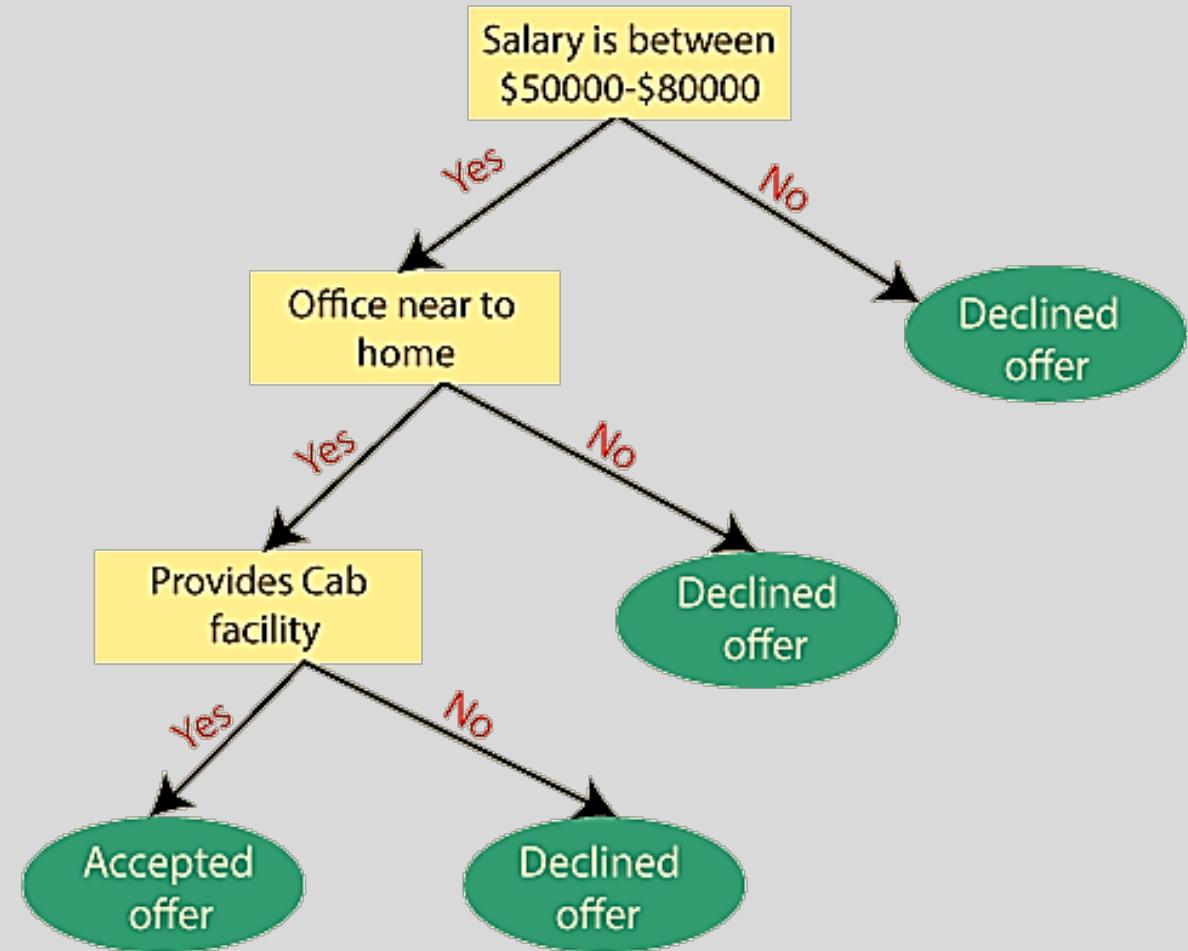
- **Root Node:** The root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

# How does the Decision Tree algorithm Work?

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:
- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM)**.
- **Step-3:** Divide the S into subsets that contain possible values for the best attributes.

# Decision Tree Workflow

- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node a leaf node.
- **Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further ..



# Attribute Selection Measures

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM**. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:
  - **Information Gain**
  - **Gini Index**

# Information Gain

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

$$\text{Information Gain} = \text{Entropy}(S) - (\text{Weighted Avg}) * \text{Entropy}(\text{each feature})$$

- **Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\text{Entropy} = -P(\text{yes}) \log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$$

Where,

$P(\text{yes})$ = probability of yes,  $P(\text{no})$ = probability of no

# Gini Index

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

- There are many algorithms for building a decision tree. They are
  1. **CART** (Classification and Regression Trees) — This makes use of Gini impurity as the metric.
  2. **ID3** (Iterative Dichotomiser 3) — This uses entropy and information gain as metric.

# Pruning: Getting an Optimal Decision tree

- *Pruning is a process of deleting unnecessary nodes from a tree 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. There are mainly two types of tree **pruning** technology used:
  - **Pre-Pruning**
  - **Post-Pruning**

# Pros and Cons of Using Decision Tree

## Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

## Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm**.
- For more class labels, the computational complexity of the decision tree may increase.

# Example

- Consider the dataset to decide whether to play football or not:

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

# Example...

- Here we have 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.
- As the first step, we have to find the parent node for our decision tree. For that follow the steps:
- ***Find the entropy of the class variable.***
- $E(S) = -[(9/14)\log(9/14) + (5/14)\log(5/14)] = 0.94$
- note: Here typically we will take log to base 2. Here total there are 14 yes/no. Out of which 9 yes and 5 no. Based on it we calculated the probability above.
- From the above data for Outlook we can arrive at the following table easily

# Example...

		play			
		yes	no	total	
Outlook	sunny	3	2	5	
	overcast	4	0	4	
	rainy	2	3	5	
				14	

- Now we have to calculate average weighted entropy. ie, we have found the total of weights of each feature multiplied by probabilities.
- $E(S, \text{outlook}) = (5/14)*E(3,2) + (4/14)*E(4,0) + (5/14)*E(2,3) = (5/14)(-(3/5)\log(3/5)-(2/5)\log(2/5)) + (4/14)(0) + (5/14)((2/5)\log(2/5)-(3/5)\log(3/5)) = 0.693$
- The next step is to find the information gain. It is the difference between parent entropy and average weighted entropy we found above.
- $IG(S, \text{outlook}) = 0.94 - 0.693 = 0.247$

# Example...

- Similarly find Information gain for Temperature, Humidity, and Windy.
- $IG(S, \text{Temperature}) = 0.940 - 0.911 = 0.029$
- $IG(S, \text{Humidity}) = 0.940 - 0.788 = 0.152$
- $IG(S, \text{Windy}) = 0.940 - 0.8932 = 0.048$
- ***Now select the feature having the largest information gain.*** Here it is Outlook. So it forms the first node(root node) of our decision tree.
- Now our data look as follows

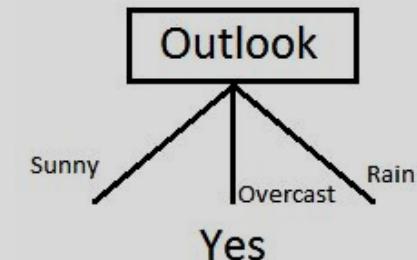
Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

# Example...

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

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

- Since overcast contains only examples of class ‘Yes’ we can set it as yes. That means If the outlook is overcast football will be played. Now our decision tree looks as follows.



# Example...

- The next step is to find the next node in our decision tree. Now we will find one under sunny. We have to determine which of the following Temperature, Humidity or Wind has higher information gain.

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

- Calculate parent entropy  $E(\text{sunny})$
- $E(\text{sunny}) = -(3/5)\log(3/5)-(2/5)\log(2/5)) = 0.971.$
- Now Calculate the information gain of Temperature.  $IG(\text{sunny}, \text{Temperature})$

		play		total
		yes	no	
Temperature	hot	0	2	2
	cool	1	1	2
	mild	1	0	1
				5

$$E(\text{sunny}, \text{Temperature}) = (2/5)*E(0,2) + (2/5)*E(1,1) + (1/5)*E(1,0)=2/5=0.4$$

Now calculate information gain.

$$IG(\text{sunny}, \text{Temperature}) = 0.971 - 0.4 = 0.571$$

Similarly we get

$$IG(\text{sunny}, \text{Humidity}) = 0.971$$

$$IG(\text{sunny}, \text{Windy}) = 0.020$$

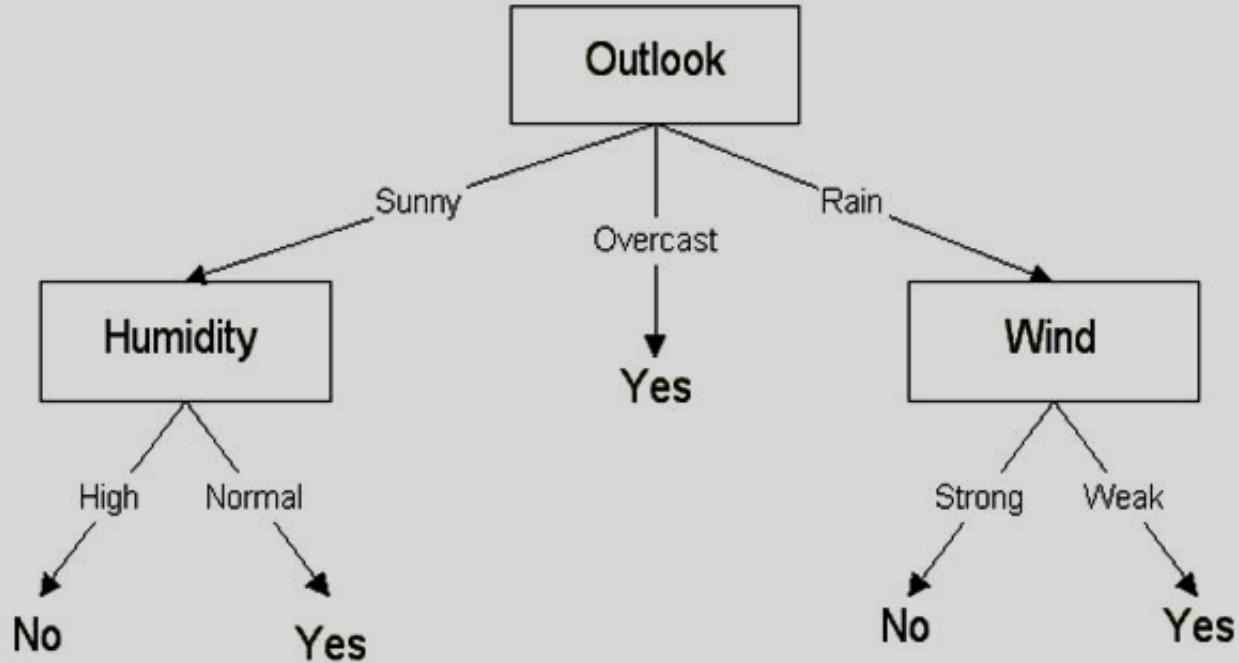
Here  $IG(\text{sunny}, \text{Humidity})$  is the largest value.

So Humidity is the node that comes under sunny.

		play	
		yes	no
Humidity	high	0	3
	normal	2	0

# Example...

- For humidity from the above table, we can say that play will occur if humidity is normal and will not occur if it is high. Similarly, find the nodes under rainy.
- ***Note: A branch with entropy more than 0 needs further splitting.***
- Finally, our decision tree will look like:



The use of the CART algorithm follows the very same procedure, except it uses Gini impurity rather than entropy.